

Faculdade de Ciência e Tecnologia de Montes Claros

Felipe Israel Corrêa

**SISTEMA AGREGADOR PARA ANÁLISE DE
CARACTERÍSTICAS DE IMÓVEIS COM RASTREAMENTO
WEB E APRENDIZADO DE MÁQUINA**

Montes Claros- MG
2018

Felipe Israel Corrêa

**SISTEMA AGREGADOR PARA ANÁLISE DE CARACTERÍSTICAS DE
IMÓVEIS COM RASTREAMENTO WEB E APRENDIZADO DE
MÁQUINA**

Monografia apresentada ao Curso de Engenharia da Computação, da Faculdade de Ciência e Tecnologia de Montes Claros, como parte dos requisitos para obtenção do título de Engenheiro da Computação.

Orientador: **PROF. DR. RENATO DOURADO MAIA.**

Montes Claros - MG
2018

FUNDAÇÃO EDUCACIONAL MONTES CLAROS
Faculdade de Ciência e Tecnologia de Montes Claros

Felipe Israel Corrêa

**SISTEMA AGREGADOR PARA ANÁLISE DE CARACTERÍSTICAS DE
IMÓVEIS COM RASTREAMENTO WEB E APRENDIZADO DE
MÁQUINA**

Esta monografia foi julgada adequada como parte dos requisitos para a obtenção do diploma de Engenheiro da Computação aprovada pela banca examinadora da Faculdade de Ciência e Tecnologia de Montes Claros.

Prof. Dr. Maurílio José Inácio
Coord. do Curso de Engenharia da Computação

Banca Examinadora

Prof. Dr. Renato Dourado Maia, FACIT / _____
(Orientador)

Prof. Examinador 1 _____

Prof. Examinador 2 _____

Montes Claros, 07 de junho de 2018.

Para Lavínia, amor maior.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela saúde.

Aos meus pais Aroldo e Isolina (*In memorian*) pelos contínuos esforços em me oferecerem uma educação satisfatória.

A minha esposa Laís pelo companheirismo e apoio em toda esta jornada.

Meus irmãos e sobrinhos pela amizade sempre presente.

Aos meus colegas e professores, em especial ao professor Renato Dourado, que como eu, acreditou neste projeto.

“Não se espante com a altura do voo. Quanto mais alto, mais longe do perigo. Quanto mais você se eleva, mais tempo há de reconhecer uma pane. É quando se está próximo do solo que se deve desconfiar. ”

Santos Dumont

RESUMO

A democratização do acesso a internet tem proporcionado uma geração de dados, sobretudo digitais, nunca antes vista na história da humanidade. Nunca se produziu tanta informação em tão pouco tempo. A partir desta perspectiva este estudo tem por objetivo coletar os dados sobre imóveis residenciais, da cidade de Montes Claros, adquiridos em sites de empresas imobiliárias e desenvolver um sistema que os agregue e gere observações relevantes sobre as características destes empreendimentos. A aquisição das informações foi feita por meio de um Rastreador Web e a análise dos dados ocorreu através da aplicação de Aprendizagem de Máquina no que tange a utilização de um algoritmo regressor e um de recomendação e busca. Para alcance do objetivo foram estudados os conceitos que englobam os processos de rastreamento e aprendizagem de máquina, verificadas as condições para desenvolvimento do trabalho e descritos os algoritmos de codificação. Como proposto, o sistema obteve êxito em predizer valores de um imóvel com base em suas características, tais como bairro, número de quartos, banheiros, vagas de estacionamento e tamanho da área, além de efetuar a recomendação e busca de imóveis. Todo ele foi confeccionado através da linguagem de programação *Python* no ambiente de desenvolvimento *Visual Studio Code*, em conjunto com as bibliotecas *Requests* e *BeautifulSoup* para o rastreador e as bibliotecas científicas e gráficas *Scikit-learn*, *Pandas*, *Numpy*, *Matplotlib* e *Bokeh* para aprendizagem de máquina.

Palavras-Chave: *Dados, Imóveis, Rastreador Web, Aprendizagem de Máquina, Python.*

ABSTRACT

The popularization of Internet access has provided a generation of data, especially digital data, never before seen in the history of mankind. Never has so much information been produced in such a short time. From this perspective, this study aims to collect data on real estate in the city of Montes Claros acquired on websites of real estate companies and develop a system that aggregates and generates relevant observations about the characteristics of these developments. The information acquisition was done through a Web Tracker and data analysis was take place through the Machine Learning application regarding the use of a regressor algorithm and a recommendation and search algorithm. In order to reach the objective, the concepts that comprise the processes of machine tracking and learning have been studied, the conditions for the development of the work were verified and the coding algorithms were described. As proposed the system is able to predict values of a property based on its characteristics, such as neighborhood, number of rooms, bathrooms, parking spaces and size of the area, in addition to recommending and searching for real estate. The system was written using the Python programming language in the Visual Studio Code development environment, together with the Requests and BeautifulSoup libraries for the tracker, and the Scikit-learn, Pandas, Numpy, Matplotlib and Bokeh scientific and graphic libraries for learning machine.

Keywords: *Data, Real Estate, Web Tracker, Machine Learning, Python.*

LISTA DE FIGURAS

FIGURA 1 - <i>Document Object Module</i>	17
FIGURA 2 - Código fonte e <i>tags</i> HTML.....	18
FIGURA 3 - Diagrama de fluxo de um rastreador <i>web</i>	18
FIGURA 4 - Fluxo de operações de um sistema de AM.....	23
FIGURA 5 - Representação intercepto e coeficiente de inclinação.....	24
FIGURA 6 - Representação do erro.....	25
FIGURA 7 - Linha de regressão da relação entre variáveis A e B.....	25
FIGURA 8 - Árvore de decisão.....	26
FIGURA 9 - Árvore de decisão e divisões no espaço.....	27
FIGURA 10 - Dissimilaridade calculada entre vetores a e b.....	31
FIGURA 11 - Produto escalar entre os vetores a e b.....	32
FIGURA 12 - Vetores a e b distantes 90°	32
FIGURA 13 - Vetores a e b com ângulo igual a 0°	32
FIGURA 14 - K-vizinhos mais próximos em classificação.....	33
FIGURA 15 - Diagrama do interpretador <i>Python</i>	35
FIGURA 16 - Exemplos de gráficos produzidos com biblioteca <i>Matplotlib</i>	39
FIGURA 17 - Exemplos de gráficos criados com a biblioteca <i>Bokeh</i>	40
FIGURA 18 - Fluxograma das aplicações.....	41
FIGURA 19 - Script rastreador <i>web</i>	42
FIGURA 20 - Exemplo de busca de <i>tags</i> para rastreador <i>web</i>	43
FIGURA 21 - Exemplo de definição das <i>tags</i> para busca dos dados.....	43
FIGURA 22 - Código para armazenamento dos dados.....	44
FIGURA 23 - Estrutura de criação da tabela no servidor <i>MySQL</i>	44
FIGURA 24 - Códigos para ajuste dos nomes dos bairros.....	46
FIGURA 25 - Porcentagem de valores nulos por coluna.....	46
FIGURA 26 - Código para imputação de valores da média.....	47
FIGURA 27 - Variáveis independentes no hiperplano.....	48
FIGURA 28 - Matriz de correlação entre as variáveis.....	48
FIGURA 29 - Processo de validação cruzada.....	49
FIGURA 30 - Separação do conjunto de dados entre treino e teste.....	50
FIGURA 31 - Método de avaliação do modelo linear.....	50
FIGURA 32 - Construção do modelo.....	51

FIGURA 33 - Avaliação e definição da profundidade da árvore de decisão.....	52
FIGURA 34 - Teste do modelo com 1 a 100 árvores.....	52
FIGURA 35 - Cálculo para floresta aleatória.....	53
FIGURA 36 - Erro global entre valores reais e preditos.....	55
FIGURA 37 - Média das características dos imóveis.....	56
FIGURA 38 - Predição de preço do imóvel.....	57
FIGURA 39 - Preços para apartamentos de 1 quarto.....	57
FIGURA 40 - Preços para apartamentos de 2 quartos.....	58
FIGURA 41 - Preços para apartamentos de 3 quartos.....	58
FIGURA 42 - Preços para apartamentos de 4 quartos.....	59
FIGURA 43 - Preços para apartamentos de 5 quartos.....	59
FIGURA 44 - Busca e armazenamento das coordenadas geográficas.....	61
FIGURA 45 - Leitura e atribuição dos conjuntos de dados.....	62
FIGURA 46 - Inserção das coordenadas geográficas.....	62
FIGURA 47 - Calculador de dissimilaridade.....	62
FIGURA 48 - Selecionar imóveis de acordo com o bairro.....	63
FIGURA 49 - Busca e recomendação de apartamentos.....	63
FIGURA 50 - Chamada dos métodos de busca e recomendação.....	63
FIGURA 51 - Resultado busca e recomendação para métrica euclidiana.....	64
FIGURA 52 - Resultado busca e recomendação para métrica do cosseno.....	65
FIGURA 53 - Itens recomendados e valor da dissimilaridade.....	66
FIGURA 54 - Implementação para cálculo da precisão e <i>recall</i>	67
FIGURA 55 - Implementação fórmula F1-score.....	68
FIGURA 56 - Imóveis apresentados calculados pela dissimilaridade euclidiana.....	68
FIGURA 57 - Imóveis apresentados calculados pela similaridade do cosseno.....	68
FIGURA 58 - Quantidade de imóveis por bairro.....	70
FIGURA 59 - Distribuição de imóveis pela cidade.....	71
FIGURA 60 - Quantidade de imóveis, por bairro, com 1 e 2 quartos.....	71
FIGURA 61 - Quantidade de imóveis, por bairro, com 3 e 4 quartos.....	72
FIGURA 62 - Quantidade de imóveis, por bairro, com 5 quartos.....	72
FIGURA 63 - Porcentagem de banheiros e vagas de garagem.....	73
FIGURA 64 – Estimativa de valor para o menor apartamento.....	73
FIGURA 65 - Estimativa de valor para o maior apartamento.....	74
FIGURA 66 - Estimativa de valor com área aumentada em 10m ²	74

FIGURA 67 - Diferença no valor do preço do imóvel a cada aumento da área.....	74
FIGURA 68 - Diferença de preços baseada no aumento do número de quartos.....	75
FIGURA 69 - Diferença de preços baseada no aumento do número de banheiros... ..	75
FIGURA 70 - Diferença de preços baseada no aumento do número de vagas.....	76

LISTA DE TABELAS

TABELA 1 - Característica dos imóveis.....	45
TABELA 2 - Conjunto de dados após pré-processamento.....	47
TABELA 3 - Resultado do cálculo das métricas do modelo linear.....	51
TABELA 4 - Resultado das métricas da árvore de decisão de acordo com sua profundidade.....	53
TABELA 5 - Resultado das métricas da floresta aleatória de acordo com quantidade de árvores e profundidade	54
TABELA 6 - Média das características dos apartamentos com base na quantidade de quartos.....	56
TABELA 7 - Resultado cálculos de eficiência das recomendações	69

SUMÁRIO

INTRODUÇÃO.....	14
CAPÍTULO 1 SISTEMA AGREGADOR PARA ANÁLISE DE CARACTERÍSTICAS DE IMÓVEIS COM RASTREAMENTO WEB E APRENDIZADO DE MÁQUINA.....	16
1.1 Rastreador Web.....	16
1.2 Fluxo de Busca.....	17
1.3 Aprendizado de Máquina.....	19
1.3.1 Aplicações.....	20
1.4 Tarefas de Aprendizado.....	20
1.5 Tipos de Aprendizagem.....	21
1.5.1 Aprendizagem Supervisionada.....	21
1.5.2 Aprendizagem Não Supervisionada.....	22
1.6 Fluxo de Funcionamento de um Sistema AM.....	22
1.7 Sistema de Regressão.....	23
1.7.1 Regressão Linear Simples e Múltipla.....	23
1.7.2 Árvore de Decisão e Floresta Aleatória.....	26
1.8 Sistema de Recomendação.....	29
1.9 Tipos Sistema de Recomendação.....	29
1.9.1 Filtragem Colaborativa.....	29
1.9.2 Recomendação Baseada em Conteúdo.....	30
1.10 Técnicas de Recomendação.....	30
1.10.1 Recomendação Baseada em Vizinhança.....	30
1.11 Distância Euclidiana.....	31
1.12 Similaridade do Cosseno.....	31
1.13 K-Vizinhos Mais Próximos.....	33
CAPÍTULO 2 MATERIAIS E MÉTODOS.....	34
2.1 Ambiente de Desenvolvimento.....	34
2.2 Linguagem <i>Python</i>	35
2.3 Bibliotecas.....	36
2.3.1 <i>Requests</i>	36
2.3.2 <i>BeautifulSoup</i>	37
2.3.3 <i>Googlemaps Geocoding</i>	37

2.3.4 <i>Mysql.connector</i>	37
2.3.5 <i>Scikit-learn</i>	37
2.3.6 <i>Numpy</i>	38
2.3.7 Pandas.....	38
2.3.8 <i>Matplotlib</i>	39
2.3.9 <i>Bokeh</i>	39
2.4 Questionário.....	40
CAPÍTULO 3 RESULTADOS: APRESENTAÇÃO, ANÁLISE E DISCUSSÃO.....	41
3.1 Rastreador <i>Web</i>	41
3.2 Regressor.....	45
3.3 Sistema de Recomendação e Busca.....	60
3.4 Análise Exploratória dos Dados.....	69
CAPÍTULO 4 APLICAÇÃO.....	77
CONSIDERAÇÕES FINAIS.....	78
REFERÊNCIAS.....	80

INTRODUÇÃO

O constante crescimento dos centros urbanos tem ocasionado, proporcionalmente, o aumento de demanda por uma moradia, tornando esse fato extremamente importante para empresas do ramo imobiliário. Contudo, esse mercado sofreu uma desaceleração causada pela recessão sofrida pelo país a partir de 2012, apesar de que, para 2018, tais empresas preveem um retorno de crescimento, consequência da baixa da taxa SELIC e novas regras definidas pelo Governo Federal em relação ao programa Minha Casa Minha Vida (DINO, 2017).

A oferta de imóveis muitas vezes é feita por meio de ferramentas tecnológicas, principalmente, por meio dos *sites* mantidos pelas empresas desse segmento. Gera-se, então, um imenso volume de dados digitais que podem chegar a milhares de *gigabytes* de informação, contendo as características de cada imóvel e a transação comercial disponibilizada, seja de compra, venda, aluguel, dentre outras. E esse aumento na produção de dados digitais é uma das características que se pode imputar à toda humanidade atual, pois, em um estudo realizado pela IDC, especialista em análise de dados, para a empresa EMC, apontou-se que o volume de dados digitais tem dobrado a cada dois anos, chegando a 4,4 trilhões de *gigabytes* de informação no ano de 2013 e que, em 2020, os dados criados e copiados chegarão à marca de 44 trilhões de *gigabytes* (EMC, 2014).

Por meio dessa perspectiva, o uso da Aprendizagem de Máquina, técnica de computação baseada em Inteligência Artificial, desenvolvida em meados da década de 70, passou a ser usada para tratativa de conjuntos de dados que são impossíveis de serem analisados mecânicamente por um humano, com o objetivo de aprendizado e predição de padrões (FACELI *et al.*, 2011).

O problema surge quando, ao buscar uma informação referente a um imóvel, o usuário se depara com inúmeras opções de *sites* e empresas, consequentemente, uma imensa quantidade de aspectos, em grande parte repetidos, e que não conseguem expor de forma clara as vantagens e desvantagens da aquisição do imóvel, principalmente, se o valor fornecido pela imobiliária é condizente com a região na qual o imóvel se encontra.

Como, então, obter informações adicionais por meio de todos esses dados e que ainda façam sentido para o usuário e possam auxiliá-lo na decisão de escolher

o melhor imóvel, ou seja, o que mais atende a seus objetivos em curto e longo prazo?

Este estudo recolheu os dados referentes aos imóveis e aplicou técnicas de Aprendizado de Máquina, tais como Regressão Linear, Árvores de Decisão e Sistemas de Recomendação, retirando aspectos que adicionam valor às informações que são disponibilizadas atualmente pelas empresas do setor imobiliário.

Por meio de um conjunto de dados de imóveis residenciais obtidos automaticamente em *sites* imobiliários da cidade de Montes Claros, este estudo teve como objetivo principal construir um sistema que pôde fornecer aos usuários não somente informações sobre os imóveis, mas também sobre o bairro e cidade. Para tal, foi necessário o atendimento aos seguintes objetivos específicos:

- a. criação de um rastreador *web*, para busca dos dados;
- b. armazenamento correto dos dados obtidos;
- c. análise e manipulação correta do conjunto de dados;
- d. criação de um sistema de regressão e recomendação consistentes;
- e. análise exploratória satisfatória dos dados.

Este estudo possibilitou retirar, dos dados disponibilizados, ideias que complementam o conhecimento prévio dos usuários, abrangendo não somente o imóvel, mas também o bairro e a cidade, de forma que auxiliem-no a tomar a melhor decisão, ou a decisão que achar pertinente para o momento; outro benefício é que o modelo foi criado por meio de ferramentas de análise e programação gratuitas.

No total, o trabalho foi composto por quatro capítulos: no Capítulo 1, o detalhamento dos conceitos teóricos referentes às técnicas de rastreamento *web* e aprendizagem de máquina; no Capítulo 2, detalham-se a metodologia e os materiais empregados para elaboração do sistema; no Capítulo 3, apresentam-se os resultados obtidos pelas ferramentas desenvolvidas, juntamente com a discussão desses. E, no Capítulo 4, é apontada uma indicação a qual área e tipos de usuário este trabalho é destinado.

CAPÍTULO 1 SISTEMA AGREGADOR PARA ANÁLISE DE CARACTERÍSTICAS DE IMÓVEIS COM RASTREAMENTO WEB E APRENDIZADO DE MÁQUINA

Neste capítulo serão discutidos os conceitos e teorias dos componentes que, juntos, foram utilizados para a criação do Sistema Agregador.

Farão parte os seguintes temas:

- A. Rastreador Web.
- B. Aprendizagem de Máquina.
- C. Sistema de Regressão.
- D. Sistema de Recomendação.
- E. Algoritmo K-vizinhos mais próximos.

1.1. Rastreador Web

A democratização do acesso à internet e as comodidades trazidas por essa plataforma ao mundo moderno provocaram o ingresso de vários setores aos serviços *online*. Atualmente, são mais de 4 bilhões de usuários conectados, o que corresponde a 53% da população mundial, segundo dados do site *Internet World Stats* (2017), navegando em mais de 966 milhões de páginas (SMITH, 2017).

Páginas de compra, venda, entretenimento são, hoje, indispensáveis para o público em geral. Frente a esse crescimento e de ser humanamente impossível consultar todos os *sites* manualmente – mesmo aqueles que sejam específicos para um usuário ou grupo de usuários – foi desenvolvida a técnica conhecida como rastreamento *web* que, segundo Bernard (2017), consiste em coletar automaticamente dados de páginas *web*, extrair informações específicas e armazená-los para uso posterior. O site *Data for thoughts* (2014) classifica essa prática como um subgrupo de uma técnica conhecida como *Data Scraping* (Raspagem de Dados), em que se procuram extrair informações legíveis de um *software* para outro.

Essa técnica possui alguns outros nomes, alguns já em desuso, que são *screen scraping*, *web harvesting* e similares (MITCHELL, 2015).

Normalmente, esses dados são apresentados de forma não estruturada, o que impede de serem utilizados em bases que possam ser interpretadas por algoritmos, portanto, faz-se necessário transformá-los em dados estruturados e que

sejam semanticamente compreensíveis e interpretáveis pela máquina (BARREIRA, 2014).

1.2. Fluxo de Busca

O método mais utilizado para busca das informações é o uso das *tags* disponíveis nos códigos-fonte das páginas HTML ou XML (HEMENWAY; CALISHAIN, 2003). Esse método tira proveito da forma semiestruturada das páginas HTML, por serem parecidas como uma árvore (BARREIRA, 2014).

Ainda, segundo Barreira (2014), a estrutura das *tags* de uma página HTML representa diferentes níveis, formando uma estrutura hierárquica, com isso, utiliza-se o *Document Object Model* (DOM), que define a estrutura lógica de um documento e a forma como ele deve ser acessado e manipulado W3C (2004), conforme mostrado na Figura 1.

