

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Fábio de Oliveira Tabalipa**

**PREDIÇÃO DE PREÇO DE IMÓVEIS**

Belo Horizonte  
2021

**Fábio de Oliveira Tabalipa**

## **PREDIÇÃO DE PREÇO DE IMÓVEIS**

Trabalho de Conclusão de Curso apresentado  
ao Curso de Especialização em Ciência de  
Dados e Big Data como requisito parcial à  
obtenção do título de especialista.

Belo Horizonte  
2021

## SUMÁRIO

<b>1. Introdução .....</b>	<b>5</b>
<b>1.1. Contextualização .....</b>	<b>5</b>
<b>1.2. O problema proposto .....</b>	<b>5</b>
<b>2. Coleta de Dados .....</b>	<b>7</b>
<b>2.1. Web Scraping do vivareal.com.br.....</b>	<b>8</b>
<b>2.2. Web Scraping do imovelweb.com.br.....</b>	<b>9</b>
<b>3. Tratamento de Dados .....</b>	<b>11</b>
<b>3.1. Duplicidades .....</b>	<b>12</b>
<b>3.2. Ausências.....</b>	<b>12</b>
<b>3.3. Variações.....</b>	<b>14</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>16</b>
<b>4.1. Estatística Descritiva .....</b>	<b>16</b>
<b>4.2. Dados Discrepantes (Outliers) .....</b>	<b>18</b>
<b>4.3. Testes de Hipóteses.....</b>	<b>19</b>
<b>4.4. Multicolinearidade .....</b>	<b>22</b>
<b>5. Criação de Modelos de Aprendizado de Máquina.....</b>	<b>24</b>
<b>5.1. Preparo dos dados .....</b>	<b>24</b>
<b>5.2. Validação Cruzada.....</b>	<b>25</b>
<b>5.3. Auxiliares .....</b>	<b>26</b>
<b>5.4. Métricas .....</b>	<b>26</b>
<b>5.5. Regressão Linear Múltipla.....</b>	<b>27</b>
<b>5.6. Regularização de Lasso.....</b>	<b>28</b>
<b>5.7. Regularização de Ridge .....</b>	<b>29</b>
<b>5.8. Regularização Elastic-Net.....</b>	<b>30</b>
<b>5.9. Árvore de Decisão .....</b>	<b>32</b>
<b>5.10. Floresta Aleatória .....</b>	<b>33</b>
<b>5.11. Árvore de Decisão Aumentada .....</b>	<b>35</b>
<b>6. Apresentação dos Resultados .....</b>	<b>40</b>
<b>6.1. Fluxo de Trabalho.....</b>	<b>40</b>
<b>6.2. Melhor Modelo .....</b>	<b>40</b>
<b>6.3. Principais Atributos .....</b>	<b>42</b>

6.4. Métricas e Interpretação .....	44
6.5. Deploy do Modelo.....	45
6.6. Conclusões .....	46
7. Links .....	47
REFERÊNCIAS.....	48
APÊNDICE A – WEB SCRAPING DO VIVAREAL.COM.BR.....	49
APÊNDICE B – WEB SCRAPING DO IMOVELWEB.COM.BR.....	51
APÊNDICE C – CORREÇÃO DAS VARIAÇÕES DE BAIRROS.....	53
APÊNDICE D – AUXILIARES PARA OS ALGORITMOS DE REGRESSÃO.....	56
APÊNDICE E – ESCORE DE IMPORTÂNCIA DOS ATRIBUTOS .....	59
APÊNDICE F – CÓDIGO-FONTE DA PÁGINA DE PREDIÇÃO.....	60
APÊNDICE G – CÓDIGO-FONTE DO SERVIÇO DE PREDIÇÃO.....	66

## **1. Introdução**

### **1.1. Contextualização**

O mercado imobiliário é o setor da economia em que são negociados os bens imóveis com operações de compra, venda e aluguel, seja diretamente dos proprietários com os compradores, ou através de serviços de imobiliárias.

Pessoas e empresas são atraídas naturalmente a este mercado, que apresenta oportunidades por demandas não só de moradia, mas também de investimento. No cenário atual, com a baixa nas taxas de juros dos bancos, o investimento em imóveis ainda é uma modalidade razoavelmente segura e com boa rentabilidade.

No entanto, tais demandas são influenciadas por uma ampla gama de fatores, como demografia, economia e política. No próprio ano de 2020 vivenciamos um abalo socioeconômico por conta de uma pandemia, com impacto significativo inclusive no setor imobiliário. É um domínio que, para ser bem praticado, envolve conhecimento de diferentes ciências, e esse fator em específico já agrupa complexidade para a atividade de se investir.

### **1.2. O problema proposto**

O preço de venda em qualquer mercado é consequência da relação entre a oferta e a demanda para um ativo. A estratégia de preço – abaixo, acima ou à preço de mercado – é fundamental para definir o balanço entre lucro e liquidez. Com um preço acima do mercado, pode ser maior o lucro, mas menor a liquidez; e vice-versa.

No entanto, diferente do mercado de ações ou de troca de câmbio, por exemplo, em que o preço do mercado para o ativo é notório, a detecção do preço de

mercado para os ativos imobiliários é mais complexa, e isso dificulta a definição da estratégia de preço.

Os *websites* de anúncio de imóveis destacam-se como fonte dos preços do mercado imobiliário. Neles, um proprietário ou um corretor cadastra um imóvel e expõe os atributos que tentem justificar o preço de oferta.

Neste projeto, iremos analisar esses atributos, identificando quais influenciam, e em qual magnitude, o preço de oferta. E o objetivo principal é desenvolver um modelo que seja capaz de predizer o preço de oferta de um imóvel de acordo com seus atributos. Essa predição poderá auxiliar na definição da estratégia de preço, importante para os investimentos em mercado imobiliário.

O setor é bastante sensível a características geográficas, e, por este motivo, é necessário definir uma granularidade para o atributo geográfico que, ao mesmo tempo, garanta esse aspecto, mas também permita viabilidade de análise. A granularidade definida foi a de bairros de um município em particular.

