Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Sistema de recomendação híbrido para *e-commerce* imobiliário

Autores: Bruno Matias Casas, Daniel Marques Rangel
Orientador: Prof. Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF
2020

Bruno Matias Casas, Daniel Marques Rangel

Sistema de recomendação híbrido para *e-commerce* imobiliário

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Vandor Roberto Vilardi Rissoli

Brasília, DF

2020

Bruno Matias Casas, Daniel Marques Rangel
Sistema de recomendação híbrido para *e-commerce* imobiliário/ Bruno Matias Casas, Daniel Marques Rangel. – Brasília, DF, 2020-
119 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vandor Roberto Vilardi Rissoli

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2020.

1. Sistema de recomendação. 2. *machine learning*. I. Prof. Dr. Vandor Roberto Vilardi Rissoli. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Sistema de recomendação híbrido para *e-commerce* imobiliário

CDU 02:141:005.6

Bruno Matias Casas, Daniel Marques Rangel

Sistema de recomendação híbrido para *e-commerce* imobiliário

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 11 de dezembro de 2019.

Prof. Dr. Vandor Roberto Vilardi Rissoli
Orientador

Prof. Dr. André Barros de Sales
Convidado 1

Artur Bersan de Faria
Convidado 2

Prof. Dr. Renato Coral Sampaio
Convidado 3

Brasília, DF
2020

Resumo

Diante da grande quantidade e variedade de informações na Internet atualmente, criou-se a necessidade da existência de sistemas que ajudem os usuários a encontrarem o que eles desejam de acordo com sua preferência, para assim diminuir o tempo gasto em pesquisas nesse ambiente. Em sítios virtuais de comércio eletrônico, por exemplo, há centenas ou milhares de produtos à mostra para os usuários, e isso dificulta o trabalho de encontrar o produto adequado às particularidades de cada cliente. O presente trabalho trata-se do desenvolvimento de um sistema de recomendação de imóveis, no contexto de negócios do tipo B2B (*Business to business*), por meio de um sítio virtual de *e-commerce* de venda de imóveis para imobiliárias apresentarem suas propriedades e obterem visibilidade de seus possíveis clientes, utilizando de um sistema de recomendação que procuraria atendê-los de maneira mais eficiente e diminuindo o esforço de procura ao que atenderia seus interesses particularidades para aquisição de um novo imóvel. O objetivo principal seria auxiliar os clientes que visitassem este sítio virtual em localizar o imóvel desejado de forma mais eficiente, desenvolvendo um sistema de recomendação mais robusto, comparado aos sistemas já existentes. As recomendações de imóveis terão como base as características desses clientes adquiridas pelo sistema de recomendação. Esse sistema seguirá uma abordagem híbrida, ou seja, contemplando duas ou mais técnicas de pesquisa, gerando melhores resultados aos clientes que utiliza-lo. Assim, foi realizado um estudo inicial por meio de pesquisas bibliográficas sobre conceitos, abordagens e técnicas utilizadas em sistemas de recomendação no contexto imobiliário para geração de uma nova proposta de sistema de recomendação imobiliário que foi desenvolvido por este trabalho.

Palavras-chave: Aprendizado de máquina. Sistema de recomendação. Comércio eletrônico. Imobiliária.

Abstract

Facing the large quantity and variety of information on the Internet nowadays, was created the necessity of the existence of systems that help the users to find what they look for according to their preferences, this way reducing the time wasted in researching within this environment. Looking around e-commerce websites, for example, there are hundreds or thousands of products on display for the users, making it hard to find a product that corresponds to the clients' desires. The following work discusses the development of a real estate recommendation system, in the context of a B2B type of business, by means of an e-commerce website for real estate sales for housing companies displaying their properties and receive visibility from their possible clients, using a recommendation system that would serve them more efficiently and decreasing the research efforts for something that could suit their interests for the acquisition of a new property. The main goal is to help clients that visit said website to find the desired property more easily, developing a more powerful recommendation system compared to existing systems. The recommendations are based on the characteristics of these clients acquired by the recommendation system. It will follow a hybrid approach, that is, contemplating two or more research techniques, generating better results for customers using this system. An initial study was conducted through bibliographical research on concepts, approaches and techniques used in real estate recommendation systems to generate a new system proposal that was developed by this work.

Key-words: Machine Learning. Recommender System. E-commerce. Real Estate.

Lista de ilustrações

Lista de tabelas

Lista de quadros

Lista de abreviaturas e siglas

API	Application Programming Interface
AWS	Amazon Web Services
B2B	Business to Business
B2C	Business to Consumer
B2G	Business to Government
BSD	Berkeley Software Distribution
C2G	Consumer to Government
CV	Conversion
CSS3	Cascading Style Sheets 3
CSV	Comma-Separated-Values
CPU	Central Processing Unit
CVR	Conversion Rate
DER	Diagrama de Entidade-Relacionamento
DRY	Don't Repeat Yourself
ECS	Amazon Container Service
ES5	ECMA Script 5
FE	Feature
FGA	Faculdade do Gama
HDF5	Hierarchical Data Format 5
HTML5	HyperText Markup Language 5
ID	Identifier
JSON	JavaScript Object Notation
MAUT	Multi-Attribute Utility Theory

MIT	Massachusetts Institute of Technology
MVC	Model-View-Controller
MV	Model-View
MVT	Model-View-Template
NAR	The American National Association of Realtors
NMF	Non-negative matrix factorization
S3	Amazon Simple Storage Service
RAM	Random Access Memory
REST	Representational State Transfer
SQL	Structured Query Language
SR	Sistema de Recomendação
OO	Orientação a Objeto
ORM	Object-Relational Mapping
TCC1	Trabalho de Conclusão de Curso 1
TCC2	Trabalho de Conclusão de Curso 2
TS	Technical Story
US	User Story
VCPU	Video Central Processing Unit
VM	Virtual Machine
XML	Extensible Markup Language

Lista de símbolos

$\sum$	Somatório
$\mathbb{R}$	Números Reais

Sumário

1 Introdução

Na contemporaneidade são estudados diversos fatores que influenciam no crescimento e sucesso de empresas baseadas em soluções tecnológicas. Grandes empresas de prestação de serviço e venda online que tem como foco a interação de itens com seus clientes consideram de suma importância ao seu sucesso um sistema de recomendação (SR).

Esse tipo de sistema (SR) tenta predizer a preferência que um dado usuário teria por um item através de ferramentas e técnicas de software (SOUZA, 2014). A maioria das empresas, por exemplo, utilizam-se de técnicas como *collaborative filtering* (filtro colaborativo), que procura predizer a utilidade de itens de comércio para um cliente em particular, baseando-se na avaliação de um cliente por uma determinada amostra ou população de outros clientes, ou seja, a recomendação gerada se baseia na similaridade entre os clientes (LINDEN; SMITH; YORK, 2003).

Segundo Ricci et al. (2010), há vários motivos para fornecedores investirem nessa tecnologia, tais como:

- Aumentar o número de itens vendidos;
- Vender itens mais diversos;
- Aumentar a satisfação do cliente;
- Aumentar a fidelidade do usuário;
- Melhor entendimento do que o usuário deseja.

Existem diversas multinacionais que utilizam o mecanismo de recomendação em seus serviços, principalmente como forma de melhorar a interação do usuário e aumentar seu potencial financeiro. Em 2017 notou-se um crescimento acelerado dos últimos quatro a cinco anos, e houve indícios que continuasse assim para os próximos anos (UNDERWOOD, 2017).

Negócios populares e de grande porte como Netflix, Amazon, Ebay, Walmart, Costco, Spotify, Youtube e outros, são conhecidos pelos seus sistemas de recomendação. Tais empresas fazem grandes investimentos nesse tipo de tecnologia, com a finalidade de obter um retorno financeiro ainda maior (UNDERWOOD, 2017).

A empresa Netflix é uma provedora internacional de séries e filmes online e se destaca entre as empresas de mídias mais bem-sucedidas financeiramente da atualidade.

Graças à sua enorme quantidade de dados, foi possível criar um mecanismo de recomendação muito eficiente, no qual 75 a 80 por cento de todo o conteúdo assistido é proveniente desse mecanismo, contra os 20 a 25 por cento assistidos que são oriundos de pesquisa (KLEINMAN, 2014). O sistema de recomendação é tão relevante para essa empresa que ela criou um campeonato, o Netflix *Prize* (Prêmio), em que o desafio era melhorar a qualidade da recomendação, sendo que a equipe vencedora "*BellKor's Pragmatic Chaos*" melhorou em pouco mais de 10% e recebeu um prêmio no valor de um milhão de dólares (NETFLIXPRIZE, 2009).

Com o crescente uso e prevalência da Internet, os *web services* se tornaram uma tendência. Empresas que necessitam vender seus produtos optam por utilizar plataformas virtuais, ou como mais popularmente são conhecidos, os *e-commerces*.

O mercado de varejo se deu muito bem com o comércio virtual. A Amazon, por exemplo, domina o comércio eletrônico varejista, sendo responsável por quase 50 por cento de todas as vendas online nos Estados Unidos em 2018, gerando 207 bilhões de dólares em operações de varejo pelo mundo (FRANKENFIELD, 2019).

O comércio de negócios para consumidores provou um aumento excelente nos últimos 15 anos, tornando o *e-commerce* algo comum na vida das pessoas (JIANG et al., 2015). Segundo a Forrester Research, vendas de varejo online da América Latina atingirão 129 bilhões de dólares até 2023. A pesquisa ainda afirma que o Brasil é o maior mercado de comércio eletrônico da região (VARON, 2019).

Essa tendência também atingiu o mercado imobiliário. Segundo o *The American National Association of Realtors* (NAR), pesquisa sobre tecnologias online, conduzida em 2011, indicou que quando se trata de busca por um lar, 88% de compradores de imóveis escolhem a Internet como fonte de informações (YUAN et al., 2013).

O rápido crescimento de empresas de vendas online atrai atenção de investidores e gera um ambiente propício para a competição. Para garantir o sucesso de um *e-commerce* é de extrema importância a busca de novos recursos para garantir novos compradores e a retenção de antigos (JIANG et al., 2015).

Intitulado como maior *e-commerce* de imóveis do Japão, o Summo.jp presa sempre em melhorias na experiência do usuário. Dentre todos os elementos de seu sistema que podem melhorar a experiência de seus usuários, o Summo.jp considera seu sistema de recomendação como o mais importante e por isso tentam sempre melhorá-lo. Seus usuários normalmente pesquisam e navegam em diversas páginas antes de encontrar um imóvel que os satisfaçam, podendo resultar em um processo cansativo e acabando em desistências. Recentemente, o Summo.jp passou por uma evolução e construiu um novo sistema de recomendação baseado nos dados gerados nas pesquisas e navegações do usuário, melhorando consideravelmente sua métrica de qualidade (LI et al., 2017).

Buscando adotar uma abordagem semelhante, este trabalho procura oferecer serviços diversos para imobiliárias, no qual um ambiente de *e-commerce* personalizado possibilitará que os clientes dessas imobiliárias possam ter acesso e visualizar informações relativas aos diversos imóveis disponíveis para venda, agendar visitas, requisitar mais informações e verificar propriedades sobre aqueles imóveis que cada cliente tenha mais interesse.

Para a elaboração de um projeto também sintonizado com a realidade do mercado imobiliário do Distrito Federal a Liva, *startup* que oferece serviços neste ramo imobiliário, cooperou com o seu desenvolvimento. Os criadores da Liva também são donos de reconhecidas imobiliárias e estão atuando nesse ramo a algumas décadas. Essa situação corresponde a um dos fatores motivacionais para a elaboração deste projeto.

Um dos principais serviços fornecidos pela Liva.vc corresponde ao *e-commerce* de vendas de imóveis para clientes de diversas imobiliárias, sendo mostrado na Figura 1 uma das janelas de acesso ao seu ambiente virtual.

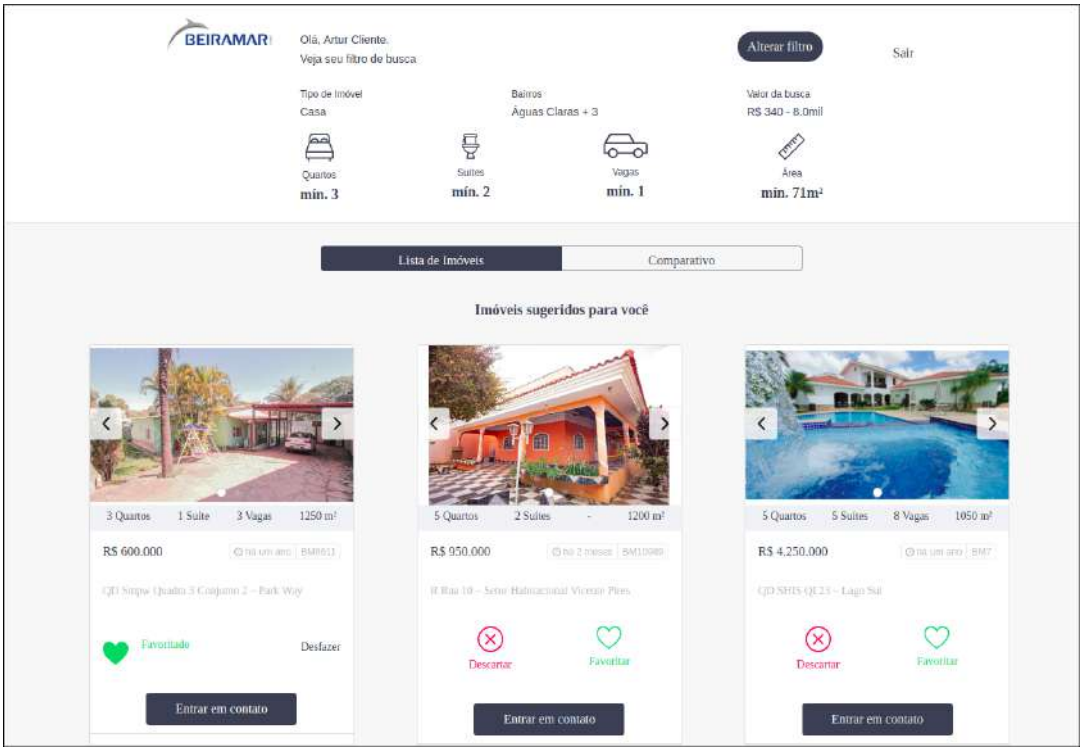

Figura 1 – Página principal da plataforma Liva.vc. Fonte: (LIVA, 2019).

Esse tipo de serviço oferece um sistema de recomendação de propriedades baseado nas preferências de cada cliente. Essas preferências são primeiramente geradas a partir de um *lead*. Em termos gerais, essa expressão *leads* corresponderia aos clientes em potencial, ou seja, pessoas que demonstraram interesse por algum tipo de produto ou serviço, assim tornando-o um cliente interessado em algo oferecido por um negócio (AGENCIAKAIZEN, 2019).

Com os *leads* gerados das plataformas diariamente em grandes quantidades, diversos usuários são postos a se tornarem clientes em potencial no Liva.vc. Com dados advindos de interações de usuários com a plataforma é possível extrair diversas informações úteis que possibilitam ainda mais refinar a preferência de cada usuário.

O processo de Mineração de Dados (*Data Mining*), que é formado por diversas ferramentas e técnicas, possibilita a exploração de dados e sua extração, que ajudam a evidenciar padrões e dessa forma auxiliam na descoberta de conhecimento (HAN; KAMBER; PEI, 2011). Tal conhecimento pode ser usado para encontrar ótimos representantes da preferência dos usuários.

Os Sistemas de Recomendação correspondem a uma subárea do Aprendizado de Máquina (*Machine Learning*), um campo da Ciência da Computação que consiste na investigação de como computadores podem aprender a partir de dados já armazenados. Sua principal área de pesquisa é em aprendizagem automática de computadores, reconhecendo padrões complexos e tomando decisões inteligentes baseadas nos dados. De maneira mais ampla, os métodos e técnicas empregadas nesse tipo de processamento são provenientes da Inteligência Artificial, estando neste caso em particular compreendendo o Aprendizado de Máquina e a Mineração de Dados (HAN; KAMBER; PEI, 2011).

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo principal elaborar um sistema de recomendação para a plataforma disponível na web Liva.vc, a fim de oferecer um serviço que atenda as expectativas de qualidade de seus clientes.

1.1.2 Objetivos Específicos

- Coletar dados oriundos de ações dos usuários na plataforma, a fim de gerar um conjunto de dados de treino inicial para o modelo de aprendizado de máquina do sistema de recomendação proposto;
- Desenvolver um recomendador de propriedades para clientes da plataforma se utilizando de técnicas de aprendizado de máquina, contemplando abordagens de recomendações conhecidas;
- Desenvolver um recomendador de propriedades se utilizando de uma abordagem de recomendação baseada em críticas, em que os usuários serão capazes de refinar suas preferências a partir de suas críticas feitas sobre características das propriedades na plataforma;

- Implantar o sistema de recomendação proposto em ambiente de produção da plataforma, a fim de analisar sua eficácia através de uma métrica adequada.

1.2 Questões de Pesquisa

Com o intuito de obter os conhecimentos necessários para realização da implementação desse projeto, será pesquisada a resposta para a seguinte questão:

- Como pode ser feito um sistema de recomendação para um site (sítio virtual) *e-commerce* focado em venda de imóveis, utilizando-se de dados, implícitos ou explícitos, referentes à preferência dos clientes, das propriedades e suas características, com abordagens e tecnologias conhecidas?

Como já explicitado, existem diversas empresas de prestação de serviços e vendas online que utilizam de sistemas de recomendação, implementados com base em tecnologias modernas, que em sua maioria são muito importantes por proporcionar recomendações relevantes de seus produtos aos clientes, de forma a ser muito benéfico para o negócio.

Cada sistema de recomendação se adéqua ao seu contexto, como por exemplo, a Netflix tem um sistema de recomendação diferente da Amazon. O tipo de produto oferecido e a regra de negócio por trás de cada empresa gera diversos fatores determinantes em um sistema de recomendação. Um exemplo de fator em filmes que se diferem de outros produtos seria em relação ao tempo em que uma avaliação foi feita pelos usuários. A nota que uma pessoa atribui para um filme tende a ser melhor caso ela tenha acabado de assistir-lo e mais moderado caso tenha assistido a um tempo atrás. Esse fator foi considerado pela equipe campeã do Netflix *Prize* (GARCIA, 2015).

Dessa forma, ao comparar técnicas, sistemas e tecnologias é possível identificar e analisar diversas informações, que possibilitam encontrar a melhor maneira de implementar um mecanismo de sugestão robusto ao contexto da Liva.

1.3 Justificativa

A motivação deste trabalho parte do princípio que o atual sistema de recomendação de imóveis implementado na Liva não é embasado em nenhum tipo de abordagem para tal, apenas se baseia em buscas e filtros básicos de imóveis para o usuário, provocando assim, uma menor chance de encontrar o imóvel desejado e consequentemente perdas de possíveis clientes.

Na maioria das vezes o usuário gasta muito tempo procurando e lendo várias páginas antes de achar um possível imóvel ideal para sua realidade. Dado que esse é um

processo bastante trabalhoso, aumenta a possibilidade de desistência por parte do usuário. Ainda assim, existe também a possibilidade de um usuário que entra em contato com o corretor, não saber se a residência de enfoque é realmente a melhor diante de sua realidade (LI et al., 2017).

Como a plataforma online implementada na Liva contém milhares de clientes ativos e imóveis cadastrados, oriundos das treze imobiliárias que utilizam o serviço, é possível implementar um mecanismo de recomendação robusto que pode ajudar os usuários na parte de escolha de domicílio, diminuindo o tempo de busca, e também o estresse causado pelo esforço de procura necessário.

Com isso, é possível implementar um software de recomendação usando abordagens bastante conhecidas nessa área, assim, aumentando a chance de rápida localização de residências que correspondam com o perfil dos usuários do *website*, ou seja, apresentam-se recomendações mais relevantes de acordo com os dados obtidos de cada cliente interessado.

1.4 Estrutura

Este trabalho de conclusão de curso (TCC) está estruturado em cinco capítulos, sendo assim descritos:

- **Capítulo 1 - Introdução:** capítulo corrente, corrente apresenta o contexto em que o trabalho está inserido, seus principais objetivos, questão de pesquisa e a justificativa para a solução proposta;
- **Capítulo 2 - Referencial teórico:** relata os conceitos pesquisados que deverão ser entendidos para ser possível implementar a proposta deste trabalho em um sistema robusto de recomendação de imóveis para a plataforma Liva.vc;
- **Capítulo 3 - Proposta:** explica em detalhes a proposta deste trabalho para atender os objetivos descritos, contemplando os métodos e as tecnologias utilizadas, além de sua arquitetura e metodologias que foram seguidas junto ao cronograma do trabalho e a simulação de viabilidade realizada;
- **Capítulo 4 - Desenvolvimento:** retrata a execução planejada no capitulo anterior (3), sendo detalhado todo o processo de desenvolvimento, junto com as dificuldades e mudanças que ocorreram.
- **Capítulo 5 - Considerações finais:** apresenta a conclusão deste trabalho a partir do planejamento, desenvolvimento e análise de resultados, juntamente com possíveis trabalhos futuros que poderão ser originados a partir de novas abstrações e requisitos, além do desenvolvimento evolutivo sobre o que foi elaborado por este TCC.

2 Referencial Teórico

2.1 Mercado imobiliário

Todo ser humano por natureza precisa de uma habitação, sendo isso uma necessidade ligada a procura de uma maior segurança contra as várias adversidades do ambiente externo. No entanto, essa necessidade de consumo por uma habitação atualmente se divide em dois principais grupos: aquelas pessoas que realmente investem em uma moradia com a finalidade de satisfazer essa necessidade básica de maior proteção, e também, aqueles indivíduos que compram com a finalidade de aumentar seu capital financeiro ou compor a base de sua economia (ARRAES; FILHO, 2008).

Um dos setores imobiliários que administra esses consumos é o setor terciário, que envolve as atividades do mercado imobiliário (compra, venda e locação) e será o contexto principal deste trabalho. Esse setor é de suma importância para o mercado nacional, pois é gerador de emprego, renda e altos níveis de investimentos. Uma das particularidades relevantes desse setor é a sua importante participação nas características de desenvolvimento de uma nação. Com essa importância o governo tende a incentivar essa área, com o intuito de incentivar os negócios imobiliários para que estimule o país a se desenvolver (CORREA et al., 2018).

O setor imobiliário é muito importante para a sociedade, entretanto, como todo setor econômico, é necessário que negócios sempre estejam à frente em novidades do mercado, para que possam dar certo e expandir. Um processo muito utilizado por imobiliárias nos dias atuais é o *marketing* digital.

2.1.1 *Marketing* Digital

Um dos meios mais poderosos utilizados pelas imobiliárias para atrair visitantes, que possam se tornar futuros clientes, é o *marketing* em meio digital, que disponibiliza serviços e conteúdos de qualidade aos seus possíveis clientes. Por meio dele é possível oferecer oportunidades para uma imobiliária se relacionar com seus clientes (BARRETO, 2019).

Como qualquer outro processo, o *marketing* digital também precisa de estudo e aperfeiçoamento constante, dado que em cada etapa há perda de clientes. Dessa forma, a representação ilustrativa desse tipo de ocorrência é mais conhecida como funil de *marketing* digital (BORGES, 2017).

É fortemente recomendado que toda e qualquer empresa planeje seu processo,

desde a atração do visitante, feita por meio da propaganda, até a aquisição do cliente que antes era simplesmente um visitante interessado em atender seus próprios objetivos. Além de planejar, também deve ocorrer o acompanhamento de toda a trajetória do visitante até a conversão deste em cliente (INSIDEOUT, 2018). Segundo o *site* Agenciainsideout.com (2018), o funil de *marketing* ajuda no andamento de vendas e aquisição de clientes, gerando certos benefícios, tais como:

- Melhoria na gestão;
- Previsão de resultados;
- Aumento de produtividade;
- Mais oportunidades de negócios com *leads* qualificados;
- Evolução na estratégia de aquisição de *leads*;
- Melhoramento do produto.

Como descrito pelo site Agenciainsideout.com (2018), pode ser observado na Figura 2 que, o funil tem quatro fases principais, sendo que podem ser feitas mudanças de acordo com o contexto de cada empresa. Para maior aprofundamento, há a descrição de cada uma dessas fases a seguir:

- 1ª fase - Aquisição (visitante): o visitante entra no ambiente virtual da empresa por algum motivo, seja por meio de anúncios em outra página virtual disponível na Internet ou por alguma pesquisa que fez. Um fator considerado importante nesta etapa é atrair o visitante para continuar a usar o sítio virtual da empresa para atender ao seu objetivo, mesmo ainda não sendo totalmente conhecido, retornando a ele e o compartilhando com outras pessoas. Algumas características importantes que podem ajudar nisso são disponibilizar conteúdos sempre recentes e também oferecer uma usabilidade interessante;
- 2ª fase - Ativação (*lead*): o visitante fornece algumas informações de contato, como por exemplo: nome completo, e-mails, telefones, entre outros, em troca de algum conteúdo, geralmente o que seria o chamado *lead*. Para obter mais *leads* é aconselhado verificar se há muitas perguntas ou se há alguma pergunta de teor evasivo;
- 3ª fase - Retenção (oportunidade): agora que já foram adquiridas algumas informações de contato é necessário começar a se relacionar com o visitante. Um fator recomendado é ajudar o visitante para que ele fique mais interessado, como por exemplo, dar uma amostra grátis de algum serviço ou parte de tal serviço. E também, analisar, se possível, o histórico daquele usuário para compreender melhor quais poderiam ser os interesses dele;

- 4ª fase - Receita (cliente): com o relacionamento estabelecido, o visitante se torna finalmente um cliente, após esse acontecimento é muito importante medir o nível de satisfação do cliente com o produto ou serviço, ou seja, uma fidelização da parte do cliente com a empresa e se está acontecendo a recomendação de sua empresa por este cliente.

Figura 2 – Funil de *marketing* digital. Fonte: (INSIDEOUT, 2018).

Diante do que foi dito, é fácil perceber que o comercio eletrônico é um importante aliado do *marketing* digital de empresas do setor imobiliário, entre outras, principalmente na 1ª fase de aquisição do visitante. Por ser de grande relevância, é importante conhecer o comercio eletrônico um pouco mais para o desenvolvimento consciente deste trabalho.

2.1.2 *E-commerce*

O comércio eletrônico ou *e-commerce*, como também é chamado, surgiu e ficou famoso na década de 90, com a ajuda da popularização da Internet e de empresas pioneiras neste setor como Ebay e Amazon.com. Esse tipo de comércio oferecia a opção de compra de qualquer produto das empresas através de seus serviços fornecidos pela rede mundial de computadores, podendo realizar a busca de um produto específico e mostrar outros que fossem relacionados àquele que estava sendo visualizado. Com os avanços de conexão e da segurança na Internet tornou-se possível que os seus usuários pagassem pelo próprio produto virtualmente e os recebesse em sua moradia, envolvendo também os serviços de postagem ou entrega comercial (NAKAMURA, 2011).

Smith, Speaker e Thompson (2000) consideram o conceito de comércio eletrônico como a realização de relações comerciais apenas através do meio digital, ou seja, através de sistemas (*e-commerce's*) que se comunicam em meio digital.

O *e-commerce* é dividido em quatro categorias, sendo elas: *Business to business* (B2B - Empresas para Empresas), *Business to consumer* (B2C - Empresas para Consumidores), *Business to government* (B2G - Empresas para o governo), *Consumer to govern-*

ment (C2G - Consumidores para o governo). Para fins deste trabalho, será aprofundado o B2B (NAKAMURA, 2011).

O B2B é um negócio entre empresas, ou seja, de uma empresa para com outra empresa, na qual são realizadas atividades de compra e venda de vários tipos de produto, disponibilização de informações e serviços por meio de algum *website* ou compartilhamento de rede privada entre companhias, assim, substituindo o comércio físico tradicional (NAKAMURA, 2011).

O B2B tem três principais grupos de *websites*, sendo eles:

- ***Website* para colaborador:** utilizado exclusivamente para comunicação interna de uma organização;
- ***Website* para parceiro:** usado com o intuito de comunicação entre a empresa e os parceiros de negócio para compartilhamento de informações;
- ***Website* para terceiro:** tem a finalidade de servir como intermediador entre compradores e vendedores para impulsionar o processo de negociação online.

Com a expansão cada vez maior da Internet junto com o crescimento do *e-commerce*, tanto no tamanho quanto na complexidade das informações contidas neles, torna-se interessante o uso de alguma ferramenta de auxílio para seus usuários encontrarem o que desejam com maior facilidade e agilidade. Entre os diversos recursos disponíveis atualmente uma tecnologia se destaca aos objetivos pretendidos por este trabalho que são os Sistemas de Recomendação (SR).

2.2 Sistemas de Recomendação

Os Sistemas de recomendação são aplicativos "inteligentes" que auxiliam usuários em suas tarefas de busca de informações, sugerindo itens que melhor atendam às suas necessidades e preferências (MAHMOOD; RICCI, 2009). Esse item descrito seria o termo geral usado para indicar o que o sistema recomenda aos usuários, podendo ser música, filme, série, dentre outros (RICCI et al., 2010).

Segundo Cimini (2019), sistemas de recomendação são tipos de algoritmos que usam do conhecimento de *machine learning* (aprendizado de máquina) com o objetivo de prever os gostos, desejos e necessidades de seus usuários. Seu esquema de funcionamento pode ser visto em alto nível na Figura 3.

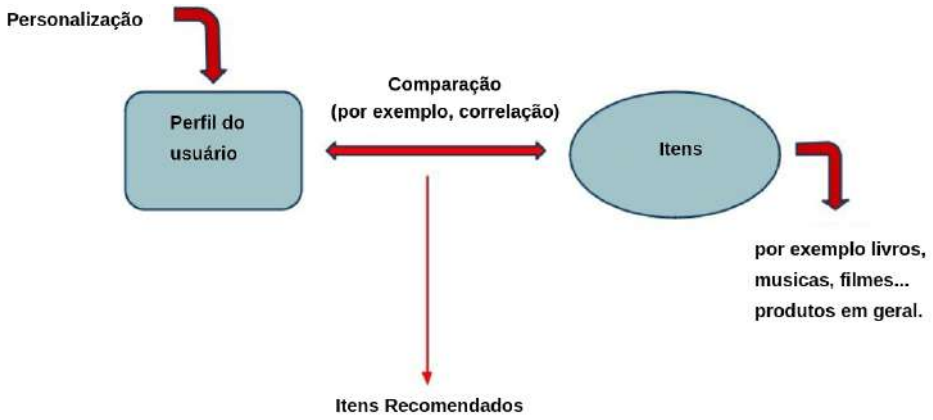

Figura 3 – Arquitetura do sistema de recomendação em alto nível. Fonte: (ZISOPOULOS et al., 2008). Tradução livre pelos autores.

Atualmente é possível comprar qualquer tipo de produto em lojas online e isso gerou um evento conhecido como "Efeito Cauda Longa", como demonstrado na Figura 4, na qual os produtos populares são poucos e podem ser encontrados com mais facilidade em lojas físicas ou em *e-commerce's*. Entretanto produtos não tão populares estão em maior quantidade e na grande maioria das vezes apenas são encontrados em sítios virtuas de vendas (PANDEY, 2019).

Não é porque o produto não é altamente popular que ele não vai ter uma boa qualidade, mas como esse tipo de produto é abundante, acaba que se torna um trabalho árduo pesquisar por algum, necessitando então de uma espécie de filtro. Nesse contexto que os sistemas de recomendação pode ser empregados com sucesso.

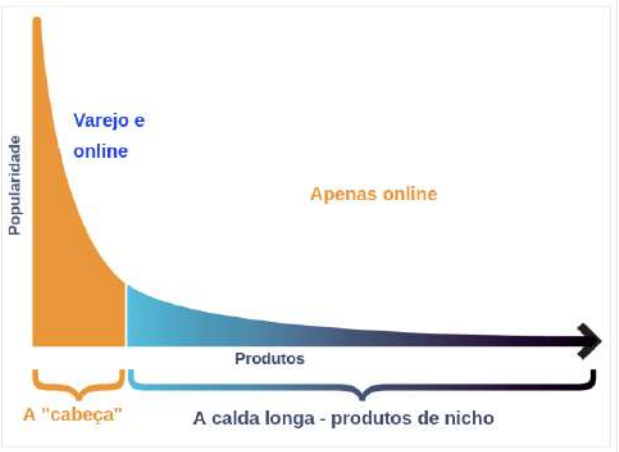

Figura 4 – Efeito Cauda Longa. Fonte: (PANDEY, 2019). Tradução livre pelos autores.

2.2.1 História do sistema de recomendação

Segundo Sharma e Singh (2016), antes mesmo da existência de computadores os sistemas de recomendações já prevaleciam na sociedade. Civilizações antigas entre o

período de 4000 a 1200 a.C utilizavam do conceito de recomendações para fins de decisão no cultivo e até mesmo em âmbito religioso.

Durante o período de colonização, as recomendações ajudavam civilizações a escolherem territórios a se conquistar, analisando-se quesitos de fertilidade para agricultura ou de mão de obra. Reis, na era dos reinos, recebiam recomendações de seus ministros de quase todos os assuntos. Eles tinham o papel de manter seus pontos de vista à frente do rei, a fim de ajudá-lo a tomar as decisões mais facilmente. Famílias antigas arranjavam casamentos, pessoas decidiam na compra de produtos ou no caminho mais curto a se fazer de um local até outro. Em todos os casos se utilizavam recomendações para se realizar o que se desejava de maneira mais acertada ou coerente com os objetivos almejados (SHARMA; SINGH, 2016).

