UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
PROGRAMA DE PÓS-GRADUAÇÃO EM ESTATÍSTICA

LUCIENE MARIA TORQUATO CERQUEIRA BATISTA

AMOSTRAGEM E ESTIMAÇÃO DE MÉDIAS DE PREÇOS INCORPORANDO INFORMAÇÕES DE *BIG DATA*

Recife
2025

LUCIENE MARIA TORQUATO CERQUEIRA BATISTA

AMOSTRAGEM E ESTIMAÇÃO DE MÉDIAS DE PREÇOS INCORPORANDO INFORMAÇÕES DE *BIG DATA*

Tese apresentada ao Programa de Pós-graduação em Estatística do Centro de Ciências Exatas e da Natureza da Universidade Federal de Pernambuco, como requisito parcial à obtenção do título de doutor(a) em Estatística.

Área de concentração: Amostragem

Orientador: Dr. Cristiano Ferraz

Coorientador: Dr. Raydonal Ospina Martinez

Recife
2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Batista, Luciene Maria Torquato Cerqueira.
Amostragem e estimação de médias de preços incorporando informações de Big data / Luciene Maria Torquato Cerqueira Batista. - Recife, 2025.
109f.: il.

Tese (Doutorado) - Universidade Federal de Pernambuco, Centro de Ciências Exatas e da Natureza, Programa de Pós-Graduação em Estatística, 2025.
Orientação: Dr. Cristiano Ferraz.
Coorientação: Dr. Raydonal Ospina Martinez.
Inclui referências e apêndices.

1. Mega dados; 2. Web Scraping; 3. Cadastros Múltiplos; 4. Amostragem de Bernoulli; 5. Amostragem por Captura e Recaptura. I. Ferraz, Dr. Cristiano. II. Martinez, Dr. Raydonal Ospina. III. Título.

UFPE-Biblioteca Central

LUCIENE MARIA TORQUATO CERQUEIRA BATISTA

AMOSTRAGEM E ESTIMAÇÃO DE MÉDIAS DE PREÇOS INCORPORANDO INFORMAÇÕES DE *BIG DATA*

Tese apresentada ao Programa de Pós-graduação em Estatística do Centro de Ciências Exatas e da Natureza da Universidade Federal de Pernambuco, como requisito parcial à obtenção do título de doutor(a) em Estatística.

Aprovada em___/___/_____.

BANCA EXAMINADORA

_____

Prof. Dr. Cristiano Ferraz
UFPE - Universidade Federal de Pernambuco
(Orientador)

_____

Prof.ª Dr.ª Lucia Pereira Barroso
IME-USP
(Examinador externo)

_____

Prof. Dr. Fernando Antônio da Silva Moura
UFRJ - Universidade Federal do Rio de Janeiro
(Examinador externo)

_____

Prof. Dr. Pedro Luis Do Nascimento Silva
ENCE - Escola Nacional de Ciências Estatísticas
(Examinador externo)

Prof.ª Dr.ª Andrea Diniz
ENCE - Escola Nacional de Ciências Estatísticas
(Examinador externo)

Prof. Dr. André Leite Wanderey
(Suplente externo)

Prof. Dr. Alex Dias Ramos
(Suplente interno)

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por me conceder forças para concluir esta pesquisa, mesmo diante das inúmeras dificuldades e desafios que a vida me impôs. Sem Sua presença constante, nada disso seria possível.

Sou eternamente grata à minha mãe (*in memoriam*), que dedicou sua vida para que eu pudesse alcançar meus sonhos. Se hoje sou uma mulher forte, determinada e resiliente, é graças ao exemplo que ela me deixou.

Agradeço profundamente ao meu irmão, meu presente terreno, que sempre foi e será meu apoio incondicional. Seu amor me trouxe certezas em momentos de dúvidas, e sei que, com ele, nunca estarei sozinha.

Minha eterna gratidão à minha companheira de vida, Renata, por estar ao meu lado em todos os momentos, seja nos dias difíceis ou nas conquistas. Sua fé em mim, seu apoio constante e seu amor incondicional foram os pilares que me mantiveram de pé.

Aos meus orientadores, Prof. Cristiano e Prof. Raydonal, sou imensamente grata por não desistirem de mim. Suas orientações, paciência e apoio foram essenciais para minha jornada acadêmica. Prof. Cristiano, sua frase “não desista” ecoou em minha mente nos momentos mais difíceis e, graças a ela, cheguei até aqui. Agradeço por me acolher com tanto carinho e por acreditar em meu potencial, mesmo quando eu mesma duvidava.

Ao NIC-br, especialmente a Marcelo Pitta, agradeço pelo reconhecimento do meu trabalho e pela oportunidade de aprender com os melhores.

Agradeço também ao Hub Regional das Nações Unidas para *Big data* no Brasil, pela valiosa troca de conhecimento ao longo desses anos, que agregou tanto ao meu desenvolvimento pessoal e acadêmico.

Por fim, sou imensamente grata a todos os meus amigos, colegas e familiares, que, de alguma forma, contribuíram para minha jornada. Cada um de vocês foi uma peça fundamental na construção do que sou hoje. Com todos vocês, continuo aprendendo, evoluindo e me transformando, buscando me tornar a melhor versão de mim mesma.

RESUMO

Esta tese apresenta uma proposta de método de amostragem para coleta de dados, com o objetivo de gerar estimativas para médias de preços, levando em consideração informações disponíveis na web, relativas a uma população de "lojas virtuais", através de algoritmos de *web scraping*, a fim de contribuir para construções de índices de preços voltados para esse tipo de comércio. Ainda que a capacidade computacional e tecnológica esteja em constante avanço, definir uma estratégia de amostragem probabilística para captura de dados de um universo *Big data*, como a web (W3), é de fundamental importância para a realização de inferência estatística adequada. Na composição de tal estratégia, esta tese propõe o uso de técnicas de amostragem em dois estágios, múltiplos cadastros, combinação de captura e recaptura com amostragem de Bernoulli e amostragem bidimensional. Parte da teoria proposta é ilustrada através de um experimento piloto para gerar estimativas de médias de preços de lojas virtuais para aparelhos celulares.

Palavras-chave: Mega dados; *web scraping*; cadastros múltiplos; amostragem de Bernoulli; amostragem por captura e recaptura.

ABSTRACT

This thesis presents a proposed sampling method for data collection, aimed at generating estimates for average prices, considering information available on the web related to a population of “online stores”, through web scraping algorithms, to contribute to the development of price indices focused on this type of commerce. Even though computational and technological capacity is constantly advancing, defining a probabilistic sampling strategy to capture data from a *Big data* universe, such as the web (W3), is of fundamental importance for carrying out adequate statistical inference. In composing such a strategy, this thesis proposes the use of two-stage sampling techniques, multiple frames, combination of capture and recapture with Bernoulli sampling and bidimensional sampling. Part of the proposed theory is illustrated through a pilot experiment aimed at generating average price estimates for online stores selling mobile phones.

Keywords: Big data; web scraping; multiple frames; Bernoulli sampling; capture-recapture sampling.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O comércio viabilizado pela internet é extenso e composto por diferentes tipos de transações, denominadas na literatura como comércio digital, também conhecido como "E-commerce", que inclui, tipicamente, vendas de produtos e serviços, praticadas pelo que chamaremos, nesta tese, de Lojas Virtuais (LV).

As lojas virtuais têm sido fundamentais para a economia mundial, uma vez que seu processo de venda, muitas vezes, é facilitado por uma estrutura em que seus dados são expostos a ponto de alcançar uma população de consumidores que não se limita somente a um determinado local físico, mas perpassa fronteiras nacionais. Existem diferentes tipos de lojas virtuais: aquelas que executam todo o processo de venda somente por meio da web e aquelas que executam suas vendas tanto por meio da web quanto de estabelecimentos físicos.

Por estar presente na internet, grande parte das informações relacionadas à população de "lojas virtuais" pertence ao que a literatura denomina de *Big Data*, termo que se refere a conjuntos massivos de dados gerados a partir de conexões e usos de diversos segmentos do mundo digital e eletrônico, como a internet, que, por meio de dispositivos como celulares e computadores, viabiliza serviços como comércio eletrônico, sistemas de nuvem, redes sociais e até mesmo plataformas digitais na área de transporte. O conceito de *Big Data* é definido pelos três V's: volume, que se refere à enorme quantidade de dados gerados continuamente; variedade, que representa a diversidade de formatos e fontes dessas informações; e velocidade, que diz respeito à rapidez com que os dados são produzidos, coletados e processados. Além disso, esses dados podem ser provenientes de dispositivos equipados com tecnologia para coleta e transmissão de informações, viabilizada pelo conceito conhecido como "Internet das Coisas". Esses dispositivos incluem relógios inteligentes, geladeiras digitais, sistemas de GPS, sensores, radares, câmeras, satélites, entre outros. Nesse contexto, essa grande fonte de dados possibilita a realização de estudos sobre comportamentos em diversos segmentos, abrangendo indivíduos, governos, fenômenos naturais e, especificamente, lojas virtuais, que são o foco principal desta pesquisa de tese.

Até pouco tempo, os dados do universo *Big data* eram grandes, voláteis, complexos e desestruturados demais para serem manuseados, exigindo muito esforço para extrair informações úteis, ou identificar uma população dentro desse universo. Contudo, atualmente, têm sido criados e desenvolvidos métodos e tecnologias eficientes e inovadoras capazes de ajudar a enfrentar diversos desafios estatísticos e computacionais, no que diz respeito à coleta, ao processamento, ao armazenamento, ao tratamento e à análise desses dados, uma vez que

inferir sobre parâmetros relacionados a esse universo requer métodos capazes de reunir e filtrar dados que possuem: Valor (filtrando somente dados úteis e pertinentes); Verdade (utilizando dados fidedignos à realidade da população-alvo) e Visualização (organizando e armazenando grandes volumes de dados, de maneira que seja possível utilizá-los e apresentá-los de forma clara e objetiva), para, dessa forma, ser possível a apresentação de estimativas que darão suporte a tomadas de decisões com base em evidências.

Uma das principais técnicas computacionais criadas para lidar com esse tipo de dados é denominada de *Web scraping*. Trata-se da extração automática de informações da web, compilando-as em conjuntos de dados utilizáveis e de fácil análise. Essa técnica pode ser implementada por meio de algoritmos projetados para extrair informações de protocolos HTTP (*Hypertext Transfer Protocol*) e códigos HTML (*HyperText Markup Language*), seja de páginas específicas, seja de plataformas que agregam dados de diversas fontes simultaneamente, como o Google.

Uma coleta de dados através de *Web scraping*, embora seja necessário especificar quais páginas terão seus dados coletados e construir uma programação personalizada para cada uma delas, é um processo que pode ser replicado, conseguindo reunir uma quantidade significativa de uma lista de informações desejadas, de maneira célere e com um custo inferior ao da coleta presencial em estabelecimentos. Além disso, é possível que nesses algoritmos sejam utilizados códigos que atuam como filtros que garimpam os dados de valor para uma pesquisa. Existem diversas fontes que podem ser consultadas para saber mais sobre *Web scraping*, incluindo: YI *et al*., 2003; Butler, 2008; Zhao, 2017; e Vargiu; Urru, 2013.

As características da técnica de *web scraping* a tornam um potencial instrumento, inclusive para agências nacionais produtoras de estatísticas oficiais, que têm buscado desenvolver melhorias na eficiência de seus processos de captura de informações e criar índices, com novas fontes e formas de pesquisa. Um dos tipos de índices que está sendo alvo desse desenvolvimento atualmente no Brasil é o Índice de Preços ao Consumidor (IPC), que mensura oscilações de custos e de poder aquisitivo, por meio de cotações de preços realizadas em determinados espaços de tempo em amostras de lojas que atuam em ambientes físicos. Esses preços são relacionados a um cadastro de produtos, cujas características são definidas previamente com base em um agregado elementar, ou seja, estabelecem-se especificações dos produtos a serem consideradas na captura de preços, como tamanho, marcas, entre outros (IBGE, 2016; IBGE, 2020; IBGE, 2023a; IBGE, 2023b).

Dentre as possíveis utilidades da técnica de *web scraping* na coleta de preços para índices oficiais, destacam-se:

a) viabilizar a criação de índices de preços referentes a operações de vendas virtuais;
b) possibilitar a extensão de índices de preços já existentes para ambientes físicos ao comércio virtual.

No entanto, considerando que o comércio virtual e o comércio físico presente em um local geográfico específico, possuem populações de lojas distintas, o uso de informações de *web scraping* tem sido frequentemente realizado de maneira equivocada em diversos estudos que buscam incluir dados provenientes de *big data* em índices de preços oficiais. Isso ocorre ao generalizar esses dados, tentando substituir a coleta de preços em toda a população de lojas físicas pelo método de *Web scraping*. Esse equívoco decorre do fato de que, embora seja relevante a integração desses dados (dado o espaço crescente que as lojas virtuais têm conquistado na economia global), os preços de ambos os canais comerciais devem ser tratados como informações complementares do comércio em uma determinada localidade (como proposto nesta tese) e não como substitutos.

Por outro lado, para facilitar a coleta de dados de preços em lojas físicas, existem ainda estudos que utilizam o método *scanner data*, um dos métodos mais eficientes para essa finalidade, por conseguir capturar informações de notas fiscais eletrônicas. Contudo, todos os estudos encontrados na literatura até o presente momento, que visam aplicar fontes de *big data* em índices de preços, utilizam amostras não probabilísticas, sendo esses estudos citados no Capítulo 2.

Considerando as limitações identificadas em pesquisas sobre índices de preços baseados em dados de *big data*, esta tese propõe a estimação das médias de preços de um conjunto de produtos ofertados por lojas em ambientes comerciais virtuais. Essa estimação é essencial, pois a média é um dos principais parâmetros que compõem índices de preços. Para isso, os preços coletados por meio de *web scraping* serão tratados como amostras de um processo de Bernoulli, possibilitando uma modelagem probabilística. Ademais, sugere a integração dessa abordagem com o método de captura e recaptura para construir estimadores não viesados de parâmetros de preços, aplicáveis na composição de índices oficiais que envolvem dados coletados de lojas virtuais, possibilitando a atribuição de uma probabilidade para cada unidade extraída da população-alvo de preços. Além disso, os planos amostrais propostos nesta pesquisa também combinam conceitos de outros métodos probabilísticos, como técnicas de amostragem em dois estágios, múltiplos cadastros e amostragem bidimensional.

Contudo, ao lidarmos com uma população originada do universo *Big data* em índices de preços, o desenvolvimento e a aplicação desses métodos de amostragem exigem adaptações nas teorias existentes na literatura, uma vez que os estudos encontrados sobre Amostragem por

Captura-Recaptura (Seber, 1982; Plante; Rivest; Tremblay, 1998; Thompson, 2002; Lavallée; Rivest, 2012) lidam com populações cujos dados não fazem parte do universo *Big data*. Além disso, estudos que utilizam Amostragem de Múltiplos Cadastros (Mecatti; Ferraz, 2015; Mecatti, 2007) e Amostragem Bidimensional (Bethlehem, 2009; Teixeira Jr., 2020) tratam de populações com cadastros conhecidos.

Para viabilizar o desenvolvimento desses planos amostrais, foi necessário, inicialmente, identificar e definir a população de lojas virtuais. Nesse sentido, esta pesquisa introduziu o termo "ambiente comercial virtual", que se refere a toda plataforma online composta por um conjunto de comerciantes que anunciam seus preços e vendem seus produtos (Seção 3.1). Nesse contexto, assume-se que os resultados obtidos nas pesquisas realizadas nessas plataformas refletem exclusivamente os vendedores que possuem capacidade de distribuir seus produtos para a localização do comprador.

Para ser possível executar a proposta desta pesquisa, também foram definidas etapas computacionais e metodológicas necessárias para a elaboração da estratégia de captura de preços por *web scraping* voltada para a construção de índices de preços (Quadro 1). Adicionalmente, foram identificados os cuidados essenciais a serem observados em cada uma dessas etapas, a fim de evitar tendências nas estimativas.

Quadro 1 – Etapas que compõem o processo de captura de preços por meio de *web scraping* para fins de inferência

ETAPA	DESCRIÇÃO
1) DEFINIÇÃO DO(S) TIPO(S) DE FONTE(S) BASE(S)	Investigar em quais ambientes virtuais é possível encontrar preços de vendedores que atendem à região relacionada ao índice e elaborar estratégias capazes de cercar, da melhor forma possível, as possibilidades de buscas por essas informações. Exemplos de tipos de ambientes virtuais que oferecem informações de preços: Plataformas de Buscas, Farejadores/Comparadores de preços, Sites de empresas, Marketplaces e Redes sociais.
2) ESPECIFICAÇÃO DAS FONTES DE DADOS	Definir as categorias de produtos que serão incluídas no cadastro, a fim de identificar as plataformas que vendem esses produtos nos ambientes identificados na Etapa 1. Por exemplo, para a venda de produtos da categoria vestuário, a população de *marketplaces* inclui lojas como Shopee, Shein, entre outras, enquanto a população de plataformas de busca engloba páginas como Google Shopping, Bing Shopping, e similares.
3) AMOSTRAGEM UTILIZANDO *WEB SCRAPING*	Implementar um plano amostral em duas etapas principais, de modo que, inicialmente, será realizada a seleção de uma amostra de produtos a partir de um cadastro previamente definido. Essa etapa precede o processo de *web scraping*, no qual, para cada item da amostra, as informações de venda — incluindo preços, nomes das lojas virtuais e descrições dos produtos — serão extraídas diretamente dos códigos HTML das páginas de resultados de pesquisa das plataformas definidas na Etapa 2.
4) CLASSIFICAÇÃO	Criar filtros especializados para processar as informações coletadas por meio de *web scraping*. Esses filtros têm como objetivo principal identificar e extrair os dados que realmente atendem aos critérios da pesquisa, como os preços relacionados especificamente às unidades da amostra do cadastro de produtos. Esse processo é fundamental, pois as capturas em ambientes virtuais não retornam

	apenas os itens pesquisados, mas também dados sobre produtos correlatos, os quais precisam ser corretamente identificados e excluídos.
5) CONSTRUÇÃO DO ÍNDICE	Desenvolver estimadores que considerem as particularidades do processo de captura de preços em lojas virtuais para a construção da expressão do índice de preços.

Fonte: Elaboração da autora (2024).

Apesar das etapas 1 e 2, expostas no Quadro 1, auxiliarem na definição da população de lojas virtuais que terão seus preços selecionados na etapa 3, é preciso atentar-se ao fato de que a web é vasta e composta por plataformas que dispõem de algoritmos programados para disponibilizar resultados de buscas com base em seus próprios bancos de dados, os quais possuem interseções de informações entre si. Assim, quando fazemos *web scraping*, em um mesmo tempo t, em mais de uma dessas páginas, ocorre o que podemos chamar de "emaranhamento de dados". Esse "emaranhamento" está relacionado ao fato de podermos encontrar o mesmo dado em mais de uma plataforma de pesquisa ou site, o que torna complexo o conhecimento de uma população, como a de preços de lojas virtuais. Por isso, o processo de amostragem pertencente à etapa 3 utiliza, inclusive, amostragem de múltiplos cadastros, a fim de lidar com esse emaranhado de dados.

Outro problema resultante da captura de dados por meio de páginas da web refere-se ao fato de que os resultados de buscas nessas páginas são programados para serem compostos de informações relacionadas (e não exatas) ao que foi pesquisado. Por isso, é possível que a captura de um grande volume de dados, por si só, ofereça estimativas distantes da realidade da população-alvo. Além disso, estimativas baseadas em conjuntos de sites escolhidos por conveniência para a captura de dados de preços não podem ser generalizadas para todo o ambiente virtual.

Por esses motivos, as etapas de 1 a 4 são importantes, pois reduzem os erros não amostrais das estimativas, como o erro de cobertura de cadastro, o de processamento, o de seleção e o de especificação. O primeiro pode ser ocasionado pela falta de um esquema amostral que cerque a população-alvo (quando a etapa 1 e a etapa 2 não são bem definidas); o segundo poderá estar presente caso o algoritmo não consiga processar os dados da melhor forma (quando há falha na etapa 3); já o terceiro e o quarto estão relacionados ao filtro construído na etapa 4, que deve ser projetado para evitar a seleção de preços de produtos fora da amostra e identificar corretamente a qual produto da amostra o preço capturado pertence.

A partir disso, pode-se perceber que acessar e considerar essa fonte de dados no processo de geração de índices de preços apresenta desafios não apenas do ponto de vista metodológico

estatístico, mas também do ponto de vista tecnológico, o que demostra que, na era *big data*, a amostragem desempenha um papel renovado, mas altamente crucial.

Para uma melhor apresentação da resolução do problema exposto, esta tese está organizada da seguinte forma: O Capítulo 2 apresenta uma revisão dos projetos que buscam incluir dados de fontes alternativas, como *web scraping*, em índices de preços, com a discussão dos principais desafios enfrentados por esses estudos e possíveis soluções para superá-los. O Capítulo 3 define a população-alvo de lojas virtuais e os parâmetros a serem estimados. Além disso, descreve os planos amostrais e estimadores não viesados desenvolvidos para inferir sobre esses parâmetros. Por fim, propõe um plano amostral que permite a estimativa do total das médias de preços das lojas físicas e virtuais para um conjunto de produtos, de forma integrada. O Capítulo 4 detalha os procedimentos realizados em um experimento piloto do plano amostral, considerando a população de preços de lojas virtuais que comercializam os celulares lançados no primeiro semestre de 2023 por marcas oficialmente vendidas no Brasil. Este experimento utiliza dados coletados nas plataformas Google Shopping, Bing Shopping e Mercado Livre e os resultados obtidos são relatados e analisados. Encerrando a tese, são apresentadas as considerações finais e sugestões para trabalhos futuros.

1.1 Objetivo Geral

Esta tese tem como objetivo geral propor um plano de amostragem para coleta de dados relativos a uma população de “lojas virtuais”, por meio de algoritmos de *web scraping*, visando gerar estimativas de preços médios, componentes de índices de preços.

1.2 Objetivos Específicos

a) Propor uma teoria de amostragem que envolva a definição de conceitos-chave relacionados a uma população de lojas virtuais que justifique o uso de planos amostrais probabilísticos em suporte a dados coletados pela web.
b) Propor uma estratégia de estimação para médias de preços relacionados a lojas virtuais.
c) Propor uma estratégia de estimação que combine estimativas de médias de preços relacionados a lojas virtuais e físicas.
d) Aplicar a metodologia proposta em uma pesquisa experimental por meio da coleta de preços de telefones celulares por *web scraping*.

2 DESAFIOS PARA O USO DE *BIG DATA* COMO FONTE DE INFORMAÇÃO DE ÍNDICES DE PREÇOS

Este capítulo abordará algumas iniciativas de estudos na literatura que visam utilizar dados da web para gerar estimativas a respeito de diferentes variáveis relacionadas com os preços ofertados por lojas virtuais. Além dos estudos que serão citados nas próximas seções, atualmente existe o Hub Regional das Nações Unidas para Big data no Brasil, sediado no Instituto Brasileiro de Geografia e Estatística (IBGE), na Escola Nacional de Ciências Estatísticas (ENCE), que trabalha em parceria com outros órgãos governamentais e não governamentais nacionais, com as Nações Unidas, com os Institutos Nacionais de Estatística regionais e com outras instituições acadêmicas e de pesquisa. Esse Hub visa contribuir para o desenvolvimento de métodos que utilizam *big data* para tornar a produção de estatísticas oficiais mais eficiente e eficaz, promovendo o compartilhamento de conhecimento e iniciativas inovadoras na América Latina e Caribe.

Com base em Chessa, Verbug e Willenborg (2017), existem exemplos de países em que seus Institutos Nacionais de Estatística (INE's) já estão explorando fontes alternativas de dados para seus respectivos índices de preços nacionais. Esses países são: Argentina, Austrália, Holanda, Noruega, Cingapura, Suécia, Suíça e Nova Zelândia. Além desses, os INE's do Reino Unido, EUA, Coreia, Itália, Holanda, Japão e Malásia também estão considerando a possibilidade de uso de dados online nos seus Índices de Preços ao Consumidor oficiais (Cavallo, 2017).

Uma vez que a execução de algoritmos de *web scraping* é um dos meios mais eficientes na atualidade para capturar informações na internet, existem pesquisas que tratam esse método como uma das fontes mais promissoras de dados para índices de preços ao consumidor. Algumas delas serão descritas nas seções abaixo e visam substituir a coleta de preços em estabelecimentos físicos por capturas de preços por meio de *web scraping*, o que parece ser uma prática equivocada, uma vez que a população de lojas virtuais é diferente das lojas que atuam em ambientes físicos. Ademais, como iremos ver na Seção 2.4, essas populações possuem comportamentos de ofertas de preços diferentes.

Por outro lado, existem pesquisas, que também serão abordadas nesta seção, como Uriarte *et al*. (2019), que buscam criar índices de preços para lojas virtuais utilizando *web scraping*, mas, para extração de preços de lojas físicas, estão pesquisando outros métodos, como Scanner Data, que, até o momento, é o método mais explorado para essa finalidade pelos países citados acima, por conseguir, eficientemente, extrair dados de preços de produtos e de

quantidade de vendas, através de notas fiscais de compras disponibilizadas pelas lojas que atuam em ambientes físicos.

Outros estudos, como Ten Bosch et al. (2018), consideram o uso do método *web crawling* como alternativa do uso de *web scraping* para extrair preços da internet. No método de *web crawling*, são rastreados todos os dados de uma página especificada, a fim de capturar *hiperlinks*, os quais são identificados e passam a ter seus dados capturados. Com isso, a diferença entre os métodos se dá devido à técnica de *web crawling* retornar informações genéricas, enquanto o método de *web scraping*, informações mais específicas.

Contudo, mesmo com todas essas pesquisas e anseios de institutos nacionais para incluir fontes alternativas nos seus índices de preços, não foram encontrados na literatura artigos que se aprofundam em metodologias de amostragens probabilísticas para índices de preços oficiais que buscam utilizar fontes de *big data*. Consequentemente, os índices que utilizam essas fontes, até agora propostos, estão englobando produtos e estabelecimentos escolhidos com base na facilidade de coleta de dados. Essa prática gera estimativas que não podem ser generalizadas para toda população de lojas virtuais, uma vez que amostras selecionadas de forma não probabilística estão sujeitas a erros não amostrais.

Além da construção de um plano amostral probabilístico para a captura de informações de preços de lojas virtuais, os estudos que se propõem a construir índices que utilizam fontes *big data* necessitam se atentar a algumas condições para construção de índices de preços em geral, que podem não ser supridas quando os dados são provenientes de *web scraping*, caso não haja um planejamento adequado para tal. Tais condições englobam: a) O índice de preços precisa conter todas as informações dos estabelecimentos e produtos que compõem a sua amostra, em todo período que se propõe a gerar estimativas; b) Em um IPC é necessário que toda informação seja identificada corretamente, respeitando o fato de que a comparação de preços deve ser exatamente entre os preços de um mesmo produto, em tempos diferentes, e devem se referir à mesma forma de pagamento, sem contabilizar descontos, o que requer a utilização de um algoritmo de mineração e classificação de dados eficaz; c) Em um IPC a população-alvo deve ser bem definida, portanto, a utilização de uma fonte alternativa de preço deve retornar o mesmo preço da fonte original. Como as populações de lojas físicas e de lojas virtuais possuem comportamentos diferentes, as substituições dos seus preços não são adequadas (ainda que sejam de lojas com mesma razão social), sendo necessária uma integração correta dessas duas populações; d) O IPC visa medir a mudança no preço de um produto específico (incluindo todas as suas características) ao longo de um tempo. Como os atributos de um item estão sujeitos a mudanças, é necessário a utilização de métodos que estime as

oscilações de qualidade do produto, para identificar e eliminar a mudança de preço no índice, ou que permita mudanças regulares do cadastro de produtos utilizado.

Com isso, abaixo iremos pontuar pesquisas que se propõem a utilizar fontes alternativas para índices de preços, organizando-as pelo problema que mais se destaca em seus métodos. Apesar de todas terem um ponto fraco em comum: não possuem um plano amostral probabilístico para a seleção de amostras de preços (foco desta tese), iniciaremos citando estudos que pensaram na resolução da maioria dos desafios citados, mas não mencionaram solução para tal.

2.1 O problema da construção de plano amostral probabilístico para lojas virtuais

Como exemplo de índice de preços que utiliza alternativas para muitos dos problemas que devem ser enfrentados ao considerar dados de fontes de *big data* nas suas estimativas, temos o "Índice Online", implementado na cidade Bahia Blanca, na Argentina. Segundo Uriarte *et al*. (2019), essa é uma proposta de Índice de Preços ao Consumidor que utiliza *web scraping* e *scanner data*, incluindo em seus cálculos informações de lojas físicas e virtuais, de modo que uma parcela significativa das categorias de produtos já presentes em índices de preços de lojas físicas tem seus preços extraídos por *scanner data*. Além disso, para completar todo o conjunto de categorias do Índice de Preços ao Consumidor do INDEC (Instituto Nacional de Estatística e Censos) da Argentina, alguns preços são obtidos da forma tradicional, por meio de coleta presencial de dados, como preços de: serviços, de lojas não virtuais, de cigarros e de mensalidades escolares. Ademais, ainda conforme Uriarte *et al*. (2019), o projeto recolhe informações que abarcam mensalmente uma média de doze mil itens e quarenta mil preços da internet através de *web scraping*. Todavia, não há uma definição da população-alvo de onde é feita a captura de preços das lojas virtuais, e os dados utilizados são amostras não probabilísticas.

Outro exemplo de estudo que não utiliza amostragem probabilística para selecionar os dados que compõem seus índices de preços propostos é descrito em Polidoro *et al.* (2015), que construiu índices relacionados a produtos eletrônicos e passagens aéreas da Itália. Esses bens e serviços foram escolhidos para terem seus preços capturados por *web scraping*, considerando que os produtos eletrônicos são os mais comprados pela internet no país em questão e pelo fato das cotações de passagens aéreas manuais serem ineficientes e custosas. Nesta pesquisa, para construir o índice de preços de produtos eletrônicos a partir de dados coletados por *web scraping*, foram utilizadas nove lojas avulsas, sem seleção por amostragem probabilística. Neste

contexto, ficou evidente a presença de erro de não resposta, também observado no índice de preços para passagens áreas, quando o conjunto de informações de preços não capturadas pela ferramenta *web scraping* diferem das obtidas por essa ferramenta. Houve também erro de cobertura, pois as listas utilizadas para a seleção dos produtos eletrônicos e das lojas virtuais que oferecem preços para esses itens não representavam a totalidade da população-alvo da pesquisa. Esse último fato é um indicativo de que além de ser complexo conhecer a população-alvo de lojas virtuais, identificar a população de todos os produtos vendidos na web, mesmo que por categoria, também é um desafio.

Em suma, o plano amostral probabilístico nas coletas de dados por *web scraping* está relacionado com as etapas 1, 2 e 3 da construção de índices de preços de fontes alternativas, mencionadas na introdução. Diante disso, pode-se perceber a importância da construção de um plano amostral probabilístico eficaz e eficiente para índices de preços que envolvem *big data*, o que desconstrói a percepção de que grandes volumes de dados, por si só, oferecem estimativas precisas. Esse desafio também foi enfrentado pelo projeto de índice de preços do Google, composto por dados da internet, que ainda não foi executado (Ten Bosch, 2018). Visto que o Google tem acesso a um grande volume de dados, os desafios enfrentados pela empresa parecem estar mais relacionados aos aspectos metodológicos e estatísticos, os quais são reconhecidamente complexos, e não à falta de informações disponíveis.

Com isso, pela escassez de estudos sobre amostras probabilísticas de preços ofertados por lojas virtuais, para índices oficiais, a proposta teórica desta tese inclui uma definição de população passível de ter seus dados coletados por *web scraping* e apresenta propostas de planos amostrais, com estimadores não viesados, capazes de inferir sobre essa população de forma eficaz e eficiente.

2.2 O problema de quebra de dados provenientes de *web scraping*

Um dos maiores desafios vistos nos estudos analisados, como em Breton *et al*. (2016), e que também é um dos desafios desta tese, é que, embora a captura de preço por *web scraping* permita que muitas observações sejam produzidas até mesmo em um dia, a oscilação no número de preços observados também é reflexo dos problemas inerentes ao processo de extração dos dados na web, que pode deixar de capturar algum dado desejado, em algum momento. Algumas possíveis razões para que os dados de uma determinada fonte deixem de ser capturados são: alteração de alguma forma do layout do site (necessitando modificar o algoritmo de *web scraping*), adição ou remoção de itens em ofertas online, problema com código de raspagem ou

servidor, remoção ou suspensão de sites ou lojas virtuais e indisponibilidade total ou temporária de um produto. Esses fatores, muitas vezes, causam não resposta na coleta de dados. Por exemplo, é possível que o *web scraping* deixe de capturar um tipo de informação de um determinado site pelo fato de o desenvolvedor ter mudado a informação de lugar, ou por bloqueio de robôs do próprio site, como ocorreu com o índice de preços de passagens aéreas desenvolvido por Polidoro *et al.* (2015), que enfrentou dificuldades de permissão de alguns sites para a realização de *web scraping* e bloqueios inesperados de capturas de dados de outros sites, ocasionando quebras de dados nos cálculos desse índice. Portanto, é necessária uma constante vigilância e a criação de caminhos para que esse tipo de erro seja corrigido, possibilitando a qualidade dos dados utilizados na composição do índice.

Assim, ainda que as técnicas de *web scraping* reduzam uma das principais fontes de erros de medição, que é o erro de medida nas coletas de dados, se o algoritmo não for bem desenvolvido, bem gerenciado e ajustado regularmente, pode gerar diversos tipos de erros não-amostrais, incluindo erros de processamento e de não resposta. Portanto, esse desafio está atrelado à etapa 3 da construção de um índice de preços que utiliza fontes de dados alternativas (descrito na introdução), sendo necessário pensar em estratégias que possibilitem a eficácia de um algoritmo de *web scraping* dentro de um plano amostral probabilístico, definindo quando e de que modo ele pode ser executado e implementado.

Ciente desse desafio, esta tese propõe utilizar o método amostral de captura e recaptura para dar noção de tamanho da população, permitindo o cálculo da probabilidade de inclusão de primeira ordem sob um plano de Bernoulli.

2.3 O problema da mineração e classificação de dados

A identificação precisa dos produtos nos resultados de *web scraping* apresenta desafios significativos para a construção de índices de preços provenientes de lojas virtuais. Esses desafios decorrem, em parte, do baixo nível de detalhamento das características dos produtos nos resultados obtidos, e do fato de que lojas virtuais distintas podem descrever o mesmo item de maneiras diferentes. Esses fatores dificultam a classificação dos produtos, a comparação de preços de um mesmo item em diferentes períodos e a exclusão de dados que não fazem parte da população-alvo.

Além disso, a falta de especificidade nos anúncios inclui problemas como a identificação de itens em liquidação, anúncios que representam conjuntos de itens ou vendas casadas, e até mesmo produtos usados. Outro ponto crítico é determinar com clareza o tipo de

preço capturado, como preços para compras à vista ou a prazo. Esses fatores introduzem ruídos nos dados, comprometendo a inferência. Portanto, torna-se essencial identificar e eliminar dados que não correspondem à população-alvo, garantindo a qualidade e a precisão na construção dos índices.

Loon e Roels (2018), por exemplo, propõe a não utilização de dados em liquidação em índices de preços, porém, afirmam ainda não ter encontrado solução para tal. O método de Índice Digital de Preços para Alimentos, proposta por Lynch, Stansfield e Olivecrona (2019), também é um exemplo de pesquisa que tem como desafio a construção de um algoritmo que classifique os produtos até um agregado elementar.

Assim, informações limitadas sobre características e descrições de produtos exigem novas abordagens para índices de preços que utilizam dados de *web scraping* e *scanner data*, pois, se os dados coletados em um tempo t incluírem informações que não podem ser comparadas com informações coletadas em um outro tempo em que se deseja fazer a comparação, eles não poderão ser utilizados em um índice de preços, já que os preços dos produtos não poderão ser relacionados, nem substituídos.

Alguns estudos utilizaram *Machine Learning* para treinar algoritmos capazes de identificar quando um anúncio não está relacionado ao produto cujo preço se deseja analisar. Uma desses estudos foi a de Breton *et al*. (2016), que ensinou o algoritmo a fazer as seguintes classificações nas descrições de produtos capturadas: ‘concorda corretamente com a classificação’, ‘concorda incorretamente com a classificação’, ‘produto novo’ e ‘discordância na classificação’. Contudo, pode-se perceber que o desenvolvimento de algoritmos de mineração e classificação de dados, nessa e em outras pesquisas que englobam dados extraídos por *web scraping* em índice de preços, são etapas desafiadoras. Por isso, soluções para tal problema são escassas na literatura.