Dados do Programa das Nações Unidas para o Desenvolvimento (2010) apontaram Florianópolis, em Santa Catarina (SC), como a capital brasileira com o melhor Índice de Desenvolvimento Humano (IDH) do país (0,847). A Associação Catarinense de Tecnologia (2018) alega que a cidade é responsável por cerca de 4% do faturamento total do setor de tecnologia nacional, com a participação da tecnologia no PIB (14%) superior à média brasileira. Fatores de destaque como esses garantem demanda de moradia na cidade.

Ademais, geograficamente, Florianópolis é uma ilha, e a maior parte da cidade é área de Preservação Ambiental, com menor percentual destinado a construções. A escassez de espaço e a alta demanda tornam o seu mercado imobiliário aquecido. Mesmo com a pandemia, o índice FipeZap (2020) aponta a capital catarinense como destaque nacional, com 5,82% de aumento no preço médio dos imóveis residenciais ao longo de 2020. Isso faz com que Florianópolis seja uma das melhores cidades para se investir em imóveis residenciais no Brasil atualmente, e será o foco deste projeto.

A análise será realizada com dados seccionais de uma data específica, apresentada na etapa da coleta de dados.

## 2. Coleta de Dados

Os dados foram obtidos através de técnicas de *web scraping* de websites que anunciam imóveis para venda com scripts em Python no dia 31 de janeiro de 2021.

Os websites escolhidos são dois dos três mais relevantes, segundo os resultados dos buscadores Google e Bing, para o termo “venda de imóveis em Florianópolis-SC”: vivareal.com.br e imovelweb.com.br. O zapimoveis.com.br foi excluído por fazer parte do Grupo Zap, que também detém o vivareal.com.br e já possui integração de bases de dados.

Restringiu-se a análise em imóveis já construídos, excluindo-se os na planta e em construção, que possuem especificação irregular. Somente os imóveis residenciais – apartamentos e casas – foram contemplados.

Os critérios utilizados para selecionar quais dados coletar foram a presença e viabilidade para *scraping* em ambos os websites de anúncios e a relevância para as hipóteses de predição.

Os dados são descritos com maiores detalhes no **Quadro 1**.

**Quadro 1** – Detalhamento dos dados capturados com *web scraping*.

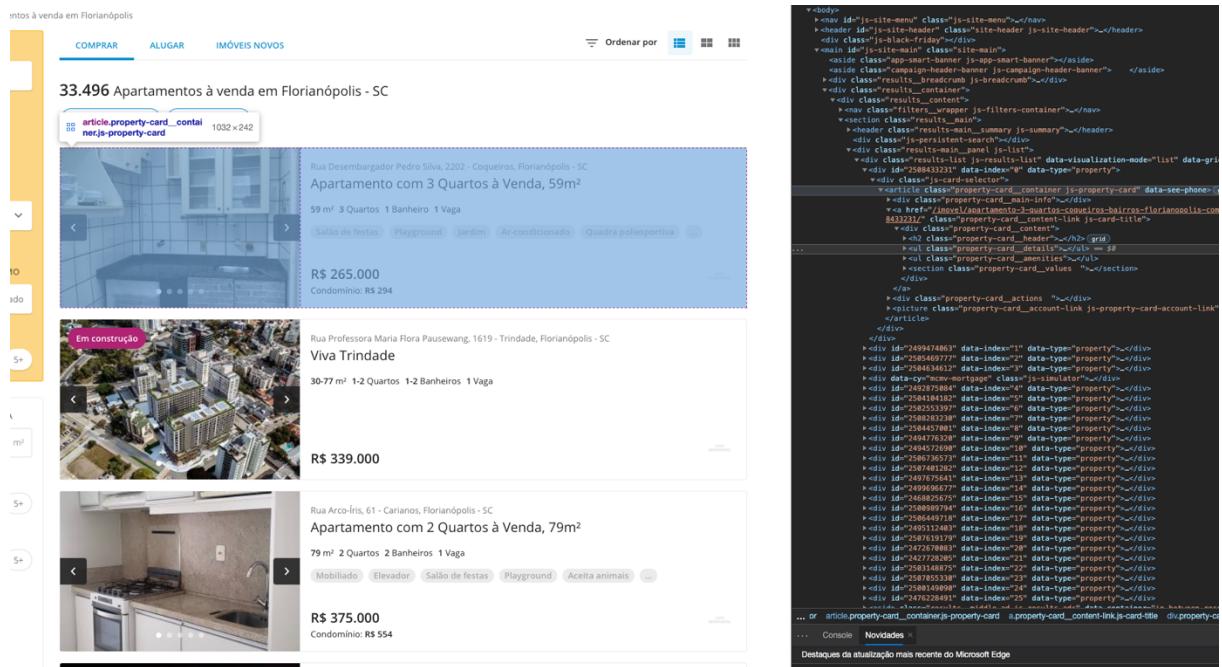
Campo	Descrição	Tipo
Tipo	Tipo de imóvel residencial (Apartamento, Cobertura, Casa, Casa em condomínio)	Texto
Área (m <sup>2</sup> )	Área do imóvel, em metros quadrados	Número
Quartos	Número de quartos do imóvel	Número
Banheiros	Número de banheiros do imóvel	Número
Vagas de garagem	Número de vagas de garagem do imóvel	Número
Preço (R\$)	Preço listado (em reais) no anúncio do imóvel	Número

Fonte: Autor.

## 2.1. Web Scraping do vivareal.com.br

Para cada dado de interesse, inspecionou-se o elemento em que estava contido através das ferramentas de desenvolvedor do navegador Chrome. Os tipos, as hierarquias, os ids e as classes das tags HTML foram essenciais para a obtenção da informação de maneira específica o suficiente para garantir a consistência da captura.

**Figura 1 – Inspeção dos elementos HTML na página de listagem do vivareal.com.br.**



Fonte: vivareal.com.br.

A captura foi realizada com o auxílio do módulo BeautifulSoup do Python. O código-fonte completo do *script* encontra-se no **Apêndice A**.