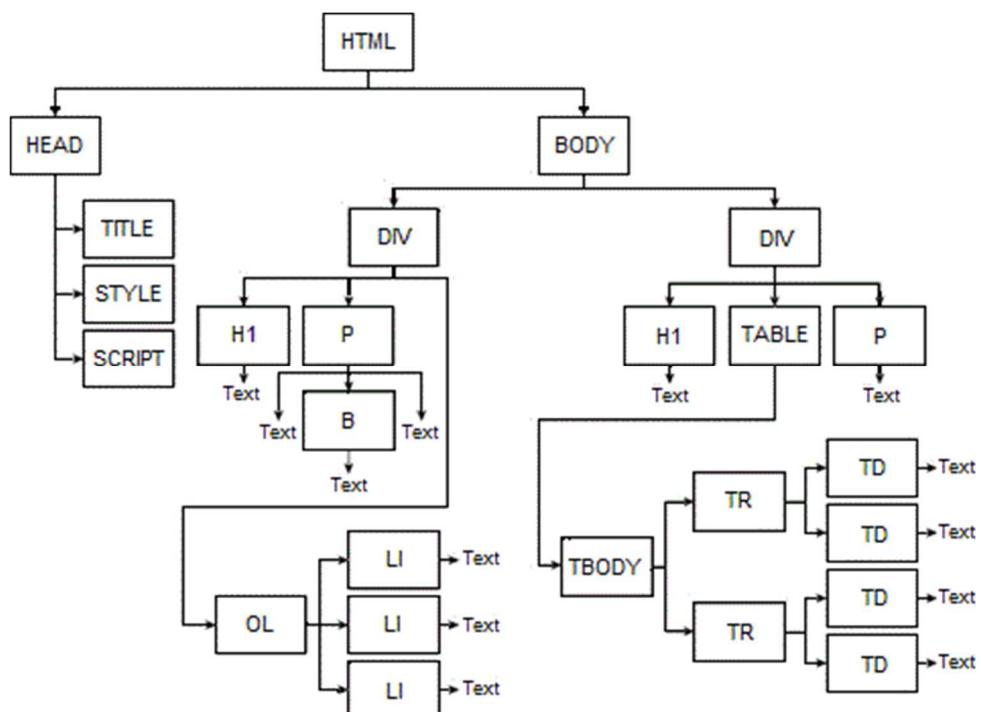


FIGURA 1: *Document Object Model*.
Fonte: Barreira (2004, p. 8 - Adaptada)

A Figura 2 detalha o código-fonte de uma página e as *tags* referentes a cada conteúdo:

- **html** → <html><head>...</head><body>...</body></html>
 - **head** → <head><title>A Useful Page<title></head>
 - **title** → <title>A Useful Page</title>
 - **body** → <body><h1>An Interesting Title</h1><div>Lorem ip...</div></body>
 - **h1** → <h1>An Interesting Title</h1>
 - **div** → <div>Lorem Ipsum dolor...</div>

FIGURA 2: Código fonte e *tags* HTML.
Fonte: MITCHELL (2015, p. 8)

Para se buscar um determinado texto, a Figura 3 apresenta o fluxo utilizado pelo rastreador.

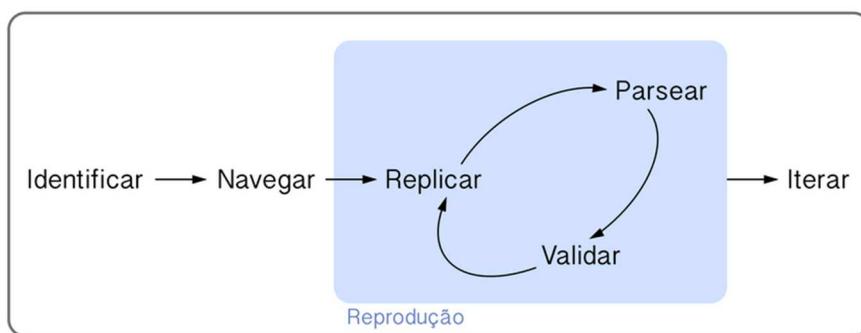


FIGURA 3: Diagrama de fluxo de um rastreador web.
Fonte: Lente (2018)

Cada um desses verbos indica uma das fases do processo e a caixa azulada indica uma iteração que será feita até que a coleta funcione (LENTE, 2018).

Abaixo são detalhados, segundo Lente (2018), os processos executados por cada verbo:

- a. Identificar: Processo em que se identifica a informação a ser coletada, bem como entender a estrutura da página web para traçar a forma como o dado será extraído;
- b. Navegar: Neste processo, utiliza-se a ferramenta de desenvolvedor do próprio navegador, a fim de encontrar as fontes dos dados que se desejam coletar, tais como uma *tag*, uma função *JavaScript*, ou outro tipo de marcador;
- c. Replicar: Processo que serve para executar várias solicitações, para obtenção da informação desejada, ou seja, as solicitações são replicadas; em suma é a solicitação de acesso à página;

- d. Parsear: Palavra originada do verbo, em inglês, *to parse*, pode significar estudar ou analisar e, no contexto de um rastreador, significa extrair os dados de um arquivo HTML ou XML.
- e. Validar: A validação é feita reproduzindo os procedimentos acima para outras páginas com o objetivo de verificar se a informação está sendo extraída de forma correta. Caso sejam encontrados erros é necessário voltar ao passo de replicação, rever o comportamento da página e parsear os dados corretamente.
- f. Iterar: Consiste em, finalizados os procedimentos acima com sucesso, colocar o rastreador para busca contínua dos dados desejados. Na maioria das vezes cria-se uma função que recebe uma série de *links* e aplica o mesmo procedimento em cada um deles de forma automática.

Todo esse fluxo, em geral, é feito por meio de uma biblioteca ou uma *Application Programming Interface* (API), traduzida como Interface de Programação de Aplicações, dentro de uma determinada linguagem de programação (MITCHELL, 2015).

1.3. Aprendizado de Máquina

Em todo o histórico da evolução das espécies, uma das características essenciais a qualquer grupo é o aprendizado. Por meio desse processo, as espécies foram capazes de se adaptar, criar laços e se estabelecer em um meio. Aprender, segundo o dicionário Ferreira (2005), significa tornar-se capaz de algo graças a estudo, observação e experiência.

Tendo em vista esses pontos, Fernandes (2003) estabelece que um sistema aprende, caso esse seja hábil em identificar a informação por ele já reconhecida e processá-la logo se torne disponível.

Assim, Faceli *et al* (2011) constatam que uma máquina resolve problemas por meio de um algoritmo que define os passos de sua resolução, portanto, o Aprendizado de Máquina (AM) tenta criar algoritmos, baseados no processo de aprendizagem a partir de uma experiência passada ou reconhecimento de padrões, que sejam capazes de organizar um novo conhecimento, atingindo o que é chamado de comportamento inteligente.

Várias são as definições de AM encontradas na literatura; em uma delas, Mueller e Massaron (2016) o consideram uma subárea da Inteligência Artificial, tendo como base teorias de matemática e estatística.

Para outros, como Raschka e Mirjalili (2017), o AM oferece uma alternativa de melhorar a predição de modelos e tomar decisões baseadas nos dados, por meio do conhecimento obtido por meio desses.

1.3.1. Aplicações

Alpaydin (2010), descreve alguns exemplos de aplicações de aprendizado de máquina mais utilizadas:

- a. Aprendizado por associações: consiste em aprender a condição de um elemento a outro. Exemplo: um consumidor ao comprar um produto X, tem mais propensão a comprar também o produto Y e não o produto Z. Tem como fórmula matemática de probabilidade condicional $P(X|Y)$, em que Y é o produto que deseja-se condicionar a X. E X o conjunto de produtos que o cliente já comprou.
- b. Classificação: como o nome sugere, consiste em classificar um dado novo em uma determinada classe de acordo com a classificação dada para dados analisados anteriormente.
- c. Regressão: consiste em estimar um valor de saída associado ao dado, baseado em suas características inseridas.

1.4. Tarefas de Aprendizado

O paradigma de aprendizado é um dos critérios utilizados para definir o tipo de tarefa desenvolvido pelo AM, sendo, de acordo com esse critério, divididas em tarefas preditivas ou descritivas (FACELI *et al*, 2011).

Alpaydin (2010) define uma tarefa preditiva quando essa é capaz de estimar dados futuros, com base nos dados passados; e do tipo descritiva, quando da possibilidade de aprender e/ou reconhecer padrões. Existe ainda a possibilidade de mesclar esses dois tipos de abordagem.

1.5. Tipos de Aprendizagem

Para Russell e Norvig (2013), um sistema estará aprendendo se for capaz de aprimorar suas estimativas futuras após realizar observações sobre o mundo. Os autores elencam três principais técnicas de aprendizagem, que são:

- a. Aprendizagem supervisionada;
- b. Aprendizagem não supervisionada;
- c. Aprendizagem por reforço.

Neste estudo, serão discutidas somente as técnicas não supervisionada e supervisionada, com ênfase maior nessa última por se tratar da técnica utilizada para criação do Sistema Agregador.

1.5.1. Aprendizagem Supervisionada

Esse tipo de aprendizagem é baseado em um conjunto de dados, cujas saídas ou algum outro tipo de representação do comportamento que o sistema deve apresentar já são previamente conhecidos (CASTRO; FERRARI, 2016). Em linhas gerais, o sistema observa amostras de dados em pares de entrada e saída e aprende uma função que faz a rota entre eles.

A técnica supervisionada é descrita por Russell e Norvig (2013) da seguinte maneira:

- a. é fornecido previamente um conjunto de treinamento com N pares de entrada e saída $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$, onde a geração de cada y é fornecida por uma função não conhecida $y = f(x)$.
- b. com essa relação observada é objetivo do sistema gerar uma outra função h que se assemelhe à função verdadeira f .
- c. sendo a função h uma hipótese, a aprendizagem se torna uma busca por hipóteses possíveis que terá uma boa eficiência mesmo caso um novo dado seja inserido.
- d. para que a eficiência da hipótese seja observada, devem-se fornecer conjuntos de testes e conjuntos de treinamentos, sendo esses diferentes.
- e. a hipótese será eficiente se prever de forma correta o valor de y para novos dados.

- f. em geral, y sendo uma saída de valores finitos, como ensolarado, nublado ou chuvoso, a aprendizagem recebe o nome de classificação, caso y seja um número, como o valor da temperatura em determinada hora do dia, a aprendizagem é chamada de regressão.

1.5.2. Aprendizagem Não Supervisionada

Nesse tipo de aprendizagem, a saída dos dados é desconhecida, portanto, o objetivo é identificar padrões (ALPAYDIN, 2010), ou seja, o algoritmo deve categorizar ou rotular os dados.

O sistema é capaz de aprender padrões com dados de entrada, mesmo não sendo fornecido um valor correspondente de saída. Tem como tarefa mais recorrente o agrupamento, ou seja, reconhecer grupos com os dados de entrada potencialmente úteis (RUSSELL; NORVIG, 2013).

1.6. Fluxo de Funcionamento de um Sistema de AM

Em geral, todos os sistemas baseados em AM obedecem a um fluxo de operações, divididos em 4 partes que são:

- a. Pré-processamento dos dados: os dados, muitas vezes, podem conter valores incorretos, inconsistentes, duplicados, apresentar diferentes características, formatos e dimensões, o que atrapalha a interpretação desses pela máquina. O pré-processamento visa melhorar qualidade desses dados, minimizando ou eliminando os problemas citados (FACELI *et al*; 2011). Nessa etapa também, o conjunto de dados é dividido, aleatoriamente, entre elementos de treino e teste, objetivando evitar que a máquina não tenha um bom desempenho somente com o conjunto de treino (RASCHKA; MIRJALILI, 2017);
- b. Definição do modelo e aprendizagem: nessa etapa se define o tipo de modelo, ou seja, o algoritmo de AM, com base na tarefa a ser resolvida, e se efetua a aprendizagem da máquina, com o conjunto de dados de treino (RASCHKA; MIRJALILI, 2017);
- c. Avaliação: após o treinamento, para verificar a performance do treino, o modelo recebe os dados de teste. Caso essa performance seja

satisfatória, o modelo está apto para receber dados novos (RASCHKA; MIRJALILI, 2017);

- d. Estimação: nessa última etapa o modelo consiste em obter os resultados de valores novos estimados pelo modelo (RASCHKA; MIRJALILI, 2017).

A Figura 4, apresenta o fluxo descrito acima.

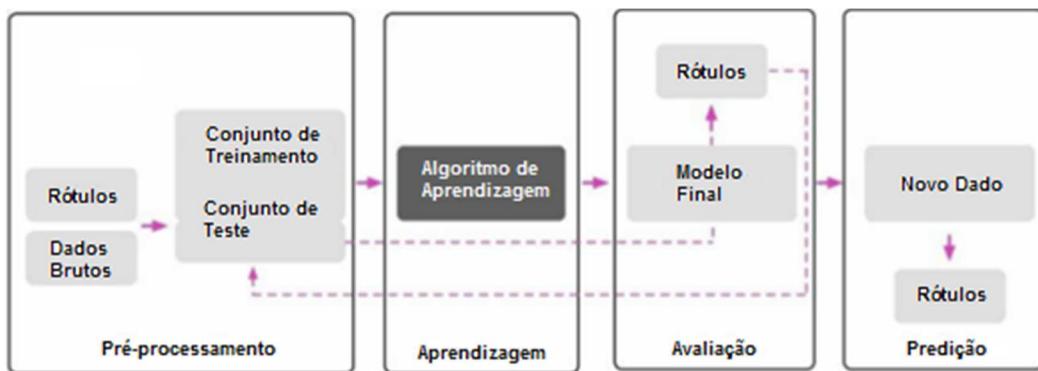


FIGURA 4: Fluxo de operações de um sistema de AM.

Fonte: Raschka e Mirjalili (2017, p. 53 – Adaptada)

1.7. Sistema de Regressão

Bishop (2006) afirma que modelos de regressão têm por objetivo estimar o valor de uma ou mais variáveis alvo contínuas, por meio de um vetor de entrada x D -dimensional.

Existem inúmeros algoritmos de AM, alicerçados em conceitos matemáticos, usados para o propósito de regressão, tais como Regressão Linear, Árvores de Decisão, Regressão Polinomial e Máquina de Vetores de Suporte (*SVR*, em inglês), mas, neste estudo, serão analisados somente os algoritmos de Regressão Linear Simples e Múltipla, além de Árvore de Decisão e Floresta Aleatória, pois esses modelos foram utilizados para criar o Sistema de Regressão, do Sistema Agregador.

1.7.1. Regressão Linear Simples e Múltipla

Segundo Castro e Ferrari (2016), modelos de regressão linear simples são capazes de estabelecer a relação entre uma variável dependente e uma independente, por meio da determinação de uma equação de linha reta que seja capaz de representar essa relação.

A essa linha dá-se o nome de linha de regressão e a equação é denominada equação de regressão.

A equação de regressão escrita por Lattin *et al* (2011) é dada pela Equação 1.

$$1. \quad Y = \alpha + \beta x$$

em que α é chamado de intercepto, e β coeficiente de inclinação, parâmetros esses a serem estimados.

O coeficiente de inclinação representa a variação média de Y , para o aumento de uma unidade da variável x ; e o intercepto o ponto do eixo das ordenadas cortado pela reta (BUSSAB; MORETTIN, 2010). A Figura 5 apresenta esses parâmetros.

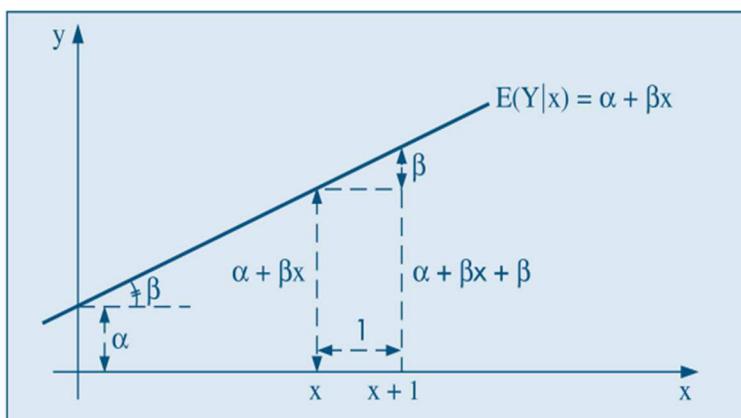


FIGURA 5: Representação intercepto e coeficiente de inclinação.
Fonte: Bussab e Morettin (2010, p. 450)

Pelo fato de essa relação não ser perfeitamente exata, à fórmula adiciona-se um erro, que reflete um ruído, ou seja, a diferença entre o valor real da variável dependente e o valor estimado (LATTIN *et al*; 2011). Por fim a equação final é dada pela Equação 2.

$$2. \quad Y = \alpha + \beta x + \varepsilon$$

O erro também é conhecido como *offset* e é representado por $|\hat{y} - y|$, em que \hat{y} corresponde ao valor real e y , ao valor estimado (RASCHKA; MIRJALILI, 2017). Conforme a Figura 6, eles são representados pelas linhas verticais.

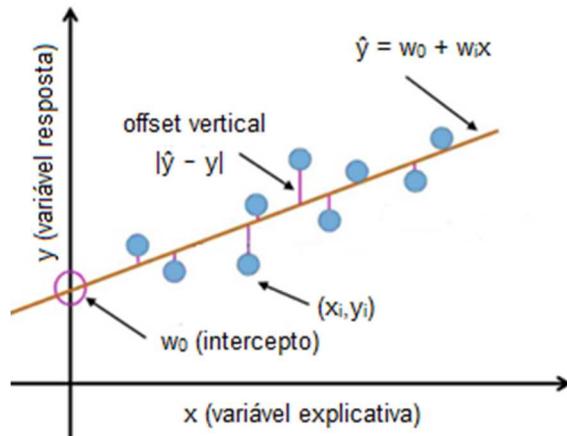


FIGURA 6: Representação do erro.
Fonte: Raschka e Mirjalili (2017, p. 451 – Adaptada)

Segundo Faceli *et al* (2011), esse erro, também conhecido como erro da hipótese f , é medido, geralmente, por meio de duas métricas conhecidas como erro quadrático médio (MSE - *mean squared error*) e distância absoluta média (MAD - *mean absolute distance*), dadas, respectivamente, pelas Equações 3 e 4.

$$3. \text{ } MSE(f) = \frac{1}{n} \sum_{i=1}^n (\hat{y} - y)^2$$

$$4. \text{ } MAD(f) = \frac{1}{n} \sum_{i=1}^n |\hat{y} - y|$$

Ambas as medidas são valores não negativos e quanto menores seus resultados, melhor o modelo de regressão (FACELI *et al*; 2011).

A Figura 7, apresenta a linha de regressão, dada pela relação entre a variável independente A e a dependente B.

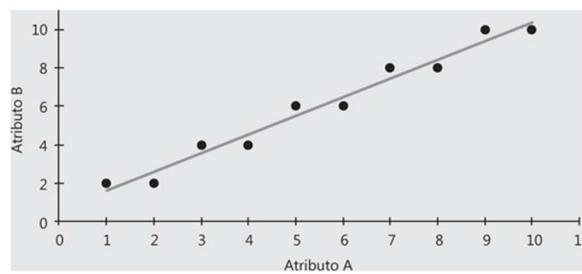


FIGURA 7: Linha de regressão da relação entre variáveis A e B.
Fonte: Castro e Ferrari (2016, p. 160)

A regressão linear múltipla se baseia nos mesmos conceitos, métricas e técnicas de avaliação da regressão linear simples. Ela é a generalização do modelo

linear para mais de uma variável independente e pode ser definida pela Equação 5 (RASCHKA; MIRJALILI, 2017):

$$5. \quad Y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \varepsilon$$

1.7.2. Árvore de Decisão e Floresta Aleatória

Alpaydin (2010) define uma árvore de decisão como uma estrutura hierárquica, composta por nós de decisão e folhas, conforme Figura 8, baseada na estratégia de conquistar e dividir. Isso significa que um problema complexo é dividido em problemas mais simples e recursivamente se aplica a mesma estratégia, começando da raiz até ao ponto em que a folha é escolhida, ou seja, na folha a saída é definida por um valor.

Tem-se na parte mais elevada da árvore o nó chamado raiz e os caminhos desse nó raiz até a um nó folha corresponde a uma regra de classificação ou regressão (CASTRO; FERRARI, 2016).

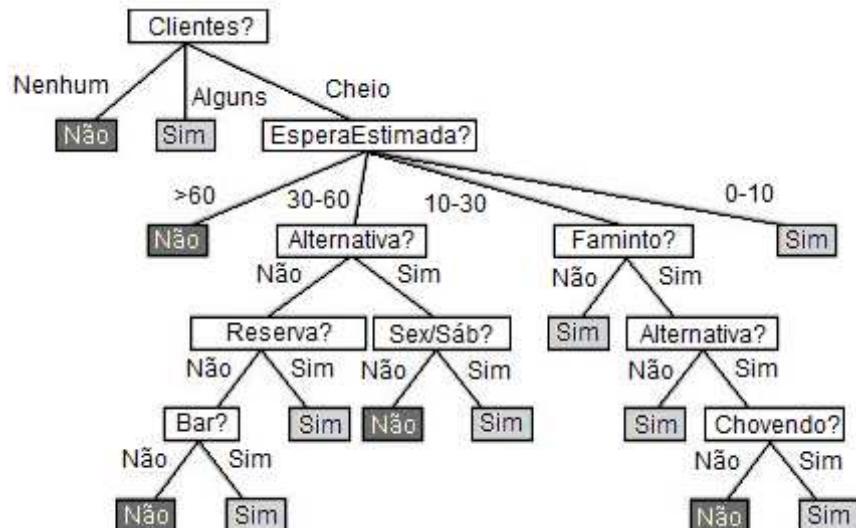


FIGURA 8: Árvore de decisão.
Fonte: Russell e Norvig (2013, p. 812 - Adaptada)

Na Figura 8, Russell e Norvig (2013) apresentam uma árvore que decide a espera ou não de uma mesa em um determinado restaurante e conta com os seguintes atributos:

- Alternativa: se existe um restaurante alternativo por perto.

- b. Bar: se existe área de bar confortável no restaurante para espera.
- c. Sex/Sáb: se funciona às sextas e sábados.
- d. Faminto: se cliente com fome.
- e. Clientes: quantidade de pessoas no restaurante.
- f. Chovendo: se chove na área externa do restaurante.
- g. Reserva: se cliente fez reserva.
- h. EsperaEstimada: tempo de espera estimada pelo gerente.

Segundo Faceli *et al* (2011), em cada nó de decisão existe um teste condicional referente aos valores de atributo e, de acordo com o resultado, um dos ramos é escolhido. Na maioria das vezes, o teste efetua a comparação do valor de atributo a uma constante e após o resultado encaminha o objeto para um próximo nó de decisão ou um nó folha (CASTRO; FERRARI, 2016).