Com a revolução industrial, o surgimento da era computacional foi possível, mudando completamente o mercado no mundo. Uma infinidade de produtos são postos à frente das pessoas e dessa forma gerando confusões no momento de decisão da compra, como por exemplo, saber se determinado produto realmente atende os requisitos de preferência de uma pessoa. Com a necessidade da existência de sistemas capazes de filtrar ou facilitar nos critérios de seleção, há, finalmente, o surgimento dos sistemas de recomendação automáticos (SHARMA; SINGH, 2016).

Segundo Ekstrand, Riedl e Konstan (2011), foi reconhecido logo cedo na história da computação a capacidade de máquinas fornecerem recomendações. Grundy, um bibliotecário baseado em computador, é considerado de grande importância para o engajamento dos sistemas de recomendação. Ele era bastante primitivo, usava de entrevistas para obter informações com usuários a fim de agrupá-los em estereótipos e assim gerar recomendações baseadas nas preferências de livros por estereótipo.

Segundo Sharma e Singh (2016), os sistemas de recomendação se tornaram uma área de pesquisa em meados da década de 1970 na Universidade de Duke. Com o avanço nos estudos começaram a surgir abordagens como *collaborative filtering* (filtragem colaborativa) no início dos anos 90 como soluções para sistemas online. O primeiro sistema de recomendação foi o *Tempestry1*. Era um sistema manual com a abordagem de filtragem colaborativa, que possibilitava usuários procurarem por itens baseados em ações de outros usuários. Sistemas de recomendação com essa abordagem começaram a se tornar uma tendência e um tópico de interesse nas áreas de pesquisa como: interação humano computador, aprendizado de máquina e mineração de dados (EKSTRAND; RIEDL; KONSTAN, 2011).

Ao final dos anos 90, os sistemas de recomendação começaram a emergir e serem implementados comercialmente. Provavelmente, o maior exemplar dessa tecnologia foi o *e-commerce* da Amazon.com que considerava o histórico de compras e ações do usuário para fazer recomendações de seus itens na plataforma desenvolvida. Logo em seguida,

diversos outros negócios começaram a implantar esse tipo de tecnologia em seus sistemas online (EKSTRAND; RIEDL; KONSTAN, 2011).

Com a quantidade e variedade crescente de informações no ambiente virtual da Internet e produtos para comercialização, os sistemas de recomendações se tornaram um meio valioso para lidar com esse tipo de problema de sobrecarga de informações, promovendo escolhas eficientes e rápidas para os seus usuários (RICCI et al., 2010).

Segundo Ricci et al. (2010), o interesse por sistema de recomendação aumentou consideravelmente nos últimos anos e esse fato é visto por causa de indicadores, como: *sites* bem avaliados (Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, e IMDb) utilizarem de sistemas de recomendações e desempenharem um papel importante pro negócio, a existência de conferências e *workshops* na área, o acolhimento desse tópico por instituições de ensino e a existência de pesquisas e desenvolvimento na área em revistas acadêmicas. Além disso, também ocorreu um grande marco, que seria a Netflix ter promovido de 2006 a 2009 o Netflix *Prize*, uma competição com o fim de premiar em 1 milhão de dólares a equipe que desenvolvesse o melhor algoritmo que melhorasse as recomendações de filmes para os usuários de sua plataforma que oferece muitos filmes e séries (NETFLIXPRIZE, 2009).

2.2.2 Funções de um sistema de recomendação

Segundo Ricci et al. (2010), existem dois tipos de funções que um sistema de recomendação pode desempenhar, um referente ao provedor do serviço e o outro ao seu usuário (cliente). Um exemplo seria um sistema de recomendação para um *e-commerce* imobiliário, em que a motivação primária do cliente ao entrar na plataforma é encontrar um imóvel adequado as suas necessidades, enquanto que para o provedor do serviço seria aumentar o número de imóveis comercializados.

Esses autores ainda descrevem algumas razões para provedores de serviço investirem em sistemas de recomendação, sendo estas razões relacionadas a seguir:

Aumentar o número de itens vendidos: considerada a função mais importante para um sistema de recomendação comercial por gerar vendas adicionais comparadas às vendas advindas de um processo sem suporte de recomendações. Itens que são recomendados, em sua maioria, atendem as necessidades dos usuários, dessa forma cumprindo o objetivo do sistema de recomendação.

Os sistemas de recomendação não-comerciais apresentam também muitas vezes objetivos semelhantes, mesmo que não haja custo para o usuário, como por exemplo, sítios virtuais de notícias que desejam aumentar o número de notícias lidas por usuários.

De forma geral, o principal objetivo de provedores de serviço em aplicar sistema de recomendação em seu negócio é aumentar a quantidade de usuários que aceitem as reco-

mendações, fazendo comparação com a quantidade de usuários que navegam no sistema.

Vender itens mais diversos: também considerada uma função muito importante para um sistema de recomendação, que possibilita para o usuário recomendações de itens que podem ser difíceis de serem encontrados "manualmente" (sem apoio de sistemas de recomendação). O provedor do serviço quer que todos os seus itens sejam vendidos, não somente os mais populares. Dessa forma, o sistema de recomendação apresenta itens que se adéquam as preferências de cada usuário, e assim apresentam itens, podendo ser populares ou não, para o usuário certo.

Aumentar a satisfação do cliente: aumentar a satisfação do usuário em um sítio virtual ou aplicativo faz com que eles gostem do sistema, aumentando suas visitas e também a chance de aceitação de recomendações. Boas interfaces de interação com o usuário e recomendações interessantes e precisas são fatores relevantes na satisfação do cliente.

Aumentar a fidelidade do usuário: tratar clientes antigos como valiosos é muito importante, pois em sistemas de recomendação, muitas vezes, é utilizada as interações dos usuários antigos como base para o seu modelo de recomendação pela classificação dos itens disponíveis. Assim, quanto mais interações os usuários fizerem, dependendo do modelo utilizado, seu modelo será refinado e poderá atender melhor as preferências dos clientes.

Melhor entendimento do que o usuário quer: com as preferências dos usuários coletadas implícita ou explicitamente pelo sistema de recomendação, há a possibilidade de reutilização do conhecimento gerado por análises de dados, para outros objetivos, como melhorar o gerenciamento do estoque ou da produção do item. Um exemplo seria em plataformas de viagens recomendarem anúncios promocionais para determinado tipo de cliente baseado em sua preferência de lugares visitados.

Os sistemas de recomendação também desempenham várias funções referentes ao usuário final. De acordo com Herlocker et al. (2004), são elas:

Anotação no contexto: o sistema de recomendação vai enfatizar itens baseado em um contexto. Por exemplo, na interface de navegação de TV a cabo, há a listagem dos programas que vão passar e o sistema de recomendação vai apresentar o que vale a pena ou não assistir (RICCI et al., 2010).

Encontrar bons itens: a principal tarefa de sistemas de recomendação é apresentar para o usuário uma lista específica e avaliada de itens, que são recomendações baseadas em previsões de quanto o usuário gostaria de tal item. Na maioria dos sistemas comerciais a probabilidade de recomendação gerada pelo modelo não é apresentada.

Encontrar todos os itens bons: a maioria dos sistemas de recomendação focam em recomendar somente alguns itens, mas existem casos que cabe apresentar todos os

itens para um usuário. Um exemplo seria sistemas para recomendar casos para advogados que procuram precedentes, em que não se pode negligenciar um único caso. Para esse tipo de recomendação o número de falsos negativos devem ser suficientemente baixos para ter uma excelente precisão no momento de recomendar os itens.

Recomendar uma sequência: existem casos que são recomendados itens para o usuário mas em uma devida sequência que seja agradável ao mesmo. Como utilizado no sistema de recomendação do Yahoo *Entertainment* (Entretenimento) na aba de musicas (launch.yahoo.com), que recomenda músicas para os usuários em uma devida sequência.

Recomendar um agrupamento: sistemas de recomendação também podem sugerir um grupo de itens que combinados seriam melhor aceitos. Por exemplo, em ambientes de viagens normalmente são recomendados com os destinos seus pontos turísticos, restaurantes, hotéis entre outras recomendações combinadas (RICCI et al., 2010).

Apenas navegar: a principal função de um recomendador é ajudar ao seu usuário (cliente) a tomar uma decisão baseada em suas preferências, mas nem sempre é assim, existem muitos usuários que utilizam o sistema sem um objetivo eminente e navegam simplesmente porque acham agradável. Nesses casos a interface, facilidade de uso e a natureza das informações importam muito mais do que o acerto do recomendador.

Encontrar recomendações confiáveis: é muito comum usuários não confiarem em recomendadores e é por isso que muitas vezes eles testam o recomendador, alterando seu perfil, por exemplo, para averiguar se o comportamento do recomendador está condizente com suas preferências. Por esse motivo, muitos sistemas de recomendação oferecem esse tipo de funcionalidade que permitem usuários conseguirem testar o comportamento do recomendador.

Melhorar o perfil: os usuários tendem a contribuir com avaliações nos sistemas com o intuito de melhorar seu perfil, e dessa forma aprimorar a qualidade da recomendação recebida. É a função que a maioria dos sistemas de recomendação utiliza, sendo primordial para a atribuição de recomendações personalizadas para clientes que são usuários desse tipo de sistema.

Expressar-se: alguns usuários não se importam com recomendações feitas para eles e sim em contribuir com suas avaliações, simplesmente pelo fato de se sentirem bem fazendo essa contribuição. Esse tipo de usuário normalmente avalia anonimamente diversos itens e tem certa facilidade em fazer contribuições. Pode-se dizer que é muito interessante para este tipo sistema fornecer mais dados e assim melhorar as recomendações para os outros usuários.

Ajudar outros: muitas vezes, os usuários tem o objetivo de contribuir com avaliações de itens para que melhore nas recomendações de outros usuários. Essa função, em sua maioria, está vinculada a função de **Expressar-se**, abordada no item anterior, mas

nem sempre elas estão implementadas juntas.

Influenciar outros: considerado como um fator infeliz, alguns usuários têm a motivação de fazer avaliações de itens para influenciar declaradamente outros usuários na compra ou visualização dos itens. Um exemplo seria na situação em que usuários avaliam filmes antes mesmo deles serem lançados para fazer com que eles sejam recomendados, e, dessa forma, influenciar as pessoas a assistirem.

2.2.3 Abordagens de recomendações

Como dito por Beliakov et al. (2011), as justificativas usadas por sistemas de recomendação para sugerirem itens para usuários que vão depender de fatores específicos das aplicações e da forma como os dados são coletados pelos sistemas. Dessa maneira, várias abordagens vêm sendo utilizadas para desenvolver sistemas de recomendação.

Nesta seção serão descritas as principais abordagens de recomendações, denominadas *collaborative filtering* (filtragem colaborativa), *content-based filtering* (filtragem baseada em conteúdo), *knowledge-based recommender systems* (sistemas de recomendação baseados em conhecimento), *hybrid recommender systems* (sistemas de recomendação híbridos) e *critiquing-based recommender systems* (sistemas de recomendação baseados em críticas).

2.2.3.1 Filtragem Colaborativa

A ideia principal dessa técnica consiste em que as pessoas que concordaram no passado tendem a concordar no futuro, ou seja, é possível analisar o histórico de cada indivíduo para predizer qual item ele gostará também, precisando apenas do histórico de preferência de um conjunto de usuários do sistema (NILASHI et al., 2013).

Essa obtenção de dados de preferência se divide em duas categorias: classificação explícita e a classificação implícita. A primeira é uma taxa de 0 a 5 atribuída pelo próprio usuário em algum item. Esse procedimento é o *feedback* mais direto para se obter o dado de preferência, mas que nem sempre é advindo de uma grande quantidade de dados. O segundo infere os gostos do sujeito indiretamente através de ações no *site*, como: quantidade de visualizações, cliques, registro de compras passadas, ter escutado ou não a alguma música entre outros (LUO, 2018).

Segundo Pazzani e Billsus (2007), conforme citado por Nilashi et al. (2013), a filtragem colaborativa tem as seguintes características:

- Criação de perfil do usuário através das classificações de usuários nos itens;
- Identificação de usuários com gostos parecidos que classificam itens de maneira semelhante para um usuário alvo usando alguma função para apurar as possíveis

similaridades;

- Recomendar um ou vários itens para o usuário alvo que os usuários com gostos parecidos preferiram anteriormente.

Essa filtragem para realizar a recomendação necessita relacionar dois itens importantes para os sistemas de recomendação, sendo eles os itens e os usuários. Para realizar a comparação entre esses dois itens ou entidades relevantes para este tipo de sistema tem-se duas principais técnicas: abordagem de vizinhança e o modelo de fatores latentes. Na relação item a item, a modelagem do perfil do usuário para um item é feita com base nas classificações do usuário em outros itens semelhantes (KOREN; BELL, 2015).

Os métodos para detecção dessa semelhança se dividem em dois tipos: o *memory-based* (baseado em memória) e o *model-based* (baseado em modelo). A seguir, são exploradas algumas das principais características destes dois tipos.

- **Baseado em memória**: esse tipo tem como característica calcular as semelhanças entre vizinhos na memória do computador. Os dados de entrada são carregados um passo antes do cálculo de detecção e assim são utilizados para fornecer a recomendação (LEVINAS, 2014). Essa categoria se divide em dois métodos: *User-based filtering* (filtragem baseada em usuários) e *Item-based filtering* (filtragem baseado em itens).
  - **Filtragem baseada em usuários**: usuários com gostos parecidos irão realizar classificações semelhantes nos mesmos itens, por isso, é certo dizer que as classificações obtidas de outros usuários com a mesma opinião do usuário alvo, pode ser usada para recomendar ao usuário alvo (CIMINI, 2019).
  - **Filtragem baseada em itens**: itens parecidos foram classificados de forma bem semelhante pelo mesmo usuário. Para predizer as classificações para um item alvo B pelo usuário A, deve-se primeiro determinar um conjunto de itens que sejam semelhantes ao item alvo B, assim, as classificações desse conjunto serão usadas para prever se o usuário A irá gostar do item B (CIMINI, 2019).

Em outras palavras, ao invés de utilizar a semelhança de comportamentos das classificações dos usuários para recomendar, será usada a semelhança entre os padrões de classificação dos itens. Se um grupo de itens tendem a ter os mesmos usuários que gostam e que não gostam, então seria certo dizer que eles são semelhantes (EKSTRAND; RIEDL; KONSTAN, 2011).

O item-item ou *item-based* (baseada nos itens) foi criado com a ideia de corrigir a característica de má escalabilidade do *user-based* (baseada nos usuários), mas na prática acaba que não corrige totalmente. As duas abordagens necessitam achar a similaridade,

seja de usuário ou seja de item, mas como na maioria dos sistemas há mais usuários do que itens, assim é mais econômico encontrar a similaridade pelo item do que pelo usuário (EKSTRAND; RIEDL; KONSTAN, 2011). Uma representação dessa abordagem pode ser observada na Figura 5.

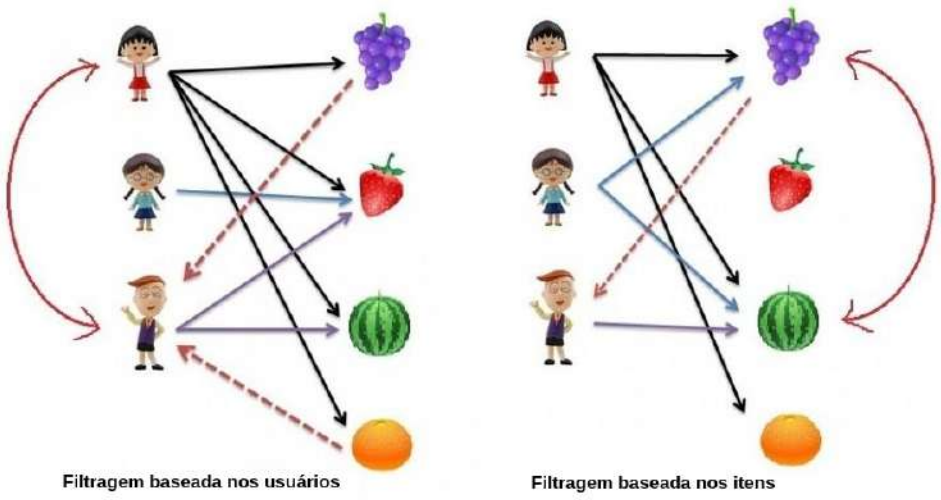

Figura 5 – Filtragem baseada nos usuários e filtragem baseada nos itens. Fonte: (PINELA, 2017). Tradução livre pelos autores.

- **Baseado em modelo**: diferente do tipo baseado em memória, a *model-based* extrai informações de um conjunto de dados para criar um "modelo" que é utilizado para calcular as recomendações. Geralmente, é produzido um modelo *offline* ao mesmo tempo que a recomendação acontece online. Foi graças ao Netflix *Prize* que essa categoria recebeu um investimento expressivo nas comunidades de pesquisa (LEVINAS, 2014).

A escalabilidade aparece como o principal problema da abordagem, pois é necessário carregar uma grande quantidade de dados em memória no servidor para poder assim realizar a recomendação. Isso só piora quanto maior for a quantidade de usuários e de itens. Dessa forma, ele demanda de uma grande quantidade de recurso computacional e acaba comprometendo o desempenho do sistema. (GROVER, 2017).

Para finalizar, como descrito por Pinela (2017), os pontos positivos da filtragem colaborativa são:

- Não depende do contexto do sistema;
- Implementação simples;
- É mais preciso comparado a outras técnicas.

Mas também há alguns pontos negativos a se levar em consideração, tais como:

- A porcentagem de usuários que avaliam itens em um sistema é relativamente baixa;
- Partida fria (*cold start*), ou seja, não irá ter muita informação de novos usuários para ser comparado com os outros usuários, e, dessa forma, surgindo recomendações ruins de inicio;
- Novo item: não haverá muitos dados sobre os novos itens, dificultando a realização de uma classificação robusta.

2.2.3.2 Filtragem baseada em conteúdo

Uma outra técnica muito usada em sistemas de recomendação é a filtragem baseada em conteúdo, na qual é analisado um conjunto de itens que são similares ao item já classificado pelo usuário anteriormente, e assim, são feitas as recomendações para esse usuário. Essa similaridade se dá pela semelhança de características de um item classificado pelo próprio usuário com outro ainda não conhecido pelo usuário (GRIMALDI, 2018). A Figura (6) a seguir retrata essa abordagem.

Segundo Abbattista et al. (2002), para analisar os desejos e gostos do usuário é necessário gerar um perfil deste, que é conhecido como *user profile* (perfil do usuário). O perfil do usuário é um conjunto de várias informações necessárias obtidas do usuário para recomendar um item para ele, extraídas a partir do momento em que ele entra no sistema.

Esse perfil é dado como uma lista de pares de atributos e valores, na qual cada atributo tem o seu respectivo valor, contendo determinada informação desse respectivo usuário. Como exemplo de atributos desse perfil, tem-se: idade, localidade, renda, gênero musical, gênero de filme, emprego, preferências e vários outros.

Esses atributos são divididos em três categorias:

- **Explícito:** valores obtidos do próprio usuário;
- **Implícito:** dados obtidos por meio das ações realizadas pelo usuário;
- **Existente:** informações coletadas a partir de outro serviço já existente ou disponível para uso.

Com esse comportamento de encontrar a similaridade com as características do item, essa abordagem evita a falha de recomendar itens novos no sistema, descrita anteriormente na abordagem de filtragem colaborativa (LUK, 2019).

Figura 6 – Esquema do funcionamento da filtragem baseada em conteúdo. Fonte: (JAIN, 2019). Tradução livre pelos autores.

2.2.3.3 Sistemas de recomendação baseados em conhecimento

Segundo Ricci et al. (2010), um sistema de recomendação baseado em um conhecimento específico de um domínio referente a como características dos itens satisfazem as necessidades e preferências dos usuários, ou, de forma geral, como um item é útil para um usuário, é considerado um sistema de recomendação baseado em conhecimento. O autor descreve que para esses sistemas, uma função de similaridade estima quanto um usuário corresponde a um item, ou seja, a similaridade corresponde a utilidade do item recomendado para um usuário.

Ainda segundo o autor a abordagem baseada em conhecimento tende a funcionar melhor que as outras no início de sua implementação, mas caso não tenham recursos de aprendizagem em sua implementação podem ser superados por outros métodos superficiais como a filtragem colaborativa.

2.2.3.4 Sistemas de recomendação híbridos

Segundo Burke (2007), sistemas de recomendação híbridos são combinações de duas ou mais abordagens descritas anteriormente. Estas recomendações são feitas para melhorar a performance do recomendador e, normalmente, para sanar o problema de partida fria, como descrito no seção 2.2.3.1 referente a filtragem colaborativa.

Existem diversas pesquisas na área que demonstram o sucesso do uso da abordagem híbrida para sistemas de recomendação. Essa abordagem tem a intenção de gerar um

sistema de recomendação baseado no ponto forte de outros singulares. Ao usar mais de uma abordagem é possível uma compensar a falha de outra, sendo então identificado como uma abordagem híbrida o sistema de recomendação a ser desenvolvido neste trabalho.

Burke (2007) descreve sete técnicas de hibridação utilizadas, sendo elas:

- ***Weighted* (Pesado)**: a pontuação de diferentes componentes de recomendação são combinados numericamente;
- ***Switching* (Comutação)**: o sistema escolhe entre componentes de recomendação, seleciona um e o aplica;
- ***Mixed* (Misturado)**: recomendações de diferentes recomendadores são apresentadas juntas;
- ***Feature Combination* (Combinação de características)**: características derivadas de diferentes fontes de conhecimento são combinadas dadas a um único sistema de recomendação;
- ***Feature Augmentation* (Aumento de características)**: uma técnica de recomendação é usada para computar uma ou um conjunto de características, que será parte da entrada de outra técnica;
- ***Cascade* (Cascata)**: os recomendadores recebem prioridade estrita, com os de menor prioridade quebrando os laços na pontuação dos mais altos;
- ***Meta-level* (Meta-nível)**: uma técnica de recomendação é aplicada e gera alguma ordenação de um modelo que é a entrada de outro.

2.2.3.5 Sistema de recomendação baseado em crítica

Segundo Konstan e Riedl (2012), conforme citado por Chen e Pu (2012), a tecnologia de recomendação, explorando similaridade nos usuários, pode ajudar usuários a encontrar itens ideais. Como descrito anteriormente, por exemplo, a filtragem colaborativa tem a premissa de recomendar itens com alta classificação de usuários que avaliam os produtos para outros usuários semelhantes. Entretanto Chen e Pu (2012) descrevem que para domínios de itens de alto risco, como por exemplo uma propriedade ou um veículo, é provável que usuários façam pesquisas e realizem a compra de produtos pela primeira vez. Isso faz com que produtos sejam pouco avaliados, pois pessoas que não compram esse tipo de produto regularmente, não demonstraram sua opinião, incapacitando os sistemas que não vão conseguir estabelecer um perfil adequado para muitos de seus candidatos às recomendações.

Chen e Pu (2012) também descrevem que para superar esse problema chamado de partida fria, descrito anteriormente, surgem os sistemas de recomendação baseados

em crítica, que são amplamente reconhecidos por serem uma tecnologia de recomendação efetiva, assim como uma busca baseada em preferência efetiva que emprega um sistema de *feedback* chamado *critiquing* (criticar).

Esses autores afirmam que nos sistemas baseados em crítica, inicialmente, as preferências do usuário não interferem na precisão da decisão, pois esse sistema se refere a um processo subsequente de críticas incrementais que ajudam os usuários a tomarem decisões mais informadas e precisas.

O sistema de recomendação baseado em crítica simula um vendedor artificial que inicialmente faz recomendações com base nas preferências atuais de um usuário e em seguida obtém *feedback* do mesmo em forma de críticas, como por exemplo: "Gostaria de algo mais barato?". É necessário que esse processo descrito seja em ciclos para que o usuário alcance seu produto ideal (CHEN; PU, 2012). Segundo Pu (2000) e Faltings (2002), citado por Chen e Pu (2012), um comprador normalmente tem muitas restrições e preferências que não são declaradas inicialmente, e ele só toma conhecimento quando as soluções propostas as violam.

Tipicamente, um sistema de recomendação baseado em crítica segue o modelo de interação sistema-usuário como apresentado na Figura 7.

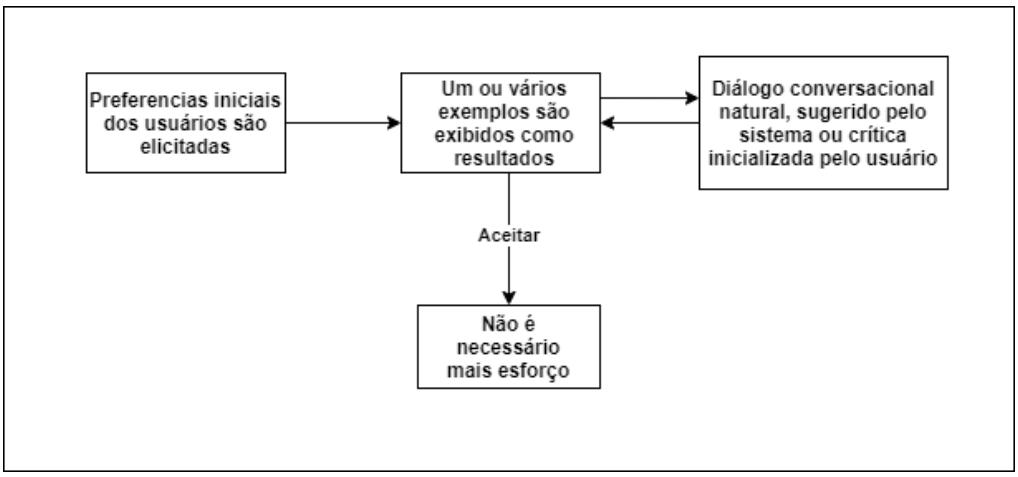

Figura 7 – Modelo de interação sistema-usuário. Fonte: (CHEN; PU, 2012). Tradução livre pelos autores.

Primeiramente, o usuário atribui sua preferência, podendo ser um item de partida ou valores específicos sobre recursos dos itens. Em seguida, o sistema recomenda um ou diversos itens para o usuário a partir de sua preferência inicial. Nesse momento um usuário seleciona um item podendo ser sua escolha final, mas normalmente ele escolhe um item quase desejável e dessa forma fará críticas sobre o item. O tipo de crítica que o usuário irá fazer vai depender do suporte à crítica do sistema. Uma vez que é feita a crítica, o sistema irá atualizar suas recomendações, ou seja, refinará a preferência desse usuário.

Os sistemas de recomendação baseados em crítica dão suporte a três tipos principais, São eles: *natural conversational critiques* (dialogo conversacional natural), *system-suggested critiques* (críticas sugeridas pelo sistema), e *user-initiated critiquing* (crítica iniciada pelo usuário) (CHEN; PU, 2012).

Crítica iniciada pelo usuário. Segundo Pu and Chen (2005), como citado por Chen e Pu (2012), a abordagem de crítica iniciada pelo usuário motiva usuários a fazerem críticas estimulando-os com exemplos. Essa abordagem fornece a liberdade para o usuário fazer a crítica para unidades ou qualquer combinação de recursos. O objetivo desse tipo de sistema é dar suporte ao usuário a executar livremente o *trade-off* (troca). Isso pode ser definido como quando um usuário altera sua preferência referente ao valor de um atributo em particular. Esse processo melhora significativamente a precisão e confiança dos usuários.

Chen e Pu (2012) descrevem *example-critiquing* (crítica de exemplo) como um tipo de sistema que consiste na interface do usuário e uma ferramenta de busca. Inicialmente, um usuário inicia sua busca especificando sua preferência por meio de uma área de consulta. Cada preferência é composta por valores de atributos e um peso atribuído a esse atributo, geralmente, chamado de "importância". Para a geração da recomendação é utilizado um modelo de preferência baseado em MAUT.

Segundo Keeney and Raiffa (1976), conforme citado por Chen e Pu (2012), MAUT ou teoria da utilidade de atributos múltiplos, é uma teoria que leva em consideração preferências de valor conflitantes e produz uma pontuação para cada item. Chen and Pu (2007), conforme citado por Kasamani (2017), definem um modelo de preferência como o par $(V_1, ..., V_n, w_1, ..., w_n)$ onde V_1 é o valor da função e w_i o peso para cada atributo A_i. A utilidade de cada item $(a_1, a_2, ..., a_n)$ pode ser computada utilizando a função de utilidade como mostrada na equação (2.1).

$$U(< a_1, a_2, ..., a_n >) = \sum_{i=1}^{n} w_i V_i(a_i) \tag{2.1}$$

Após atribuída a primeira preferência do usuário o mecanismo de busca construído classificará todos os itens com suas devidas pontuações, o que permitirá sua ordenação e assim o sistema retornará "k" principais itens. Segundo Faltings et al. (2004), como citado por Chen e Pu (2012), normalmente são retornados entre 5 a 20 itens. Então o usuário aceita um item ou escolhe uma solução próxima e interage com o painel de troca (*trade-off*), em que pode atribuir críticas. Em seguida, com um conjunto de críticas o sistema irá refinar o modelo de preferência do usuário juntamente com as importâncias (pesos) dos atributos. Por fim, neste processo o seu valor será recalculado e um novo conjunto de recomendações obtidas.

2.3 Aprendizado de Máquina

Segundo Mitchell (1997), desde a invenção dos computadores sempre existiu a duvida da possibilidade de computadores aprenderem com a experiência automaticamente e se seria possível entender e programá-los para esse fim. Ainda não se sabe como programar computadores para aprenderem tão bem quanto humanos, mas, nos dias atuais, existem algoritmos que são eficazes para certos tipos de tarefas de "aprendizagem".

Segundo Alpaydin (2010), algoritmos são uma sequência de instruções em que se recebe uma entrada e se encontra uma saída. Por exemplo, um algoritmo pode ser feito para ordenar números. A entrada do algoritmo é uma lista de números em qualquer ordem ou desordem e a saída será uma lista desses números ordenados.

Para algumas tarefas, entretanto, algoritmos não são capazes de estabelecer uma saída coerente à esperada. Por exemplo, para identificar *spams* (lixo eletrônico) em um conjunto de e-mails. Como entrada é passado um e-mail e como saída é desejado classificar entre "sim" ou "não" para os possíveis lixos eletrônicos. Não é possível identificar o que pode ser considerado um lixo eletrônico para todos os casos, pois isso é relativo de tempo em tempo e de individuo para individuo (ALPAYDIN, 2010).

O que é "faltante" em conhecimento é compensado em dados. É possível, ainda referente ao exemplo dos e-mails, processar uma quantidade significativa de mensagens de exemplo, indicando ser um e-mail lixo eletrônico ou não, com o objetivo de "aprender" e conseguir classificar mensagens a partir disso. O que é desejável é encontrar um algoritmo para essa tarefa. Para algoritmos de ordenação, por exemplo, não é necessário "aprender" para ordenar, mas existem para várias situações, não um algoritmo, mas dados de exemplos (ALPAYDIN, 2010).

Segundo Witten et al. (2016), o mundo atual está sobrecarregado de dados, e cada vez mais parece continuar aumentando, rapidamente. Os computadores proporcionam armazenamento de dados para salvar coisas que antes provavelmente teria sido jogado fora. Com avanço das tecnologias e seu barateamento, existe um aumento no acesso de pessoas aos computadores. A cada passo no mundo atual é mais um dado no banco de dados. A Internet promove diversas informações e a cada escolha que se faz, mais um registro é efetuado.

Escondido nesses dados existem informações potencialmente úteis (WITTEN et al., 2016). Banco de dados para os fins como exemplificados anteriormente de armazenamento se tornam úteis somente quando analisados e transformados em informações que sejam possível fazer uso, como por exemplo para fazer predições. Nos exemplo dos e-mails, não é completamente randômico a forma em que as mensagens são escritas, há um padrão nessas mensagens, e analisando esse fator é possível fazer predições e classificar os e-mails (ALPAYDIN, 2010).

Segundo Witten et al. (2016), o aprendizado de máquina se trata de um campo do conhecimento em que são desenvolvidas técnicas para encontrar e descrever padrões estruturais em dados. Segundo estes autores, não há nada de novo em tentar encontrar padrões em dados. As pessoas vêm procurando padrões em dados desde o início da humanidade. Os caçadores para encontrar sua caça, buscavam padrões na migração dos animais, agricultores buscavam padrões no crescimento das colheitas, políticos buscavam padrões na opinião dos eleitores entre outras situações.

Segundo Han, Kamber e Pei (2011), o aprendizado de máquina investiga como computadores podem "aprender" ou melhorar sua performance analisando dados. Ele diz ainda que a principal área de pesquisa refere-se ao desenvolvimento de programas de computador que aprendem automaticamente a reconhecer padrões complexos e a tomar decisões inteligentes com base nos dados.

Segundo Alpaydin (2010), a aplicação de métodos de aprendizado de máquina em grandes bases de dados é denominado de *Data Mining* (Mineração de Dados). Han, Kamber e Pei (2011) afirmam que Mineração de dados é um assunto interdisciplinar e pode ser definido de várias maneiras diferentes. Esses autores consideram "mineração de conhecimento a partir de dados" como um nome mais apropriado, pois faz referência a mineração de conhecimento referente as grandes quantidades de dados.