O vivareal.com.br possui um carregamento dinâmico das informações de cada página, que é dependente de sessão e interações em Javascript com a página. Requisições isoladas para a URL de cada página de listagem não provocam o carregamento das informações. Por isso, foi utilizado um *web driver* da ferramenta de testes Selenium, para simular as interações. Cada página de listagem é percorrida através de cliques automatizados no botão de “próxima página” e as informações capturadas vão incrementando um dicionário Python. Quando o clique

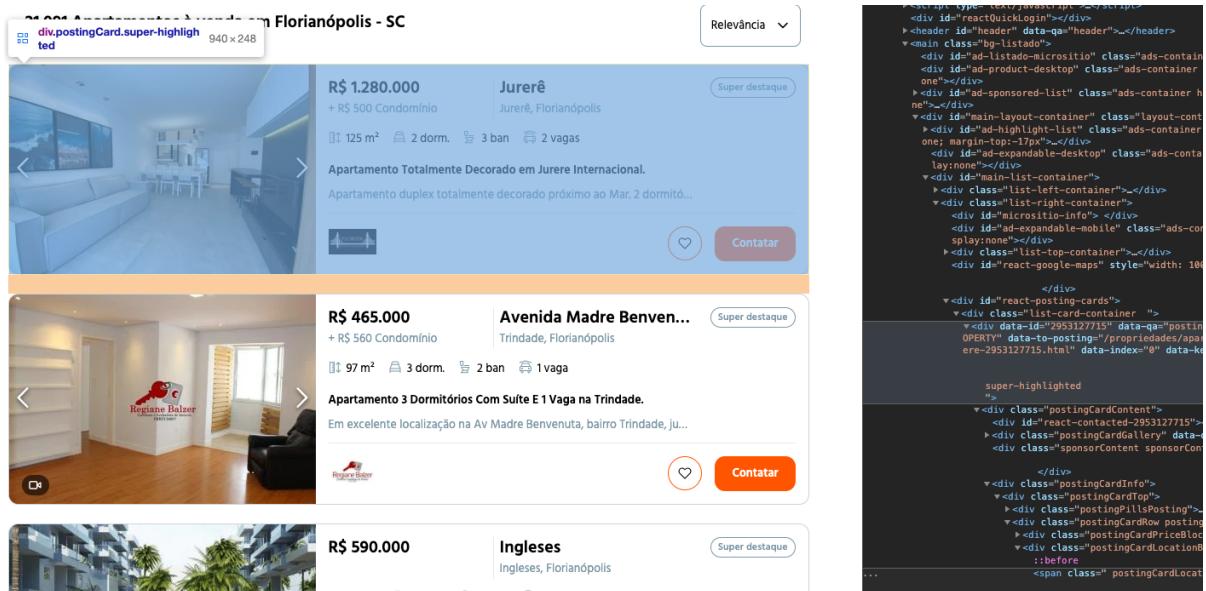
falha, significa que o botão não está mais disponível (última página) e o ciclo de captura é interrompido.

Por fim, o dicionário Python é convertido em *dataframe* do módulo Pandas e as informações são exportadas em um arquivo TSV.

## 2.2. Web Scraping do imovelweb.com.br

A captura dos dados do imovelweb.com.br é semelhante à do vivareal.com.br e também foi feita com o auxílio do módulo BeautifulSoup.

**Figura 2 – Inspeção dos elementos HTML na página de listagem do imovelweb.com.br.**



Fonte: imovelweb.com.br.

O código-fonte completo do *script* de *web scraping* do imovelweb.com.br encontra-se no **Apêndice B**.

Nesse caso, não foi necessário utilizar o Selenium. O *script* executa ciclos de requisições percorrendo todas as páginas de listagem e anúncio disponíveis, e adicionando os dados relevantes em um dicionário Python.

Após diversos testes, foi constatado que o botão de “próxima página” da página de listagem é inconsistente, e o imovelweb.com.br parece continuar mostrando imóveis repetidos. Por isso, o critério de encerramento dos ciclos de

captura foi detectar quantos imóveis já foram obtidos dos previstos pela contagem de resultados do próprio imovelweb.com.br.

O dicionário Python também é convertido em *dataframe* do módulo Pandas e as informações exportadas em um arquivo TSV.

### 3. Tratamento de Dados

O tratamento dos dados foi realizado com um *script* em Python. O conjunto de dados do vivareal.com.br apresentou 26.444 registros, enquanto o do imovelweb.com.br apresentou 29.128. O primeiro passo foi a realização de concatenação de ambos, totalizando **55.572 registros**.

**Figura 3** – Carregamento e concatenação.

```
import pandas as pd

# Carregamento

vr_df = pd.read_csv('/content/vivareal.tsv', sep='\t', dtype={
    'tipo': 'string',
    'bairro': 'string',
    'área (m2)': 'float',
    'quartos': 'float',
    'banheiros': 'float',
    'vagas de garagem': 'float',
    'preço': 'float'
})

vr_df_start_len = len(vr_df)
print('vivareal - registros: ', vr_df_start_len)

iw_df = pd.read_csv('/content/imovelweb.tsv', sep='\t', dtype={
    'tipo': 'string',
    'bairro': 'string',
    'área (m2)': 'float',
    'quartos': 'float',
    'banheiros': 'float',
    'vagas de garagem': 'float',
    'preço': 'float'
})

iw_df_start_len = len(iw_df)
print('imovelweb - registros: ', iw_df_start_len)

# Concatenação

df = pd.concat([vr_df, iw_df])
df_start_len = len(df)
print('total registros: ', df_start_len)
```

Fonte: Autor.

O **Quadro 2** sumariza os problemas de qualidade encontrados no conjunto de dados consolidado, os quais são explanados a seguir.

**Quadro 2 – Problemas de qualidade no conjunto de dados.**

Problema	Registros	
Duplicidades	12.502	
Ausências	Área (m <sup>2</sup> )	223
	Quartos	1
	Banheiros	1.491
	Vagas de Garagem	2.064
Variações	Bairro	2.422

Fonte: Autor.