Para realizar o experimento piloto dos planos amostrais propostos na Seção 3.4, foi necessária a adoção de algumas soluções para esse problema. Para isso, foi desenvolvido um algoritmo (Apêndice B) que, além de simular buscas por produtos com o maior nível de detalhamento possível – permitindo que as plataformas de busca aproximem seus resultados ao que está realmente sendo procurado –, incorpora filtros de mineração de dados. Esses filtros utilizam as palavras das descrições de cada produto para filtrar somente as informações realmente condizentes com os itens que compõem o cadastro pré-definido utilizado. Deste modo, caso um produto seja descontinuado no mercado, o algoritmo de *web scraping* deixará de capturar informações relacionadas a ele e automaticamente o removerá do cadastro.

Sabe-se que essas soluções adotadas não são as mais eficazes, pois a seleção dos dados fica refém do conjunto de palavras definidas como decisivas para a classificação. Por exemplo, pode ocorrer de preços de produtos desejados não serem capturados devido à ausência, na descrição do produto, de alguma palavra desse conjunto. Neste caso, percebe-se que, quanto maior o número de palavras utilizadas, maiores são as chances de não capturar dados relevantes; por outro lado, quanto menor o número de palavras, aumenta o risco de capturar dados de itens que não correspondem ao desejado.

Contudo, por não ser um objetivo desta tese, os resultados de *web scraping* dessa pesquisa, utilizando essas soluções, foram considerados aceitáveis, embora seja parte de trabalhos futuros solucionar esse problema através de técnica *Fuzzy* ou através de algoritmo de *machine learning* não supervisionado de classificação.

Além de classificar os produtos, é fundamental eliminar anúncios duplicados nas capturas de *web scraping*. Anúncios duplicados são aqueles que se referem ao mesmo produto, na mesma loja, na mesma plataforma e com o mesmo valor, evitando que tais informações sejam contabilizadas mais de uma vez nas estimativas. Para isso, tornou-se essencial identificar os links de origem de cada informação. No entanto, como um mesmo link pode alterar sua estrutura final dependendo da plataforma em que é acessado, foi necessário identificar padrões nos códigos utilizados por cada plataforma em seus links. Isso permitiu extrair apenas os links "crus" (desprovidos de códigos únicos das plataformas), tornando-os comparáveis. Com essa abordagem, foi possível detectar tanto duplicidades dentro de uma mesma plataforma quanto interseções entre diferentes plataformas.

2.4 O problema do comportamento distinto dos preços das lojas físicas e virtuais

O comportamento distinto dos preços em lojas físicas e virtuais, considerando um mesmo tempo t, é um dos desafios a serem enfrentados em índices de preços que utilizam ambas as fontes de forma concomitante. Esse problema está diretamente relacionado à etapa 5 da construção de índices de preços que utilizam fontes de *big data* (Quadro 1), especificamente no que diz respeito à formulação da Expressão do índice. Embora essa etapa não esteja incluída nos objetivos desta tese, sua análise é relevante, pois pode contribuir para estratégias mais eficazes de integração dos estimadores relacionados a esses dois tipos de lojas, considerando suas diferenças de comportamento.

Um exemplo de estudo que desconsiderou esse problema é descrito por Uriarte *et al*. (2019), que combinou preços de lojas físicas e virtuais em um único índice, tratando-os, de

forma equivocada, como substitutos diretos. A evidência de que esses canais de venda apresentam comportamentos distintos é demonstrada por Cavallo (2017). Nesse estudo, realizado em diversos países, foi identificado que, em geral, os preços entre esses canais não são sincronizados. No caso específico do Brasil, foi observado que apenas 18% das alterações semanais de preços ocorrem simultaneamente entre lojas físicas e virtuais. Além disso, constatou-se que os preços são idênticos em apenas 42% das consultas (variando entre setores), sendo predominantemente mais baixos no ambiente virtual. Esse resultado esteve de acordo com o estudo de Machado e Crispim (2017), na qual foi identificada uma diferença média de 13,3% entre os preços das lojas físicas e virtuais, ao mesmo tempo em que 90% das suas comparações observaram inferioridade nos preços das lojas virtuais.

Cavallo e Rigobon (2016) trazem evidências de que a inflação dos preços online tende a antecipar a inflação dos preços nas lojas físicas. Essa relação também está alinhada com análises como a de Ellison e Ellison (2009), que destacam as lojas virtuais como um ambiente mais competitivo, uma vez que na internet, ao contrário do ambiente físico, não é necessário se deslocar entre diferentes lojas para comparar preços, o que torna o processo mais eficiente e acessível para os consumidores. Portanto, existem fatores que afetam significativamente os comportamentos dos preços das lojas virtuais em detrimento dos preços das lojas físicas, o que impede a substituição desses preços em um índice específico, em um tempo t.

Diante do exposto, esta tese propõe a definição de uma população de lojas virtuais para viabilizar a coleta de preços e a construção de um determinado índice de preços. Essa definição permitirá a integração entre a população de lojas físicas e a população de lojas virtuais, formando uma única população passível de inferência probabilística. Para tanto, o plano amostral deverá ser estruturado de forma que a seleção de preços seja realizada separadamente para cada tipo de mercado, assegurando também que ambos sejam amostrados no mesmo período t.

Além disso, como o método tradicional de coleta de preços das lojas físicas é mais custosa e demorada, em relação à coleta de preços por *web scraping*, é sugestivo que o método de integração dos preços dos mercados físicos e virtuais proposta por essa tese seja replicada em intervalos de tempo curtos (como semanalmente), por meio de atualizações dos preços dos mercados físicos, coletados mensalmente, com base nas oscilações de preços dos mercados virtuais da semana vigente. Outra sugestão para utilizar a metodologia de integração dessa tese com maior frequência, é realizar a coleta dos preços dos mercados físicos através de técnica de *scanner data* (processo mais eficiente), e de pesquisa de campo para produtos e mercados que ainda não podem ser alcançados por essa fonte alternativa.

2.5 O problema do acompanhamento do “ciclo de vida” do produto

A constante alteração dos atributos dos produtos, por meio de atualizações de versões, ou a retirada do produto do mercado — ou seja, seu “ciclo de vida” — pode resultar na impossibilidade de encontrar produtos pesquisados em um momento específico ao longo do tempo. Isso foi a motivação de alguns estudos, pois representa um desafio significativo na construção de índices de preços, visto que é necessário desenvolver uma metodologia que permita a comparação de informações em dois momentos distintos. Em Uriarte *et al.* (2019), à medida que os produtos aparecem e desaparecem das listas da web, a cesta de bens e serviços que representam cada categoria muda em sua composição, impossibilitando que uma mesma cesta seja comparada diretamente por períodos longos no seu índice. Polidoro *et al*. (2015), por sua vez, fez para cada produto uma segmentação, com base em dados técnicos e especificações de desempenho, fixada anualmente, possibilitando o estabelecimento de requisitos mínimos para os produtos serem identificados nas coletas mensais. Além disso, foi proposta a substituição contínua dos produtos que perdem a importância na loja por outros atualizados, porém, não houve detalhamento de como isso pode ser automatizado.

Já o método proposto em Lynch Stansfield e Olivecrona (2019), utiliza somente uma cesta fixa, que não se adapta facilmente à introdução de novos itens ou à remoção de itens antigos. Reconhecemos isso como um ponto fraco em um índice de preços, pois as preferências e necessidades dos consumidores mudam constantemente, além do fato de os produtos serem criados, atualizados e modificados também com frequência.

Outros artigos focaram na estrutura da expressão de índices de preços que utilizam *web scraping* e *scanner data* como fontes de dados, contudo essas pesquisas também não seguiram nenhum tipo de amostragem probabilística para selecionar esses dados, logo, utilizaram todos os dados “achados” e “garimpados” por códigos de mineração, sem ao menos definir sua população-alvo. Apesar da construção do cálculo do índice, no que diz respeito a comparação de preços entre dois tempos, não ser o foco dessa pesquisa, é importante discutir a respeito desse problema, pois ele norteia como e quando deve ser realizado o plano amostral definido. Além disso, pode ajudar pesquisas que desejam definir em qual índice aplicar os estimadores propostos nesta tese.

Exemplos de estudos que buscam resolver esse desafio são: Krsinich (2011), que criou o Índice de Efeitos Fixos (FE), Haan e Krsinich (2014), que criou o índice ITRYGEKS, e Krsinich (2016), que desenvolveu o Índice de Emenda de Janela de Efeitos Fixos – FEWS. Esses índices focaram em problemas inerentes à característica volátil dos dados provenientes

de códigos de barra ou de identificadores de produtos e à definição das características e qualidade de um produto utilizado no índice. Como esses índices requerem informações sobre as características dos itens envolvidos, Krsinich (2016) utilizou regressão hedônica, partindo do pressuposto de que os dados de *Scanner Data* e *Web scraping* não possuem informações suficientes sobre características dos produtos para o acompanhamento de mudanças nos seus atributos (o que ele chama de "ajuste de qualidade"). Por isso, o método FEWS incorpora na expressão um fator de revisão, que reflete os movimentos de preços implícitos associados à introdução de novos produtos no período, e de alterações de item (como embalagem, quantidade ou qualquer outra qualidade) em uma determinada janela de tempo, porém, isso leva o índice a não abarcar produtos que são inseridos na loja no período entre as janelas utilizadas.

Além disso, estudos como Haan e Hendriks (2013) e Williams e Sager (2019), também apontam através de resultados empíricos que o Índice GEKS, o Índices de Jevons Encadeado e os índices de Efeitos Fixos em geral podem não ser adequados para dados da web, por não possuírem estrutura de cálculo para estimar variação de preços de informações de produtos que possuem identificadores que mudam com frequência. Estudos como Silver e Heravi (2005) e Greenlees e McClelland (2010), por sua vez, descobriram que os métodos hedónicos convencionais também não abordam os efeitos do ciclo de vida do produto, apresentando grandes declínios, a menos que os coeficientes sejam limitados a um valor fixo ao longo do intervalo de tempo do índice estimado.

Como é necessário lidar com os efeitos do ciclo de vida dos produtos, ou seja, acompanhar as mudanças das características dos produtos no cálculo dos índices de preços dentro de toda a janela de tempo em que os mesmos são calculados, outros estudos, como Breton *et al*. (2015, 2016), propõe a utilização do índice GEKSJ, que é a combinação do índice de GEKS e o índice Jevons encadeado, para calcular a variação das médias geométricas de preços capturados através de *web scraping* e *scanner data* entre determinados períodos. entretanto, classificações incorretas nos dados afetarão o índice GEKSJ em todos os momentos. Isso ocorre porque um índice GEKS usa todas as informações históricas para calcular o índice atual. Deste modo, as expressões dos índices ficam reféns dos melhoramentos da classificação dos produtos.

Uma das soluções dada por Konny, Williams e Friedman (2022), foi a combinação e substituição de preços individuais no IPC por um preço médio de um item único ou de um conjunto definido de itens. Alternativamente, sugeriram substituir os relativos de preços no IPC por estimativas de variação de preços baseadas em novas metodologias. Assim, nesse estudo foi utilizado o índice Tornqvist para uma seleção de oito categorias de itens em uma cidade,

porém, os índices dos modelos correspondentes caíram vertiginosamente, e várias categorias de itens mostraram um declínio de mais de 90% em menos de dois anos. Isso ocorre pois os produtos são lançados com preços altos e neles são empregados descontos ao longo do tempo. Para a solução desse problema, Breton *et al*. (2016) aponta que o índice RYGEKS (Ivancic; Diewert; Fox, 2011), pode ser uma boa opção para resolver esse problema, apesar de ainda utilizar séries temporais de preços de curta duração.

Ademais, levando em consideração o estudo de Cavallo e Rigobon (2016), a qual identificou que 76% dos produtos ofertados por lojas físicas também são encontrados online, e que o Brasil é um dos países que mais tem produtos vendidos tanto online como em lojas físicas, nesta tese é sugerido que a definição do cadastro de produtos que terão seus preços coletados na internet por *web scraping* seja previamente feita com base na coleta de preços dos mercados físicos, não possibilitando assim muitas alterações na composição da cesta de produtos ao longo do tempo, o que também viabiliza a redução do erro não amostral ocasionado pela substituição de produtos nas estimativas. Além disso, é necessário que o cadastro de produtos seja composto pelo maior nível de agregado elementar de cada unidade que o compõe, ou seja, deve-se considerar todas as características de todos os produtos, para que seja possível a coleta dos seus preços, sem erro de especificação.

3 PROPOSTA TEÓRICA

Para elaborar um plano amostral probabilístico, é fundamental, primeiramente, definir a população-alvo da pesquisa. Nesse contexto, na Seção 3.1, será realizada uma análise do comércio virtual, explicando a organização e a estrutura das lojas dentro do comércio eletrônico como um todo, considerando um determinado momento t. A Seção 3.2, por sua vez, irá delimitar o conjunto de vendedores, denominados "lojas virtuais", que compõem a população de lojas responsáveis pela oferta dos preços incluídos nos parâmetros apresentados na Seção 3.3. Estes parâmetros serão estimados a partir do plano amostral desenvolvido na Seção 3.4, que permitirá a obtenção de uma estimativa aproximadamente não enviesada para índices de preços praticados por lojas virtuais. Além disso, na Seção 3.5 será apresentada uma analogia entre o problema resolvido pelo método de captura e recaptura nesta pesquisa e sua aplicação tradicional em uma população de peixes. Acredita-se que essa analogia ajudará a ilustrar, de maneira mais acessível, o funcionamento e a aplicação do método de captura e recaptura no contexto dos dados de lojas virtuais, facilitando a compreensão da estratégia adotada para estimar parâmetros de preços em ambientes comerciais virtuais.

Por fim, na Seção 3.6 será apresentada uma proposta de integração dos planos amostrais referentes a lojas virtuais e físicas. Nesta abordagem, será detalhado o desenho da população que combina esses dois tipos de lojas, a expressão do parâmetro que representa a soma das médias de preços dessa população para um mesmo cadastro de produtos, bem como o estimador desenvolvido para este parâmetro. Além disso, serão expostas a variância do estimador e o estimador desta variância.

3.1 O Comércio Virtual e uma Proposta de definição de Lojas Virtuais

A classificação de transações comerciais virtuais ainda é um campo em construção, com diversas denominações propostas por diferentes pesquisadores, como o termo 'E-commerce', o qual, nesta tese, se refere a todas as transações comerciais de produtos e serviços realizadas através de ferramentas online. Estas atividades são exercidas pelas **Lojas Virtuais (LV)**, conceito também definido por esta pesquisa.

É prática comum que internautas busquem informações sobre lojas virtuais e produtos através de plataformas que têm como finalidade facilitar esse processo, fornecendo os dados desejados ou meios para concluir a compra e venda. Sem essas plataformas, seria quase impossível localizar as informações desejadas em um ambiente tão vasto quanto a Web. Nessa

senda, para viabilizar um processo de amostragem probabilística sem a disponibilidade de um cadastro, foi feita uma análise do comércio de Lojas Virtuais. Partiu-se do pressuposto de que, assim como as lojas físicas, localizadas em endereços fixos em determinadas cidades, com vitrines e portas abertas para receber o consumidor durante a busca por produtos, um vendedor na internet deve ser classificado como uma Loja Virtual somente quando anuncia e/ou viabiliza suas vendas, disponibilizando opções de pagamento e meios de negociações de venda e pós-venda, através do que denominamos de "**Ambientes Comerciais Virtuais"**. Esses ambientes são como as ruas comerciais municipais repletas de vitrines, com a diferença de que, para o consumidor acessá-las, é preciso pesquisar um produto por vez. Deste modo, foram pontuados e detectados diferentes tipos de "Ambientes Comerciais Virtuais", que podem ser destinados exclusivamente à venda de um único tipo de produto – por exemplo, plataformas voltadas para o comércio de produtos eletrônicos – ou atuar de forma mais ampla, viabilizando a venda de lojas de diferentes segmentos.

Tipos de plataformas da internet que se destacam dentre os ambientes comerciais virtuais são:

- **Plataformas de busca ou comparação de preços**:

São ambientes virtuais criados para fins comerciais, amplamente utilizados por consumidores que desejam pesquisar preços ou identificar as melhores ofertas de produtos ou serviços oferecidos por lojas virtuais. Essas lojas devem atender à localidade onde o consumidor pretende receber a compra. Tais plataformas direcionam os consumidores para os sites das lojas virtuais, que podem ser tanto sites próprios quanto marketplaces, que funcionam como intermediários de venda.

Exemplos notáveis incluem Google Shopping, Bing Shopping e Buscapé. Os dois primeiros são plataformas de busca, enquanto o último se enquadra como uma plataforma de comparação de preços. Note que o exemplo mencionado se refere exclusivamente ao Google Shopping, excluindo a plataforma Google Search como um todo. Essa exclusão ocorre porque, embora o Google Search também exiba anúncios de lojas que comercializam o produto em estudo, ele não apresenta as características específicas de uma plataforma desenvolvida exclusivamente para fins comerciais.

Por fim, podemos perceber que os resultados de busca por um determinado produto em diferentes plataformas de busca ou comparação de preços possui interseções, pois uma mesma loja virtual pode estar presente em duas ou mais plataformas distintas, anunciando a mesma

página de venda. No entanto, cada plataforma possui características próprias em termos de alcance de lojas virtuais, refletindo suas particularidades no ambiente de comércio virtual.

- **Plataformas de *Marketing Places***:

São Ambientes Comerciais Virtuais que vendem produtos de outras Lojas Virtuais, onde o consumidor também pode pesquisar por um produto e, a partir disso, receber uma lista de lojas com suas respectivas ofertas de preços, prontos para efetuar o processo de venda. Esse tipo de ambiente comercial virtual normalmente é anunciado por Plataformas de busca de preços como uma única Loja Virtual, porém, nele se encontram diversas lojas virtuais anunciando seus produtos e serviços. Contudo, é importante ressaltar a inexistência de interseções entre diferentes *Marketing Places*, pois, ainda que um mesmo vendedor comercialize seus produtos em *Marketing Places* distintos, podem comercializar um mesmo produto por preços diferentes e através de nome diferente de lojas. Além disso, diferentemente das plataformas de busca ou comparação de preços, pesquisas em diferentes *Marketing Places* sempre retornará links diferentes, o que permite a definição de lojas distintas. Um típico exemplo de Mercado Virtual é o Mercado Livre.

Para ilustrar melhor as relações entre os diferentes tipos de plataformas e os tipos de dados obtidos em cada uma, a Figura 1 segmenta as "Plataformas de busca ou comparação de preços" em duas categorias: "Plataformas de busca de preços" e "Plataformas de comparação de preços".

O sistema representado nesta figura mostra que as plataformas de busca de preços não fornecem apenas informações diretamente vinculadas às lojas virtuais, mas também redirecionam para plataformas de comparação de preços e marketplaces. Como estas últimas, na maioria das vezes, não realizam vendas diretas, elas funcionam como intermediárias, agregando anúncios de diferentes marketplaces e lojas virtuais. Esse fluxo de informações entre as plataformas evidencia a conexão entre os diversos ambientes comerciais virtuais e reforça a necessidade de um mapeamento cuidadoso para identificar corretamente as lojas virtuais, que são o nosso principal foco.

Figura 1 – Relações dos resultados de cada tipo de plataforma com as demais e com as lojas virtuais

Fonte: Elaboração da autora (2024).

Além das plataformas apresentadas na Figura 1, também é possível identificar redes sociais (como Facebook, Instagram e TikTok) e aplicativos de mensagens instantâneas (como WhatsApp, Telegram, Viber e e-mail) como outros tipos de ambientes virtuais que viabilizam o comércio, em diversos segmentos. De forma geral, esses canais direcionam o consumidor para um link onde a compra pode ser finalizada. No entanto, a execução de *web scraping* nestes ambientes é mais complexa. Isso ocorre porque seu propósito principal não é o comércio, há restrições impostas pelas políticas de privacidade das contas que publicam os anúncios e, além disso, as pesquisas por produtos nesses canais tendem a ser pouco eficazes. Portanto, por não serem projetados especificamente para fins comerciais e por apresentarem limitações na busca por produtos, as redes sociais e os aplicativos de mensagens instantâneas não podem ser caracterizadas como um ambiente comercial virtual. Como consequência, as vendas realizadas exclusivamente nesses canais serão tratadas como parte do comércio físico informal, os quais não possuem registro ou cadastro em órgãos produtores de estatísticas oficiais.

Diante do exposto, como as plataformas de busca, comparação de preços e marketplaces foram os únicos canais de venda que se encaixaram na definição de Ambiente Comercial Virtual, doravante serão denominados nesta tese simplesmente como “Plataformas”.

Além disso, assumiremos que os resultados de busca em uma plataforma, no tempo t, são baseados em um processo de amostragem de Bernoulli em uma população, desconhecida e específica de cada plataforma, que inclui outras plataformas e lojas virtuais. Estas populações são compostas por lojas e plataformas que negociaram de forma gratuita, paga ou foram identificadas por seus algoritmos como relevantes. Desta forma, a união das populações de lojas virtuais pertencentes a cada plataforma considerada neste estudo, no tempo t, constitui a população de lojas virtuais deste mesmo período.

Essas definições e hipóteses fornecerão a base para as discussões nas próximas seções e viabilizarão o plano amostral apresentado na Seção 3.4.

3.2 População de Preços das Lojas Virtuais

Para utilizar a população de lojas virtuais, apresentada na seção anterior, como base para amostragem probabilística, é necessário considerar certas características inerentes à distribuição dos dados dessa população. Um dos principais desafios é (i) como lidar com as lojas virtuais pertencentes às interseções entre as populações de cada plataforma. Esse fato refere-se à situação em que uma mesma informação pode ser acessada por meio de diferentes páginas ou plataformas. Além disso, a (ii) velocidade com que os dados das lojas virtuais são incluídos e excluídos das plataformas virtuais é desconhecida, e (iii) o modo como essas informações estão disponíveis para diferentes usuários da internet representa outro tipo de desafio para a seleção de uma amostra probabilística. Para lidar com os problemas (i) e (ii), sugere-se utilizar técnicas de amostragem de cadastros múltiplos e o método de captura e recaptura. Este último, assume que em um intervalo de tempo Δ, considerado negligível, e identificado como o tempo $t \pm \Delta$, a população de lojas virtuais não muda, tendo o mesmo tamanho. Imagina-se ainda que o problema (iii) esteja relacionado com uma outra fonte de variação: a diferença entre os hardwares utilizados nas buscas. Para investigar essa última possibilidade, executamos simultaneamente o mesmo código de *web scraping* em dois computadores localizados em residências distintas, porém na mesma cidade, e utilizando endereços IP diferentes.

Quadro 2 - Dados de simulação de *web scraping* simultânea com dois computadores

PC	Características	Total de Registros	Hr. final	Quantidade de registros diferentes			Quantidade de lojas únicas			
				Mercado Livre	Google Shopping	Bing Shopping	Mercado Livre	Google Shopping	Bing Shopping	Dentre todas as plataformas
1	Notbook Acer, Processador Intel Core i5, 8GB de RAM	281	18:48	4	4	4	12	39	100	140
2	Notbook Dell Inspiron 15, Processador Intel Core i3, 8GB DE RAM	282	18:49	4	4	4	12	37	100	139

Fonte: Elaboração da autora (2024).

O objetivo foi verificar se, em um mesmo instante de tempo, as plataformas fornecem resultados idênticos para uma mesma consulta, independentemente da localização dos computadores dentro da mesma cidade.

A simulação foi realizada em três plataformas: Google, Bing e Mercado Livre. Os dois algoritmos foram iniciados às 17h13 do dia 1º de março de 2024. No computador 1, a simulação retornou 281 resultados, sendo concluída às 18h48, enquanto no computador 2 foram obtidos 282 resultados, com término às 18h49. Em relação às discrepâncias, no computador 1 foram encontrados dois anúncios ausentes nos resultados do computador 2. Da mesma forma, no computador 2 também foram registrados dois anúncios que não apareceram na simulação do computador 1, totalizando uma diferença de quatro anúncios distintos dentre as simulações. No conjunto geral, foram identificados 12 anúncios diferentes ao considerar os resultados únicos de cada plataforma. Adicionalmente, a análise revelou que a quantidade de lojas únicas registradas para cada plataforma foi semelhante em ambas as simulações. No total, o computador 1 identificou 140 lojas únicas, enquanto o computador 2 registrou 139 lojas únicas.

Embora esse experimento seja simples e suas conclusões limitadas, para efeito desta tese seu resultado é lido como sugestivo de que a eventual variação entre hardware, dentro de um mesmo tempo t, é negligível. Isso nos leva a identificar que os preços online não mudam com base no endereço IP que se conecta à plataforma ou site, nem por hábitos de navegação persistentes, e que as lojas virtuais têm um preço online único para cada produto, independentemente da localização do comprador.

Desta forma, resumindo em uma visão macro, é possível imaginar que existe uma população de lojas virtuais dispersa entre plataformas, dentro do universo da web, em um tempo t, sendo possível que informações de preços dos seus produtos, bem como os meios de negociação e pagamento, estejam hospedadas em diferentes tipos de páginas pertencentes aos ambientes comerciais virtuais, onde esses preços são apresentados, organizados e ordenados em estruturas e maneiras diferentes, de acordo com cada página. Levando-se em consideração esta concepção, é possível adotar o seguinte conjunto de notações para a construção dos parâmetros de interesse e respectivos estimadores relacionados à população-alvo:

• $\mathcal{P}(t)$ é o conjunto de P_t produtos utilizados na definição da população de preços das lojas virtuais em um tempo t: $\mathcal{P}(t) = \{1, \dots, i, \dots, P_t\}$;

• $\mathcal{R}(t)$ é a população de R_t plataformas consideradas na definição de população de preços de lojas virtuais em um tempo t: $\mathcal{R}(t) = \{1, \dots, r, \dots, R_t\}$;

• $\mathcal{V}_i^r(t)$ é o conjunto das V_{it}^r lojas virtuais que comercializam o produto $i \in \mathcal{P}(t)$ na plataforma $r \in \mathcal{R}(t)$, no tempo t: $\mathcal{V}_i^r(t) = \{1,2,\ldots,j,\ldots,\mathrm{V}_{it}^r\}$;

• $\mathcal{V}_i(t)$ é a população das V_{it} lojas que vendem o produto $i \in \mathcal{P}(t)$ em alguma das plataformas $r \in \mathcal{R}(t)$, no tempo t, dado que $\mathcal{V}_i(t) = \bigcup_{r \in R} \mathcal{V}_i^r(t)$ e $\mathcal{V}_i(t) = \{1, \ldots, j, \ldots, V_{it}\}$;

• $\mathcal{V}(t) = \bigcup_{i \in \mathcal{P}} \mathcal{V}_i(t)$ é o conjunto de todas as V_t lojas virtuais que comercializam pelo menos um produto $i \in \mathcal{P}(t)$, em pelo menos uma das plataformas $r \in \mathcal{R}(t)$: $\mathcal{V}(t) = \{1, 2, \ldots, j, \ldots, V_t\}$, sendo que $\mathcal{V}_i(t) \subset \mathcal{V}(t)$;

• $y_{ij}(t)$ é o preço do produto $i \in \mathcal{P}(t)$ vendido no tempo t, na loja virtual $j \in \mathcal{V}_i(t)$, logo, é o preço da loja j que vende o produto i.

Levando em consideração que essas definições pressupõem que cada loja virtual oferta somente um preço para cada produto $i \in \mathcal{P}(t)$, em um tempo t, para melhor ilustrar a estrutura de uma população $\mathcal{V}(t)$ de lojas virtuais, o Quadro 3 considera o caso de um cadastro $\mathcal{P}(t)$ de produtos de tamanho $P_t = 3$, definidos em cada linha. Considere ainda que a população $\mathcal{R}(t)$ tem um tamanho $R_t = 3$, com suas unidades representadas por cada coluna da tabela. Como cada plataforma $r \in \mathcal{R}(t)$ possui uma população $\mathcal{V}_i^r(t)$ de lojas virtuais, em um tempo t, o Quadro 3 utiliza cores distintas para identificar cada loja e destacar sua participação em cada plataforma.

Neste caso, a população $\mathcal{V}(t)$ de lojas virtuais é composta por três lojas que vendem o produto $i = 1$: as lojas 1, 3 e 5; quatro lojas que vendem o produto $i = 2$: as lojas 1, 3, 4 e 5; e três lojas que vendem o produto $i = 3$: as lojas 2, 3 e 5. Portanto, temos que a população $\mathcal{V}(t)$ de lojas virtuais é composta por cinco unidades: $\mathcal{V}(t) = \{1,2,3,4,5\}$, sendo que $\mathcal{V}_1(t) = \{1,3,5\}$; $\mathcal{V}_2(t) = \{1,3,4,5\}$; $\mathcal{V}_3(t) = \{2,3,5\}$; $\mathcal{V}_1^1(t) = \{1,3,5\}$, $\mathcal{V}_1^2(t) = \{1,5\}$, $\mathcal{V}_1^3(t) = \{1,3\}$; $\mathcal{V}_2^1(t) = \{3,4,5\}$, $\mathcal{V}_2^2(t) = \{1,3,4,5\}$, $\mathcal{V}_2^3(t) = \{1,3,5\}$; $\mathcal{V}_3^1(t) = \{3\}$, $\mathcal{V}_3^2(t) = \{5\}$, $\mathcal{V}_3^3(t) = \{2,3,5\}$.

Quadro 3 - Ilustração da população de lojas virtuais.

CADASTRO $\mathcal{P}$ (t) DE PRODUTOS	CADASTRO $\mathcal{R}(t)$ DE PLATAFORMAS														
	r =1					r = 2					r = 3				
	$\mathcal{V}_i^1(t)$					$\mathcal{V}_i^2(t)$					$\mathcal{V}_i^3(t)$				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	y_{11}	-	y_{13}	-	y_{15}	y_{11}	-	-	-	y_{15}	y_{11}	-	y_{13}	-	-
2	-	-	y_{23}	y_{24}	y_{25}	y_{21}	-	y_{23}	y_{24}	y_{25}	y_{21}	-	y_{23}	-	y_{25}
3	-	-	y_{33}	-	-	-	-	-	-	y_{35}	-	y_{32}	y_{33}	-	y_{35}

Fonte: Elaboração da autora (2024).

Perceba que as colunas do Quadro 3 são divididas em sub colunas que sinalizam a existência da loja virtual $j \in \mathcal{V}(t)$ na população $\mathcal{V}_i^r(t)$ das respectivas plataformas $r \in \mathcal{R}(t)$, através dos preços $y_{ij}(t)$ ofertados por lojas $j \in \mathcal{V}_i(t)$. Por outro lado, "-" indica que não existe preço para o cruzamento entre o produto $i \in \mathcal{P}(t)$, na loja $j \in \mathcal{V}_i(t)$; anunciada pela plataforma $r \in \mathcal{R}(t)$.

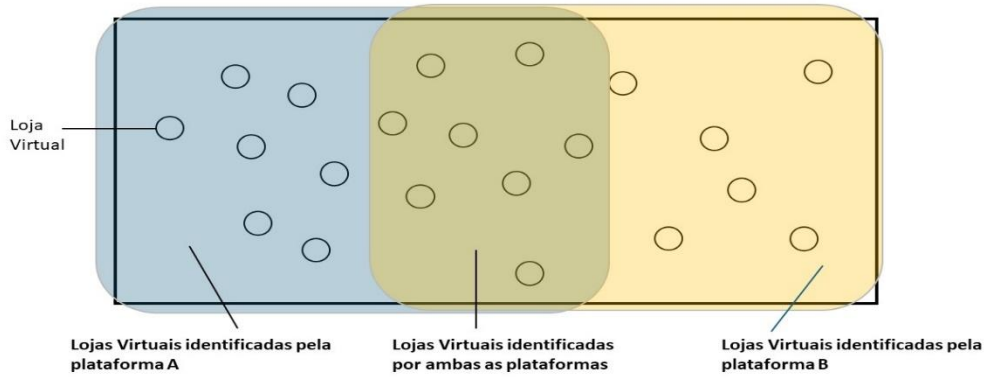
Deste modo, no Quadro 3 pode-se perceber que todas as plataformas anunciam pelo menos uma loja virtual que vende o produto $i \in \mathcal{P}(t)$, pois estamos assumindo que a definição da população $\mathcal{R}(t)$ de plataformas é direcionada para abarcar o tipo do produto que se refere um índice de preços em questão. Em outras palavras, sendo $\mathcal{V}^r(t) = \bigcup_{i\in\mathcal{P}} \mathcal{V}_i^r(t)$ a união de todas as lojas que podem ser anunciadas na plataforma $r \in \mathcal{R}(t)$ para qualquer $i \in \mathcal{P}(t)$, apesar de não termos como garantir que todas as lojas $j \in \mathcal{V}^r(t)$ vendam todos os produtos $i \in \mathcal{P}(t)$, não existe a possibilidade de um produto $i \in \mathcal{P}(t)$, não ser vendido em nenhuma das lojas $j \in \mathcal{V}^r(t)$, ou seja, dado $i \in \mathcal{P}(t)$, não é possível que $\mathcal{V}_i^r(t) = \emptyset$ e, consequentemente, também não é possível que uma plataforma não venda nenhum produto $i \in \mathcal{P}(t)$ ($\mathcal{V}^r(t) = \emptyset$), nem que $\mathcal{V}_i(t) = \emptyset$, já que estamos assumindo que todos os produtos $i \in \mathcal{P}(t)$ são vendidos na web.

Além disso, pode-se perceber ainda no Quadro 3 que existem interseções de informações entre as populações $\mathcal{V}_i^r(t)$, para $i \in \mathcal{P}(t)$ e $r \in \mathcal{R}(t)$, provenientes do fato de que uma mesma loja virtual pode anunciar um mesmo produto em diferentes plataformas. Desta forma, é possível, por meio de *web scraping*, capturar um preço que corresponda ao mesmo produto, na mesma loja e com o mesmo valor. Por ser duplicidade de informação, esses dados devem ser identificados e não podem ser contabilizados duas vezes. No Quadro 3, podemos ver esse caso através do preço $y_{15}(t)$, que se refere ao produto $i = 1$ e à loja virtual $j = 5$. Esse

preço pode ser capturado no tempo t através da plataforma $r = 1$ e através da plataforma $r = 2$.

Uma forma de visualizar melhor uma população $\mathcal{V}_i(t)$ de lojas virtuais, dado um produto $i \in \mathcal{P}(t)$, e suas interseções, é através da Figura 2 abaixo. Essa ilustração considera a existência de somente duas plataformas $r \in \mathcal{R}(t)$ na web, as quais estão representadas pelos retângulos que envolvem os círculos, que são as representações das lojas virtuais $j \in \mathcal{V}_i(t)$ e, consequentemente, dos seus preços. Portanto, podemos perceber que a área azul representa as lojas que, no tempo t, estão anunciando vendas do produto $i \in \mathcal{P}(t)$ somente através da plataforma A, e a área laranja representa as lojas que, no tempo t, estão anunciando vendas do mesmo produto $i \in \mathcal{P}(t)$ somente através da plataforma B. O retângulo marrom, por sua vez, indica as lojas que, no tempo t, anunciam nas duas plataformas.

Figura 2 – População de Lojas Virtuais que vendem o produto i no tempo t: $\mathcal{V}_i(t)$



Fonte: Elaboração da autora (2023).

Na prática, a captura de dados por *web scraping* também pode acarretar coleta de informações em duplicidade em uma mesma plataforma, as quais também devem ser excluídas do banco de dados destinado à inferência. Essa duplicidade pode acontecer quando um resultado de pesquisa em uma plataforma de busca retorna anúncios iguais ou, ainda, quando alguma(s) característica(s) do produto é/são negligenciada(s). Exemplo: o produto Apple iPhone 14 Pro Max 512GB, pode aparecer com o mesmo preço mais de uma vez, quando se trata do mesmo aparelho em cores distintas. Assim, cabe ao planejamento da pesquisa definir o agregado elementar dos produtos que são considerados, ou seja, escolher o nível de detalhamento de características dos produtos do cadastro utilizado, já que em uma situação na qual a cor do

aparelho não é levada em conta, aparelhos de cores distintas não serão considerados como produtos distintos, o que implicaria na observação de repetição da mesma informação.