Ainda, segundo os autores, Castro e Ferrari (2016), os nós folhas fornecem uma classificação ou valor, nos casos de regressão, a todo objeto que atinge a folha.

Em problemas de regressão a média dos valores contidos na folha é utilizada para determinar o valor do objeto e nos casos de classificação se utiliza a moda das classes contidas também na folha (FACELI *et al*; 2011).

A Figura 9, apresenta uma árvore de decisão e a divisão no espaço referente aos atributos x_1 e x_2 e cada nó corresponde a uma região nesse espaço.

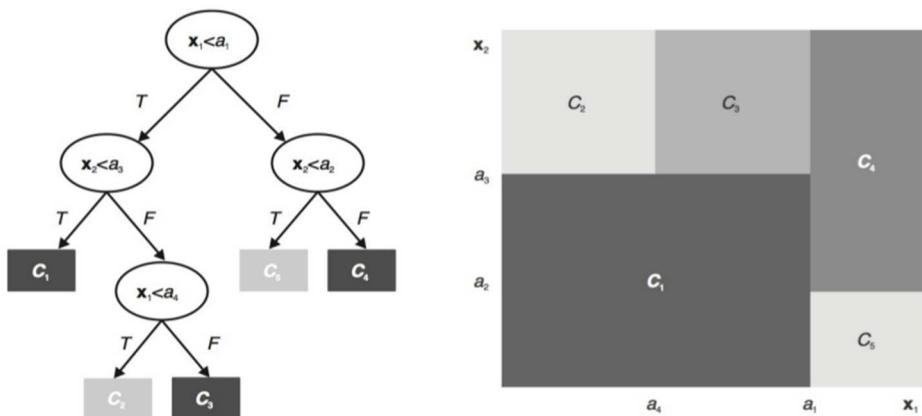


FIGURA 9: Árvore de decisão e divisões no espaço.
Fonte: FACELI *et al* (2011, p. 84)

Ainda, segundo Faceli *et al* (2011), por a árvore abranger todo espaço em instâncias, ela se torna capaz de efetuar predições para qualquer exemplo de entrada e cada uma das divisões é estipulada, em geral, por meio da métrica de

Redução do Desvio Padrão (*Standard Deviation Reduction* ou SDR), por meio dos seguintes processos efetuados em um conjunto D com n amostras:

- Calcula-se a variância da variável y , dependente, pela Equação 6.

$$6. \text{ } sd(D, y) = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y})^2}$$

- Divide-se o conjunto D em dois subconjuntos D_1 e D_2 com cardinalidades n_1 e n_2 .
- Como a variância de y será sempre igual ou menor à sua variância antes da divisão, estima-se a redução dessa pela Equação 7.

$$7. \text{ } SDR = sd(D, y) - \frac{n_1}{n} \times sd(D_1, y) - \frac{n_2}{n} \times sd(D_2, y)$$

- É calculado o valor da redução da variância para cada um dos atributos e em cada teste possível no valor do atributo.
- O teste que possuir a maior redução na variância é escolhido como nó de decisão.

Já o cálculo do valor de predição é dado pela média dos elementos pertencentes ao espaço ao qual a folha pertence, aplicando a Equação 8 (RASCHKA; MIRJALILI, 2017).

$$8. \text{ } \hat{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

em que n é a quantidade de elementos do conjunto, e y os valores contidos no conjunto.

Uma floresta aleatória é o conjunto de n árvores de decisão, sendo n definido pelo usuário, tendo o valor de predição calculado a partir da média de todos os valores preditos por cada árvore (RASCHKA; MIRJALILI, 2017).

1.8. Sistema de Recomendação

Ricci *et al* (2015) definem um sistema de recomendação como um conjunto de ferramentas e técnicas capaz de fornecer sugestões de itens, os quais sejam o mais parecido possível ao interesse de um usuário em particular, seja qual música ouvir, qual produto comprar, dentre outros. Item nesse contexto é um termo genérico que se refere ao produto, serviço, ou outros, fornecido pelo sistema.

Seu princípio básico consiste em observar as dependências significativas entre o item e o usuário, ou seja, se um usuário se interessa por um documentário, muito provavelmente esse se interessaria mais por outro documentário ou um programa educacional do que por um filme de ação (AGGARWAL, 2016).

1.9. Tipos de Sistema de Recomendação

Um sistema de recomendação possui vários tipos de abordagem, e neste estudo serão detalhados os funcionamentos dos tipos filtragem colaborativa e baseada em conteúdo.

1.9.1. Filtragem Colaborativa

Composta por duas técnicas, que são a baseada em usuário e a em item, a filtragem colaborativa visa efetuar as recomendações de itens para determinados usuários fundamentadas no histórico de usuários que compartilhem gostos similares (GORAKALA, 2016).

Na filtragem colaborativa baseada em usuário, após a avaliação de determinados itens por um usuário A, a ideia básica é localizar as avaliações semelhantes aos mesmos itens avaliados por A, por um grupo de usuários B e recomendar para A itens desconhecidos que B avaliou (AGGARWAL, 2016).

Segundo Segaran (2007), na filtragem colaborativa baseada em item, primeiramente são observadas as similaridades entre todos os itens. A recomendação ocorre com os itens que, após consideradas as avaliações de um determinado usuário, apresentam maior similaridade entre si.

1.9.2. Recomendação Baseada em Conteúdo

Gorakala (2016) estabelece que nessa abordagem, o conteúdo dos itens é utilizado para efetuar a recomendação. A similaridade entre eles é calculada por meio da característica associada a cada item mediante comparação entre todos os itens. Em linhas gerais, nesse tipo de recomendação, os itens têm suas características comparadas e se recomenda ao usuário os que forem mais similares ao item que esse já avaliou.

Neste estudo essa foi a abordagem utilizada para criação do Sistema de Recomendação e Busca.

1.10. Técnicas de Recomendação

Existem inúmeras técnicas utilizadas para criação de um sistema de recomendação, tais como a técnica baseada em modelo, em restrições e em vizinhança. Essa última foi a técnica escolhida para criação do sistema de recomendação e busca, portanto, essa será abordada e suas características detalhadas.

1.10.1. Recomendação Baseada em Vizinhança

Essa técnica é uma das mais antigas utilizadas e se apoia no princípio da similaridade entre pares de elementos, sejam eles, preferências e avaliações de usuários, itens, dentre outros (RICCI *et al*, 2015). Os pares de itens ou avaliações, preferências, ou qualquer outra fonte de dados, conforme Gorakala (2016), são considerados vetores e um cálculo de dimissimilaridade é aplicado a cada um deles para determinar o quão próximos estão.

Para o cálculo da similaridade, é comum o uso das dissimilaridades euclidiana, Minkowski, similaridade do cosseno, de Jaccard, coeficiente de correlação Pearson, e outras métricas. Sendo que, neste estudo, serão expostos os aspectos das dissimilaridade euclidiana e similaridade do cosseno, por essas terem sido utilizadas.

1.11. Dissimilaridade Euclidiana

A dissimilaridade euclidiana calcula a dissimilaridade entre dois pontos, no espaço denominado, espaço de entrada (FACELI *et. al*; 2011). É definida pela Equação 9.

$$9. d(x_i, y_j) = \sqrt{\sum_{l=1}^d |x_i^l - y_j^l|^2}$$

É uma das medidas mais populares para cálculo de dissimilaridade e conforme a Figura 10, pode-se ver a dissimilaridade calculada entre dois pontos a e b.

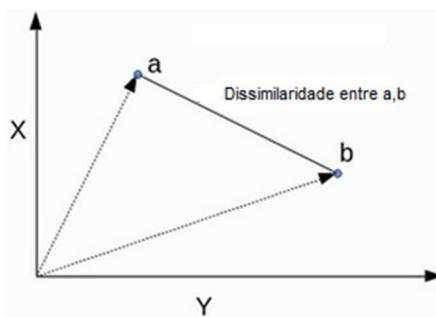


FIGURA 10: Dissimilaridade calculada entre pontos a e b.
Fonte: Gorakala (2016, p. 70 – Adaptada)

1.12. Similaridade do Cosseno

Segundo Gorakala (2016), a similaridade do cosseno, conhecida também como medida de separação angular, efetua o cálculo da similaridade entre os vetores a e b por meio do ângulo existente entre eles. É dada pela Equação 10.

$$10. \cos(\theta) = \frac{a \cdot b}{\|a\| \|b\|}$$

A Figura 11, apresenta o espaço vetorial do produto escalar entre os vetores a e b.

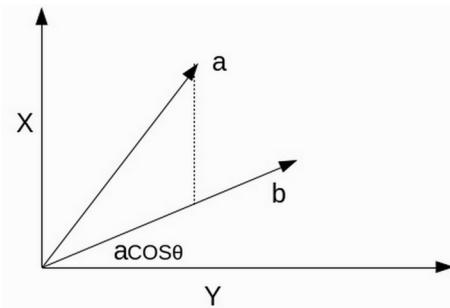


FIGURA 11: Produto escalar entre os vetores a e b.
Fonte: Gorakala (2016, p. 71)

Caso o ângulo entre os dois vetores seja 90° , o resultado de todos os pontos entre eles será 0, pois $\cos 90 = 0$, significando que os elementos estão muito distantes entre si. O contrário ocorre quando o ângulo entre os elementos é reduzido, ou seja, com a redução desse ângulo eles se tornam mais similares (GORAKALA, 2016). As Figuras 12 e 13, a seguir, exemplificam as situações descritas.

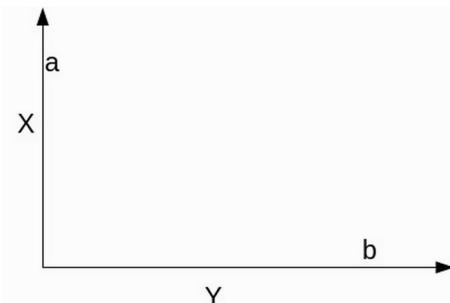


FIGURA 12: Vetores a e b distantes 90° .
Fonte: Gorakala (2016, p. 71)

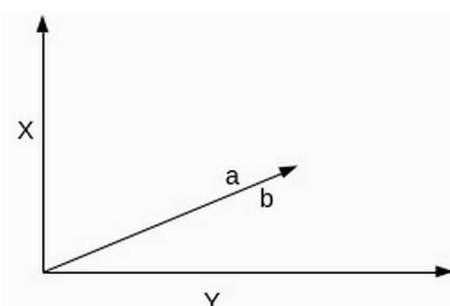


FIGURA 13: Vetores a e b com ângulo igual a 0° .
Fonte: Gorakala (2016, p. 72)

A aplicação de algoritmos de AM para sistemas de recomendação, são em sua maioria baseados em conceitos de agrupamento e este estudo fez uso do algoritmo denominado k-vizinhos mais próximos (*K-Nearest Neighbors*), o qual será detalhado.

1.13. K-Vizinhos Mais Próximos

Esse algoritmo consiste em observar k pontos vizinhos mais próximos em relação a um ponto x . Portanto, para implementação, é importante a medida de dissimilaridade entre pontos pares de observação (SMOLA; VISHWANATHAN, 2010).

É também conhecido como algoritmo de aprendizagem preguiçosa (*lazy learning algorithm*), pois, conforme Raschka e Mirjalili (2017), ele memoriza os dados do conjunto de treinamento em vez de aprender uma função discriminante que os defina. Por esse motivo, é categorizado como modelo não paramétrico, com abordagem baseada em exemplos ou memória (RUSSELL; NORVIG, 2013).

O funcionamento desse algoritmo pode ser definido pelas seguintes etapas:

- Define-se a quantidade de k vizinhos, sendo que, Faceli *et al.* (2011) estipulam um número pequeno e ímpar para evitar empates.
- Para cada ponto do conjunto de dados, calcula-se a dissimilaridade ao k -ésimo vizinho mais próximo (LATTIN *et al.*; 2011).
- Identifica-se os k vizinhos mais próximos, por meio do resultado da dissimilaridade, ou seja, quanto menor a dissimilaridade mais próximos os elementos estão (GORAKALA, 2016).

A seguir, a Figura 14 apresenta o uso do algoritmo em problemas de classificação, onde a classe escolhida é a que possui mais representantes.

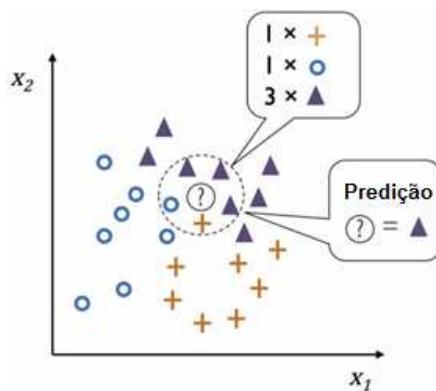


FIGURA 14: K-vizinhos mais próximos em classificação.
Fonte: Raschka e Mirjalili (2017, p. 174)

Segundo Russell e Norvig (2013), para problemas de regressão pode-se tirar a média ou mediana dos k -vizinhos.

CAPÍTULO 2 MATERIAIS E MÉTODOS

Neste capítulo serão discutidos e detalhados as tecnologias e métodos para criação do Sistema Agregador.

A primeira parte deste estudo consistiu em analisar e escolher as ferramentas mais apropriadas para elaboração do sistema. Após feita a análise, definiu-se pela codificação em linguagem de programação *Python*, banco de dados *MySQL* e plataforma de desenvolvimento *VSCode*.

Este estudo possuiu aspectos quantitativo e qualitativo, pois fará uso tanto de dados referentes às características dos apartamentos, que foram adquiridos de forma automática, quanto de dados adquiridos por meio de pesquisa.

Foi uma pesquisa aplicada, pois, segundo Gerhardt e Silveira (2009), esse tipo de pesquisa gera conhecimentos para aplicação prática, o que foi justamente a proposta deste estudo, aplicar as técnicas estudadas para análise de dados.

Além de aplicada, é uma pesquisa explicativa, pois tem-se o interesse de explicar os resultados por meio dos procedimentos efetuados.

Metodologicamente o estudo obedeceu à seguinte sequência:

- a. escolha da linguagem de programação e bibliotecas;
- b. criação do rastreador *web* e obtenção dos dados;
- c. armazenamento dos dados obtidos em banco de dados;
- d. criação dos modelos de aprendizado de máquina, tanto para regressão, como para busca e recomendação;
- e. análise exploratória dos dados.

A seguir são detalhados todos os materiais utilizados.

2.1. Ambiente de Desenvolvimento

O ambiente de desenvolvimento escolhido para este estudo foi o *Visual Studio Code*, mais conhecido como *VSCode* da empresa *Microsoft*, pois possui uma grande comunidade de usuários na internet, além de várias ferramentas que aumentam a produtividade do programador, como atalhos para identação, comentários, execução do código e versionamento.

2.2. Linguagem *Python*

A linguagem *Python* foi escolhida, pela sua versatilidade e grande quantidade de bibliotecas para uso científico, principalmente para manipulação de dados e aplicações de aprendizado de máquina. É por meio dessa linguagem que os dados adquiridos serão manipulados, analisados e ao final, apresentados os resultados.

A linguagem *Python* foi desenvolvida, a partir da linguagem ABC, em 1990 por Guido Van Rossum no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda. Foi criada como o objetivo inicial de suprir as necessidades de usuários como físicos e engenheiros (BORGES, 2010).

Segundo Cruz (2015), *Python* é uma linguagem de alto nível interpretada. Possui tipagem forte e dinâmica, gerenciamento de memória automático e escopo léxico. Ela também é capaz de suportar inúmeros paradigmas de programação, tais como imperativo, orientado a objetos e funcional.

É uma linguagem de código aberto, (licença *General Public License – GLP*), e por ter menos restrições, pode ser usada em produtos proprietários. Sua especificação e controle de versões é mantida pela *Python Software Foundation – PSF* (BORGES, 2010). O processo de interpretação é exibido na Figura 15.

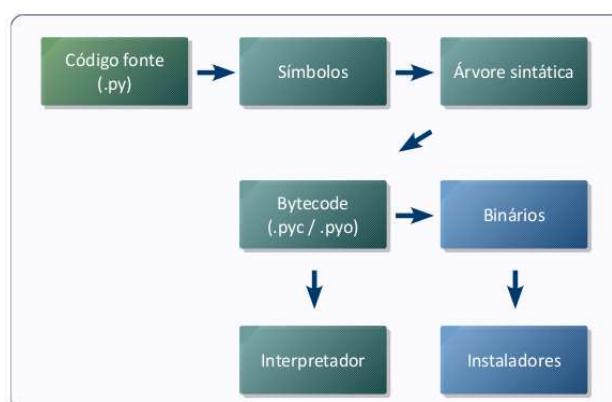


FIGURA 15: Diagrama do interpretador *Python*.
Fonte: Borges (2010, p. 16)

Borges (2010) descreve que ao ser interpretado pelo *Python*, o código-fonte é traduzido para *bytecode*, que é um formato binário com as instruções para o interpretador. Para que não necessite traduzir o código novamente, o interpretador armazena o *bytecode* em disco, fazendo com que o tempo de carga na execução seja menor.

Esse *bytecode* é armazenado com as extensões .pyc (normal) ou .pyo (otimizado).

2.3. Bibliotecas

Ao longo de todo o projeto foram utilizadas bibliotecas, pertencentes à linguagem de programação escolhida, e que auxiliaram no desenvolvimento de tarefas específicas.

Ao todo foram utilizadas nove bibliotecas, que são:

- a. *Requests*: responsável por acessar e resgatar o conteúdo HTML das páginas;
- b. *BeautifulSoup*: buscar os dados desejados, no conteúdo HTML das páginas;
- c. *Googlemaps Geocoding*: retornar os dados sobre as coordenadas geográficas, latitude e longitude, com base no nome dos bairros;
- d. *Mysql.connector*: responsável por criar uma conexão com o banco de dados;
- e. *Scikit-learn*: criar os modelos regressores e métricas de eficiência.
- f. *Numpy*: responsável pelas funções matemáticas e manipulação de matrizes;
- g. *Pandas*: manipular os dados do conjunto de dados;
- h. *Matplotlib* e *Bokeh*: responsáveis pela geração de gráficos.

Conforme descrito na introdução do capítulo, as bibliotecas *Requests*, *BeautifulSoup* e *GoogleMaps* foram utilizadas para elaboração do rastreador web e obtenção dos dados.

2.3.1 Requests

A *Requests* é uma biblioteca que está sob a licença Apache2, portanto é de código aberto e pode ser utilizada em códigos proprietários. Ela é capaz de realizar requisições em sites http, além de ser disponibilizada como biblioteca *built-in*, ou seja, ao instalar a linguagem *Python* ela já faz parte do sistema, sendo desnecessária a instalação de qualquer outra ferramenta.

2.3.2. *BeautifulSoup*

Essa biblioteca é capaz de retirar, de arquivos HTML e XML, dados de acordo com o desejo do usuário.

Possui uma gama de métodos, todos voltados para esse objetivo, além de converter os dados automaticamente para o padrão UTF-8. Atua sob licença MIT e é de código aberto.

2.3.3. *Googlemaps Geocoding*

Biblioteca sob licença Apache2 que possibilita resgatar informações de geocodificação – processo capaz de converter endereços em coordenadas geográficas e geocodificação reversa –, nesse caso, quando fornecidas as coordenadas geográficas retorna o endereço.

2.3.4. *Mysql.connector*

Biblioteca capaz de criar um *driver* de comunicação entre o algoritmo em linguagem *Python* e um servidor *MySQL*.

Com essa ferramenta foi possível armazenar os dados obtidos em um servidor de banco de dados, concluindo-se assim a terceira parte do projeto.

2.3.5. *Scikit-learn*

Scikit-learn é uma biblioteca para *Python*, que integra uma grande variedade de algoritmos para aprendizagem de máquina supervisionados ou não supervisionados (PEDREGOSA *et al*, 2011).

Essa biblioteca começou a ser desenvolvida em 2007, por David Cournapeau, como um projeto da *Google Summer of Code* e possui vários algoritmos de classificação, regressão, agrupamento (*clustering*), *gradient boosting* e outras técnicas. Atua sobre a licença *Berkeley Software Distribution* – BSD, portanto, é de código aberto e permite que seja usada comercialmente.

2.3.6. Numpy

A entidade Numpy.org (2018) define essa biblioteca como sendo matemática possibilitando a linguagem *Python*, trabalhar com arranjos, vetores e matrizes de N dimensões. Possui vários recursos, incluindo:

- a. implementação de arranjos multidimensionais;
- b. ferramentas para cálculos de matrizes;
- c. ferramentas para álgebra linear;
- d. transformadas básicas de Fourier;
- e. ferramentas de geração de números aleatórios sofisticadas.

Além da aplicação científica, pode ser usada para abrigar dados genéricos multidimensionais, permitindo a integração com inúmeros tipos de conjuntos de dados (NUMPY.ORG, 2018).

Atua sob a licença *Berkeley Software Distribution – BSD*.

Com o uso das bibliotecas *Scikit-learn* e *Numpy*, conclui-se a etapa quatro do projeto, conforme descrito na introdução.

As próximas bibliotecas listadas foram utilizadas para elaboração das etapas três e quatro do trabalho, relativas à manipulação do conjunto de dados e à criação de gráficos.

2.3.7. Pandas

Pandas é uma biblioteca, *open-source* licença BSD, que permite à *Python* uma alta performance e facilidades na operação e manipulação de dados estruturados. Segundo Pandas (2018), entre seus recursos, podem-se destacar:

- a. eficiente objeto *DataFrame* para manipulação de dados;
- b. ferramentas para leitura e gravação de dados para diferentes formatos;
- c. remodelagem e rotacionamento de conjunto de dados;
- d. alta performance em mesclar e unir conjuntos de dados;
- e. colunas podem ser inseridas e deletadas das estruturas de dados;
- f. manuseio de dados faltantes e alinhamento de dados;
- g. manipulação de dados confusos de forma ordenada.