### 3.1. Duplicidades

O conjunto de dados apresentou cerca de **22,49%** de seus registros iniciais duplicados. Tendo em mente que as tarefas de predição envolverão algoritmos de regressão, optou-se por remover as duplicidades, uma vez que poderiam influenciar no erro padrão dos coeficientes. Restaram ainda **43.070** registros únicos após a remoção.

**Figura 4 – Remoção das duplicidades.**

```
#...
df = df.drop_duplicates(ignore_index=True)
print('registros duplicados: ', df_start_len - len(df))
```

Fonte: Autor.

### 3.2. Ausências

O único registro com o campo “quartos” ausente foi removido, independentemente do mecanismo de ausência presumido, porque sua falta não trará impacto significante à análise.

**Figura 5 – Remoção do registro com “quartos” ausente.**

```
#...
df = df[~df['quartos'].isna()]
```

Fonte: Autor.

Os registros com o campo “vagas de garagens” ausentes foram preenchidos com zero, pois não parece ter um mecanismo de ausência, e sim um desalinhamento de parametrização, uma vez que os websites anunciantes não trazem o valor zero para vagas de garagem, quando inexistentes.

**Figura 6** – Preenchimento dos registros com “vagas de garagem” ausente.

```
#...
df['vagas de garagem'] = df['vagas de garagem'].fillna(0)
```

Fonte: Autor.

Nos registros com o campo “área (m<sup>2</sup>)” vazio, optou-se pela utilização do método *Multivariate Imputation by Chained Equation* (MICE), com imputação do campo “área útil” a partir dos campos “quartos”, “banheiros” e “vagas de garagem”. O mecanismo de ausência presumido é informativo (NMAR – *Not Missing At Random*), e pode trazer mais viés à análise se não for abordado.

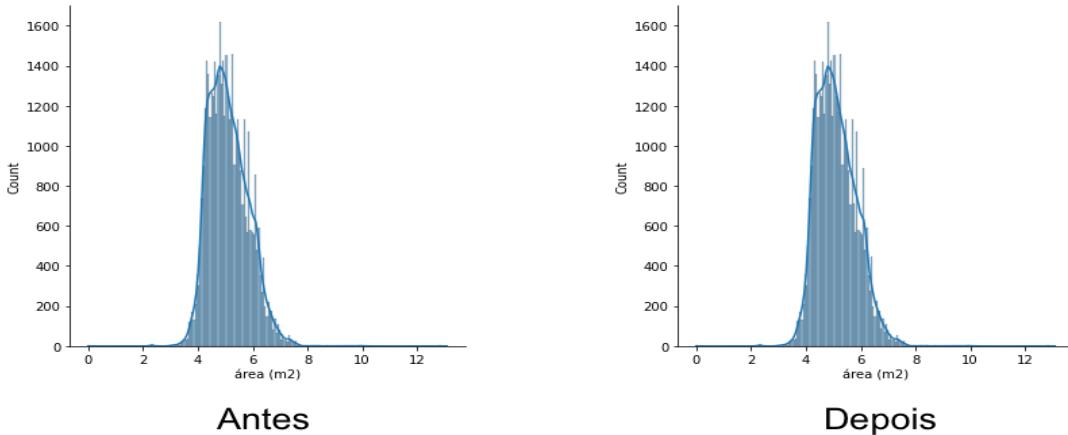
**Figura 7** – Imputação de área útil com o método MICE.

```
from impute import mice
#...
imputed = mice(df[['área (m2)', 'quartos', 'banheiros', 'vagas de garagem']].values)
mice_area = imputed[:, 0]
df['área (m2)'] = np.around(np.abs(mice_area))
```

Fonte: Autor.

A análise dos histogramas de antes e depois da imputação, conforme na **Figura 8**, ajuda a visualizar se a técnica inseriu viés nas distribuições, o que não parece ter sido o caso.

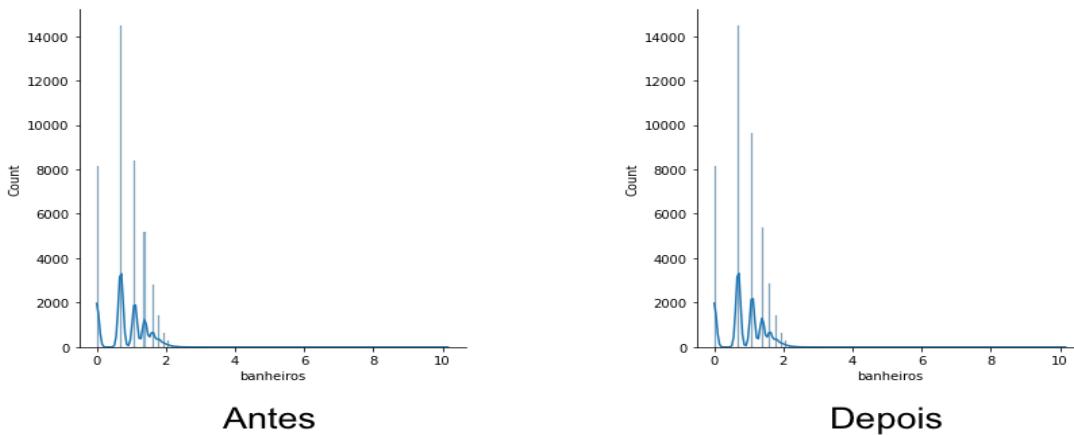
**Figura 8 – Histogramas de área ( $m^2$ ) em escala logarítmica antes e depois da imputação.**



Fonte: Autor.

Uma transformação logarítmica foi aplicada para que as distribuições se tornassem mais evidentes à plotagem. O método MICE também foi aplicado para preencher os registros com o campo “banheiros” ausente, pois também se presume um mecanismo NMAR.

**Figura 9 – Histogramas de banheiros em escala logarítmica antes e depois da imputação.**



Fonte: Autor.