Ainda na prática, é possível que uma busca, por um produto $i \in \mathcal{P}(t)$, em uma plataforma $r \in \mathcal{R}(t)$, que objetiva buscar ou comparar preços, em um tempo t, retorne anúncios em que a venda está relacionada a um *marketing place*, e não à loja virtual que está viabilizando suas vendas através desse *marketing place*. Portanto, é comum visualizar em uma mesma plataforma de busca de preços, informações de preços diferentes para um mesmo produto, referentes a um mesmo *marketing place*. Da mesma forma, pode-se encontrar anúncios de um mesmo *marketing place* ofertando preços distintos (ou iguais) para um determinado produto $i \in \mathcal{P}(t)$ em diferentes plataformas $r \in \mathcal{R}(t)$. Isso evidencia que ao fixarmos um produto $i \in \mathcal{P}(t)$ e uma loja virtual $j \in \mathcal{V}_i^r(t)$, o anúncio do preço $y_{ij}(t)$ deve ser identificado pelo link fornecido pela plataforma $r \in \mathcal{R}(t)$.

Além disso, é importante destacar que um mesmo link pode ser compartilhado entre diferentes plataformas de busca de preços, mas as informações contidas nele permanecem consistentes em um dado tempo t, independentemente da plataforma onde é exibido. Sem essa abordagem, seria inviável incluir lojas virtuais presentes em *marketplaces* na inferência sobre a população de lojas virtuais, o que representa um desafio significativo para muitas pesquisas atuais.

3.3 Parâmetros utilizados em Índice de Preços para Lojas Virtuais

Os índices de preços têm como objetivo principal medir a variação nos preços de um conjunto de produtos dentro de um nível específico de agregado elementar. Para estimar essa variação, as médias de preços de um conjunto de produtos podem ser utilizadas em fórmulas de índices, sejam elas ponderadas, não ponderadas, encadeadas ou de preços unitários.

O Índice de Laspeyres Bilateral Ponderado, por exemplo, é atualmente utilizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) para analisar os preços das lojas físicas (IBGE, 2023a). Esse índice calcula a média ponderada das variações de preços dos produtos $i \in \mathcal{P}(t)$, comparando o período base (tempo 0) com um período atual (tempo t), utilizando as médias aritméticas dos preços de cada produto. A expressão básica de um Índice de Preços baseado em Laspeyres, pode ser descrita da seguinte forma:

$$I^{0,t} = \frac{\sum_{i \in \mathcal{P}(t)} \bar{Y}_i(t) W_i(0)}{\sum_{i \in \mathcal{P}(t)} \bar{Y}_i(0) W_i(0)}, \quad (1)$$

em que:

- $\bar{Y}_i(t)$ é o preço médio, no tempo t, do item $i \in \mathcal{P}(t)$, calculado como $\bar{Y}_i(t) = \frac{\sum_{j \in \mathcal{F}(t)} y_{ij}(t)}{F_{it}}$. Aqui $y_{ij}(t)$ representa o preço do item i, na loja j, no tempo t; $\mathcal{F}(t)$ representa o cadastro de lojas físicas considerado; e F_{it} é o número total de lojas físicas pertencentes ao cadastro $\mathcal{F}(t)$ que vendem o produto i, no tempo t;
- $\bar{Y}_i(0)$ é o preço médio do item $i \in \mathcal{P}(t)$, em um tempo base, nas lojas pertencentes a $\mathcal{F}(t)$;
- $W_i(0)$ é o peso do item $i \in \mathcal{P}(t)$, geralmente baseado nas quantidades consumidas ou vendidas no tempo base; sendo que $\sum_{i \in \mathcal{P}(t)} W_i(0) = 1$.

Como no ambiente virtual não dispomos de informações sobre as quantidades consumidas ou vendidas (ou seja, sobre $W_i(0)$), inicialmente calcularemos um índice de preços $I_{0,t}^{\mathcal{V}}$ que permitirá inferir sobre a variação dos preços praticados por uma população de lojas virtuais, com base na comparação das médias aritméticas dos preços entre o período base (tempo 0) e o período t. O cadastro de produtos utilizado será definido por um agregado elementar, ou seja, um conjunto fixo de características que descrevem os produtos, como categoria, marca, entre outros.

Assim, o índice de preços básicos virtuais $I_{0,t}^{\mathcal{V}}$ será denotado da seguinte forma:

$$I_{0,t}^{\mathcal{V}} = \frac{\sum_{i \in \mathcal{P}(t)} \bar{Y}_i(t)}{\sum_{i \in \mathcal{P}(t)} \bar{Y}_i(0)}, \quad (2)$$

Uma vez que o método para estimar a média $\bar{Y}_i(0)$ de preços no tempo base é a mesma utilizada para inferir sobre a média $\bar{Y}_i(t)$ de preços no tempo t, o parâmetro que desejamos definir e estimar com base na realidade da população de lojas virtuais será somente $I_t^{\mathcal{V}} = \sum_{i \in \mathcal{P}(t)} \bar{Y}_i(t)$: o somatório das médias de preços dos itens $i \in \mathcal{P}(t)$ nas lojas $j \in \mathcal{V}(t)$, no tempo t. Para isso, foi necessário lidar com a complexidade das informações das lojas virtuais, ilustrada no Quadro 3, por meio da definição de variáveis apropriadas para integrar a expressão de $I_t^{\mathcal{V}}$. Nesse contexto, como a população $\mathcal{R}(t)$ de plataformas é considerada conhecida,

podemos visualizar a população $\mathcal{V}_i(t)$ de lojas que comercializam o produto $i \in \mathcal{P}(t)$ e, consequentemente, a população $\mathcal{V}(t)$ de lojas virtuais, a partir dessa população.

Assim, se considerássemos somente uma plataforma $r \in \mathcal{R}(t)$, ou seja, tivéssemos como população-alvo a população $\mathcal{V}_i^r(t)$ de lojas virtuais que comercializam o produto $i \in \mathcal{P}(t)$ na plataforma $r \in \mathcal{R}(t)$, no tempo t, dado que essa população tem tamanho V_{it}^r, a média $\bar{Y}_i^r(t)$ de preços do produto i na plataforma $r \in \mathcal{R}(t)$ pode ser definida da seguinte forma:

$$\bar{Y}_i^r(t) = \frac{1}{\mathrm{V}_{it}^r} \sum_{j \in \mathcal{V}_i^r(t)} y_{ij}(t), \tag{3}$$

Por outro lado, se o objetivo for estimar a média de preços de um determinado produto i em uma população-alvo $\mathcal{V}_i(t)$ de lojas virtuais que anunciam suas vendas em pelo menos uma das plataformas pertencentes $\mathcal{R}(t)$, é necessário inicialmente definir um fator de multiplicidade (Mecatti, 2007):

- $m_{ij}(t)$: é o número de plataformas que identificam a loja virtual $j \in \mathcal{V}(t)$, que vende o produto $i \in \mathcal{P}(t)$, no tempo t. $m_{ij}(t)$ é uma constante tal que $1 \leq m_{ij}(t) \leq R_t$.

Dessa forma, temos que o parâmetro $Y_i(t)$ do total de preços para um dado preço $i \in \mathcal{P}(t)$ pode ser dado por:

$$Y_i(t) = \sum_{r \in \mathcal{R}(t)} \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)}, \tag{4}$$

Logo, o parâmetro da média de preços, que temos interesse, pode ser descrito como:

$$\bar{Y}_i(t) = \frac{1}{V_{it}} \sum_{r \in \mathcal{R}(t)} \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)}, \tag{5}$$

em que:

$$V_{it} = \sum_{r \in \mathcal{R}(t)} \sum_{j \in \mathcal{V}_i^r(t)} \frac{1}{m_{ij}(t)}. \tag{6}$$

As definições acima partem da suposição de que o universo de lojas virtuais que vende o produto i, em um dado tempo t, cujo tamanho é V_{it}, é completamente coberto pelos resultados de buscas realizadas através das R_t plataformas, contempladas em $\mathcal{R}(t)$. A partir disso, é possível definir o total populacional dos preços médios calculados para todos os produtos pertencentes a $\mathcal{P}(t)$:

$$I_t^{\mathcal{V}} = \sum_{i \in \mathcal{P}(t)} \bar{Y}_i(t) = \sum_{i \in \mathcal{P}(t)} \sum_{r \in \mathcal{R}(t)} \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{V_{it} m_{ij}(t)} \tag{7}$$

É importante notar que a soma de V_{it}^r, o total de lojas virtuais que vendem o produto ou serviço $i \in \mathcal{P}(t)$, no tempo t, identificados pela plataforma $r \in \mathcal{R}(t)$ não equivale a V_{it}. Como plataformas distintas podem identificar uma mesma loja virtual, tem-se tipicamente que: $\sum_{r=1}^{R_t} \mathrm{V}_{it}^r \geq V_{it}$.

Na próxima seção apresentaremos um plano amostral capaz de inferir sobre os parâmetros aqui descritos.

3.4 Plano Amostral e Estimadores que envolvem médias de preços das lojas virtuais.

Consideramos um plano amostral em dois estágios:

1º Estágio: Seleção de uma amostra aleatória simples, denotada por $S_{\mathcal{P}}(t)$, extraída do cadastro de produtos $\mathcal{P}(t)$, onde cada produto i é selecionado com uma probabilidade constante, dada por:

$$\pi_i(t) = \frac{n_{\mathcal{P}}(t)}{P_t};$$

2º Estágio: Processo de captura e recaptura, assumindo um modelo de amostragem de Bernoulli, com parâmetro $\pi_{r|i}(t)$, para a seleção de preços das lojas virtuais $j \in \mathcal{V}(t)$ que vendem o produto $i \in S_{\mathcal{P}}(t)$. A coleta é realizada por um algoritmo de *web scraping* capaz de capturar todos os anúncios resultantes de cada busca por $i \in S_{\mathcal{P}}(t)$ em uma plataforma $r \in \mathcal{R}(t)$, de forma independente. Assim, será associada uma probabilidade constante $\pi_{r|i}(t)$ a cada preço capturado na plataforma r, relacionado ao produto i , sendo esta probabilidade derivada

do próprio modelo de amostragem de Bernoulli assumido para cada plataforma, dado um produto.

A adoção do modelo de amostragem de Bernoulli para representar os resultados de *web scraping* se justifica pelo fato de ele permitir um tamanho amostral variável, característica essencial para lidar com a dinâmica dos dados extraídos. Em cada captura, realizada em intervalos curtos de tempo para um determinado produto $i \in S_{\mathcal{P}}(t)$ na plataforma $r \in \mathcal{R}(t)$, o número de lojas obtidas pode variar significativamente, o que inviabiliza a adoção de um modelo de amostragem com tamanho fixo. Além disso, a amostragem de Bernoulli captura a natureza binária do processo de coleta, onde cada tentativa pode resultar na obtenção ($\mathcal{I}$ =1) ou não ($\mathcal{I}$ =0) de determinada informação. Esse modelo também utiliza amostras independentes, o que está alinhado com a natureza do nosso problema, pois estamos assumindo que os resultados de *web scraping* são independentes entre si, ou seja, cada coleta não exerce influência direta sobre as coletas anteriores.

Contudo, ao assumir que os resultados de captura e recaptura através de *web scraping*, em uma plataforma específica, podem ser descritos como réplicas independentes de amostras de Bernoulli, com probabilidade de seleção $\pi_{r|i}(t)$ e de não seleção $1 - \pi_{r|i}(t)$, nos deparamos com a limitação de não conhecer diretamente estas probabilidades, devido à ausência de um cadastro completo e acessível de preços. Nesse cenário, o método de captura e recaptura é significativamente útil e necessário, pois permite estimar o total V_{it}^{r} de lojas virtuais anunciadas na plataforma $r \in \mathcal{R}(t)$ que vendem o produto $i \in S_{\mathcal{P}}(t)$, no tempo t.

Para isso, consideraremos duas amostras independentes: a captura $S_{i\_1}^{r}(t)$, de tamanho $\mathrm{v}_{i\_1}^{r}(t)$; e a recaptura $S_{i\_2}^{r}(t)$, de tamanho $\mathrm{v}_{i\_2}^{r}(t)$ (realizada imediatamente após a captura). Neste processo, as capturas e recapturas são representadas pelas seguintes variáveis indicadoras:

$$\mathcal{I}_{ij\_1}^{r}(t) = \begin{cases} 1, \text{caso a loja virtual } j \in S_{i\_1}^{r}(t); \\ 0, \text{caso contrário.} \end{cases}$$

e

$$\mathcal{I}_{ij\_2}^{r}(t) = \begin{cases} 1, \text{caso a loja virtual } j \in S_{i\_2}^{r}(t); \\ 0, \text{caso contrário.} \end{cases}$$

Com estas variáveis previamente definidas, podemos estabelecer a variável indicadora de interseção entre as amostras, como segue:

$$\mathcal{I}^r_{ij\_12}(t) = \mathcal{I}^r_{ij_1}(t) * \mathcal{I}^r_{ij_2}(t).$$

Uma vez que os resultados capturados por *web scraping* não fornecem controle sobre os seus tamanhos, para inferir sobre o universo $j \in \mathcal{V}^r_i(t)$, dado $i \in S_{\mathcal{P}}(t)$, consideraremos que as amostras $S^r_{i\_1}(t)$ e $S^r_{i\_2}(t)$ serão selecionadas com a mesma probabilidade de inclusão, dada por $\pi_{r|i} = p\big(\mathcal{I}^r_{ij\_1}(t) = 1\big) = p\big(\mathcal{I}^r_{ij\_2}(t) = 1\big)$, para todo $j \in \mathcal{V}^r_i(t)$. Estas amostras serão como réplicas independentes de uma amostra $S^r_i(t)$, gerada por um processo de amostragem de Bernoulli com probabilidade $\pi_{r|i}(t)$. Neste contexto, se V^r_{it} fosse conhecido, poderíamos definir:

$$\pi_{r|i}(t) = \frac{E_{BE}(\mathrm{v}^r_i(t))}{V^r_{it}}, \tag{8}$$

em que $\mathrm{v}^r_i(t)$ representa o tamanho aleatório da amostra $S^r_i(t)$, cuja esperança em relação ao modelo adotado está denotada em (8) como $E_{BE}(\mathrm{v}^r_i(t))$.

Neste caso, as lojas virtuais com $\mathcal{I}^r_{ij\_1}(t) = 0$, $\mathcal{I}^r_{ij\_2}(t) = 0$ e $\mathcal{I}^r_{ij\_12} = 0$ não são observadas, portanto, temos acesso apenas às quantidades:

$$\mathrm{v}^r_{i\_1}(t) = \sum_{j \in \mathcal{V}^r_i(t)} \mathcal{I}^r_{ij\_1}(t)\,; \qquad \mathrm{v}^r_{i\_2}(t) = \sum_{j \in \mathcal{V}^r_i(t)} \mathcal{I}^r_{ij\_2}(t)\,; \qquad \mathrm{v}^r_{i\_12}(t) = \sum_{j \in \mathcal{V}^r_i(t)} \mathcal{I}^r_{ij\_12}(t),$$

em que $\mathrm{v}^r_{i\_12}(t)$ representa o número de observações presentes em ambas as amostras $S^r_{i\_1}(t)$ e $S^r_{i\_2}(t)$, ou seja, a interseção entre as duas capturas.

Com base nestas definições, considerando que $E_{BE}(\mathrm{v}^r_i(t)) = V^r_{it}\,\pi_{r|i}(t)$, o estimador de $\pi_{r|i}(t)$ pode ser baseado em um plano amostral de captura-recaptura Bernoulli (CRB), podendo ser expresso da seguinte forma:

$$\hat{\pi}_{r|i}(t) = \frac{\hat{E}_{BE}\big(\mathrm{v}^r_i(t)\big)}{\hat{V}^r_{it}}, \tag{9}$$

em que:

- $\hat{E}_{BE}(\mathrm{v}_i^r(t))$ é a estimativa da esperança da variável aleatória $\mathrm{v}_i^r(t)$, dada por:

$$\hat{E}_{BE}(\mathrm{v}_i^r(t)) = \frac{\left(\mathrm{v}_{i\_1}^r(t) + \mathrm{v}_{i\_2}^r(t)\right)}{2}; \tag{10}$$

- $\hat{V}_{it}^r$ é o estimador do tamanho populacional de lojas que vendem o produto $i \in S_{\mathcal{P}}(t)$ na plataforma $r \in \mathcal{R}(t)$, no tempo t, com base no plano amostral de captura e recaptura, conforme expresso em (11).

$$\hat{V}_{it}^r = \frac{\mathrm{v}_{i\_1}^r(t)\mathrm{v}_{i\_2}^r(t)}{\mathrm{v}_{i\_12}^r(t)}. \tag{11}$$

Portanto, com estas definições temos que $\hat{\pi}_{r|i}(t)$ é composta pela seguinte razão:

$$\hat{\pi}_{r|i}(t) = \frac{\left(\mathrm{v}_{i\_1}^r(t) + \mathrm{v}_{i\_2}^r(t)\right)\ \mathrm{v}_{i\_12}^r(t)}{2\ \mathrm{v}_{i\_1}^r(t)\mathrm{v}_{i\_2}^r(t)}. \tag{12}$$

Considerando ainda o plano de amostragem CRB, definiremos o estimador da média de preços de um produto $i \in S_{\mathcal{P}}(t)$, em uma plataforma $r \in \mathcal{R}(t)$ (parâmetro definido em (3)), da seguinte maneira:

$$\hat{\bar{Y}}_i^r(t) = \frac{1}{2}\left[\hat{\bar{Y}}_{i\_1}^r(t) + \hat{\bar{Y}}_{i\_2}^r(t)\right], \tag{13}$$

em que $\hat{\bar{Y}}_{i\_1}^r(t)$ e $\hat{\bar{Y}}_{i\_2}^r(t)$ são, respectivamente, médias derivadas das amostras $S_{i\_1}^r(t)$ (captura) e $S_{i\_2}^r(t)$ (recaptura), consideradas réplicas do estimador da média com base em um plano amostral de Bernoulli, expresso por:

$$\hat{\bar{Y}}_i^r(t)^{BE} = \sum_{j \in S_i^r(t)} \frac{y_{ij}(t)}{\mathrm{v}_i^r(t)}, \tag{14}$$

em que:

$$\mathrm{v}_i^r(t) = \sum_{j \in \mathcal{V}_i^r(t)} \mathcal{I}_{ij}^r(t);$$

e

$$\mathcal{I}_{ij}^{r}(t) = \begin{cases} 1, \text{caso a loja virtual } j \in S_i^r(t); \\ 0, \text{caso contrário.} \end{cases}$$

Ao considerar que $\hat{\bar{Y}}_{i\_1}^{r}(t)$ e $\hat{\bar{Y}}_{i\_2}^{r}(t)$ são réplicas de $\hat{\bar{Y}}_i^r(t)^{BE}$, temos que o estimador descrito em (13) pode ser avaliado estatisticamente com base nas propriedades do estimador (14), de forma que:

$$E\left(\hat{\bar{Y}}_i^r(t)\right) = E\left(\frac{1}{2}\left[\hat{\bar{Y}}_{i\_1}^{r}(t) + \hat{\bar{Y}}_{i\_2}^{r}(t)\right]\right) = \frac{1}{2}E\left(2\hat{\bar{Y}}_i^r(t)^{BE}\right) = \ (E_{BE}\left(\hat{\bar{Y}}_i^r(t)^{BE}\right).$$

Adicionalmente, pode-se demonstrar que o estimador $\hat{\bar{Y}}_i^r(t)^{BE}$ é não viesado para $\bar{Y}_i^r(t)$ com base nas propriedades fundamentais da amostragem de Bernoulli condicional, considerando $E_{BE}\left(\mathrm{v}_i^r(t)\right) = \mathrm{v}_i^r(t) = \mathrm{v}_{i0}^r(t)$ e assumindo $V_{it}^r = \hat{V}_{it}^r$. Isto pode ser descrito da seguinte forma:

$$\begin{aligned} E_{BE}\left(\hat{\bar{Y}}_i^r(t)^{BE} \middle|\, \mathrm{v}_i^r(t) = \mathrm{v}_{i0}^r(t), V_{it}^r = \hat{V}_{it}^r\right) &= \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{\mathrm{v}_{i0}^r(t)} E_{BE}\left(\mathcal{I}_{ij}^r(t)\right) \\ &= \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{\mathrm{v}_{i0}^r(t)} \pi_{r|i}(t) = \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{V_{it}^r} = \bar{Y}_i^r(t). \end{aligned}$$

Além disso, com base em Särndal, Swensson e Wretman (1992, p. 259), a variância condicional de $\hat{\bar{Y}}_i^r(t)^{BE}$ pode ser dada pela seguinte expressão:

$$\begin{aligned} &Var_{BE}\left(\hat{\bar{Y}}_i^r(t)^{BE} \middle| \mathrm{v}_i^r(t) = \mathrm{v}_{i0}^r(t), V_{it}^r = \hat{V}_{it}^r\right) \\ &\quad = \frac{1}{4}\Big[Var_{BE}\left(\hat{\bar{Y}}_{i\_1}^r(t) \middle| \mathrm{v}_i^r(t) = \mathrm{v}_{i0}^r(t), V_{it}^r = \hat{V}_{it}^r\right) \\ &\quad + Var_{BE}\left(\hat{\bar{Y}}_{i\_2}^r(t) \middle| \mathrm{v}_i^r(t) = \mathrm{v}_{i0}^r(t), V_{it}^r = \hat{V}_{it}^r\right)\Big] \\ &\quad = \frac{1}{2} Var_{BE}\left(\hat{\bar{Y}}_i^r(t)^{BE} \middle| \mathrm{v}_i^r(t) = \mathrm{v}_{i0}^r(t), V_{it}^r = \hat{V}_{it}^r\right) \\ &\quad = \frac{1}{2}\left(1 - \frac{\mathrm{v}_{i0}^r(t)}{V_{it}^r}\right)\frac{\sigma_{y_i}^2(t)}{\mathrm{v}_{i0}^r(t)} = \frac{1}{2}\left(\frac{\sigma_{y_i}^2(t)}{\mathrm{v}_{i0}^r(t)} - \frac{\sigma_{y_i}^2(t)}{V_{it}^r}\right), \end{aligned} \tag{15}$$

em que $\sigma_{y_i}^2(t)$ é a variância do preço do produto $i \in S_{\mathcal{P}}(t)$, na população $\mathcal{V}_i^r(t)$, no tempo t, dado $r \in \mathcal{R}(t)$.

A expressão (15) baseia-se no fato de que $Var\left(\hat{\bar{Y}}_{i\_1}^{r}(t)\right) = Var\left(\hat{\bar{Y}}_{i}^{r}(t)^{BE}\right) = Var(\hat{\bar{Y}}_{i\_2}^{r}(t))$. Sabendo disso, temos ainda que a variância da soma das médias de preços obtidas na captura e na recaptura, dividida por 4, equivale a variância de $\hat{\bar{Y}}_{i}^{r}(t)^{BE}$. Além disso, sabe-se que $Var_{BE}\left(\hat{\bar{Y}}_{i}^{r}(t)^{BE}\right)$ corresponde ao dobro da variância de uma captura para um produto i na plataforma r, o que pode ser demonstrado da seguinte maneira:

$$Var\left(\hat{\bar{Y}}_{i}^{r}(t)\right) = Var\left\{\frac{1}{2}\left[\hat{\bar{Y}}_{i\_1}^{r}(t) + \hat{\bar{Y}}_{i\_2}^{r}(t)\right]\right\} = \frac{1}{4}Var_{BE}\left\{\left[\hat{\bar{Y}}_{i\_1}^{r} + \hat{\bar{Y}}_{i\_2}^{r}\right]\right\} = \frac{1}{2}Var_{BE}\left(2\,\hat{\bar{Y}}_{i}^{r}(t)^{BE}\right)$$
$$= \frac{1}{2}Var_{BE}\left(\hat{\bar{Y}}_{i}^{r}(t)^{BE}\right).$$

Ademais, como lidamos com grandes amostras, assumimos em (15) um fator de correção para população finita aproximadamente igual a 1.

Com isso, a variância de $\hat{\bar{Y}}_{i}^{r}(t)^{BE}$ pode ser estimada de forma não viesada por:

$$\widehat{Var}_{BE}\left(\hat{\bar{Y}}_{i}^{r}(t)^{BE} | \mathrm{v}_{i}^{r}(t) = \mathrm{v}_{i0}^{r}(t), V_{it}^{r} = \hat{V}_{it}^{r}\right) = \frac{1}{2}\left(1 - \frac{\hat{\mathrm{v}}_{i0}^{r}(t)}{\hat{V}_{it}^{r}}\right)\frac{\hat{\sigma}_{y_i}^{2}(t)}{\hat{\mathrm{v}}_{i0}^{r}(t)}$$
$$= \frac{1}{2}\left(\frac{\hat{\sigma}_{y_i}^{2}(t)}{\hat{\mathrm{v}}_{i0}^{r}(t)} - \frac{\hat{\sigma}_{y_i}^{2}(t)}{\hat{V}_{it}^{r}}\right), \tag{16}$$

em que $\hat{\sigma}_{y_i}^{2}(t)$ é uma réplica de:

$$S_{i}^{r}(t)^{2} = \sum_{j \in S_{i}^{r}(t)} \frac{\left(y_{ij}(t) - \hat{\bar{Y}}_{i}^{r}(t)^{BE}\right)^{2}}{\mathrm{v}_{i}^{r}(t) - 1}. \tag{17}$$

Agora, para estimar a média de preços do produto $i \in S_{\mathcal{P}}(t)$ ofertados pelos mercados $j \in \mathcal{V}_i(t)$ anunciados pelo conjunto $\mathcal{R}(t)$ de plataformas, integraremos ao plano amostral proposto a amostragem por cadastros múltiplos (Mecatti, 2007). Esta abordagem se justifica, pois, as interseções entre os resultados das diferentes plataformas para um mesmo produto i podem ser controladas pelo processo de amostragem de cadastros múltiplos, que atribui um fator de multiplicidade $m_{ij}(t)$ para cada par (i, j).

Desta forma, $\bar{Y}_i(t)$ (parâmetro definido na expressão (5)), pode ser descrito como:

$$\hat{\bar{Y}}_i(t) = \frac{1}{2}\left[\hat{\bar{Y}}_{i\_1}(t) + \hat{\bar{Y}}_{i\_2}(t)\right], \tag{18}$$

em que $\hat{\bar{Y}}_{i\_1}(t)$ e $\hat{\bar{Y}}_{i\_2}(t)$ são réplicas da média estimada por um processo de amostragem que envolve cadastros múltiplos, dada por:

$$\hat{\bar{Y}}_i(t)^{CM} = \frac{\hat{Y}_i(t)}{\hat{V}_{it}}, \tag{19}$$

em que $\hat{Y}_i(t)$ é o estimador do total $Y_i(t)$ (Expressão (4)) dos preços das lojas virtuais para um produto $i \in S_{\mathcal{P}}(t)$, considerando as lojas cobertas por um cadastro de plataformas $r \in \mathcal{R}(t)$, em um tempo t. Esse estimador pode ser expresso da seguinte forma:

$$\hat{Y}_i(t) = \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)\hat{\pi}_{r|i}(t)}, \tag{20}$$

e $\hat{V}_{it}$ é o estimador de V_{it} (Expressão (6)), expresso como:

$$\hat{V}_{it} = \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \frac{1}{m_{ij}(t)\hat{\pi}_{r|i}(t)}. \tag{21}$$

Como $\hat{\bar{Y}}_i(t)^{CM}$ é um estimador típico para uma razão populacional, baseado em cadastros múltiplos, com amostragem de Bernoulli aplicado em cada cadastro, pode-se demonstrar com a expansão linear de Taylor (Apêndice A), usada para aproximar $\hat{\bar{Y}}_i(t)^{CM}$ do seu parâmetro, que $\hat{\bar{Y}}_i(t)^{CM}$ é aproximadamente não viesado, com variância dada por:

$$Var_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right) = \frac{1}{V_{it}^2} \sum_{r\in\mathcal{R}(t)} \sum_{j\in V_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \frac{\bar{Y}_i(t)}{m_{ij}(t)}\right]^2 \left(\frac{1}{\pi_{r|i}(t)} - 1\right). \tag{22}$$

Além disso, $Var_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right)$ pode ser estimada de forma aproximadamente não viesada usando:

$$\widehat{Var}_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right) = \frac{\mathcal{I}_i(t)}{\hat{V}_{it}^2} \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \frac{\hat{\bar{Y}}_i(t)^{CM}}{m_{ij}(t)}\right]^2 \frac{1}{\hat{\pi}_{r|i}(t)}\left(\frac{1}{\hat{\pi}_{r|i}(t)} - 1\right), \quad (23)$$

em que $\mathcal{I}_i(t)$ é o indicador de que o produto $i \in \mathcal{P}(t)$ pertence a amostra $S_{\mathcal{P}}(t)$ de produtos, no tempo t, sendo $\mathcal{I}_i(t) = 1$ quando isto ocorre e $\mathcal{I}_i(t) = 1$, caso contrário.

Desta forma, como estamos assumindo que $Var\left(\hat{\bar{Y}}_{i\_1}(t)\right) = Var\left(\hat{\bar{Y}}_i(t)^{CM}\right) = Var(\hat{\bar{Y}}_{i\_2}(t))$, sabe-se que o estimador $\hat{\bar{Y}}_i(t)$ é aproximadamente não viesado, com variância aproximada fornecida pela metade do valor de (22), que pode ser estimada, também de forma aproximadamente não viesada, usando metade do valor de (23). Logo, para chegar até $\widehat{Var}\left(\hat{\bar{Y}}_i(t)\right)$ é preciso somar os estimadores de variância: $\widehat{Var}\left(\hat{\bar{Y}}_{i\_1}(t)\right)$ e $\widehat{Var}(\hat{\bar{Y}}_{i\_2}(t))$, e dividir esta soma por 4 (quatro).

A partir destas definições, finalmente podemos definir na Expressão (24) o estimador do total $I_t^{\mathcal{V}}$ das médias de preços dos produtos $i \in \mathcal{P}(t)$ nas plataformas $r \in \mathcal{R}(t)$, de forma que estamos considerando que foi selecionada uma amostra aleatória simples $S_{\mathcal{P}}(t)$ de produtos no cadastro $\mathcal{P}(t)$, com probabilidade $\pi_i(t) = \frac{n_{\mathcal{P}}(t)}{P_t}$.

$$\hat{I}_t^{\mathcal{V}} = \sum_{i\in\mathcal{P}(t)} \frac{\mathcal{I}_i(t)}{\pi_i(t)} \hat{\bar{Y}}_i(t) = \sum_{i\in\mathcal{P}(t)} \frac{\hat{Y}_i(t)\mathcal{I}_i(t)}{\hat{V}_{it}\pi_i(t)}. \quad (24)$$

É importante ressaltar, que como temos uma amostragem em dois estágios, a probabilidade total $\pi_{ij}^r(t)$ de seleção de um preço específico da loja virtual $j \in \mathcal{V}_i^r(t)$, dado $r \in \mathcal{R}(t)$ e $i \in S_{\mathcal{P}}(t)$, é a multiplicação das probabilidades dos dois estágios, ou seja: $\pi_{ij}^r(t) = \pi_i(t).\hat{\pi}_{r|i}(t)$.

Assim, sob o plano de amostragem em dois estágios, $\hat{I}_t^{\mathcal{V}}$ é um estimador assintoticamente não viesado porque estamos assumindo que as médias estimadas $\hat{\bar{Y}}_i(t)$ de preços dos produtos $i \in S_{\mathcal{P}}(t)$ convergem para as verdadeiras médias populacionais com o aumento do tamanho da amostra (APÊNDICE A), e porque $E\left(\mathcal{I}_i(t)\right) = \pi_i(t)$.

Além disso, assumimos que as esperança de $\hat{I}_t^{\mathcal{V}}$ é igual a esperança de $\hat{I}_t^{\mathcal{V}}$ no plano de amostragem em dois estágios (Särndal, Swensson e Wretman, 1992), combinando amostragem

aleatória simples (AAS) e um plano amostral de cadastros múltiplos (CM), que é igual à esperança, sob uma amostragem aleatória simples, da esperança condicional de uma amostragem por cadastro múltiplo, dada uma amostra no primeiro estágio:

$$E_{2est}(\hat{I}_t^{\mathcal{V}}) = E_{AAS,CM}(\hat{I}_t^{\mathcal{V}}) = E_{AAS}\left(E_{CM}(\hat{I}_t^{\mathcal{V}}|S_{1st})\right).$$

Desta forma, a variância do estimador descrito na Expressão (24), é dada com base na variância para Amostragem em Dois Estágios, ou seja:

$$Var_{2est}(\hat{I}_t^{\mathcal{V}}) = E_{AAS}\left(Var_{CM}(\hat{I}_t^{\mathcal{V}}|S_{1st})\right) + Var_{AAS}\left(E_{CM}(\hat{I}_t^{\mathcal{V}}|S_{1st})\right) \quad (25)$$

em que:

$$\begin{aligned}
E_{AAS}\left(Var_{CM}(\hat{I}_t^{\mathcal{V}}|S_{1st})\right) &= E_{AAS}\left(\frac{{P_t}^2}{n_{\mathcal{P}}(t)^2}\sum_{i\in S_{\mathcal{P}}(t)} Var_{CM}(\hat{\bar{Y}}_i(t)^{CM}|S_{\mathcal{P}}(t))\right)\\
&= E_{AAS}\left(\frac{{P_t}^2}{n_{\mathcal{P}}(t)^2}\sum_{i\in S_{\mathcal{P}}(t)}\frac{1}{V_{it}^2}\sum_{r\in\mathcal{R}(t)}\sum_{j\in\mathcal{V}_i^r(t)}\left[\frac{y_{ij}(t)}{m_{ij}(t)}-\frac{\bar{Y}_i(t)}{m_{ij}(t)}\right]^2\left(\frac{1}{\pi_{r|i}(t)}-1\right)\right)\\
&= \frac{{P_t}^2}{n_{\mathcal{P}}(t)^2}\sum_{i\in\mathcal{P}(t)}\frac{1}{V_{it}^2}E_{AAS}(\mathcal{I}_i(t))\sum_{r\in\mathcal{R}(t)}\sum_{j\in\mathcal{V}_i^r(t)}\left[\frac{y_{ij}(t)}{m_{ij}(t)}-\frac{\bar{Y}_i(t)}{m_{ij}(t)}\right]^2\left(\frac{1}{\pi_{r|i}(t)}-1\right)\\
&= \frac{P_t}{n_{\mathcal{P}}(t)}\sum_{i\in\mathcal{P}(t)}\frac{1}{V_{it}^2}\sum_{r\in\mathcal{R}(t)}\sum_{j\in\mathcal{V}_i^r(t)}\left[\frac{y_{ij}(t)}{m_{ij}(t)}-\frac{\bar{Y}_i(t)}{m_{ij}(t)}\right]^2\left(\frac{1}{\pi_{r|i}(t)}-1\right);
\end{aligned}$$

e,

$$\begin{aligned}
Var_{AAS}\left(E_{CM}(\hat{I}_t^{\mathcal{V}}|S_{1st})\right) &= Var_{AAS}\left(\frac{P_t}{n_{\mathcal{P}}(t)}\sum_{i\in S_{\mathcal{P}}(t)}\bar{Y}_i(t)\right) = \left(\frac{P_t}{n_{\mathcal{P}}(t)}\right)^2\sum_{i\in S_{\mathcal{P}}(t)} Var_{AAS}(\bar{Y}_i(t))\\
&= \left(\frac{P_t}{n_{\mathcal{P}}(t)}\right)^2\left(1-\frac{n_{\mathcal{P}}(t)}{P_t}\right)n_{\mathcal{P}}(t)\sigma^2 = \frac{{P_t}^2}{n_{\mathcal{P}}(t)}\left(1-\frac{n_{\mathcal{P}}(t)}{P_t}\right)\sigma^2\\
&= \frac{{P_t}^2}{n_{\mathcal{P}}(t)}\left(1-\frac{n_{\mathcal{P}}(t)}{P_t}\right)\sum_{i\in\mathcal{P}(t)}\frac{\left(\bar{Y}_i(t)-\bar{Y}(t)\right)^2}{P_t-1},
\end{aligned}$$

em que $\bar{Y}(t) = \frac{1}{P_t}\sum_{i\in\mathcal{P}(t)}\bar{Y}_i(t)$.