2.3.8. *Matplotlib*

A *Matplotlib* foi desenvolvida por Jhon Hunter a partir de 2003, com atuação sob a licença BSD. Alguns exemplos de gráficos podem ser vistos na Figura 16.

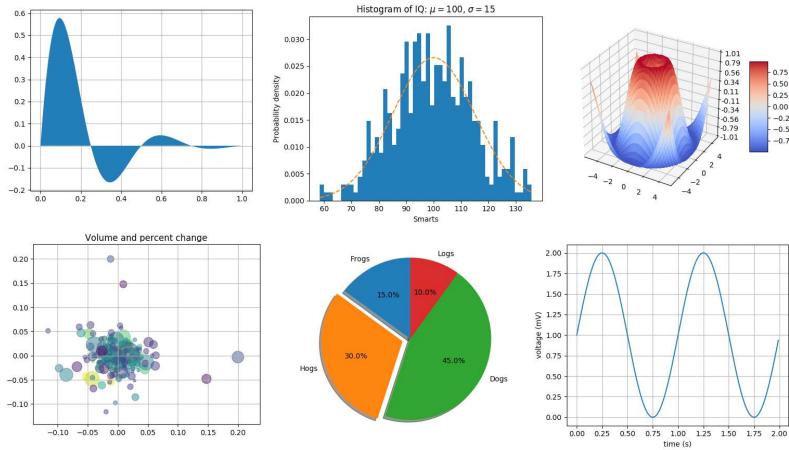


FIGURA 16: Exemplos de gráficos produzidos com a biblioteca *Matplotlib*.
Fonte: Matplotlib (2017 – Adaptada)

É uma biblioteca de produção de gráficos em 2D para *Python*, com capacidade de gerar, além de gráficos de linhas, histogramas, espectros de potência, gráficos de barra, erros e dispersão (MATPLOTLIB.ORG, 2018).

Para trabalhos em 3D existe a possibilidade de instalação do kit de ferramenta *mplot3d*.

2.3.9. *Bokeh*

Assim como a *Matplotlib*, a biblioteca *Bokeh*, possui a finalidade de criação de gráficos; a diferença é sua capacidade em gerar gráficos interativos, sendo criados como objetos HTML, ou seja, são criados com a possibilidade de serem usados em navegadores *web* (BOKEH, 2018).

A Figura 17, apresenta alguns exemplos.

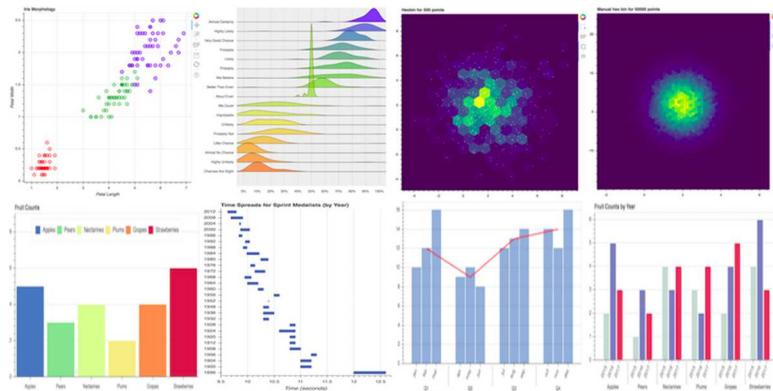


FIGURA 17: Exemplos de gráficos criados com a biblioteca *Bokeh*.
Fonte: Bokeh (2015 – Adaptada)

É também um projeto *open-source* e atua sob licença BSD.

2.4. Questionário

O questionário aplicado neste estudo foi desenvolvido por meio da plataforma da empresa *Google*, chamada *Google Forms*, que foi escolhida pela facilidade de criação, envio e resgate das respostas. Foi disponibilizado durante os dias 20/04/2018 a 23/04/2018, obtendo 111 respostas.

As respostas foram coletadas e armazenadas em uma tabela por meio da própria plataforma e os usuários entrevistados foram escolhidos aleatoriamente, mas todos com idade acima de dezoito anos.

O questionário foi utilizado na quarta etapa deste trabalho, relativa à elaboração de um sistema de recomendação e busca.

CAPÍTULO 3 RESULTADOS: APRESENTAÇÃO, ANÁLISE E DISCUSSÃO

Neste capítulo serão apresentadas as etapas de construção dos algoritmos que, juntos, constituem o sistema agregador que objetiva, por meio dos dados de imóveis coletados em cinco *sites* de imobiliárias da cidade de Montes Claros, fornecer estimativas de preços, gráficos de análises, recomendação e busca, com base na característica de cada imóvel.

É composto por quatro algoritmos principais, sendo eles:

- Rastreador web;
- Sistema de regressão;
- Sistema de recomendação e busca;
- Análise exploratória do conjunto de dados.

Por meio da Figura 18, é apresentado o fluxograma do funcionamento de cada algoritmo:

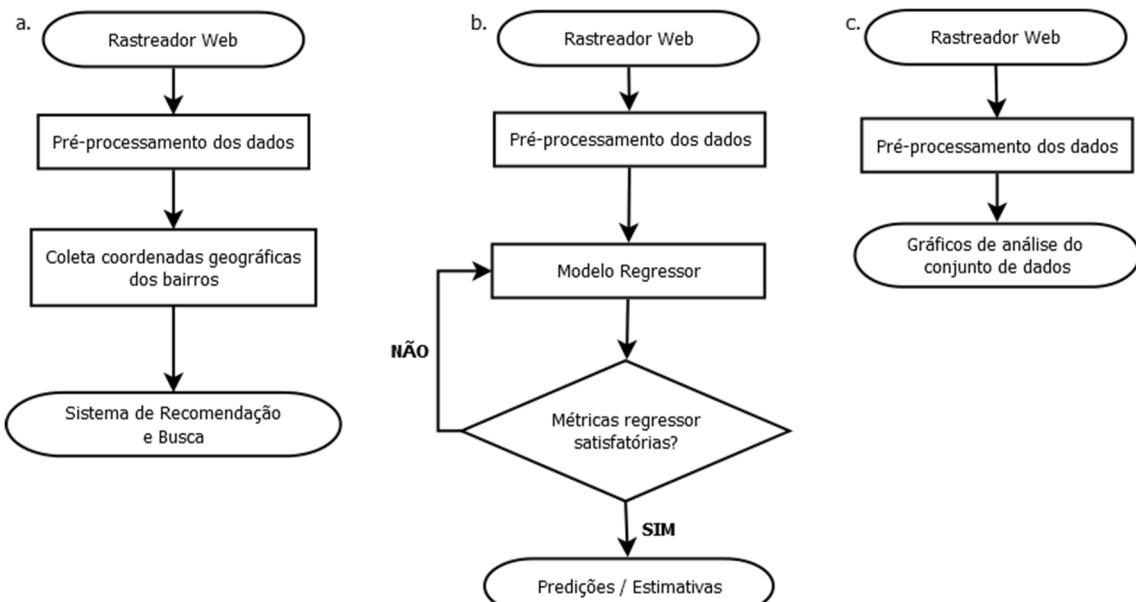


FIGURA 18: Fluxograma das aplicações. a) Fluxograma do sistema de recomendação e busca. b) Fluxograma do sistema de regressão. c) Fluxograma de análise exploratória do conjunto de dados.

Fonte: O Autor

3.1. Rastreador Web

O rastreador *web* é a parte do sistema responsável por obter e salvar dados disponibilizados pelos *sites* das empresas imobiliárias.

Foram escolhidos cinco *sites*, os quais possuíam a maior quantidade de imóveis cadastrados, para atuação do rastreador, o que gerou cinco *scripts* em linguagem *Python*, utilizando as bibliotecas *BeautifulSoup* e *Requests*, para a aplicação.

A lógica para o *script* de elaboração do sistema rastreador é apresentada na Figura 19.

```

1  from requests import get
2  from bs4 import BeautifulSoup as bs
3
4  url = 'pagina_http_requisitada'
5
6  link = get(url)
7
8  imovel = bs(link.text, 'html.parser')
9  bairros = bairros + imovel.find_all('tag', {'class' : 'tag'})
10 quartos = quartos + imovel.find_all('tag', {'class' : 'tag'})
11 banheiros = banheiros + imovel.find_all('tag', {'class' : 'tag'})
12 vagas = vagas + imovel.find_all('tag', {'class' : 'tag'})
13 areas = areas + imovel.find_all('tag', {'class' : 'tag'})
14 valores = valores + imovel.find_all('tag', {'class' : 'tag'})
15

```

FIGURA 19: Script rastreador web.

Fonte: O Autor

Primeiro são adicionadas as bibliotecas, sendo que a *requests*, por meio do método *get*, é responsável por acessar a página *web*, em que seu conteúdo é armazenado na variável intitulada ‘link’.

À variável ‘imovel’ é atribuída um objeto da classe *BeautifulSoup*, renomeada como *bs* para melhor legibilidade do código.

Para criação do objeto, são passados como parâmetros o conteúdo textual da variável ‘link’ e o tipo de *parser*, ou seja, o tipo de estrutura de arquivo que o *parser* deve atuar, nesse caso um arquivo do tipo HTML.

De acordo com o dicionário Collins (2018), um *parser* é um programa capaz de interpretar os dados de entrada, reconhecendo palavras-chave e analisando sua estrutura.

Da classe denominada *BeautifulSoup* é utilizado também o método ‘*find_all*’, responsável por buscar as informações das *tags* definidas pelo usuário de acordo com a forma em que são disponibilizadas no *site*.

Neste estudo, as *tags* escolhidas para rastreamento foram as que apresentavam as características dos apartamentos, como bairro, quantidade de quartos, banheiros e vagas de garagem, além da área e o valor de venda.

A Figura 20 exemplifica como os marcadores são disponibilizados e encontrados quando efetua-se uma exploração do código HTML da página.

```

<!--Imóvel-->
<comprar></comprar>
<div id="4854924" class="resultado">
  > <div class="foto">□</div>
  > <div class="info_imoveis">
    > <h3 class="tipo">APARTAMENTO</h3>
    > <h4 class="cidade">MONTES CLAROS - MG</h4>
    > <h4 class="bairro">EDGAR PEREIRA</h4>
    > <div class="valor">
      > <small>Venda</small>
      > <h5>R$ 155.000,00</h5>
    > </div>
    > <div class="detalhes">
      > <div class="detalhe" title="Dormitórios">
        > <i class="fa fa-bed">□</i>
        > <span>2</span>
      > </div>
      > <div class="detalhe" title="Banheiros">
        > <i class="fa fa-bath">□</i>
        > <span>1</span>
      > </div>
      > <div class="detalhe" title="Vagas">
        > <i class="fa fa-car">□</i>
        > <span>1</span>
      > </div>
      > <div class="detalhe" title="Área" style="clear:both; float:left; margin:15px 0 0 0;">
        > <i class="fa fa-expand">□</i>
        > <span>40.5</span>
        > <span>m2</span>
      > </div>
    > </div>
  > </div>
</div>

```

FIGURA 20: Exemplo de busca de *tags* para rastreador web.

Fonte: O Autor

Com esses dados em mãos, a busca dos dados se dará de acordo com a Figura 21.

```

1 imovel = bs(link.text, 'html.parser')
2 bairros = bairros + imovel.find_all('h4', {'class' : 'bairro'})
3 valores = valores + imovel.find_all('div', {'class' : 'valor'})
4 quartos = quartos + imovel.find_all('div', {'title' : 'Dormitórios'})
5 banheiros = banheiros + imovel.find_all('div', {'title' : 'Banheiros'})
6 vagas = vagas + imovel.find_all('div', {'title' : 'Vagas'})
7 areas = areas + imovel.find_all('div', {'title' : 'Área'})
8

```

FIGURA 21: Exemplo de definição das *tags* para busca dos dados.

Fonte: O Autor

Ao final, todos os dados obtidos foram armazenados em um banco de dados no *software MySQL*, por meio da biblioteca *mysql.connector* que, como o nome mostra, cria uma conexão com o servidor de banco de dados.

O código para armazenamento é exemplificado na Figura 22.

```

1 import mysql.connector
2 from datetime import date
3
4 dia = date.today()
5
6 conector = mysql.connector.connect(user, password, host, database)
7 cursor = conector.cursor()
8 for bairro, dormitorio, banheiro, vaga, area, valor in \
9     zip(bairros, dormitorios, banheiros, vagas, areas, valores):
10     cursor.execute("""INSERT INTO {} (dia, bairro, quartos, banheiros, garagens, area, valor) \
11         VALUES (%s, %s, %s, %s, %s, %s)""".format(nome_table),
12         (dia, bairro, dormitorio, banheiro, vaga, area, valor))
13     conector.commit()
14 conector.close()

```

FIGURA 22: Código para armazenamento dos dados.

Fonte: O Autor

Importam-se as bibliotecas para conexão no servidor *MySQL* e para obtenção da data corrente, por meio da biblioteca *datetime*.

À variável ‘conector’ atribui-se um objeto *mysql.connector*, que recebe como parâmetros, o nome do usuário, senha, endereço do servidor e nome do banco de dados.

Após criada a conexão, a variável ‘cursor’ recebe um objeto do tipo *conector.cursor*, responsável por apontar para uma determinada linha na tabela SQL e executar uma ação.

Com todos os elementos obtidos na página é feita uma iteração, conforme linhas 8 e 9, executando uma solicitação de inserção de dados, conforme linhas 10 e 11.

A cada inserção é efetuado um comando ‘commit’, linha 13, para conclusão desse processo na tabela.

Após todos os elementos serem inseridos na tabela, a conexão com o banco de dados é fechada conforme comando definido na linha 14. As tabelas criadas no banco de dados possuem a estrutura apresentada na Figura 23.

```

create table nome_tabela(
    id int not null auto_increment,
    dia date not null,
    bairro varchar(50) not null,
    quartos tinyint,
    banheiros tinyint,
    garagens tinyint,
    area float,
    valor bigint,
    primary key(id)
) default charset = utf8;

```

FIGURA 23: Estrutura de criação da tabela no servidor *MySQL*.

Fonte: O Autor

Finalizadas as inserções de dados, geraram-se cinco tabelas com o seguinte padrão.

TABELA 1

Características dos imóveis

ID	DIA	BAIRRO	QUARTOS	BANHEIROS	GARAGENS	AREA	VALOR
1	2018-04-02	Augusta Mota	3	1	2	90	320000
2	2018-04-02	Augusta Mota	2	1	1	85	300000
3	2018-04-02	Ibituruna	3	2	2	0	287500
4	2018-04-02	Canelas	2	1	1	62.5	198000
5	2018-04-02	São José	4	3	1	0	285000

Fonte: O Autor

Ao todo neste estudo foram obtidos dados relativos a 670 imóveis.

Essas tabelas foram utilizadas para o sistema regressor, análise exploratória e sistemas de recomendação e busca.

A formatação dos dados, conforme mostrado na Tabela 1, foi imprescindível para a criação e funcionamento do sistema agregador, pois, por meio dela e da forma como os dados foram dispostos, a máquina foi capaz de interpretar e processar as informações.

3.2. Regressor

O regressor foi concebido objetivando a predição dos preços dos imóveis com base em suas características. Por meio desse será possível ao usuário ter uma ideia do valor de um determinado imóvel em um bairro específico, sem a necessidade de consultar várias imobiliárias.

Para que funcionasse como esperado foi necessário, primeiramente, efetuar o pré-processamento dos dados que serão utilizados.

Esse pré-processamento foi dividido em 3 partes:

- a. ajuste dos nomes dos bairros;
- b. verificação da quantidade de valores nulos por coluna e exclusão das colunas que possuíam mais de 80% de valores nulos;
- c. imputação de valores aos valores nulos, utilizando o critério de média.

Os nomes dos bairros cadastrados nos *sites* das imobiliárias, em sua grande maioria, possuem diferenças, seja de acentuação, seja no uso de letras maiúsculas. Portanto, na primeira parte do pré-processamento, eliminou-se a acentuação e os nomes dos bairros passaram a possuir letra maiúscula somente na letra inicial, inclusive dos elementos conectores, tais como ‘de’, ‘o’, ‘os’.

A Figura 24 apresenta o código desenvolvido para tal procedimento.

```

1 def retirar_acento(texto):
2     return normalize('NFKD', texto).encode('ASCII', 'ignore').decode('ASCII')
3
4 def ajustar_nome(nome_bairro):
5     nome_bairro = nome_bairro.str.title().str.strip()
6     nome_bairro = retirar_acento(nome_bairro)
7     return nome_bairro
8

```

FIGURA 24: Códigos para ajuste dos nomes dos bairros.

Fonte: O Autor

O próximo passo foi verificar a quantidade de valores nulos por coluna e optou-se por não eliminar colunas com menos de 80% de valores nulos. A Figura 25 apresenta a porcentagem encontrada em cada uma delas.

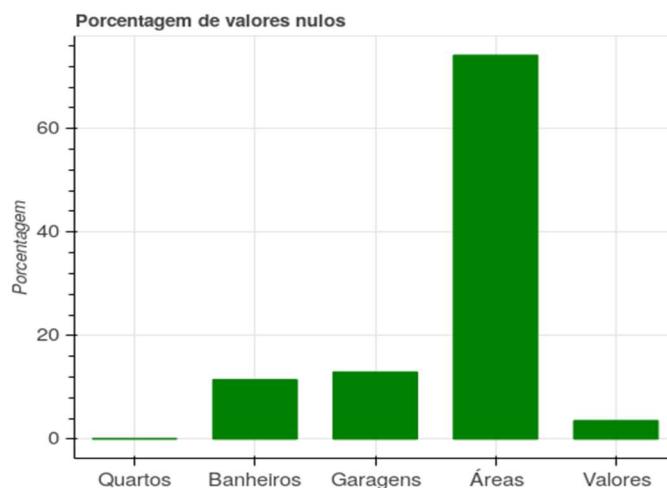


FIGURA 25: Porcentagem de valores nulos por coluna.

Fonte: O Autor

Nesse caso, nenhuma das colunas atingiu 80% de valores nulos, portanto, não ocorreram exclusões.

O último passo do pré-processamento consistiu em imputar valores aos que se encontravam nulos no conjunto de dados. Esse procedimento foi feito atribuindo a média de um mesmo conjunto dos valores ausentes, com exceção da coluna

'garagens', pois é possível que um apartamento não possua vaga. A Figura 26 apresenta o código para tal procedimento.

```

1 def imputer_media(dataset, coluna):
2     mask1 = dataset[coluna] == 0
3     mask2 = dataset[coluna] != 0
4     dataset.loc[mask1, coluna] = dataset.loc[mask2, coluna].mean()
5     return dataset

```

FIGURA 26: Código de imputação de valores da média.

Fonte: O Autor

Ao final, os dados ficaram disponíveis para manipulação, conforme demonstra a Tabela 2.

TABELA 2

Conjunto de dados após pré-processamento

ID	DIA	BAIRRO	QUARTOS	BANHEIROS	GARAGENS	AREA	VALOR
1	2018-02-04	Planalto	2	1	1	48	333075.79
2	2018-02-04	Todos Os Santos	2	2	0	51.215	333075.79
3	2018-02-04	Cidade Santa Maria	3	2	2	120	480000
4	2018-02-04	Sao Jose	4	3	1	90.752	285000
5	2018-02-04	Centro	4	5	2	149.838	750000
6	2018-02-04	Morada Do Sol	3	1	2	155.432	370000

Fonte: O Autor

Finalizado o pré-processamento a próxima etapa foi a escolha do modelo de regressor.

Para definição desse modelo, foi necessário verificar, por meio de um diagrama de dispersão, a distribuição dos valores das variáveis no hiperplano, além do índice de correlação entre as variáveis independentes e a dependente, ou seja, a relação entre as variáveis 'quartos', 'banheiros', 'garagens' e 'areas' com a variável 'valor'.

Na Figura 27, são apresentados os diagramas dos valores das variáveis independentes com relação a variável dependente.

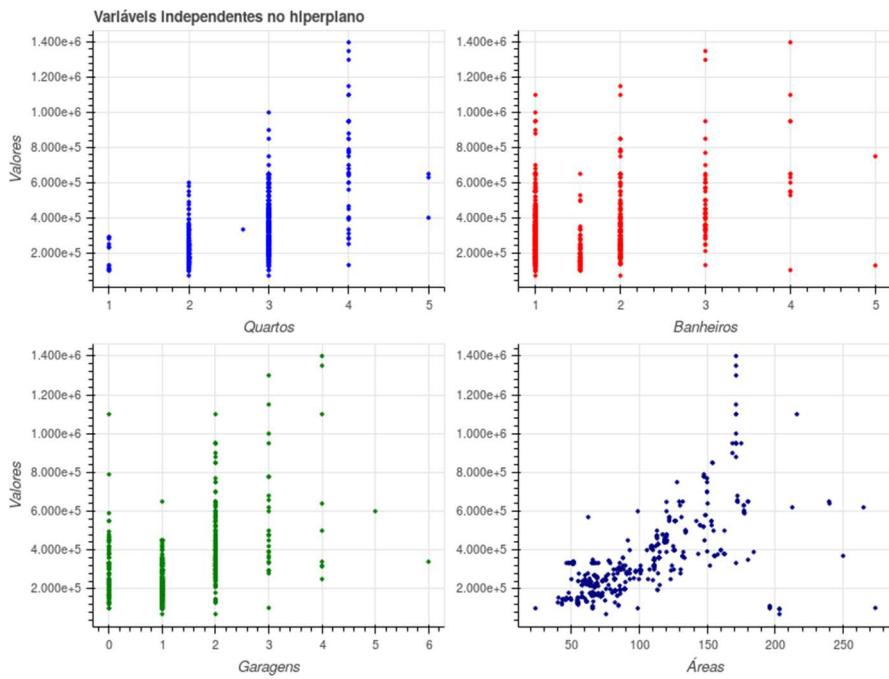


FIGURA 27: Variáveis independentes no hiperplano.