### 3.3. Variações

O campo “bairro” é um registro qualitativo com risco de variação, uma vez que o mesmo bairro poderia ter diferentes nomes em distintos anúncios ou websites, ou nomes iguais, mas preenchidos com capitalização ou acentuação desigual. Para

identificar essa possibilidade, foi feita uma inspeção de todos os valores únicos para este campo, com ordenação alfabética, facilitando a identificação das variações.

**Figura 10** – Inspeção de variações no campo “bairro”.

```
# ...  
  
neighborhoods = set()  
for idx, neighborhood in df['bairro'].items():  
    neighborhoods.add(neighborhood)  
print(sorted(neighborhoods))
```

Fonte: Autor.

Ao todo, **2.422** variações estavam presentes e foram resolvidos com o script apresentado no **Apêndice C**.

## 4. Análise e Exploração dos Dados

### 4.1. Estatística Descritiva

A partir do tratamento dos dados, restaram **43.069 registros**.

**Figura 11** – Estatística descritiva das variáveis numéricas.

```

import pandas as pd

df = pd.read_csv('/content/dataset.tsv', sep='\t')
df['id'] = df.index + 1

print(df.info())

sts_columns = ['Atributo', 'Mín.', 'Máx.', 'μ', 'M', 'σ']
sts_rows = []

for c in df.columns:
    if c in ('tipo', 'bairro'):
        continue

    sts_rows.append([
        c,
        round(df[c].min(), 2),
        round(df[c].max(), 2),
        round(df[c].mean(), 2),
        round(df[c].median(), 2),
        round(df[c].std(), 2)
    ])

sts_df = pd.DataFrame(sts_rows, columns=sts_columns)
display(sts_df)

```

Fonte: Autor.

**Tabela 1** – Sumarização da estatística descritiva das variáveis numéricas.

Atributo	Mínimo	Máximo	μ	M	σ
Área (m <sup>2</sup> )	0,00	482.190,0	311,34	153,00	5.014,56
Quartos	1,00	34,00	3,01	3,00	1,23
Banheiros	1,00	25.800,00	3,37	2,00	124,32
Vagas de garagem	0,00	1011,00	2,21	2,00	8,52
Preço (R\$)	15.000,00	37.000.000,00	1.306.005,64	800.000,00	1.693.048,68

Fonte: Autor.

Chama a atenção o alto grau de dispersão das distribuições de área útil, banheiros e preço, além dos valores máximos inverossímeis para imóveis residenciais de área útil, banheiros e vagas de garagem.

A **Tabela 2** exibe um ranking decrescente dos preços médios para cada tipo de imóvel.

**Tabela 2** – Preço médio e frequência para cada tipo de imóvel.

<b>Tipo</b>	<b>Frequência</b>	<b>Preço Médio (R\$)</b>
Casa de condomínio	2.465	2.428.250,00
Casa	13.603	1.632.653,00
Cobertura	6.347	1.599.763,00
Apartamento	20.654	866.662,60

Fonte: Autor.

A **Tabela 3** exibe os dez bairros mais caros, em termos de preço médio, e a **Tabela 4** exibe os dez bairros com mais anúncios.

**Tabela 3** – Preços médios dos dez bairros mais caros.

<b>Tipo</b>	<b>Preço Médio (R\$)</b>
Cacupé	4.232.265,00
Jurerê Internacional	4.165.743,00
Beira Mar Norte	3.630.729,00
Lagoa da Conceição	1.942.360,00
João Paulo	1.940.053,00
Porto da Lagoa	1.939.506,00
Praia da Lagoinha	1.835.714,00
Jardim Anchieta	1.833.739,00
Agronômica	1.802.626,00
Jurerê	1.779.507,00

Fonte: Autor.

**Tabela 4** – Frequências dos dez bairros com mais anúncios.

<b>Tipo</b>	<b>Preço Médio (R\$)</b>
Ingleses do Rio Vermelho	6.165
Centro	4.242
Jurerê Internacional	2.545
Campeche	2.373
Itacorubi	2.324
Ingleses	1.982
Jurerê	1.721
Agronômica	1.563