Assim, a $Var_{2est}(\hat{I}_t^{\mathcal{V}})$ pode ser estimada de forma não viesada usando:

$$\widehat{Var_{2est}}(\hat{I}_t^V) = \hat{E}_{AAS}\left(\widehat{Var}_{CM}(\hat{I}_t^V|S_{1st})\right) + \widehat{Var}_{AAS}\left(\hat{E}_{CM}(\hat{I}_t^V|S_{1st})\right), \quad (26)$$

em que:

$$\hat{E}_{AAS}\left(\widehat{Var}_{CM}(\hat{I}_t^V|S_{1st})\right) = \frac{P_t}{n_{\mathcal{P}}(t)} \sum_{i \in S_{\mathcal{P}}(t)} \frac{\mathcal{I}_i(t)}{\hat{V}_{it}^2} \sum_{r \in \mathcal{R}(t)} \sum_{j \in S_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \frac{\hat{\bar{Y}}_i(t)^{CM}}{m_{ij}(t)}\right]^2 \frac{1}{\hat{\pi}_{r|i}(t)}\left(\frac{1}{\hat{\pi}_{r|i}(t)} - 1\right);$$

e,

$$\widehat{Var}_{AAS}\left(\hat{E}_{CM}(\hat{I}_t^V|S_{1st})\right) = \frac{{P_t}^2}{n_{\mathcal{P}}(t)}\left(1 - \frac{n_{\mathcal{P}}(t)}{P_t}\right) \sum_{i \in S_{\mathcal{P}}(t)} \frac{\left(\hat{\bar{Y}}_i(t)^{CM} - \hat{\bar{Y}}(t)\right)^2}{P_t - 1},$$

em que $\hat{\bar{Y}}(t) = \frac{1}{n_{\mathcal{P}}(t)} \sum_{i \in S_{\mathcal{P}}(t)} \hat{\bar{Y}}_i(t)^{CM}$.

3.5 Analogia do Método de Captura e Recaptura com o Plano Amostral Proposto

Para ilustrar a aplicação do método de amostragem por Captura e Recaptura na estimativa da quantidade real de preços associados a um produto nos resultados de pesquisa nas plataformas $r \in \mathcal{R}(t)$, em um dado tempo t, podemos utilizar uma analogia com o problema clássico de estimar a quantidade de peixes em um lago.

No caso dos peixes no lago, para aplicar o método de Captura e Recaptura é possível que seja lançada uma tarrafa no lago e, posteriormente, os peixes capturados são marcados e devolvidos para seu local de origem. Se dividirmos esse lago em partes, é possível estimar sua quantidade total de peixes. Contudo existem duas alternativas para utilizar o método amostral mencionado para estimar o total de peixes de todo o lago:

a) Jogar uma tarrafa em uma região do lago e, posteriormente, executar recapturas em outras regiões.

b) Executar captura e recaptura em cada região do lago, inferindo individualmente sobre a quantidade de peixes em cada região do lago.

Para esta tese, o método de Captura e Recaptura será adaptado ao contexto das lojas virtuais e das plataformas de vendas online, adotando a segunda abordagem descrita acima. Essa escolha permite que a análise seja realizada respeitando as particularidades de cada

plataforma $r \in \mathcal{R}(t)$, que funcionam como as "regiões" do mercado online onde as amostras são coletadas.

Com isso, é possível estimar a quantidade V_{it}^{r} de lojas em cada plataforma de forma independente. Ao integrar essas estimativas obtém-se uma visão mais abrangente sobre a distribuição das lojas e dos preços associados ao produto $i \in \mathcal{P}(t)$ na população $\mathcal{V}(t)$ de lojas virtuais. Dessa forma, estimamos a quantidade total de lojas virtuais que vendem o produto i na população $\mathcal{V}(t)$, baseando-nos na interseção das amostras capturadas e recapturadas.

Assim, no contexto do problema em questão, a analogia pode ser interpretada da seguinte forma:

- O "lago" representa o conjunto abrangente de informações relacionadas às vendas virtuais de um produto $i \in \mathcal{P}(t)$;
- Cada "região" definida do lago é uma plataforma $r \in \mathcal{R}(t)$, contendo as lojas virtuais que anunciam o produto i e, consequentemente, seus respectivos preços;
- A "tarrafa" representa o processo de *web scraping*, utilizado para coletar os dados de preços de cada loja virtual nas respectivas plataformas, que correspondem às "regiões" do lago;
- Os peixes correspondem às lojas virtuais que oferecem um preço único para um determinado produto, conforme a definição apresentada na Seção 3.2.

Nesse contexto, podemos imaginar que, assim como existem lojas pertencentes somente à população de determinada(as) plataforma(as) – embora existam interseções entre estas populações –, existem peixes que só podem ser pescados em uma determinada região do lago, o que leva a cada região possuir um conjunto único de peixes. Portanto, nossa analogia será feita considerando que iremos capturar e recapturar os peixes em cada região, ou seja, iremos capturar uma amostra de preços de um produto em cada uma das plataformas, em um tempo t, e depois faremos a recaptura em cada uma delas. Isso porque, da mesma forma que não existe a garantia de que uma tarrafa consiga capturar todos os peixes contidos em uma determinada região de um lago, não existe a garantia de que com uma captura de *web scraping* iremos capturar todos os anúncios apresentados por uma dada plataforma.

Contudo, como capturas diferentes podem pescar o mesmo peixe, esse método amostral nos permite obter as quantidades necessárias para o cálculo do estimador de Captura e Recaptura: as quantidades de peixes capturados nas capturas, as quantidades de peixes capturados nas recapturas e as quantidades de interseções identificadas entre as capturas, em cada região do lago. Portanto, podemos também utilizar a amostragem de captura e recaptura

para estimar a quantidade de peixes de cada região (ou plataforma) e, consequentemente, do lago inteiro (de todas as lojas virtuais que vendem um determinado produto i). Em outras palavras, podemos assumir que a amostragem por captura e recaptura de preços nas plataformas $r \in \mathcal{R}(t)$ é capaz de fornecer uma estimativa sobre os preços ofertados pela população $\mathcal{V}_i(t)$ de lojas virtuais que vendem o produto $i \in S_{\mathcal{P}}(t)$ em um tempo t, e através dessas estimativas podemos estimar sobre os preços ofertados por toda população $\mathcal{V}(t)$ de lojas virtuais que vendem todos os produtos $i \in \mathcal{P}(t)$.

Além disso, precisamos identificar as interseções entre as capturas por preços do produto $i \in \mathcal{P}(t)$ em diferentes plataformas $r \in \mathcal{R}(t)$, pois podemos obter preços inéditos ou preços já "pescados", os quais podem ser reconhecidos como tal se a captura de preços de um mesmo produto $i \in \mathcal{P}(t)$, através de *web scraping*, retornar um mesmo link.

3.6 Proposta de Integração de Planos Amostrais Referentes a Lojas Virtuais e Físicas

Agora, assumimos que $\mathcal{P}(t) = \{1, \dots i, \dots, P_t\}$ representa um cadastro de produtos elementares i vendidos por um conjunto $\mathcal{L}(t) = \mathcal{V}(t) \cup \mathcal{F}(t)$ de lojas. Este conjunto formado pela união de duas populações de lojas: 1) A população $\mathcal{F}(t) = \{1, \dots, k, \dots, F_t\}$ de lojas físicas localizadas na área de interesse de um índice de preços; e 2) A população $\mathcal{V}(t) = \{1, \dots, j, \dots, V_t\}$, definida no Capítulo 3, composta por lojas virtuais que podem atender às regiões onde as lojas físicas estão localizadas. Desta forma, temos que a população $\mathcal{L}(t)$ de lojas possui tamanho $L_t = \sum_{i \in \mathcal{P}(t)} (V_{it} + F_{it}) = V_t + F_t$.

Como $\mathcal{L}(t)$ é composta por lojas físicas e virtuais que vendem pelo menos um produto $i \in \mathcal{P}(t)$, definimos para cada i um conjunto $\mathcal{L}_i(t)$ de lojas que vendem esse produto no tempo t, ou seja, $\mathcal{L}_i(t) = \mathcal{V}_i(t) \cup \mathcal{F}_i(t)$, em que $\mathcal{F}_i(t) = \{1, \dots, k, \dots, F_{it}\}$ é o conjunto de lojas físicas que vendem o produto $i \in \mathcal{P}(t)$, no tempo t. Assim, podemos definir $\mathcal{F}(t) = \bigcup_{i \in \mathcal{P}} \mathcal{F}_i(t)$ e, consequentemente, podemos dizer que $\mathcal{L}(t) = \bigcup_{i \in \mathcal{P}} \mathcal{L}_i(t)$.

Ao longo da pesquisa desta tese, observou-se que lojas que operam tanto fisicamente quanto virtualmente podem oferecer conjuntos de produtos e preços distintos em cada canal de venda. Para fins de análise, estas lojas serão tratadas como diferentes, mesmo que possuam a mesma razão social, uma vez que, geralmente, possuem CNPJs distintos para suas operações físicas e virtuais. Desta forma, assume-se que não há interseção entre estas duas populações: $\mathcal{V}_i(t) \cap \mathcal{F}_i(t) = \emptyset$.

A partir destas definições, o índice $I^*_{0,t}$, que abrange tanto lojas físicas quanto virtuais, será denotado da seguinte forma:

$$I^*_{0,t} = \frac{I^*_t}{I^*_0}, \tag{27}$$

em que:

- I^*_t é a soma dos preços médios dos itens $i \in \mathcal{P}(t)$ nas lojas pertencentes a $\mathcal{L}(t)$, no tempo t;
- I^*_0 é a soma dos preços médios dos itens $i \in \mathcal{P}(t)$ nas lojas pertencentes a $\mathcal{L}(t)$, no tempo base.

Como o método utilizado para estimar I^*_0 é equivalente à aplicada para estimar I^*_t, o foco deste estudo será exclusivamente a estimativa de I^*_t. Portanto, a expressão deste parâmetro será definida utilizando a definição do total $Y_i(t)$ de preços das lojas virtuais, dado $i \in \mathcal{P}(t)$, descrito em (4). Além disso, como I^*_t integra informações de preços das populações $\mathcal{F}(t)$ e $\mathcal{V}(t)$, este parâmetro também abarcará o total $Y_i^{\mathcal{F}}(t)$ dos preços praticados por lojas físicas. Assim, considerando que será utilizado o mesmo conjunto $\mathcal{P}(t)$ de produtos para inferir tanto sobre os preços das lojas virtuais quanto das lojas físicas, inicialmente, esta seção irá apresentar a configuração da população de lojas físicas e propor um estimador para o total $Y_i^{\mathcal{F}}(t)$. Neste caso, estamos assumindo que conhecemos o cadastro $\mathcal{F}(t)$ de lojas físicas e, portanto, os tamanhos F_{it} de lojas físicas que vendem cada $i \in \mathcal{P}(t)$.

Conforme Teixeira Júnior (2020), para inferir sobre a população de preços praticados pela população $\mathcal{F}(t)$ de lojas físicas, podemos lidar com esta como uma população bidimensional. A amostragem bidimensional aplica-se a populações identificadas por duas dimensões distintas, representadas por dois cadastros independentes. No caso de uma população de preços, essas dimensões correspondem a um cadastro de produtos e a uma população de lojas onde as amostras serão coletadas.

Desta forma, ao investigar preços de produtos em diferentes lojas, cada par (produto, loja) pode ser representado como um ponto em uma matriz bidimensional. Suponha que a dimensão “produto” possua P_t unidades, e a dimensão “loja” possua L_t unidades. Nesse caso,

a população pode ser visualizada como uma matriz de tamanho $P_t \times L_t$, onde cada célula representa o preço de um produto específico em uma loja específica.

Assim, o parâmetro $Y_i^{\mathcal{F}}(t)$, que representa o total populacional de preços do produto $i \in \mathcal{P}(t)$ em uma população $\mathcal{F}(t)$ de lojas físicas localizadas em uma determinada cidade no tempo t, pode ser descrito conforme apresentado na Expressão (28).

$$Y_i^{\mathcal{F}}(t) = \sum_{k \in \mathcal{F}_i(t)} y_{ik}(t). \tag{28}$$

Para estimar este parâmetro, assumimos que, por meio de amostragem aleatória simples, já foi selecionada uma amostra $S_{\mathcal{P}}(t) = \{1, 2, \dots, i \dots, n_{\mathcal{P}}(t)\}$ do cadastro $\mathcal{P}(t)$ de produtos, a qual também será utilizada na amostragem de preços das lojas virtuais. A partir disso, é necessário selecionar do cadastro $\mathcal{F}(t)$ de lojas físicas uma amostra probabilística $\not{f}(t) = \{1, 2, \dots, k, \dots, \mathrm{f}_t\}$, de forma independente. O cruzamento da amostra $S_{\mathcal{P}}(t)$ com a amostra $\not{f}(t)$ de lojas físicas resultará em uma amostra bidimensional $S_{\mathcal{F}}(t) = [(i,k): i \in S_{\mathcal{P}}(t), k \in \not{f}(t)\,]$ de preços, com tamanho $n_{\mathcal{P}}(t) \times \mathrm{f}_t$, onde teremos $n_{\mathcal{P}}(t)$ amostras $S_i^{\mathcal{F}}$ de preços, considerando todo $i \in S_{\mathcal{P}}(t)$,.

Com isso, o estimador de Horvitz Thompson pode ser utilizado para estimar $Y_i^{\mathcal{F}}(t)$, por ser um estimador não viesado, de modo que $\hat{Y}_i^{\mathcal{F}}(t)$ pode ser descrito da seguinte maneira:

$$\hat{Y}_i^{\mathcal{F}}(t) = \sum_{k \in \mathcal{F}_i(t)} \frac{y_{ik}(t)\mathcal{I}_k^{\mathcal{F}}(t)}{\pi_k^{\mathcal{F}}(t)},$$

em que:

- $\mathcal{I}_k^{\mathcal{F}}(t) = (1, \text{se } k \in \not{f}(t)\,;\, 0, \text{se } k \notin \not{f}(t))$ é a indicadora de que a loja $k \in \mathcal{F}(t)$ foi selecionada na amostra $\not{f}(t)$.
- $\pi_k^{\mathcal{F}}(t) = P\,(k \in \not{f}(t)) = P[\mathcal{I}_k(t) = 1]$ é a probabilidade da loja $k \in \mathcal{F}(t)$ ser selecionada na amostra $\not{f}(t)$ de lojas físicas.

Considerando que foi realizada uma amostra aleatória simples de lojas virtuais e que $\pi_k = \frac{\mathrm{f}_{it}}{F_{it}}$, temos que:

$$\hat{Y}_i^{\mathcal{F}}(t) = \frac{F_{it}}{\mathrm{f}_{it}} \sum_{k \in \not{f}_i(t)} y_{ik}(t), \tag{29}$$

em que $\not{f}_i(t)$ é a amostra de lojas físicas que vendem o produto i.

A variância deste estimador pode ser definida com base nos resultados de Teixeira Júnior (2020) da seguinte forma:

$$Var\left(\hat{Y}_i^{\mathcal{F}}(t)\right) = \frac{1}{F_t - 1} \sum_{k \in \mathcal{F}(t)} \left(\bar{Y}_k^{\mathcal{F}}(t) - \bar{Y}^{\mathcal{F}}(t)\right)^2, \tag{30}$$

em que:

$$\bar{Y}_k^{\mathcal{F}}(t) = \frac{1}{P_t} \sum_{i \in \mathcal{P}(t)} y_{ik}(t)\,;$$

e,

$$\bar{Y}^{\mathcal{F}}(t) = \frac{1}{P_t F_t} \sum_{i \in \mathcal{P}(t)} \sum_{k \in \mathcal{F}(t)} y_{ik}\,(t).$$

Ainda conforme Teixeira Júnior (2020), um estimador não viesado para a variância descrita em (30) é dado por:

$$\widehat{Var}\left(\hat{Y}_i^{\mathcal{F}}(t)\right) = \frac{1}{\mathrm{f}_{it} - 1} \sum_{k \in \not{f}(t)} \left(\widehat{Y_k}(t) - \widehat{Y^{\mathcal{F}}}(t)\right)^2 - \frac{1}{n_{\mathcal{P}}(t)} \frac{1}{P_t} \frac{\widehat{\sigma^2_{\mathcal{F}\mathcal{P}}}(t)}{n_{\mathcal{P}}(t)}, \tag{31}$$

em que:

$$\widehat{\sigma^2_{\mathcal{F}\mathcal{P}}}(t) = \frac{1}{n_{\mathcal{P}}(t) - 1} \frac{1}{\mathrm{f}_t - 1} \sum_{i \in S_{\mathcal{P}}(t)} \sum_{k \in \not{f}(t)} \left(y_{ik}(t) - \widehat{Y_\iota}(t) - \widehat{Y_k}(t) + \widehat{Y^{\mathcal{F}}}(t)\right),^2$$

em que:

$$\widehat{\bar{Y}_\iota}(t) = \frac{1}{\mathrm{f}_t} \sum_{k \in \mathcal{F}(t)} y_{ik}\,(t);$$

$$\widehat{Y_k}(t) = \frac{1}{n_{\mathcal{P}}(t)} \sum_{i \in S_{\mathcal{P}}(t)} y_{ik}\,(t);$$

$$\widehat{Y^{\mathcal{F}}}(t) = \frac{1}{\mathrm{f}_t n_{\mathcal{P}}(t)} \sum_{i \in S_{\mathcal{P}}(t)} \sum_{k \in \mathcal{F}(t)} y_{ik}(t),$$

Assim, para auxiliar na definição da expressão do parâmetro I_t^*, foi desenhado no Quadro 4 um exemplo de população $\mathcal{L}(t)$, que é a integração da população de lojas físicas e virtuais que vendem os produtos $i \in \mathcal{P}(t)$, levando em consideração a realidade da web. Assim, a parte do Quadro 4 composta pela população de lojas virtuais representa a união das lojas anunciadas pelas plataformas $r \in \mathcal{R}(t)$, ou seja, é o desenho do Quadro 3, sem duplicidade de informações, configurada nas múltiplas plataformas que podem identificar um mesmo anúncio de uma loja virtual. Portanto, no exemplo estamos considerando um cadastro $\mathcal{P}(t)$ de produto de tamanho $P_t = 3$ e a mesma população $\mathcal{V}(t)$ de lojas virtuais, exemplificada no Quadro 3, de tamanho $V_t = 5$. Adicionalmente, estamos considerando uma população $\mathcal{F}(t)$ de lojas físicas de tamanho $F_t = 4$.

Quadro 4- Representação da População-alvo do Índice de Preços das lojas físicas e virtuais

$\mathcal{P}(t)$	População $\mathcal{L}(t)$ de lojas								
	Lojas Físicas $k \in \mathcal{F}(t)$				Lojas Virtuais $j \in \mathcal{V}(t)$				
	1	2	3	4	1	2	3	4	5
1	$y_{11}(t)$	_	$y_{13}(t)$	_	$y_{11}(t)$	_	$y_{13}(t)$	_	$y_{15}(t)$
2	$y_{21}(t)$	$y_{22}(t)$	$y_{23}(t)$	$y_{24}(t)$	$y_{21}(t)$	_	$y_{23}(t)$	$y_{24}(t)$	$y_{25}(t)$
3	$y_{31}(t)$	_	_	_	_	y_{23}	$y_{23}(t)$	_	$y_{23}(t)$

Fonte: Elaboração da autora (2024).

No exemplo do Quadro 4, temos que o produto $i = 2$ pode ser encontrado em quase todas as lojas da população $\mathcal{V}(t)$ de lojas virtuais; ele somente não pode ser encontrado na loja $j = 2$. Ademais, podemos perceber também no Quadro 4, que a loja $k = 1$ do cadastro $\mathcal{F}(t)$ vende todos os produtos $i \in \mathcal{P}(t)$, e a loja física $k = 2$ vende somente o produto $i = 2$.

A partir disso, foi percebido que, ao utilizar o método e os estimadores descritos na Seção 3.4 para tratar dos preços das lojas virtuais e a amostragem bidimensional para tratar dos preços das lojas físicas, é possível integrar os totais de preços destes dois tipos de lojas para um determinado conjunto de produtos $\mathcal{P}(t)$, a fim de se chegar ao preço médio desejado.

Deste modo, o total populacional I_t^* das médias de preços dos produtos $i \in \mathcal{P}(t)$, vendidos por uma população $\mathcal{L}(t)$ de lojas físicas e virtuais, pode ser descrito como:

$$I_t^* = \sum_{i \in \mathcal{P}(t)} \frac{Y_i(t) + Y_i^{\mathcal{F}}(t)}{V_{it} + F_{it}} = \sum_{i \in \mathcal{P}(t)} \frac{1}{V_{it} + F_{it}} \left[\sum_{r \in \mathcal{R}(t)} \sum_{j \in \mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)} + \sum_{k \in \mathcal{F}(t)} y_{ik}(t) \right]. \quad (32)$$

Portanto, com base no estimador de $\hat{Y}_i(t)$, definido em (20), e no estimador do total de preços da população $\mathcal{F}(t)$ de lojas físicas, construído a partir do plano amostral bidimensional (Teixeira Júnior, 2020), descrito anteriormente, o estimador para I_t^* pode ser expresso da seguinte forma:

$$\hat{I}_t^* = \sum_{i \in \mathcal{P}(t)} \frac{\mathcal{I}_i(t)}{\pi_i(t)(\hat{V}_{it} + F_{it})} \left(\sum_{r \in \mathcal{R}(t)} \sum_{j \in S_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)\hat{\pi}_{r|i}(t)} + \sum_{k \in \mathcal{F}(t)} \frac{y_{ik}(t)\mathcal{I}_k^{\mathcal{F}}(t)}{\pi_k^{\mathcal{F}}(t)} \right), \quad (33)$$

em que:

$$\hat{V}_{it} = \frac{\hat{V}_{i\_1}(t)\hat{V}_{i\_2}(t)}{\hat{V}_{i\_12}(t)},$$

em que:

$$\hat{V}_{i\_1}(t) = \sum_{r \in \mathcal{R}(t)} \sum_{j \in S_{i\_1}^r(t)} \frac{1}{m_{ij}(t)\hat{\pi}_{r|i}(t)};$$

$$\hat{V}_{i\_2}(t) = \sum_{r \in \mathcal{R}(t)} \sum_{j \in S_{i\_2}^r(t)} \frac{1}{m_{ij}(t)\hat{\pi}_{r|i}(t)};$$

$$\hat{V}_{i\_12}(t) = \sum_{r \in \mathcal{R}(t)} \sum_{j \in S_{i\_12}^r(t)} \frac{1}{m_{ij}(t)\hat{\pi}_{r|i}(t)};$$

Agora considerando que tanto a amostra de produtos quanto a amostra de lojas físicas serão selecionadas com base no plano de amostragem aleatória simples, o estimador $\hat{I}_t^*$, descrito na Expressão (33), pode ser expresso da seguinte maneira:

$$\hat{I}_t^* = \sum_{i \in \mathcal{P}(t)} \frac{P_t \mathcal{I}_i(t)}{n_{\mathcal{P}}(t)(\hat{V}_{it} + F_{it})} \left(\hat{Y}_i(t) + \frac{F_t}{\mathrm{f}_{it}} \sum_{k \in \mathcal{F}(t)} y_{ij}(t)\mathcal{I}_j^{\mathcal{F}}(t) \right). \quad (34)$$

Considerando que $E_{CMCR}\left(\hat{Y}_i(t)\right)$ é a esperança do total estimado $\hat{Y}_i(t)$ de preços do produto $i \in \mathcal{P}(t)$ nas lojas virtuais $j \in \mathcal{V}_i(t)$, sob o plano amostral CMCR (que envolve amostragem de cadastros múltiplos e amostragem por captura e recaptura), apresentado na Seção 3.4, e que $E_{Bidimensional}(\hat{Y}_i^{\mathcal{F}}(t))$ é a esperança do total estimado $\hat{Y}_i^{\mathcal{F}}(t)$ sob o plano amostral bidimensional, temos que o estimador $\hat{I}_t^*$ é não viesado com a seguinte prova condicionada:

$$E\left(\hat{I}_t^* | V_{it} = \hat{V}_{it}\right) = \frac{\hat{V}_{it} E_{CMCR}\left(\hat{Y}_i(t)\right) + F_{it} E_{Bidimensional}(\hat{Y}_i^{\mathcal{F}}(t))}{\hat{V}_{it} + F_{it}} = \hat{I}_t^*.$$

A variância deste estimador é a combinação da variância do estimador do total $Y_i(t)$ de preços do produto $i \in \mathcal{P}(t)$ ofertados pelas lojas $j \in \mathcal{V}_i(t)$ e a variância do estimador do total $Y_i^{\mathcal{F}}(t)$ de preços do produto $i \in \mathcal{P}(t)$ vendido nas lojas físicas $j \in \mathcal{F}(t)$. Assim, como a covariância entre os preços observados nas lojas físicas e virtuais é igual a 0, temos que:

$$Var\left(\hat{I}_t^* | V_{it} = \hat{V}_{it}\right) = \frac{\hat{V}_{it}{}^2 Var_{CMCR}\left(\hat{Y}_i(t)\right) + F_{it}{}^2 Var_{Bidimensional}(\hat{Y}_i(t)^{\mathcal{F}})}{(\hat{V}_{it} + F_{it})^2}.$$

O estimador desta variância é composto pelo estimador da variância de $\hat{Y}_i(t)$ e pelo estimador da variância de $\hat{Y}_i(t)^{\mathcal{F}}$. Como o primeiro é aproximadamente não viesado, conforme discutido na Seção 3.4, e o segundo é não viesado, conforme Teixeira Júnior (2020), temos também que $\widehat{Var}\left(\hat{I}_t^*\right)$, descrito abaixo, é não viesado em relação a $Var\left(\hat{I}_t^*\right)$.

$$\widehat{Var}\left(\hat{I}_t^* | V_{it} = \hat{V}_{it}\right) = \frac{\hat{V}_{it}{}^2 \widehat{Var}_{CMCR}\left(\hat{Y}_i(t)\right) + F_{it}{}^2 \widehat{Var}_{Bidimensional}(\hat{Y}_i(t)^{\mathcal{F}})}{(\hat{V}_{it} + F_{it})^2}.$$

4 EXPERIMENTO PILOTO

Neste capítulo, iniciaremos apresentando os métodos utilizados para que o experimento piloto proposto nesta tese fosse realizado de forma eficiente e eficaz. Isto inclui uma descrição detalhada do passo a passo seguido na construção do algoritmo de *web scraping*, bem como das ferramentas empregadas nesse processo. Em seguida, será realizada uma análise dos dados gerados pelo experimento, iniciando pela avaliação das quantidades obtidas em cada captura e concluindo com a aplicação dos estimadores da média de preços, propostos na Seção 3.4, tanto para cada par de produto e plataforma quanto para cada produto.

É importante destacar que, devido à ausência de um cadastro de lojas físicas que permitiria o experimento do plano amostral descrito na Seção 3.6, limitaremos o experimento aos estimadores referentes aos preços das lojas virtuais.

4.1 Método de Web Scraping

Para o experimento piloto desta pesquisa, foram escolhidos como objeto de análise os aparelhos de celulares smartphones. Essa escolha se justifica pelo fato de os celulares estarem incluídos no Índice de Preços ao Consumidor Amplo (IPCA), calculado atualmente pelo IBGE, na categoria “Comunicação”. Além disso, trata-se de um produto amplamente disponível em diversos sites, com informações de preços acessíveis, figurando entre os itens mais vendidos na internet (Silva; Huzar, 2020), o que o torna particularmente relevante para um índice de preços voltado a lojas virtuais.

A cidade de Itabuna, localizada no interior da Bahia, nordeste do Brasil, foi definida como referência para o estudo de preços do experimento, por ser a cidade onde estava localizado o computador que executou o algoritmo de *web scraping*. Desta forma, considerando que, na prática, não dispomos de um cadastro completo de celulares com todas as especificações detalhadas de cada modelo vendido por lojas físicas e virtuais nessa cidade, a inferência de preços será realizada com base em um conjunto de produtos que abrange os lançamentos do primeiro semestre de 2023 das principais marcas comercializadas no Brasil por meio de lojas físicas e virtuais. Para os fins deste experimento, esse conjunto de produtos será considerado como nosso cadastro $\mathcal{P}(t)$, dispensando a necessidade de seleção de uma amostra, como seria exigido em um cadastro real de produtos, conforme o plano amostral proposto nesta tese.

Como em uma pesquisa de preços é fundamental definir o nível elementar do produto analisado, no Quadro 5 são apresentadas as especificações dos produtos $i \in \mathcal{P}(t)$, com um nível

de detalhamento (ou agregado elementar) das características dos celulares, abrangendo apenas atributos como marca, linha, modelo e capacidade de armazenamento. Atributos adicionais como memória RAM, tipo de processador, tecnologia de rede suportada e número de chips, serão desconsiderados nesta análise para abranger um maior conjunto de dados.

Quadro 5 – Atributos dos celulares utilizados para coleta de preço das lojas virtuais

i	**MARCA**	**LINHA**	**MODELO**	**ARMAZENAMENTO**
1	SAMSUNG	GALAXY	S23	128GB
2	MOTOROLA	EDGE	30 ultra	256GB
3	APPLE	IPHONE	14 Pro Max	128GB
4	XIAOMI	REDMI	NOTE 11	128GB
5	NOKIA	-	G60	128GB
6	ASUS	-	6 Pro	128GB
7	LG	VELVET	-	128GB
8	PHILCO	HIT	P13	128GB

Fonte: Elaboração da autora (2024).

A coleta de preços será realizada nas plataformas virtuais $\mathcal{R}(t)$, compostas por Google Shopping, Bing Shopping e Mercado Livre. O objetivo deste experimento piloto é estimar a média de preços de cada celular descrito e indexado no Quadro 5, em um instante t, para cada plataforma $r \in \mathcal{R}(t)$, e para todas as plataformas que compõem $\mathcal{R}(t)$. Essa estimativa será baseada na hipótese de que os anúncios das lojas virtuais exibidos nessas plataformas estão oferecendo os produtos para venda e entrega na cidade de Itabuna, pois a máquina está logada nessa localidade.

Apesar de, tecnicamente, um script de *web scraping* poder ser codificado na maioria das linguagens de programação comuns, como Python e R, onde existem diferentes pacotes pré-programados disponíveis para esse fim, foram utilizados neste experimento os pacotes *Beautiful Soup* (Richardson, 2007) *e Requests* (Da Cunha, 2018) na linguagem Python, pois a combinação desses instrumentos conseguiu atender as demandas precisas para a resolução do problema de maneira ágil e descomplicada.

Para iniciar a execução do método de *web scraping* e indicar ao algoritmo as páginas de onde ele deve capturar as informações, considerando todos os produtos e plataformas envolvidos, foram estabelecidos os padrões de URLs que correspondem à busca de preços por um determinado celular nas plataformas que compõem nosso cadastro $\mathcal{R}(t)$. Esses padrões, apresentados abaixo para cada plataforma, foram elaborados com base na estrutura típica das URLs de resultados de busca observada. Identificou-se que, em geral, elas são compostas por

uma parte fixa, específica de cada plataforma (destacada em vermelho), e por uma parte variável, composta pelas palavras-chave utilizadas na pesquisa — neste caso, os atributos definidos no Quadro 5.

a) Google:

https://www.google.com.br/search?q=celular+smartphone+marca+linha+modelo+armazenamento

b) Bing:

https://www.bing.com/shop?q=celular+smartphone+marca+linha+modelo+armazenamento

c) Mercado Livre:

https://lista.lojalivre.com.br/celular+smartphone+marca+linha+modelo+armazenamento

A utilização dessa estrutura permite que o algoritmo de *web scraping* direcione as buscas não apenas para os produtos especificados, mas para grandes conjuntos de produtos. Isso ocorre porque ele será capaz de acessar a página de resultados de busca para esses produtos no instante t, a partir de uma planilha de Excel ou qualquer outro tipo de lista contendo os atributos dos celulares requeridos. No entanto, embora essa abordagem facilite a captura dos resultados das plataformas, ela não garante que todos os anúncios capturados correspondam exatamente ao produto desejado. Pode haver no resultado, por exemplo, a inclusão de anúncios de produtos com nomes semelhantes, acessórios relacionados aos celulares pesquisados, ou vendas casadas. Por isso, uma etapa adicional de verificação é necessária para assegurar a qualidade dos dados.

Para este propósito, o algoritmo de *web scraping* desenvolvido (APÊNDICE B) incorpora um modelo de *machine learning*, *Random Forest* (Rigatti, 2017), utilizando-o como classificador para aprimorar a filtragem dos resultados. O *Random Forest* é eficaz por combinar múltiplas árvores de decisão, permitindo lidar com dados complexos e variados, ao mesmo tempo em que reduz o risco de incluir anúncios irrelevantes.

Neste experimento, foi criada uma planilha com 1.300 resultados de *web scraping*, contendo descrições de anúncios, links e preços. Cada anúncio foi classificado manualmente, sendo rotulado com o nome da marca, caso correspondesse a um dos produtos requeridos, ou marcado como “Não Condiz”, caso contrário. A partir dessa planilha, o algoritmo foi treinado para classificar anúncios em futuros resultados de *web scraping*, aprendendo a identificar como os atributos definidos no Quadro 5 (marca, linha, modelo e capacidade de armazenamento) se apresentam nas descrições e links dos anúncios.

Além disso, o algoritmo também aprende sobre os preços dos produtos, entendendo que, mesmo na ausência de um atributo específico, um anúncio pode ser considerado condizente se os outros critérios e o preço estiverem alinhados. Isso permite verificar se o anúncio corresponde ao produto desejado, levando em conta que um mesmo produto pode ser descrito de maneiras diferentes em anúncios distintos. Assim, anúncios que não atendem a esses critérios são automaticamente excluídos da análise. Essa etapa de classificação é de suma importância para o processo de estimativa de preços, pois, caso todos os resultados fossem utilizados sem a aplicação de um filtro que identificasse com precisão o produto desejado, haveria um viés de seleção ou de cobertura, já que seriam incluídos nas análises dados de produtos que não pertenceriam à população pesquisada.

É importante ressaltar que é fundamental dosar a quantidade de critérios de classificação, pois a inclusão excessiva de especificações pode resultar na exclusão de anúncios relevantes, afetando a precisão da estimativa de preços. Por outro lado, a exclusão de atributos essenciais pode ampliar demais o escopo da busca, levando à inclusão de preços de produtos que não correspondem ao modelo desejado, comprometendo a qualidade dos dados utilizados no cálculo do estimador.

Cabe destacar também que temos conhecimento de que o uso de uma planilha manual para ensinar o algoritmo a respeito da classificação dos dados seria inviável para um grande conjunto de produtos. Portanto, para atender trabalhos futuros, já está sendo desenvolvido um algoritmo de aprendizado de máquina não supervisionado. Esse algoritmo utilizará técnicas como K-means e redução de dimensionalidade (Análise de Componentes Principais - PCA) para identificar padrões ou clusters nos dados, dispensando a necessidade de rótulos de treinamento.

Ademais, como nossa análise se limita a apenas um celular por marca, a categorização dos produtos que correspondem aos critérios pesquisados é feita utilizando o nome da marca como referência principal, o que facilita a visualização e análise, especialmente nas representações gráficas subsequentes. No entanto, reconhecemos que em análises futuras, envolvendo múltiplos produtos por marca, esse modo de resumir o produto utilizando a marca não seria suficiente. Nesse caso, seria necessário utilizar palavras-chave adicionais, para diferenciar os anúncios de forma mais precisa, garantindo que cada produto seja corretamente categorizado e analisado de forma visualmente simplificada.

Dando continuidade à abordagem adotada, a organização e o armazenamento dos dados capturados por *web scraping* nas três plataformas de $\mathcal{R}(t)$ foram feitos em um arquivo Excel (.xlsx), estruturado como um banco de dados. Esse formato facilita a gestão e análise das

informações, que incluem colunas como: nome da plataforma de origem, descrição do produto, preço à vista (excluindo custos de envio), nome da loja, link do anúncio, data e hora da coleta, e a indicação de conformidade com os produtos (especificando o nome da marca ou a expressão “Não Condiz”). Dessa forma, é possível que somente os anúncios relevantes para a pesquisa sejam considerados nas análises. Além disso, como uma mesma busca por um produto pode resultar em múltiplas ocorrências do mesmo anúncio, esse formato de banco de dados possibilita a remoção de duplicatas, identificando-as através dos links, que, nesse caso, se apresentam de forma idêntica para o mesmo anúncio em uma plataforma.

Por outro lado, para lidar com a situação em que um mesmo anúncio pode ser capturado em diferentes plataformas, foi necessário realizar um estudo detalhado sobre a construção das URLs dos anúncios e o significado de cada parte delas. Foi descoberto que os códigos podem variar dependendo do endereço IP do computador utilizado e da plataforma de origem do link. Para eliminar esses elementos adicionais em URLs contendo o domínio “www.mercadolivre.com.br”, por exemplo, utilizou-se a função replace() para substituir o código “#searchVariation%3D” por “?item_id=” e, em seguida, aplicou-se a função “split('%')[0]” para remover qualquer conteúdo remanescente após o símbolo "%".