Han, Kamber e Pei (2011) dizem ainda que muitas pessoas consideram a mineração de dados como uma fase essencial do processo de descoberta de conhecimento. O processo é demonstrado na Figura 8 que é uma sequência interativa de sete etapas existentes. Estas etapas são descritas por estes autores como:

- ***Data cleaning*** **(Limpeza de Dados)**: para remover dados inconsistentes;
- ***Data integration*** **(Integração de Dados)**: em que várias fontes de dados são combinadas;
- ***Data selection*** **(Seleção de Dados)**: onde os dados relevantes para a tarefa em análise são recuperados da base de dados;
- ***Data transformation*** **(Transformação de Dados)**: em que os dados são transformados e consolidados em formas apropriadas para mineração, executando operações de resumo ou agregação;
- ***Data mining*** **(Mineração de Dados)**: processo essencial em que métodos inteligentes são aplicados para extrair padrões;
- ***Pattern evaluation*** **(Avaliação de padrões)**: identifica os padrões realmente interessantes que representam o conhecimento com base em medidas de interesse;

- ***Knowledge presentation*** **(Apresentação de conhecimento)**: onde as técnicas de visualização e representação de conhecimento são usadas para apresentar conhecimento extraído das bases para os usuários.

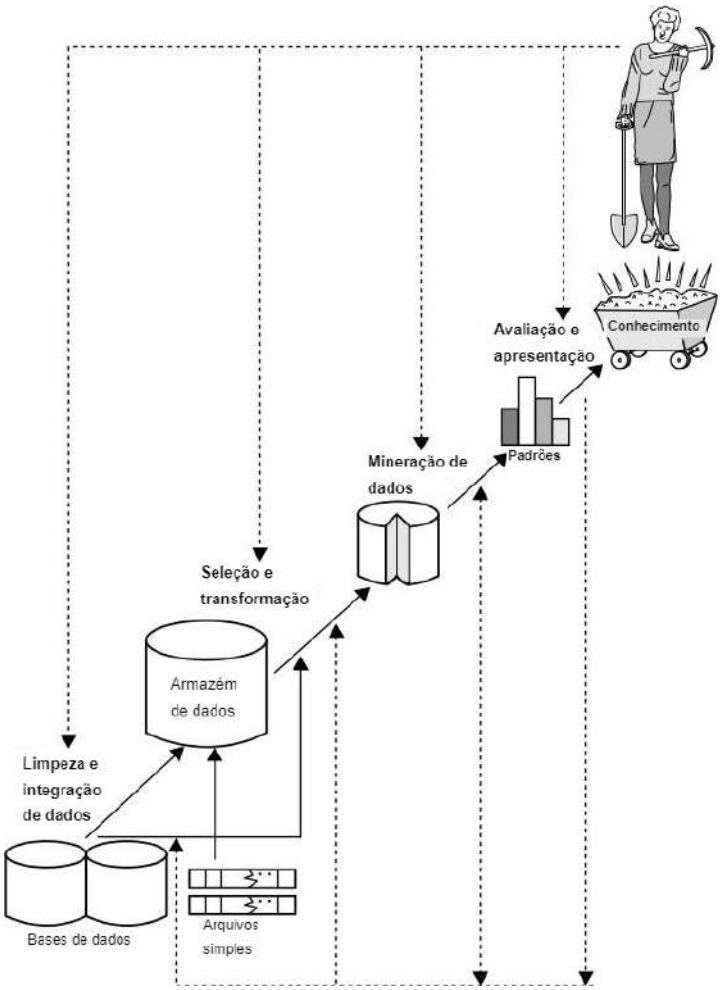

Figura 8 – Etapas do processo de descoberta de conhecimento. Fonte: (HAN; KAMBER; PEI, 2011). Tradução livre pelos autores.

Das etapas de limpeza a transformação de dados os autores consideram como formas diferentes de pré-processamento de dados, que assim são propagados para mineração. Após os dados serem minerados eles são apresentados para o usuário ou postos em uma base de conhecimento. O dado apresentado pode ser ou não reaproveitado.

Dado que muitas outras áreas têm diferentes definições para Mineração de Dados, Han, Kamber e Pei (2011, p. 8) definem-a como:

> "Processo de descobrir padrões e conhecimentos interessantes a partir de grandes quantidades de dados. As fontes de dados podem incluir bancos de dados, *data warehouses*, a *Web*, outros repositórios de informações ou dados transmitidos dinamicamente para o sistema."

2.3.1 Aprendizado indutivo

Como dito anteriormente, a área de aprendizado de máquina permitiu com que aplicações sejam capazes de aprender e melhorar seu desempenho por meio da observação (análise dos dados armazenados). Segundo Batista (2003), existem diversas abordagens diferentes que podem ser utilizadas por uma aplicação como, por exemplo, o aprendizado por hábito, por instrução, por dedução, por analogia e por indução. O autor afirma que o aprendizado indutivo é um dos mais úteis, pois possibilita novos conhecimentos a partir de exemplos ou casos previamente observados, mas também um dos mais desafiadores, porque o conhecimento gerado ultrapassa os limites das premissas e ainda não há a garantia do conhecimento ser verdadeiro.

Monard e Baranauskas (2003) definem a indução como uma forma de inferência lógica que possibilita a obtenção de conclusões genéricas sobre um conjunto de exemplos. Ela é caracterizada por um raciocínio originário de uma generalização de um conceito específico, dessa forma, da parte para o todo. Para um conceito ser aprendido é feita uma inferência indutiva sobre os exemplos apresentados. Mesmo que as hipóteses geradas pela inferência possam ou não preservar a verdade, é um dos principais métodos utilizados para derivar conhecimento novo e predizer eventos futuros.

Os autores definem que o aprendizado indutivo se divide em dois, supervisionado e não-supervisionado. No aprendizado supervisionado os dados apresentados ao algoritmo são exemplos postos a serem treinados e possuem o rótulo da classe associada conhecido. De maneira geral, os exemplos são compostos de características e um rótulo da classe associada. O algoritmo de aprendizado tem como objetivo construir um classificador capaz de determinar a classe de exemplos ainda não rotuladas. Para rótulos de classe com valores contínuos são definidos problemas denominados de “regressão”, para valores discretos “classificação”.

Han, Kamber e Pei (2011) definem aprendizado não-supervisionado essencialmente como sinônimo de *cluster* (grupo). Ele não é supervisionado por seus exemplos de entrada não serem rotulados como classe. Normalmente é possível usar agrupamento para descobrir classes nos dados (características comuns).

Para o presente trabalho será estudado e modelado uma solução em aprendizado de máquina para um problema de classificação supervisionada. A Figura 9 apresenta a hierarquia do aprendizado indutivo destacando a classificação.

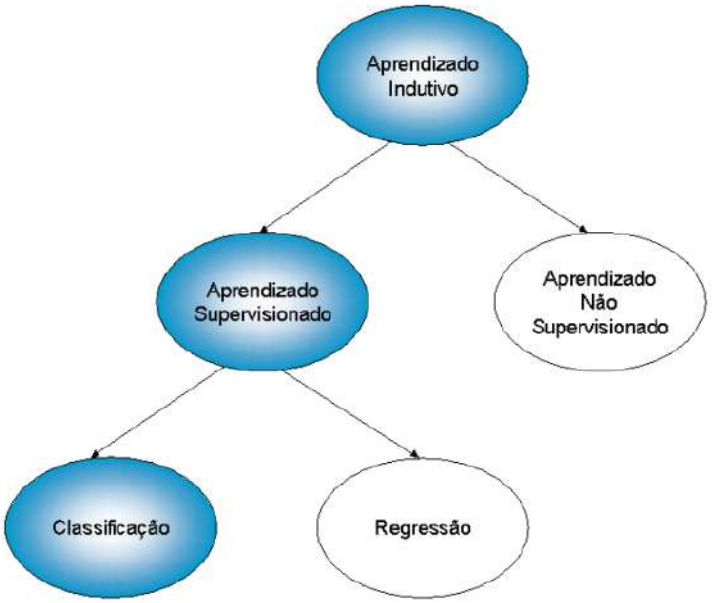

Figura 9 – Hierarquia do aprendizado indutivo. Fonte: (MONARD; BARANAUSKAS, 2003).

Para exemplificar, Monard e Baranauskas (2003) apresentam o processo de classificação como na Figura 10. Estes autores descrevem que os dados de entrada do indutor são provenientes do conhecimento sobre um domínio. Após feita a indução o classificador é avaliado, podendo ou não ser repetido, dado que podem ser adicionados novos atributos ou exemplos, e ainda podem ser feitos ajustes nos parâmetros no processo de indução.

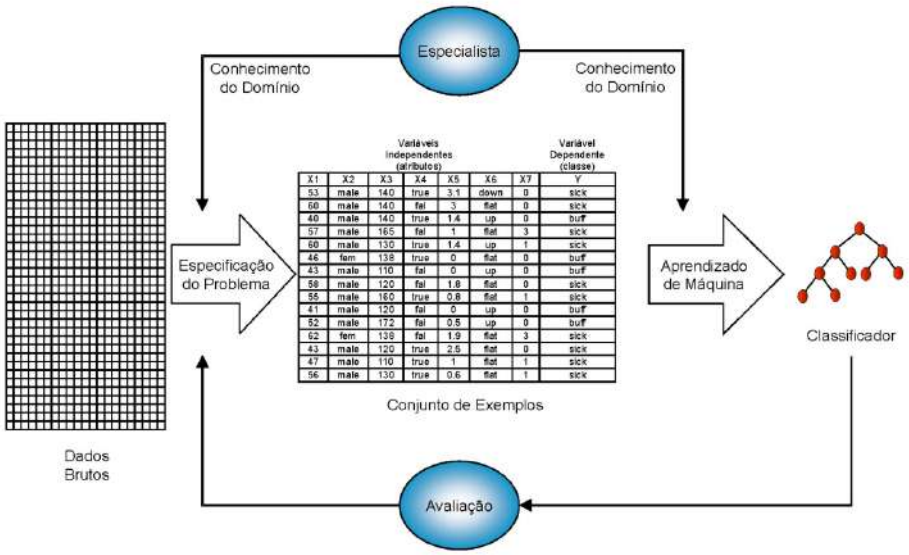

Figura 10 – Processo de classificação. Fonte: (MONARD; BARANAUSKAS, 2003).

2.3.1.1 Questões gerais dos algoritmos de aprendizado supervisionado

Segundo Kotsiantis (2007), o processo que descreve a aplicação de aprendizado de máquina supervisionado para um problema do mundo real é como apresentado no fluxograma representado na Figura 11.

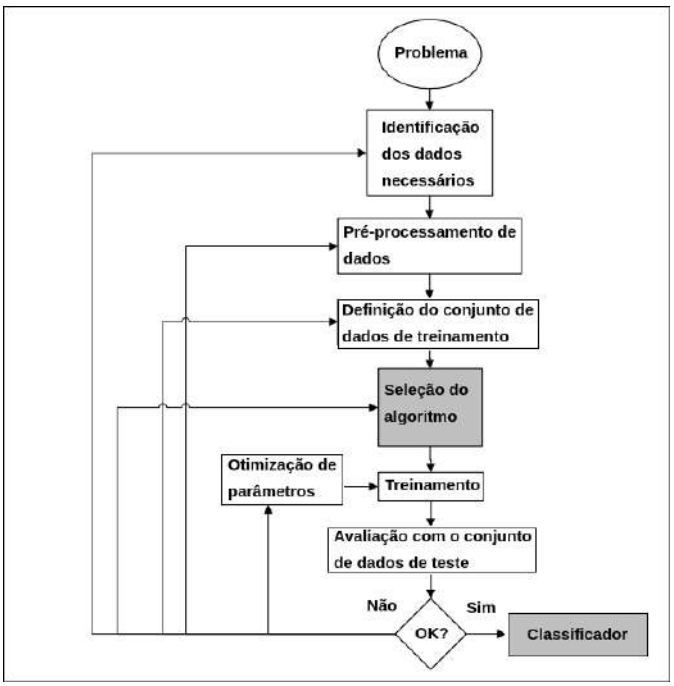

Figura 11 – Aplicação de aprendizado de máquina supervisionada no mundo real. Fonte: (KOTSIANTIS, 2007). Tradução livre pelos autores.

O autor define o primeiro passo como a coleta de dados. Separar ou isolar quais são os campos mais informativos e relevantes para o problema. O segundo passo é o pré-processamento dos dados. Zhang et al. (2002), como citado pelo autor, considera um passo de muita importância, pois os dados podem ter ruídos e valores de recursos ausentes. Há diversas pesquisas referentes a detecção de dados ausentes (ou faltantes) e ruídos e como lidar com isso. Para lidar com esse tipo de problema muitas vezes é feita uma seleção de dados, que é considerada como um problema de otimização que tenta manter a qualidade da mineração minimizando o tamanho da amostra. Segundo Liu e Motoda (2001), conforme citado por Kotsiantis (2007), haverá uma redução nos dados, mas fará com que o algoritmo funcione efetivamente com vários grandes conjuntos de dados.

Em seguida, vem a definição dos dados de treino. Segundo Yu e Liu (2004), conforme citado pelo autor Kotsiantis (2007), define como o processo de identificar e remover o máximo de campos ou características irrelevantes. Isso faz com que os algoritmos de mineração de dados operem mais rápidos e com mais eficácia, pois diminui a dimensionalidade dos dados. De acordo com o Markovitch e Rosenstein (2002), referidos por Kotsiantis (2007), muitas vezes há campos que dependem muito um do outro e isso é ruim, pois influenciam negativamente no modelo de aprendizado de máquina. Uma solução para isso

é a aplicação da técnica de transformação ou construção de novos campos a partir de outros. Isso vai levar a classificadores mais concisos e precisos, além de melhorar a sua compreensibilidade.

Logo após o passo de escolha do algoritmo de aprendizagem, que é de muita criticidade, os dados são treinados. Quando o teste preliminar é julgado satisfatório, um classificador é criado e está pronto para ser usado.

Para se avaliar um classificador é comumente usado a métrica de acurácia (porcentagem de previsões corretas dividido pelo número total de previsões). Segundo Kotsiantis 2007, existem pelo menos três técnicas para calcular a acurácia do classificador. Essas técnicas necessitam de uma base de dados de teste, normalmente obtidas pela divisão do conjunto de dados, que possibilitam o cálculo de acurácia.

Caso a taxa de erro encontrada seja insatisfatória, deve-se retornar para a etapa anterior do processo de classificação supervisionada. Fatores devem ser examinados com o propósito de melhorar o modelo. De acordo com Japkowicz e Stephen (2002), citado pelo Kotsiantis (2007), muitas vezes fatores como: características importantes não estão sendo usadas, base de teste está muito pequena, dimensionalidade do problema é muito alta, foi selecionado um algoritmo inadequado para o problema ou há a necessidade de ajustes em seus parâmetros e ainda o conjunto de dados está desequilibrado. Uma forma comum de se comparar algoritmos para encontrar a melhor escolha é por meio de comparações estáticas da acurácia para um determinado conjunto de dados de teste.

Para os chamados "sistemas inteligentes" a classificação supervisionada é a tarefa mais realizada. Dessa forma, foram desenvolvidas várias técnicas baseadas em Inteligência Artificial, Perceptron e Estatísticas.

2.3.1.2 Método *Ensemble*

Segundo Sirikulviriya e Sinthupinyo (2011), existe uma técnica muito popular de aprendizado de máquina, de grande interesse nas comunidades de mineração de dados, que é chamada de método *ensemble* (conjunto). Ele afirma que é amplamente aceito que a precisão do conjunto de vários classificadores fracos (classificadores com baixa capacidade preditiva) é geralmente melhor que um único classificador, dada a mesma quantidade de dados de treino. Algoritmos eficazes diversos foram inventados para tal função nos últimos 15 anos, como por exemplo: *Bagging* (Ensacamento), *Boosting* (Aumento), *Arching* (Arquear) e *Random Forest* (Floresta Aleatória).

O ***Random Forest*** é um classificador de conjunto (*ensemble*) proposto por Breiman (2001). De forma geral, a maneira utilizada para criação deste classificador é a construção de várias árvores de decisão, em que cada uma utiliza um subconjunto de atributos selecionados aleatoriamente.

Segundo Mitchell (1997), conforme citado por Amal (2014), árvores de decisão categorizam as instâncias ou exemplos, como um arranjo de uma árvore, definindo-as da raiz a um nó de folha específico. Todo nó na árvore representa um teste referente a algum atributo da instância, enquanto cada ramificação descendente do nó representará um valor de um atributo. O autor sugere um exemplo como mostrado na Figura 12. Ele descreve que a árvore de decisão foi construída com base no atributo "perspectiva", em que existem três valores: nublado, ensolarado e chuvoso. Há também valores com subárvores como chuvoso e ensolarado. Ao final será classificado entre brincar e não brincar nesse dia de acordo com a distribuição de exemplos de maior quantidade na folha, que inicialmente nove eram para brincar e cinco para não brincar. Como dito anteriormente, e agora reforçado por Mitchell (1997), a classificação de exemplos em um conjunto distinto de prováveis agrupamentos é denominado como problema de classificação.

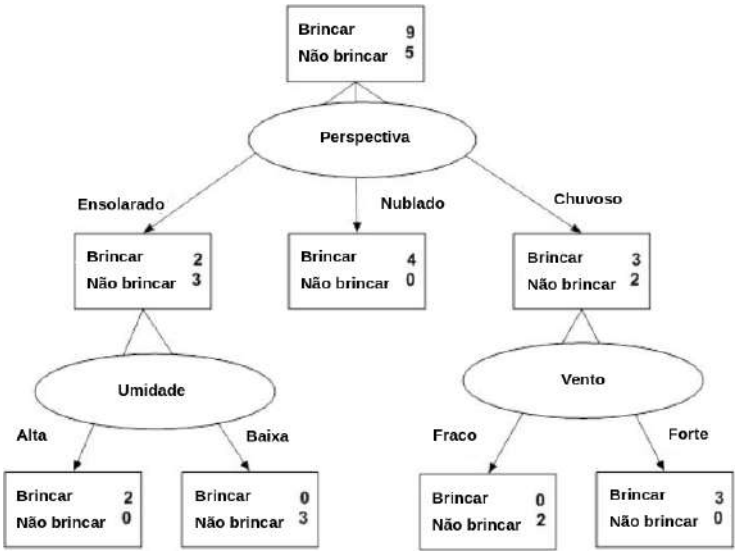

Figura 12 – Exemplo de uma árvore de decisão. Fonte: (ALMANA; AKSOY, 2014). Tradução livre pelos autores.

Para o modelo de árvore é inicialmente alcançada a categorização em grupos de observação. Após isso segue com a pontuação dos grupos específicos gerados. É possível definir modelo de árvore como um processo recursivo em que uma quantidade de unidades estáticas, em um conjunto, são postas em grupos. O agrupamento das unidades existentes dependem de dadas regras de divisão que são progressivas. O principal objetivo da regra de divisão é a maximização da homogeneidade ou a medida da pureza da variável de resposta em seu grupo, conforme elas vão sendo obtidas. A regra de divisão executa uma forma de dividir as observações, em que cada característica a ser dividida e a regra de divisão definida são essenciais no procedimento de divisão do processo. O principal resultado de uma árvore de decisão é a partição final das observações (ALMANA; AKSOY, 2014).

Segundo Thomas (2017), a árvore de decisão é uma técnica de modelagem de aprendizado de máquina não paramétrica amplamente utilizada. Métodos não paramétri-

cos são quando não há suposições subjacentes sobre a distribuição dos erros ou dos dados, ou seja o modelo é construído baseado na observação dos dados.

Segundo Breiman (2001), citado por Drummond (2017), o algoritmo da Floresta Aleatória cria, a partir de um treinamento único, diversas árvores de decisão para fora do subconjunto de dados. Esse treinamento é feito utilizando algoritmo de Ensacamento (*bagging*), método utilizado para melhorar a classificação ou regressão de modelos a partir da estabilidade e precisão da classificação. O método da Floresta Aleatória faz sua predição conforme sua decisão, por meio da contagem de votos dos componentes preditores de cada classe. Dessa forma, é selecionada uma classe vencedora referente do número de votos acumulados para cada classe. Sua estrutura pode ser observada na Figura 13.

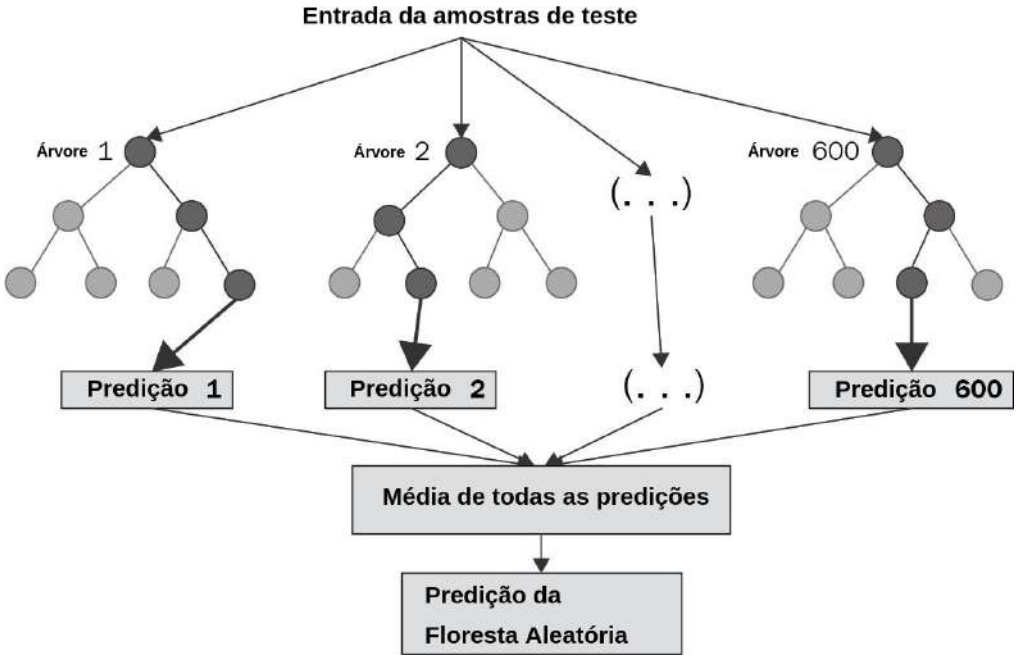

Figura 13 – Representação estrutura da Floresta Aleatória (*Random Forest*). Fonte: (CHAKURE, 2019). Tradução livre pelos autores.

Segundo Mayr, Binder, Gefeller e Schmid (2014), conforme citado por Vecmanis (2019), o algoritmo de Aumento (***Boosting***) é também um classificador de conjunto (*ensemble*) baseado em algoritmos semelhantes ao Floresta Aleatória. Ele cria um modelo preditivo mais preciso por meio de uma otimização passo a passo, que envolve uma sequência de várias iterações analíticas. De forma geral, qualquer classificador fraco pode ser potencialmente melhorado (*boosted*) para se tornar também um classificador forte.

Segundo o Hoare (2019), o aprendizado de máquina aumentado (*boosting*) começa treinando um modelo inicial, como por exemplo utilizando uma árvore de decisão. Em seguida, é construído um segundo modelo que se concentra em prever com melhor precisão os casos em que o primeiro apresentou baixa precisão. Dessa forma, é esperado que a combinação desses dois modelos seja melhor que apenas um. A cada modelo sucessivo, tenta-se

corrigir as deficiências da combinação do conjunto de árvores de decisão anteriores.

O *Gradient boosting* (Aumento de gradiente) corresponde a um algoritmo de aprendizado de máquina aumentado. A ideia principal dele é encontrar um melhor modelo posterior, com a combinação de outros modelos, e minimizar o erro de previsão total (Figura 14). Ele funciona estabelecendo saídas (*targets*) para o modelo posterior a fim de minimizar o erro. Para calcular essa saída a ser estabelecida vai depender de quanto a alteração da previsão desse caso (amostra do conjunto de dados de treino) afeta o erro total de previsão. Se uma pequena alteração na previsão de um caso causar uma grande queda no erro, o próximo resultado desejado do caso será um valor alto. As previsões do novo modelo que estão próximas de seus objetivos reduzirão o erro. Se uma pequena alteração na previsão de um caso não causar alteração no erro, o próximo resultado desejado do caso será zero, pois alterar essa previsão não diminuirá o erro. O *gradient boosting* (aumento de gradiente) é um algoritmo de *gradient descent* (gradiente descendente), que é assim denominado porque os resultados de destino para cada caso são definidos com base no gradiente descendente do erro em relação à previsão (HOARE, 2019).

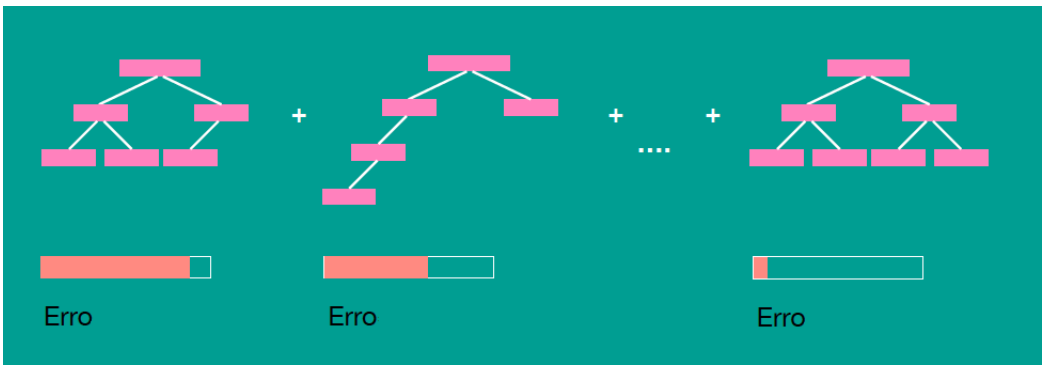

Figura 14 – *Gradient Boosting*. Fonte: (CHEPENKO, 2019). Tradução livre pelos autores.

Para deixar mais clara a ideia do conceito de *gradient descent* (gradiente descendente), Ben (2017) apresenta o seguinte exemplo: uma função de perda (*loss function*), L, capaz de calcular o erro de previsão total, se apresenta como na equação (2.2).

$$L(x_1, x_2) = \frac{1}{2}(x_1 - 15)^2 + \frac{1}{2}(x_2 - 25)^2 \tag{2.2}$$

O objetivo é encontrar o par (x_1, x_2) que minimiza a função L. Ela pode ser interpretada como calculando o erro ao quadrado para dois pontos de dados, 15 e 25, com dois valores de previsão, x_1 e x_2. Para essa função demonstrada acima é possível minimizá-la diretamente, mas o *gradient descent* (gradiente descendente) nos permitirá minimizar funções de perda mais complicadas que não são possíveis de minimizar diretamente.

2.4 Conclusão

Perante a proposta do presente trabalho, que refere-se a uma solução tecnológica para um problema ligado a um *e-commerce* imobiliário, foram apresentados neste capitulo, além do contexto imobiliário, as principais abordagens utilizadas para construção de um sistema de recomendação.

Para a construção da proposta a seguir, será necessário o entendimento de todo processo que refere-se a venda de imoveis online, além de todas as abordagens de recomendação apresentadas, seguido do uso de algoritmos de aprendizado de máquina, que proporcionam melhores soluções diante do contexto do presente trabalho.

3 Proposta

Este capítulo descreve as minúcias da implementação deste trabalho, abrangendo, metodologias e técnicas, cronogramas, tecnologias, arquitetura, simulação e os resultados obtidos.

No apêndice C deste trabalho tem disponível a simulação elaborada quando da submissão inicial dessa proposta, tendo tal simulação o intuito de validar e averiguar a viabilidade da proposta do trabalho.

3.1 Metodologia

O ser humano, desde os seus primeiros anos até seus últimos dias de vida busca por mais conhecimento de alguma forma, sendo as dúvidas e questionamentos as principais causas para descobrir respostas para suas perguntas (CARLOS; ORSI; CRUZ, 2018). A pesquisa auxilia o ser humano nessa trajetória de busca por mais conhecimento, para solucionar problemas do mundo real (GIL, 2008).

Segundo Gil (2002), o objetivo de pesquisa se divide em três categorias: **Exploratória**, **Descritiva** e **Explicativa**. A primeira possibilita uma maior familiaridade com o contexto da problemática, com o princípio de aprimorar ideias para a solução. A segunda permite a descrição das características de algum determinado fenômeno ou um grupo de seres vivos ou relação entre variáveis. A terceira promove a identificação dos fatores que ocasionam ou ajudam para o acontecimento de determinado fenômeno.

Conforme Freitas (2013), para a abordagem do problema existem dois tipos: **Quantitativo** e **Qualitativo**. A primeira considera que tudo pode ser quantificado, isto é, converter informações em números para poder ser classificado e analisado posteriormente, sendo então interessante o uso de técnicas estatísticas. A segunda não exige técnicas estatísticas, pois a ideia principal dela é analisar indutivamente os dados, sendo que o processo e seu significado são os pontos de foco.

De acordo com Freitas (2013), sobre a natureza da pesquisa, ela pode ser: **Básica** e **Aplicada**. A primeira tem o intuito de produzir novos conhecimentos úteis para o crescimento da ciência, sem ter uma previsão de aplicação prática daquilo que foi pesquisado. A segunda também tem o objetivo de gerar novos conhecimentos, mas visado a aplicação prática para a solução de problemas específicos da vida real.

Um dos procedimentos de pesquisa que existem é denominado Pesquisa-Ação, que consiste em quando há um interesse coletivo para solucionar um determinado problema, no qual os pesquisadores e os participantes estão envolvidos nesse problema de forma coo-

perativa ou participativa, com o intuito de solucionar ou no mínimo esclarecer o problema da situação, aumentando o nível de conhecimento dos pesquisadores e dos participantes (FREITAS, 2013).

Segundo Thiollent (1997), citado por Costa, Politano e Pereira (2014), as etapas da Pesquisa-ação são: **Diagnóstico**, **Planejamento da ação**, **Execução da ação** e **Avaliação da ação**. A primeira, identificar o problema no contexto que o pesquisador tem interesse em resolver. A segunda, arquitetar ações alternativas para a solução do problema, melhor dizendo, desenvolver um planejamento de ação. A terceira, aplicação do planejamento da fase anterior com o auxílio de um roteiro. A quarta, monitorar as ações implementadas para compreender se os resultados são os esperados.

Para este trabalho foi decidido aplicar como objetivo de pesquisa a categoria exploratória, pois como o conhecimento sobre sistemas de recomendação não eram suficientes a solução do problema apresentado seria necessário adquirir mais conhecimento sobre os devidos conceitos. Assim sendo, houve um empenho maior em realizar pesquisas bibliográficas para achar conteúdos da área e trabalhos semelhantes, podendo ainda ser utilizada dessas pesquisas para explorar ou conhecer novos conteúdos que possam vir a serem necessários (MORETTI, 2018).

Para o contexto do presente trabalho foi considerado a abordagem quantitativa, visto que a ideia principal do mesmo é apresentar resultados de natureza estatística como a métrica CVR (*Conversion Rate*) que será aplicada ao final e abordada mais a frente no trabalho na seção 3.3.2. Com essa métrica será possível medir o grau de melhoria aplicando-se uma abordagem de recomendação comparado com uma já existente em outro sistema.

Como o objetivo deste trabalho é desenvolver um produto específico (Sistema de Recomendação) para o cliente e colaborador Liva, sendo necessário também obter mais conhecimento sobre o contexto do problema em questão para elaboração da solução adequada. Assim, foi considerado para a natureza de pesquisa a categoria aplicada.

Para o procedimento de pesquisa foi utilizada a pesquisa bibliográfica no início do trabalho para conhecer os conceitos, abordagens e técnicas empregadas nos Sistema de Recomendação atualmente.

A organização para elaboração desse projeto realizou as fases de diagnóstico e planejamento do produto como a primeira etapa, sendo identificada como a parte do TCC1 (Trabalho de Conclusão de Curso 1), enquanto que a fase de execução da ação e avaliação foram executadas na etapa correspondente ao TCC2 (Trabalho de Conclusão de Curso 2) do projeto.

Na fase de diagnóstico foram realizadas reuniões presenciais e remotas com o gerente de produtos da Liva, sendo efetuados o levantamento de informações do problema.

Reuniões presenciais com o orientador também aconteceram envolvendo, principalmente, as questões de pesquisa e objetivos, métodos e prazos para a elaboração do projeto e o posterior desenvolvimento do produto.

A fase de maior duração foi a de planejamento, pois foi realizado uma extensa pesquisa e coleta de referências bibliográficas sobre os Sistemas de Recomendação, a fim de se decidir quais abordagens são interessantes para o contexto do trabalho, além da pesquisa sobre soluções similares. Com isso foi construído o referencial teórico e efetuado o planejamento da proposta sobre como seria a arquitetura de software (produto), metodologias de pesquisa e de desenvolvimento, levantamento de funcionalidades (requisitos), protótipo de baixa e alta fidelidade e as tecnologias envolvidas.

A execução da ação será a implementação do sistema de recomendação como um serviço a parte dos serviços da *startup* Liva, sendo sintetizadas as definições desse trabalho no Quadro 3.1.