Fonte: O Autor

Por meio da Figura 27, observa-se que esse conjunto de dados possui uma baixa correlação linear com a variável dependente ‘valor’, pois, conforme Morettin e Bussab (2010), a correlação linear é mais forte quando os dados das variáveis envolvidas se apresentam próximos a uma reta. A matriz de correlação, conforme a Figura 28, evidencia o baixo índice de correlação linear.

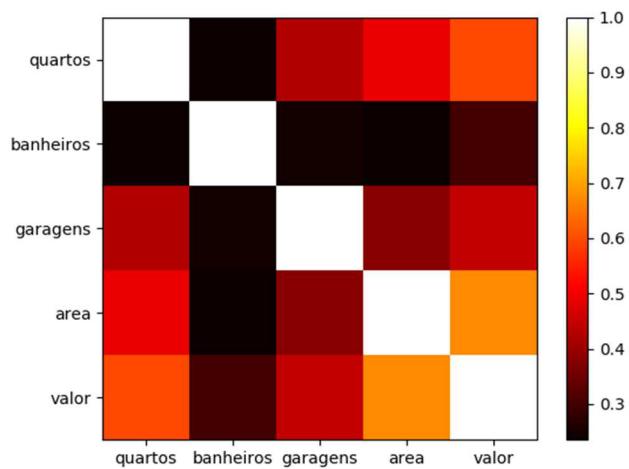


FIGURA 28: Matriz de correlação entre as variáveis.

Fonte: O Autor

Conforme observado, a variável independente ‘área’ é a que possui o maior índice de correlação, algo em torno de 0,7, indicando que essa abordagem não seria a mais ideal. O coeficiente de correlação, segundo Bussab e Morettin (2010), é uma medida de relação linear entre duas variáveis e varia entre -1 e $+1$.

Seja a correlação entre x e y dada por $p(y, x)$. Ela é dita perfeita se $p(y, x) = \pm 1$, isso ocorre pois $y = ax + b$, portanto $p(y, x) = 1$ se $a > 0$ e $p(y, x) = -1$ se $a < 0$ (BUSSAB; MORETTIN, 2010).

Sendo assim, os valores de correlação das variáveis se mostraram baixos para se estabelecer uma associação linear consistente.

Outras métricas para avaliar o modelo de regressão linear, foram as listadas abaixo:

- média do resultado da validação cruzada;
- desvio-padrão da validação cruzada;
- média do erro absoluto;
- valor de coeficiente de determinação (R^2).

A validação cruzada divide, aleatoriamente, os dados utilizados para treino, em n partes iguais, definidas pelo usuário. Com isso, a cada iteração, uma dessas partes é utilizada para teste; enquanto as outras para treino. Esse processo ocorre até que todas as partes sejam treinadas e testadas. Ao final se obtém a média geral de acurácia de todos os treinos e testes além do desvio-padrão médio, ou seja, a média total de quanto cada resultado de teste e treino está distante da média geral (MUELLER; MASSARON, 2016). Idealmente o valor de média da acurácia deve ser próximo de 1 e o desvio-padrão o menor possível.

A Figura 29 demonstra o funcionamento desse processo.

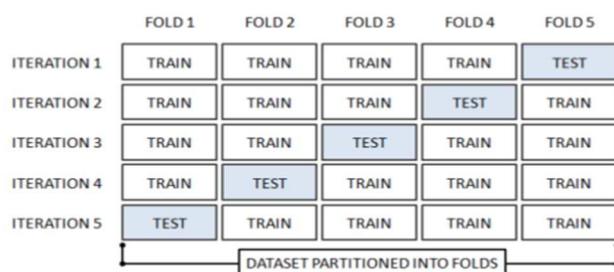


FIGURA 29: Processo de validação cruzada.
Fonte: Mueller e Massaron (2016, p. 192)

Conforme descrito, o valor da média de acurácia da validação cruzada foi utilizado para verificar a efetividade do modelo.

O erro absoluto é o valor real de x subtraído pelo valor estimado de \bar{x} , ou seja, $|x - \bar{x}|$ (REAMAT, 2018). Portanto, o ideal é que esse valor seja o menor possível.

O valor de R^2 fornece uma medida absoluta de ajuste do modelo aos dados fornecidos, ou seja, o quanto que uma variável dependente pode ser explicada pela variável independente e o ideal é que esse valor seja próximo de 1 (JAMES *et al*; 2017).

Para cálculo das métricas, o conjunto de dados necessitou ser dividido entre treinamento e teste, de acordo com a Figura 30.

```

1 x = dataset.iloc[:, 0:5].values
2 y = dataset.iloc[:, 5].values
3
4 from sklearn.model_selection import train_test_split
5 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

```

FIGURA 30: Separação do conjunto de dados entre treino e teste.

Fonte: O Autor

De acordo com o apresentado na Figura 30, as linhas 1 e 2 são responsáveis por atribuir as variáveis ‘ x ’ e ‘ y ’ os valores contidos no *dataset*, em forma de um *array*. Importa-se da biblioteca o método responsável, linha 4. Por fim, atribui-se às variáveis ‘ x_train ’, ‘ x_test ’, ‘ y_train ’, ‘ y_test ’ o resultado da separação que, nesse caso, foi configurado, pelo parâmetro *test_size*, que o conjunto de teste teria 20% do tamanho total desse conjunto.

A Figura 31 apresenta o método para cálculo e exibição das métricas.

```

1 def linear():
2     print('Média Validação Cruzada,', 'Desvio-Padrão Validação Cruzada,', 
3         'Média Erro Absoluto,', 'R²,', 'Variância Explicada')
4     regressor = LinearRegression()
5     regressor.fit(x_train, y_train)
6     y_predict = regressor.predict(x_test)
7     acuracia = cross_val_score(regressor, x_train, y_train, cv=10, n_jobs=-1)
8     print(acuracia.mean(), ',', acuracia.std(), ',', metrics.mean_absolute_error(y_test, y_predict),
9           ',', metrics.r2_score(y_test, y_predict), ',', metrics.explained_variance_score(y_test, y_predict))
10

```

FIGURA 31: Método de avaliação do modelo linear.

Fonte: O Autor

Nas linhas 4 e 5, cria-se um objeto regressor linear e efetua-se o ajuste do conjunto de dados para treinamento. A linha 6 executa o método de predição com o conjunto de teste e, na linha 7, a variável ‘acuracia’ recebe os resultados executados pela função de validação cruzada com os conjuntos de treinamento, sendo que são efetuados 10 testes, configurados pelo parâmetro ‘cv’.

Nas linhas 8 e 9 solicita-se a impressão do resultado dos testes.

Na Tabela 3, são apresentados os resultados dos cálculos das métricas do modelo linear.

TABELA 3

Resultado do cálculo das métricas do modelo linear

Média Validação Cruzada	Desvio-Padrão da Validação Cruzada	Média do Erro Absoluto (R\$)	R ²
0,6389	0,0916	67.268,09	0,4527

Fonte: O Autor

Conforme visto na Tabela 3, os valores das métricas da validação cruzada e do R² se mantiveram baixas, fazendo com que outra técnica para o sistema regressor fosse escolhida.

Após análises de alguns modelos escolheu-se a técnica de árvore de decisão, pois uma de suas vantagens é poder ser utilizada em sistemas não lineares (RASCHKA; MIRJALILI, 2017).

A construção da árvore de decisão foi efetuada em quatro etapas:

- a. pré-processamento dos dados;
- b. separação do conjunto de dados entre treinamento e teste;
- c. construção do modelo;
- d. definição da profundidade, por meio dos resultados apresentados após cada treinamento dos dados de treinamento e predição com o conjunto de teste.

O pré-processamento dos dados e divisão do conjunto de dados entre treinamento e teste foram feitos da mesma forma que para o regressor linear.

A construção do modelo, etapa três, se deu conforme apresentado na Figura 32.

```
1 from sklearn.tree import DecisionTreeRegressor
2 regressor = DecisionTreeRegressor()
```

FIGURA 32: Construção do modelo.

Fonte: O Autor

A última etapa consistiu em definir a profundidade do nível da árvore de decisão e, para isso, foi gerado um método que alterava o valor de profundidade,

treinava o modelo, com o conjunto de treinamento e efetuava a predição com os dados de teste.

Após cada treinamento e teste, os resultados dos indicadores da efetividade do modelo eram impressos; os indicadores foram os mesmos utilizados para o modelo linear.

A Figura 33 apresenta o método:

```

1 def tree():
2     print('Profundidade, ', 'Média Validação Cruzada, ', 'Desvio-Padrão Validação Cruzada, ',
3           'Média Erro Absoluto, ', 'R², ', 'Variância Explícada')
4     for profundidade in range(5, 30, 5):
5         regressor = DecisionTreeRegressor(max_depth=profundidade, random_state=0)
6         regressor.fit(x_train, y_train)
7         y_predict = regressor.predict(x_test)
8         acuracia = cross_val_score(regressor, x_train, y_train, cv=10, n_jobs=-1)
9         print(p, ',', acuracia.mean(), ',', acuracia.std(), ',', metrics.mean_absolute_error(y_test, y_predict),
10               ',', metrics.r2_score(y_test, y_predict), ',', metrics.explained_variance_score(y_test, y_predict))

```

FIGURA 33: Avaliação e definição da profundidade da árvore de regressão.

Fonte: O Autor

Os níveis de profundidade da árvore foram testados, conforme linha 4, dos valores de 5 até 25.

O valor máximo de 25 foi escolhido, pois, conforme a Figura 34, em um teste variando a quantidade de árvores de 1 até 100, as maiores variações de métrica ocorrem até, aproximadamente, esse valor.

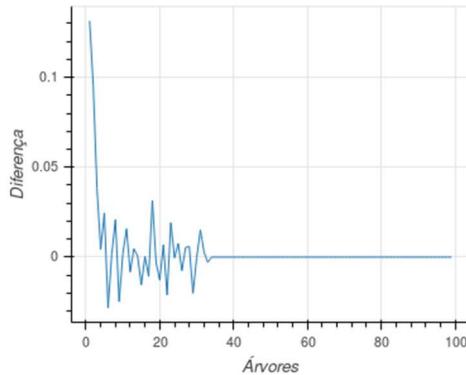


FIGURA 34: Teste do modelo com 1 a 100 árvores.

Fonte: O Autor

Conforme Figura 33, para configurar o nível de profundidade o parâmetro ‘*max_depth*’, recebeu o valor corrente da variável ‘*p*’ e para garantir que o conjunto de dados fosse treinado sempre na mesma ordem o parâmetro ‘*random_state*’, recebeu o valor 0.

Na linha 7, a variável ‘*y_predict*’ recebe os resultados, referentes à predição do conjunto de teste.

As linhas 9 e 10 são responsáveis por receber e imprimir o resultado das métricas da eficiência de cada nível de profundidade da árvore.

A Tabela 4 exibe os resultados da métricas de cada nível.

TABELA 4

Resultado das métricas da árvore de decisão de acordo com sua profundidade

Profundidade	Média Validação Cruzada	Desvio-Padrão da Validação Cruzada	Média do Erro Absoluto (R\$)	R ²
5	0,7683	0,1230	51.808,86	0,5145
10	0,7623	0,1556	37.086,16	0,5738
15	0,7769	0,1398	35.937,07	0,5805
20	0,7797	0,1308	32.791,86	0,6265
25	0,7714	0,1506	36.885,14	0,5408

Fonte: O Autor

Com base nos valores apresentados na Tabela 4, o nível de profundidade com maior eficiência é o 20, pois apresentou o maior valor de média da acurácia da validação cruzada e R², e os menores valores de desvio-padrão e média do erro absoluto. Mas, apesar da melhora em relação ao modelo linear, os valores de média de validação cruzada e R² se mantiveram bem abaixo do 1 ideal, o que indicou que somente uma árvore não era suficiente para garantir um regressor consistente.

Devido a desse motivo, o mesmo método de cálculo de métricas foi aplicado em uma floresta aleatória, em que a quantidade de árvores e profundidade eram alterados automaticamente e o resultado exibido ao final.

A seguir, na Figura 35, tem-se o método criado para teste de uma floresta aleatória.

```

1 def random_forest():
2     print('Árvores/Profundidade,', 'Média Validação Cruzada,', 'Desvio-Padrão Validação Cruzada,', 
3           'Média Erro Absoluto,', 'R²,', 'Variância Explícada')
4     for arvores in range(5, 30, 5):
5         for profundidade in range(5, 30, 5):
6             regressor = RandomForestRegressor(n_estimators=arvores, max_depth=profundidade, random_state=0)
7             regressor.fit(x_train, y_train)
8             y_predict = regressor.predict(x_test)
9             acuracia = cross_val_score(regressor, x_train, y_train, cv=10, n_jobs=-1)
10            print(e, '/', p, ',', acuracia.mean(), ',', acuracia.std(), ',', metrics.mean_absolute_error(y_test, y_predict),
11                  ',', metrics.r2_score(y_test, y_predict), ',', metrics.explained_variance_score(y_test, y_predict))
12

```

FIGURA 35: Cálculo para floresta aleatória.

Fonte: O Autor

Em relação ao método anterior, as únicas diferenças se encontram nas linhas 4 a 6, pois, nas linhas 4 e 5, são feitas duas iterações que alteram o número de árvores e a profundidade. Na linha 6, cria-se um regressor de Floresta Aleatória.

Os resultados das métricas são apresentados na Tabela 5.

TABELA 5

Resultado das métricas da floresta aleatória de acordo com a quantidade de árvores e profundidade

Árvores / Profundidade	Média Validação Cruzada	Desvio-Padrão da Validação Cruzada	Média do Erro Absoluto (R\$)	R ²
5 / 5	0,8048	0,1071	51.561,28	0,6585
5 / 10	0,8358	0,8300	34.714,23	0,7047
5 / 15	0,8491	0,0747	36.565,53	0,6918
5 / 20	0,8405	0,0813	34.840,55	0,6929
5 / 25	0,8442	0,0797	35.548,32	0,7051
10 / 5	0,8085	0,1094	47.564,53	0,7132
10 / 10	0,8393	0,0885	33.223,20	0,7607
10 / 15	0,8467	0,0872	33.581,44	0,7517
10 / 20	0,8381	0,0952	32.772,93	0,7498
10 / 25	0,8405	0,0932	33.817,69	0,7524
15 / 5	0,8174	0,0947	46.055,14	0,7065
15 / 10	0,8500	0,0768	32.568,22	0,7523
15 / 15	0,8575	0,0739	32.370,09	0,7647
15 / 20	0,8496	0,0811	31.824,14	0,7507
15 / 25	0,8523	0,0781	32.647,13	0,7541
20 / 5	0,8240	0,0867	45.210,21	0,7284
20 / 10	0,8555	0,0694	32.516,06	0,7598
20 / 15	0,8606	0,0706	31.518,09	0,7833

20 / 20	0,8545	0,0766	30.885,83	0,7714
20 / 25	0,8560	0,0734	31.539,07	0,7711
25 / 5	0,8253	0,0859	43.852,90	0,7406
25 / 10	0,8583	0,0673	31.711,47	0,7657
25 / 15	0,8610	0,0681	30.933,56	0,7945
25 / 20	0,8577	0,0749	30.708,28	0,7711
25 / 25	0,8568	0,0737	30.828,96	0,7860

Fonte: O Autor

Pode-se verificar pelas métricas que o melhor resultado é o que contém 25 árvores com profundidade de 15 folhas, pois, conforme visto na Tabela 5, esse modelo figura entre os menores valores de média de erro absoluto e desvio-padrão, além de possuir os maiores valores de média da acurácia da validação cruzada e R². Para verificar a efetividade desse regressor, mais um teste foi feito, que pode ser observado no histograma apresentado na Figura 36. Esse teste consistiu em avaliar a diferença entre os valores reais do conjunto de teste e os valores estimados por meio desse mesmo conjunto.

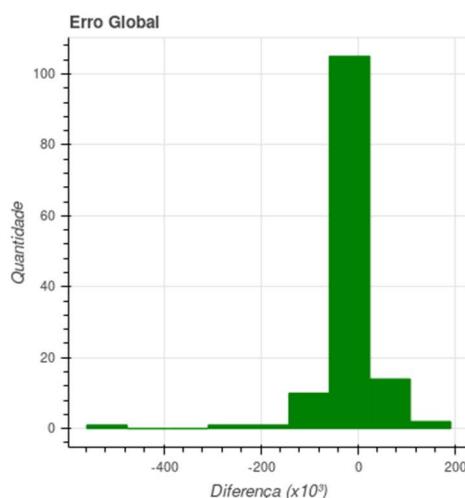


FIGURA 36: Erro global entre valores reais e preditos.
Fonte: O Autor

Nota-se que a maioria dos valores de diferença se compreendem próximos a zero, o que indicava uma ótima performance do regressor.

Um último teste foi efetuado com o objetivo de verificar o comportamento do regressor com relação ao bairro onde o imóvel se encontra e, para tal, estimaram-se valores de preços de imóveis com características semelhantes, alterando-se somente o bairro.

A definição dos aspectos dos imóveis para esse teste foi feita por meio da média das características de cada um deles encontradas no conjunto de dados, ou seja, para o imóvel de dois quartos verificou-se, antes de estimar, a média da quantidade de banheiros, garagens e o tamanho da área que um imóvel desse possui.

O método para resgate dessas informações se encontra na Figura 37.

```

1 def medias(dataset):
2     medias_geral = []
3     for i in range(1, 6):
4         md_banheiro = dataset.loc[dataset['quartos'] == i, 'banheiros'].mean()
5         md_garagem = dataset.loc[dataset['quartos'] == i, 'garagens'].mean()
6         md_area = dataset.loc[dataset['quartos'] == i, 'area'].mean()
7         medias_geral.append([i, md_banheiro, md_garagem, md_area])
8
return medias_geral

```

FIGURA 37: Média das características dos imóveis.

Fonte: O Autor

Nesse método acontece uma iteração de 1 a 5 em que os valores das médias são armazenados na variável ‘medias_geral’, conforme as linhas 3 a 7.

Na Tabela 6 são apresentados os resultados das médias dos imóveis.

TABELA 6

Média das características dos apartamentos com base na quantidade de quartos

Quartos	Média Banheiros	Média Garagens	Média Área (m ²)
1	1,28	0,53	70,43
2	1,39	1,02	73,40
3	1,53	1,54	104,85
4	2,20	2,34	150,68
5	2,66	2	128,37

Fonte: O Autor

Conforme apresentado na Tabela 6, o tamanho da área de apartamentos de cinco quartos é menor que o de apartamentos de quatro quartos, isso ocorre devido

à baixa disponibilidade de imóveis desse tipo à venda, conforme observado na seção 3.4 deste capítulo. Essa situação é parecida com os imóveis de um quarto em relação aos apartamentos de dois quartos, que possuem áreas de tamanho parecidas, devido à baixa disponibilidade de imóveis de um quarto.

Baseados nos valores de média apresentados na Tabela 6 e para um resultado mais preciso, os valores de quantidade de banheiros e garagens foram arredondados para um número inteiro.

Na Figura 38 é apresentado o método criado para predição do valor de imóvel.

```

1 def previsao_preco(bairro, quartos, banheiros, garagens, area):
2     features = [quartos, banheiros, area]
3     valor_preditivo = regressor.predict([features])
4     return valor_preditivo

```

FIGURA 38: Predição de preço do imóvel.
Fonte: O Autor.

Nas Figuras de 39 a 43, são exibidos os resultados de um mesmo tipo de apartamento em todos os bairros disponíveis no conjunto de dados.