Dessa forma, o estudo das URLs resultou em um código delimitador (ou normalizador), capaz de remover códigos externos presentes no final dos links capturados. Esse processo gera URLs “limpas” e padronizadas, facilitando a comparação entre links. Isso permite identificar, de forma eficiente, as interseções entre os anúncios utilizando o link “cru”, tornando possível verificar se um mesmo anúncio foi capturado mais de uma vez, seja na mesma plataforma ou em plataformas distintas.

Sem a normalização dos links, a identificação de anúncios duplicados ficaria limitada aos nomes das lojas virtuais e às descrições dos produtos capturados, dificultando o processo, especialmente porque os anúncios podem se apresentar de forma distinta em plataformas diferentes. Esse problema é ainda mais evidente em *marketplaces*, onde lojas virtuais distintas podem ser listadas (em uma mesma plataforma ou em plataformas diferentes) apenas com o nome do *marketplace*, sem expor os nomes reais das lojas que estão comercializando dentro dele. Além disso, uma mesma loja pode anunciar produtos diferentes e, devido ao nível elementar considerado na coleta de preços, esses anúncios podem ser classificados no banco de dados como referente a um único item. Nesse cenário, se as descrições dos anúncios forem semelhantes, omitindo características ignoradas, e os preços apresentados forem idênticos, a ausência de normalização dos links pode resultar na identificação equivocada de duplicatas.

Além do desafio descrito, o processo de *web scraping* exigiu o tratamento de links capturados que atuam somente como direcionadores para as URLs finais, contendo apenas parâmetros de rastreamento e redirecionamento. Exemplos incluem links capturados que iniciam com os seguintes domínios: https://www.bing.com:443/alink/link, https://validate.perfdrive.com/ e https://www.google.com/aclk?sa. Para solucionar esse problema, foi desenvolvido um código que identifica e processa esses links intermediários, acessando-os e extraindo as URLs finais de interesse. Esse processo utilizou técnicas de requisições HTTP para seguir os redirecionamentos e capturar corretamente as URLs relevantes para a análise. A identificação e o tratamento correto desses links intermediários foram essenciais para garantir que apenas as URLs finais fossem processadas, eliminando ruídos e garantindo a precisão dos dados coletados.

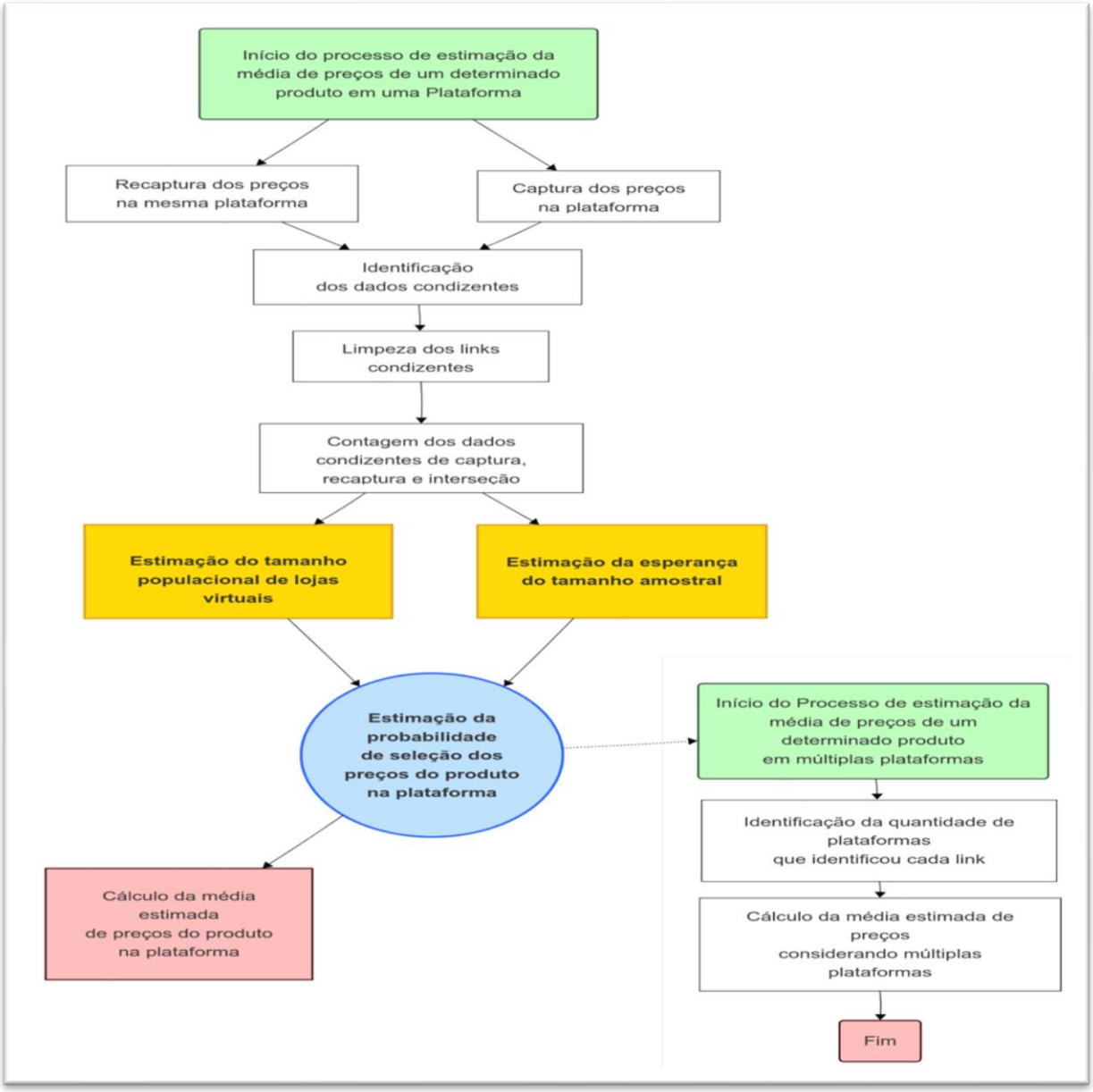
Após os tratamentos realizados, duas colunas extras foram adicionadas ao banco de dados. A primeira indica o número total de vezes que cada anúncio específico foi capturado, enquanto a segunda informa as plataformas em que foram encontrados. Como as duplicatas foram eliminadas, cada anúncio aparece apenas uma vez por plataforma, resultando em uma contagem que varia de 1 (quando o anúncio foi encontrado em apenas uma plataforma) a 3 (correspondendo ao total de plataformas analisadas). Essa contagem equivale à quantidade m_{ij} presente nas expressões dos parâmetros e estimadores relacionados à média de preços de um determinado produto em todas as plataformas $r \in \mathcal{R}(t)$, definidas, respectivamente, nas Seções 3.3 e 3.4. Assim, com todos os procedimentos descritos, foi possível determinar a quantidade de plataformas que exibem cada anúncio, o que é fundamental para alcançar a estimativa proposta no plano amostral.

Ademais, durante o desenvolvimento do código de *web scraping* para coleta de preços, foram enfrentados diversos outros desafios técnicos e metodológicos. Os principais obstáculos incluíram mudanças frequentes na estrutura das páginas web, bloqueios por IP e falhas de conexão. Para lidar com essas questões, foi empregada uma série de soluções. O uso do pacote *BeautifulSoup* permitiu flexibilidade no código, por ser adaptado facilmente quando existe alterações na estrutura HTML das páginas. Para contornar bloqueios por IP, implementamos diferentes *User-Agents*, técnicas de *retries* (que envolvem a repetição de uma operação após falhas específicas), bem como pausas e recomeços automáticos, garantindo a continuidade da coleta de dados. Para garantir a robustez do sistema diante de falhas de conexão, implementamos verificações de conectividade e *retries* automáticos. Além disso, a manipulação dos dados foi facilitada com a biblioteca pandas, permitindo limpeza e padronização de dados

para análises futuras; já a utilização da função *ThreadPoolExecutor* (Sodian, 2022) possibilitou a execução paralela das requisições, acelerando o processo.

Com todas as etapas descritas nesta seção, foi construída uma base de dados rica e bem estruturada, pronta para o cálculo das estimativas através dos estimadores descritos na Seção 3.4. No entanto, como estamos utilizando uma lista de produtos, e não uma amostra conforme o plano amostral originalmente proposto, os estimadores serão calculados individualmente para cada produto e plataforma. Isso ocorre porque, nesse contexto, calcular a média de todos os produtos em todas as plataformas de $\mathcal{R}(t)$ seria uma tarefa trivial e não acrescentaria valor a este experimento piloto. Desta forma, o experimento seguirá o seguinte fluxo:

Figura 3 – Fluxograma das etapas seguidas no experimento piloto



Fonte: Elaborado pela autora (2025).

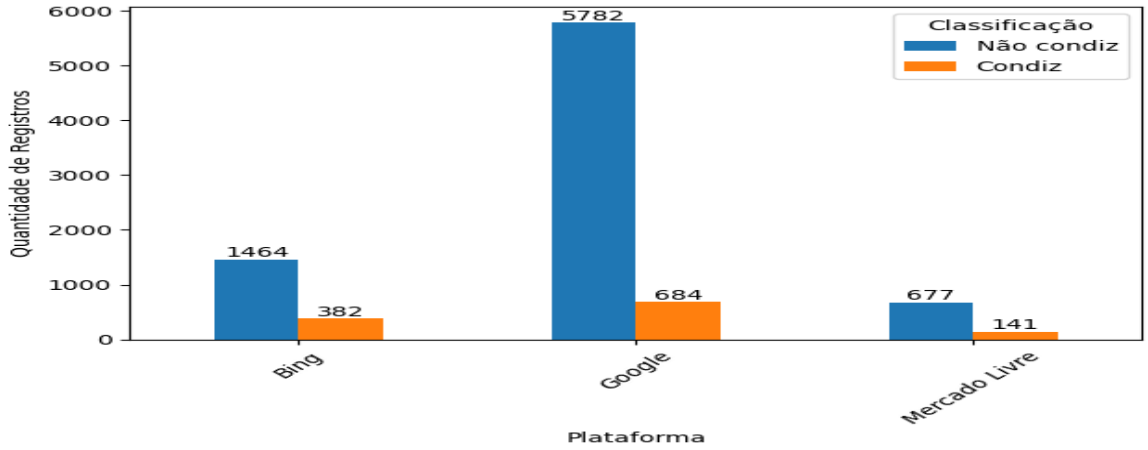
4.2 Análise dos dados do experimento piloto

Seguindo o plano amostral, proposto na Seção 3.4, para estimar a média de preços dos produtos descritos no Quadro 5 nas lojas virtuais que anunciam suas vendas em cada plataforma de $\mathcal{R}(t)$ — Bing Shopping, Google Shopping e Mercado Livre — e em todas elas, foi empregada o método de captura e recaptura. A captura ocorreu entre as 22h00 do dia 30/09/2024 e as 07h35 de 01/10/2024, enquanto a recaptura foi realizada entre as 23h30 de 01/10/2024 e as 09h43 de 02/10/2024. Embora essas coletas tenham sido realizadas em momentos distintos, serão tratadas como pertencentes a um único instante t, como orienta nossa proposta metodológica, já que a recaptura é feita depois de um tempo significativamente pequeno depois da captura.

Como durante o processo de *web scraping*, não capturamos exclusivamente anúncios relacionados aos produtos que estamos considerando, os dados foram segmentados em dois grupos: anúncios que realmente condizem com o que estávamos pesquisando e aqueles que não condizem. Esta classificação foi realizada utilizando um método de classificação baseada em *machine learning*, detalhada na seção anterior, garantindo precisão na identificação de anúncios relevantes para a análise, uma vez que obtemos uma alta acurácia (0,97), o que significa que 97% das previsões feitas pelo modelo estavam corretas.

Dessa forma, o Gráfico 1 apresenta as quantidades de anúncios obtidas nas etapas de captura e recaptura para as três plataformas analisadas, excluindo os links duplicados dentro de cada captura. O gráfico destaca a expressiva quantidade de dados coletados que não estão relacionados aos produtos de interesse, os quais foram descartados para garantir uma análise precisa e alinhada aos objetivos da pesquisa. Na plataforma Bing, por exemplo, foram capturados 382 anúncios que correspondem aos produtos pesquisados, enquanto outros 1.464 eram irrelevantes. Já no Google, observamos um volume substancial de anúncios capturados, com 684 relacionados aos produtos de interesse, mas uma maioria significativa, 5.782 anúncios, não condizendo com o que foi buscado. Por fim, no Mercado Livre, a quantidade total de anúncios capturados foi menor, sendo 141 correspondentes aos produtos pesquisados e 677 classificados como não condizentes.

Gráfico 1– Quantidade de anúncios que condizem com os requeridos, descontadas as duplicidades entre captura e recaptura



Fonte: Elaborado pela autora (2024).

Estes resultados evidenciam a importância de filtrar os dados capturados para assegurar a qualidade e relevância das informações analisadas, já que mais de 70% dos resultados, em geral, não condizem com os produtos que são de interesse para pesquisa de preço.

A partir da classificação dos anúncios condizentes, é possível realizar o plano amostral de captura-recaptura Bernoulli (CRB), proposto em 3.4, o qual viabiliza estimar a média de preços de cada produto $i \in \mathcal{P}(t)$ (representado pelo conjunto de produtos do Quadro 5) em cada plataforma $r \in \mathcal{R}(t)$, conforme descrito na Expressão (13).

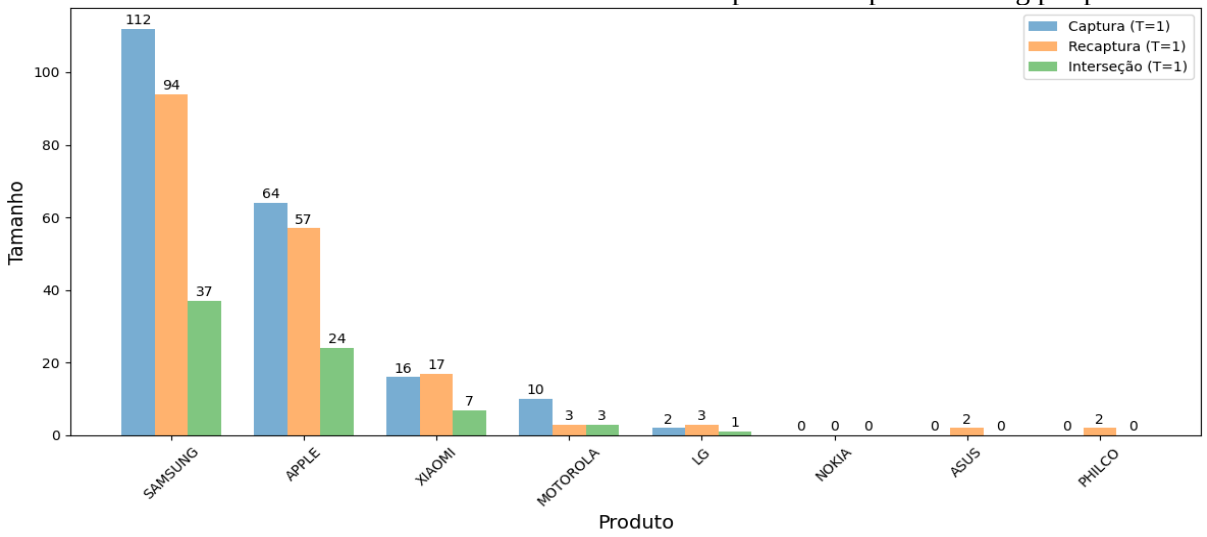
Para seguir com o experimento, indexamos Bing Shopping como $r = 1$, Google Shopping como $r = 2$ e Mercado Livre como $r = 3$, além de seguirmos a indexação do Quadro 5 para os produtos. A partir disso, analisamos inicialmente os tamanhos de captura e recaptura obtidos na plataforma Bing, cujos resultados estão apresentados no Gráfico 2. Observa-se no gráfico que os produtos das marcas Samsung ($i = 1$) e Apple ($i = 3$) apresentaram os maiores números de preços $y_{ij}(t)$ capturados e recapturados na plataforma em questão. No caso do celular da Samsung, a amostra $S^1_{1\_1}(t)$ de preços, referente à captura, possuiu um tamanho $\mathrm{v}^1_{1\_1}(t) = 112$, enquanto a amostra da recaptura, $S^1_{1\_2}(t)$, obteve tamanho $\mathrm{v}^1_{1\_2}(t) = 94$. Em relação às amostras do produto da Apple, $S^1_{3\_1}(t)$ apresentou $\mathrm{v}^1_{3\_1}(t) = 64$ capturas, e $S^1_{3\_2}(t)$ obteve $\mathrm{v}^1_{3\_2}(t) = 57$ recapturas. Entretanto, as interseções entre as amostras $S^1_{1\_12}$ e $S^1_{3\_12}$, obtidas das capturas e recapturas, tiveram tamanho: $\mathrm{v}^1_{1\_12}(t) = 37$ para o celular da Samsung e $\mathrm{v}^1_{3\_12}(t) = 24$ para o celular da Apple.

Estes resultados refletem a alta visibilidade digital dos celulares da Samsung e da Apple, impulsionada por sua ampla cobertura de mercado e estratégias de marketing eficazes. Além disso, a variação nas quantidades de capturas e recapturas observadas no Gráfico 2 está alinhada

ao modelo adotado, que considera as capturas por *web scraping* como amostras independentes, sujeitas a variações inerentes ao processo de amostragem de Bernoulli.

Para os celulares de outras marcas, como Xiaomi, Motorola, LG, Nokia, Asus e Philco, os números de capturas e recapturas foram significativamente menores. Como exemplo, temos que a amostra $S^1_{4\_1}(t)$ obtida no Bing, relacionada ao celular da Xiaomi, apresentou tamanho $\mathrm{v}^1_{4\_1}(t) = 16$, já a recaptura, $\mathrm{v}^1_{4\_2}(t) = 17$. O tamanho da interseção, por sua vez, foi de apenas $\mathrm{v}^1_{4\_12}(t) = 7$. Já os celulares das marcas como Nokia, ASUS e Philco não tiveram links capturados ou recapturados. Este comportamento reflete a menor presença digital desses produtos na plataforma Bing.

Gráfico 2 - Tamanhos das amostras de Bernoulli na captura e recaptura no Bing por produto.



Fonte: Elaborado pela autora (2024).

De maneira semelhante ao observado no Bing, podemos perceber no Gráfico 3, que no Google os produtos das marcas Apple e Samsung continuam liderando em números de capturas e recapturas, seguidas também por uma presença significativa de produtos da Xiaomi e Motorola. No entanto, uma diferença notável no Google é a maior estabilidade entre os dados de captura e recaptura, evidenciada pela elevada interseção de links.

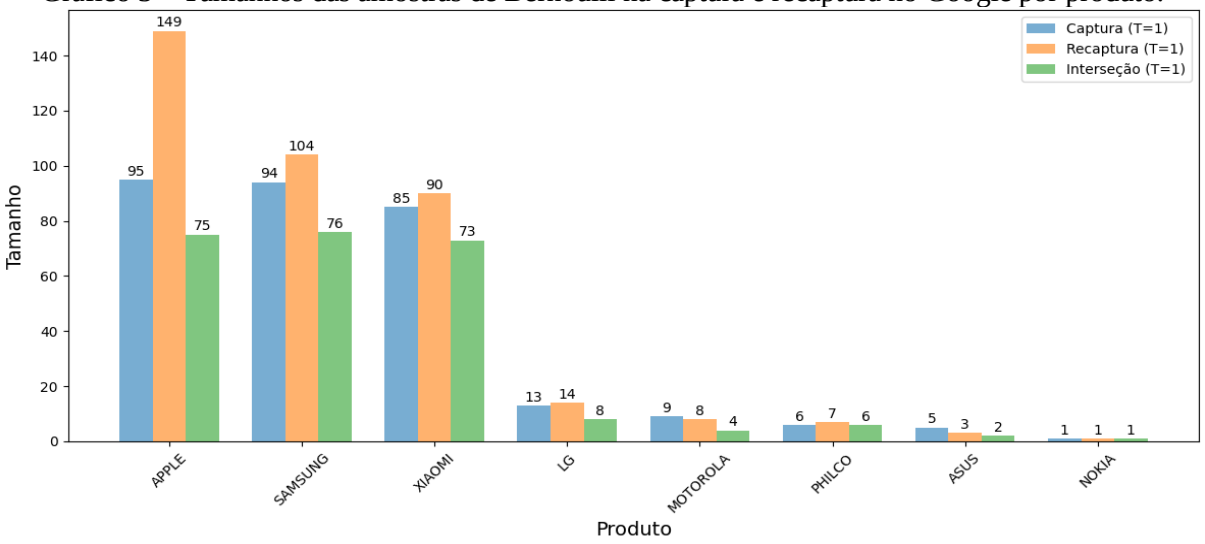
Em relação ao comportamento do produto da Apple ($i = 3$) nessa plataforma, o tamanho da captura $S^2_{3_1}(t)$ foi de $\mathrm{v}^2_{3\_1}(t) = 95$, enquanto a recaptura $S^2_{3_2}(t)$ alcançou $\mathrm{v}^2_{3\_1}(t) = 149$. A interseção entre as capturas e recapturas, $S^2_{3_{12}}(t)$, atingiu $\mathrm{v}^2_{3\_12}(t) = 75$, indicando que uma proporção considerável de anúncios permanece indexada entre as execuções de *web scraping*.

Para o produto da Samsung ($i = 1$), a captura $S^2_{1_1}(t)$ apresentou um tamanho $\mathrm{v}^2_{1\_1}(t) = 94$, próximo ao da captura do produto da Apple, e a amostra $S^2_{1\_2}(t)$ obteve $\mathrm{v}^2_{1\_2}(t) = 104$

recapturas. Já a amostra da interseção, $S^2_{1\_12}(t)$, obteve um tamanho de $\mathrm{v}^2_{1\_12}(t) = 76$ registros, reforçando também a estabilidade dessa plataforma.

Na plataforma Google, todos os produtos apresentaram quantidades maiores de capturas e recapturas em comparação com as outras plataformas. O produto da Xiaomi ($i = 4$), por exemplo, obteve $\mathrm{v}^2_{4\_1}(t) = 85$ capturas, $\mathrm{v}^2_{4\_2}(t) = 90$ recapturas e $\mathrm{v}^2_{4\_12}(t) = 73$ interseções. Já os resultados de *web scraping* para o produto da Motorola, resultou em números mais baixos e com resultados de captura e recaptura similares: $\mathrm{v}^2_{2\_1}(t) = 9$ capturas, $\mathrm{v}^2_{2\_2}(t) = 8$ recapturas e $\mathrm{v}^2_{2\_12}(t) = 04$ interseções. Outras marcas, como Nokia ($i = 5$), Asus ($i = 6$), LG ($i = 7$) e Philco ($i = 8$), apresentam números de capturas e recapturas significativamente menores, como também foi observado no experimento feito na plataforma Bing.

Gráfico 3 – Tamanhos das amostras de Bernoulli na captura e recaptura no Google por produto.



Fonte: Elaborado pela autora (2024).

No caso do Mercado Livre ($r = 3$), o Gráfico 4 demonstra que a Xiaomi apresenta um desempenho notável, ocupando a segunda posição em capturas e recapturas, com valores que superam os da Apple. Assim, especificamente, para a Xiaomi ($i = 4$), o tamanho da captura $S^3_{4\_1}(t)$ foi de $\mathrm{v}^3_{4\_1}(t) = 20$, enquanto a recaptura $S^3_{4\_2}(t)$ atingiu $\mathrm{v}^3_{4_2}(t) = 23$. Além disso, a interseção $S^3_{4\_12}(t)$ entre as capturas e recapturas resultou em $\mathrm{v}^3_{4\_2}(t) = 19$, indicando que a maioria dos anúncios capturados permaneceu indexada entre as execuções.

Essa presença significativa da Xiaomi no Mercado Livre pode ser explicada pelas características da plataforma, que é amplamente conhecida por oferecer uma grande diversidade de produtos, incluindo aqueles de marcas emergentes ou de maior competitividade em termos de preço. Isso também sugere que as lojas virtuais podem estar investindo ativamente na

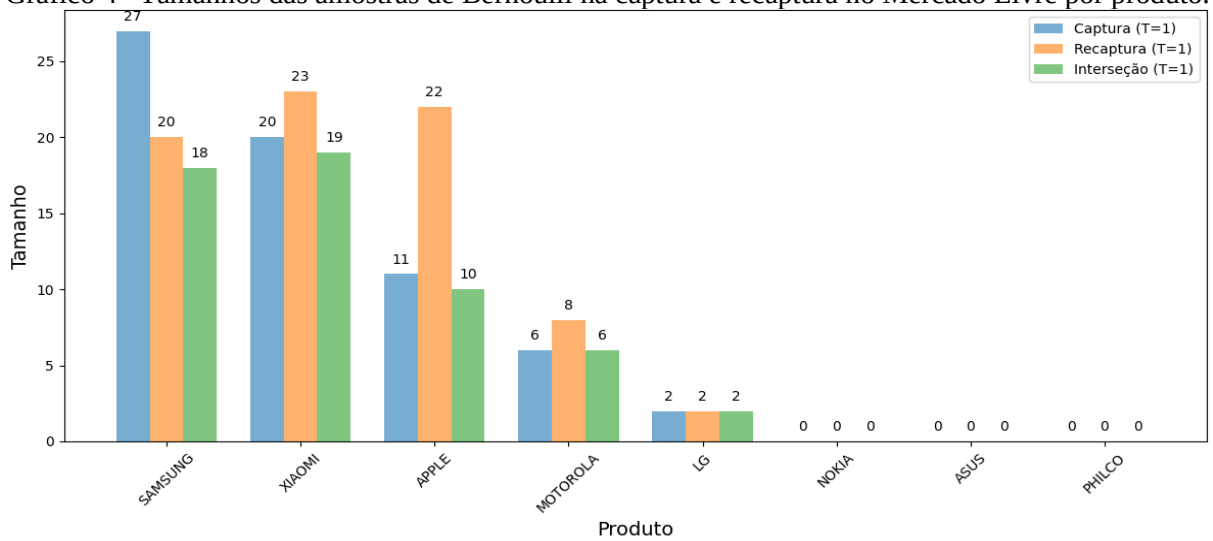
visibilidade desse celular nesse *Marketing Place*, aproveitando o foco da plataforma em atender a consumidores que buscam opções com bom custo-benefício.

Portanto, somente no Mercado Livre o produto da Samsung ($i = 1$) lidera em termos de quantidade de capturas, com um tamanho de $\mathrm{v}_{1\_1}^{3}(t) = 27$ na amostra de captura, enquanto a amostra $S_{1\_2}^{3}(t)$ da recaptura atingiu $\mathrm{v}_{1\_2}^{3}(t) = 20$. A interseção $\mathrm{v}_{1\_12}^{3}(t) = 18$ indica um bom nível de estabilidade nos anúncios desse produto entre capturas e recapturas dentro da plataforma, no tempo em questão. Já para o produto da Apple ($i = 3$), a captura $S_{3\_1}^{3}(t)$ teve tamanho $\mathrm{v}_{3\_1}^{3}(t) = 11$, enquanto a recaptura $S_{3\_2}^{3}(t)$ foi $\mathrm{v}_{3\_2}^{3}(t) = 22$, com uma interseção $S_{3\_12}^{3}(t)$ de tamanho $\mathrm{v}_{3\_12}^{3}(t) = 10$, reforçando a percepção de estabilidade dos anúncios da plataforma em um curto período.

Além disso, o Gráfico 4 destaca a ausência de capturas e recapturas para marcas como Nokia, Asus e Philco, o que está em conformidade com o observado nas outras plataformas. Essa ausência reflete o foco do Mercado Livre em produtos de marcas com maior demanda e reconhecimento no mercado, corroborando a estratégia da plataforma de priorizar itens que possuem alta competitividade e maior rotatividade nas vendas.

Em suma, embora o Mercado Livre apresente menor volume geral de registros em comparação a outras plataformas como Google Shopping e Bing Shopping, ainda mantém uma estrutura que privilegia anúncios de marcas populares, especialmente aquelas com forte presença no mercado, como Samsung e Xiaomi.

Gráfico 4– Tamanhos das amostras de Bernoulli na captura e recaptura no Mercado Livre por produto.



Fonte: Elaborado pela autora (2024).

De forma geral, os resultados apresentados evidenciam padrões distintos entre as plataformas analisadas, com destaque para a liderança da Samsung, Apple e Xiaomi em capturas e recapturas, variando de acordo com as características de cada ambiente. No entanto, os números relativamente baixos para algumas marcas podem ser explicados pelo fato de estarmos considerando produtos lançados no primeiro semestre de 2023, enquanto este experimento foi realizado no segundo semestre de 2024. Isso significa que alguns produtos podem ter sido descontinuados. Dependendo da estratégia de cada plataforma, que pode priorizar itens mais recentes ou com maior demanda, esses produtos podem não ser mais amplamente disponíveis. Um exemplo é a marca LG, que nesse período anunciou sua saída do mercado de smartphones, o que resultou na descontinuação de sua linha de dispositivos.

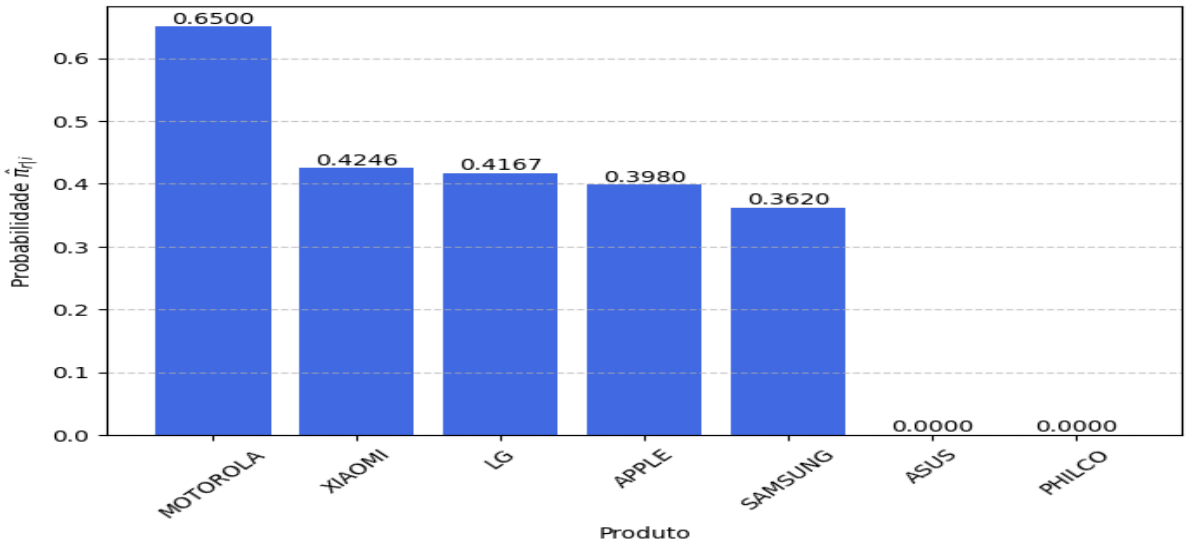
4.3 Estimativas envolvendo dados do experimento piloto.

Como as amostras apresentadas na seção anterior, referentes à captura e recaptura em cada plataforma $r \in \mathcal{R}(t)$ para cada produto $i \in \mathcal{P}(t)$, estão sendo tratadas como réplicas de uma amostragem de Bernoulli, com probabilidade $\pi_{r|i}(t)$ (Expressão (8)(19)), estimaremos esta probabilidade sob um plano amostral de Captura-Recaptura Bernoulli (CRB), conforme orientado na Seção 3.4. Para isso, utilizaremos os tamanhos $\mathrm{v}_{i\_1}^{r}(t)$, $\mathrm{v}_{i\_2}^{r}(t)$, $\mathrm{v}_{i\_12}^{r}(t)$, $\forall i \in \mathcal{P}(t)$ e $\forall r \in \mathcal{R}(t)$, obtidos a partir deste experimento piloto e apresentados na seção anterior.

A partir destes valores, empregaremos as Expressões (10) e (11) para estimar, respectivamente, a esperança $E_{BE}(\mathrm{v}_i^r(t))$ do tamanho amostral de preços associados ao produto $i \in \mathcal{P}(t)$ e à plataforma $r \in \mathcal{R}(t)$ no tempo t, bem como o tamanho populacional V_{it}^{r} correspondente. Desta forma, obtemos as estimativas de $\pi_{r|i}(t)$, com base em (9), que, mais especificamente, correspondem às probabilidades estimadas de seleção de preço dos produtos listados no Quadro 5, identificados por suas respectivas marcas, em uma nova execução de *web scraping* na plataforma $r \in \mathcal{R}(t)$.

Com isto, no Gráfico 5, pode-se perceber que a Bing ($r = 1$) apresentou, em geral, probabilidades estimadas de seleção de preços baixas, menores do que 50%, como as relacionadas ao produto da Samsung ($\hat{\pi}_{1|1}(t)$ = 0,362) e da Apple ($\hat{\pi}_{1|3}(t) = 0{,}398$). Em contrapartida, o celular da Motorola foi o que obteve maior probabilidade estimada de seleção ($\hat{\pi}_{1|2}(t) = 0{,}65$).

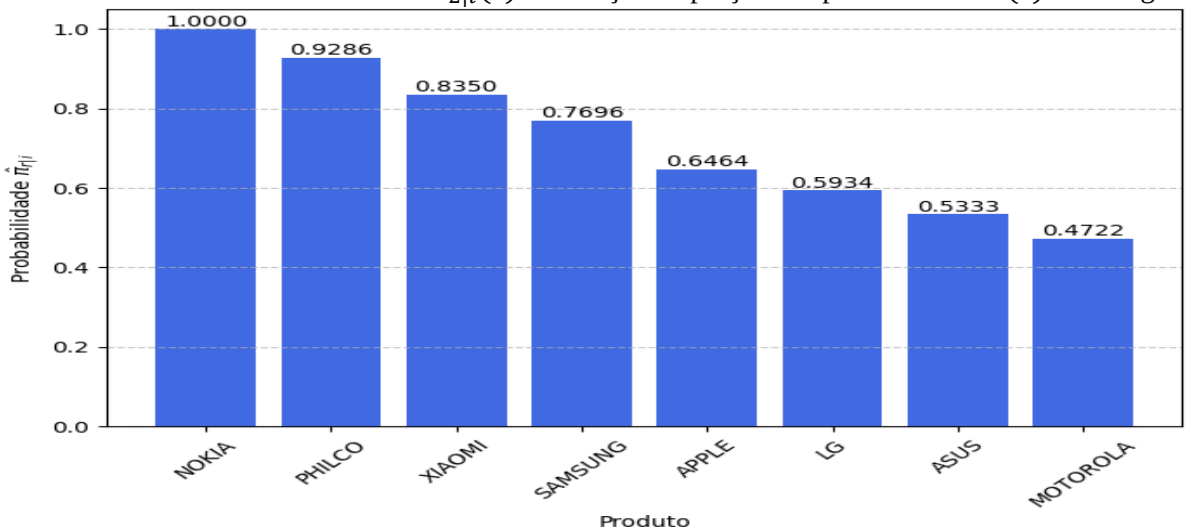
Gráfico 5- Probabilidades estimadas $\hat{\pi}_{1|i}(t)$ de seleção de preços dos produtos $i \in \mathcal{P}(t)$ no Bing



Fonte: Elaboração da autora (2024).

Já a Google ($r = 2$), conforme Gráfico 6, apresentou, em geral, probabilidades de seleção de preços mais altas, em comparação com a plataforma Bing. Isto pode ser decorrente do fato de que esta plataforma agrega um número maior de lojas, e conforme a definição de $\hat{\pi}_{r|i}(t)$, em (9), quanto maior a estimativa do tamanho da população de lojas que vendem um determinado produto em uma plataforma, menor é a probabilidade estimada de seleção. Assim, as probabilidades estimadas de seleção mais altas na Google são observadas para o celular da Nokia e da Philco, com valores de $\hat{\pi}_{2|5}(t) = 1{,}0$ e $\hat{\pi}_{2|8}(t) = 0{,}928$, respectivamente, sendo que estes produtos, nesta plataforma, apresentaram um tamanho reduzido de amostras nas capturas e uma quantidade significativa de interseção (Gráfico 3).