Quanto à abordagem	**Quanto ao objetivo**	**Quanto à natureza**	**Quanto ao procedimento**
Pesquisa quantitativa	Pesquisa exploratória	Pesquisa aplicada	Pesquisa bibliográfica e Pesquisa-ação

Quadro 3.1 – Escolha de metodologia de pesquisa.

Cada tarefa do fluxo de atividades do TCC1 (Figura 15) foi sucintamente descrita a seguir:

- **Definir tema para o TCC:** foi discutido a área em que seria feito o trabalho até chegar em uma decisão final;
- **Levantar referencial teórico inicial:** foi estudado de forma introdutória sobre o tema escolhido para averiguar se é possível realizar a proposta do trabalho;
- **Elaborar proposta da monografia:** estabelecer escopo do trabalho, objetivo, questões de pesquisa, justificativa e metodologia que seria usada, podendo haver alterações de acordo com o andamento do trabalho;
- **Estabelecer metodologia de pesquisa:** após o início do estudo do tema selecionado anteriormente foi possível definir quais metodologias de pesquisa seriam usadas;
- **Pesquisar sobre sistema de recomendação:** envolve a pesquisa de artigos, revistas e de trabalhos científicos sobre sistemas de recomendação de forma detalhada para servir de fundamento para os objetivos definidos anteriormente;
- **Estabelecer suporte tecnológico:** seleção das tecnologias que seriam usadas para o desenvolvimento do trabalho;

- **Estabelecer metodologia de desenvolvimento:** seleção e planejamento da metodologia de desenvolvimento que seria utilizada para implementação da proposta;
- **Implementar simulação:** fase de implementação inicial da solução para validação da proposta e identificação de possíveis riscos que possam interferir no desenvolvimento do trabalho;
- **Revisar TCC1:** revisar todos os tópicos da parte escrita do trabalho para verificar se seria necessário algum ajuste ou complementação;
- **Apresentar TCC1:** apresentação do tema para a banca examinadora com o intuito de aprimorar a proposta, caso necessário, e inicio da segunda etapa (TCC2).

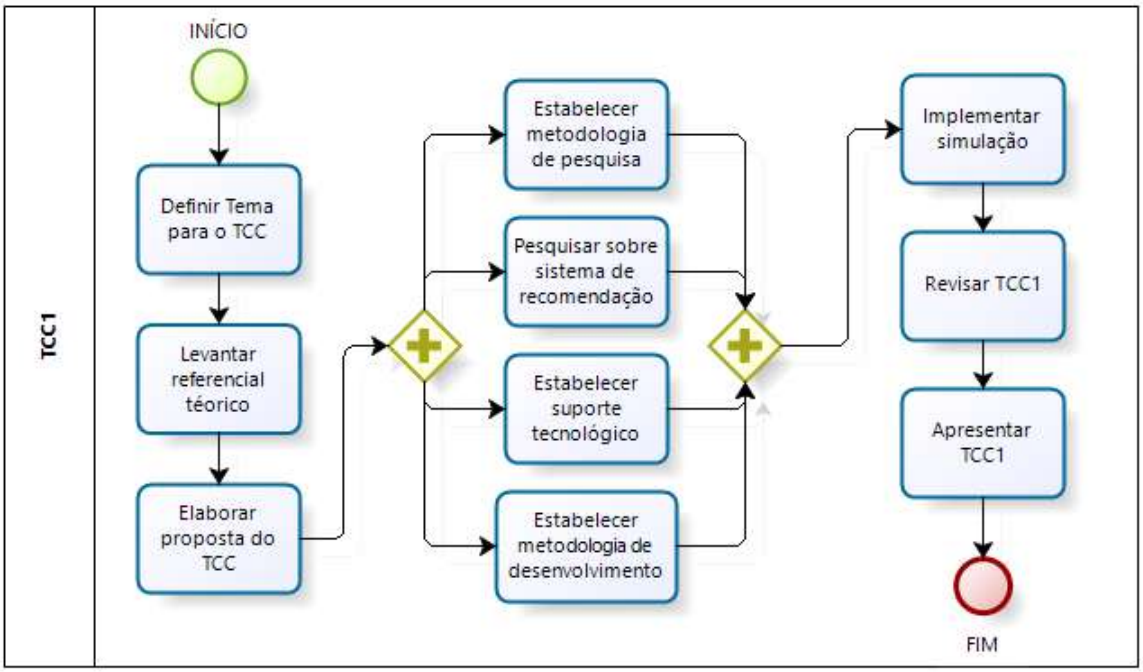

Figura 15 – Fluxo de atividades da primeira etapa (TCC1).

Na Figura 16 pode ser visto o fluxo de tarefas para condução do TCC2 e uma breve descrição de cada tarefa é apresentada a seguir:

- **Realizar as devidas correções:** após apresentação para a banca examinadora do TCC1 foram realizados os ajustes e complementações solicitadas;
- **Desenvolver o sistema de recomendação:** implementar o sistema de recomendação usando as três abordagens estudadas (*collaborative filtering, content-based filtering e critiquing-based*), sendo duas delas elaboradas com aprendizado de máquina. Será implementada a parte de virtualização de ambiente completo, tanto do ambiente de desenvolvimento quanto de produção, a integração contínua e o *deploy* (entrega) automático no ambiente de homologação;

- **Implantar sistema de recomendação em ambiente de produção:** após o desenvolvimento do sistema de recomendação proposto no trabalho foi feita implantação do mesmo em ambiente de produção da Liva, para assim ser utilizado pelos usuários.
- **Averiguar resultados obtidos:** com a coleta de dados através da implantação do sistema de recomendação em ambiente de produção, foi averiguada com a métrica CVR (seção 3.3.1.3) a eficácia do sistema proposto;
- **Revisar TCC2:** verificar se os objetivos do trabalho foram cumpridos como um todo e revisar todos os tópicos da monografia final;
- **Apresentar TCC2:** apresentação do projeto proposto (etapa TCC1) e o seu desenvolvimento com os resultados da etapa final (TCC2).

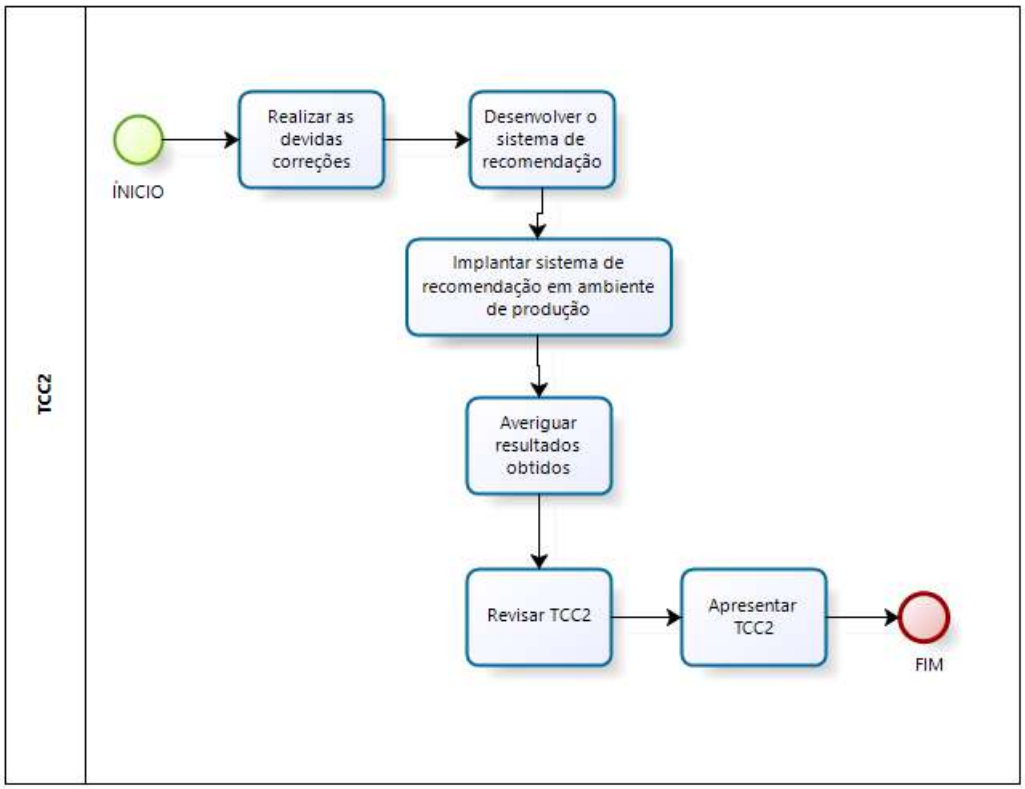

Figura 16 – Fluxo de atividades da segunda etapa do projeto (TCC2).

Para o presente trabalho foi escolhido como metodologia de desenvolvimento o *Scrum*, metodologia ágil e muito utilizada nos últimos anos pela indústria de desenvolvimento de software, acompanhado com a metodologia Kanban.

3.1.1 *Scrum*

O *Scrum* possibilita um tipo de gerenciamento organizacional ágil e oferece às organizações de software uma nova e dinâmica gestão para não só sobreviver, mas ter sucesso no setor que constantemente tem mudanças e inovações (KARABULUT; ERGUN, 2018).

Sua metodologia é baseada na ideia de que muitos processos no desenvolvimento de software são difíceis de serem previstos, sendo assim, a flexibilidade adotada por essa metodologia ágil facilitam muito mais a implementação das aplicações (KARABULUT; ERGUN, 2018).

Como este trabalho elaborará um software que deve ser entregue com uma alta qualidade esperada pelos superiores da Liva, e deve ser colocado que imprevistos são bem prováveis de acontecer, além de mudanças no decorrer do trabalho, é necessário empregar um *framework* que atenda a essas características, ou seja, uma metodologia em que seja permitido entregar subconjuntos do projeto a cada período de tempo e que possa ser melhorado no andamento do desenvolvimento junto com um *feedback* contínuo.

O princípio do *Scrum* é entregar produtos de alta qualidade, após vários períodos de desenvolvimento chamados de *Sprints* (ciclos) que geralmente tem duração entre duas e quatro semanas, mas para esse trabalho terá a duração de 2 semanas. Em cada início de um ciclo será realizado uma reunião chamada de *Sprint Planning* (planejamento do ciclo), na qual é feito o planejamento do que será implementado e entregue ao cliente. Da segunda *Sprint* em diante é realizado uma outra reunião chamada de *Sprint Review* (revisão do ciclo) que seria a revisão do ciclo passado com o propósito de rever o que foi feito na *Sprint* antes de realizar o *Sprint Planning* (KARABULUT; ERGUN, 2018).

Como instrumento para organização do que é necessário fazer é realizado o *Product Backlog* (lista de funcionalidade do produto), que consiste em uma lista com todas as funcionalidades que devem ter com base os requisitos levantados com o cliente. Cada funcionalidade será agrupada pelo seu nível de prioridade. Essa lista é dinâmica e sofre atualização de acordo com o andamento do projeto, identificando mais necessidades, como correções e melhorias.

A lista de funcionalidades que é planejada para ser feita no período da *Sprint* é chamada de *Sprint Backlog* (lista de funcionalidades do ciclo), sendo recomendado possuir pelo menos uma melhoria de alta prioridade ao cliente que foi identificada anteriormente (KARABULUT; ERGUN, 2018).

Cada funcionalidade do *Product Backlog* é descrita como *User Story* (estória de usuário), na qual uma representação de cada necessidade do cliente é gerada sob seu ponto de vista (KNOWLEDGE21, 2019).

3.1.2 Kanban

Kanban (cartão) foi uma outra metodologia ágil utilizada no presente projeto. Ela é muito usada junto com o *Scrum*, sendo um sistema visual para controlar a gestão de atividades em um processo. O Kanban possibilita a observação fácil para o controle de fluxo de trabalho, balancear a ordem dos processos e respeitar a produtividade da equipe (ARTIA, 2019).

O Kanban é divido em três colunas: *to do* (fazer), *doing* (fazendo) e *done* (concluído), mas é possível colocar mais colunas caso seja desejado, para assim, ficar mais relacionado ao contexto do processo. Para o presente trabalho foram utilizadas seis colunas: *Product Backlog* (lista de funcionalidades do produto), *Sprint Backlog* (lista de funcionalidades do ciclo), *Doing* (fazendo), *Testing* (teste), *Review* (revisão) e *Done* (concluído). Para cada coluna há cartões (*cards*) que representam uma atividade que está em uma dessas respectivas fases do processo. De acordo com o andamento, os *cards (cartão)* foram movidos para as fases seguintes (ARTIA, 2019).

Para realização dessa metodologia foi utilizada a ferramenta ZenHub que está descrita na seção 3.4.22.

3.1.3 *Product Backlog* inicial

Para obter um conhecimento inicial sobre o projeto, foram levantado as seguintes *features* (características), *user stories* (estórias de usuário) e *technical stories* (estória técnica):

- Características (*Features*):
  - **FE01 - Recomendador baseado em crítica:** possibilitará ao usuário receber recomendações de imóveis baseadas em críticas sobre características de imóveis feitas pelo próprio usuário;
  - **FE02 - Recomendador baseado em aprendizado de máquina:** permitirá ao usuário receber recomendações de imóveis baseado nas ações dele no sítio virtual.
- Estória do usuário (*User stories*):
  - **FE01US01 - Visualizar perfil de busca:** eu, como usuário, desejo visualizar meu perfil de busca para ter conhecimento de como está configurado;
  - **FE01US02 - Alterar perfil de busca:** eu, como usuário, desejo alterar meu perfil de busca para ajustar as preferência e receber recomendações mais precisas;

  - **FE01US03 - Visualizar recomendações na página principal:** eu, como usuário, desejo visualizar os imóveis recomendados com base nas minhas preferências configuradas no perfil de busca;
  - **FE01US04 - Criticar características da propriedade:** eu, como usuário, desejo criticar as características de um ou vários imóveis para ter recomendações mais adequadas com o que eu desejo;
  - **FE02US05 - Visualizar recomendações na página do imóvel:** eu, como usuário, desejo visualizar recomendações de imóveis baseadas nas minhas ações no sítio virtal e na página de detalhes do imóvel para diminuir o tempo de procura do imóvel ideal para mim;

- Estórias técnicas (*Technical stories*):

  - **FE02TS01 - Registrar ações do usuário no sítio virtual:** eu, como desenvolvedor desejo registrar as ações do usuário no site para poder predizer quais são os imóveis mais prováveis da preferência do usuário.

3.1.4 Requisitos não funcionais

Foram selecionados os requisitos não funcionais do sistema de recomendação proposto em categorias. Foram escolhidas as consideradas mais importantes, que são elas: usabilidade, desempenho, segurança, disponibilidade, portabilidade e confiabilidade.

3.1.4.1 Usabilidade

O sistema de recomendação contém dois componentes na plataforma *web* Liva. Ambos possuindo um design atrativo e intuitivo para intrigar o usuário ao uso e também responsivo para poder ser visualizado em qualquer tela, independente do dispositivo ser um computador, *smartphone* ou *tablet*. Também deverá conter ícones para cada atributo do imóvel, a fim de obter uma melhor associação entre cada atributo do imóvel.

3.1.4.2 Desempenho

No módulo de Aprendizado de Máquina, devido ao alto volume de dados e processamentos demandados do treino e predição do modelo, na qual é recomendado ser processado em menos de 10 segundos para o usuário que será descrito em 3.3.1.2, são demandadas instâncias de servidores adequadas ao contexto. Para isso, a instância a ser escolhida deverá ter os seguintes recursos computacionais:

- 2 Unidades de vCPU (Unidade central de processamento virtual). Cada vCPU representa um único núcleo físico da CPU pelo sistema operacional da VM;

- 8 gigabytes de memória RAM;
- 20 gigabytes de espaço em disco (crescendo horizontalmente se necessário).

3.1.4.3 Segurança

A API do sistema poderá ser acessada por sistemas externos de domínio da Liva. Este acesso deve ser seguro, com autenticação em nível do servidor e em nível da aplicação. Os usuários devem ter conta na plataforma da Liva, para assim, terem acesso as funcionalidades do sistema de recomendação.

3.1.4.4 Disponibilidade

O novo sistema de recomendação precisa funcionar vinte e quatro horas por dia, sete dias por semana (24 x 7) na operação da Liva, pois há, constantemente, clientes acessando o sistema. Dessa forma, é necessário que a infraestrutura suporte esse período e cresça horizontalmente em seus recursos de acordo com o que é demandado, para assim se manter sempre funcionando.

3.1.4.5 Portabilidade

O sistema proposto deverá rodar nos navegadores mais populares, como: Mozilla Firefox, Google Chrome, Microsoft Edge e Opera. Navegadores mais antigos como o Internet Explorer podem apresentar instabilidade na utilização do site, por não haver suporte com ES5 que é requisito da tecnologia da plataforma.

3.1.4.6 Confiabilidade

O sistema de recomendação deve ser um sistema eficiente que apresente sempre recomendações adequadas, que representem o mais próximo do perfil do cliente. Dessa forma, o modelo de Aprendizado de Máquina deve apresentar métricas de qualidade, e assim apresentar recomendações coerentes, fazendo os clientes se sentirem mais confiantes em relação ao uso do sistema.

3.1.5 Papéis

Product Owner (Dono do Produto). É responsável por potencializar o retorno sobre o ROI (*Return over Investment* - Retorno Sobre Investimento), e para isso é necessário que esse papel identifique as características do produto, decidindo quais são as funcionalidades com mais prioridades, criando uma lista ordenada com base na prioridade para a próxima *Sprint* e re-priorizando e refinando continuamente essa lista. Em síntese, o dono do produto é responsável pelos lucros e perdas do produto (SUTHERLAND, 2010).

Esse papel será desempenhado pelo gerente de produtos da Liva, pois ele tem a visão de produto deste trabalho que está sendo desenvolvido, além de também ter conhecimento de Engenharia de Software.

Scrum Master (Mestre Scrum): Tem como atividades ensinar o *Scrum* para a equipe do produto e aplicá-lo de fato. Seu objetivo principal é ajudar a equipe a alcançar o sucesso, protegendo-os de possíveis interferências externas. Além disso, ele também educa e orienta o dono do produto e a todos do grupo no uso da metodologia ágil. Quando necessário ele ajuda a liderar a organização nas difíceis mudanças fundamentais para o sucesso do desenvolvimento ágil (SUTHERLAND, 2010).

Quem irá desempenhar esse papel será o professor e orientador, por ser quem vai acompanhar e auxiliar o andamento do trabalho.

Time de desenvolvimento: Serão as pessoas que farão a implementação do produto, fazendo entregas a cada *Sprint* (SUTHERLAND, 2010). Esse papel será desempenhado pelos orientandos do trabalho (estudantes envolvidos com o TCC), por possuírem conhecimento técnico sobre as tecnologias, metodologias, além do contexto do trabalho.

3.1.6 Fluxo de desenvolvimento

Na atividade "Desenvolver o sistema de recomendação" da Figura 16 foi realizado o fluxo de desenvolvimento representado a seguir (Figura 17).

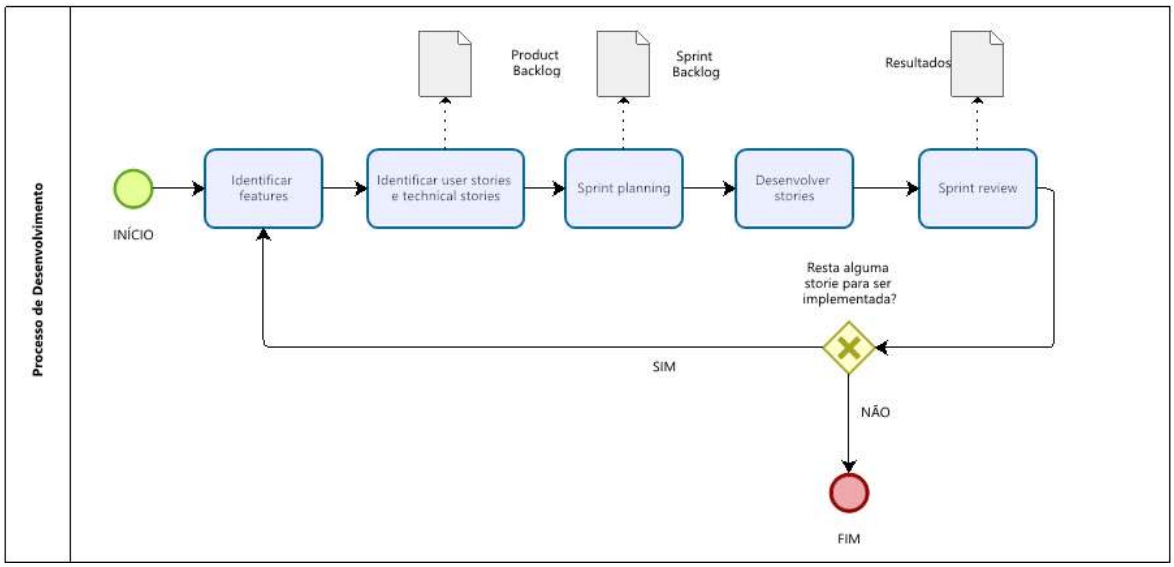

Figura 17 – Fluxo de desenvolvimento do trabalho.

A descrição resumida de cada tarefa se encontra relacionada a seguir:

- **Identificar *features* (características):** averiguar quais são os módulos de requisitos do projeto existentes.

- **Identificar *user stories* (estórias de usuário) e *technical stories* (estórias técnicas):** identificar as estórias de cada *feature* (característica) definida anteriormente e priorizá-las com o nível de importância para o cliente;
- ***Sprint planning* (planejamento do ciclo) :** planejar quais estórias serão desenvolvidas na *Sprint*, sendo seguida a ordem de prioridade descrita no *product backlog* (lista de funcionalidades do produto);
- **Desenvolver *stories* (estórias):** desenvolver as estórias planejadas para aquela *Sprint* até a conclusão das mesmas. A conclusão se define em implementar, testar e verificar se está de acordo com os critérios de aceitação da estória definida;
- ***Sprint review* (revisão de ciclo):** verificar quais estórias foram realmente feitas e quais estão pendentes, em que se averígua a necessidade de criar novas, seja estória de usuário ou estória técnica.

3.2 Cronograma

Foi planejado dois cronogramas para serem seguidos no TCC1 e no TCC2 com intuito de dividir as atividades durante os respectivos semestres. O primeiro cronograma é referente a primeira parte do projeto que abrange todas as atividades referentes ao TCC1 e o segundo cronograma é referente as atividades que foram propostas para o TCC2. A utilização do cronograma se faz necessária para ter um gerenciamento de tempo das atividades que precisam ser realizadas para a conclusão deste trabalho. É possível observar nas tabelas a seguir (Tabela 1 e Tabela 2) como será organizado as atividades do TCC1 e do TCC2 respectivamente.

Tabela 1 – Cronograma do TCC1.

	Ago	**Set**	**Ou**	**Nov**	**Dez**
Escrever monografia	X	X	X	X	
Estudo inicial do tema	X				
Definir de escopo e objetivo	X				
Desenvolver proposta		X	X		
Determinar metodologias			X		
Revisar TCC1			X	X	
Apresentar TCC1					X

Tabela 2 – Cronograma do TCC2.

	Mar	Abr	Mai	Jun	Jul
Efetuar revisão da banca	X				
Desenvolver o sistema de recomendação	X	X	X		
Implantar sistema de recomendação em ambiente de produção			X		
Averiguar resultados obtidos		X	X		
Revisar TCC2		X	X		
Apresentar TCC2			X		

3.3 Sistema de recomendação proposto

O sistema proposto consiste de um novo sistema de recomendação para a plataforma Web Liva.vc. Esse sistema de recomendação seguirá duas abordagens: sistema de recomendação baseado em crítica (seção 2.2.3.5 desse trabalho) e sistema de recomendação baseado em aprendizado de máquina (seção 2.3 desse trabalho). Por se tratar do uso de duas abordagens, isso faz com que se torne um sistema de recomendação híbrido (seção 2.2.3.4 desse trabalho). Para fazer a combinação dessas duas abordagens é utilizada a técnica de hibridação *mixed* (mista), que como já abordado, refere-se a recomendações de diferentes recomendadores que são apresentadas juntas.

Assim, o sistema proposto estará combinando essas duas abordagens que irão englobar diferentes páginas do sítio virtual da Liva. Existem duas principais páginas que irão ser atualizadas para o novo sistema de recomendação: página principal do cliente (Figura 18) e página de detalhes de uma propriedade (Figura 19).

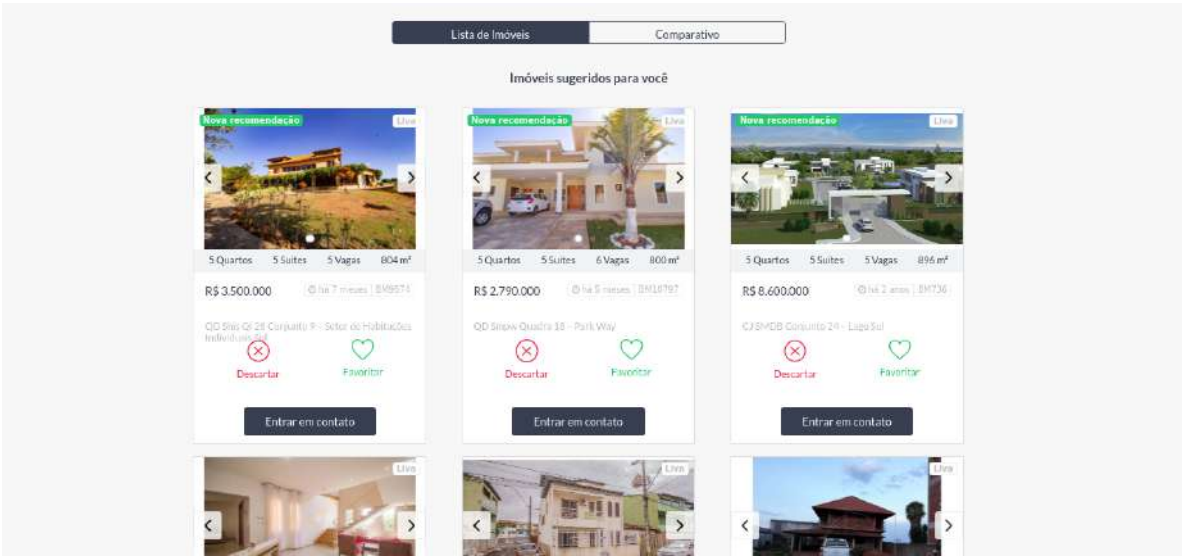

Figura 18 – Página principal do cliente. Fonte: (LIVA, 2019).

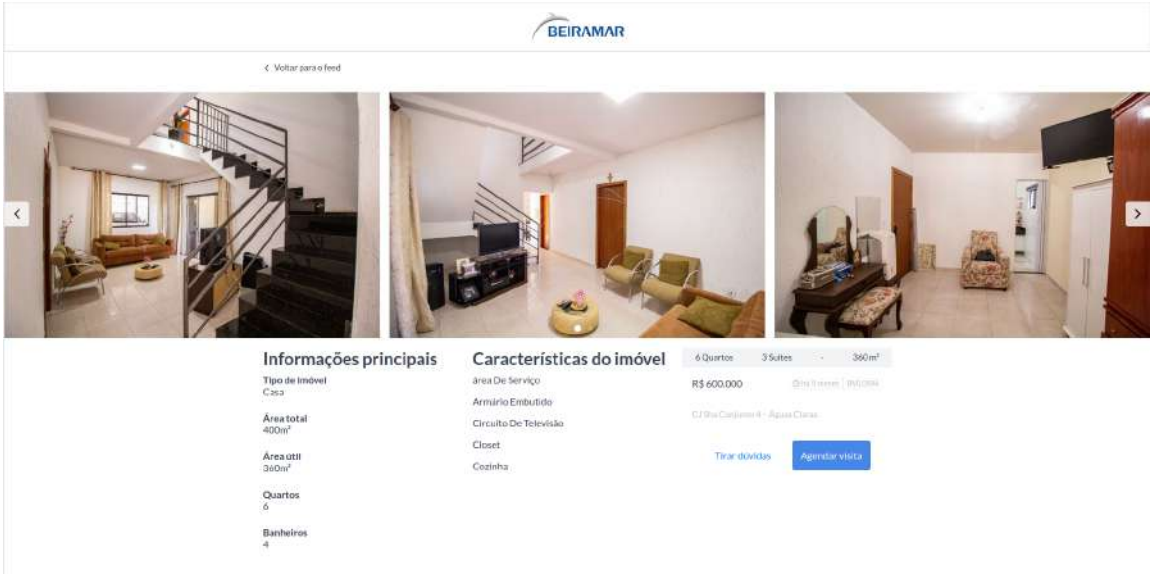

Figura 19 – Página de detalhes da propriedade. Fonte: (LIVA, 2019).

Na página principal existe uma ferramenta de busca denominada “Perfil de Busca” e pode ser acessada por uma janela, como mostrado na Figura 20.

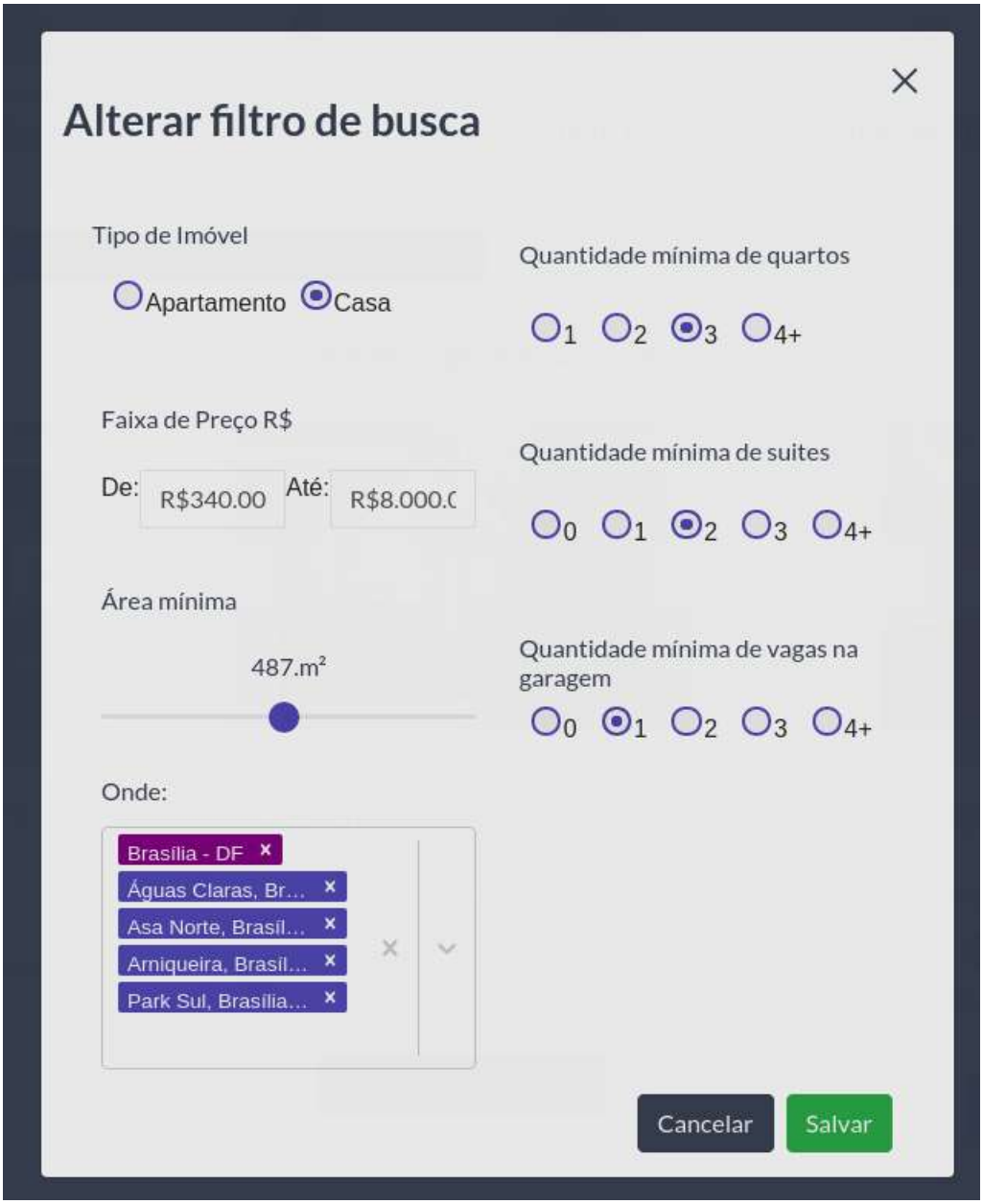

Figura 20 – Filtro de busca. Fonte: (LIVA, 2019).

Aproveitando-se dessa ferramenta de busca para implementação do sistema de recomendação baseado em crítica será necessário alguns ajustes. Primeiramente, serão colocados novos campos para determinar a importância de cada característica de preferência do usuário. Além disso, será adicionado um novo campo de quantidade referente a número de banheiros. Os campos que representam a quantidade das características serão alterados para quantidades fixas e não mínimas. A localidade irá se referir somente a um bairro. Essas alterações podem ser vistas no protótipo elaborado na Figura 21.

Figura 21 – Protótipo do filtro de busca.

Ainda na página principal é possível observar a existência de um *feed* (lista com conteúdo atualizado periodicamente), em que se encontram as recomendações do atual recomendador usado pela Liva. Esse *feed* servirá para visualizar as recomendações do novo recomendador baseado em crítica.