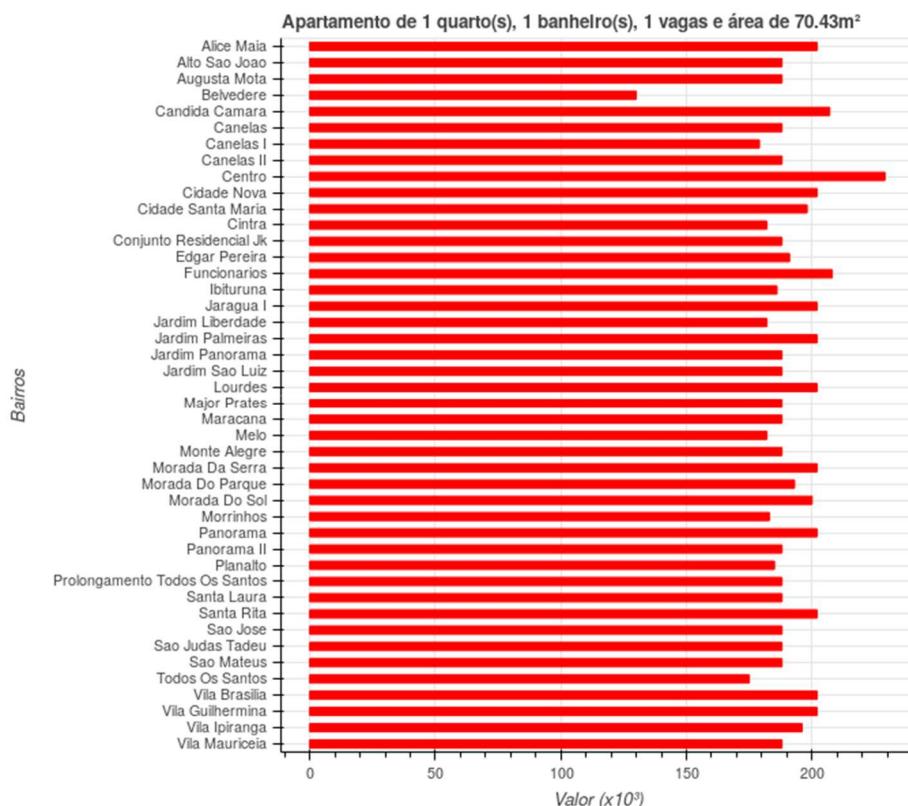


FIGURA 39: Preços para apartamentos de 1 quarto.
Fonte: O Autor

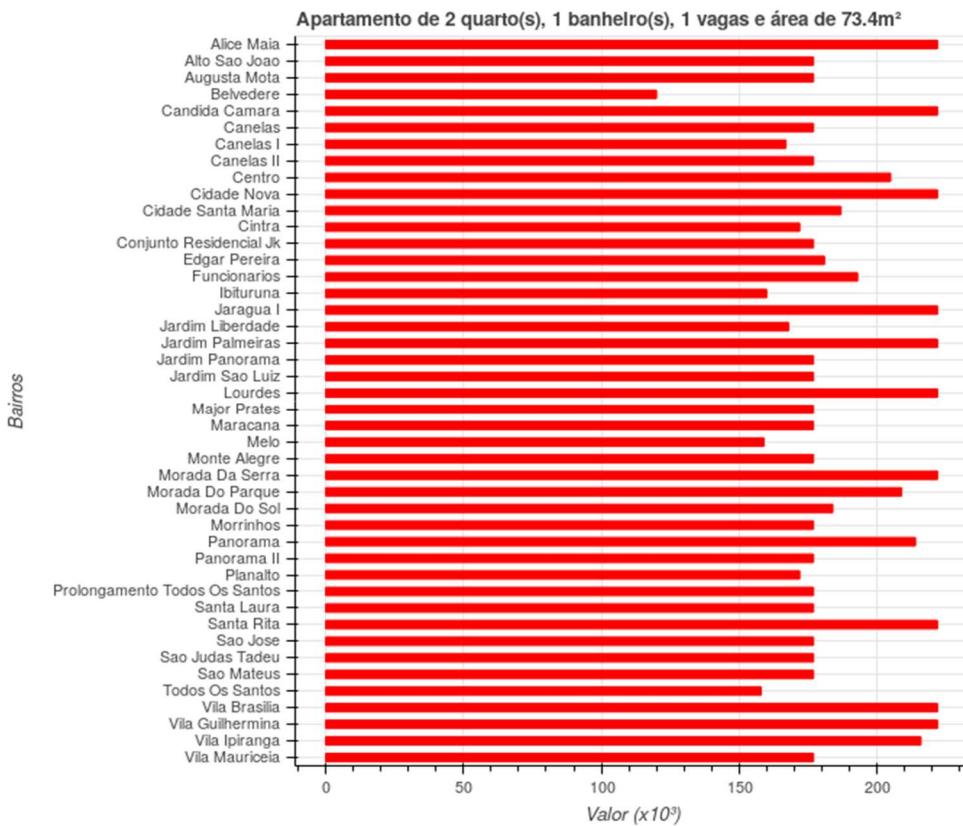


FIGURA 40: Preços para apartamentos de 2 quartos.

Fonte: O Autor



FIGURA 41: Preços para apartamentos de 3 quartos.

Fonte: O Autor

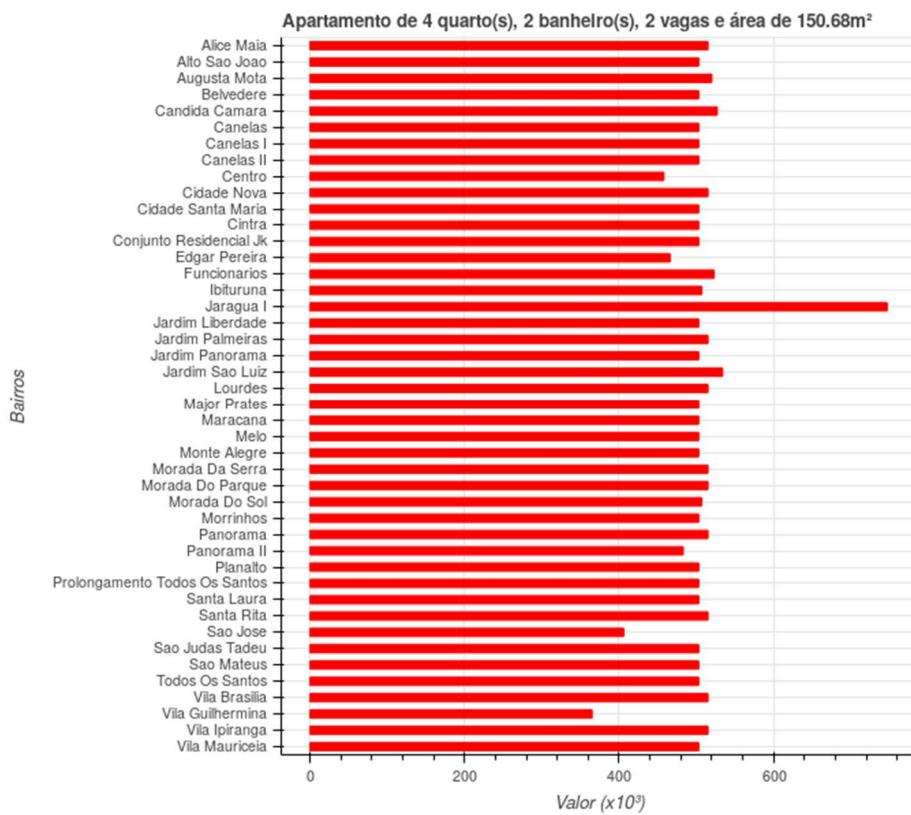


FIGURA 42: Preços para apartamentos de 4 quartos.

Fonte: O Autor

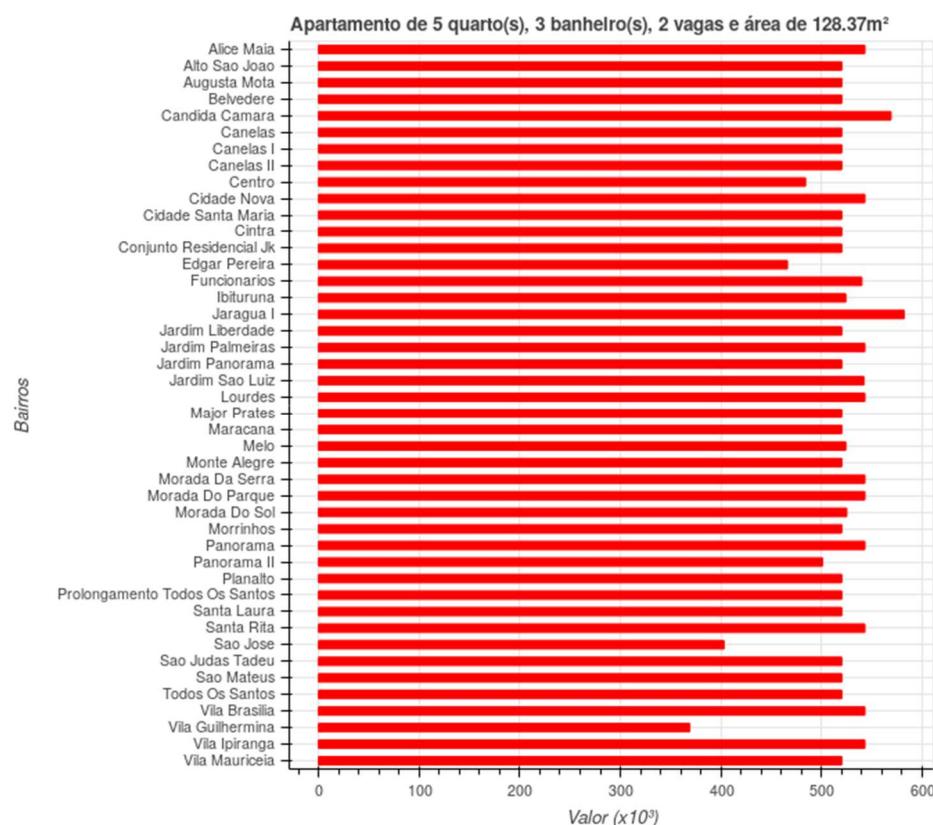


FIGURA 43: Preços para apartamentos de 5 quartos.

Fonte: O Autor

Como é possível verificar nos testes, os apartamentos, apesar de possuírem características semelhantes, recebem preços diferentes de acordo com o bairro. Esse comportamento denota a aprendizagem adquirida pelo regressor, pois conforme se observa, os valores estimados de um apartamento em um bairro mantiveram-se, em geral, estáveis, ou seja, os apartamentos com preços mais altos permanecem figurando entre os mais caros em todas as amostras, e o mesmo ocorre para apartamentos mais baratos. Comprova-se assim o funcionamento do modelo regressor.

3.3. Sistema de Recomendação e Busca

O sistema de recomendação e busca foi criado com o propósito de apresentar ao usuário opções de imóveis que sejam mais similares quanto às suas características, seja número de quartos, vagas de garagem, o bairro, e até mesmo o valor.

No processo de busca são levados em consideração somente os imóveis pertencentes ao mesmo bairro, já o processo de recomendação consulta bairros diferentes.

Para efetuar esses procedimentos, além das características dos imóveis, tais como quartos, vagas e valor, foram acrescidas as coordenadas geográficas, latitude e longitude, dos bairros. Esse acréscimo tem como objetivo possibilitar a identificação de bairros que sejam menos distantes, ou seja, tenham coordenadas geográficas mais similares.

Para obtenção das coordenadas foi utilizada uma Interface de Programação de Aplicações, sigla *API* em inglês, disponibilizada pela empresa *Google* em seu repositório de aplicativos.

O código desenvolvido é exibido na Figura 44.

```

1 import googlemaps
2 import pandas as pd
3 from unicodedata import normalize
4
5 gmaps = googlemaps.Client(key='chave')
6 dataset = pd.read_csv('bairros.csv')
7
8 def retirar_acento(texto):
9     """Retira o acento das palavras."""
10    return normalize('NFKD', texto).encode('ASCII', 'ignore').decode('ASCII')
11
12 for i in range(len(dataset['BAIRROS'])):
13    dataset['BAIRROS'][i] = retirar_acento(dataset['BAIRROS'][i])
14
15 bairros = dataset.iloc[:, 0].values
16 latitudes, longitudes = [], []
17 for bairro in bairros:
18    bairro = bairro + ' - MONTES CLAROS - MG'
19    g_result = gmaps.geocode(bairro)
20    temp = g_result[0]
21    temp = temp['geometry'].get('location')
22    latitudes.append(temp.get('lat'))
23    longitudes.append(temp.get('lng'))
24
25 with open('bairros_coordenadas.csv', 'w') as _file:
26     _file.write('BAIRROS,LATITUDE,LONGITUDE\n')
27 for bairro, latitude, longitude in zip(bairros, latitudes, longitudes):
28     _file.writelines((bairro, ',', latitude, ',', longitude, '\n'))
29 _file.close()
30

```

FIGURA 44: Busca e armazenamento das coordenadas geográficas.
Fonte: O Autor

As bibliotecas importadas possuem as seguintes funções:

- linha 1: *API do Google*, responsável pela busca das coordenadas geográficas.
- linha 2: Biblioteca responsável por manipulação de conjuntos de dados.
- linha 3: Responsável pela manipulação de textos.

Na linha 5 é feita a autenticação na *API* e na 6 a variável ‘dataset’ recebe o conjunto de dados, com o nome de todos os bairros.

Por divergências encontradas nos nomes dos bairros, principalmente pela acentuação, optou-se por eliminar quaisquer tipos de acentos. Esse procedimento, executado das linhas 8 a 13, tem por objetivo eliminar qualquer obstáculo ao efetuar a busca das coordenadas pela *API*.

A partir da linha 15 até a 22, é efetuado o procedimento de busca das coordenadas, por meio de uma iteração do conjunto de nomes atribuída à variável ‘bairros’.

Da linha 25 até 29, é gerado um arquivo chamado ‘bairros_coordenadas.csv’, que recebe o nome dos bairros, bem como as coordenadas.

Finalizados tais procedimentos, o arquivo gerado está pronto para ser utilizado no processo de recomendação e busca. O código final desse sistema é apresentado nas figuras que se seguem.

```

1 import pandas as pd
2 from sklearn.neighbors import NearestNeighbors
3
4 df1, df2, df3, df4, df5 = map(pd.read_csv, ['imob1.csv', 'imob2.csv',
5                                              'imob3.csv', 'imob4.csv',
6                                              'imob5.csv'])
7 coordenadas = pd.read_csv('bairros_coordenadas.csv')
8
9 dataset = pd.concat([df1, df2, df3, df4, df5], join='outer')
10 dataset = dataset.iloc[:, 2:]

```

FIGURA 45: Leitura e atribuição dos conjuntos de dados.

Fonte: O Autor

Pode-se ver na linha 2, da Figura 45, a importação da biblioteca *sklearn.neighbors* que contém a classe ‘*NearestNeighbors*’. Essa classe é responsável por criar o objeto que efetua o cálculo dos imóveis mais similares.

Na Figura 45 vê-se, ainda, que foram atribuídas às variáveis de ‘df1’ a ‘df5’ os conjuntos de dados dos imóveis e, na linha 7, os dados das coordenadas.

Na linha 9 é feita uma concatenação dos conjuntos de dados e na 10 foram retiradas as colunas de ‘id’ e ‘dia’ da variável ‘dataset’, pois essas não são necessárias.

O método, mostrado na Figura 46, é responsável por adicionar as informações de coordenadas geográficas no conjunto de dados.

```

1 def inserir_coordenadas_geograficas(self, dataset, dataset_com_coordenadas):
2     dataset.insert(1, 'latitude', 0)
3     dataset.insert(2, 'longitude', 0)
4     for i in range(len(dataset_com_coordenadas)):
5         mask = dataset['bairro'] == dataset_com_coordenadas['BAIRROS'][i]
6         dataset.loc[mask, ['latitude', 'longitude']] = dataset_com_coordenadas[['LATITUDE', 'LONGITUDE']].iloc[i].values
7     return dataset

```

FIGURA 46: Inserção das coordenadas geográficas.

Fonte: O Autor

Já com as coordenadas geográficas inseridas no conjunto de dados, o próximo passo foi criar um método para efetuar o cálculo de dissimilaridade entre os imóveis.

A Figura 47 exibe o método.

```

1 def vizinho_mais_proximo(quantidade_vizinhos, X, features):
2     neigh = NearestNeighbors(n_neighbors=quantidade_vizinhos, metric='euclidean').fit(X)
3     indices = neigh.kneighbors([features], return_distance=False)
4     return list(indices[0])

```

FIGURA 47: Calculador de dissimilaridade.

Fonte: O Autor

Conforme observado na Figura 47, na linha 2, a variável ‘neigh’ recebe um objeto do tipo ‘*NearestNeighbors*’ que efetua o cálculo de dissimilaridade entre todos os elementos do conjunto de dados. Esse objeto recebe a quantidade de elementos que se deseja agrupar e a métrica, definida pelo parâmetro ‘*metric*’, desejada.

Na linha 3 são passadas as características dos imóveis e retorna-se à posição dos elementos que são mais similares.

O próximo método, exibido na Figura 48, foi criado para selecionar os imóveis que devem ser utilizados para busca e recomendação, ou seja, em se tratando de busca devem ser consultados somente apartamentos no mesmo bairro, caso seja recomendação somente apartamentos de bairros diferentes.

```

68 def bairro_igual_ou_diferente(bairro, opcao):
69     if opcao == 0:
70         mask = dataset['bairro'] != bairro
71     if opcao == 1:
72         mask = dataset['bairro'] == bairro
73     data = dataset.loc[mask]
74     data = reset_dataset_index(data)
75     return data, data.iloc[:, 1:].values

```

FIGURA 48: Selecionar imóveis de acordo com o bairro.

Fonte: O Autor

Na Figura 49, são mostrados os métodos que efetuam a busca e a recomendação dos imóveis e ao final imprime-se o resultado.

```

78 def buscar(bairro, quartos, banheiros, garagens, area, valor):
79     vizinhos = 5
80     data, x = bairro_igual_ou_diferente(bairro, 1)
81     pontos = coordenadas_geograficas(bairro)
82     features = pontos + [quartos, banheiros, garagens, area, valor]
83     indices = vizinho_mais_proximo(vizinhos, x, features)
84     print('Apartamentos encontrados: ')
85     apartamentos(indices, data)
86
87
88 def recomendar(bairro, quartos, banheiros, garagens, area, valor):
89     vizinhos = 5
90     data, x = bairro_igual_ou_diferente(bairro, 0)
91     pontos = coordenadas_geograficas(bairro)
92     features = pontos + [quartos, banheiros, garagens, area, valor]
93     indices = vizinho_mais_proximo(vizinhos, x, features)
94     print('Você pode se interessar por estes: ')
95     apartamentos(indices, data)

```

FIGURA 49: Busca e recomendação de apartamentos.

Fonte: O Autor

Após finalizadas as criações dos métodos, foram feitos alguns testes com bairros diferentes, escolhidos aleatoriamente, mas que possuíam características iguais.

Esses primeiros testes objetivavam verificar o funcionamento dos métodos e, portanto, o parâmetro ‘metric’, exibido na Figura 47, foi configurado para dois tipos de cálculos de dissimilaridade, a euclidiana (‘euclidean’) e a cosseno (‘coseno’); ambas suportadas pela classe ‘*NearestNeighbors*’.

A Figura 50 apresenta os imóveis escolhidos para busca e recomendação.

```

163 buscar('Centro', 3, 2, 1, 70, 200000)
164 recomendar('Centro', 3, 2, 1, 70, 200000)
165 buscar('Cidade Nova', 3, 2, 1, 70, 200000)
166 recomendar('Cidade Nova', 3, 2, 1, 70, 200000)
167 buscar('Jardim Panorama', 3, 2, 1, 70, 200000)
168 recomendar('Jardim Panorama', 3, 2, 1, 70, 200000)

```

FIGURA 50: Chamada dos métodos de busca e recomendação.

Fonte: O Autor

Na Figura 51, são exibidos os resultados utilizando a métrica euclidiana.

Primeiro Imóvel

Apartamentos encontrados:

Bairro Centro com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Centro com 2 quarto(s), 1 banheiro(s), 0 vaga(s) de garagem, área de 0.0m² e valor de R\$ 225000.0.
 Bairro Centro com 2 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 64.4m² e valor de R\$ 230000.0.
 Bairro Centro com 2 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 60.65m² e valor de R\$ 230000.0.
 Bairro Centro com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 230000.0.

Imóveis recomendados:

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.
 Bairro Todos Os Santos com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Canelas I com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Morada Do Parque com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Canelas com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.

Segundo Imóvel

Apartamentos encontrados:

Bairro Cidade Nova com 3 quarto(s), 1 banheiro(s), 2 vaga(s) de garagem, área de 1.0m² e valor de R\$ 220000.0.
 Bairro Cidade Nova com 3 quarto(s), 1 banheiro(s), 2 vaga(s) de garagem, área de 0.0m² e valor de R\$ 220000.0.
 Bairro Cidade Nova com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 230000.0.
 Bairro Cidade Nova com 3 quarto(s), 1 banheiro(s), 2 vaga(s) de garagem, área de 96.0m² e valor de R\$ 255000.0.
 Bairro Cidade Nova com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 260000.0.

Imóveis recomendados:

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.
 Bairro Todos Os Santos com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Canelas I com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Morada Do Parque com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Canelas com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.

Terceiro Imóvel

Apartamentos encontrados:

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.
 Bairro Jardim Panorama com 2 quarto(s), 2 banheiro(s), 0 vaga(s) de garagem, área de 0.0m² e valor de R\$ 210000.0.
 Bairro Jardim Panorama com 2 quarto(s), 2 banheiro(s), 0 vaga(s) de garagem, área de 0.0m² e valor de R\$ 210000.0.
 Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 190000.0.
 Bairro Jardim Panorama com 2 quarto(s), 2 banheiro(s), 0 vaga(s) de garagem, área de 0.0m² e valor de R\$ 210000.0.

Imóveis recomendados:

Bairro Todos Os Santos com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Canelas I com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Morada Do Parque com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Ibituruna com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
 Bairro Centro com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.

FIGURA 51: Resultado busca e recomendação para métrica euclidiana.

Fonte: O Autor

E na Figura 52 são exibidos os resultados para busca e recomendação utilizando métrica de similaridade do cosseno.

Primeiro Imóvel

Apartamentos encontrados:

Bairro Centro com 2 quarto(s), 2 banheiro(s), 0 vaga(s) de garagem, área de 89.74m² e valor de R\$ 235000.0.

Bairro Centro com 3 quarto(s), 1 banheiro(s), 0 vaga(s) de garagem, área de 99.75m² e valor de R\$ 250000.0.

Bairro Centro com 2 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 64.4m² e valor de R\$ 230000.0.

Bairro Centro com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 100.0m² e valor de R\$ 298000.0.

Bairro Centro com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 64.4m² e valor de R\$ 235000.0.

Imóveis recomendados:

Bairro Candida Camara com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 77.35m² e valor de R\$ 220000.0.

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.

Bairro Prolongamento Todos Os Santos com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 69.75m² e valor de R\$ 215000.0.

Bairro Sao Jose com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 57.72m² e valor de R\$ 180000.0.