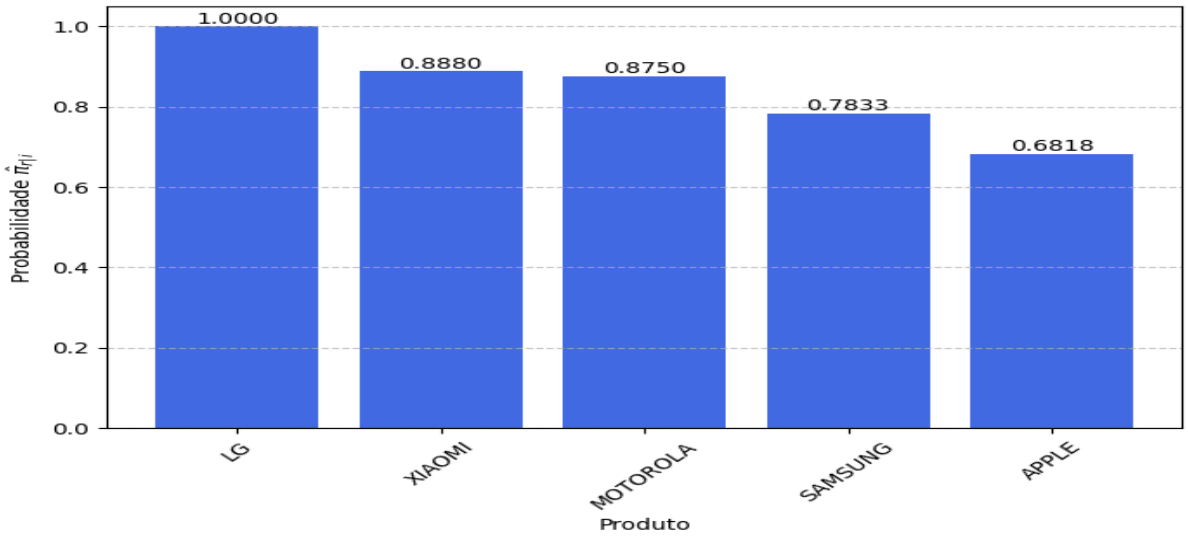
Gráfico 6 - Probabilidades estimadas $\hat{\pi}_{2|i}(t)$ de seleção de preços dos produtos $i \in \mathcal{P}(t)$ no Google



Fonte: Elaboração da autora (2024).

O Mercado Livre ($r = 3$), por sua vez, em geral, apresenta altas probabilidades estimadas de seleção para a maioria dos produtos listados no Quadro 5, conforme ilustrado no Gráfico 7, com destaque para os celulares da LG ($\hat{\pi}_{3|7}(t)$ = 1,00), Xiaomi ($\hat{\pi}_{3|4}(t)$ = 0,888) e Motorola ($\hat{\pi}_{3|2}(t) = 0{,}875$). Em contrapartida, os celulares da Philco, Nokia e Asus apresentaram uma probabilidade estimada nula de seleção nesta plataforma.

Gráfico 7 - Probabilidades estimadas $\hat{\pi}_{3|i}(t)$ de seleção de preços dos produtos $i \in \mathcal{P}(t)$ no Mercado Livre



Fonte: Elaboração da autora (2024).

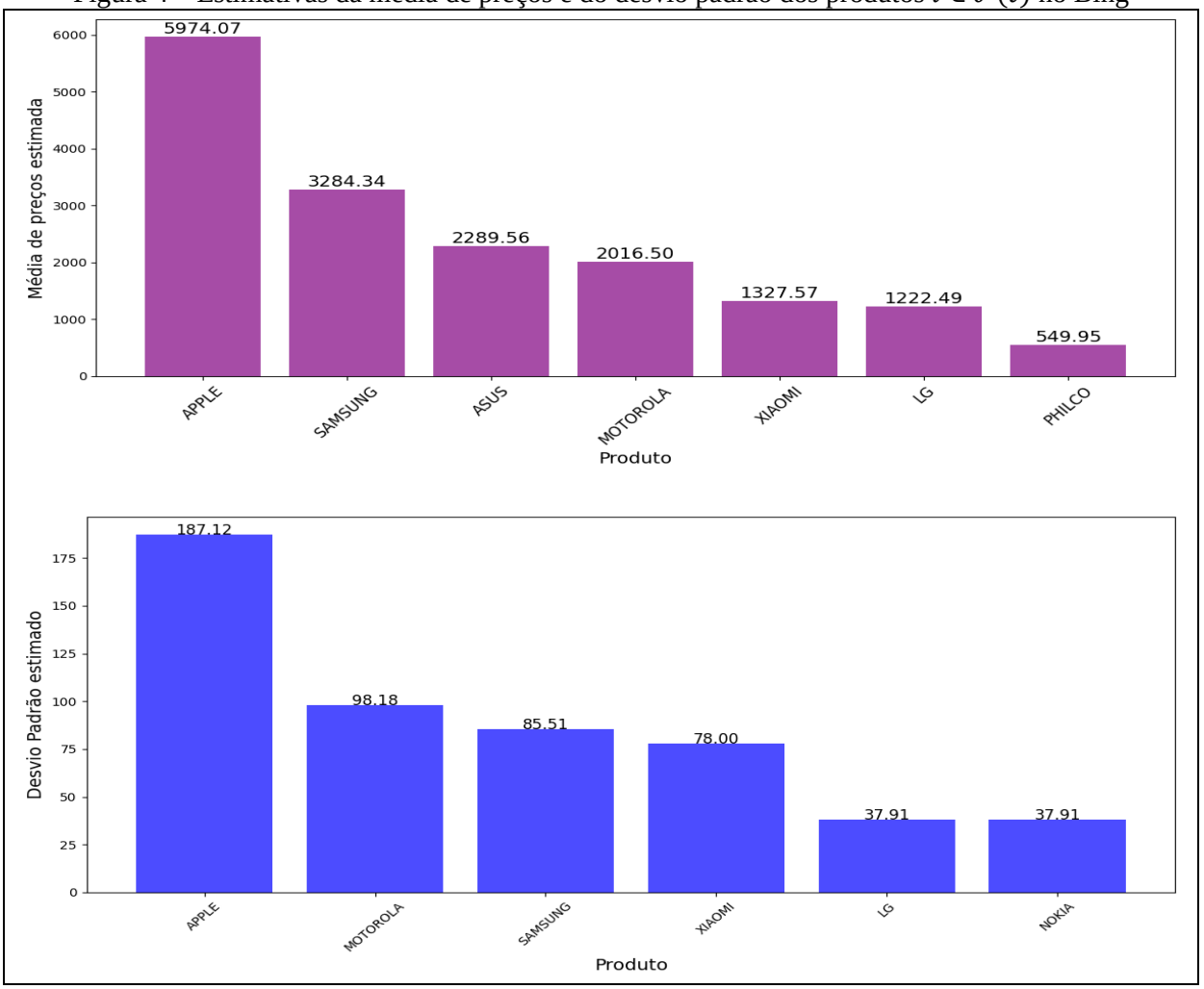
Em relação às estimavas das médias de preços para cada produto $i \in \mathcal{P}(t)$ em cada plataforma $r \in \mathcal{R}(t)$, neste experimento utilizamos o estimador $\hat{\bar{Y}}_i^r(t)$ descrito em (13), que segue um plano de amostragem CRB. Este estimador é composto pelas estimativas de duas médias, $\hat{\bar{Y}}_{i\_1}^r(t)$ e $\hat{\bar{Y}}_{i\_2}^r(t)$, derivadas das amostras $S_{i\_1}^r(t)$ (captura), de tamanho $v_{i\_1}^r(t)$, e $S_{i\_2}^r(t)$ (recaptura), de tamanho $v_{i\_2}^r(t)$. Como estamos assumindo que os resultados destas duas pesquisas se referem a amostras de Bernoulli selecionadas a partir de um mesmo conjunto de dados, o estimador $\hat{\bar{Y}}_i^r(t)$ representa a média dessas duas médias. Ademais, utilizamos o estimador $\widehat{Var}\left(\hat{\bar{Y}}_i^r(t)\right)$, descrito na Expressão (16), para estimar e avaliar as variâncias dos estimadores de médias de preços ofertados pelas lojas $j \in \mathcal{V}_i^r(t)$, de modo que $\widehat{Var}\left(\hat{\bar{Y}}_i^r(t)\right) = \frac{1}{4}\widehat{Var}_{BE}\left\{\left[\hat{\bar{Y}}_{i\_1}^r + \hat{\bar{Y}}_{i\_2}^r\right]\right\}$.

Assim, aplicamos, inicialmente, o referido estimador nos dados capturados e recapturados por *web scraping* na plataforma Bing ($r = 1$), para cada produto listado no

Quadro 5. Como pode ser observado na Figura 4, o produto da Apple, nesta plataforma, apresentou a maior média estimada de preços ($\hat{\bar{Y}}_3^1(t)$ = R$ 5.974,07), associada ao maior desvio padrão (R$ 187,12) em comparação aos demais celulares pesquisados. Esta variação pode estar associada à diversidade de atributos do produto que não foram considerados na coleta de preços, como a cor do aparelho, memória RAM e outros fatores que influenciam o preço final.

O celular da Samsung obteve a segunda maior média estimada ($\hat{\bar{Y}}_1^1(t)$ = R$ 3.284,34) no Bing, mas com um desvio padrão menor (R$ 85,51) em relação ao celular da Apple. Ademais, os celulares da Asus e Motorola apresentaram médias estimadas intermediárias, enquanto o produto da Xiaomi, LG e Philco encontraram-se na faixa de celulares com uma média estimada de preços mais acessível.

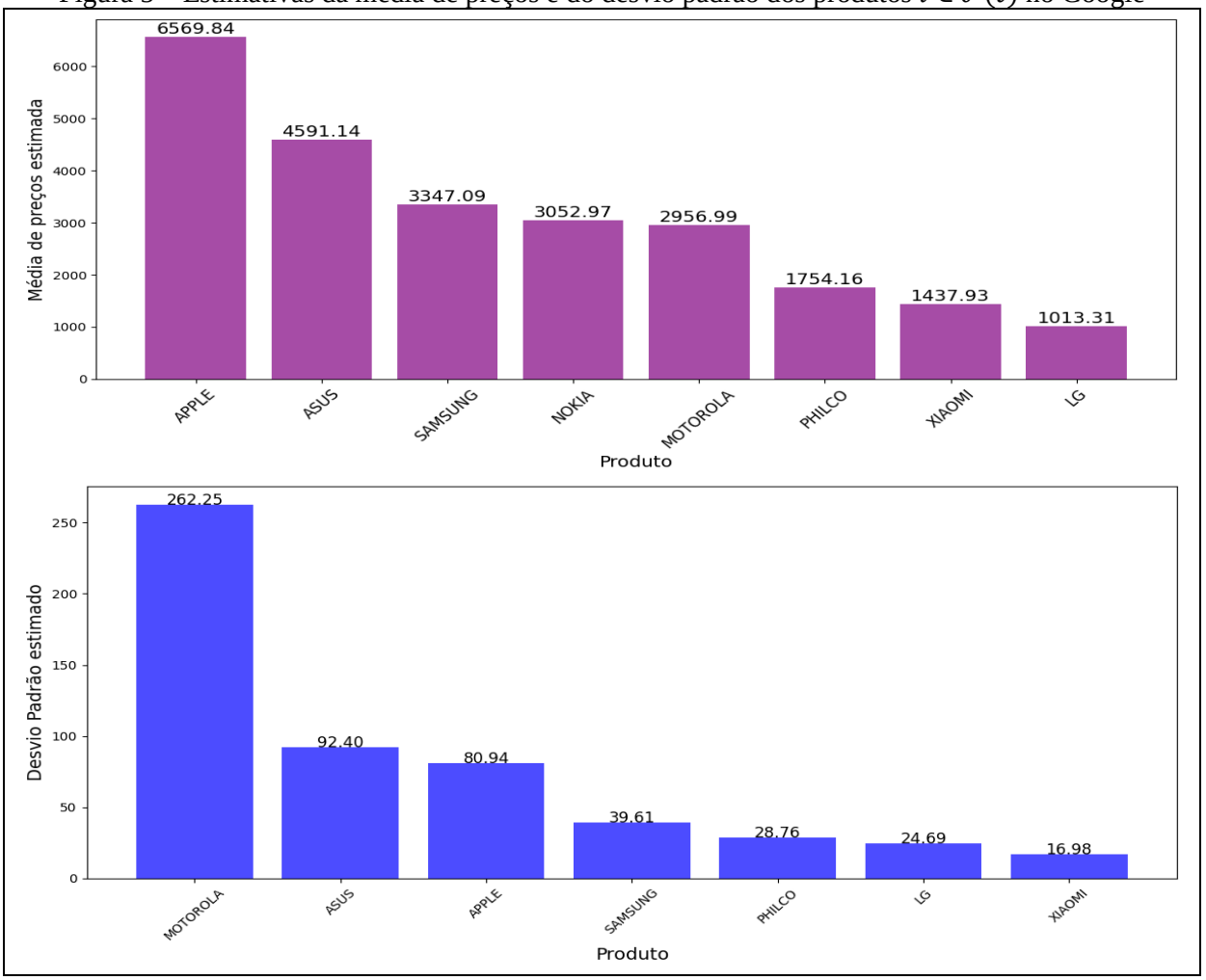
Figura 4 – Estimativas da média de preços e do desvio padrão dos produtos $i \in \mathcal{P}(t)$ no Bing



Fonte: Elaboração da autora (2024).

Na plataforma Google ($r = 2$), conforme destacado na Figura 5, o produto da Apple mantém sua posição de destaque com a maior média estimada de preços ($\hat{\bar{Y}}_3^2(t)$ = R$ 6.569,84), embora com um desvio padrão mais baixo em comparação ao estimado no Bing (R$ 80,94). Por outro lado, o celular da Motorola permanece apresentando uma média estimada intermediária de preços, juntamente com os celulares da Samsung e Nokia, mas com um desvio padrão estimado mais alto (R$ 262,25). Os demais celulares, assim como na Bing, apresentaram estimativas de médias e desvios padrão relativamente baixos. A Xiaomi, por exemplo, obteve uma média estimada de $\hat{\bar{Y}}_4^2(t)$ = R$ 1.437,93 e um desvio padrão de R$ 16,98, consolidando sua posição como uma opção acessível e mais estável no mercado.

Figura 5 – Estimativas da média de preços e do desvio padrão dos produtos $i \in \mathcal{P}(t)$ no Google



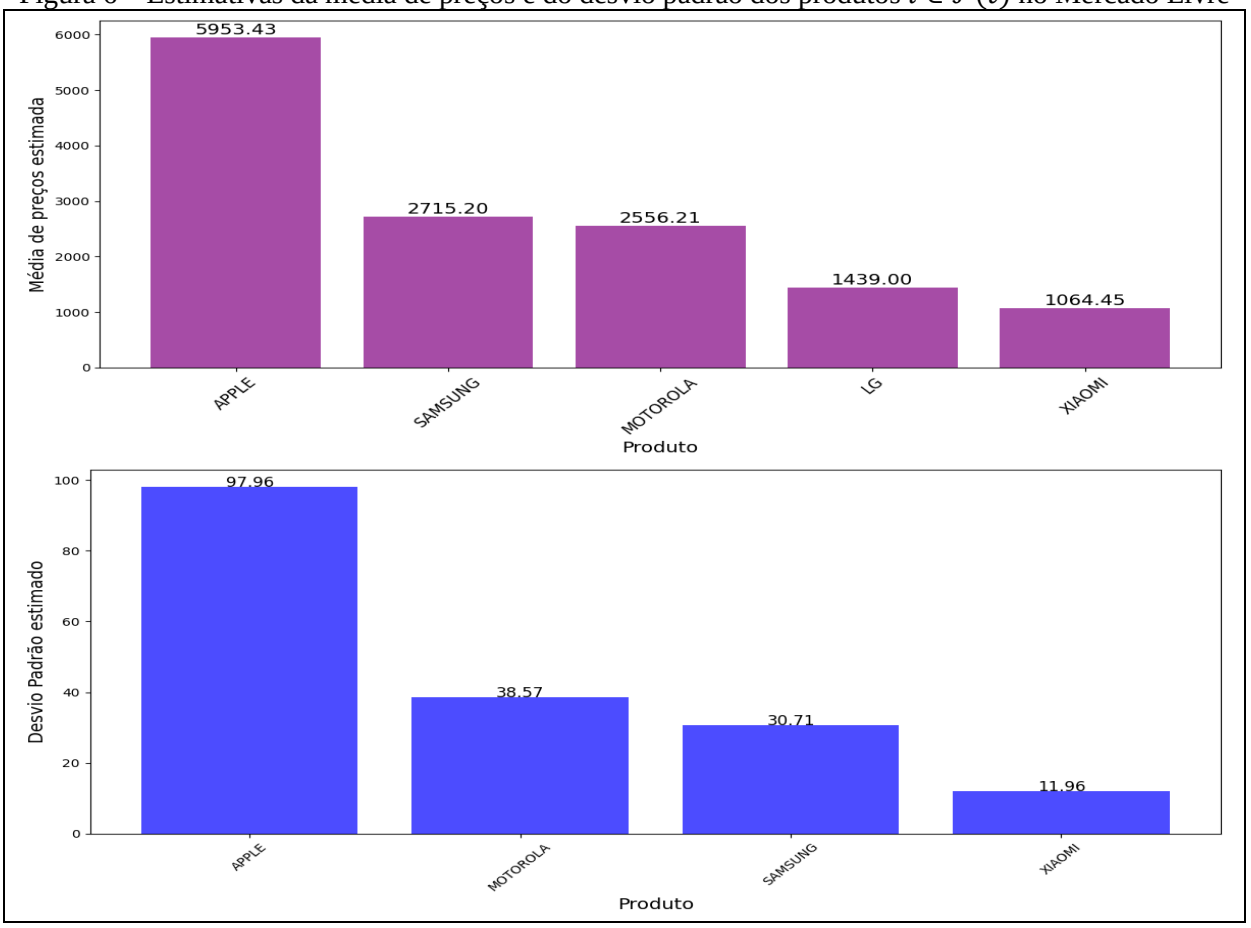
Fonte: Elaboração da autora (2024).

No caso do Mercado Livre ($r = 3$), a Figura 6 evidencia que o produto da Apple continua se destacando com a maior média estimada ($\hat{\bar{Y}}_3^3(t)$ = R$ 5.953,43), acompanhada de um desvio padrão estimado de R$ 97,96. Em seguida, o preço médio estimado do produto da

Samsung ($\hat{\bar{Y}}_1^3(t) = \text{R\$ } 2.715{,}20$) e da Motorola ($\hat{\bar{Y}}_2^3(t) = \text{R\$ } 2.556{,}21$) caracterizaram-se como intermediários, mantendo, em geral, o padrão observado nas demais plataformas analisadas. No entanto, no Mercado Livre, o celular da Samsung e da Motorola apresentaram desvios padrão menores (R$ 30,51 e R$ 38,57, respectivamente) em comparação ao da Apple, sugerindo uma maior consistência nos preços. Em contrapartida, o produto da LG obteve desvio padrão nulo, associado a um tamanho de captura mais restrito, possivelmente devido à sua saída do mercado de smartphones.

O comportamento das médias de preços dos produtos no Mercado Livre, incluindo a ausência de representatividade dos celulares das marcas Nokia, Asus e Philco em sua base de dados, pode ser atribuído ao foco da plataforma em celulares com maior penetração no mercado. Além disso, observa-se que esta plataforma apresenta médias gerais de preços mais baixas, o que pode ser explicado pela sua característica de anunciar tanto vendas no atacado quanto no varejo.

Figura 6 – Estimativas da média de preços e do desvio padrão dos produtos $i \in \mathcal{P}(t)$ no Mercado Livre



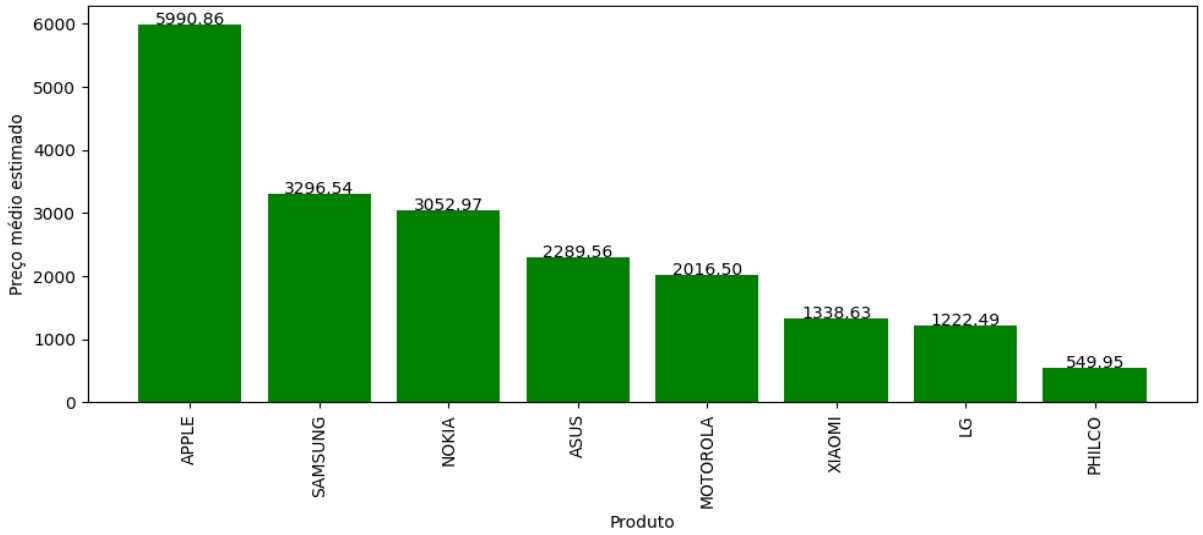
Fonte: Elaboração da autora (2024).

Em resumo, observa-se que as médias de preços dos produtos pesquisados seguem um padrão semelhante entre as plataformas, refletindo a consistência do plano amostral utilizado para gerar as estimativas dos parâmetros descritos na Seção 3.4. No entanto, as discrepâncias observadas podem estar atribuídas às particularidades de cada plataforma.

Assim como apresentamos as estimativas das médias de preços de cada produto $i \in \mathcal{P}(t)$ em cada plataforma $r \in \mathcal{R}(t)$, agora vamos apresentar as estimativas das médias de preços destes produtos em toda a população $\mathcal{V}_i(t)$ de lojas virtuais alcançáveis pelas plataformas de $\mathcal{R}(t)$, utilizando o estimador descrito na Expressão (18). Para isso, foi integrado ao plano amostral anteriormente utilizado a amostragem de cadastros múltiplos, que nos exigiu a identificação da quantidade m_{ij} de plataformas que anunciam a venda do produto $i \in \mathcal{P}(t)$ na loja $j \in \mathcal{V}_i(t)$. Neste experimento, estas quantidades foram identificadas por meio de um algoritmo de análise de dados, permitindo que os estimadores $\hat{\bar{Y}}_{i\_1}(t)$ e $\hat{\bar{Y}}_{i\_2}(t)$, considerados na Expressão (18), fossem baseados na amostragem de cadastros múltiplos.

Portanto, ao calcular a média das estimativas fornecidas por estes estimadores, conforme indicado na referida expressão, chegamos à estimativa da média $\bar{Y}_i(t)$ dos preços dos produtos listados no Quadro 5, ofertados pelas lojas alcançáveis pelo conjunto de plataformas considerado neste experimento. Estes resultados estão apresentados no Gráfico 8, onde é possível observar que eles estão alinhados com as estimativas apresentadas (anteriormente) para cada par de produto e plataforma. Isto é evidenciado, inclusive, pela posição de destaque do celular da Apple, cuja média de preços é a mais alta, alcançando $\hat{\bar{Y}}_3(t) =$ R$ 5.990,86. Além disso, os produtos relacionados às marcas LG, Xiaomi e Philco apresentaram preços médios estimados mais acessíveis, de $\hat{\bar{Y}}_7(t) =$ R$ 1.222,49; $\hat{\bar{Y}}_4(t) =$ R$ 1.338,63 e $\hat{\bar{Y}}_8(t) =$ R$ 549,95, respectivamente. Em contrapartida, os celulares das marcas Samsung, Nokia, Asus e Motorola ocupam uma faixa intermediária, com médias de $\hat{\bar{Y}}_1(t) =$ R$ 3.296,54; $\hat{\bar{Y}}_5(t) =$ R$ 3.052,97; $\hat{\bar{Y}}_6(t) =$ R$ 2.289,56 e $\hat{\bar{Y}}_2(t) =$ R$ 2.016,50, respectivamente.

Gráfico 8 – Estimativas das médias $\bar{Y}_i(t)$ de preços dos produtos $i \in \mathcal{P}(t)$ na população $\mathcal{V}_i(t)$ de lojas virtuais

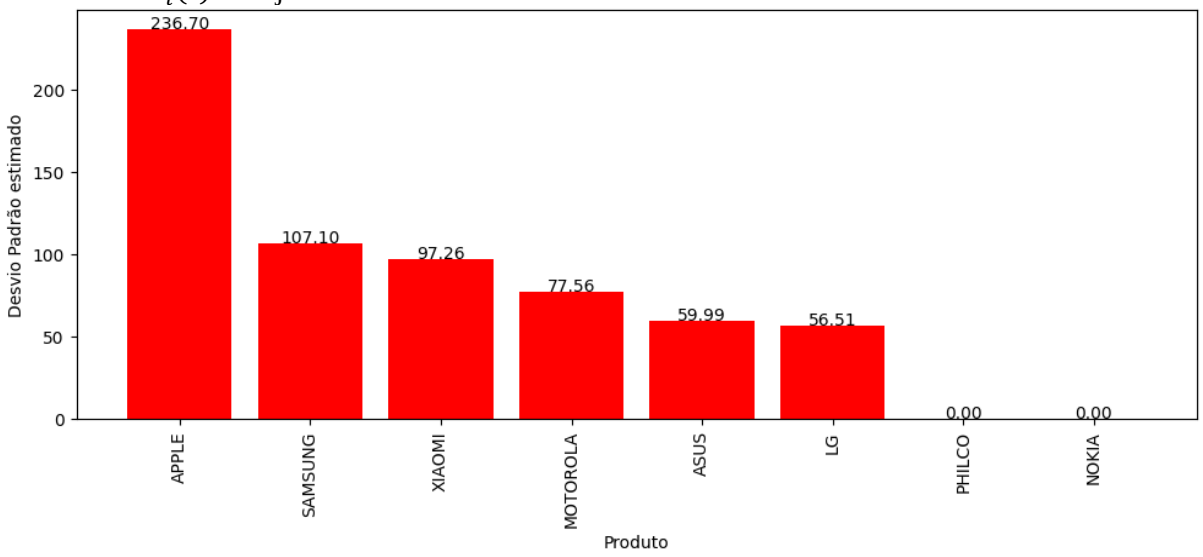


Fonte: Elaboração da autora (2024).

Ademais, para estimar e avaliar os desvios padrão dos preços ofertados pelas lojas $j \in \mathcal{V}_i(t)$; $i \in \mathcal{P}(t)$, foi inicialmente utilizado o estimador $\widehat{Var}_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right)$, descrito na Expressão (23), para estimar a variância da média de preços obtida tanto na captura quanto na recaptura, para o produto $i \in \mathcal{P}(t)$ em todas as plataformas de $\mathcal{R}(t)$. A soma destes estimadores, dividida por 4 (quatro), resulta no estimador $\widehat{Var}\left(\hat{\bar{Y}}_i(t)\right)$, que foi utilizado para gerar as estimativas dos desvios padrão dos preços ofertados para o produto $i \in \mathcal{P}(t)$ nas diferentes lojas virtuais de $\mathcal{V}_i(t)$. Estas estimativas foram apresentadas no Gráfico 9, o qual revela uma variabilidade notável nos preços do celular da Apple, cujo desvio padrão estimado é o mais elevado entre os analisados, alcançando R$ 236,70. Os demais celulares apresentaram desvios padrão menores, como os das marcas Samsung e Xiaomi, que obtiveram desvios padrão de: R$ 107,10 e R$ 97,26, respectivamente.

Por outro lado, marcas como Nokia e Philco exibiram desvios padrão estimados nulos. Este resultado pode ser atribuído à quantidade limitada de anúncios capturados relacionados a estes produtos, resultando em uma uniformidade nos preços registrados.

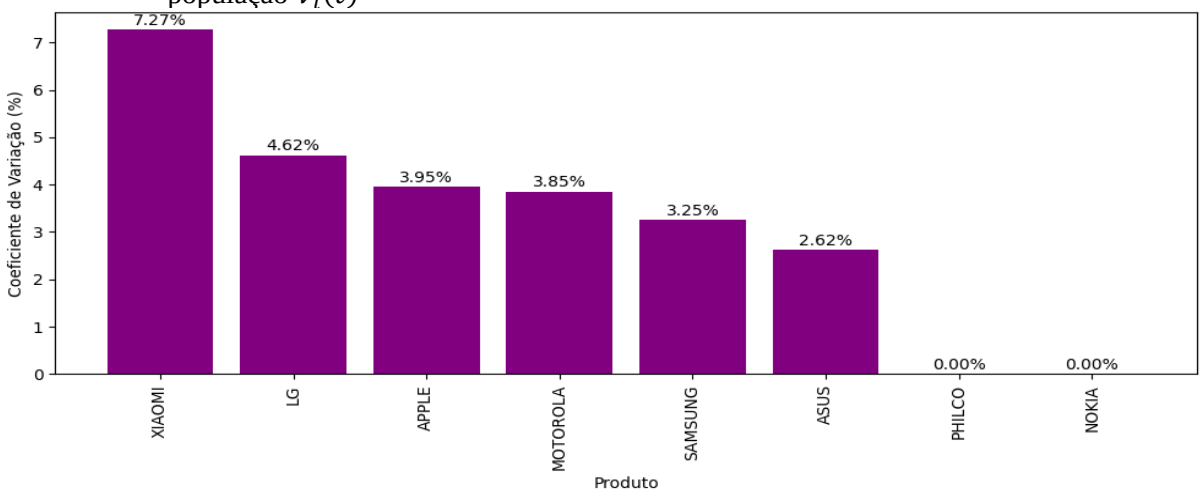
Gráfico 9 – Desvio padrão estimado da média estimada dos preços do produto $i \in \mathcal{P}(t)$ na população $\mathcal{V}_i(t)$ de lojas virtuais



Fonte: Elaboração da autora (2024).

Para complementar a análise, o Gráfico 10 apresenta os valores do Coeficiente de Variação estimado para cada produto $i \in \mathcal{P}(t)$ na população $\mathcal{V}_i(t)$ de lojas virtuais, que corresponde à razão entre o desvio padrão estimado e a média estimada $\widehat{\overline{Y}}_i(t)$, apresentados anteriormente. Valores mais altos, como o referente ao produto da Xiaomi (7,27%), sugeriram maior variabilidade nos preços, enquanto valores baixos, como o do celular da Nokia e o da Philco (0,00%), indicaram uniformidade nos preços. Além disso, os preços dos celulares da Apple, Samsung, Motorola e LG apresentaram variações moderadas, com um coeficiente de variação estimado entre 3,25% e 4,62%, refletindo uma dispersão significativa, mas ainda controlada, nos preços destes produtos.

Gráfico 10 – Coeficiente de variação estimado da média estimada dos preços do produto $i \in \mathcal{P}(t)$ na população $\mathcal{V}_i(t)$



Fonte: Elaboração da autora (2024).

Por fim, temos que o experimento realizado cumpriu os objetivos propostos, desenvolvendo os estimadores apresentados na Seção 3.4 e superando desafios inerentes a esse ambiente. Entre as dificuldades enfrentadas, destacam-se o reconhecimento dos links de cada loja em diferentes plataformas, a captura completa dos resultados de pesquisa por meio de técnicas eficazes de *web scraping* e a classificação adequada dos anúncios coletados, distinguindo-os entre condizentes e não condizentes com o que foi pesquisado. Embora haja espaço para aprimorar essa classificação, a maioria dos obstáculos foi superada com sucesso.

No capítulo seguinte, serão apresentadas as conclusões gerais desta pesquisa, discutindo as implicações desta pesquisa e sugerindo direções para estudos futuros, com o intuito de aprimorar a compreensão e a aplicação de técnicas de *web scraping* em capturas de dados para índices de preços oficiais.

CONCLUSÃO

Esta tese propôs uma abordagem robusta e eficiente para a coleta de dados de preços de lojas virtuais e para a estimação precisa das médias destes preços, com o objetivo de compor índices oficiais. A amostragem por dois estágios desempenhou um papel crucial ao definir os estimadores para as médias de preços de um conjunto de produtos ofertados por uma população de lojas virtuais anunciadas por uma população de plataformas. No primeiro estágio, propõe-se realizar uma seleção de produtos por amostragem aleatória simples, enquanto o segundo estágio utiliza um processo de captura e recaptura de preços, assumindo que os resultados de *web scraping* para uma captura de preços das lojas virtuais $j \in \mathcal{V}(t)$ que vendem os produtos $i \in \mathcal{P}(t)$ seguem uma amostragem de Bernoulli. Além disso, a combinação dessa abordagem com a amostragem por múltiplos cadastros permitiu lidar com o fato de que a população de lojas virtuais está distribuída em plataformas cujas bases de dados apresentam interseções. Essa estratégia possibilitou o desenvolvimento de estimadores aproximadamente não viesados, capazes de compor índices de preços que envolvem exclusivamente lojas virtuais, bem como índices que incluem também lojas físicas.

A estratégia metodológica de *web scraping* desenvolvida nesta pesquisa visou garantir a eficiência na coleta de dados, minimizando erros não amostrais e filtrando informações irrelevantes ou distorcidas, desafios comuns em estudos que utilizam essa abordagem. Para alcançar este objetivo, o uso de bibliotecas de *scraping*, aliado a algoritmos de processamento, manipulação e análise de dados, e integrado a um algoritmo de *machine learning* de classificação, viabilizou uma coleta contínua, escalável e capaz de alcançar os princípios de "Valor" (dados úteis e relevantes), "Verdade" (dados que condizem com a realidade da população-alvo) e "Visualização" (dados organizados para uso e apresentação clara), definidos no conceito de *Big data*.

Para estudos futuros, pretende-se aperfeiçoar os estimadores e expandir o índice proposto para abranger outros tipos de produtos, ampliando sua aplicabilidade. Também se propõe o desenvolvimento de uma metodologia de ponderação para índices de preços em lojas virtuais, similar à utilizada em índices oficiais de lojas físicas, considerando variáveis específicas do mercado virtual, como acessos a links de venda. Além disso, a combinação de índices de lojas físicas e virtuais representa uma oportunidade de avançar na construção de uma visão mais ampla e integrada sobre a ponderação de preços, de forma que a ponderação das lojas físicas pode servir como base para a ponderação dos preços das lojas virtuais.

Uma outra sugestão para trabalhos futuros é a utilização de técnicas de classificação mais eficientes no algoritmo empregado, como a técnica *fuzzy* e algoritmos de *machine learning* não supervisionado de classificação.

Em síntese, esta tese representa um marco no uso de técnicas estatísticas para coleta e análise de dados em *Big data*, demonstrando o potencial de combinar métodos probabilísticos e *web scraping* na construção de índices de preços robustos e adaptáveis à realidade das lojas virtuais.

REFERÊNCIAS

BETHLEHEM, J. G. **Applied survey methods**: a statistical perspective. New Jersey: John Wiley & Sons, 2009.

BRETON, R. *et al*. Research indices using web scraped data. **Office for National Statistics UK**, p. 1-22, 2015. Disponível em: Session_2_UK_Research_indices_using_web_scraped_data.pdf. Acesso em: 27 abr. 2023.

BRETON, R. *et al*. Research indices using web scraped data: **Office for National Statistics UK**, p. 1-34, 2016. Disponível em: https://www.semanticscholar.org/paper/Research-indices-using-web-scraped-data%3A-May-2016-Breton-Flower/a04fe8ebd3e4b5dbf2bc281ce793c3faa00ead74. Acesso em: 14 abr. 2023.

BUTLER, J. Visual web page analytics. **Patent Application Publication**, 2008. Disponível em: https://patentimages.storage.googleapis.com/e3/69/7e/49022cba38485d/US20080046562A1.pdf. Acesso em: 06 jan. 2024.

CAVALLO, A. Are online and offline prices similar? evidence from large multi-channel retailers. **American Economic Review**, v. 107, n. 1, p. 283-303, 2017. Disponível em: https://patentimages.storage.googleapis.com/e3/69/7e/49022cba38485d/US20080046562A1.pdf. Acesso em: 22 fev. 2023.