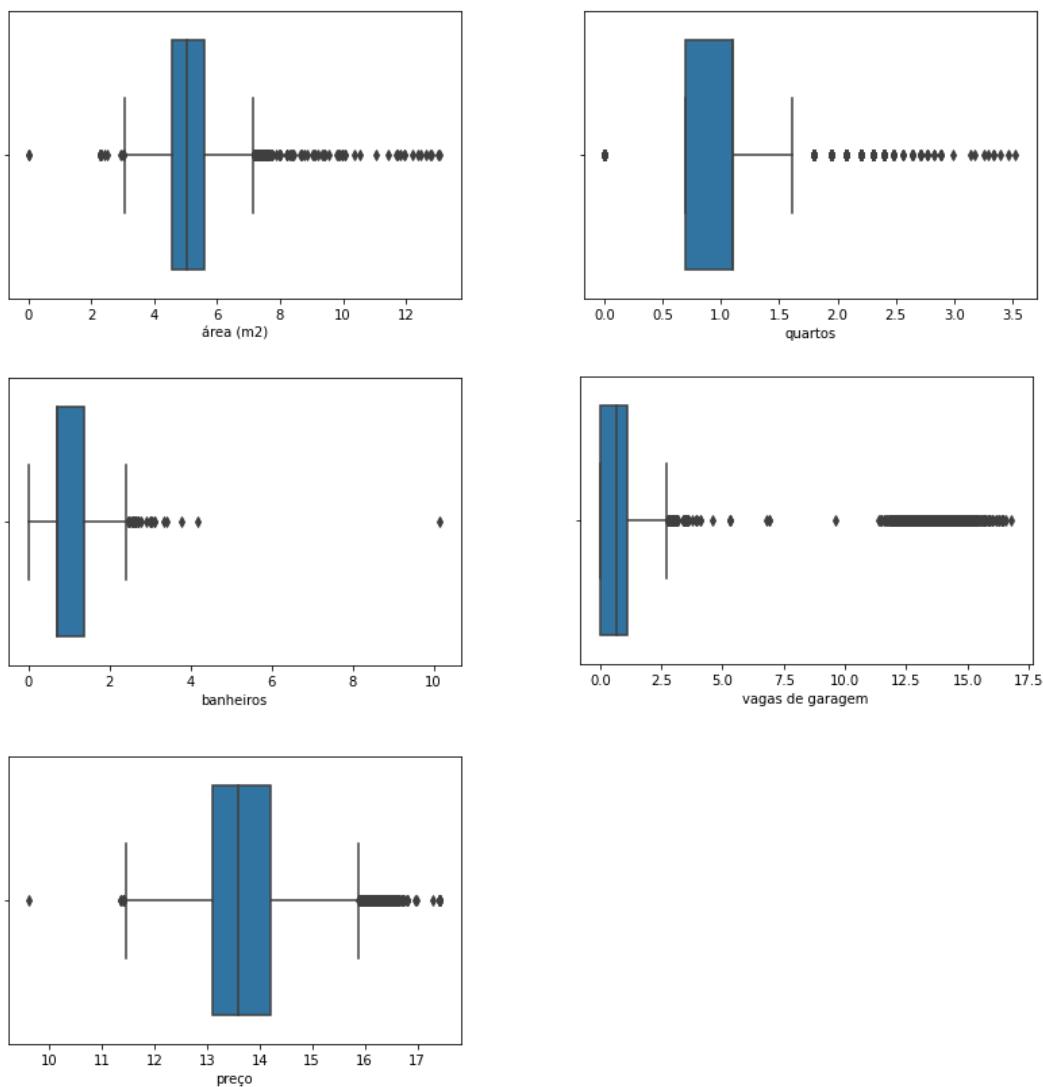
Canasvieiras	1.553
Estreito	1.518

Fonte: Autor.

#### 4.2. Dados Discrepantes (Outliers)

A **Figura 12** contém diagramas de caixa das variáveis numéricas após transformação logarítmica, para atenuar o efeito da variância na visualização dos *outliers*. Visualmente, constatam-se pontos de dados discrepantes em todos. Deve-se ter isso em mente no planejamento dos algoritmos de regressão, uma vez que alguns são mais sensíveis à presença de *outliers*.

**Figura 12** – Diagramas de caixa das variáveis numéricas.



Fonte: Autor.

Os diagramas foram gerados por instruções de código em Python, com o auxílio do módulo Seaborn.

**Figura 13 – Instruções para geração dos diagramas de caixa.**

```
● ● ●

import seaborn as sns

display(sns.boxplot(
    x=np.log(df['área (m²)'], where=(df['área (m²)'] != 0)))
)

display(sns.boxplot(
    x=np.log(df['quartos'], where=(df['quartos'] != 0)))
)

display(sns.boxplot(
    x=np.log(df['banheiros'], where=(df['banheiros'] != 0)))
)

display(sns.boxplot(
    x=np.log(df['vagas de garagem'], where=(df['vagas de garagem'] != 0)))
)

display(sns.boxplot(
    x=np.log(df['preço'], where=(df['preço'] != 0)))
)
```

Fonte: Autor.

### 4.3. Testes de Hipóteses

Antes de realizar os testes de hipóteses para verificar associação entre as variáveis independentes (atributos) com a dependente (preço), realizou-se o teste de D'Agostino-Pearson para detectar se a distribuição é normal.

**Figura 14 – Teste de D'Agostino-Pearson.**

```
● ● ●

from scipy import stats

# ...

k2, p = stats.normaltest(df['preço'])

print('normaltest k2: ', k2)
print('normaltest p', p)

# ...
```

Fonte: Autor.

O K<sup>2</sup> apresentou um valor alto, de 52.439,84, com um valor de p igual a 0,00 (< 0,05), o que refuta a hipótese nula de que a distribuição seria normal. Portanto, não cumpriria premissas de testes de hipóteses paramétricos.

O teste H de Kruskal-Wallis verifica significância estatística na diferença entre as distribuições de uma mesma variável quantitativa contínua ou qualitativa ordinal em três ou mais grupos, representados por uma variável qualitativa categórica.

Esse teste não é paramétrico e, portanto, adequa-se ao verificar as relações entre os atributos “tipo” e “bairro” com o preço.

**Figura 15 – Teste H de Kruskal-Wallis.**

```
● ● ●

from scipy import stats
import pandas as pd

# ...

type_codes, type_labels = pd.factorize(df['tipo'])
print(stats.kruskal(type_codes, df['preço']))

neighborhoods_codes, neighborhoods_labels = pd.factorize(df['bairro'])
print(stats.kruskal(neighborhoods_codes, df['preço']))

# ...
```

Fonte: Autor.

Conforme demonstra a **Figura 15**, antes do teste foi preciso fatorar as classes das variáveis categóricas em valores numéricos únicos.

Os resultados são apresentados na **Tabela 5**. Ambos os atributos apresentaram associação estatisticamente significante com o preço.