A página de detalhes de uma propriedade é acessível por meio de um clique em sua imagem no *feed* da página principal. Nela é possível visualizar todas as características da propriedade. Além disso, é possível entrar em contato com o corretor de duas formas diferentes: pedindo mais informações da propriedade e agendando uma visita. Para o recomendador baseado em crítica será atualizado o componente de visualização de características da propriedade para que possam ser feitas críticas pelo usuário sobre essas características disponibilizadas para o acesso. Esse novo componente de crítica pode ser observado pelos protótipos mostrados nas Figuras 22, 23, 24 e 25, respectivamente.

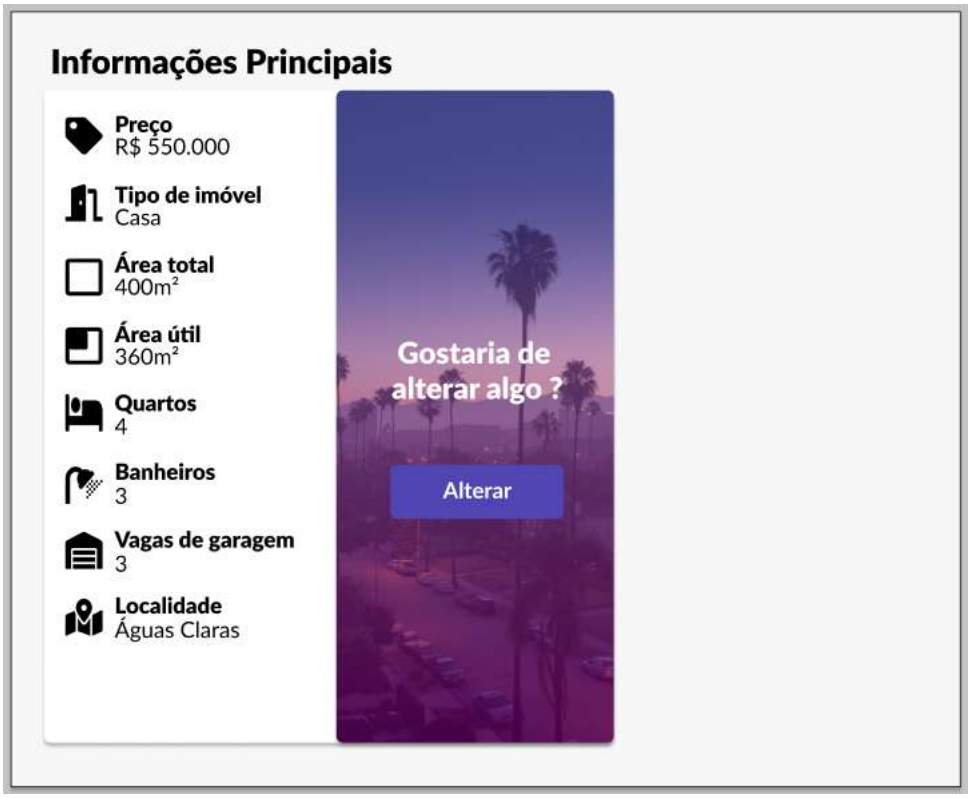

Figura 22 – Detalhes do imóvel com menu de crítica parte 1.

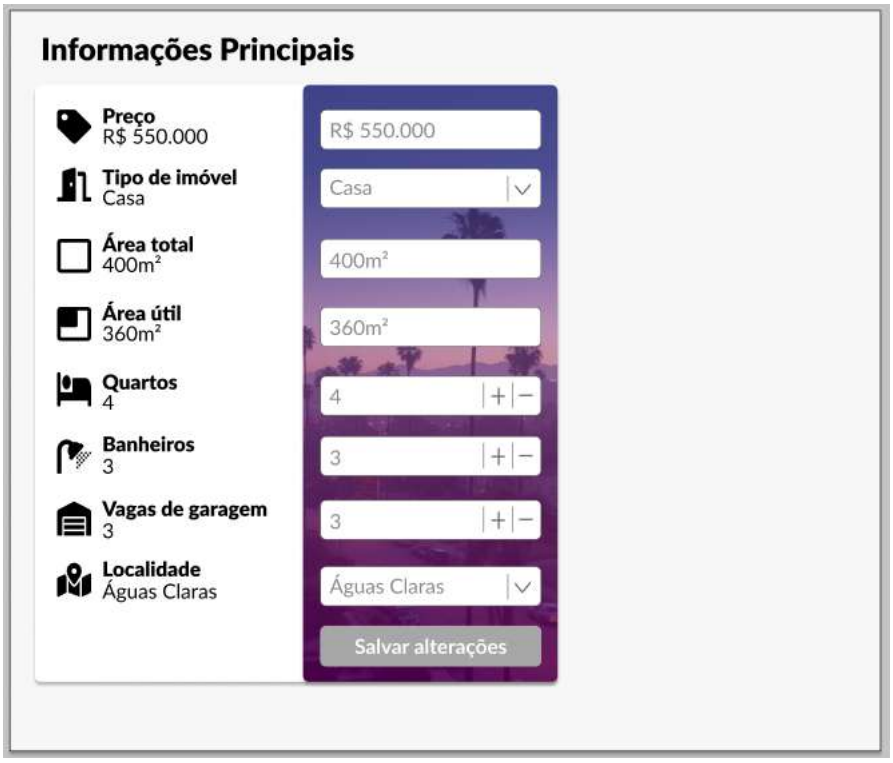

Figura 23 – Detalhes do imóvel com menu de crítica parte 2.

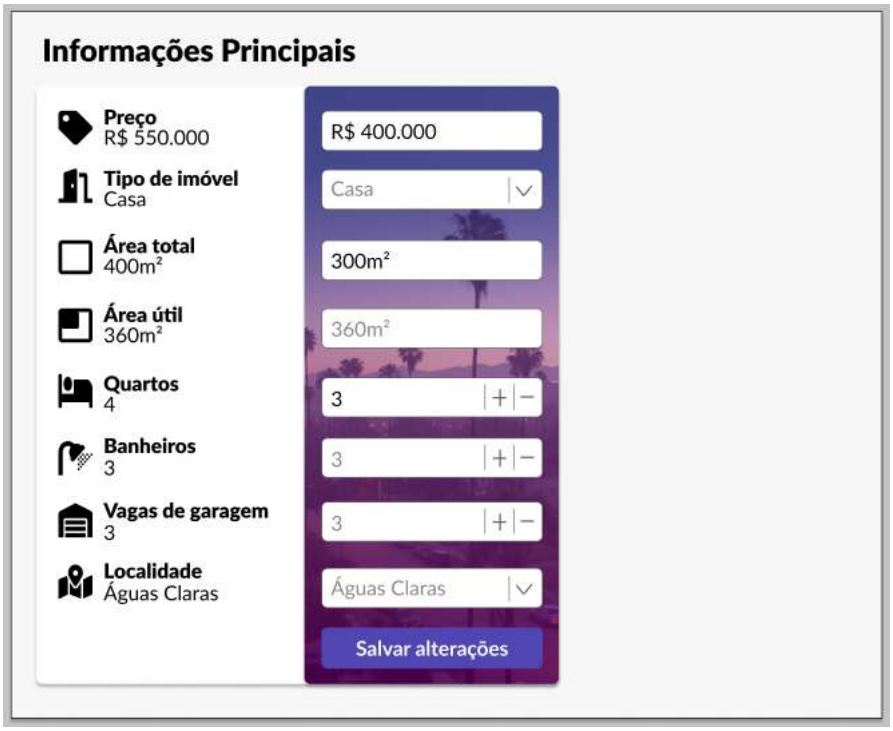

Figura 24 – Detalhes do imóvel com menu de crítica parte 3.

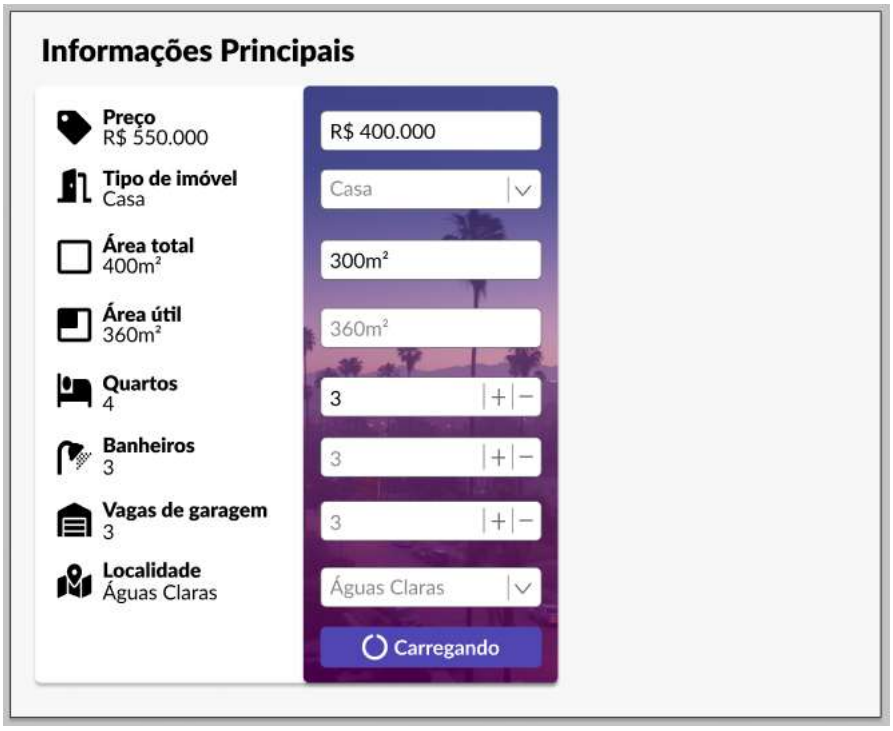

Figura 25 – Detalhes do imóvel com menu de crítica parte 4.

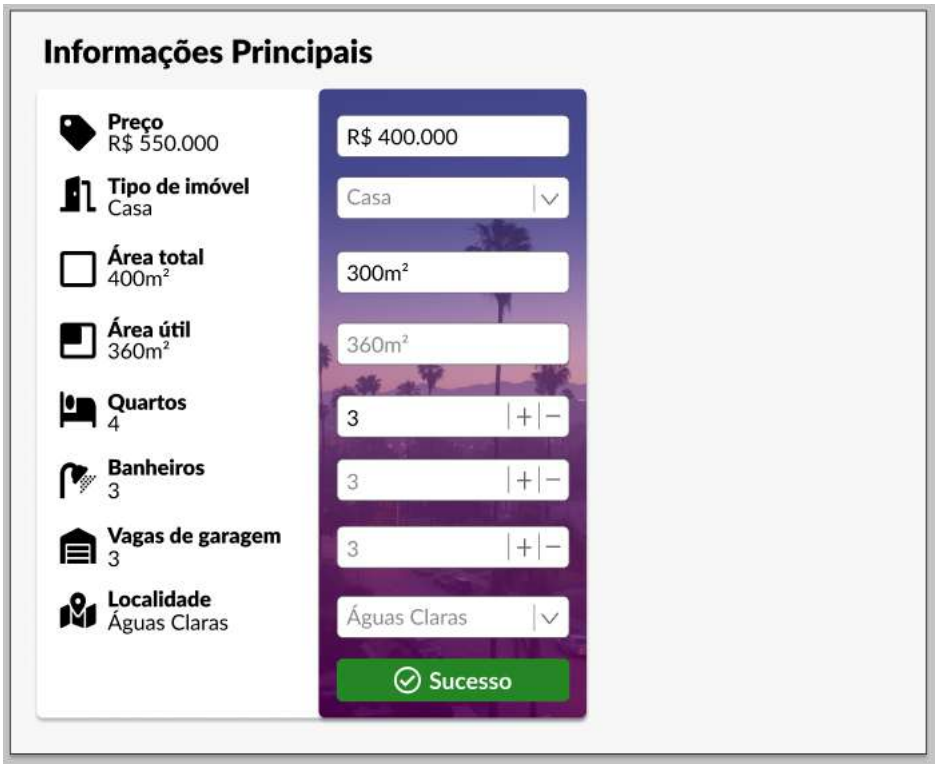

Figura 26 – Detalhes do imóvel com menu de crítica parte 5.

O usuário poderá criticar uma propriedade escolhendo entre manter e mudar os valores para cada característica. Após as críticas serem atribuídas as características de uma propriedade, o perfil de busca desse usuário será aprimorado, gerando novas recomendações no *feed* da página principal.

Ainda referente à página de detalhes foi adicionado também um novo componente na lateral direita para a visualização das recomendações do módulo de aprendizagem de máquina. Esse componente será a representação de cinco propriedades contendo todas suas características principais. Esse componente está representado na Figura 27.

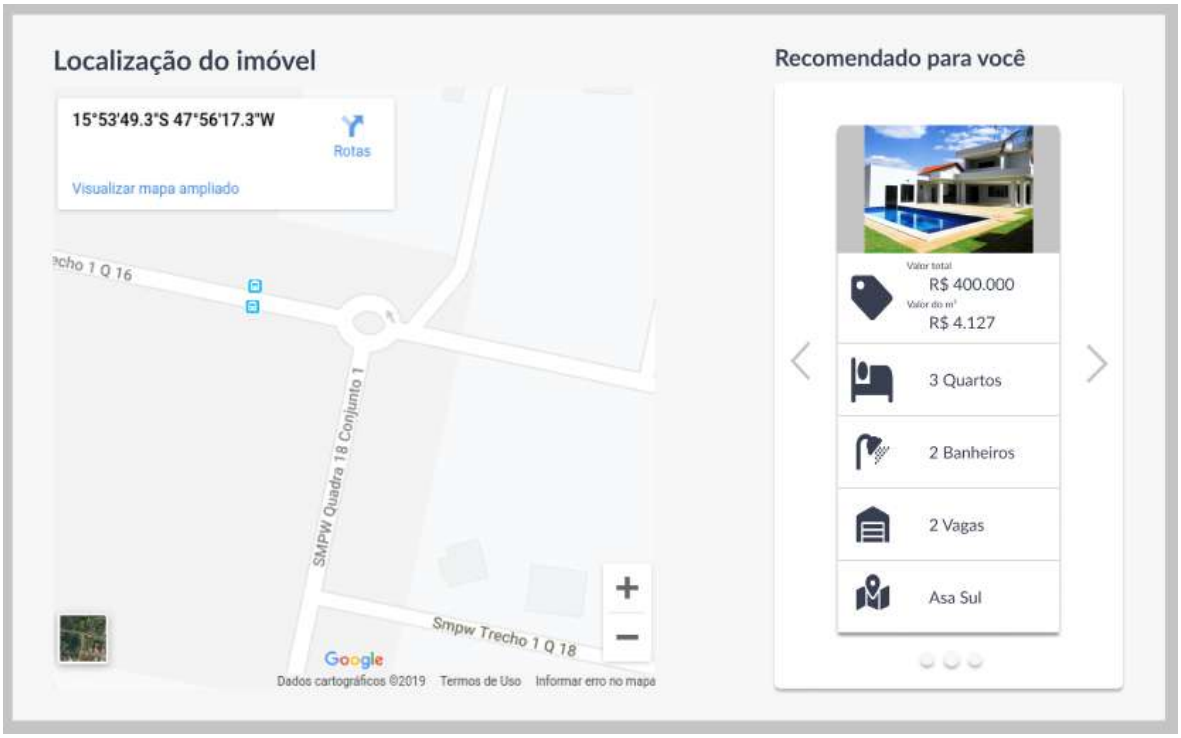

Figura 27 – Protótipo do carrossel de recomendações.

O recomendador do módulo de aprendizado de máquina tem como dados de entrada registros de ações de usuários na plataforma, como: cliques ou descartes em propriedades no *feed*, alterações no perfil de busca e até mesmo em requisições de contato com o cliente. Dessa forma, as recomendações são realizadas em tempo real, no momento em que um usuário entra na página de detalhes de uma propriedade.

3.3.1 Arquitetura

Foi desenvolvido uma aplicação web como um serviço de sistema de recomendação para o sítio virtual da Liva.vc. Sua principal funcionalidade será fornecer recomendações de propriedades para usuários que sejam clientes de imobiliárias vinculadas a Liva.

Para apresentar a arquitetura do projeto houve uma divisão em três níveis, em que são apresentadas as características e os motivos de suas implementações. Começando com o primeiro nível uma visão mais generalizada do sistema é apresentada. Em seguida, serão apresentadas características mais detalhadas nos seguintes níveis, a fim de melhor esclarecer o projeto proposto que se deseja implementar.

3.3.1.1 Nível 1

Considerando o sistema de recomendação como uma caixa preta, a Figura 28 demonstra a interação da Liva com o novo sistema de recomendação, descrevendo suas entradas e saídas de dados.

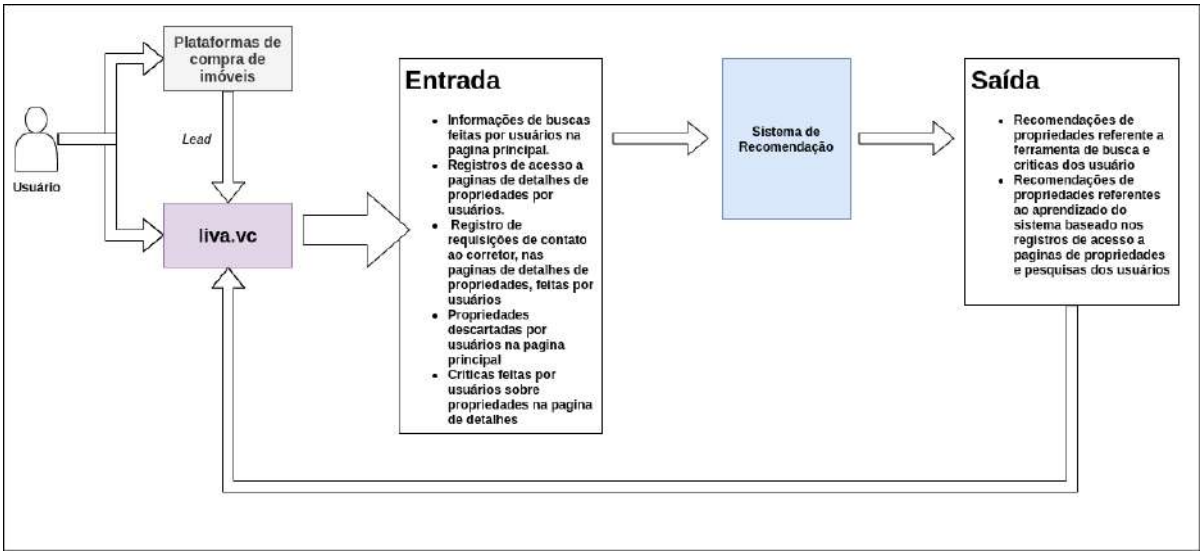

Figura 28 – Sistema de recomendação como caixa preta.

Para os usuários terem acesso ao ambiente virtual do Liva eles devem, primeiramente, ter interagido com uma outra plataforma de vendas de imóveis tais como Zap Imóveis ou Olx. Essa interação refere-se a um *lead*, que é uma requisição de informações a um corretor a respeito de um imóvel de uma imobiliária vinculada a Liva. Quando usuários acessam a Liva pela primeira vez eles já possuem diversos dados advindos de seu primeiro *lead*. Com isso, logo no primeiro acesso à página principal o usuário já tem os dados de entrada necessários para receber recomendações. A partir desse momento dados de navegação relevantes começam a ser registrados, tais como: trocar seu Perfil de Busca, clicar para visualizar propriedades, descartar propriedades, requisitar contato com o corretor (agendar visita ou pedir informações da propriedade) e criticar características das propriedades. Todo novo registro é enviado ao sistema de recomendação, que por sua vez, apresenta novas recomendações. Dessa forma, a cada novo registro as recomendações são reprocessadas e tornam-se eventualmente mais enriquecidas para fornecer outras recomendações ao usuário.

Normalmente, um usuário consulta diversas vezes e lê várias páginas antes de encontrar a propriedade certa. Esse processo é considerado muito trabalhoso e pode gerar frustrações aos usuários a ponto de fazê-los desistirem. O foco é melhorar a experiência do usuário no sistema durante esse processo, gerando recomendações ainda mais precisas a cada nova interação.

3.3.1.2 Nível 2

Diminuindo ainda mais o nível de abstração da arquitetura, a Figura 29 apresenta os serviços, suas tecnologias e interações do projeto.

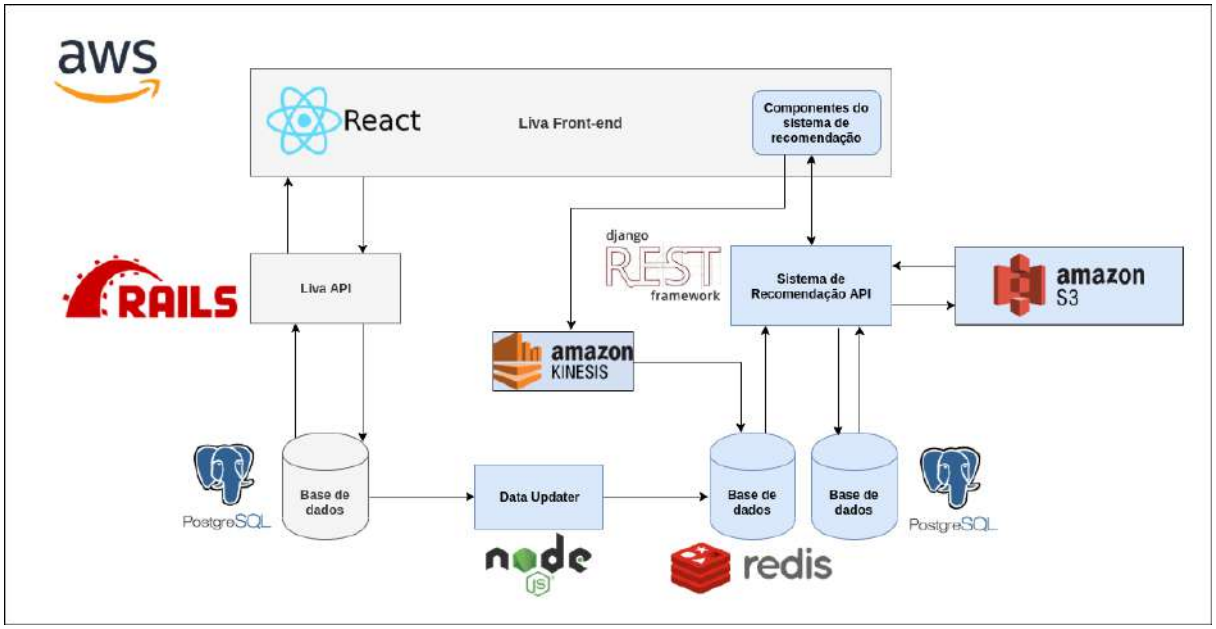

Figura 29 – Arquitetura geral do sistema divida em serviços.

A arquitetura existente no sistema da Liva segue o estilo Cliente-Servidor. Seu cliente é desenvolvido na linguagem JavaScript e com o *framework* React. Em seu *back-end*, Liva API (Application Programming Interface - Interface de Programação de Aplicação) está na linguagem Ruby com suporte do *framework* Ruby on Rails, é integrado a um sistema gerenciador de banco de dados PostgreSQL.

A fim de desenvolver um sistema como um serviço que possa ser utilizado pela Liva, sem interferir diretamente em sua estrutura, foi implantado uma arquitetura em que o sistema de recomendação a ser implementado está separado do restante do sistema, como demonstrado na Figura 29, destacado em azul. O sistema é composto por um cliente que é representado pelos novos componentes de interface atribuídos ao *front-end* da Liva. Esses componentes são: nova ferramenta de busca com o componente de importância para cada atributo (Figura 20), componente de crítica (Figura 22) e por fim o componente de recomendação, responsável por listar recomendações advindas do módulo de aprendizado de máquina (Figura 27). Eles se comunicarão através do protocolo HTTP (Protocolo de Transferência de Hipertexto) com a API (Interface de Programação de Aplicações) de recomendação e com a tecnologia Amazon Kinesis, que transmitirá dados para o Redis em tempo real.

A API de recomendação foi desenvolvida em Python com o *framework* Django REST e se comunicará com outras três tecnologias: Redis, PostgreSQL e Amazon S3.

A escolha dessa tecnologia foi feita pelo fato de que existem diversas bibliotecas e *frameworks* desenvolvidos em Python que dão suporte para suprir todos os requisitos do sistema a ser desenvolvido, como por exemplo, o XGboost, Scikit-Learn, Numpy, Pandas e o Pickle para solução do módulo de aprendizado de máquina, além do Django REST

que possibilita o desenvolvimento de uma API REST.

Como foram desenvolvidos novos componentes na interface gráfica referentes ao sistema de recomendação, eles foram desenvolvidos na linguagem JavaScript com o suporte do *framework* React já utilizado no ambiente da Liva, consequentemente, sendo facilmente integrável.

O sistema proposto é composto por um banco de dados de grande escala para o armazenamento de registros dos usuários e propriedades em memória RAM (Redis) para otimização no tempo de respostas de requisições a respeito da recomendação. Além disso, há a existência de um banco relacional para *backup* (cópia de segurança) dos dados e para salvar novos dados de treinos do modelo de aprendizado de máquina (PostgreSQL). O Redis foi escolhido para esse propósito, por ser rápido e por ser um banco de dados de arquitetura valor-chave, o que facilita no mapeamento dos dados. Essas escolhas são essenciais para o sistema de recomendação, pois as recomendações das propriedades precisam ser geradas rapidamente para o usuário. Segundo Li et al. (2017) na plataforma Suumo.jp eles calculam de dez a vinte segundos para um usuário ler as informações da propriedade, sendo somente em seguida visualizada as recomendações na página de detalhes.

O Amazon S3 servirá para guardar os dados de treino e o modelo treinado do módulo de aprendizagem de máquina. Essa tecnologia é utilizada para melhorar no tempo de treino do modelo, que será feito em lotes, e a fácil distribuição do modelo.

O sistema também é composto por um serviço de *upload* (modificação) de dados (*Data Updater*), que será desenvolvido na linguagem Javascript com o ambiente de execução NodeJS, capaz de ler o banco de dados da Liva e atualizar o banco do sistema de recomendação. Esses dados são basicamente referentes a propriedades.

O sistema de recomendação e todos os serviços que o envolvem foram implantados em ambiente de produção, por meio de contêineres, possibilitado pela tecnologia Docker e serão orquestrados pelo serviço do AWS, Amazon ECS, já utilizados na Liva.

3.3.1.3 Nível 3

3.3.2 Sistema de recomendação baseado em Aprendizado de Máquina

A escolha da abordagem de aprendizado de máquina justifica-se pela experiência descrita a seguir, feita na plataforma japonesa Suumo.jp (LI et al., 2017), que assim como a Liva, categoriza-se como um ambiente virtual de *e-commerce* imobiliário.

Li et al. (2017) demonstram que para seu contexto os registros de consulta dos usuários como entrada apresentam melhor performance no modelo de aprendizado de máquina quando comparado com outras abordagens. Para fazer essa comparação foi utilizada uma métrica chamada *Conversion Rate* (CVR - Taxa de Conversão). CV ou *Conversion*

(Conversão) refere-se a requisição de informação para um corretor de uma propriedade feita por um usuário. O CVR se dá pela quantidade requisições feitas em propriedades recomendadas (A) dividido por todas as recomendações clicadas (B), como apresentado na equação (3.1).

$$CVR = \frac{A}{B} \tag{3.1}$$

No sítio virtual Suumo.jp, pioneira no uso de sistema de recomendações aplicado ao contexto de imóveis, foi primeiramente desenvolvido um modelo híbrido de filtragem colaborativa com filtragem baseada em conteúdo, descritos na seção 2.2.3.1 e 2.2.3.2, respectivamente. Os resultados apresentaram uma melhora de 25% na taxa de cliques em propriedades em geral. Em seguida, foi implementado um modelo incremental de filtragem colaborativa restrito a algumas páginas para fazer uma comparação com o primeiro modelo. O segundo modelo produziu uma melhoria de 20% no CVR. O problema desse tipo de abordagem é que, como já apresentado anteriormente (seção 2.2.3.1), para que um item seja recomendado é preciso que muitas interações de usuários sejam feitas, ou seja, muitas visualizações, o que faz com que novas propriedades tenham menos chance de serem recomendadas. O recomendador baseado em aprendizado de máquina proposto ao final apresentou uma melhora de 250% na performance. Esse resultado demonstra que mesmo as ações mais sutis podem refletir nas preferências de um usuário e têm um efeito positivo na previsão de suas necessidades.

3.3.2.1 *Design* do Modelo

Com a finalidade de demonstrar a arquitetura do módulo de aprendizado de máquina e suas principais características, a Figura 30 apresenta o processo de recomendação mais detalhado dentro da API e as interações com as outras tecnologias.

Baseado na arquitetura da Figura 30, primeiro é passado, via requisição, o identificador único (chave primária) do usuário para o *endpoint* (endereço de rede para acesso do cliente) de recomendação, no momento em que um usuário entra na página de detalhes de uma propriedade, a fim de serem geradas as recomendações. Com o ID do usuário é possível recuperar os seguintes três tipos de variáveis:

- **Última busca realizada pelo usuário (BU):** refere-se ao atual Perfil de Busca do usuário. Quando um usuário clica para visualizar uma propriedade, ela está sobre diversas condições de busca, como preço mínimo e máximo, tipo de imóvel, quantidade de quartos, dentre outras condições. Essas informações são guardadas na tabela LAST_SEARCH_PROFILE_HISTORY (Último Histórico do Perfil de Pesquisa);

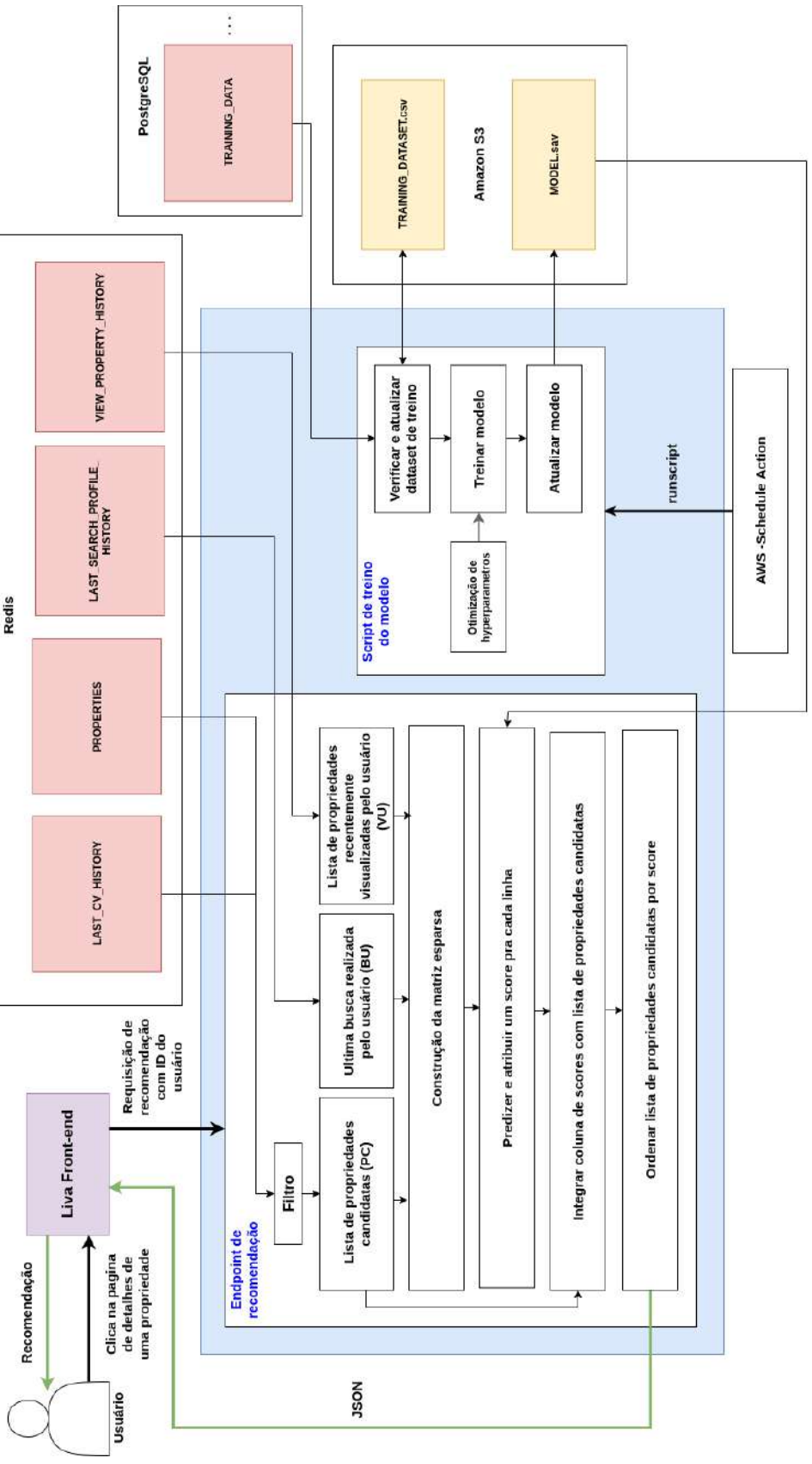

Figura 30 – Arquitetura do módulo de Aprendizado de Máquina.