Bairro Jardim Liberdade com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 78.59m² e valor de R\$ 198100.0.

Segundo Imóvel

Apartamentos encontrados:

Bairro Cidade Nova com 3 quarto(s), 1 banheiro(s), 2 vaga(s) de garagem, área de 96.0m² e valor de R\$ 255000.0.

Bairro Cidade Nova com 4 quarto(s), 1 banheiro(s), 3 vaga(s) de garagem, área de 184.21m² e valor de R\$ 390000.0.

Bairro Cidade Nova com 3 quarto(s), 1 banheiro(s), 2 vaga(s) de garagem, área de 1.0m² e valor de R\$ 220000.0.

Bairro Cidade Nova com 3 quarto(s), 1 banheiro(s), 2 vaga(s) de garagem, área de 0.0m² e valor de R\$ 220000.0.

Bairro Cidade Nova com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 230000.0.

Imóveis recomendados:

Bairro Candida Camara com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 77.35m² e valor de R\$ 220000.0.

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.

Bairro Prolongamento Todos Os Santos com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 69.75m² e valor de R\$ 215000.0.

Bairro Sao Jose com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 57.72m² e valor de R\$ 180000.0.

Bairro Jardim Liberdade com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 78.59m² e valor de R\$ 198100.0.

Terceiro Imóvel

Apartamentos encontrados:

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.

Bairro Jardim Panorama com 2 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 65.0m² e valor de R\$ 215000.0.

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 54.83m² e valor de R\$ 150000.0.

Bairro Jardim Panorama com 2 quarto(s), 0 banheiro(s), 0 vaga(s) de garagem, área de 110.0m² e valor de R\$ 245000.0.

Bairro Jardim Panorama com 3 quarto(s), 3 banheiro(s), 0 vaga(s) de garagem, área de 112.0m² e valor de R\$ 420000.0.

Imóveis recomendados:

Bairro Candida Camara com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 77.35m² e valor de R\$ 220000.0.

Bairro Prolongamento Todos Os Santos com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 69.75m² e valor de R\$ 215000.0.

Bairro Sao Jose com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 57.72m² e valor de R\$ 180000.0.

Bairro Jardim Liberdade com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 78.59m² e valor de R\$ 198100.0.

Bairro Ibituruna com 2 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 67.76m² e valor de R\$ 220000.0.

FIGURA 52: Resultado busca e recomendação para métrica do cosseno.

Fonte: O Autor

A quantidade de cinco imóveis para busca e recomendação foi definida, pois conforme a Figura 53, a dissimilaridade entre o imóvel buscado e o imóvel

apresentado aumenta a cada unidade recomendada, causando a recomendação de imóveis com aspectos muito diferentes do desejado.

Imóveis recomendados:

Bairro Jardim Panorama com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 75.53m² e valor de R\$ 200000.0.
Dissimilaridade: 5.708

Bairro Todos Os Santos com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0

Bairro Canelas I com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0071

Bairro Morada Do Parque com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0072

Bairro Canelas com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0143

Bairro Ibituruna com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0143

Bairro Ibituruna com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0357

Bairro Augusta Mota com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 200000.0.
Dissimilaridade: 70.0357

Bairro Jardim Liberdade com 3 quarto(s), 2 banheiro(s), 1 vaga(s) de garagem, área de 78.59m² e valor de R\$ 198100.0.
Dissimilaridade: 1900.0194

Bairro Jardim Liberdade com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 198100.0.
Dissimilaridade: 1901.2893

Bairro Ibituruna com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 198000.0.
Dissimilaridade: 2001.2251

Bairro Ibituruna com 2 quarto(s), 1 banheiro(s), 0 vaga(s) de garagem, área de 0.0m² e valor de R\$ 202000.0.
Dissimilaridade: 2001.2254

Bairro Ibituruna com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 202000.0.
Dissimilaridade: 2001.2259

Bairro Ibituruna com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 198000.0.
Dissimilaridade: 2001.2259

Bairro Maracana com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 113.37m² e valor de R\$ 195000.0.
Dissimilaridade: 5000.1882

Bairro Maracana com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 113.37m² e valor de R\$ 195000.0.
Dissimilaridade: 5000.1882

Bairro Maracana com 3 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 195000.0.
Dissimilaridade: 5000.4901

Bairro Sao Mateus com 2 quarto(s), 2 banheiro(s), 0 vaga(s) de garagem, área de 0.0m² e valor de R\$ 195000.0.
Dissimilaridade: 5000.4902

Bairro Ibituruna com 2 quarto(s), 1 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 195000.0.
Dissimilaridade: 5000.4902

Bairro Panorama II com 2 quarto(s), 0 banheiro(s), 1 vaga(s) de garagem, área de 0.0m² e valor de R\$ 195000.0.
Dissimilaridade: 5000.4905

FIGURA 53: Itens recomendados e valor da dissimilaridade.

Fonte: O Autor

Por meio desses resultados, fica comprovado o funcionamento dos sistemas de recomendação e busca.

A última parte de implementação do sistema de recomendação consistiu em avaliar a eficiência das recomendações por meio de uma métrica, que para este estudo foi escolhida a de precisão-*recall*, pois, segundo Aggarwal (2016), esse tipo de métrica é capaz de avaliar e comparar a eficiência entre tipos de medidas de dissimilaridade.

A precisão representa a porcentagem de itens recomendados que realmente é relevante e o *recall* a porcentagem de todos os itens relevantes que foi recomendada (AGGARWAL, 2016).

Os cálculos de precisão e *recall* são efetuados, respectivamente, pelas Equações 11 e 12.

$$11. \text{ precisão} = \frac{\text{itens relevantes usuário}}{\text{todos os itens recomendados}}$$

$$12. \text{ recall} = \frac{\text{itens relevantes usuário}}{\text{todos os itens relevantes}}$$

A Figura 54 apresenta a implementação das Equações 11 e 12 em código.

```

98 def precisao(relevantes_usuario, recomendados):
99     valores = []
100    for relevante in relevantes_usuario:
101        valores.append(relevante/recomendados)
102    return valores
103
104
105 def recall(relevantes_usuario, relevantes_total):
106    valores = []
107    for relevante in relevantes_usuario:
108        valores.append(relevante/relevantes_total)
109    return valores

```

FIGURA 54: Implementação para cálculo da precisão e *recall*.
Fonte: O Autor

Ainda segundo Aggarwal (2016), existe a métrica F1-score, que combina os valores de precisão e *recall* em uma média harmônica.

O valor de F1-score é obtido por meio da Equação 13.

$$13. \text{ } F1 - score = \frac{2 \cdot \text{precisão usuário} \cdot \text{recall usuário}}{\text{precisão usuário} + \text{recall usuário}}$$

A implementação da Equação 13 é apresentada na Figura 55.

```

113 - def f1_score(precisao_usuarios, recall_usuarios):
114     score = []
115 -     for precisao_usuario, recall_usuario in zip(precisao_usuarios, recall_usuarios):
116         score.append(2*precisao_usuario*recall_usuario/(precisao_usuario+recall_usuario))
117     return score

```

FIGURA 55: Implementação fórmula F1-score.

Fonte: O Autor

A avaliação da eficiência entre as métricas de dissimilaridade euclidiana e similaridade do cosseno se deu por meio de um formulário de pesquisa.

Nesse formulário, o usuário era apresentado a um texto sobre a compra de um imóvel com as características de ser localizado no bairro Centro, possuir três quartos, dois banheiros, uma vaga de garagem e um valor de R\$ 230.000,00.

Nesse mesmo texto, o usuário era informado de que não existia, no bairro Centro, algum imóvel à venda, mesmo com características diferentes. Eram então apresentadas a ele dez opções de imóveis semelhantes, sendo cinco calculadas por meio da dissimilaridade euclidiana e cinco por meio da similaridade do cosseno.

Solicitava-se, ao usuário escolher as opções recomendadas as quais ele achava mais relevante.

As Figuras 56 e 57, exibem as opções apresentadas para o usuário.

- Ibituruna, 3 quarto(s), 2 banheiro(s), 1 vaga(s) e valor de R\$ 230000.
- Edgar Pereira, 3 quarto(s), 2 banheiro(s), 1 vaga(s) e valor de R\$ 230000.
- Morada Do Sol, 2 quarto(s), 1 banheiro(s), 0 vaga(s) e valor de R\$ 230000.
- Cidade Nova, 2 quarto(s), 1 banheiro(s), 1 vaga(s) e valor de R\$ 230000.
- Funcionários, 2 quarto(s), 1 banheiro(s), 1 vaga(s) e valor de R\$ 230000.

FIGURA 56: Imóveis apresentados calculados pela dissimilaridade euclidiana.

Fonte: O Autor

- Cândida Câmara, 3 quarto(s), 2 banheiro(s), 1 vaga(s) e valor de R\$ 220000.
- Prolongamento Todos Os Santos, 2 quarto(s), 1 banheiro(s), 1 vaga(s) e valor de R\$ 215000.
- Ibituruna, 2 quarto(s), 2 banheiro(s), 2 vaga(s) e valor de R\$ 250000.
- Cidade Nova, 3 quarto(s), 1 banheiro(s), 2 vaga(s) e valor de R\$ 255000.
- Ibituruna, 3 quarto(s), 2 banheiro(s), 2 vaga(s) e valor de R\$ 280000.

FIGURA 57: Imóveis apresentados calculados pela similaridade do cosseno.

Fonte: O Autor

O formulário ficou disponível de 20/04/2018 a 23/04/2018 e, no total, foram obtidas 111 respostas, as quais foram utilizadas para efetuar os cálculos precisão, *recall* e *F1-score*.

O valor da média de *F1-score* foi utilizado para definição da métrica de dissimilaridade mais eficiente.

A Tabela 7 apresenta os resultados.

TABELA 7

Resultado cálculos de eficiência das recomendações

	Média F1-score	Média Precisão	Média Recall
Dissimilaridade Euclidiana	0,2322	0,2090	0,2613
Similaridade do Cosseno	0,1959	0,1566	0,2613

Fonte: O Autor

Conforme mostrado na Tabela 7, a média de *F1-score* da dissimilaridade euclidiana foi maior e, portanto, essa medida foi definida, com base nos itens mais relevantes escolhidos por usuários, como padrão no sistema de recomendação e busca.

3.4. Análise Exploratória dos Dados

A análise exploratória dos dados possuiu o objetivo de apresentar algumas observações a respeito do conjunto de dados.

Nessa primeira etapa da análise exploratória, o pré-processamento foi feito somente para regularizar o nome dos bairros, os demais elementos do conjunto de dados permaneceram iguais.

No estudo, esse conjunto, obtido após atuação do rastreador *web* e armazenamento em banco de dados, possui 670 amostras de apartamentos e, na Figura 58, são apresentadas as quantidades de imóveis por bairro.

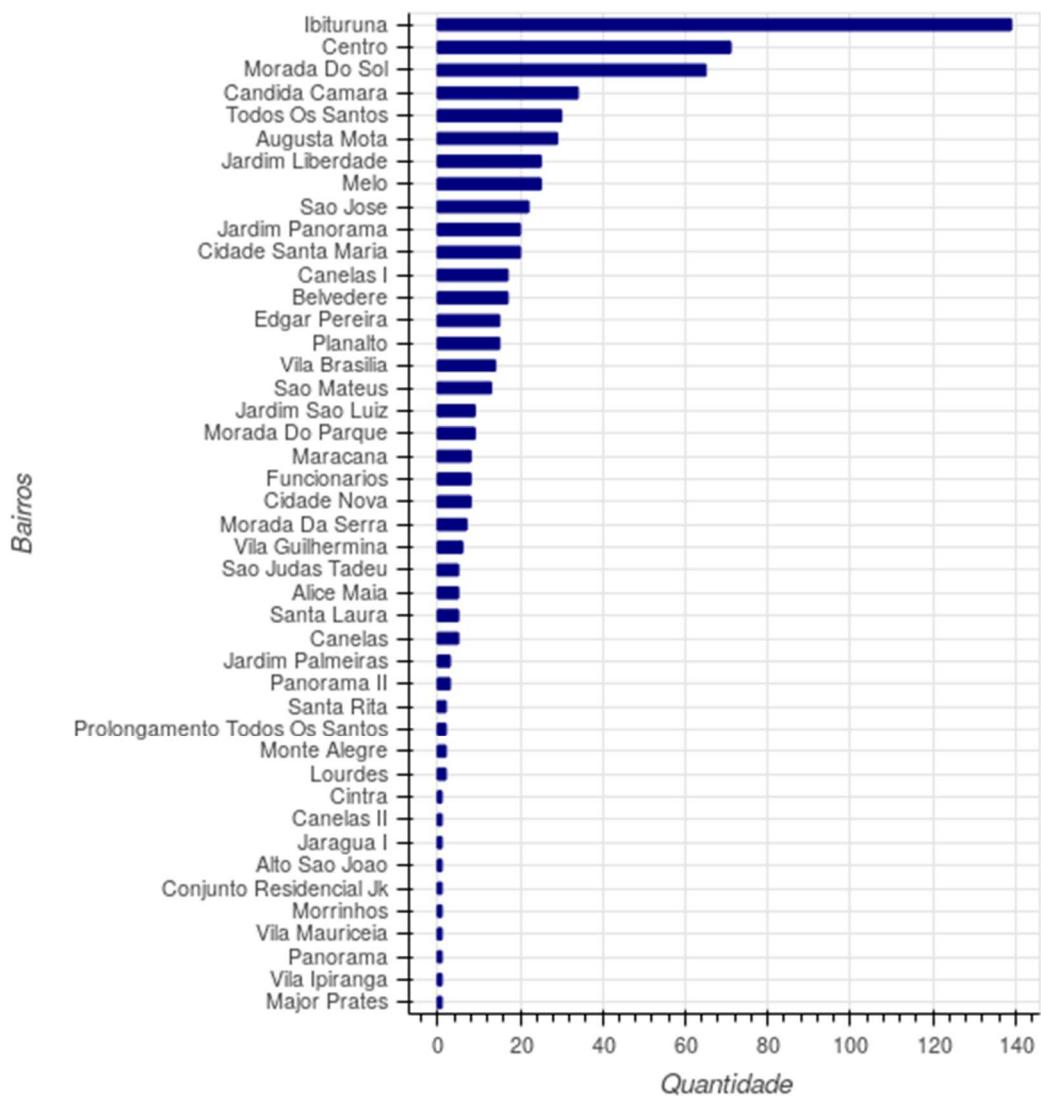


FIGURA 58: Quantidade de imóveis por bairro.

Fonte: O Autor

A Figura 59 destaca a distribuição desses imóveis pela cidade, considerados os seguintes aspectos:

- o diâmetro dos círculos é proporcional à quantidade de apartamentos;
- nos círculos vermelhos, a quantidade mínima de apartamentos é 30;
- círculos azuis possuem entre 20 e 29 apartamentos;
- círculos verdes possuem entre 10 e 19 apartamentos;
- círculos pretos possuem de 1 a 9 apartamentos.

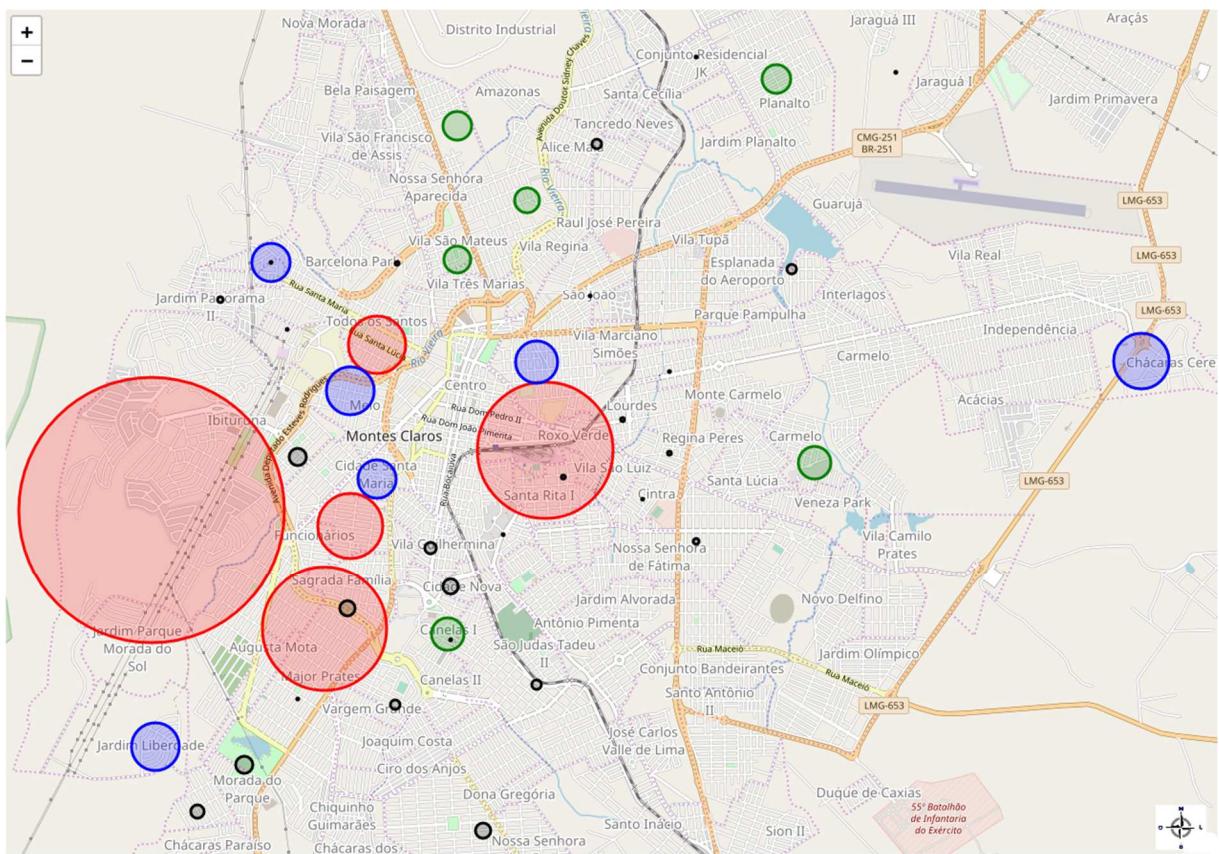


FIGURA 59: Distribuição de imóveis pela cidade.

Fonte: O Autor

Nota-se, por meio da Figura 59, uma hegemonia dos imóveis compreendida na região oeste da cidade de Montes Claros.

As Figura 60 a 62 apresentam, com base na quantidade de quartos, o número de imóveis em cada bairro.

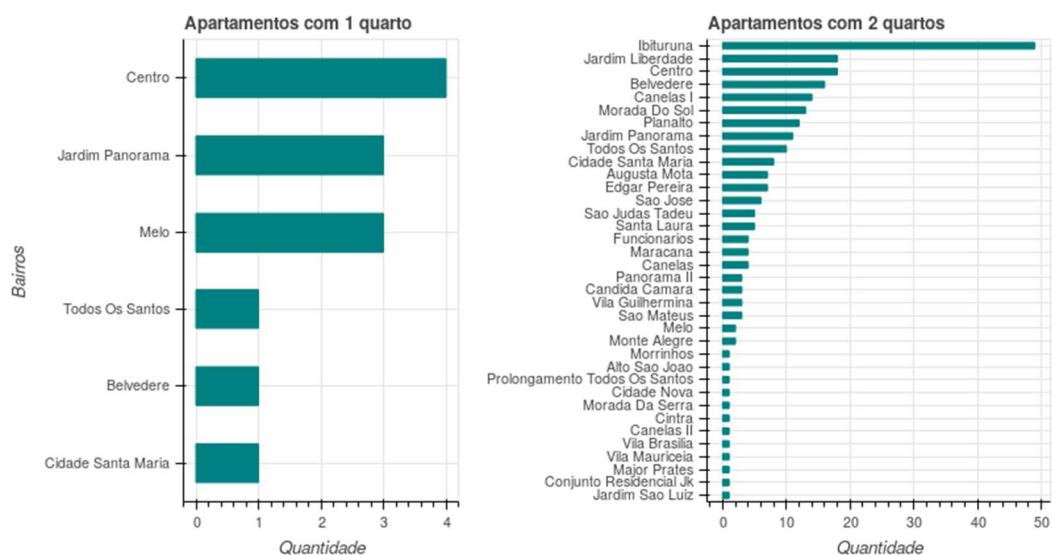


FIGURA 60: Quantidade de imóveis, por bairro, com 1 e 2 quartos.