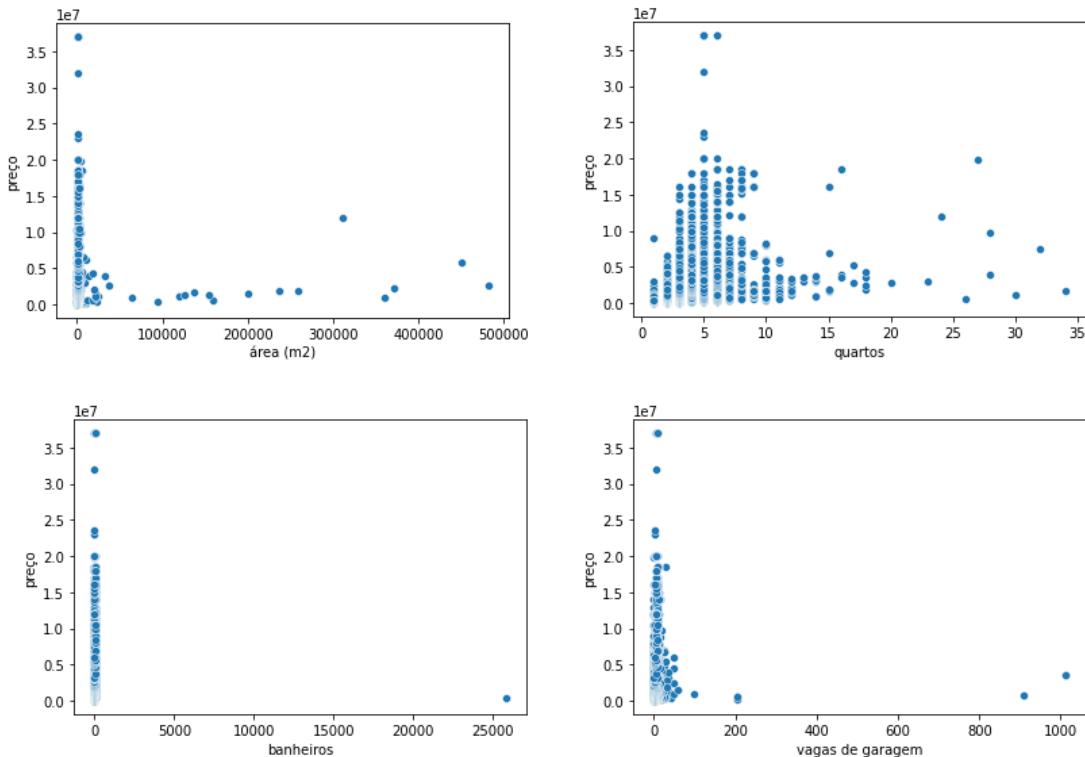
**Tabela 5 – Resultados dos testes de hipóteses não-paramétricos para cada atributo.**

Atributo	H	p
Tipo	65.797,17	< 0,05
Bairro	64.642,47	< 0,05

Fonte: Autor.

A **Figura 16** contém os diagramas de dispersão de todos os atributos numéricos em par com o preço. Visualmente, os atributos não parecem manter correlação linear com o preço.

**Figura 16** – Diagramas de dispersão.



Fonte: Autor.

A correlação dos postos de Spearman avalia com que intensidade a relação entre duas variáveis pode ser descrita pelo uso de uma função monótona, seja a relação linear ou não.

**Figura 17** – Correlações de Spearman.

```
● ● ●
from scipy import stats

print('spearman área', stats.spearmanr(df['área (m2)'], df['preço']))
print('spearman quartos', stats.spearmanr(df['quartos'], df['preço']))
print('spearman banheiros', stats.spearmanr(df['banheiros'], df['preço']))
print('spearman vagas de garagem', stats.spearmanr(df['vagas de garagem'], df['preço']))
```

Fonte: Autor.

Os resultados dos testes são apresentados em seguida, na **Tabela 6**.

**Tabela 6** – Resultados dos testes de correlação para cada atributo.

Atributo	rho	p
Área (m <sup>2</sup> )	0,71	< 0,05
Quartos	0,61	< 0,05
Banheiros	0,62	< 0,05
Vagas de garagem	0,60	< 0,05

Fonte: Autor.

Todos os atributos numéricos apresentaram correlação estatisticamente significante com o preço, em diferentes magnitudes.

#### 4.4. Multicolinearidade

A existência de correlação linear significativa entre duas variáveis independentes de uma regressão pode aumentar o erro padrão dos coeficientes. Por isso, foi realizada uma verificação de multicolinearidade para avaliar se os atributos são ortogonais, isto é, independentes entre si. Para tanto, foi feito o cálculo do Fator de Inflação da Variância (FIV) de todos os atributos numéricos.

**Figura 18** – Cálculos de FIV para cada atributo.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
df_X_num = df[['área (m2)', 'quartos', 'banheiros', 'vagas de garagem']]

for i in range(df_X_num.shape[1]):
    print('FIV {}:'.format(df_X_num.columns[i]), variance_inflation_factor(df_X_num.values, i))
```

Fonte: Autor.

**Tabela 7** – Resultado do FIV de cada atributo.

Atributo	FIV
Área útil (m <sup>2</sup> )	1,00
Quartos	1,08
Suítes	1,00
Banheiros	1,88
Vagas de garagem	1,07

Fonte: Autor.

Conforme os resultados apresentados na **Tabela 7**, nenhum atributo apresentou multicolinearidade significativa ( $FIV > 5$ ).

## 5. Criação de Modelos de Aprendizado de Máquina

Foram experimentados diferentes algoritmos de regressão para a predição dos preços de imóveis residenciais a partir dos atributos de interesse. Para cada algoritmo, o melhor modelo foi selecionado de acordo com métricas que auxiliam avaliação de desempenho, capacidade de generalização e nível de otimização, através de ajustes de hiperparâmetros, quando aplicáveis.

### 5.1. Preparo dos dados

No preparo, os dados foram inicialmente divididos em dois conjuntos de dados, um contendo as entradas dos atributos selecionados (X), e outro contendo os rótulos da variável de saída (y). Os atributos selecionados são “tipo”, “bairro”, “área”, “quartos”, “banheiros” e “vagas de garagem”. A variável de saída é o preço.

**Figura 19** – Divisão entre entradas dos atributos e rótulos da variável de saída.

```
● ● ●

import pandas as pd

df = pd.read_csv('/content/dataset.tsv', sep='\t')

X = df[['tipo', 'bairro', 'área (m²)', 'quartos', 'banheiros', 'vagas de garagem']]
y = df[['preço']]
```

Fonte: Autor.

Os modelos de regressão demandam uma tratativa para as variáveis qualitativas. Considerou-se criar variáveis fictícias (*dummy*), na qual cada possível categoria de uma variável qualitativa se torna uma nova variável binária, ou seja, preenchida com 0 ou 1, dependendo da categoria relacionada a cada observação.

A desvantagem desta abordagem é que se introduz multicolinearidade perfeita entre essas variáveis, um problema conhecido como a armadilha da variável fictícia (*dummy variable trap*). Para contorná-lo, pode-se omitir uma das variáveis, considerando-se a ausência de todas as outras como sua presença. É importante que se tenha a variável omitida em mente ao interpretar os coeficientes.

**Figura 20** – Geração das variáveis fictícias.