- **Visualizações recentes em propriedades feitas pelo usuário (VU):** quando um usuário clica para visualizar os detalhes de uma propriedade. A cada clique é salvo um novo registro na tabela VIEW_PROPERTY_HISTORY (Visão do Histórico de Propriedade). São salvos no máximo 100 registros dos últimos 3 meses para cada usuário. Esses registros são compostos de informações como preço, área, quantidade de quartos, dentre outras características de um imóvel;
- **Propriedade candidata a recomendação (PC):** refere-se às características da propriedade candidata a recomendação como preço, área, quantidade de quartos, dentre outras características. As propriedades são mantidas na tabela PROPERTIES (propriedades).

Para a entrada do modelo de aprendizado de máquina existe um conjunto de BU, VU, PC e a saída é a pontuação (*score*) que reflete a probabilidade de CV, ou seja, a probabilidade de um usuário requisitar contato a um corretor de uma propriedade em específico, como apresentado na equação (3.2).

$$\{BU, VU, PC\} \xrightarrow{F} \mathbb{R} \tag{3.2}$$

Em que F é o modelo que atribui um *score* para um registro composto por BU, VU e PC. Um maior *score* significa maior chance de um CV. O conjunto de dados de treino é composto por BU, VU e as características da propriedade de cada CV realizado ou descarte de uma propriedade do *feed* referente a todos os usuários, ou seja, a variável alvo nesse caso é o CV, podendo ser 0 ou 1. O algoritmo de aprendizagem de máquina vai construir um modelo capaz de distinguir entre CV = 1 e CV = 0 . No caso de um usuário descartar uma propriedade é considerado CV = 0.

A cada nova interação de descarte ou requisição de informação de uma propriedade os dados BU, VU e as características da propriedade em foco são salvos na tabela NEW_TRAINING_DATA (Novos Dados de Treinamento). O treino é realizado por lotes agendados diariamente. Toda vez que o *script* de treino é executado, primeiramente, é verificado a existência de um arquivo .csv na Amazon S3 e em seguida é feita uma integração com os dados de treino da tabela. Logo em seguida é realizada a otimização de hiper-parâmetros com a técnica *Grid Search* (pesquisa em grade). O modelo é treinado e salvo como arquivo .sav na Amazon S3.

Toda vez que um usuário alcança a página de detalhes de uma propriedade é recuperado seu BU e VU (Tabela 3). Dessa forma, é apurado o *score* para cada candidato PC. Os candidatos são escolhidos baseados na mesma localidade da presente propriedade visualizada pelo usuário. Após realizado o *score* é feita uma ordenação dessas propriedades e em seguida são apresentadas as cinco primeiras ao usuário.

Tabela 3 – Exemplo de conjunto de dados de treino.

Preço máximo (BU)	**...**	**Quantidade de quartos 1 (VU)**	**Preço 2 (VU)**	**Quantidade de banheiro 2 (VU)**	**...**	**CV**
400000	...	1	150000	2	...	1
100000	...	3	null	null	...	0

Para o presente problema de classificação foi escolhido o modelo de GBDT (*Gradient Boosting Decision Tree* - Árvore de Decisão com Aumento de Gradiente), discutido na seção 2.3.1.2. Como a característica do conjunto de dados torna possível a existência de colunas com dados ausentes, é muito importante a utilização de um método capaz de lidar com tal problema bem, além de manter uma alta precisão. Por ser um modelo baseado em árvore de decisão, ele consegue lidar com atributos em seu valor bruto, o que é preferível em questões de manutenibilidade e performance, pois não necessitará de pré-processamento. Além disso, a tecnologia usada para o modelo lida naturalmente com dados faltantes. Segundo Synced (2017), internamente, o XGBoost aprenderá automaticamente qual é a melhor direção a seguir quando um valor estiver faltando. Equivalentemente, isso pode ser visto como "aprender" automaticamente qual é o melhor valor de imputação para valores faltantes com base na minimização da perda do treinamento.

Os três tipos de variáveis apresentadas anteriormente para entrada do modelo são relacionadas a dois tipos matemáticos, numéricos e categóricos. Para os dados numéricos como preço ou quantidade de quartos, por exemplo, foi decidido manter seus valores em sua forma bruta, por questões de manutenibilidade e performance. Para as variáveis categóricas, algumas colunas são excluídas, como o identificador único do usuário, que não é útil para a predição. A localidade será trocada pela sua latitude e longitude e o tipo de imóvel será trocado por variáveis fictícias binárias (0 e 1).

No Suumo.jp, Li et al. (2017) encontra uma distribuição de tuplas no conjunto de dados de treino entre CV = 1 e CV = 0, para cada uma as linhas com CV = 1 são geradas dez linhas com CV = 0. Em seu modelo são propostas dez linhas CV = 0 para dez propriedades aleatórias a cada um CV = 1. Dado que no modelo proposto para esse projeto se tem a interação de descarte de uma propriedade, foi decidido não utilizar essa proporção e aleatoriedade, pois isso fere com a realidade dos dados dentro do contexto.

Li et al. (2017) afirma que as variáveis categóricas devem ser tratadas em um modelo diferente das variáveis numéricas. Isso porque, como descrito anteriormente na seção 2.3, na construção da árvore de decisão, o algoritmo de ramificação se concentra na divisão de variáveis numéricas, a frequência de aparecimento de variáveis categóricas igualmente importantes para os usuários é relativamente reduzida, ou seja, as árvores de decisão vão favorecer as ramificações com variáveis numéricas.

Dito isso Li et al. (2017) propõe a separação de dois modelos, um somente com variáveis categóricas e o outro com numéricas. Para fazer a junção dos *scores* (pontuações) gerados ele propõe a utilização de uma função de soma linear.

Como há uma quantidade muito pequena de características categóricas para o modelo proposto, não será feito a partição entre categóricas e numéricas.

3.3.3 Sistema de recomendação baseado em crítica

Com o intuito de não alterar o que já foi proposto no site da Liva, que faz o uso de uma ferramenta de busca baseada em preferência (Perfil de Busca) para recomendar propriedades, e ainda deixar o sistema de recomendação mais robusto seguindo uma abordagem, decidiu-se construir um sistema de recomendação baseado em crítica como descrito na seção 2.2.3.5 seguindo a abordagem de crítica de exemplo.

Segundo Burke, Felfernig e Goker (2011), e como já dito anteriormente (seções 2.2.3.1 e 2.2.3.2), as abordagens tradicionais de recomendação, filtragem colaborativa e baseada em conteúdo, não são adequadas em situações de itens de alto valor, como propriedades ou veículos, que não são comprados com frequência. A abordagem baseada em conhecimento pode mitigar esse problema explorando requisitos explícitos do usuário, mas ela sofre de gargalos de aquisição de conhecimento associados aos esforços iniciais necessários para gerar o conhecimento do domínio.

Dessa forma, a abordagem baseada em crítica surgiu para ser uma eficiente tecnologia de recomendação baseada na preferência do usuário e evitar os problemas descritos anteriormente.

3.3.3.1 Modelo da preferência

Para o presente projeto decidiu-se por realizar a implementação da abordagem de crítica de exemplo, melhorando a ferramenta de busca baseada no ambiente virtual da Liva. A ferramenta de busca baseada em preferências é utilizada para obter as preferências iniciais dos usuários, enquanto a crítica de exemplo é uma abordagem que permite aos usuários aprimorar suas preferências para localizar o item ideal que atenda às suas necessidades.

O usuário segue um processo como demonstrado na Figura 31:

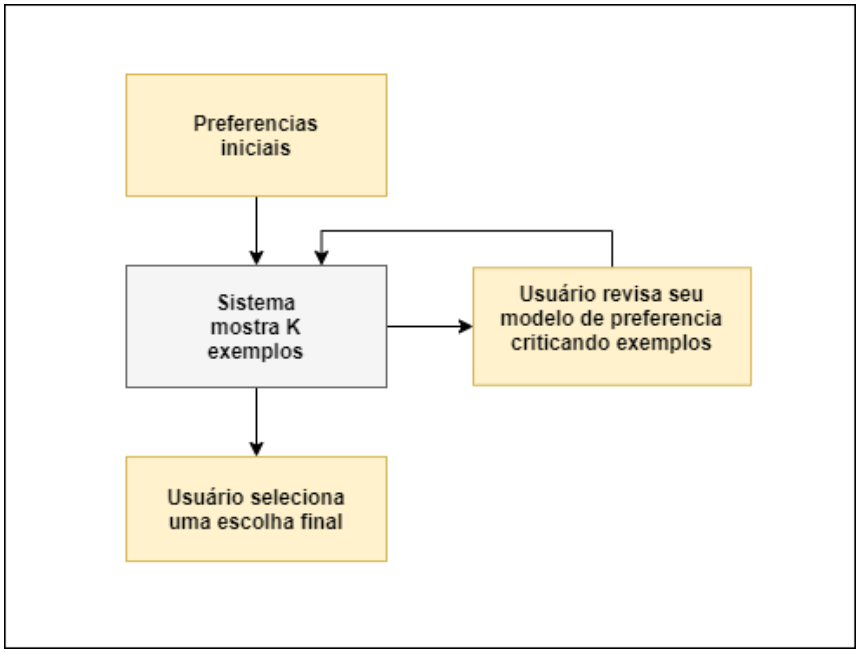

Figura 31 – Interação do usuário com a abordagem de crítica de exemplo. Fonte: (VIAPPIANI; FALTINGS; PU, 2006). Tradução livre pelos autores.

Após o usuário ter definido seu modelo de preferência inicial o sistema é capaz de apresentar exemplos que o usuário possa considerar. Esses exemplos são propriedades que são ótimas para a atual busca de preferência (Perfil de Busca) configurada. O usuário irá revisar seu modelo de preferência através de críticas sobre os exemplos apresentados. Quando o usuário identifica o item certo o processo termina.

O usuário irá apresentar suas preferências através do Perfil de Busca como na Figura 20. São comportados os seguintes atributos das propriedades: tipo de imóvel, faixa de preço, área, localização (Cidade e Bairro), quantidade de quartos, quantidade de banheiros e quantidade de vagas na garagem. Além disso, usuário seleciona a importância de cada atributo (peso de 1 a 5). O usuário irá criticar as propriedades de acordo com a Figura 22, em que ele pode postar críticas para a solução próxima, com as escolhas de manter ou melhorar algum atributo.

Como já demonstrado na seção 2.2.3.5, o modelo de preferência será baseado no MAUT (teoria da utilidade de atributos múltiplos), em que será computada a utilidade para cada propriedade e assim serem ordenadas e apresentadas.

Segundo Viappiani et al. (2006) na abordagem de crítica de exemplo, cada crítica pode ser considerada como uma restrição leve e o modelo de preferência pode ser desenvolvido simplesmente coletando críticas incrementalmente. Uma restrição leve é uma função de um atributo ou uma combinação de atributos para um número que indica o grau em que a restrição é violada.

Viappiani et al. (2006) dá o seguinte exemplo: um atributo que pode assumir os

valores **a**, **b** e **c**, pode, para uma restrição leve, indicar preferência pelo valor b. Nesse caso haveria o mapeamento dos valores **a** e **c** para 1 e **b** para 0, indicando que somente **b** não quebrou a restrição. Uma preferência pela área de superfície de pelo menos 30 metros quadrados, em que uma pequena violação de até 5 metros quadrados pode ser aceitável, pode ser expressa por uma função linear por partes, como apresentado na equação (3.3).

$$f(n) = \begin{cases} 1 & \text{if } x < 25 \\ 0.2(30 - x) & \text{if } 25 \leq x \leq 30 \\ 0 & \text{if } x > 30 \end{cases} \tag{3.3}$$

O autor considera que as restrições leves permitem que os usuários expressem preferências relativamente complexas de maneira intuitiva. Isso torna as restrições leves uma forma útil para modelos de preferência de um *example-critiquing* (crítica de exemplo).

3.3.4 Banco de dados

Como já descrito anteriormente, foram utilizados dois tipos de banco de dados, um relacional com a tecnologia PostgreSQL e outro de valor-chave com Redis. Esses dois bancos são somente referentes ao módulo de aprendizado de máquina, pois o módulo de crítica não necessita de armazenar dados.

Na base de dados relacional foram armazenados dados para backup como o histórico de visualizações dos usuários (VIEW PROPERTY HISTORY - Histórico de propriedades visualizadas), histórico de buscas realizadas pelo usuário (SEARCH PROFILE HISTORY - Histórico de perfis de busca) e as propriedades (PROPERTIES). Além disso, são guardados dados de treino do modelo de aprendizado de máquina (TRAINING DATA). Nas Figuras 32 e 33 são apresentados os diagramas do banco de dados relacional.

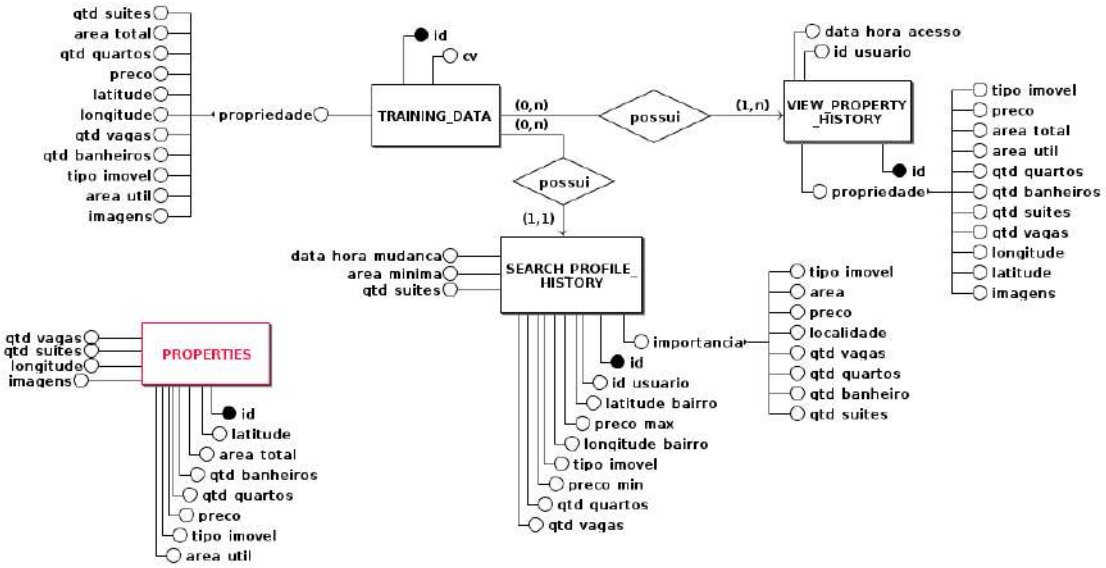

Figura 32 – Diagrama de Entidade-Relacionamento (DE-R).

Figura 33 – Diagrama Lógico de Dados (DLD).

Inicialmente, a tabela SEARCH_PROFILE_HISTORY será populada com base nos perfis de buscas atuais dos usuários, adquiridos na base da dados da Liva. A tabela de propriedades em vermelho no diagrama da Figura 32 está destacada dessa forma porque ela não será armazenada no banco do sistema de recomendação, mas como já existente, na base da Liva. No banco de dados da Liva as propriedades são atualizadas diariamente

para cada imobiliária existente. Como os dados vindos de cada imobiliária não seguem um padrão e muitas vezes há a existência de muitas características ausentes, foi decidido focar somente em características comuns e raramente faltantes como: tipo do imóvel, área total e útil, preço, localidade, quantidade de quartos, banheiros e suítes.

A tabela referente ao modelo de aprendizado de máquina será populada com dados que serão coletados do início do primeiro semestre de 2020 até a conclusão do sistema em junho, com o fim de se gerar um conjunto de dados (*dataset*) de treino inicial.

Diante a pandemia que se iniciou em março de 2020 causando o cancelamento do primeiro semestre, adiando o calendário acadêmico da UnB, foi alterada a estratégia para a população da tabela referente ao modelo e aprendizado de máquina descrito acima, que se encontra especificado na seção (4.2).

Em relação ao banco de dados no Redis, como já apresentado na arquitetura anteriormente, existem quatro entidades como mostrado na Figura 34.

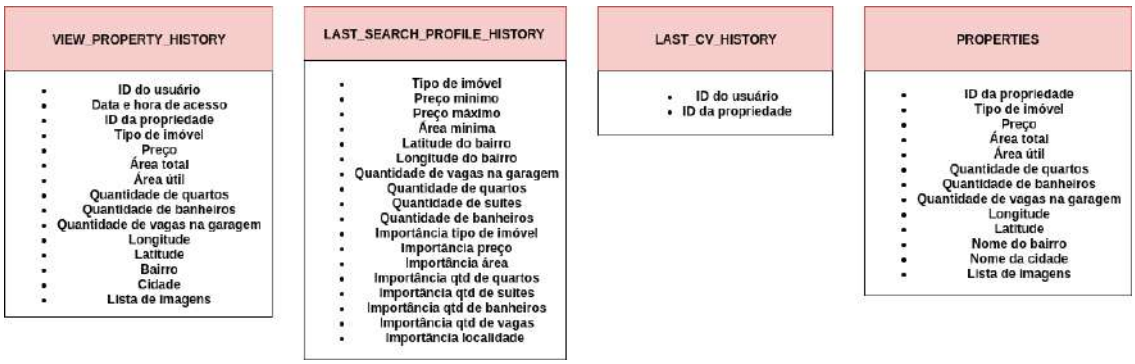

Figura 34 – Representação do banco de dados no Redis.

Os dados da entidade PROPRIEDADES serão atualizados constantemente pelo serviço descrito na arquitetura, *Data Updater*. O serviço irá atualizar os dados a partir da leitura da tabela PROPERTIES concentrada no banco de dados da Liva.

3.4 Suporte Tecnológico

3.4.1 Python

Python é uma linguagem de programação de alto nível com o paradigma OO (Orientado a Objeto), sendo ela uma linguagem interpretada e não compilada, como por exemplo, a linguagem C. Conhecida também por ter uma sintaxe muito clara e exigir poucas linhas de código comparadas a outras linguagens de programação a linguagem Python é muito utilizada em vários serviços de grandes empresas, sendo uma linguagem formada por uma reconhecia comunidade (PYTHON, 2019).

A versão que será implementada no serviço "Sistema de Recomendação API" é a 3.7.3.

3.4.2 Django

O Django é um *framework* sob a licença BSD, criado para a linguagem Python que usa o padrão MVT (*Model-View-Template*), na qual é uma adaptação do clássico padrão MVC (*Model-View-Controller*). Essa tecnologia visa o desenvolvimento rápido, pois é baseada no princípio DRY (*Don't Repeat Yourself*), em que a ideia principal é evitar ao máximo o retrabalho no desenvolvimento de software (DJANGO, 2019).

Ele também já cuida de várias rotinas comuns de aplicações *Web*, como por exemplo: servidor, conexão com o banco de dados, organização de rotas (*endpoints*), autenticação entre outras rotinas. Contém também a tecnologia ORM (*Object-relational mapping*), que tem o intuito de mapear cada atributo e relacionamento das classes para o banco de dados, ou seja, não é necessário criar *scripts* SQL (*Structured Query Language*) na mão para tal.

A versão que será usada no projeto para o serviço "Sistema de Recomendação API" é a 2.2.

3.4.3 Django REST *Framework*

Para construção da API (Interface de Programação de Aplicações) com o paradigma arquitetural REST será utilizado a ferramenta denominada Django REST *framework* que disponibiliza diversas ferramentas para construção da mesma, sendo que ela ainda possibilita algumas funcionalidades extras, como o *serializer*, que permite a conversão de tipos de dado como JSON e XML para os tipos de dados nativos do python, e vice e versa (REST, 2019).

A respeito dessa ferramenta, ela usa o padrão MV (*Model-View*), pois sendo uma API, não há o *Template* que teria o papel de um *front-end*, como no Django. Essa tecnologia também possui uma grande comunidade, excelente documentação para consulta e é altamente utilizada por grandes empresas, como: Mozilla , Red Hat , Heroku e Eventbrite (REST, 2019).

A versão que será usada para implementação do serviço "Sistema de Recomendação API" será a 3.10.3.

3.4.4 Scikit-Learn

O Scikit-Learn é um módulo do python que possui vários algoritmos de aprendizagem de máquina (por exemplo *random forest e gradient boosting*) para problemas

supervisionados e não supervisionados. Está biblioteca é construída em Numpy e Scipy e tem como objetivo trazer conhecimento de aprendizado de máquina para os usuários não especialistas usando uma linguagem de alto nível (Python). Um dos destaques dessa tecnologia está na facilidade de uso, no desempenho e na documentação (PREDEGOSA, 2011).

A versão que será usada para implementação será a 0.21.3.

3.4.5 Pandas

Pandas é uma biblioteca do Python de código aberto com a licença BSD que preza em um bom desempenho e facilidade de uso. O propósito deste módulo é realizar análise e modelagem de dados (bem parecido com a linguagem R) usando o objeto conhecido como "*DataFrame*" (Quadro de dados), podendo ler e escrever dados no formato CSV (valores separados por vírgula), texto, SQL (*Structured Query Language* - Linguagem de consulta estruturada), HDF5 e Excel (PANDAS, 2019).

A versão que será usada para implementação será a 0.3.1.

3.4.6 Numpy

Numpy é um pacote fundamental do python com a licença BSD (Berkeley *Software Distribution*) quando se trata de Ciência de Dados que adiciona um poderoso objeto com matriz N-dimensional, podendo integrar com código C, C++ e Fortran, além de possuir recursos importantes de álgebra linear (transformação de Fourier, números aleatórios, por exemplo) (NUMPY, 2019).

A versão que será usada para implementação será a 1.17.3.

3.4.7 Pickle

O Pickle é um módulo do python que implementa protocolos binários com a finalidade de serializar e desserializar uma estrutura de objetos dessa mesma linguagem. O "*Pickling*" significa transformar um objeto da hierarquia do python para fluxo em bytes, já o desserializar é o fluxo contrário a isso (PICKLE, 2019).

A versão que será usada para implementação será a 5.

3.4.8 Xgboost

Xgboost é uma biblioteca de licença Apache que proporciona o *gradient boosting* com suporte para C++, Python, R e outras linguagens de programação. Criada para ser altamente eficiente, flexível e portátil, resolvendo vários problemas de Ciência de Da-

dos utilizando algoritmos de aprendizagem de máquina sob a estrutura *gradient boosting* (XGBOOST, 2019).

A versão que será utilizada no serviço "Sistema de Recomendação API" será a 0.90.

3.4.9 JavaScript

JavaScript é uma linguagem de programação interpretada (assim como o python) muito utilizada no lado do cliente, mas também podendo ser usado do lado do servidor para tratamento de dados. Sendo a linguagem mais usada atualmente no lado do cliente, ou seja, no navegador do usuário, juntamente com HTML5 e CSS3, deixando as páginas Web estáticas, agora interativas. Com o avanço da linguagem e a criação do *framework* NodeJS, atualmente JavaScript é muito usado no *back-end* dos serviços (JAVASCRIPT, 2019).

A versão que será utilizada para a implementação dos serviços "Data Updater" será a EcmaScript 6.

3.4.10 NodeJS

O NodeJS é definido como um "ambiente de execução Javascript do lado do servidor", ou seja, é um *framework* orientado a eventos baseado na linguagem Javascript que possibilita a criação de aplicações não dependentes de um navegador. Um ponto positivo que levou a sua alta adoção pelos programadores é a sua alta escalabilidade, além de sua arquitetura flexível e sendo um ótimo aliado para a arquitetura de microsserviços (NODEJS, 2019).

Um dos diferenciais do NodeJS é o gerenciador de pacotes que ele tem, o NPM, que contém o maior repositório de software do mundo, sendo que um dos pacotes é o Express, na qual é um *framework* para desenvolvimento *Web* (LENON, 2018).

A versão que será utilizada para a implementação do serviço "Data Updater" será a 13.1.0.

3.4.11 ReactJS

ReactJS é um biblioteca Javascript orientada a componentes que tem como objetivo criar interfaces interativas de usuário sem muita complexidade. Cada componente tem seu próprio estado chamado de "*state*" e a cada mudança desse estado, o React atualiza e renderiza quase instantaneamente aquele componente. Essa tecnologia é compatível com a sintaxe JSX, mas sendo opcional e não obrigatório seu uso pelo programador (REACTJS, 2019).

A versão já utilizada no serviço de *front-end* da Liva e que também será utilizada para implementação dos novos componentes do sistema de recomendação será a 16.11.

3.4.12 Bootstrap

Bootstrap é uma ferramenta de código livre para desenvolvimento de sites e aplicativos web que tem HTML, CSS e JavaScript. Sendo um dos *frameworks* mais famosos do mundo usado para criar ambientes virtuais responsivos e móveis (*mobile*), disponibilizando componentes para web (Internet) com estilos prontos e *templates* de *sites* (BOOTSTRAP, 2019).

A versão já usada no serviço de *front-end* da Liva e que também será utilizada para implementação dos novos componentes do sistema de recomendação será a 4.3.

3.4.13 Ruby

Ruby é uma linguagem de programação de código totalmente livre orientada a objeto, no qual que tudo em ruby é objeto, pois cada parte do código tem suas propriedades e ações relacionadas a aquele tipo de objeto. O Ruby também é flexível, ou seja, os utilizadores podem remover e redefinir partes da linguagem a vontade. Além disso, o Ruby usa o padrão "*sugar syntax*" (açúcar sintático) que facilita lembrar de como escrever o código (RUBY, 2019).

A versão já usada no serviço *back-end* "Liva API" é a 2.6.5.

3.4.14 Rails

Rails é um meta-framework (um framework compostos de outros frameworks) feito com a linguagem Ruby e também é de código aberto sob a licença MIT, assim como a própria linguagem. O padrão arquitetural seguido por essa tecnologia é o clássico MVC (*Model-View-Controller*), e também os padrões Web, como JSON e XML. A ideia principal do Rails é aumentar a velocidade de desenvolver uma aplicação Web (PORTALGSTI, 2019).

A versão já usada no serviço *back-end* "Liva API" é a 6.0.1.

3.4.15 PostgreSQL

PostgreSQL é um banco e dados relacional de código aberto que usa a linguagem de *script* SQL. Ele também contém vários recursos para a escala e a segurança das cargas de trabalho de dados. Esse banco ganhou sua reputação através da sua arquitetura robusta, confiabilidade, integridade, um conjunto de recursos e a dedicação de sua comunidade. Atualmente, ele é o banco relacional de código aberto preferido pelos desenvolvedores

(POSTEGRESQL, 2019). Fora essas características, o PostgreSQL é altamente extensível, no sentindo do desenvolvedor poder criar seus próprios tipos de dados e funções personalizadas (POSTEGRESQL, 2019).

A versão que já está sendo usada como banco de dados para o serviço "Liva API" será usada no banco de dados do sistema proposto que será á 12.0.

3.4.16 Redis

O Redis é uma base de dados em memória com licença BSD, ou seja, tem código aberto. Muito utilizado como cache e banco de dados em serviços, suportando vários tipos de dados, como: *strings, hashes, lists, sets, sorted sets* dentre outros. Ele permite realizar algumas operações atômicas com os dados, por exemplo, inserir em uma *string*, incrementar algum valor numa *hash*, inserir elementos numa *list* e outras operações (REDIS, 2019).

O responsável pelo seu excelente desempenho é o conjunto de dados na memória, mas caso seja necessário é capaz de gravar em disco ou anexar informações em *log* (REDIS, 2019).

A versão que será utilizada para se conectar com o serviço "*Data Updater*" e o "Sistema de Recomendação API" é a 5.0.

3.4.17 Docker

Docker é uma ferramenta de virtualização muito usada em vários serviços, na qual foi projetada para facilitar a criação e a implantação usando os chamados "contêineres" de modo que cada *container* pode ser feito um empacotamento de algum determinado serviço com todos os pacotes e configurações necessárias para a execução dele (OPENSOURCE, 2019).

O Docker disponibiliza uma grande quantidade de imagens no repositório chamado DockerHub, sejam elas oficiais ou feitas por desenvolvedores ao redor do mundo. Essas imagens contém Linux, possibilitando a base para criação da virtualização da aplicação (DOCKERHUB, 2019).

A versão que será usada para virtualização dos serviços "*Data Updater*" e "Sistema de Recomendação API" é a 19.03.

3.4.18 Amazon *Elastic Container Service*

O Amazon ECS é uma plataforma de gerenciamento de contêineres rápida e dimensível que possibilita o gerenciamento de contêineres dentro de um *cluster*. O usuário é capaz de iniciar ou parar aplicações que rodam nesses contêineres com um simples comando,

inclusive disponibiliza o estado de cada serviço rodando dentro do *cluster* (AMAZON, 2019b).

Ele dá a possibilidade do usuário escolher os recursos que deseja para rodar determinado serviço e sem se preocupar com as configurações de cada recurso ou com a escalabilidade da infraestrutura de gerenciamento (AMAZON, 2019b).

3.4.19 Amazon Kinesis

O Amazon Kinesis é um serviço que serve para coleta, análise e transmissão (*streaming*) de grandes quantidades de dados em tempo real. O Kinesis pode ser executado em instâncias do Amazon EC2 com a biblioteca Kinesis Client. Essa tecnologia possibilita enviar os dados processados para vários outros serviços AWS (KINESIS, 2019).

3.4.20 Amazon S3

O Amazon Simple Storage Service (Amazon S3) é um serviço de armazenamento de dados junto com escalabilidade, durabilidade, segurança e performance, ou seja, é possível armazenar qualquer volume de dados em uma grande escala de variabilidade. O gerenciamento e a organização dos dados é feito de maneira fácil (AMAZON, 2019a).

3.4.21 Git

Git é uma ferramenta de código aberto de controle de versão planejado para pequenos até grandes projetos com rapidez. O seu diferencial comparado com outras ferramentas de versionamento é a sua estrutura de *branches* (ramificações) e *mergings* (fundição), na qual é possível criar várias *branches* locais totalmente independentes uma da outra, além da criação e da exclusão ser fácil e rápida (GIT, 2019). O repositório de código escolhido será o GitHub.

A versão que será usada nos serviços "*Data Updater*" e "Sistema de Recomendação API" será a 2.24.

3.4.22 ZenHub

ZenHub é uma ferramenta de gerenciamento de projetos ágeis para desenvolvedores de software integrado a plataforma do GitHub. Usando as informações do projeto trazidas do próprio GitHub. Essa ferramenta é capaz de gerar gráficos, relatórios, fluxo de trabalho dos desenvolvedores e o andamento do projeto (ZENHUB, 2019).

4 Desenvolvimento

Este capítulo refere-se a execução da proposta apresentada no Capitulo 3, correspondendo ao fluxo de desenvolvimento indicado na seção 3.1.6 deste trabalho. Será abordado como foi o processo de desenvolvimento do sistema de recomendação proposto, assim como as dificuldades e alterações realizadas sobre a proposta inicial para sua maior integridade.

Seguindo a metodologia *Scrum*, foi realizado o *backlog* do produto inicial (encontrado no apêndice A), contendo as *stories* descritas em 3.1.3, juntamente com as *sprints* e as prioridades de cada *story*.

O capítulo foi dividido por *sprints* presentes no *backlog* do produto, sendo apresentado ao final os artefatos gerados e uma breve discussão. No desfecho do capítulo será exibido as análises gerais realizadas no trabalho.

4.1 1° *Sprint*

Na primeira *sprint*, foram criados novos componentes no *front-end* da Liva e feitas alterações nas páginas, inclusive correções de *bugs* para se adequarem ao novo sistema de recomendação que irá entrar em produção após sua conclusão. Essas alterações são tais como: design da modal de alteração do perfil de busca (Figura 35) e a adição do novo componente de apresentação de imoveis recomendados, além do sistema de comunicação implementado para transmissão de dados de ações dos clientes para API.

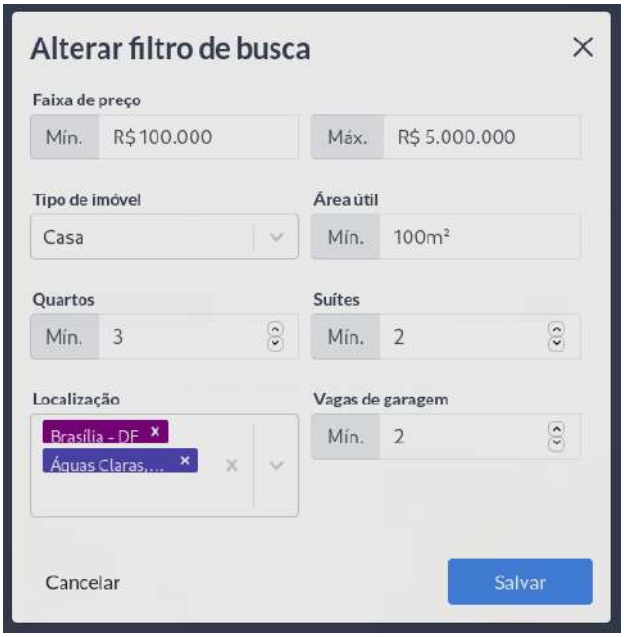

Figura 35 – Modal de alterar filtro de busca.

Também foi realizada a configuração da integração e *deploy* contínuo utilizando a ferramenta Actions do próprio Github, sendo nele realizado a execução dos testes unitários e a execução do *deploy* automático no AWS.