Fonte: O Autor

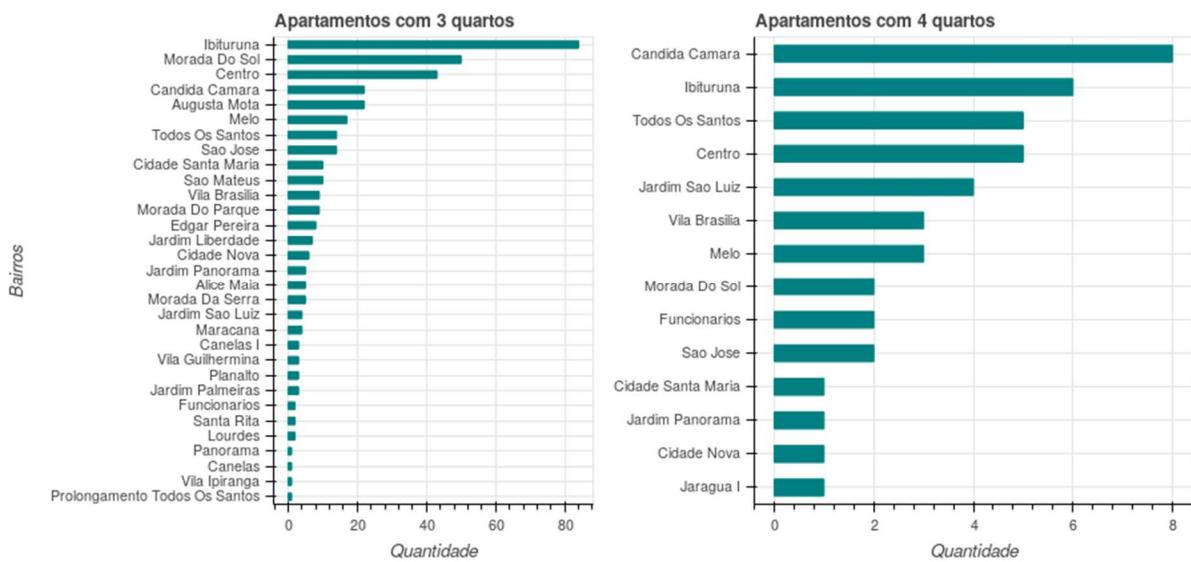


FIGURA 61: Quantidade de imóveis, por bairro, com 3 e 4 quartos.
Fonte: O Autor

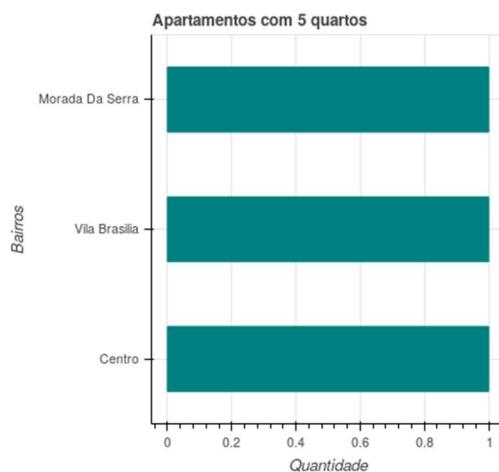


FIGURA 62: Quantidade de imóveis, por bairro, 5 quartos.
Fonte: O Autor

Como esperado, a quantidade de imóveis com três quartos é predominante e a de apartamentos de cinco quartos muito pequena. Outro ponto importante a se observar, nas Figuras 60 a 62, é que somente o bairro Centro possui imóveis em todas as representações.

E, por fim, na Figura 63, são apresentadas as porcentagens de banheiros e vagas de garagem com relação ao número de apartamentos.

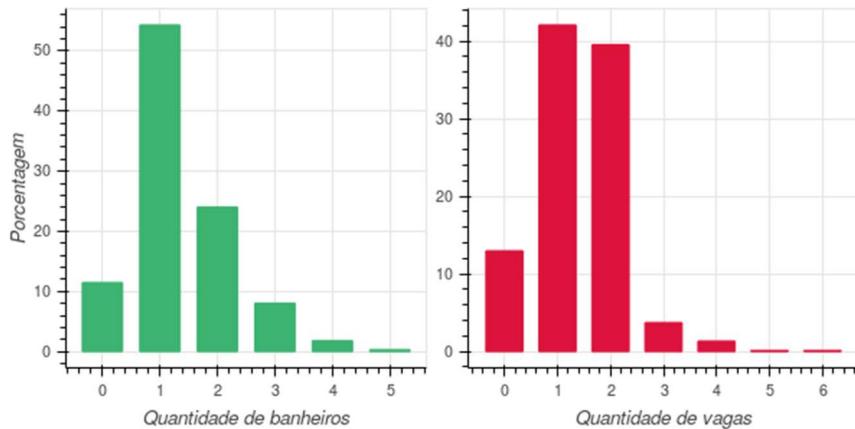


FIGURA 63: Porcentagem de banheiros e vagas de garagem.
Fonte: O Autor

Conforme observado na Figura 63, a porcentagem de imóveis sem banheiro é expressiva e é muito provável que essa falha foi ocasionada por erro humano ao catalogar os imóveis.

Merece destaque, também, a pequena diferença entre a quantidade de apartamentos com uma e duas vagas de garagem.

Sabendo-se do funcionamento correto do regressor e de que o bairro Centro possui imóveis com todas as configurações, algumas perguntas foram feitas:

- Qual a estimativa de preço para o menor apartamento, no bairro Centro, com os valores obtidos por meio dos dados?
- E para o apartamento com a maior área?
- Quanto o valor de um apartamento aumenta se aumentar a área?
- E se aumentar a quantidade quartos, banheiros e vagas?

Nessa etapa foi efetuado o pré-processamento dos dados da mesma forma que para o regressor linear.

Para a primeira pergunta, a Figura 64, apresenta o resultado da questão:

```
7 dataset['area'].min()
8 previsao_preco('Centro', 1, 1, 0, 23.5)
```

Imóvel no Centro com 1 quarto(s), 1 banheiro(s), 0 vaga(s) de garagem e área de 23.5m². Valor previsto R\$ 153197.37
FIGURA 64: Estimativa de valor para o menor apartamento.
Fonte: O Autor

Na linha 7 é obtido o valor da menor área de um apartamento do conjunto de dados; na linha 8, é feita a estimativa e, por fim, o resultado é apresentado, um valor estimado de R\$ 153.197,37.

A lógica para a segunda pergunta é a mesma, só obtendo o valor máximo de cada característica, e o resultado é apresentado na Figura 65:

```

7 dataset['quartos'].max()
8 dataset['banheiros'].max()
9 dataset['garagens'].max()
10 dataset['area'].max()
11 previsao_preco('Centro', 5, 5, 6, 273.74)

```

Imóvel no Centro com 5 quarto(s), 5 banheiro(s), 6 vaga(s) de garagem e área de 273.74m². Valor previsto R\$ 688400.0

FIGURA 65: Estimativa de valor para o maior apartamento.

Fonte: O Autor

Conforme apresentado na Figura 65, um imóvel com essas características teria um valor estimado de R\$ 688.400,00. A predição desse valor pode ser respondida ao se observar que, no conjunto de dados, os valores de imóveis de 5 quartos variam de R\$ 400.000,00 a R\$ 650.000,00.

Na terceira pergunta a estimativa tomou como base as configurações do menor apartamento e aumentou-se a área em 10m².

Imóvel no Centro com 1 quarto(s), 1 banheiro(s), 0 vaga(s) de garagem e área de 33.5m². Valor previsto R\$ 155077.37

FIGURA 66: Estimativa de valor com área aumentada em 10m².

Fonte: O Autor

A Figura 66 mostra que o acréscimo de 10m² na área aumentou o valor do imóvel em R\$ 1.880,00, que é explicado pelo preço equivalente entre apartamentos com áreas entre 20m² e 35m², observado no conjunto de dados.

A Figura 67 apresenta o aumento da área em 10m² até 103.5m² e a diferença do preço em relação ao valor imediatamente anterior.

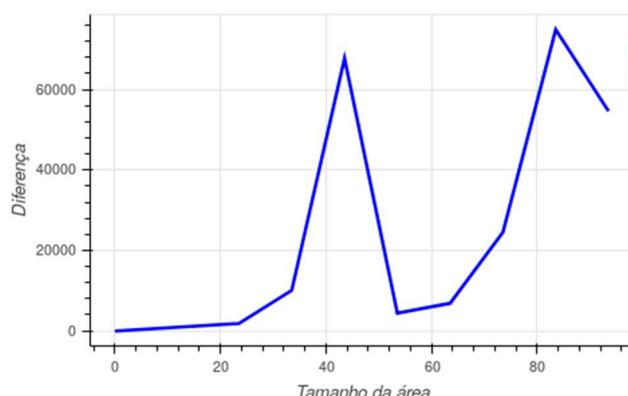


FIGURA 67: Diferença no valor do preço do imóvel a cada aumento de área.

Fonte: O Autor

Observa-se, na Figura 67, que a cada vez que o imóvel dobra de tamanho o aumento no valor é mais acentuado.

A resposta à última pergunta foi baseada no mesmo modelo de imóvel e abaixo são exibidos os resultados.

A Figura 68 representa a diferença de valor, baseada na quantidade de quartos.

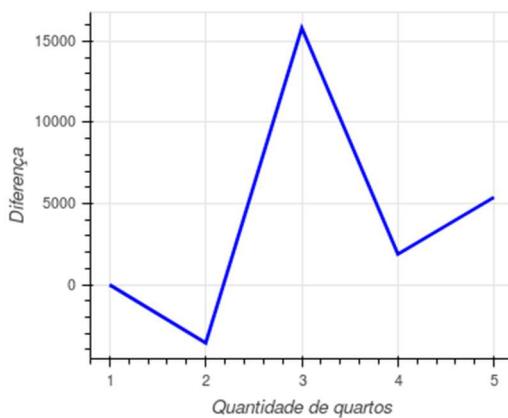


FIGURA 68: Diferença de preços baseada no aumento do número de quartos.

Fonte: O Autor

O que se observa na Figura 68 é uma redução do preço entre 1 e 2 quartos, o que pode ser explicado pela pequena demanda de apartamentos de 1 quarto com preços equivalentes a apartamentos de 2 quartos na região. Tem-se, então, que o valor médio por quarto adicionado é de R\$ 4.872,39.

A Figura 69 apresenta o resultado da diferença baseada no aumento do número de banheiros.

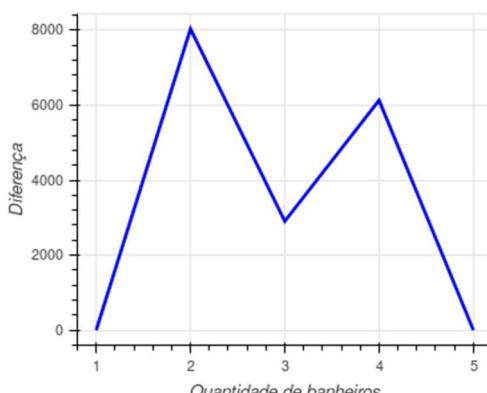


FIGURA 69: Diferença de preços baseada no aumento do número de banheiros.

Fonte: O Autor

Conforme observado na Figura 69, a grande diferença de preços ocorre quando se aumenta para 2 o número de banheiros; e outro ponto importante a se observar é que não há diferença de preço em um apartamento de 4 e 5 quartos. A média de preço no valor do apartamento ao se acrescentar um banheiro é de R\$ 4.270,59.

E, por fim, na Figura 70, tem-se a diferença de preço em um apartamento ao se acrescentarem vagas de estacionamento.

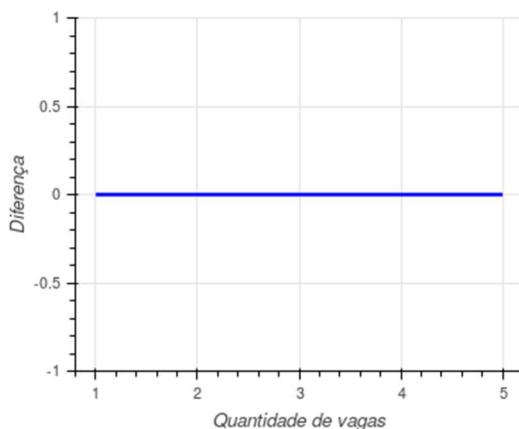


FIGURA 70: Diferença de preços baseada no aumento do número de vagas.

Fonte: O Autor

A Figura 70 mostra que, independentemente da quantidade de vagas, o valor do apartamento se mantém constante.

Os testes e resultados apresentados comprovam o funcionamento do Sistema Agregador de Informações sobre Imóveis.

CAPÍTULO 4 APLICAÇÃO

Neste capítulo serão indicadas algumas áreas em que este estudo pode ser aplicado.

Após a análise dos resultados obtidos e descritos no Capítulo 3, em que foram listados as particularidades, a abordagem e o funcionamento do sistema, constatou-se que este deve ser aplicado às empresas e/ou corretoras imobiliárias possibilitando-as a obterem aspectos relevantes sobre a situação dos imóveis na cidade quanto à oferta, preço, localização, dentre outros.

Pode ser aplicado também aos usuários que desejam realizar a compra de um apartamento e para tal desejam possuir uma ideia inicial sobre o valor do imóvel baseado em suas características.

CONSIDERAÇÕES FINAIS

O aproveitamento da internet para uso comercial se prova, a cada ano, de extrema importância para a economia, sobretudo em empresas ligadas ao setor imobiliário, em que o crescente aumento da população reflete a necessidade constante por moradia.

Sabendo-se também da grande quantidade de dados digitais gerados por esse meio tecnológico, existem atualmente ferramentas capazes de, por meio das informações disponibilizadas, aprender e predizer padrões. Por meio dessa perspectiva, este estudo procurou desenvolver um sistema que fornecesse ao usuário aspectos e ideias novas sobre um determinado imóvel, auxiliando-o na decisão de efetivar a compra ou até mesmo realizar a busca do apartamento ideal.

O resultado obtido por meio desse trabalho foi possível com a utilização de técnicas de rastreamento de dados *web*, armazenamento e manipulação dos dados para interpretação do computador, aprendizado de máquina e análise exploratória. Para que esse fosse conquistado, houve uma pesquisa sobre as tecnologias que seriam utilizadas, tais como a linguagem de programação *Python*, ambiente de desenvolvimento *VSCODE* e servidor de banco de dados *MySQL*, além de uma extensa pesquisa bibliográfica para entender os aspectos matemáticos e estatísticos que envolvem os algoritmos de aprendizagem de máquina e a busca de informações por meio do rastreamento de dados em *sites*.

O rastreador *web* foi desenvolvido com o objetivo de resgatar as características dos apartamentos, bairro em que se encontra, quantidade de quartos, banheiros, vagas de estacionamento, tamanho da área e valor do imóvel, disponibilizadas nos *sites* imobiliários. Essas características, obtidas por meio do rastreador, foram armazenadas em um servidor de banco de dados *MySQL*, corretamente configurado para recebê-las, sendo possível resgatá-las para uso posterior.

Após obtenção e armazenamento dos dados, para interpretabilidade desses pelo computador, ocorreu uma manipulação, a fim de retirar inconsistências que prejudicassem o processamento das técnicas de aprendizagem de máquina.

A técnica de aprendizado de máquina, para regressão, foi utilizada neste projeto objetivando estimar o preço de um imóvel com base em suas características se mostrando eficiente após avaliação de suas métricas e resultados adquiridos por

meio de testes. A técnica de aprendizado para recomendação objetivou buscar e/ou recomendar, apartamentos de características semelhantes ao imóvel que um usuário tenha interesse, sendo a métrica mais eficiente para busca e/ou recomendação definida por meio de um questionário, no qual os usuários eram apresentados a alternativas e indicavam as mais relevantes, tornando sua funcionalidade mais adaptada à realidade dos entrevistados.

Por fim, a análise exploratória utilizou-se dos dados obtidos e do sistema de regressão para apresentar características sobre a disponibilidade e distribuição dos apartamentos na cidade e responder perguntas relativas à inferência no preço de um imóvel dadas condições específicas de quantidade de quartos, banheiros, vagas de garagem e tamanho da área.

Todos os procedimentos propostos, sendo eles, a criação de um rastreador web, armazenamento e manipulação correta dos dados, desenvolvimento de um sistema de regressão e recomendação, além da análise exploratória dos dados foram alcançados com sucesso.

Como melhorias futuras, sugere-se a abrangência de outros tipos de imóveis, tais como casas, sítios, condomínios e outras condições de transação financeira, além de compra; criação de um sistema *web* para disponibilização das informações e a partir de uma quantidade suficiente de dados obtida durante meses, índices de valorização ou desvalorização de um bairro.

REFERÊNCIAS

AGGARWAL, Charu C. *Recommender Systems*. New York: Springer International Publishing, 2016.

ALPAYDIN, Ethem. *Introduction to Machine Learning*. 2.ed. Massachusetts: MIT Press, 2010.

BARREIRA, Elisa da Conceição Marques. *População e Enriquecimento de Ontologias através de Web Scraping*. Porto, 2014. Tese (Engenharia Informática). Instituto Superior de Engenharia do Porto.

BERNARD, Benoit. *Web Scraping and Crawling Are Perfectly Legal, Right?* 2017. Disponível em: <<https://benbernardblog.com/web-scraping-and-crawling-are-perfectly-legal-right/>>. Acesso em: 23 mar. 2018.

BISHOP, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer Science + Business Media, 2006.

BOKEH. *About*. 2017. Disponível em: <<https://bokehplots.com/pages/about-bokeh.html>>. Acesso em: 5 abr. 2018.

BORGES, Luiz Eduardo. *Python para Desenvolvedores*. 2.ed. Rio de Janeiro: Edição do Autor, 2010.

BUSSAB, Wilton de O.; MORETTIN, Pedro A. *Estatística Básica*. 6.ed. São Paulo: Saraiva, 2010.

CASTRO, Leonardo Nunes de; FERRARI, Daniel Gomes. *Introdução à Mineração de Dados: Conceitos Básicos, Algoritmos e Aplicações*. São Paulo: Saraiva, 2016.

COLLINS. *Definição de Parser*. 2018. Disponível em <<https://www.collinsdictionary.com/pt/dictionary/english/parser>>. Acesso em: 22 mar. 2018.

CRUZ, Felipe. *Python: escreva seus primeiros programas*. 1.ed. São Paulo: Casa do Código, 2015.

DATA FOR THOUGHTS. *Passo-a-passo: Data Scraping - Aliceweb*. 2014. Disponível em: <<https://dataforthoughts.wordpress.com/2014/01/29/passo-a-passo-data-scraping-aliceweb/>>. Acesso em: 17 mar. 2018.

DINO. Perspectivas do mercado imobiliário em 2018. *Revista Exame*, 19 de outubro de 2017. Disponível em: <<https://exame.abril.com.br/negocios/dino/perspectivas-do-mercado-imobiliario-em-2018/>>. Acesso em: 19 jan. 2018.

EMC. *Executive Summary: Data Growth, Business Opportunities, and IT Imperatives*. 2014. Disponível em: <<https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>>. Acesso em: 21 jan. 2018.

FACELI, et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. Rio de Janeiro: LTC, 2011.

FERNANDES, Anita Maria da Rocha. *Inteligência Artificial: noções gerais*. 1.ed. Florianópolis: Visual Books, 2003.

FERREIRA, Aurélio Buarque de Holanda. *Miniaurélio: o dicionário da língua portuguesa*. 6.ed. Curitiba: Positivo, 2005.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. *Métodos de Pesquisa*. 1.ed. Porto Alegre: Editora da UFRGS, 2009.

GORAKALA, Suresh Kumar. *Building Recommendation Engines*. Birmingham: Packt Publishing, 2016.

HEMENWAY, Kevin; CALISHAIN, Tara. *Spidering Hacks: 100 Industrial-Strength Tips & Tools*. Sebastopol: O'Reilly, 2003.

INTERNET WORLD STATS. Internet Users in the World by Regions. 2017. Disponível em: <<https://www.internetworldstats.com/stats.htm>>. Acesso em: 19 mar. 2018.

JAMES *et al.* *An Introduction to Statistical Learning: with Applications in R*. New York: Springer Science + Business Media, 2013.

LATTIN, James; CARROLL, J. Douglas; GREEN, Paul E.. *Análise de Dados Multivariados*. São Paulo: Cengage Learning, 2011.

LENTE, Caio. *O Fluxo do Web Scraping*. 18/02/2018. Disponível em: <<http://curso-r.com/blog/2018/02/18/2018-02-18-fluxo-scraping/>>. Acesso em: 26 mar. 2018.

MATPLOTLIB. *Matplotlib*. 2017. Disponível em: <<https://matplotlib.org>>. Acesso em 10 fev. 2018.

MITCHELL, Ryan. *Web Scraping with Python: Colleting Data from the Modern Web*. 1.ed. Sebastopol: O'Reilly Media, 2015.

MUELLER, John Paul; MASSARON Luca. *Machine Learning for Dummies*. New Jersey: John Wiley & Sons, 2016.

NUMPY. *Numpy*. 2017. Disponível em: <<http://www.numpy.org>>. Acesso em: 12 fev. 2018.

PANDAS. *Pandas*. 2018. Disponível em: <<http://pandas.pydata.org/index.html>>. Acesso em: 5 mar. 2018.

PEDREGOSA, *et al.* *Scikit-learn: machine learning in python*. Journal of Machine Learning Research, n.12, p. 2825-2830, 2011. Disponível em: <<http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>>. Acesso em: 12 fev. 2018.

RASCHKA, Sebastian; MIRJALILI, Vahid. *Python Machine Learning*. 2.ed. Birmingham: Packt Publishing, 2017.

REAMAT. *Tipos de Erros*. 2018. Disponível em: <https://www.ufrgs.br/reamat/CalculoNumerico/livro-sci/rdneadm-tipos_de_erro.html>. Acesso em: 10 mai. 2018.

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. *Recommender Systems Handbook*. 2.ed. New York: Springer Science + Business Media, 2015.

RUSSELL, Stuart J.; NORVIG, Peter. *Inteligência Artificial*. 3.ed. Rio de Janeiro: Elsevier, 2013.

SEGARAN, Toby. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. 1.ed. Sebastopol: O'Reilly Media, 2007.

SMITH, Gina. *The State of the Internet 2017: All Statistics Here*. 2017. Disponível em: <<http://anewdomain.net/2017-internet-statistics-the-state-of-the-internet-web-growth/>>. Acesso em: 22 mar. 2018.

SMOLA, Alex; VISHWANATHAN, S. V. N. *Introduction to Machine Learning*. 1.ed. United Kingdom: Cambridge University Press, 2008.

W3C. *Document Object Model (DOM)*. 2005. Disponível em: <<https://www.w3.org/DOM/>>. Acesso em: 16 fev. 2018.