CAVALLO, A.; RIGOBON, R. The billion prices project: using online prices for measurement and research. **Journal of Economic Perspectives**, v. 30, n. 2, p. 151-78, 2016. Disponível em: https://www.aeaweb.org/articles?id=10.1257/jep.30.2.151. Acesso em: 24 fev. 2023

CHESSA, A. G.; VERBURG, J.; WLLENBORG, L. A comparison of price index methods for scanner data. **5th meeting of the Ottawa Group**, Eltville, p. 9-12, mai. 2017. DOI: 10.13140/RG.2.2.25673.16484.

DA CUNHA, Pedro André Pereira. **Centroproduto**: Fluxo de Verificação do Software. 2018. Dissertação de Mestrado. Universidade de Aveiro, Portugal, 2018.

ELLISON, G., ELLISON, S. Search, obfuscation, and price elasticities on the internet. **Econometrica**, v. 77, n. 2, p. 427-452, 2009. DOI:10.3982/ECTA5708.

GREENLEES, J. S.; MCCLELLAND, R. Superlative and regression-based consumer price indexes for apparel using US scanner data. *In*: CONFERENCE OF THE INTERNATIONAL ASSOCIATION FOR RESEARCH IN INCOME AND WEALTH, St. Gallen, Switzerland, 2010.

HAAN J. de; HENDRIKS, R. Online data, fixed effects and the construction of high-frequency price indexes. Economic Measurement Group Workshop, p. 28-29, 2013. DOI: 10.1007/s11135-024-01848-3.

HAAN J. de; KRSINICH, Frances. Scanner data and the treatment of quality change in nonrevisable price indexes. **Journal of Business & Economic Statistics**, v, 32, n. 3, 341-358, 2014. DOI: 10.1080/07350015.2014.880059.

IBGE - INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Para compreender o INPC: um texto simplificado**. IBGE, Coordenação de Índices de Preços. - 7. ed. - Rio de Janeiro: IBGE, 2016. 62 p.

IBGE - INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Sistema Nacional de Índices de Preços ao Consumidor: Estruturas de ponderação a partir da Pesquisa de Orçamentos Familiares 2017-2018**. IBGE, Coordenação de Índices de Preços. Rio de Janeiro: IBGE, 2020. 215 p. - (Relatórios metodológicos, ISSN 0101-2843; v. 46).

IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **INPC - Índice Nacional de Preços ao Consumidor**. Rio de Janeiro: IBGE, 2023a. Disponível em: < https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9258-indice-nacional-de-precos-ao-consumidor.html?=&t=conceitos-e-metodos>. Acesso em: 25 mai. 2023.

IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **IPCA - Índice Nacional de Preços ao Consumidor Amplo**. Rio de Janeiro: IBGE, 2023b. Disponível em: < https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9256-indice-nacional-de-precos-ao-consumidor-amplo.html?=&t=conceitos-e-metodos>. Acesso em: 25 mai. 2023.

IVANCIC, L.; DIEWERT, W. E.; FOX, K. J. Scanner data, time aggregation and the construction of price indexes. **Journal of Econometrics**, v. 161, n. 1, p. 24-35, 2011. DOI: https://doi.org/10.1016/j.jeconom.2010.09.003.

KONNY, C. G.; WILLIAMS, B. K.; FRIEDMAN, D. M. Big data in the US Consumer. *In*: ABRAHAM, K. G. **Big data for Twenty-First-Century Economic Statistics**. Chicago: University of Chicago Press, p. 69-98, 2022. Disponível em: https://www.degruyter.com/document/doi/10.7208/chicago/9780226801391-004/pdf?licenseType=restricted. Acesso em: 26 ago. 2024.

KRSINICH, F. Measuring the price movements of used cars and residential rents in the New Zealand consumers price index. **Statistics**. 2TH MEETING OF THE OTTAWA GROUP, Wellington, New Zealand, 2011. DOI: 10.13140/2.1.2114.9442.

KRSINICH F. The FEWS Index: fixed effects with a Window Splice. **Journal of Official Statistics**. v. 32, n. 2, p. 375-404, 2016. DOI: https://doi.org/10.1515/jos-2016-0021.

LAVALLÉE, P.; RIVEST, L. P. Capture–recapture sampling and indirect sampling. **Journal of Official Statistics**, v. 28, n. 1, p. 1-27, 2012. Disponível em: https://www.researchgate.net/publication/287746147_Capture-Recapture_Sampling_and_Indirect_Sampling. Acesso em: 12 jan. 2023.

LOON, K. V.; ROELS, D. Integrating big data into the Belgian CPI. **Stats**. Artigo. MEETING OF THE GROUP OF EXPERTS ON CONSUMER PRICE INDICES. Geneva, Switzerland, 2018. Disponível em: https://unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.22/2018/Belgium.pdf. Acesso em: 20 fev. 2023.

LYNCH, D.; STANSFIELD, M.; OLIVECRONA, S. Creating a digital food price index from Web Scraped Data. **Stats**. Artigo. OTTAWA GROUP CONFERENCE MEETING, 2019. Wellington, New Zealand. Disponível em:

https://eventos.fgv.br/sites/eventos.fgv.br/files/arquivos/u161/paul_pascoe_fiona_smillie_-_paper.pdf. Acesso em: 12 ago. 2023.

MACHADO, M. D. dos S.; CRISPIM, S. F. Diferenças no composto varejista de lojas físicas e virtual da mesma rede. **Revista de Administração Contemporânea**, v. 21, n. 02, 2017. DOI: 10.1590/1982-7849rac2017150295.

MECATTI, F. (2007). A single frame multiplicity estimator for multiple frame surveys. Survey Methodology, 33, 151-58.

MECATTI, F.; FERRAZ, C.. Multiple Frames Surveys: a promising tool for agricultural statistics. In: 60th ISI World Statistics Congress ISI2015, 2015, Rio de Janeiro, Brasil. Proceedings of the 60th World Statistics Congress of the International Statistical Institute, ISI2015, 2015. p. 0-5.

PLANTE, N.; RIVEST, L. P.; TREMBLAY, G. Stratified capture recapture estimation of the size of a closed population. **Biometrics**, v. 54, p. 47-60, 1998. DOI: 10.2307/2533994

POLIDORO F. *et. al*. Web scraping techniques to collect data on consumer electronics and airfares for Italian HICP compilation. Statistical Journal of the IAOS, v. 31, p. 165-176, 2015. DOI: 10.3233/SJI-150901.

RICHARDSON, L. **Beautiful soup documentation**. 2007.

RIGATTI, S. J. Random Forest. **Journal of Insurance Medicine**, v. 47, n. 1, p. 31-39, 2017. DOI: https://doi.org/10.17849/insm-47-01-31-39.1.

SÄRNDAL, C. E.; SWENSSON, B.; WRETMAN, J. *Model assisted survey sampling*. New York: Springer-Verlag, 1992.

SEBER, G.A.F. **The estimation of animal abundance and related parameters**. 2. ed. London: Griffin, 1982.

SILVA JUNIOR, D. J.; HUZAR, V. Marketing digital. **Administração**, p. 12-12, 2020. Disponível em: http://repositorio.ucpparana.edu.br/index.php/adm/issue/view/12. Acesso em: 21 set. 2023.

SILVER, M.; HERAVI, S. A failure in the measurement of inflation: results from a hedonic and matched experiment using scanner data. **Journal of Business & Economic Statistics**, v. 23, n. 3, p. 269-281, 2005. Disponível em: https://www.jstor.org/stable/27638820. Acesso em: 25 set. 2023.

SODIAN, L. *et al*. Concurrency and parallelism in speeding up I/O and CPU-Bound tasks in Python 3.10. *In*: 2ND INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE, ELECTRONIC INFORMATION ENGINEERING AND INTELLIGENT CONTROL TECHNOLOGY (CEI). Nanjing, China, p. 560-564, 2022. DOI: 10.1109/CEI57409.2022.9950068.

TEIXEIRA JÚNIOR, Antônio Etevaldo. **Amostragem Bidimensional: Desenvolvimento e Aplicações**. 2020. Tese (Doutorado em Estatística) - Universidade Federal de Pernambuco, Recife, 2020.

TEN BOSCH, O. *et al*. Web scraping meets survey design: combining forces. *In*: BIGSURV18 CONFERENCE, p. 1-13, 2018.

THOMPSON, S.K. **Sampling**. 2. ed. New York: John Wiley and Sons, 2002.

URIARTE, J. I. *et. al*. Web scraping based online consumer price index: the “IPC Online” case. **Journal of Economic and Social Measurement**, v. 44, p. 141-159, 2019. DOI: https://doi.org/10.3233/JEM-190464.

VARGIU, E.; URRU, M. Exploiting web scraping in a collaborative filtering-based approach to web advertising. **Artificial Intelligence Research**, v. 2, n. 1, 2013. DOI: https://doi.org/10.5430/air.v2n1p44.

WILLIAMS, B.; SAGER, E. A New vehicles transaction price index: offsetting the effects of price discrimination and product cycle bias with a year-over-year index. **Economic Working Papers**, n. 514, Bureau of Labor Statistics, 2019. Disponível em: https://ideas.repec.org/p/bls/wpaper/514.html. Acesso em: 12 out. 2023.

YI, J.; NASUKAWA, T.; BUNESCU, R.; NIBLACK, W. Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques. *In*: THIRD IEEE INTERNATIONAL CONFERENCE ON DATA MINING, Melbourne, FL, USA, p. 427–434, 2003.

ZHAO, B. Web scraping. **Encyclopedia of Big Data**, p. 1-3, 2017. DOI: 10.1007/978-3-319-32001-4_483-1.

APÊNDICE A

Como a ideia central da aproximação de Taylor é aproximar uma função não linear usando uma expansão linear (primeira ordem), usamos a expansão de Taylor para linearizar $\hat{\bar{Y}}_i(t)^{CM}$ ao redor de $\bar{Y}_i(t)$, e assim provar as propriedades do estimador descrito na Expressão (19). Sabendo que $\bar{Y}_i(t)$ depende de t_y (que corresponde $Y_i(t)$) e de t_1 (que corresponde V_{it}), temos formalmente que:

$$\bar{Y}_i(t) = f(t_y, t_1) = \frac{t_y}{t_1},$$

em que:

$$t_y = Y_i(t) = \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)} \text{ e } t_1 = V_{it} = \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \frac{1}{m_{ij}(t)}.$$

Como a fórmula básica de Taylor para funções multivariadas é:

$$f(x) \approx f(a) + f'(a)(x - a),$$

em que $f(x)$ é a função original, $f'(a)$ é a derivada da função em a, e $x - a$ é a variação ao redor de a, para $\hat{\bar{Y}}_i(t)^{CM} = f(\hat{t}_y, \hat{t}_1) = \frac{\hat{t}_y}{\hat{t}_1}$, usando $\hat{t}_y \approx t_y$ e $\hat{t}_1 \approx t_1$, a expansão linearizada pode ser dada por:

$$\hat{\bar{Y}}_i(t)^{CM} \approx \bar{Y}_i(t) + \frac{1}{t_1}(\hat{t}_y - \bar{Y}_i(t)\hat{t}_1)$$

Como os estimadores $\hat{t}_y$ e $\hat{t}_1$ são, respectivamente, não viesados para t_y e t_1, dados por:

$$\hat{t}_y = \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \frac{y_{ij}(t)}{m_{ij}(t)\pi_{r|i}(t)},$$

$$\hat{t}_1 = \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \frac{1}{m_{ij}(t)\pi_{r|i}(t)},$$

substituímos essas expressões na expansão de Taylor para calcular a variação ao redor do valor esperado:

$$\hat{\bar{Y}}_i(t)^{CM} \approx \bar{Y}_i(t)_L = \bar{Y}_i(t) + \frac{1}{t_1}\left(\hat{t}_y - \bar{Y}_i(t)\hat{t}_1\right)$$

$$= \bar{Y}_i(t) + \frac{1}{V_{it}} \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \left[\left(\frac{y_{ij}(t)}{m_{ij}(t)}\right) - \bar{Y}_i(t)\frac{1}{m_{ij}(t)}\right]\frac{1}{\pi_{r|i}(t)}$$

$$= \bar{Y}_i(t) + \frac{1}{V_{it}} \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \left[\left(\frac{y_{ij}(t)}{m_{ij}(t)}\right) - \bar{Y}_i(t)\frac{1}{m_{ij}(t)}\right]\frac{1}{\pi_{r|i}(t)}\mathcal{I}_{ij}^r(t).$$

Assim, a aproximação garante que $\hat{\bar{Y}}_i(t)^{CM}$ é aproximadamente não viesado, ou seja:

$$E_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right) = E_{BE}(\bar{Y}_i(t)_L)$$

$$= \bar{Y}_i(t) + \frac{1}{V_{it}} \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \bar{Y}_i(t)\frac{1}{m_{ij}(t)}\right]\frac{1}{\pi_{r|i}(t)}E_{BE}\left(\mathcal{I}_{ij}^r(t)\right)$$

$$= \bar{Y}_i(t) + \bar{Y}_i(t) - \bar{Y}_i(t) = \bar{Y}_i(t).$$

A variância do estimador é analisada com a fórmula derivada:

$$\therefore Var_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right) = Var_{BE}(\bar{Y}_i(t)_L)$$

$$= \frac{1}{V_{it}^2} \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \bar{Y}_i(t)\frac{1}{m_{ij}(t)}\right]^2 \frac{1}{\pi_{r|i}^2(t)}V_{BE}\left(\mathcal{I}_{ij}^r(t)\right)$$

$$= \frac{1}{V_{it}^2} \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \bar{Y}_i(t)\frac{1}{m_{ij}(t)}\right]^2 \frac{1}{\pi_{r|i}^2(t)}\pi_{r|i}(t)\left(1 - \pi_{r|i}(t)\right)$$

$$= \frac{1}{V_{it}^2} \sum_{r\in\mathcal{R}(t)} \sum_{j\in\mathcal{V}_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \bar{Y}_i(t)\frac{1}{m_{ij}(t)}\right]^2 \left(\frac{1}{\pi_{r|i}(t)} - 1\right).$$

$$\therefore \widehat{Var}_{BE}\left(\hat{\bar{Y}}_i(t)^{CM}\right) = \widehat{Var}_{BE}(\bar{Y}_i(t)_L)$$

$$= \frac{1}{\hat{V}_{it}^2} \sum_{r\in\mathcal{R}(t)} \sum_{j\in S_i^r(t)} \left[\frac{y_{ij}(t)}{m_{ij}(t)} - \hat{\bar{Y}}_i(t)^{CM}\frac{1}{m_{ij}(t)}\right]^2 \frac{1}{\pi_{r|i}(t)}\left(\frac{1}{\pi_{r|i}(t)} - 1\right).$$

APÊNDICE B

Abaixo temos o código utilizado para a execução do *web scraping* proposto nesta tese.

```
#Importação dos pacotes utilizados

import requests
from bs4 import BeautifulSoup
import os
import datetime as dt
from datetime import date, time
from datetime import datetime
import numpy as np
from IPython.display import display
import asyncio
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
from time import sleep
import soupsieve as sv
import random
from random import sample,seed
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service as ChromeService
from webdriver_manager.chrome import ChromeDriverManager
from urllib.parse import urljoin, parse_qs, urlparse, unquote
import certifi
from requests.adapters import HTTPAdapter
from requests.packages.urllib3.util.retry import Retry
import socket
from requests.packages.urllib3.exceptions import InsecureRequestWarning
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import joblib
from concurrent.futures import ThreadPoolExecutor
import urllib.parse

from concurrent.futures import ThreadPoolExecutor, as_completed

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', None)


# DATAFRAME COM OS ATRIBUTOS DE CADA PRODUTO QUE SERÁ PESQUISADO

ProdutoCod = pd.read_excel('Pasta.xlsx',na_filter=['NaN'])

# DEFINIÇÃO DOS LINKS DE PESQUISA DE TODAS AS PLATAFORMAS ONDE SERÃO UTILIZADOS OS ATRIBUTOS DO DATAFRAME ACIMA

ProdutoCod['MercLivreCod'] = 'https://lista.lojalivre.com.br/celulares-telefones/celulares-smartphones/' +\
                        ProdutoCod['MARCA']+'-'+\
                        ProdutoCod['LINHA'].map(lambda x: x.replace(' ','-')+'-' if str(x)!='nan' else '') +\
                        ProdutoCod['MODELO'].map(lambda x: x.replace(' ','-') if str(x)!='nan' else '')+\
                        '_NoIndex_True'
```

```
ProdutoCod['BingCod'] = 'https://www.bing.com/shop?q=' +\
                        ProdutoCod['TIPO DO APARELHO'].map(lambda x: x.replace(' ',"))+'+'+\
                        ProdutoCod['MARCA']+'+'+\
                        ProdutoCod['LINHA'].map(lambda x: x.replace(' ','+')+'+' if str(x)!='nan' else '') +\
                        ProdutoCod['MODELO'].map(lambda x: x.replace(' ','+') if str(x)!='nan' else '')

ProdutoCod['GoogleCod'] = 'https://www.google.com.br/search?q=' +\
                        ProdutoCod['TIPO DO APARELHO'].map(lambda x: x.replace(' ',"))+'+'+\
                        ProdutoCod['MARCA']+'+'+\
                        ProdutoCod['LINHA'].map(lambda x: x.replace(' ','+')+'+' if str(x)!='nan' else '') +\
                        ProdutoCod['MODELO'].map(lambda x: x.replace(' ','+') if str(x)!='nan' else '')+\
                        '&source=lnms&tbm=shop&as'

# WEB SCRAPING - MERCADO LIVRE

vetorMercLivreCod =ProdutoCod['MercLivreCod']

headers={'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.0.0 Safari/537.36'}

tabela = []
seed(2023)

for link in vetorMercLivreCod:

    flag = True
    i = 0
    while flag:

        if i !=0:
            text = '_Desde_'+str(i*50+1)+'_NoIndex_True'
            linknew = link.replace('_NoIndex_True',text)
        else:
            linknew = link

        response=requests.get(linknew,headers=headers)
        soup= BeautifulSoup(response.content, 'html.parser')
        caixas=soup.find_all('li',class_='ui-search-layout__item')

        for caixa in caixas:

            produto  = caixa.find('h2',class_='ui-search-item__title')
            preco = caixa.find('span',class_='andes-money-amount ui-search-price__part ui-search-price__part--medium andes-
money-amount--cents-superscript')
            mercado = caixa.find('a').get('href')

            try:
                nome_mercado = caixa.find('p',class_='ui-search-official-store-label ui-search-item__group__element ui-search-
color--GRAY').get_text().replace('por ','')
            except:
                nome_mercado = None

            if nome_mercado == None or not isinstance(nome_mercado, str) :
                urlProduto = caixa.find('a',class_='ui-search-item__group__element').get_attribute_list('href')[0]
                responseProduto = requests.get(urlProduto,headers=headers)
                soupProduto= BeautifulSoup(responseProduto.content, 'html.parser')
                try:
                    nome_mercado = soupProduto.find('a',attrs={'class':"ui-pdp-media__action ui-box-
component__action"}).get_attribute_list('href')[0]
                    nome_mercado= nome_mercado.replace('por ','').replace('https://perfil.mercadolivre.com.br/','').replace('+',' ')
                except:
                    nome_mercado = None
                if nome_mercado == None:
                    try:
                        nome_mercado = soupProduto.find('span',attrs={'class':"ui-pdp-color--BLUE ui-pdp-family--
REGULAR"}).get_text().replace('por ','')
```

```
                except:
                    nome_mercado = None

            row = {"Plataforma":"Mercado Livre"}

            try:
                produtoTransf=produto.get_text()
                row.update({'Produto':produtoTransf})
            except:
                row.update({'Produto':"Não encontrado"})
            try:
                precoTransf= preco.get_text().replace(u'\xa0',u").replace(u'.',u").replace('R$',")
                precoTransf= precoTransf+(',00' if ',' not in precoTransf else ")
                row.update({'Preço':precoTransf})
            except:
                row.update({'Preço':"00,00"})
            try:
                mercadoTransf= mercado
                row.update({'Mercado':mercadoTransf})
            except:
                row.update({'Mercado':"Não encontrado"})
            try:
                mercadoTransf_Filtro = mercado.split('#')[0]
                if mercadoTransf_Filtro:
                    row.update({'Mercado_Filtro': mercadoTransf_Filtro})
                else:
                    row.update({'Mercado_Filtro': "Não encontrado"})
            except:
                row.update({'Mercado_Filtro': "Erro ao processar a URL"})
            try:
                nome_mercadoTransf= nome_mercado.replace('Por ',")
                row.update({'Nome_Mercado':nome_mercadoTransf})
            except:
                row.update({'Nome_Mercado':"Não encontrado"})

            tabela += [row]

        nextPage = soup.find('li',class_='andes-pagination__button andes-pagination__button--next')

        if nextPage != None:
            i = i+1
        else:
            flag = False

dataFrame1 = pd.DataFrame(tabela)

#WEB SCRAPING GOOGLE SHOPPING

vetorGoogleCod = ProdutoCod['GoogleCod']

#Função para encontrar os links das paginas

def get_url_mercado(tmp_link):
    url_mercado_bing = requests.get(tmp_link, headers=headers).url
    return url_mercado_bing

def get_next_page_url(soup):
    """Obtém o URL da próxima página a partir do HTML da página atual."""
    next_page = soup.find('a', id='pnnext')
    if next_page:
        return 'https://www.google.com' + next_page.get('href')
    return None

def scrape_google_shopping(url):
    """Faz o scraping dos resultados do Google Shopping a partir da URL inicial."""
    gravarMercado = []
    gravarProduto = []
```

```
    gravarPreco = []

    while url:
        print(f"Scraping: {url}")
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.text, 'html.parser')

        # Patrocinados
        mercados_patr = soup.find_all('a', class_='shntl sh-np__click-target')
        produtos_patr = soup.find_all(class_='sh-np__product-title')
        precos_patr = soup.find_all(class_='hn9kf')

        for mercado in mercados_patr:
            try:
                mercadoTransf = 'https://www.google.com' + mercado.get('href')
                gravarMercado.append(mercadoTransf)
            except:
                gravarMercado.append('Não Encontrado')

        for produto in produtos_patr:
            try:
                produtoTransf = produto.get_text()
                gravarProduto.append(produtoTransf)
            except:
                gravarProduto.append('Não Encontrado')

        for preco in precos_patr:
            try:
                precoTransf = preco.get_text().split("R$")[1].replace(u'\xa0', u").replace(u'.', u")
                gravarPreco.append(precoTransf)
            except:
                gravarPreco.append('Não Encontrado')

        # Demais Mercados
        mercados = soup.find_all(class_='mnIHsc')
        produtos = soup.find_all(class_='tAxDx')
        precos = soup.find_all(class_='a8Pemb OFFNJ')

        for produto in produtos:
            try:
                produtoTranf = produto.get_text()
                gravarProduto.append(produtoTranf)
            except:
                gravarProduto.append('Não encontrado')
        for mercado in mercados:
            link_tag = mercado.find('a', href=True)
            if link_tag:
                url_mercado_google = link_tag['href'].replace("/url?url=", ")
                gravarMercado.append(url_mercado_google)
            else:
                gravarMercado.append('Não encontrado')
        for preco in precos:
            try:
                precoTransf = preco.get_text().split("R$")[1].replace(u'\xa0', u").replace(u'.', u").replace('Recondicionado',
").replace('+ impostos', ")
                gravarPreco.append(precoTransf)
            except:
                gravarPreco.append('00,00')

        # Obter o URL da próxima página
        url = get_next_page_url(soup)

        # Atraso para evitar ser bloqueado
        sleep(random.uniform(1, 3))

    # Certificando-se de que todas as listas têm o mesmo tamanho
    max_length = max(len(gravarMercado), len(gravarProduto), len(gravarPreco))
```

```
    while len(gravarMercado) < max_length:
        gravarMercado.append('Não Encontrado')
    while len(gravarProduto) < max_length:
        gravarProduto.append('Não Encontrado')
    while len(gravarPreco) < max_length:
        gravarPreco.append('00,00')

    # Criando DataFrame
    tabela = {
        "Plataforma": "Google",
        "Mercado": gravarMercado,
        "Produto": gravarProduto,
        "Preço": gravarPreco}

    dataFrame2 = pd.DataFrame(tabela)
    dataFrame2 = dataFrame2[dataFrame2.Preço != '00,00']
    return dataFrame2

# URL inicial de pesquisa
initial_url = "https://www.google.com/search?q=samsung+s23&tbm=shop"

# Executando o scraping
dataFrame2 = scrape_google_shopping(initial_url)

print(len(dataFrame2))
# Ajustando colunas
for i in range(len(dataFrame2)):
    if dataFrame2.loc[i, 'Mercado'] is not None:
        if 'https://www.google.com/aclk?sa' in dataFrame2.loc[i, 'Mercado']:
            dataFrame2.loc[i, 'Mercado'] = get_url_mercado(dataFrame2.loc[i, 'Mercado'])
            print(f"{i} - {dataFrame2.loc[i, 'Mercado']}")
        else:
            pass

# Criando filtro para alcançar os links “crus”

def extract_market_filter(url):
    for sep in ['?','&','%']:
        if sep in url:
            return url.split(sep)[0]
    return url

dataFrame2['Mercado_Filtro'] =  dataFrame2['Mercado'].apply(extract_market_filter)

# Aplicar a lógica para gerar 'Mercado_Filtro'
for i in range(len(dataFrame2)):
    mercado = dataFrame2.loc[i, 'Mercado']

    if mercado is not None:
        if 'https://validate.perfdrive.com/' in mercado:
            partes = mercado.split("=")
            if len(partes) > 3:
                # Junta tudo após a terceira ocorrência de "="
                mercado = "=".join(partes[3:])
                # Substitui %2F por /
                mercado = mercado.replace('%2F', '/').replace('%3A',':')
                # Remove tudo após o primeiro "&" (caso exista)
                mercado = mercado.split("%")[0]
            else:
                mercado = ''
        dataFrame2.loc[i, 'Mercado'] = mercado
        nome_mercado = mercado.split(".com")[0].replace("https://", "").replace("www.", "").replace("https://www.", "")
        dataFrame2.loc[i, 'Nome_Mercado'] = nome_mercado

        if "zoom" or "buscape" in nome_mercado:
            if r'lead%3Foid%3D' in mercado:
```

```
            dataFrame2.loc[i, 'Mercado_Filtro'] = mercado.replace(r'lead%3Foid%3D', 'lead?oid=').split("%")[0]
        else:
            dataFrame2.loc[i, 'Mercado_Filtro'] = mercado.split("&")[0]
    else:
        dataFrame2.loc[i, 'Mercado'] = ''
        dataFrame2.loc[i, 'Nome_Mercado'] = ''
        dataFrame2.loc[i, 'Mercado_Filtro'] = ''

for i in range(len(dataFrame2)):
    if  dataFrame2.loc[i, 'Nome_Mercado']!='zoom' and dataFrame2.loc[i, 'Nome_Mercado']!="buscape": #and
dataFrame2.loc[i, 'Nome_Mercado']!= "mercadolivre":
        dataFrame2.loc[i, 'Mercado_Filtro'] = dataFrame2.loc[i, 'Mercado_Filtro'].split("?")[0]
        dataFrame2.loc[i, 'Mercado_Filtro'] = dataFrame2.loc[i, 'Mercado_Filtro'].split("%")[0]


#WEB SCRAPING BING SHOPPING

vetorBingCod = ProdutoCod['BingCod']
warnings.simplefilter('ignore', InsecureRequestWarning)

def configure_session():
    session = requests.Session()
    retries = Retry(total=3, backoff_factor=0.3, status_forcelist=[500, 502, 503, 504])
    adapter = HTTPAdapter(max_retries=retries)
    session.mount('https://', adapter)
    session.mount('http://', adapter)
    return session

def check_internet():
    try:
        socket.create_connection(("www.google.com", 80))
        return True
    except OSError:
        return False

def get_page_with_retries(session, url, timeout=10):
    if not check_internet():
        print("Sem conexão com a internet.")
        return None
    try:
        response = session.get(url, headers=headers, verify=certifi.where(), timeout=timeout)
        return response.text
    except requests.RequestException as e:
        print(f"Erro ao acessar a página {url}: {e}")
        return None

def extract_real_url(encoded_url):
    parsed_url = urlparse(encoded_url)
    query_params = parse_qs(parsed_url.query)
    if 'u' in query_params:
        return unquote(query_params['u'][0])
    return encoded_url

def process_page(session, tmp_link):#,is_demais_mercados=False
    page_content = get_page_with_retries(session, tmp_link)
    if page_content is None:
        return []
    soup = BeautifulSoup(page_content, 'html.parser')
    rows = []

    # PATROCINADOS
    caixas_p = soup.find_all(class_='br-offLink')
    for caixa_p in caixas_p:
        row = {"Plataforma": "Bing"}
        produtos_patr = caixa_p.find(class_='br-offTtl')
        if produtos_patr:
            span_produto = produtos_patr.find('span')
```

```
            title_produto = span_produto.get('title') if span_produto else produtos_patr.get_text(strip=True)
        else:
            title_produto = "Não encontrado"
        row['Produto'] = title_produto

        precos_patr = caixa_p.find('div', class_='br-price') or caixa_p.find('div', class_='br-offPrice')
        if precos_patr:
            precoTransf_p = precos_patr.get_text(strip=True).split("R$")[1].replace(u'\xa0', u").replace(u'.', u").replace('agora', ")
        else:
            precoTransf_p = "00,00"
        row['Preço'] = precoTransf_p
        mercados_patr = caixa_p.get('href', 'Não encontrado')
        final_url = requests.get(mercados_patr, headers=headers, verify=False, allow_redirects=True).url
        final_url=extract_real_url(final_url)
        row['Mercado'] = final_url
        rows.append(row)

    # DEMAIS MERCADOS

    caixas = soup.find_all(class_="br-wholeCardClickable br-card br-small br-ocCiteHvrItm br-crdhvr br-tallCrd dsblHvr")
    for caixa in caixas:
        row = {"Plataforma": "Bing"}
        mercado_classe = caixa.find('a', class_='br-compareSellers').get('href')
        mercado_classe_url = urljoin(tmp_link, mercado_classe)
        mercado_page_content = get_page_with_retries(session, mercado_classe_url)
        if not mercado_page_content:
            continue
        mercado_soup = BeautifulSoup(mercado_page_content, 'html.parser')

        # Procurar pela tag <a> com a classe "br-oboSnOptLink" na nova página
        mercado_tag = mercado_soup.find('a', class_='br-oboSnOptLink')
        if mercado_tag:
            mercado_encoded_url = mercado_tag.get('href', 'Não encontrado')  # bing.click
            redirect_response = get_page_with_retries(session, mercado_encoded_url, timeout=15)
            if redirect_response:
                final_url = requests.get(mercado_encoded_url, headers=headers, verify=False, allow_redirects=True).url
                final_url=extract_real_url(final_url)
                row['Mercado'] = final_url
            else:
                row['Mercado'] = 'Não encontrado'
        else:
            row['Mercado'] = 'Não encontrado'

        produto = mercado_soup.find(class_='br-pdTtl')
        preco = mercado_soup.find(class_='br-oboSnDp')
        produtoTransf = produto.get('title', 'Não encontrado')
        row['Produto'] = produtoTransf

        if preco:
            precoTransf = preco.get_text(strip=True).split("R$")[1].replace(u'\xa0', u").replace(u'.', u").replace('agora', ")
        else:
            precoTransf = "00,00"
        row['Preço'] = precoTransf
        rows.append(row)

        # Obter o próximo link
    try:
        next_page_link = soup.find('a', {'aria-label': "Next"}).get('href')
        if next_page_link:
            next_page_url = urljoin('https://www.bing.com/', next_page_link)
            next_page_url=extract_real_url(next_page_url)
            rows.extend(process_page(session, next_page_url))  # Processo para próxima página
    except AttributeError:
        pass
    return rows

# Criar sessão para requisições
```

```
session = configure_session()

# Processar páginas sequencialmente
rows = []

with ThreadPoolExecutor(max_workers=4) as executor:
    futures = []
    for link in vetorBingCod:
        # Enviar duas tarefas para cada link
        futures.append(executor.submit(process_page, session, link))#, is_demais_mercados=False
        futures.append(executor.submit(process_page, session, link))#, is_demais_mercados=True
    for future in as_completed(futures):
        try:
            rows.extend(future.result())
        except Exception as e:
            print(f"Erro ao processar uma das páginas: {e}")

dataFrame3 = pd.DataFrame(rows)

def extract_market_filter(url):
    for sep in ['?', '%', '&']:
        if sep in url:
            return url.split(sep)[0]
    return url

#dataFrame3 = dataFrame3.loc[dataFrame3['Preço'] != '00,00'].reset_index(drop=True)
dataFrame3['Mercado_Filtro'] =  dataFrame3['Mercado'].apply(extract_market_filter)

for i in range(len(dataFrame3)):
    mercado = dataFrame3.loc[i, 'Mercado']
    if mercado is not None:
        if 'https://validate.perfdrive.com/' in mercado:
            partes = mercado.split("=")
            if len(partes) > 3:
                # Junta tudo após a terceira ocorrência de "="
                mercado = "=".join(partes[3:])
                # Substitui %2F por /
                mercado = mercado.replace('%2F', '/').replace('%3A',':')
                # Remove tudo após o primeiro "&" (caso exista)
                mercado = mercado.split("%")[0]
        if 'https://www.bing.com:443/alink/link' in mercado:
            partes = mercado.split("=")
            if len(partes) > 1:
                # Junta tudo após a terceira ocorrência de "="
                mercado = "=".join(partes[1:])
                # Substitui %2F por /
                mercado = mercado.replace('%2F', '/').replace('%3A',':').replace(r'lead%3Foid%3D', 'lead?oid=')
                # Remove tudo após o primeiro "&" (caso exista)
                mercado = mercado.split("%")[0]

        nome_mercado = mercado.split(".com")[0].replace("https://", "").replace("www.", "").replace("https://www.", "")
        dataFrame3.loc[i, 'Nome_Mercado'] = nome_mercado

        if "zoom" or "buscape" in nome_mercado:
            if r'lead%3Foid%3D' in mercado:
                dataFrame3.loc[i, 'Mercado_Filtro'] = mercado.replace(r'lead%3Foid%3D', 'lead?oid=').split("%")[0]
            else:
                dataFrame3.loc[i, 'Mercado_Filtro'] = mercado.split("&")[0]
    else:
        dataFrame3.loc[i, 'Mercado'] = ''
        dataFrame3.loc[i, 'Nome_Mercado'] = ''
        dataFrame3.loc[i, 'Mercado_Filtro'] = ''

for i in range(len(dataFrame3)):
    if  dataFrame3.loc[i, 'Nome_Mercado']!='zoom' and dataFrame3.loc[i, 'Nome_Mercado']!="buscape": #and
dataFrame3.loc[i, 'Nome_Mercado']!= "mercadolivre":
        dataFrame3.loc[i, 'Mercado_Filtro'] = dataFrame3.loc[i, 'Mercado_Filtro'].split("?")[0]
```

```
    dataFrame3.loc[i, 'Mercado_Filtro'] = dataFrame3.loc[i, 'Mercado_Filtro'].split("%")[0]


#GRAVANDO TODOS OS PREÇOS CAPTURADOS POR WEB SCRAPRING EM CADA PLATAFORMA

dataFrame = pd.concat([dataFrame1,dataFrame2,dataFrame3],ignore_index=True)
dataFrame['Marca'] = None

strings_out=["capa","película"]
# Função de contagem de atributos correspondentes
def contar_atributos_correspondentes(texto, atributos):
    return sum(attr in texto for attr in atributos)

# Função para classificar a marca
def classificar_marca(row, produto_cod):
    mercado_filtro = row['Mercado_Filtro']
    produto = row['Produto']

    # Converte o conteúdo das células para letras minúsculas e verifica se não são None
    mercado_filtro = mercado_filtro.lower() if pd.notnull(mercado_filtro) else ''
    produto = produto.lower() if pd.notnull(produto) else ''

    # Separar a palavra "GB" no campo Produto
    produto = produto.replace("gb", " gb")

    # Verificar se alguma das strings em strings_out está presente em "Mercado_Filtro"
    if any(out in mercado_filtro for out in strings_out):
        return "Não Condiz"

    for i in range(len(produto_cod)):
        marca = produto_cod.loc[i, 'MARCA'].lower()

        # Verificar se o nome da marca está presente em "Mercado_Filtro"
        if marca in mercado_filtro or marca in produto:
            atributos = []
            if produto_cod.loc[i, 'MARCA'].capitalize() not in ['Asus', 'Apple']:
                for col in ['LINHA', 'MODELO', 'ARMAZENAMENTO', 'RAM']:
                    string = str(produto_cod.loc[i, col]).lower()
                    if string != 'nan':
                        atributos.append(string)
            else:
                for col in ['LINHA', 'MODELO']:
                    string = str(produto_cod.loc[i, col]).lower()
                    if string != 'nan':
                        atributos.append(string)

            # Verificar se os atributos estão presentes no produto
            if contar_atributos_correspondentes(produto, atributos) == len(atributos):
                return produto_cod.loc[i, 'MARCA']

    return "Não Condiz"

# Aplica a função classificar_marca ao dataFrame
dataFrame['Marca'] = dataFrame.apply(lambda row: classificar_marca(row, ProdutoCod), axis=1)

dataFrame[~dataFrame.Marca.isnull()]
dataFrame = dataFrame[~dataFrame.Marca.isnull() & dataFrame.Preço != "00,00"]
dataFrame['Preço'] = dataFrame.Preço.map(lambda x: str(str(x).replace(',','.').replace('estimativa','')))
data_e_hora_atuais = datetime.now()
data_e_hora_em_texto = data_e_hora_atuais.strftime("%d-%m-%Y %H_%M")
valor_indesejado = "https://click1.mercadolivre.com.br/mclics/clicks/external/MLB/count"
dataFrame = dataFrame[dataFrame['Mercado_Filtro'] != valor_indesejado]

##TREINAMENTO POR MACHINE LEARNING – RANDOM FOREST

df_train=pd.read_excel('TREINAMENTO.xlsx')
# Carregar os dados de treinamento
```

```
# Supondo que você já tenha um DataFrame 'df_train' com colunas 'Produto', 'Mercado_Filtro', e 'Marca'

# Pré-processamento: Combinar 'Produto' e 'Mercado_Filtro' em uma única coluna de texto
df_train['text'] = df_train['Produto'] + ' ' + df_train['Mercado_Filtro']

# Dividir os dados em treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(df_train['text'], df_train['Marca'], test_size=0.2, random_state=42)

# Criar o pipeline
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', RandomForestClassifier(n_estimators=100, random_state=42))])

# Treinar o modelo
pipeline.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = pipeline.predict(X_test)

# Avaliar o modelo
print(classification_report(y_test, y_pred))

# Salvar o modelo treinado
joblib.dump(pipeline, 'modelo_classificacao_marca_rf.pkl')

# Passo 1: Carregar o modelo treinado
pipeline = joblib.load('modelo_classificacao_marca_rf.pkl')

# Passo 2: Carregar os novos dados
# Supondo que você já tenha o DataFrame carregado em 'dataFrame'

# Substituir valores NaN por strings vazias
dataFrame.fillna('', inplace=True)

# Criar a coluna 'text' combinando 'Produto' e 'Mercado_Filtro'
dataFrame['text'] = dataFrame['Produto'] + ' ' + dataFrame['Mercado_Filtro']

# Remover linhas onde 'text' está vazio após a concatenação
dataFrame = dataFrame[dataFrame['text'].str.strip() != '']

# Passo 3: Fazer as previsões
predicoes = pipeline.predict(dataFrame['text'])

# Adicionar as previsões ao DataFrame
dataFrame['Predicao_Marca'] = predicoes

dataFrame['Predicao_Marca_1']=" "

def preencher_coluna_marca(row):
    if row['Predicao_Marca'] == row['Marca']:
        return row['Predicao_Marca']
    elif row['Predicao_Marca'] == 'Não Condiz':
        return row['Marca']
    elif row['Marca'] == 'Não Condiz':
        return row['Predicao_Marca']
    else:
        return 'Ambiguous'

# Aplicar a função linha a linha
dataFrame['Predicao_Marca_1'] = dataFrame.apply(preencher_coluna_marca, axis=1)


# TRATAMENTO DE DADOS E CONTAGEM DE LOJAS ENTRE PLATAFORMAS

# Passo 1: Substituir valores NaN por string vazia e limpar a coluna 'Preço'
dataFrame.fillna('', inplace=True)
```

```
# Passo 2: Converte para string e remove espaços em branco
dataFrame['Preço'] = dataFrame['Preço'].astype(str).str.strip()

# Passo 3: Remove caracteres não numéricos exceto o ponto decimal
dataFrame['Preço'] = dataFrame['Preço'].str.replace(r'[^0-9.]', '', regex=True)

# Passo 4: Remover valores iguais a '00.00'
dataFrame = dataFrame[dataFrame['Preço'] != '00.00']

# Passo 5: Remover valores vazios que podem ter sido criados ao limpar a coluna
dataFrame = dataFrame[dataFrame['Preço'].str.strip() != '']

# 4. Contagem dos duplicados em cada plataforma e Mercado_Filtro
grouped_df_1 = dataFrame.groupby(['Plataforma', 'Mercado_Filtro']).size().reset_index(name='Count_1')
#grouped_df_1.to_excel(f'estimativas/conta_dupl_das_plat/plat-comdupl_{data_e_hora_em_texto}.xlsx', index=False,
header=True)

# 5. Remover duplicados considerando apenas 'Plataforma' e 'Mercado_Filtro'
df_unique = dataFrame.drop_duplicates(subset=['Plataforma', 'Mercado_Filtro'])

# 6. Contar ocorrências de cada valor único em 'Mercado_Filtro'
contagem_mercado = df_unique.groupby(['Mercado_Filtro']).size().reset_index(name='Contagem')

# 7. Adicionar 'Marca' e 'Preço' ao DataFrame 'contagem_mercado'
contagem_mercado = contagem_mercado.merge(dataFrame[['Mercado_Filtro', 'Predicao_Marca_1', 'Predicao_Marca',
'Marca', 'Preço']], on='Mercado_Filtro', how='left').drop_duplicates()

# 8. Criar um dicionário para mapear 'Mercado_Filtro' para 'Plataforma'
mercado_plataforma_dict = df_unique.groupby(['Mercado_Filtro'])['Plataforma'].apply(lambda x: ',
'.join(x.unique())).to_dict()

# 9. Adicionar a coluna 'Plataformas' ao DataFrame 'contagem_mercado'
contagem_mercado['Plataformas'] = contagem_mercado['Mercado_Filtro'].map(mercado_plataforma_dict)

# 10. Remover duplicados do DataFrame 'contagem_mercado'
#contagem_mercado_ = contagem_mercado.drop_duplicates(subset=['Mercado_Filtro', 'Preço', 'Plataformas'])
contagem_mercado['Preço'] = contagem_mercado['Preço'].astype(float)
contagem_mercado['Preço_Truncado'] = contagem_mercado['Preço'].apply(lambda x: int(x))

# 2. Remover duplicados com base na nova coluna 'Preço_Truncado', além de 'Mercado_Filtro' e 'Plataformas'
contagem_mercado_ = contagem_mercado.drop_duplicates(subset=['Mercado_Filtro', 'Preço_Truncado', 'Plataformas'])

# 3. Remover a coluna temporária 'Preço_Truncado', se não for mais necessária
contagem_mercado_ = contagem_mercado_.drop(columns=['Preço_Truncado'])

# 11. Resetar o índice do DataFrame 'contagem_mercado_'
contagem_mercado_.reset_index(drop=True, inplace=True)

contagem_mercado_['Preço'] = contagem_mercado_['Preço'].round(2)
contagem_mercado_ = contagem_mercado_[contagem_mercado_['Preço'] >= 100]
contagem_mercado_ = contagem_mercado_[contagem_mercado_['Mercado_Filtro'].str.startswith('https')]
# 12. Salvar o DataFrame 'contagem_mercado_' em um novo arquivo Excel
contagem_mercado_.to_excel(f'estimativas/dataframes/dataframe_ajustado_recaptura {data_e_hora_em_texto}.xlsx',
index=False, header=True)

#CONCATENAR DATAFRAMES

# Define o diretório base
base_dir = 'estimativas/dataframes'
base_dir_ = 'estimativas/resultados'

# Verifica e cria o diretório base se necessário
os.makedirs(base_dir, exist_ok=True)

def concatenar(caminho):
    juntos = []
    for arquivo in os.listdir(caminho):
```

```
        if arquivo.endswith('.xlsx'):
            caminho_arquivo = os.path.join(caminho, arquivo)
            dt_arq = pd.read_excel(caminho_arquivo)

            # Adiciona colunas para Tipo, Data e Hora
            dt_arq['Tipo'] = ""
            dt_arq['Data'] = ""
            dt_arq['Hora'] = ""

            # Separa o nome do arquivo para extrair tipo, data e hora
            nome_arquivo, extensao = os.path.splitext(arquivo)

            # Remove a extensão e separa por espaço
            partes = nome_arquivo.split(' ')
            if len(partes) == 3:
                tipo = partes[0]
                data = partes[1]
                hora = partes[2].replace('_', ':')

                # Atribui valores ao DataFrame
                dt_arq['Tipo'] = tipo
                dt_arq['Data'] = data
                dt_arq['Hora'] = hora

                print(f'Carregando arquivo {arquivo} . . .')
                juntos.append(dt_arq)
            else:
                print(f'O nome do arquivo {arquivo} não segue o padrão esperado.')

    if juntos:
        df_principal = pd.concat(juntos, axis=0, ignore_index=True)

        df_principal['Repetido'] = 'Falso'

        # Identifique as combinações únicas de 'Mercado_Filtro' e 'Preço' por 'Hora'
        combinacoes = df_principal.groupby(['Mercado_Filtro', 'Preço','Plataformas'])['Hora'].nunique()

        # Encontre combinações que aparecem em mais de uma data
        repetidos = combinacoes[combinacoes > 1].index

        # Atualize a coluna 'Repetido' para 'Verdadeiro' para as combinações repetidas
        for mercado, preco, plataforma in repetidos:
            df_principal.loc[(df_principal['Mercado_Filtro'] == mercado) & (df_principal['Preço'] == preco) &
(df_principal['Plataformas'] == plataforma), 'Repetido'] = 'Verdadeiro'

        return df_principal
    else:
        return None

# Exemplo de uso
df_concatenado = concatenar(base_dir)

# Salvar o DataFrame concatenado em um novo arquivo Excel
if df_concatenado is not None:
    data_e_hora_em_texto = pd.Timestamp.now().strftime('%d-%m-%Y %H_%M')
    df_concatenado=df_concatenado[~df_concatenado['Mercado_Filtro'].str.startswith('https://click1.mercadolivre.com.br/mcl
ics/clicks')]
    df_concatenado.to_excel(f'estimativas/resultados/resultado_{data_e_hora_em_texto}.xlsx', index=False, header=True)
else:
    print("Nenhum arquivo encontrado para processar.")
```