O componente de apresentação de imóveis que anteriormente tinha sido colocado na lateral da página de detalhes, atualmente foi inserido de forma a ocupar a largura toda do corpo da página como pode ser visto na Figura 36, pois dessa forma o usuário consegue ver várias recomendações ao mesmo tempo, sem precisar ficar clicando mais vezes para ver recomendações. Antes, ele tinha sido colocado na lateral da página de detalhes, ao lado do mapa de localização, por razão de menor ocupação do espaço da página, como foi apresentado na Figura 27.

No componente são apresentados 5 novas propriedades para o usuário, no qual é possível rolar os *cards* de imóveis com as setas horizontais. Em cada *card* é exibido as seguintes características: imagens da propriedade, preço, tipo de imóvel, cidade em que o imóvel é localizado, quantidade de quartos, quantidade de suítes, quantidade de área e quantidade de vagas de garagem.

Figura 36 – Componente de apresentação de imóveis recomendados.

Na implementação do registro das ações dos usuários foi notado que não era necessário o uso da ferramenta KINESIS do AWS, pois foi percebido que para transmissão de dados desse tipo não havia a necessidade de uma ferramenta tão poderosa. Essa ferramenta do AWS normalmente é utilizada para transmissão de vídeos e outros dados mais pesados, conseguindo capturar continuamente gigabytes de dados por segundo de centenas de milhares de origens (KINESIS, 2019).

Ao final da *sprint*, na reunião de *sprint review* foi decidido que era necessário criar uma *story* para substituir a tecnologia KINESIS da AWS.

4.1.1 *Sprint backlog* e resultados

O quadro 4.1 referente a primeira *sprint*, demonstra as *stories* concluídas, pendentes e as novas que surgiram na reunião de *sprint planning*. Para essa *sprint* surgiram as *stories* técnicas TS08 e TS06 pela necessidade da melhora do design dos componentes de perfil de busca e crítica. Também surgiram as *stories* TS12 E TS13, referentes a construção de um CI/CD, que são muito importantes para a manutenção contínua do software.

Stories planejadas	Concluídas	Pendentes	Novas
US01 - Visualizar perfil de busca	X		
US02 - Alterar perfil de busca	X		
TS01 - Corrigir bugs existentes na página de detalhes do imóvel	X		
TS02 - Corrigir bug de agendamento de visitas	X		
TS03 - Registrar ações do usuário no sítio virtual		X	
TS08 - Melhorar design do componente de recomendação	X		X
TS06 - Melhorar design da modal do perfil de busca	X		X
TS07 - Melhorar design do componente de crítica	X		X
TS12 - Configurar integração contínua	X		X
TS13 - Configurar deploy automático	X		X

Quadro 4.1 – *Sprint backlog* e resultados da *sprint* 1.

4.2 2º *Sprint*

Para a segunda *sprint* como descrito anteriormente, a *story* de registro de ações dos usuários permaneceu pelo fato de haver a necessidade da troca de tecnologia para as transmissões de dados. Foi realizado também, o início da coleta de dados para o módulo de aprendizado de máquina.

Web-sockets é um protocolo que ao contrário do HTTP permite uma comunicação bidirecional no cliente e no servidor. Diferentemente do *sockets*, ele é executado a partir de *browsers* que se conectam ao servidor de aplicativos por meio de um protocolo semelhante ao HTTP que é executado por meio de TCP/IP. Eles são mensageiros instantâneos que conseguem trazer de maneira rápida e garantida a entrega de dados. Uma forma de visualizar seu funcionamento seria compará-lo a uma ligação telefônica onde alguém faz uma ligação para outra pessoa e quando esta atende, é criado um canal de comunicação entre os dois falantes. (FRANCO, 2005)

Dessa forma, a transmissão de dados seria mais adequada com o uso do protocolo via *Web-sockets*. Com isso, foi utilizada a biblioteca *channels* para criar os "consumidores" na API e a biblioteca "socket.io" para utilização do *socket* no *front-end*. Assim, os dados são enviados em tempo real sem custo adicional de outra ferramenta e atendendo a demanda existente.

Desde o início do desenvolvimento existia a necessidade de coletar uma base de

dados inicial para treinar o modelo e assim tê-lo eficiente com boas recomendações assim que entrasse em produção. Com isso, a ideia inicial era a de implementar uma captura de dados de conversões (CVs), *likes*, *dislikes* e perfis de busca no site da Liva e deixar o tempo o suficiente para obtenção de uma base de dados robusta. Mas devido a fatores relacionados a pandemia, o fluxo de pessoas no site da Liva diminuiu drasticamente, impossibilitando pôr em prática essa estratégia descrita anteriormente para obtenção dessa base de dados robusta inicial.

Dessa forma, houve a necessidade de uma nova estratégia para obtenção da base inicial. Após algum tempo analisando possibilidades foi escolhido um novo plano, que consistiu na utilização da base de dados dos *Leads*, adaptado para o modelo proposto.

Como os *Leads* são conversões (CVs) advindos de outras plataformas eles continuam sendo uma fonte valiosa de dados para o modelo proposto. Dessa forma, o novo plano baseia-se em manipular essa base de dados para se adequar ao modelo. Com a existência de dois tipos de *Leads*, primeiro e segundo contato, há a possibilidade de criar o que seriam os cliques em propriedades do cliente com os *Leads* de segundo contato. O *Lead* de primeiro contato seria o primeiro de um cliente demonstrando interesse em uma propriedade, enquanto o *Lead* de segundo contato são referentes aos *Leads* de clientes já cadastrados, ou seja, a partir da sua segunda interação. O "candidato" ou propriedade convertida seria a última propriedade de segundo contato. Com a existência do perfil de busca em cada *Lead* não houve problema com as *features* iniciais relacionadas ao perfil de busca.

Assim, foi feito um *script* na linguagem Ruby que foi executado no ambiente de produção da Liva, manipulando a base de dados dos *Leads*, incluindo uma "limpeza" nos dados, retirando dados de teste, repopulando campos nulos a partir de contato direto com as imobiliárias, adaptando os dados de classes e adicionando os perfis de busca. Com isso foi criada uma nova base de dados de conversões positivas, ou seja, com alvo igual a 1 ou CV igual a 1. Como todo modelo que irá classificar a probabilidade de um evento, há a necessidade de dados que representam a "não conversão" de propriedades, e isso a base de dados dos *Leads* não consegue proporcionar.

Por isso uma estratégia foi escolhida para popular a nova base de dados com "não conversões", ou de alvo 0. Ela consiste em, como igualmente uma estratégia adotada pelo *site* Summo.jp, a cada uma conversão de alvo 1, dez não conversões ou de alvo 0 são criadas, como descrito em (3.3.2.1). Dessa forma foi possível a geração de uma base de dados inicial com oitenta mil e trezentas tuplas aproximadamente.

Com a base de dados inicial obtida foi possível notar um desbalanceamento dos dados, pois existem mais alvos 0 do que 1. Neste caso, o desbalanceamento vem como algo positivo, pois na natureza do problema, em que existe uma conversão de um imóvel e outras 10 "não conversões" de outros imóveis para as mesmas *features* relacionadas as

pesquisas e interesses do usuário a ideia pode ser generalizada para o caso em que cada usuário tende a apresentar muitos comportamentos semelhantes até algum tipo de ação. O motivo de não ser feito uma base de dados em que uma conversão acontece quando um usuário pede informação e a "não conversão" quando um usuário não pede, é exatamente pelo fato de que muitas vezes o usuário tende a ver imóveis muito parecidos antes de realizar uma ação, ou seja ele pode voltar no imóvel a todo momento (LI et al., 2017).

4.2.1 *Sprint backlog* e resultados

Após a primeira *sprint*, em que foi possível adiantar o sistema, e com a pendencia criada com a *story* de registro das ações dos usuários foi criada a *story* técnica TS09 para aplicação do Socket TCP. Da mesma forma surgiu a *story* TS05 que refere-se a criação da nova base de dados, essencial para o prosseguimento do sistema, já que a primeira tentativa da criação da mesma não foi bem sucedida. O Quadro 4.2 apresenta o *backlog* e resultados da segunda *sprint*.

Stories planejadas	**Concluídas**	**Pendentes**	**Novas**
TS09 - Conectar *front-end* por *socket* com a API de recomendação	X		X
TS05 - Criar *dataset* a partir dos *leads*	X		X

Quadro 4.2 – *Sprint backlog* e resultados da *sprint* 2.

4.3 3° *Sprint*

Para esta *sprint* foi implementado um modelo de aprendizado de máquina simples com XGBoost utilizando a base de dados gerada dos *leads* e todo o fluxo de recepção e entrega de dados da API, assim como a integração com o AWS S3, a fim de serem feitos testes no fluxo completo do novo recomendador.

Para a implementação do *endpoint* de predição e o *script* de treino do modelo houve a necessidade de trazer os dados referente as propriedades, como descrito anteriormente (3.3.1.2), que primariamente seriam trazidos do banco da Liva através do serviço *DataUpdater*. Possibilitado pela tecnologia Django, a conexão com mais de um banco de dado simultaneamente, e sendo direta ao banco de dados da Liva, foi possível fazer buscas rápidas e fáceis das propriedades. Por se tratar apenas de uma tabela foi decidido descartar o sistema *DataUpdater*, que traria o custo de mais um serviço a ser implementado e mais recursos para mantê-lo.

Nesse ponto, com algumas reuniões relacionada ao produto geral da Liva, foi estabelecido que o usuário da plataforma deve selecionar mais de um bairro ou cidade no

campo de localidade do perfil de busca, pois isso afeta diretamente em outros produtos da Liva. Com isso, foi necessário adaptar o modelo com um perfil de busca com a possibilidade de existir mais de uma localidade, para não afetar o fluxo já existente da Liva.

Utilizando-se de um cálculo de média simples entre a latitude e longitude das possíveis localidades foi assim solucionado o problema apresentado anteriormente. Essa solução não é a mais ideal, mas devido ao modelo estar com uma dimensão muito alta, 576 colunas, tratar com métodos tradicionais como *one-hot-encoder*, ou até mesmo como foram tratados os "cliques" dos usuários, poderia aumentar ainda mais a dimensão do modelo, podendo trazer diversos problemas.

Agora, com uma base de dados integra, para se obter um melhor estimador possível e testar o modelo com a nova base de treino foi utilizada a técnica de pesquisa exaustiva sobre valores de parâmetros com *Grid Search*, como especificado no tópico 3.3.2.1.

Mesmo a obtenção do melhor estimador sendo automatizada, foi feito uma análise inicial dos hiper parâmetros com o objetivo de ser possível executar o modelo de forma rápida. Dessa forma, os hiper parâmetros analisados foram o "max_depth" e o "n_estimators", que são referentes ao nível máximo da árvore e número de folhas da árvore respectivamente. Foram testados para o nível da árvore, os valores de 2 á 10 e para o número de folhas, os valores de 50 à 400. Assim foi percebido que para o número de folhas sempre tendia ao maior número, enquanto para o nível máximo se manteve em 6.

Obtendo o melhor estimador com a melhor combinação de hiper parâmetros foi obtido uma acurácia de 98 por cento com o modelo. Com testes manuais foi possível analisar as recomendações geradas do novo modelo que apresentaram grande coerência. Porém foi notado um grande demora no tempo de resposta nas predições, na qual demorava em média de 15 a 20 segundos cada, e assim foi decidido a criação de uma nova *story* a fim de solucionar esse empecilho.

4.3.1 *Sprint backlog* e resultados

O foco escolhido para a terceira *sprint* foi em cima da *User Story* 05 que refere-se ao funcionamento por completo do sistema de recomendação baseado em aprendizado de máquina, já que da *sprint* anterior foi possível extrair uma base de dados e assim ser possível montar um modelo eficaz. Na implementação do *end-point* predição surgiu a *story* técnica TS04, que se refere ao problema de lidar com mais de uma localidade na matriz de predição. O quadro 4.3 apresenta o *backlog* e resultados da terceira *sprint*.

Stories planejadas	Concluídas	Pendentes	Novas
US05 - Visualizar recomendações na página do imóvel	X		
TS04 - Calcular média de latitude e longitude	X		X

Quadro 4.3 – *Sprint backlog* e resultados da *sprint* 3.

4.4 4° *Sprint*

Na *sprint* passada foi possível observar que o sistema não estava atendendo o requisito de tempo mínimo de espera pelas recomendações descritas no tópico de requisitos não funcionais de desempenho (3.1.4.2). Dessa forma, o próximo passo foi focado em encontrar um solução para minimizar o tempo de resposta do sistema. Além disso, toda a implementação do módulo baseado em crítica foi desenvolvido e também feita a configuração para o *deploy* no AWS.

Reimplementando a manipulação de dados no *endpoint* de predição do modelo foi possível otimizar bastante o tempo de resposta. Foi trocado todo fluxo de manipulação da matriz de predição por algoritmos de mais baixo nível, além da utilização de *Threads* (processamento em paralelo dos dados). Com isso, de 15 a 20 segundos em média esperados do clientes anteriormente, na primeira visita ao site, passaram a ser 2 segundos, e após a primeira visita, juntamente com a integração do Redis que irá carregar as propriedades em *cache*, instantaneamente.

Referente ao sistema de recomendação baseado em crítica, primeiramente houve a construção do componente no *front-end* referente as críticas presentes na página de detalhes. Com a sua construção já é possível gerar recomendação juntamente com o "alterar filtro" no *feed*, mas utilizando da recomendação já existente na Liva citada anteriormente, sem a utilização de técnicas.

Com reuniões na companhia do *Product Owner* a respeito das regras de negócio da Liva, juntamente com o gerente da empresa, foi decidido a não interferência do sistema de recomendação referente ao *feed*, temporariamente, pois o algoritmo de gerar recomendações está sendo utilizado em outro produto da empresa e iria interferir diretamente. Dessa forma, para manter o sistema de recomendação da *startup* e ao mesmo tempo aplicar o novo sistema de recomendação baseado em crítica, foi decido a criação de uma ferramenta de busca em uma nova aba na página principal do sistema, como mostrado nas seguintes Figuras 37, 38, 39, 40, 41 e 42.

Figura 37 – Nova aba de busca e o filtro de importância.

Figura 38 – Filtro de importância ao passar mouse.

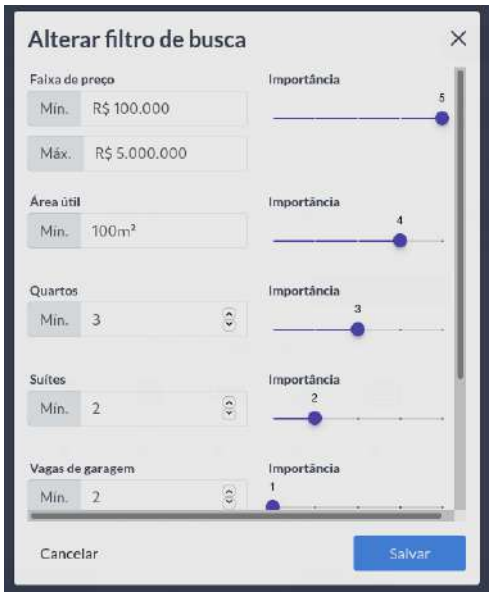

Figura 39 – Modal de alteração de filtro de busca - parte 1.

Figura 40 – Modal de alteração de filtro de busca - parte 2.

Figura 41 – Componente de descrição e crítica - parte 1.

Figura 42 – Componente de descrição e crítica - parte 2.

Essa ferramenta de busca se baseia nas características do perfil de busca, igualmente como acontece no algoritmo de recomendação do *feed*, mas difere-se no quesito em que o algoritmo utilizado é conforme o proposto nesse trabalho (3.3.3), ou seja, aplicando as técnicas de recomendações descritas. Um dos motivadores da utilização de uma ferramenta de busca é pelo fator de que, como descrito em 2.2.3.4, o sistema de recomendação baseado em crítica tem uma melhor performance quando utilizado juntamente com uma ferramenta de busca. Essa técnica foi feita para funcionar com tal tipo de ferramenta.

Em seguida, foi feita a implementação de todas as funções de pontuação referentes aos parâmetros do perfil de busca. Para dados quantitativos como quantidade de quartos, suítes e vagas na garagem, foi utilizado uma pontuação tendendo ao valor escolhido, ou seja, há uma pontuação máxima (100) caso o candidato tenha o valor escolhido no perfil de busca, e caso ele tenha diferença positiva, diminui da pontuação gradativamente.

Primeiramente, para calcular a pontuação referente a localização, foi utilizada uma equação linear, onde o valor resultante está entre 0 a 100. A localização da propriedade do candidato com menor distância ganha a maior pontuação (100) e a maior distância ganha a menor pontuação (0) sendo baseado nisso os outros candidatos ganham sua pontuação. Após testes manuais e reuniões foi notado que esse sistema de pontuação para localidade de imoveis não é adequado, pois existem bairros com diferenças econômicas muito grandes, ou seja, se tem um bairro colado a um outro, o sistema descrito anteriormente pode recomendar imoveis de um bairro para o outro e eles podem ser muito diferentes. Por esse motivo, foi decidido que esse filtro será estático, em que os candidatos a recomendação

estarão somente na localização definida pelo usuário. O tipo do imóvel foi outro filtro escolhido como estático, pois assim como a localização e por se tratar de uma ferramenta de busca, em que o usuário vai experimentando a ferramenta e descobrindo seu alcance, se encaixam mais como estáticos, filtrando somente o tipo que foi selecionado.

Para o preço, assim como anteriormente para a localização, foi feita uma equação linear utilizando-se dos parâmetros mínimo e máximo do perfil de busca, em que o preço de venda do imóvel candidato mais perto do preço mínimo definido no perfil de busca tende ao máximo de pontuação (100). Valores fora do mínimo e máximo são considerados de pontuação 0.

Para calcular a pontuação da área, por se tratar de um valor quadrático, foi retirado o grau da área do candidato pela areá miníma definida no perfil de busca. Caso o resultado passe de 100, por se tratar de uma área miníma, mantém o valor de 100 para a pontuação.

Para realizar o *deploy* no AWS, primeiro foi criado um repositório no ECR com o objetivo de armazenar e versionar a imagem do Docker do projeto. Ao mesmo tempo foi criado uma instância (servidor virtual na nuvem) no RDS para comportar o banco de dados Postgres. Com o registro da imagem armazenada, foi possível subir a API através do serviço ECS que ira mantê-lo em uma nova instância descrita em 3.1.4.2.

4.4.1 *Sprint backlog* e resultados

Passada a terceira *sprint* em que foi possível fazer testes manuais no recomendador de aprendizado de maquina, foi possível observar e calcular o tempo de resposta do algoritmo de predição que não estava adequado. Com isso, para essa *sprint* surgiu a *story* técnica TS10 com o objetivo de aumentar o desempenho da predição. Também é priorizado a US04 e US06 do recomendador de critica com o foco de finalizar o sistema para colocá-lo para coletar dados em produção. No decorrer da *sprint* com uma reunião antes de colocar o sistema em produção, um novo requisito surge para não ferir a regra de negocio da Liva, que é referente a nova ferramenta de busca com as *stories* de usuário US07 e US08. O quadro 4.4 apresenta o *backlog* e resultados da quarta *sprint*.

Stories planejadas	Concluídas	Pendentes	Novas
TS10 - Aumentar desempenho na parte de predição	X		X
US04 - Criticar características da propriedade	X		
US06 - Visualizar filtro de importância	X		X
US07 - Alterar filtro de importância	X		X
US08 - Acessar aba de busca no feed	X		X

Quadro 4.4 – *Sprint backlog* e resultados da *sprint* 4.

4.5 5° *Sprint*

Com o novo sistema de recomendação posto em produção no site da Liva foram possíveis fazer diversas análises iniciais. Uma delas foi relacionada ao fluxo de pessoas utilizando as recomendações, que estava baixo. Dessa forma, veio a decisão de adicionar um componente na lateral da página de detalhes e abaixo do componente de agendar visitas e tirar dúvidas, com o intuito de chamar a atenção do usuário, na qual um botão que se clicado rolará a página de detalhes para a posição do componente de apresentação de imóveis como na Figura 43.

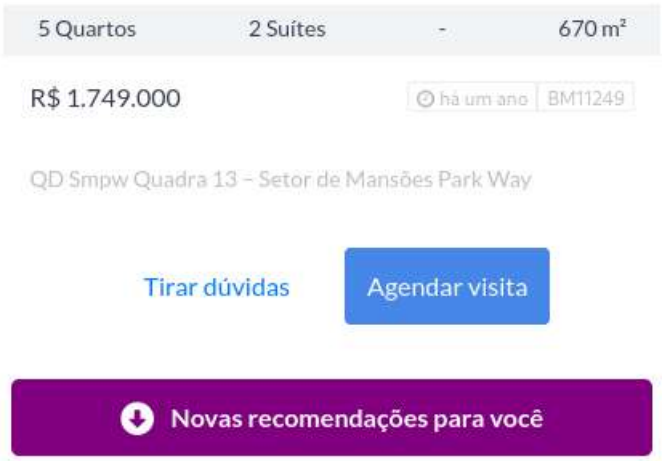

Figura 43 – Componente de encaminhamento.

Outra análise foi relacionada ao modelo de aprendizado de máquina, em que após alguns testes em um período longo de uso foi notado que quando o usuário chega a uma quantidade alta de cliques em propriedades, tende muito a recomendação das mesmas propriedades, ou seja, estava ocorrendo um *overfitting*. Dessa maneira, a solução foi descartar

alguns dados que não correspondiam com o "normal" e tinham muito mais interações que o comum e ainda diminuir para 30 o número de interações de cliques do modelo.

Por último, foi determinado melhorar o sistema adotado de localização, que como descrito anteriormente, para o perfil de busca que comporta vários bairros ou cidade é tirada a média de sua latitude e longitude. Para melhorar ainda mais a precisão do modelo foi decidido que um usuário clicar em uma propriedade, e a localidade dessa propriedade se encontra em seu perfil de busca, então ela é escolhida para entrada do modelo, assim deixando a localidade mais precisa e condizente com a realidade.

4.5.1 *Sprint backlog* e resultados

Com o sistema completo em produção, algumas modificações surgiram como necessárias. Primeiramente com a observação relacionada ao baixo fluxo de uso do componente de recomendação surge a US09 para implementação o botão para rolar a página. Outra *story* técnica surge (TS11) com a observação da tendencia do recomendador de aprendizado de máquina quando há muitos cliques pelo cliente, demonstrando um *overfitting*. O quadro 4.5 apresenta o *backlog* e resultados da quinta *sprint*.

Stories planejadas	**Concluídas**	**Pendentes**	**Novas**
US09 - Rolar página do imóvel para visualizar recomendações	X		X
TS11 - Limpeza do *dataset*	X		X

Quadro 4.5 – *Sprint backlog* e resultados da *sprint* 5.

4.6 Análises gerais

O novo sistema de recomendação foi colocado em produção por completo, com os dois sistemas de recomendação funcionando, na data de 08/10/2020 e foram coletados dados para serem analisados até a data de 11/11/2020. Neste período foram feitas diversas alterações nos recomendadores, como descrito anteriormente, sempre com o intuito de melhorá-los. Foram feitas mudanças na base de dados inicial, no algoritmo de predição, no design de componentes, nas funções de pontuação e dentre outras alterações importantes.

Para ser possível a medição da métrica descrita anteriormente em (3.1), além de novas análises, foram coletados dados do período de 03/09/2020 até 08/10/2020, para compará-los com os do período em que o novo sistema de recomendação entrou em produção. Para a coleta desses dados foi utilizada a ferramenta MixPanel, na qual foi utilizada também na simulação (Apêndice C), além de buscas no banco de dados da Liva.

Logo de início, ao buscar dados relacionados a quantidade de conversões positivas e não conversões e suas origens, dos diferentes períodos, temos a seguintes gráficos referentes

as Figuras 44 e 45:

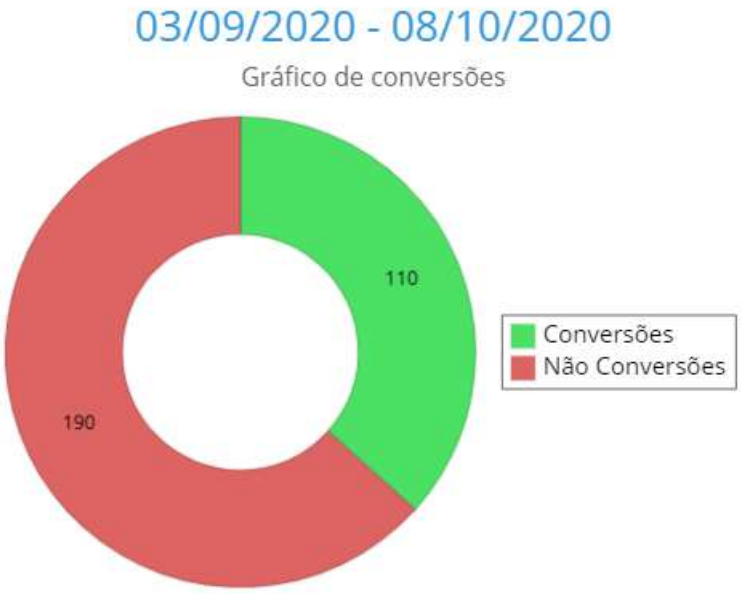

Figura 44 – Gráfico de conversões antes do novo recomendador.

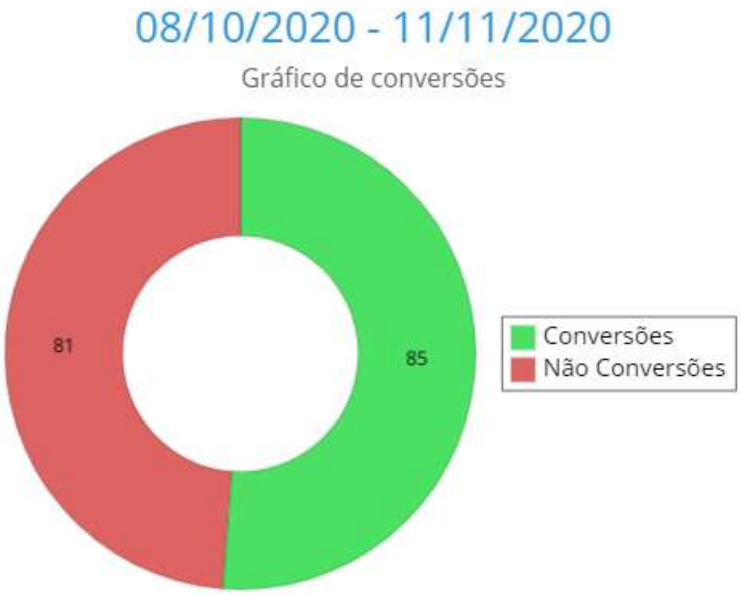

Figura 45 – Gráfico de conversões durante o novo recomendado.

Comparando as quantidades dentre conversões positivas e não conversões dos diferentes períodos é possível notar, primeiramente, para o período anterior a nova recomendação, uma quantidade muito maior de descartes de imoveis, o que mudou muito para o segundo período, em que as conversões se mantém equilibradas, ou seja, houve uma menor taxa de descartes, pro segundo período, como ocorrido também na simulação descrita no Apêndice (C).

Para calcular a métrica de CVR ou *conversion rate* descrita na Seção 3.1, coletamos, além das conversões, todos os cliques em recomendações dos usuários e a quantidade de usuários ativos nos dois períodos, demonstrados na Tabela (4).

Tabela 4 – Conversion Rate.

Data do início - Data do fim	03/09/2020 - 08/10/2020	08/10/2020 - 11/11/2020
Quantidade de cliques	623	293
Quantidade de Conversões	110	85
Quantidade de usuários ativos	125	113
CVR Aproximado	18%	29%

Comparando os resultados do CVR é possível notar uma melhora de 11% com a Liva compondo o novo sistema de recomendação elaborado. Na Tabela 4 é possível também perceber uma grande diminuição na quantidade de cliques, isso pode ser interpretado pelo motivo dos usuários estarem demorando menos tempo para encontrar o imóvel a ser convertido. Dessa forma, foi decidido trazer os dados referentes a frequência de usuários diários nos dois períodos.

Para o período anterior a nova recomendação existem 125 usuários ativos e para o período durante o uso, 113. Esses dados reforçam ainda mais que a grande diminuição na quantidade de cliques nos dois períodos é relacionada ao tempo do usuário encontrar o imóvel ideal, pois o fluxo de usuários se manteve muito parecido, com apenas 12 usuários de diferença.

5 Considerações finais

Esse último capítulo apresenta os resultados atingidos da solução depois do fluxo de desenvolvimento, além da avaliação do mesmo no qual foi analisado se o objetivo proposto foi alcançado. Além disso, será sugerido possíveis trabalhos futuros que podem ser desenvolvidos a partir dessa solução.

5.1 Conclusão

Para alcançar os objetivos descritos na Seção 1.1, que referem-se ao desenvolvimento de um sistema de recomendação, foi realizado inicialmente um levantamento bibliográfico acerca do tema, além da realização da documentação referente a arquitetura do projeto. Dessa forma, a descrição de como seria todo funcionamento do sistema proposto e seu processo de desenvolvimento a partir da análise e planejamento, projeto e desenvolvimento envolvendo a metodologia adequada aos objetivos almejados.

Algumas mudanças na fase de desenvolvimento foram necessárias como foi descrito no Capítulo 4, para assim poder atingir os objetivos com melhor qualidade no processo.

O objetivo principal desse trabalho foi a construção de um sistema de recomendação que funcionaria como um serviço mais interessante para o ambiente virtual da Liva.vc, podendo ele proporcionar ao usuário, possível cliente, uma melhor experiência de uso com recomendações mais coerentes com as demandas e preferências (re)conhecidas.

Após a fase de desenvolvimento (Capítulo 4) foi observado que o sistema de recomendação elaborado conseguiu contribuir para agregar um maior valor aos clientes do sistema da Liva. Essa observação foi possível através de cálculos e analises de métricas como a CVR (Conversion Rate).

Ao coletar dados do sistema de recomendação elaborado em ambiente de produção por aproximadamente um mês juntamente com o período anterior de mesma duração, foi possível fazer o cálculo da métrica CVR, que se refere a taxa de conversões de imóveis dentre as recomendações geradas para os usuários. Dessa forma, foi possível comparar o CVR para os dois períodos que demonstrou uma melhora de 11% para o período do novo sistema de recomendação elaborado. Além disso, foi comparado também a taxa de usuários ativos para os diferentes períodos, na qual demonstrou uma proporção parecida, conferindo maior credibilidade para a métrica CVR.

Foi constatado que os objetivos indicados na seção 1.1 foram atingidos no final de todo o processo, como a construção de todo o sistema de recomendação, a base de dados robusta para o modelo de aprendizado de máquina e a análise final em cima das métricas.

Para cada usuário, o sistema de recomendação baseado em aprendizado de máquina irá sugerir imóveis semelhantes com o seu perfil, utilizando-se de características de imóveis em que o usuário estava navegando e em seu filtro de busca. O outro módulo do sistema de recomendação implementado é baseado na abordagem de crítica, na qual o usuário é capaz de "criticar" o imóvel sendo visualizado e utilizar também uma ferramenta de busca de imóveis, em que ele pode colocar o nível de importância em diferentes atributos de propriedade.

Dessa forma, é possível observar o benefício gerado pelo novo sistema de recomendação implantado, em que o usuário ao entrar e navegar no sistema da Liva poderá ver recomendações de propriedades, facilitando mais sua pesquisa por um imóvel ideal, aumentando as chances de encontrá- lo e diminuindo o tempo de busca. A cada imóvel que o usuário navega melhorará a precisão das sugestões do sistema de recomendação para aquele determinado usuário, e quanto mais usuários utilizarem a ferramenta, melhores serão as recomendações geradas.

5.2 Trabalhos futuros

Esse trabalho tinha como escopo desenvolver um sistema de recomendação híbrido para uma plataforma de *e-commerce* para a empresa Liva no ramo imobiliário, a fim de melhorar o serviço já utilizado por seus clientes, recomendando imóveis parecidos com o perfil de cada usuário a partir de dois módulos de recomendação desenvolvidos (baseado em crítica e baseado em aprendizado de máquina). No capítulo 4 foram detalhadas as mudanças necessárias a partir das modificações na proposta inicial do trabalho para melhor alcançar os objetivos propostos.

No sentido de sempre continuar melhorando o software é necessário prosseguir com novas evoluções com o intuito de trazer melhorias para o usuário final. Com isso, foram levantadas algumas *features* em que aumentariam o valor de negócio do sistema de recomendação proposto. Estas novas *features* seriam:

- Na parte de alterar os valores das características do imóvel no componente de crítica (Figura 42), apresentaria dicas de possíveis valores para cada atributo do imóvel, com base em outros imóveis parecidos com aquele que está sendo visualizado, ou seja, ao usuário criticar o preço de alguma propriedade, o sistema irá comparar esse imóvel com outros imóveis da mesma localidade e com valores dos atributos parecidos. Assim poderá apresentar para o usuário possíveis valores de preço para aquele domínio, sem ele precisar saber qual seria a faixa de preço para aquele tipo de imóvel que está sendo procurado. Outro exemplo, seria exibir outros possíveis valores de quarto, banheiro, suíte ou vagas de garagem compatível.