APÊNDICE C

A seguir, apresenta-se o código utilizado para calcular os estimadores aplicados no experimento piloto desta tese (Seção 4), cujas expressões estão descritas na Seção 3.4.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Supondo que 'dataFrame' seja seu DataFrame

# Filtrando os registros de um tempo igual a 1
dataFrame_filtrado = dataFrame[dataFrame['T'] == 1]

# Contando os registros que têm "Não Condiz" na coluna 'i' para cada plataforma em 'r', considerando apenas os registros filtrados
condiz_count = dataFrame_filtrado[dataFrame_filtrado['i'] == 'Não Condiz'].groupby('r').size()
nao_condiz_count = dataFrame_filtrado[dataFrame_filtrado['i'] != 'Não Condiz'].groupby('r').size()

# Criando um DataFrame para facilitar a visualização
df_counts = pd.DataFrame({
    'Não condiz': condiz_count,
    'Condiz': nao_condiz_count
}).fillna(0)  # Caso haja plataformas sem valores em uma das categorias

# Plotando o gráfico
ax = df_counts.plot(kind='bar', stacked=False)

# Adicionando rótulos de dados
for container in ax.containers:
    ax.bar_label(container, label_type='edge', fontsize=10, color='black')

plt.xlabel('Plataforma')
plt.ylabel('Quantidade de Registros')
plt.xticks(rotation=45)
plt.legend(title='Classificação', loc='upper right')

# Exibindo o gráfico
plt.tight_layout()
plt.show()
# Remover registros onde Marca == 'Não Condiz'
dataFrame = dataFrame[dataFrame['i'] != "Não Condiz"]

# Remover duplicatas considerando "Mercado Filtro", "r" (plataforma) e "i" (produto)
#dataFrame = dataFrame.drop_duplicates(subset=['Mercado_Filtro', 'r', 'i'])

# Garantir que as colunas numéricas estejam no formato correto
dataFrame['y_ijr(t)'] = pd.to_numeric(dataFrame['y_ijr(t)'], errors='coerce')
dataFrame['m_ij'] = pd.to_numeric(dataFrame['m_ij'], errors='coerce')

# Substituir valores NaN por valores padrão
dataFrame['y_ijr(t)'].fillna(0, inplace=True)
dataFrame['m_ij'].fillna(1, inplace=True)

# DataFrames para T=1
dataFrame_T1 = dataFrame[dataFrame['Data'].isin(['2024-10-01', '2024-10-02'])]

# Função para calcular captura, recaptura e interseção, e média de y para cada plataforma e i
def calcular_resultados_por_T(df, captura_data, recaptura_data):
    resultados = {'r': [], 'i': [], 'v_ir1': [], 'v_ir2': [], 'v_ir12': []}

    for plataforma in df['r'].unique():
```

```
        for i in df['i'].unique():
            df_plat_prod = df[(df['r'] == plataforma) & (df['i'] == i)]

            # Captura e recaptura
            v_ir1 = df_plat_prod[df_plat_prod['Data'] == captura_data]
            v_ir2 = df_plat_prod[df_plat_prod['Data'] == recaptura_data]

            # Remover duplicatas
           # v_i1 = v_i1.drop_duplicates(subset=['Mercado_Filtro', 'i'])
            #v_i2 = v_i2.drop_duplicates(subset=['Mercado_Filtro', 'i'])

            # Interseção
            v_ir12 = pd.merge(v_ir1, v_ir2, on=['Mercado_Filtro', 'i'], suffixes=('_1', '_2'))

            # Adicionar resultados
            resultados['r'].append(plataforma)
            resultados['i'].append(i)
            resultados['v_ir1'].append(len(v_ir1))
            resultados['v_ir2'].append(len(v_ir2))
            resultados['v_ir12'].append(len(v_ir12))

    return pd.DataFrame(resultados)

# Calcular resultados para T=1
resultados_T1 = calcular_resultados_por_T(dataFrame_T1, '2024-10-01', '2024-10-02')

# Função para organizar e plotar os gráficos de captura, recaptura e interseção
# Função para organizar e plotar os gráficos de captura, recaptura e interseção com rótulos
def plotar_graficos_organizados(resultados_df, T_value):
    for plataforma in resultados_df['r'].unique():
        resultados_plat = resultados_df[resultados_df['r'] == plataforma]

        # Ordenar os produtos (i) por frequência de captura (v_ir1)
        resultados_plat = resultados_plat.sort_values(by='v_ir1', ascending=False)

        plt.figure(figsize=(12, 6))
        x = np.arange(len(resultados_plat['i']))  # posições no eixo x
        largura = 0.25  # largura das barras

        # Barras
        barras_captura = plt.bar(x - largura, resultados_plat['v_ir1'], width=largura, label=f'Captura (T={T_value})', alpha=0.6)
        barras_recaptura = plt.bar(x, resultados_plat['v_ir2'], width=largura, label=f'Recaptura (T={T_value})', alpha=0.6)
        barras_interseção = plt.bar(x + largura, resultados_plat['v_ir12'], width=largura, label=f'Interseção (T={T_value})',
alpha=0.6)

        # Adicionar rótulos nas barras
        for barras, valores in zip([barras_captura, barras_recaptura, barras_interseção],
                                   [resultados_plat['v_ir1'], resultados_plat['v_ir2'], resultados_plat['v_ir12']]):
            for barra, valor in zip(barras, valores):
                plt.text(barra.get_x() + barra.get_width() / 2, barra.get_height() + 0.5,  # Posição do texto
                         f'{int(valor)}', ha='center', va='bottom', fontsize=10)

        # Configurações do gráfico
        plt.title(f'Tamanhos de Captura, Recaptura e Interseção para {plataforma} por Produto (T={T_value})', fontsize=16)
        plt.xlabel('Produto', fontsize=14)
        plt.ylabel('Tamanho', fontsize=14)
        plt.xticks(x, resultados_plat['i'], rotation=45)
        plt.legend()
        plt.tight_layout()
        plt.show()

# Plotar gráficos para T=1 com rótulos
plotar_graficos_organizados(resultados_T1, T_value=1)

# Exibir DataFrames de resultados para T=1
print("Resultados para T=1:")
print(resultados_T1)
```

```
# Função para calcular os estimadores
def calcular_estimadores(df, T):
    resultados_total = []

    # Dicionário para armazenar os valores de π̂(r|i) organizados por plataforma
    resultados_por_plataforma = {}

    for produto in df['i'].unique():
        df_produto = df[df['i'] == produto]

        for plataforma in df_produto['r'].unique():
            df_plataforma = df_produto[df_produto['r'] == plataforma]

            # Selecionar registros para capturas e recapturas
            captura = df_plataforma[df_plataforma['Data'].isin(['2024-10-01'])]
            recaptura = df_plataforma[df_plataforma['Data'].isin(['2024-10-02'])]

            # Cálculo da interseção
            interseccao = pd.merge(captura, recaptura, on=['Mercado_Filtro', 'i'], suffixes=('_1', '_2'))

            # Agora, utilizamos as contagens de mercados para calcular \(\hat{\pi}_{r|i}\)
            if len(captura) > 0 and len(recaptura) > 0 and len(interseccao) > 0:
                v_i1_r = len(captura['Mercado_Filtro'].unique())  # Contagem de mercados na captura
                v_i2_r = len(recaptura['Mercado_Filtro'].unique())  # Contagem de mercados na recaptura
                v_i12_r = len(interseccao['Mercado_Filtro'].unique())  # Contagem de mercados na interseção

                pi_hat = ((v_i1_r + v_i2_r) * v_i12_r) / (2 * v_i1_r * v_i2_r)
            else:
                pi_hat = 1e-6  # Evitar divisão por zero com valor mínimo

            # Armazenar resultados para gráfico
            if plataforma not in resultados_por_plataforma:
                resultados_por_plataforma[plataforma] = {}
            resultados_por_plataforma[plataforma][produto] = pi_hat

    # Criar gráficos para cada plataforma
    for plataforma, produtos in resultados_por_plataforma.items():
        plt.figure(figsize=(8, 5))
        bars = plt.bar(produtos.keys(), produtos.values(), color='royalblue')

        # Adicionar rótulos de dados
        for bar in bars:
            yval = bar.get_height()
            plt.text(bar.get_x() + bar.get_width()/2, yval, f'{yval:.4f}', ha='center', va='bottom')

        plt.xlabel("Produto")
        plt.ylabel("Probabilidade $\hat{\pi}_{r|i}$")
        plt.title(f"Probabilidades estimadas por Produto - Plataforma {plataforma}")
        plt.xticks(rotation=45)
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.show()

calcular_estimadores(dataFrame_T1, 1)

# ESTIMATIVAS RELACIONADOS A MÉDIA POR PRODUTO E PLATAFORMA

# Função para calcular captura, recaptura, interseção e média de y para cada plataforma e produto
def calcular_resultados_por_T(df, captura_data, recaptura_data):
    resultados = {'r': [], 'i': [], 'v_ir1': [], 'v_ir2': [], 'Média_y': []}

    for plataforma in df['r'].unique():

        for i in df['i'].unique():
            df_plat_prod = df[(df['r'] == plataforma) & (df['i'] == i)]

            # Captura e recaptura (filtrar apenas os dados de captura e recaptura para a plataforma e produto)
            v_ir1 = df_plat_prod[df_plat_prod['Data'] == captura_data]
```

```
        v_ir2 = df_plat_prod[df_plat_prod['Data'] == recaptura_data]

        # Calcular a média de y_ijr(t) para captura e recaptura
        media_y_v_i1 = v_ir1['y_ijr(t)'].mean() if len(v_ir1) > 0 else 0
        media_y_v_i2 = v_ir2['y_ijr(t)'].mean() if len(v_ir2) > 0 else 0

        # Calcular a média final como a soma das médias de captura e recaptura dividido por 2
        media_y = (media_y_v_i1 + media_y_v_i2) / 2

        # Adicionar resultados
        resultados['r'].append(plataforma)
        resultados['i'].append(i)
        resultados['v_ir1'].append(len(v_ir1))
        resultados['v_ir2'].append(len(v_ir2))
        resultados['Média_y'].append(media_y)

    resultados_df = pd.DataFrame(resultados)

    return resultados_df

# Função para calcular o estimador da variância para cada plataforma e produto
def calcular_variancia(df, captura_data, recaptura_data):
    variancias = {'r': [], 'i': [], 'Variancia_Total': []}

    for plataforma in df['r'].unique():
        for i in df['i'].unique():
            # Filtrar os dados para a plataforma e produto específicos
            df_plat_prod = df[(df['r'] == plataforma) & (df['i'] == i)]
            v_ir1 = df_plat_prod[df_plat_prod['Data'] == captura_data]
            v_ir2 = df_plat_prod[df_plat_prod['Data'] == recaptura_data]

            # Interseção entre captura e recaptura
            v_ir12 = pd.merge(v_ir1, v_ir2, on=['Mercado_Filtro', 'i'], suffixes=('_1', '_2'))

            # Tamanho das amostras de captura, recaptura e interseção
            v_i1 = len(v_ir1)
            v_i2 = len(v_ir2)
            v_i12 = len(v_ir12)

            # Estimativa do tamanho médio da amostra
            v_i0 = (v_i1 + v_i2) / 2

            if v_i12 <= 0:
              V_it = 1
            else:
            # Estimativa do tamanho total da população
              V_it = (v_i1 * v_i2) / v_i12

            # Verificar se há registros suficientes
            if len(v_ir1) > 0:

                # Média amostral da captura e recaptura
                Y_barra_i1 = v_ir1['y_ijr(t)'].mean()

                # Variância da captura (S_i1^2)
                soma_quadrados_1 = ((v_ir1['y_ijr(t)'] - Y_barra_i1) ** 2).sum()
                sigma_hat_yi_1 = soma_quadrados_1 / (v_i1 - 1)
                #sigma_hat_yi_1 = 0.5 * (sigma_hat_yi_1 + sigma_hat_yi_1)
                variancia_1 = (0.5 * (1-(v_i0/V_it)) * (sigma_hat_yi_1 /v_i0))
            else:
              variancia_total_1 = 0

            if len(v_ir2) > 0:
                Y_barra_i2 = v_ir2['y_ijr(t)'].mean()
                # Variância da recaptura (S_i2^2)
                soma_quadrados_2 = ((v_ir2['y_ijr(t)'] - Y_barra_i2) ** 2).sum()
                sigma_hat_yi_2 = soma_quadrados_2 / (v_i2 - 1)
```

```
            variancia_2 = (0.5 * (1-(v_i0/V_it)) * (sigma_hat_yi_2 /v_i0))
          else:
            variancia_total_1 = 0

          # Variância total ajustada (soma das variâncias dividida por 4)
          variancia_total = variancia_1 +variancia_2 / 4

        # Caso não haja registros suficientes

          # Adicionar resultados
          variancias['r'].append(plataforma)
          variancias['i'].append(i)
          variancias['Variancia_Total'].append(variancia_total)

    variancia_df = pd.DataFrame(variancias)

    return variancia_df

# Calcular as variâncias para T=1
variancias_T1 = calcular_variancia(dataFrame_T1, '2024-10-01', '2024-10-02')

# Mostrar as variâncias calculadas
print(resultados_T1)
print(variancias_T1)


# Função para plotar a média de y por plataforma e produto
def plotar_media_y(resultados_df, T_value):
    for plataforma in resultados_df['r'].unique():
        resultados_plat = resultados_df[resultados_df['r'] == plataforma]

        # Ordenar pela média de y
        resultados_plat = resultados_plat.sort_values(by='Média_y', ascending=False)

        # Configuração do gráfico
        plt.figure(figsize=(12, 6))
        x = np.arange(len(resultados_plat['i']))  # posições no eixo x

        # Plotando o gráfico de barras
        plt.bar(x, resultados_plat['Média_y'], color='purple', alpha=0.7)

        # Título e rótulos
        plt.title(f'Estimativa da média de preços para {plataforma} por produto (T={T_value})', fontsize=16)
        plt.xlabel('Produto', fontsize=14)
        plt.ylabel('Média de preços estimada', fontsize=14)
        plt.xticks(x, resultados_plat['i'], rotation=45)

        # Adicionar rótulos de dados nas barras
        for i, v in enumerate(resultados_plat['Média_y']):
            plt.text(x[i], v + 0.02, f'{v:.2f}', ha='center', va='bottom', fontsize=12)

        plt.tight_layout()
        plt.show()

# Plotar gráficos das médias de y para T=1
plotar_media_y(resultados_T1, T_value=1)



# Função para plotar o desvio padrão
def plotar_desvio_padrao(variancias_df, T_value):
    for plataforma in variancias_df['r'].unique():
        variancias_plat = variancias_df[variancias_df['r'] == plataforma]

        # Ordenar pela variância
        variancias_plat = variancias_plat.sort_values(by='Variancia', ascending=False)

        # Calcular o desvio padrão a partir da variância
```

```
        desvio_padrao = np.sqrt(variancias_plat['Variancia'])

        plt.figure(figsize=(12, 6))
        x = np.arange(len(desvio_padrao))  # posições no eixo x

        plt.bar(x, desvio_padrao, color='blue', alpha=0.7)
        plt.title(f'Desvio Padrão estimado dos preços de cada produto, na plataforma {plataforma} (T={T_value})', fontsize=16)
        plt.xlabel('Produto', fontsize=14)
        plt.ylabel('Desvio Padrão estimado', fontsize=14)
        plt.xticks(x, variancias_plat['i'], rotation=45)

        # Adicionar rótulos de dados
        for i, v in enumerate(desvio_padrao):
            plt.text(i, v + 0.01, f'{v:.2f}', ha='center', fontsize=12)

        plt.tight_layout()
        plt.show()

# Plotar gráficos do desvio padrão para T=1
plotar_desvio_padrao(variancias_T1, T_value=1)


            # ESTIMATIVAS RELACIONADOS A MÉDIA POR PRODUTO
# Ler o arquivo Excel
dataFrame = dataFrame.rename(columns={'Plataforma': 'r'})

# Remover registros onde Marca == 'Não Condiz'
dataFrame = dataFrame[dataFrame['i'] != "Não Condiz"]

# Garantir que as colunas numéricas estejam no formato correto
dataFrame['y_ijr(t)'] = pd.to_numeric(dataFrame['y_ijr(t)'], errors='coerce')
dataFrame['m_ij'] = pd.to_numeric(dataFrame['m_ij'], errors='coerce')

# Substituir valores NaN por valores padrão
dataFrame['y_ijr(t)'].fillna(0, inplace=True)
dataFrame['m_ij'].fillna(1, inplace=True)

# Dividir o DataFrame por T
df_T1 = dataFrame[dataFrame['T'] == 1]

# Função para calcular os estimadores
def calcular_estimadores(df, T):
    resultados_total = []

    for produto in df['i'].unique():
        df_produto = df[df['i'] == produto]

        # Dicionário para armazenar π̂(r|i) para cada plataforma
        pi_hat_r_i = {}

        for plataforma in df_produto['r'].unique():
            df_plataforma = df_produto[df_produto['r'] == plataforma]

            # Selecionar registros para capturas e recapturas
            captura = df_plataforma[df_plataforma['Data'].isin(['2024-10-01'])]
            recaptura = df_plataforma[df_plataforma['Data'].isin(['2024-10-02'])]

            # Cálculo da interseção
            interseccao = pd.merge(captura, recaptura, on=['Mercado_Filtro', 'i'], suffixes=('_1', '_2'))

            # Agora, utilizamos as contagens de mercados para calcular π̂(r|i)
            if len(captura) > 0 and len(recaptura) > 0 and len(interseccao) > 0:
                # Obtendo as contagens de mercados de cada réplica
                v_i1_r = len(captura['Mercado_Filtro'].unique())  # Contagem de mercados na captura
                v_i2_r = len(recaptura['Mercado_Filtro'].unique())  # Contagem de mercados na recaptura
                v_i12_r = len(interseccao['Mercado_Filtro'].unique())  # Contagem de mercados na interseção
```

```
        # Cálculo de π̂(r|i)
        pi_hat_r_i[plataforma] = ((v_i1_r + v_i2_r) * v_i12_r) / (2 * v_i1_r * v_i2_r)
    else:
        pi_hat_r_i[plataforma] = 1e-6  # Evitar divisão por zero com valor mínimo

# Inicializar variáveis para Ŷ_i e V̂_it para captura e recaptura
Y_hat_i_total_captura = 0
V_hat_it_total_captura = 0
Y_hat_i_total_recaptura = 0
V_hat_it_total_recaptura = 0
variancia_captura = 0
variancia_recaptura = 0

# Calcular Ŷ_i e V̂_it para captura
for index, row in captura.iterrows():
    y_ijr_t = row['y_ijr(t)']
    m_ij = row['m_ij']
    plataforma_atual = row['r']
    pi_hat_r_i_plataforma = pi_hat_r_i.get(plataforma_atual, 0)

    if m_ij > 0 and pi_hat_r_i_plataforma != 0:
        Y_hat_i_captura = y_ijr_t / (m_ij * pi_hat_r_i_plataforma)
        V_hat_it_captura = 1 / (m_ij * pi_hat_r_i_plataforma)
    else:
        Y_hat_i_captura = 0
        V_hat_it_captura = 0

    Y_hat_i_total_captura += Y_hat_i_captura
    V_hat_it_total_captura += V_hat_it_captura

# Calcular Ŷ_i e V̂_it para recaptura
for index, row in recaptura.iterrows():
    y_ijr_t = row['y_ijr(t)']
    m_ij = row['m_ij']
    plataforma_atual = row['r']
    pi_hat_r_i_plataforma = pi_hat_r_i.get(plataforma_atual, 0)

    if m_ij > 0 and pi_hat_r_i_plataforma != 0:
        Y_hat_i_recaptura = y_ijr_t / (m_ij * pi_hat_r_i_plataforma)
        V_hat_it_recaptura = 1 / (m_ij * pi_hat_r_i_plataforma)
    else:
        Y_hat_i_recaptura = 0
        V_hat_it_recaptura = 0

    Y_hat_i_total_recaptura += Y_hat_i_recaptura
    V_hat_it_total_recaptura += V_hat_it_recaptura

# Calcular Y_mean_i_captura e Y_mean_i_recaptura
if V_hat_it_total_captura > 0:
    Y_mean_i_captura = Y_hat_i_total_captura / V_hat_it_total_captura
else:
    Y_mean_i_captura = 0

if V_hat_it_total_recaptura > 0:
    Y_mean_i_recaptura = Y_hat_i_total_recaptura / V_hat_it_total_recaptura
else:
    Y_mean_i_recaptura = 0

# Calcular a variância para captura usando a fórmula fornecida
for index, row in captura.iterrows():
    y_ijr_t = row['y_ijr(t)']
    m_ij = row['m_ij']
    plataforma_atual = row['r']
    pi_hat_r_i_plataforma = pi_hat_r_i.get(plataforma_atual, 0)

    if m_ij > 0 and pi_hat_r_i_plataforma != 0:
```

```
                variancia_captura += ((y_ijr_t / m_ij) - (Y_mean_i_captura / m_ij)) ** 2 * (1 / pi_hat_r_i_plataforma) * (1 / pi_hat_r_i_plataforma - 1)

        # Calcular a variância para recaptura usando a fórmula fornecida
        for index, row in recaptura.iterrows():
            y_ijr_t = row['y_ijr(t)']
            m_ij = row['m_ij']
            plataforma_atual = row['r']
            pi_hat_r_i_plataforma = pi_hat_r_i.get(plataforma_atual, 0)

            if m_ij > 0 and pi_hat_r_i_plataforma != 0:
                variancia_recaptura += ((y_ijr_t / m_ij) - (Y_mean_i_recaptura / m_ij)) ** 2 * (1 / pi_hat_r_i_plataforma) * (1 / pi_hat_r_i_plataforma - 1)

        # Evitar divisão por zero
        variancia_captura /= V_hat_it_total_captura ** 2 if V_hat_it_total_captura > 0 else 1
        variancia_recaptura /= V_hat_it_total_recaptura ** 2 if V_hat_it_total_recaptura > 0 else 1

        # Cálculo da média ponderada final
        total_v = (V_hat_it_total_captura + V_hat_it_total_recaptura) / 2
        if total_v > 0:
            Y_mean_i_final = (Y_mean_i_captura + Y_mean_i_recaptura) / 2
        else:
            Y_mean_i_final = 0

        # Calcular a variância final como a média das variâncias de captura e recaptura
        variancia_final = (variancia_captura + variancia_recaptura) / 2 if total_v > 0 else 0
        desvio_padrao = variancia_final ** 0.5 if variancia_final > 0 else 0
        coeficiente_variacao = (desvio_padrao / Y_mean_i_final) * 100 if Y_mean_i_final > 0 else 0

        # Adicionar resultados do produto
        resultados_total.append({
            'i': produto,
            'T': T,
            'Y_mean_i_t': Y_mean_i_final,
            'variancia_final': variancia_final,
            'variancia_captura': variancia_captura,
            'variancia_recaptura': variancia_recaptura,
            'desvio_padrao': desvio_padrao,
            'coeficiente_de_variação': coeficiente_variacao,
        })

    # Criar DataFrame com os resultados
    df_resultados_total = pd.DataFrame(resultados_total)
    return df_resultados_total

# Calcular estimadores para T=1
df_resultados_total_T1 = calcular_estimadores(df_T1, T=1)

# Salvar os DataFrames em arquivos Excel separados
with pd.ExcelWriter('resultados_estimadores.xlsx') as writer:
    df_resultados_total_T1.to_excel(writer, sheet_name='T1_Total', index=False)

# Exibir os resultados
print(df_resultados_total_T1)

def gerar_graficos_estimativas(df_resultados, T):

    # Gráfico de Y_mean_i_t
    df_ordenado_y_mean_i_t = df_resultados.sort_values(by='Y_mean_i_t', ascending=False)
    plt.figure(figsize=(10, 5))
    plt.bar(df_ordenado_y_mean_i_t['i'], df_ordenado_y_mean_i_t['Y_mean_i_t'], color='green')
    plt.title(f'Média de preços estimada por Produto para T={T}')
    plt.xlabel('Produto')
    plt.ylabel('Preço médio estimado')
    plt.xticks(rotation=90)
```

```
    # Adicionar rótulos de dados para Y_mean_i_t
    for i, v in enumerate(df_ordenado_y_mean_i_t['Y_mean_i_t']):
        plt.text(i, v + 0.01, f'{v:.2f}', ha='center', fontsize=10)

    plt.tight_layout()
    plt.show()

    # Gráfico de variância BE
    df_ordenado_variancia_be = df_resultados.sort_values(by='variancia_final', ascending=False)
    plt.figure(figsize=(10, 5))
    plt.bar(df_ordenado_variancia_be['i'], df_ordenado_variancia_be['variancia_final'], color='red')
    plt.title(f'Variância estimada por Produto para T={T}')
    plt.xlabel('Produto')
    plt.ylabel('Variância estimada')
    plt.xticks(rotation=90)

    # Adicionar rótulos de dados para variância BE
    for i, v in enumerate(df_ordenado_variancia_be['variancia_final']):
        plt.text(i, v + 0.01, f'{v:.2f}', ha='center', fontsize=10)

    plt.tight_layout()
    plt.show()

    # Gráfico de desvio padrão
    df_ordenado_desvio_padrao = df_resultados.sort_values(by='desvio_padrao', ascending=False)
    plt.figure(figsize=(10, 5))
    plt.bar(df_ordenado_desvio_padrao['i'], df_ordenado_desvio_padrao['desvio_padrao'], color='red')
    plt.title(f'Desvio Padrão estimado por Produto para T={T}')
    plt.xlabel('Produto')
    plt.ylabel('Desvio Padrão estimado')
    plt.xticks(rotation=90)

    # Adicionar rótulos de dados para desvio padrão
    for i, v in enumerate(df_ordenado_desvio_padrao['desvio_padrao']):
        plt.text(i, v + 0.01, f'{v:.2f}', ha='center', fontsize=10)

    plt.tight_layout()
    plt.show()

# Gerar gráficos para T=1
gerar_graficos_estimativas(df_resultados_total_T1, T=1)
```