- Um módulo de qualificação do cliente poderia ser agregado no modelo de aprendizado de máquina gerando novas *features*, em que se poderia obter, por meio de formulários ao cliente, como por exemplo com perguntas do tipo: "Utilizará FGTS na compra do imóvel?". Dessa forma, será possível determinar o momento de compra do cliente gerando novas *features* para o modelo e assim melhorando sua eficácia. Outra possibilidade de novas *features* seria relacionado as ações dos corretores, como por exemplo possíveis imóveis recomendados diretamente do corretor, ou um *feedback* do corretor em relação ao cliente.

- Mais qualificações em cima das propriedades, criando algoritmos para gerar *features* com *tags* de localidade, como por exemplo se há hospitais, mercados e parques próximos ao imóvel ou características do bairro, se é mais seguro, ou ainda ter características como personalidades de pessoas que moram próximas. Todas essas características agregariam ainda mais no perfil do cliente e ajudaria a encontrar melhores recomendações.

Referências

ABBATTISTA et al. Learning user profiles for content-based filtering in e-commerce. 08 2002. Citado na página 34.

AGENCIAKAIZEN. *O que são leads*. 2019. Disponível em: <https://www.agenciakaizen.com.br/leads>. Citado na página 18.

ALMANA, A.; AKSOY, M. An overview of inductive learning algorithms. *International Journal of Computer Applications*, v. 88, 01 2014. Citado na página 46.

ALPAYDIN, E. *Introduction to Machine Learning*. 2nd. ed. [S.l.]: The MIT Press, 2010. ISBN 026201243X, 9780262012430. Citado 2 vezes nas páginas 39 e 40.

AMAZON. *Amazon S3*. 2019. Disponível em: <https://aws.amazon.com/pt/s3/>. Citado na página 86.

AMAZON. *O que é Amazon Elastic Container Service?* 2019. Disponível em: <https://docs.aws.amazon.com/pt_br/AmazonECS/latest/developerguide/Welcome.html>. Citado na página 86.

ARRAES, R.; FILHO, E. Externalidades e formação de preços no mercado imobiliário urbano brasileiro: Um estudo de caso. *Economia Aplicada*, v. 12, 01 2008. Citado na página 22.

ARTIA. *O que é e TUDO sobre como gerenciar fluxos de trabalho*. 2019. Disponível em: <https://artia.com/kanban/>. Citado na página 56.

BARRETO, R. O marketing na criação da imagem positiva da imobiliária. 2019. ISSN 2448-2889. Citado na página 22.

BATISTA, G. *Pré-processamento de dados em aprendizado de máquina supervisionado*. Tese (Doutorado) — USP, 01 2003. Citado na página 42.

BELIAKOV, G.; CALVO, T.; JAMES, S. Aggregation of preferences in recommender systems. In: *Recommender Systems Handbook*. [S.l.: s.n.], 2011. Citado na página 31.

BOOTSTRAP. *Bootstrap*. 2019. Disponível em: <https://getbootstrap.com/>. Citado na página 84.

BORGES, L. *Funil de Marketing Digital. São Paulo: Portal QLuz: Planilhas empresariais*. 2017. Disponível em: <https://blog.luz.vc/o-que-e/funil-de-marketing-digital/>. Citado na página 22.

BREIMAN, L. Random forests. *Mach. Learn.*, Kluwer Academic Publishers, Norwell, MA, USA, v. 45, n. 1, p. 5–32, out. 2001. ISSN 0885-6125. Disponível em: <https://doi.org/10.1023/A:1010933404324>. Citado 2 vezes nas páginas 45 e 47.

BURKE, R. The adaptive web. In: BRUSILOVSKY, P.; KOBSA, A.; NEJDL, W. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2007. cap. Hybrid Web Recommender Systems, p. 377–408. ISBN 978-3-540-72078-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1768197.1768211>. Citado 2 vezes nas páginas 35 e 36.

BURKE, R.; FELFERNIG, A.; GöKER, M. Recommender systems: An overview. *Ai Magazine*, v. 32, p. 13–18, 09 2011. Citado na página 76.

CARLOS, P.; ORSI, M.; CRUZ, B. H. Caracteristicas qualitativas, quantitativas de abordagens cientificas: estudos de caso na subarea do design ergonomico. *Revista de Design, Tecnologia e Sociedade*, v. 2, n. 1, p. 65–78, out. 2018. Disponível em: <https://periodicos.unb.br/index.php/design-tecnologia-sociedade/article/view/15699>. Citado na página 50.

CHAKURE, A. *Random Forest Regression*. 2019. Disponível em: <https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f>. Citado na página 47.

CHEN, L.; PU, P. Critiquing-based recommenders: Survey and emerging trends. *User Modeling and User-Adapted Interaction*, v. 22, 04 2012. Citado 3 vezes nas páginas 36, 37 e 38.

CHEPENKO, D. *Introduction to gradient boosting on decision trees with Catboost*. 2019. Disponível em: <https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14>. Citado na página 48.

CIMINI, M. *Deep Learning for Collaborative Filtering (using FastAI)*. 2019. Disponível em: <https://mc.ai/deep-learning-for-collabora>. Citado 2 vezes nas páginas 25 e 32.

CORREA, R. et al. A influência do conhecimento e da atitude no comportamento de investidor do mercado imobiliário. *Revista de Administração FACES Journal*, v. 17, p. 100–116, 10 2018. Citado na página 22.

COSTA, E.; POLITANO, P.; PEREIRA, N. Exemplo de aplicação do método de pesquisa-ação para a solução de um problema de sistema de informação em uma empresa produtora de cana-de-açúcar. *Gestão e Produção*, v. 21, p. 895–905, 12 2014. Citado na página 51.

DJANGO. *Why Django?* 2019. Disponível em: <https://www.djangoproject.com/start/overview/>. Citado na página 81.

DOCKERHUB. *Build and Ship any Application Anywhere*. 2019. Disponível em: <https://hub.docker.com/>. Citado na página 85.

EKSTRAND, M. D.; RIEDL, J. T.; KONSTAN, J. A. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, Now Publishers Inc., Hanover, MA, USA, v. 4, n. 2, p. 81–173, fev. 2011. ISSN 1551-3955. Disponível em: <http://dx.doi.org/10.1561/1100000009>. Citado 4 vezes nas páginas 27, 28, 32 e 33.

FRANCO, D. d. S. C. AplicaÇÃo de sockets em java para monitoramento de processos em estaÇÕes de trabalho. 2005. Disponível em: <https://www.aedb.br/seget/arquivos/artigos05/342_ARTIGO_SeGET%20(19.09.2005).pdf>. Citado na página 90.

FRANKENFIELD, J. *How Amazon Makes Money*. 2019. Disponível em: <https://www.investopedia.com/how-amazon-makes-money-4587523>. Citado na página 17.

FREITAS, C. de. *Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição*. Editora Feevale, 2013. ISBN 9788577171583. Disponível em: <https://books.google.com.br/books?id=zUDsAQAAQBAJ>. Citado 2 vezes nas páginas 50 e 51.

GARCIA, J. M. G. Machine learning in a recommendation system. 2015. Citado na página 20.

GIL, A. C. Como elaborar projetos de pesquisa. p. 16–17, 2002. Citado na página 50.

GIL, A. C. Métodos e técnicas de pesquisa social. 2008. Citado na página 50.

GIT. *Sobre*. 2019. Disponível em: <https://git-scm.com/about>. Citado na página 86.

GODOY, T. D. L. . A. G. . R. H. C. P. . W. F. Aplicação do algoritmo random forest como classificador de padrões de falhas em rolamentos de motores de indução. 2017. Citado na página 47.

GORMAN, B. *A Kaggle Master Explains Gradient Boosting*. 2017. Disponível em: <http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/>. Citado na página 48.

GRIMALDI, E. *How to build a content-based movie recommender system with Natural Language Processing*. 2018. Disponível em: <https://bit.ly/2rb6XG9>. Citado na página 34.

GROVER, P. *Various Implementations of Collaborative Filtering*. 2017. Disponível em: <https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0>. Citado na página 33.

HAN, J.; KAMBER, M.; PEI, J. *Data Mining: Concepts and Techniques*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 0123814790, 9780123814791. Citado 4 vezes nas páginas 19, 40, 41 e 42.

HOARE, J. *XGBoost Hyperparameter Tuning - A Visual Guide*. 2019. Disponível em: <https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/>. Citado 2 vezes nas páginas 47 e 48.

INSIDEOUT. *Como acompanhar um funil de marketing?* 2018. Disponível em: <https://agenciainsideout.com/como-acompanhar-um-funil-de-marketing/>. Citado 2 vezes nas páginas 23 e 24.

JAIN, A. *Movie Recommendation System — Content Filtering*. 2019. Disponível em: <https://medium.com/data-science-101/movie-recommendation-system-content-filtering-7ba425ca0920>. Citado na página 35.

JAVASCRIPT. *JavaScript*. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Glossario/JavaScript>. Citado na página 83.

JIANG, Y. et al. Redesigning promotion strategy for e-commerce competitiveness through pricing and recommendation. *International Journal of Production Economics*, v. 167, n. C, p. 257–270, 2015. Disponível em: <https://ideas.repec.org/a/eee/proeco/v167y2015icp257-270.html>. Citado na página 17.

KARABULUT, A.; ERGUN, E. A new way of management: A scrum management. *International Journal of Commerce and Finance*, v. 4, n. 2, p. 108–117, 2018. ISSN 2149-9608. Disponível em: <http://ijcf.ticaret.edu.tr/index.php/ijcf/article/view/94>. Citado na página 55.

KASAMANI, B.; NAHIMANA, G. A system for recommending rental properties. *Journal of Systems Integration*, v. 8, 08 2017. Citado na página 38.

KINESIS. *O que é o Amazon Kinesis Data Streams?* 2019. Disponível em: <https://docs.aws.amazon.com/pt_br/streams/latest/dev/introduction.html>. Citado 2 vezes nas páginas 86 e 89.

KLEINMAN, A. *Netflix May Only Give You 3 Or 4 Recommendations In The Future*. 2014. Disponível em: <https://www.huffpostbrasil.com/2014/05/20/netflix-recommendations_n_5357619.html>. Citado na página 17.

KNOWLEDGE21. *O que é a User Story?* 2019. Disponível em: <https://www.knowledge21.com.br/sobreagilidade/user-stories/o-que-e-user-story>. Citado na página 55.

KOREN, Y.; BELL, R. *Advances in Collaborative Filtering*. [S.l.: s.n.], 2015. 77-118 p. ISBN 978-1-4899-7636-9. Citado na página 32.

KOTSIANTIS, S. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, v. 31, 10 2007. Citado 2 vezes nas páginas 44 e 45.

LENON. *Node.js – O que é, como funciona e quais as vantagens*. 2018. Disponível em: <https://www.opus-software.com.br/node-js/>. Citado na página 83.

LEVINAS, C. A. An analysis of memory based collaborative filtering recommender systems with improvement proposals. In: . [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 32 e 33.

LI, S. et al. Web-scale personalized real-time recommender system on suumo. p. 521–538, 04 2017. Citado 6 vezes nas páginas 17, 21, 71, 75, 76 e 92.

LINDEN, G.; SMITH, B.; YORK, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 7, n. 1, p. 76–80, jan. 2003. ISSN 1089-7801. Disponível em: <http://dx.doi.org/10.1109/MIC.2003.1167344>. Citado na página 16.

LIVA. *Uma nova forma de comprar e vender imóveis*. 2019. Disponível em: <https://liva.vc>. Citado 4 vezes nas páginas 18, 61, 62 e 63.

LUK, K. *Introduction to TWO approaches of Content-based Recommendation System*. 2019. Disponível em: <https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c>. Citado na página 34.

LUO, S. *Introduction to Recommender Systemg*. 2018. Disponível em: <https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26>. Citado na página 31.

MAHMOOD, T.; RICCI, F. Improving recommender systems with adaptive conversational strategies. In: *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*. New York, NY, USA: ACM, 2009. (HT '09), p. 73–82. ISBN 978-1-60558-486-7. Disponível em: <http://doi.acm.org/10.1145/1557914.1557930>. Citado na página 25.

MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072. Citado 2 vezes nas páginas 39 e 46.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. In: *Sistemas Inteligentes Fundamentos e Aplicações*. 1. ed. Barueri-SP: Manole Ltda, 2003. p. 89–114. ISBN 85-204-168. Citado 2 vezes nas páginas 42 e 43.

MORETTI, I. *Metodologia de Pesquisa do TCC: conheça os tipos e veja como definir*. 2018. Disponível em: <https://viacarreira.com/metodologia-de-pesquisa-do-tcc/>. Citado na página 51.

NAKAMURA, A. M. ComÉrcio eletrÔnicoriscos nas compras pela internet. 2011. Citado 2 vezes nas páginas 24 e 25.

NETFLIXPRIZE. *Netflix Prize*. 2009. Disponível em: <https://www.netflixprize.com/index.html>. Citado 2 vezes nas páginas 17 e 28.

NILASHI, M. et al. Collaborative filtering recommender systems. *Research Journal of Applied Sciences, Engineering and Technology*, v. 5, p. 4168–4182, 04 2013. Citado na página 31.

NODEJS. *Sobre o Node.js*. 2019. Disponível em: <https://nodejs.org/en/about/>. Citado na página 83.

NUMPY. *Numpy*. 2019. Disponível em: <https://numpy.org>. Citado na página 82.

OPENSOURCE. *What is Docker?* 2019. Disponível em: <https://opensource.com/resources/what-docker>. Citado na página 85.

PANDAS. *Python Data Analysis Library*. 2019. Disponível em: <https://pandas.pydata.org>. Citado na página 82.

PANDEY, P. *The Remarkable world of Recommender Systems - An overview of the Recommendation systems and how they provide an effective form of targeted marketing*. 2019. Disponível em: <https://towardsdatascience.com/the-remarkable-world-of-recommender-systems-bff4b9cbe6a7>. Citado na página 26.

PICKLE. *Python object serialization*. 2019. Disponível em: <https://docs.python.org/3/library/pickle.html>. Citado na página 82.

PINELA, C. *Recommender Systems — User-Based and Item-Based Collaborative Filtering*. 2017. Disponível em: <https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f>. Citado na página 33.

PLAPINGER, T. *What is a Decision Tree?* 2017. Disponível em: <https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1>. Citado na página 46.

PORTALGSTI. *O que é Ruby on Rails?* 2019. Disponível em: <https://www.portalgsti.com.br/ruby-on-rails/sobre/>. Citado na página 84.

POSTEGRESQL. *Sobre*. 2019. Disponível em: <https://www.postgresql.org/about/>. Citado na página 85.

PREDEGOSA. *Scikit-learn: Aprendizado de Máquina em Python*. 2011. Disponível em: <http://jmlr.org/papers/v12/pedregosa11a.html>. Citado na página 82.

PYTHON. *O tutorial do Python*. 2019. Disponível em: <https://docs.python.org/3/tutorial/index.html#tutorial-index>. Citado na página 80.

REACTJS. *Uma biblioteca JavaScript para criar interfaces de usuário*. 2019. Disponível em: <https://pt-br.reactjs.org/>. Citado na página 83.

REDIS. *Introduction to Redis*. 2019. Disponível em: <https://redis.io/topics/introduction>. Citado na página 85.

REST, D. *Estrutura Rest do Django*. 2019. Disponível em: <https://www.django-rest-framework.org>. Citado na página 81.

RICCI, F. et al. *Recommender Systems Handbook*. 1st. ed. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN 0387858199, 9780387858197. Citado 6 vezes nas páginas 16, 25, 28, 29, 30 e 35.

RIEDL, J. H. . J. K. . L. T. . J. . Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, v. 22, p. 5–53, 01 2004. Citado na página 29.

RUBY. *Sobre o Ruby*. 2019. Disponível em: <https://www.ruby-lang.org/pt/about/>. Citado na página 84.

SHARMA, R.; SINGH, R. Evolution of recommender systems from ancient times to modern era: A survey. *Indian Journal of Science and Technology*, v. 9, 05 2016. Citado 2 vezes nas páginas 26 e 27.

SIRIKULVIRIYA, N.; SINTHUPINYO, S. Integration of rules from a random forest. 2011. Citado na página 45.

SMITH, R.; SPEAKER, M.; THOMPSON, M. *O mais completo guia sobre e-commerce*. Futura, 2000. ISBN 9788574130385. Disponível em: <https://books.google.com.br/books?id=f2SOtgAACAAJ>. Citado na página 24.

SOUZA, R. G. D. de. *Sistemas de Recomendação - Aplicando Sistemas de Recomendação em Situações Práticas*. 2014. Disponível em: <https://www.ibm.com/developerworks/br/local/data/sistemas_recomendacao/index.html>. Citado na página 16.

SUTHERLAND, J. *Jeff Sutherland's Scrum Handbook*. [S.l.: s.n.], 2010. Citado 2 vezes nas páginas 58 e 59.

SYNCED. *Tree Boosting With XGBoost — Why Does XGBoost Win "Every" Machine Learning Competition?* 2017. Disponível em: <https://syncedreview.com/2017/10/22/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition/>. Citado na página 75.

UNDERWOOD, C. *Use Cases of Recommendation Systems in Business - Current Applications and Methods*. 2017. Disponível em: <https://www.techemergence.com/use-cases-recommendation-systems/>. Citado na página 16.

VARON, L. *Forrester Has Expanded Its Coverage Of eCommerce In Latin America!* 2019. Disponível em: <https://go.forrester.com/blogs/forrester-has-expanded-our-coverage-of-ecommerce-in-latin-america/>. Citado na página 17.

VECMANIS, K. *XGBoost Hyperparameter Tuning - A Visual Guide*. 2019. Disponível em: <https://kevinvecmanis.io/machine%20learning/hyperparameter%20tuning/dataviz/python/2019/05/11/XGBoost-Tuning-Visual-Guide.html>. Citado na página 47.

VIAPPIANI, P.; FALTINGS, B.; PU, P. Evaluating preference-based search tools: a tale of two approaches. *Proceedings of the National Conference on Artificial Intelligence*, v. 1, 01 2006. Citado na página 77.

WITTEN, I. H. et al. *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques*. 4th. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. ISBN 0128042915, 9780128042915. Citado 2 vezes nas páginas 39 e 40.

XGBOOST. *Documentação do XGBoost*. 2019. Disponível em: <https://xgboost.readthedocs.io/en/latest/>. Citado na página 83.

YUAN, X. et al. Toward a user-oriented recommendation system for real estate websites. *Inf. Syst.*, Elsevier Science Ltd., Oxford, UK, UK, v. 38, n. 2, p. 231–243, abr. 2013. ISSN 0306-4379. Disponível em: <http://dx.doi.org/10.1016/j.is.2012.08.004>. Citado na página 17.

ZENHUB. *Agile Project Management and Product Roadmaps Inside GitHub*. 2019. Disponível em: <https://www.zenhub.com>. Citado na página 86.

ZISOPOULOS, H. et al. Content-based recommendation systems. 11 2008. Citado na página 26.

Apêndices

APÊNDICE A – Backlog do Produto Inicial

FEATURE	STORIE	NOME	EU, COMO	DESEJO	PARA	PRIORIDADE	CALENDÁRIO
FE01 - Recomendador baseado em crítica	**US01**	Visualizar perfil de busca	usuário	visualizar meu perfil de busca	ter conhecimento de como está configurado	Must	1º sprint
	US02	Alterar perfil de busca	usuário	alterar meu perfil de busca	ajustar as preferências e receber recomendações mais precisas	Must	1º sprint
	US03	Visualizar recomendações na página principal	usuário	visualizar os imóveis recomendados com base nas minhas preferências configuradas no perfil de busca	saber quais imóveis são recomendados para mim	Must	3º sprint
	US04	Criticar características da propriedade	usuário	criticar as características de um ou vários imóveis	ter recomendações mais adequadas com o que eu desejo	Must	4º sprint
FE02 - Recomendador baseado em aprendizado de máquina	**US05**	Visualizar recomendações na página do imóvel	usuário	visualizar recomendações de imóveis baseadas nas minhas ações no sítio virtal e na página de detalhes do imóvel	diminuir o tempo de procura do imóvel ideal para mim	Must	3º sprint

FEATURE	STORIE	NOME	EU, COMO	DESEJO	PARA	PRIORIDADE	CALENDÁRIO
FE03 - Bugs no sistema da Liva	**TS01**	Corrigir bugs existentes na página de detalhes do imóvel	desenvolvedor	corrigir bugs que existem na página de detalhes do imóvel	poder implementar o sistema de recomendação	Must	1º sprint
	TS02	Corrigir bug de agendamento de visitas	desenvolvedor	corrigir bug de agendamento de visitas	o usuário poder marcar sua agenda ao imóvel	Must	1º sprint
FE02 - Recomendador baseado em aprendizado de máquina	**TS03**	Registrar ações do usuário no sítio virtual	desenvolvedor	registrar as ações do usuário no site	poder predizer quais são os imóveis mais prováveis da preferência do usuário	Must	1º sprint

FEATURE	DESCRIÇÃO
FE01 - Recomendador baseado em crítica	possibilitará ao usuário receber recomendações de imóveis baseadas em críticas sobre características de imóveis, feitas pelo próprio usuário
FE02 - Recomendador baseado em aprendizado de máquina	permitirá ao usuário receber recomendações de imóveis baseado nas ações dele no sítio virtual
FE03 - Bugs no sistema da Liva	permitirá o desenvolvedor começar a implementar o sistema de recomendação no sistema da Liva

Figura 46 – Backlog do produto inicial.

APÊNDICE B – Backlog do Produto final

FEATURE	STORIE	NOME	EU, COMO	DESEJO	PARA	PRIORIDADE	CALENDÁRIO
FE01 - Recomendador baseado em crítica	US01	Visualizar perfil de busca	usuário	visualizar meu perfil de busca	ter conhecimento de como está configurado	Must	1º sprint
	US02	Alterar perfil de busca	usuário	alterar meu perfil de busca	ajustar as preferências e receber recomendações mais precisas	Must	1º sprint
	US03	Visualizar recomendações na página principal	usuário	visualizar os imóveis recomendados com base nas minhas preferências configuradas no perfil de busca	saber quais imóveis são recomendados para mim	Must	3º sprint
	US04	Criticar características da propriedade	usuário	criticar as características de um ou vários imóveis	ter recomendações mais adequadas com o que eu desejo	Must	4º sprint
	US06	Visualizar filtro de importância	usuário	visualizar o grau da importância dos atributos do imóvel no filtro	ter conhecimento de como a busca está configurado	Should	5º sprint
	US07	Alterar filtro de importância	usuário	alterar o grau de importância dos atributos do imóvel no filtro	ajustar as preferências e visualizar os resultados da busca que eu quero	Should	5º sprint
	US08	Acessar aba de busca no feed	usuário	acessar a parte de busca de imóveis na plataforma	poder buscar por imóveis da minha preferência	Should	5º sprint
FE02 - Recomendador baseado em aprendizado de máquina	US05	Visualizar recomendações na página do imóvel	usuário	visualizar recomendações de imóveis baseadas nas minhas ações no sítio virtal e na página de detalhes do imóvel	diminuir o tempo de procura do imóvel ideal para mim	Must	3º sprint
	US09	Rolar página do imóvel para visualizar recomendações	usuário	rolar página do imóvel até visualizar recomendações em um clique	visualizar recomendações de imóveis de forma fácil e rápida	Should	5º sprint

FEATURE	STORIE	NOME	EU, COMO	DESEJO	PARA	PRIORIDADE	CALENDÁRIO
FE03 - Bugs no sistema da Liva	TS01	Corrigir bugs existentes na página de detalhes do imóvel	desenvolvedor	corrigir bugs que existem na página de detalhes do imóvel	poder implementar o sistema de recomendação	Must	1º sprint
	TS02	Corrigir bug de agendamento de visitas	desenvolvedor	corrigir bug de agendamento de visitas	o usuário poder marcar sua agenda ao imóvel	Must	1º sprint
FE02 - Recomendador baseado em aprendizado de máquina	TS03	Registrar ações do usuário no sítio virtual	desenvolvedor	registrar as ações do usuário no site	poder predizer quais são os imóveis mais prováveis da preferência do usuário	Must	1º sprint, 2º sprint
	TS04	Calcular média de latitude e longitude	desenvolvedor	calcular média de latitude e longitude	caso o usuário esteje procurando um imóvel em diferentes localidades	Should	3º sprint
	TS05	Criar dataset a partir dos leads	desenvolvedor	criar dataset a partir dos leads	ter uma melhor predição	Must	2º sprint
	TS08	Melhorar design do componente de recomendação	desenvolvedor	melhorar design do componente de recomendação	prover uma melhorar experiência ao usuário	Could	1º sprint
	TS09	Conectar front-end por socket com a API de recomendação	desenvolvedor	substituir AWS Kinesis por conexão socket entre o front-end e a API de recomendação	fornecer uma conexão de alto desempenho entre os dois serviços	Must	2º sprint
	TS10	Aumentar desempenho na parte de predição	desenvolvedor	modificar o set dos dados e implementar thread no processo de predição	diminuir o tempo da predição	Could	4º sprint
	TS11	Limpeza do dataset	desenvolvedor	realizar a limpeza do dataset	para evitar overfitting no modelo	Could	5º sprint
FE01 - Recomendador baseado em crítica	TS06	Melhorar design da modal do perfil de busca	desenvolvedor	melhorar design da modal do perfil de busca	prover uma melhorar experiência ao usuário	Could	1º sprint
	TS07	Melhorar design do componente de crítica	desenvolvedor	melhorar design do componente de crítica	prover uma melhorar experiência ao usuário	Could	1º sprint
FE04 - Configuração da infraestrutura	TS12	Configurar integração contínua	desenvolvedor	configurar integração contínua	realizar testes unitários, verificação da folha de estilho e de qualidade de código	Must	1º sprint
	TS13	Configurar deploy automático	desenvolvedor	configurar deploy automático	disponibilizar serviço em produção para poder ser utilizado	Must	1º sprint

FEATURE	DESCRIÇÃO
FE01 - Recomendador baseado em crítica	possibilitará ao usuário receber recomendações de imóveis baseadas em críticas sobre características de imóveis, feitas pelo próprio usuário
FE02 - Recomendador baseado em aprendizado de máquina	permitirá ao usuário receber recomendações de imóveis baseado nas ações dele no sítio virtual
FE03 - Bugs no sistema da Liva	permitirá o desenvolvedor começar a implementar o sistema de recomendação no sistema da Liva
FE04 - Configuração da infraestrutura	permitirá a verificação do sistema pelo desenvolvedor e a disponibilização do serviço para o usuário

Figura 47 – Backlog do produto atualizado.

APÊNDICE C – Simulação da Proposta

C.1 Introdução

Com o objetivo de validar a aplicabilidade da proposta, foi desenvolvido um protótipo , com partes disponíveis no Apêndice A deste trabalho. Para o desenvolvimento desse protótipo, como a solução se baseia em duas abordagens diferentes (módulo de critica e módulo de aprendizado de máquina), foi decidido fazer a implementação das duas de forma reduzida e minimalista. De forma geral, para o recomendador baseado em critica foi disponibilizado para o usuário uma pergunta que servirá para criticar apenas uma característica do imóvel, como apresentado nas Figuras 48, 49 e 50.

Figura 48 – Protótipo da simulação da crítica parte 1.

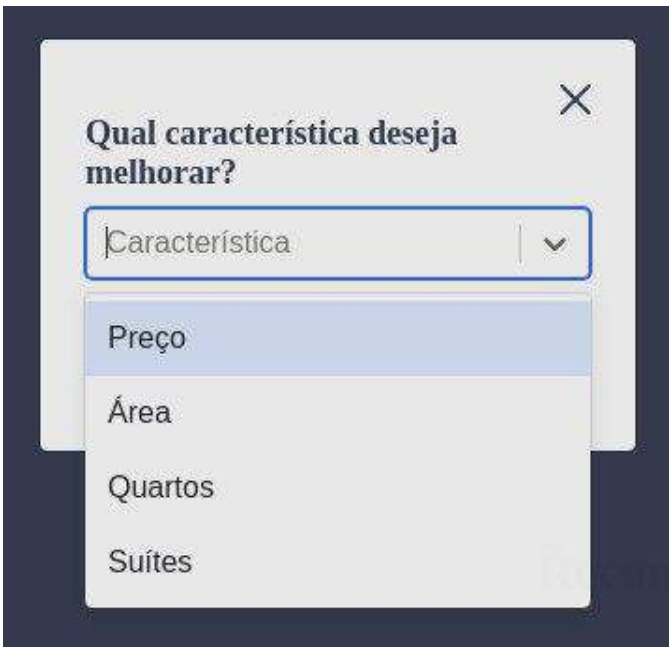

Figura 49 – Protótipo da simulação da crítica parte 2.

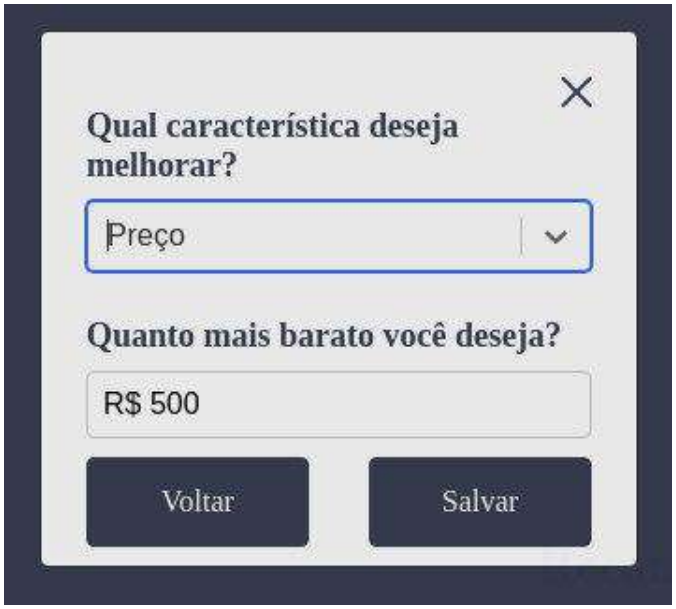

Figura 50 – Protótipo da simulação da crítica parte 3.

Dessa forma o usuário selecionará uma das características e atribuirá o valor de sua preferência. Em seguida, o perfil de busca do usuário será apropriado com base na crítica dessa característica em si, e assim serão geradas novas recomendações em seu *feed*.

No recomendador de aprendizado de máquina foi decidido a construção de uma filtragem colaborativa baseado em modelo, utilizando-se da técnica de fatoração matricial, com o algoritmo de aprendizado de máquina Fatoração matricial não negativa (NMF) disponibilizado pela biblioteca scikit-learn. Para simular a integração do sistema de recomendação como um serviço, foi construída uma API própria para o protótipo. Seu *deploy* foi feito por meio do serviço ECS do AWS, e tem um banco de dados próprio.Os dados de entrada do modelo foram retirados do banco de dados da Liva. Esses dados são referentes as ações de "Favoritar"e "Descartar"um imóvel feitas pelos usuários e foram copiados para o banco de dados da API.

Os dados dos usuários serão atualizados assim como modelo será treinado diariamente. A transformação da matriz que terá todos os usuário e todos os imóveis "avaliados", disponibilizará uma pontuação para cada propriedade. Dessa forma será possível recomendar para todos os usuários as cinco melhores propriedades geradas pelo algorítimo. Essas recomendações são apresentadas na página de detalhes do imóvel, assim como na solução proposta, e por meio de um componente carrossel do ReactJS chamado "*swiper*"como na Figura 51.

Figura 51 – Protótipo do sistema de recomendação.

Foram obtidos dados por meio da ferramenta Mixpanel em integração com o *front-end* da Liva. Esses dados referem-se a consultas realizadas pelo usuário dado um período de tempo. São registradas a frequência de acesso ao *feed*, visualização na pagina de detalhes, edição do perfil de busca, avaliação (favoritar ou descartar) de um imóvel, solicitação de mais imóveis a serem recomendados no *feed*, alteração de ação de favoritar ou descartar imóvel dentre outros. Além disso, foram separados todas as recomendações feitas para os usuários do banco de dados da Liva.

O protótipo ficou em produção do dia 26 ao dia 27 de novembro. Assim o resultado obtido foi comparado com os dois dias da semana anterior (19 e 20 de novembro). Abaixo (Tabela 5) se encontra a representação dos dados obtidos.

Tabela 5 – Quantidade de recomendações, imóveis favoritados e descartados.

Data do inicio - Data do fim	19/11/19 - 20/11/19	26/11/19 - 27/11/19
Quantidade de recomendações	715	537
Quantidade de imóveis favoritados	0	6
Quantidade de imóveis descartados	19	7

C.1.1 Analise dos resultados

Com os dados obtidos foi possível analisar a aprovação das recomendações, medindo o grau de imóveis favoritados, dividindo o numero de imóveis recomendados favoritados pelo número total de recomendações multiplicado por 100, além do grau de imoveis descartados com o número de imóveis descartados, como mostrado na tabela 6.

Tabela 6 – Grau de imóveis favoritados e descartados.

Data do inicio - Data do fim	19 /11/19 - 20/11/19	26/11/19 - 27/11/19
Grau de imóveis favoritados	0.0	1.11
Grau de imóveis descartados	2.65	1.30

Apesar do período pequeno da simulação, em virtude do encerramento do prazo para a finalização da primeira etapa (TCC1) e disponibilização para os avaliadores do trabalho, foi possível perceber uma melhora no grau de imoveis favoritados durante este breve período, com a aplicação e comparação dos dois recomendadores com abordagens diferentes, além de apresentar uma menor taxa de descartes.