```
import pandas as pd

type_dummies = pd.get_dummies(X['tipo']).drop('Apartamento', axis=1)
neighborhood_dummies = pd.get_dummies(X['bairro']).drop('Tapera', axis=1)
X = X.drop(['tipo', 'bairro'], axis=1)
X = pd.merge(X, type_dummies, left_index=True, right_index=True)
X = pd.merge(X, neighborhood_dummies, left_index=True, right_index=True)
```

Fonte: Autor.

## 5.2. Validação Cruzada

A validação cruzada é uma técnica para avaliar a capacidade de generalização de um modelo. Para tal, foi definida uma função geradora que separa 80% dos registros em um subconjunto que servirá para treinamento dos algoritmos (treino) e os outros 20% em um subconjunto para testá-los (teste). A função gera cinco pares de subconjuntos aleatórios quando chamada, atenuando-se a potencial diferença de resultados que possa haver em subconjuntos com distribuição tão dispersa quanto a observada.

O desempenho dos algoritmos será avaliado levando-se em consideração a média das métricas obtidas para esses cinco pares de subconjuntos.

**Figura 21** – Função geradora de divisão entre subconjuntos de treino e teste.

```
from sklearn.model_selection import ShuffleSplit

def shuffled_split_gen(X, y, random_state=10):
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=random_state)

    for train_idx, test_idx in cv.split(X, y):
        X_train = X.iloc[train_idx, :]
        y_train = y.iloc[train_idx, :]
        X_test = X.iloc[test_idx, :]
        y_test = y.iloc[test_idx, :]

        yield X_train, y_train, X_test, y_test
```

Fonte: Autor.

### 5.3. Auxiliares

Foram definidas funções auxiliares para cálculo de distância de Mahalanobis para detecção e remoção de *outliers*. A distância de Mahalanobis é uma medida que pode ser calculada em um espaço multidimensional, mensurada relativa a um centroide nesse espaço. É útil para a detecção de *outliers* nas regressões múltiplas.

Além disso, foi desenvolvida uma classe auxiliar para demonstração das métricas e cálculo das médias na validação cruzada. O código-fonte encontra-se no **Apêndice D**.

### 5.4. Métricas

Essas métricas foram utilizadas para comparação entre os modelos, por serem padronizadas, de fácil interpretabilidade e já virem definidas como parâmetro de avaliação padrão dos módulos utilizados:

- **Coeficiente de determinação ( $R^2$ )**: representa a magnitude com que a variação da variável de saída é explicada pela variação dos atributos.
- **Raiz do erro quadrático médio (REQM)**: é o desvio padrão dos resíduos em relação aos valores preditos, representando o quanto em média estão distantes. A REQM é expressa na mesma unidade de valor da variável de saída – no caso, em R\$ –, podendo ser comparada ao desvio padrão da distribuição do preço.

As demais complementam os resultados da regressão do melhor modelo:

- **Erro absoluto médio (EAM)**: é a média das diferenças absolutas entre os valores preditos e os resíduos. O EAM também é expresso na mesma unidade de valor da variável de saída.

- **Erro quadrático médio (EQM)**: é a média das áreas dos quadrados formados a partir da distância dos valores preditos aos resíduos. Não é expresso na mesma unidade de valor da variável de saída.

## 5.5. Regressão Linear Múltipla

A regressão linear múltipla gera um modelo que determina coeficientes para cada variável independente - quantitativa ou qualitativa - em sua correlação linear com uma variável dependente quantitativa contínua, cujas estimativas são refletidas em uma reta de regressão. O treinamento foi conduzido com e sem os *outliers*, para comparação de desempenho.

**Figura 22** – Regressão linear múltipla.

```
● ● ●

from sklearn.linear_model import LinearRegression

X_, y_ = drop_outliers(X, y)

for name, X_i, y_i in (
    'Regressão linear múltipla com outliers',
    X,
    y
), (
    'Regressão linear múltipla sem outliers',
    X_,
    y_
):
    results = Results(name)

    for X_train, y_train, X_test, y_test in shuffled_split_gen(X_i, y_i):
        model = LinearRegression()

        model.fit(X_train, y_train)

        results.add(
            model,
            X_train,
            y_train,
            X_test,
            y_test
        )

    print(results.base_model_name)
    display(results.get_df())
    print('\n')
```

Fonte: Autor.

**Tabela 9** – Resultados dos modelos de regressão linear múltipla.

<b>Regressão Linear Múltipla</b>	<b>R<sup>2</sup> médio</b>		<b>REQM média</b>	
	<i>Treino</i>	<i>Teste</i>	<i>Treino</i>	<i>Teste</i>
Com <i>outliers</i>	0,44	0,44	1.281.297,09	18.297.941,56
Sem <i>outliers</i>	<b>0,50</b>	<b>0,49</b>	<b>1.205.737,89</b>	<b>1.224.583,68</b>

Fonte: Autor.

A remoção dos *outliers* melhorou a capacidade de generalização do modelo, aproximando as REQM de teste das do treino, reduziu a REQM e aumentou o valor do coeficiente de determinação médio.

## 5.6. Regularização de Lasso

A regressão com regularização é aquela em que se aplica uma penalidade com o objetivo de reduzir o erro de generalização de um modelo, isto é, atenuar o sobreajuste (*overfitting*).

No caso da regularização de Lasso (L1), a penalidade é a seleção de atributos, reduzindo o coeficiente de alguns para zero.

**Tabela 10** – Resultados dos modelos de regularização de Lasso.

<b>Hiperparâmetro</b>	<b>R<sup>2</sup> médio</b>		<b>REQM média</b>	
	<i>Treino</i>	<i>Teste</i>	<i>Treino</i>	<i>Teste</i>
alpha				
0,001	<b>0,49</b>	<b>0,46</b>	<b>1.205.737,89</b>	<b>1.224.583,68</b>
0,01	0,49	0,46	1.205.737,89	1.224.583,68
0,1	0,49	0,46	1.205.737,89	1.224.583,70
0,5	0,49	0,46	1.205.737,89	1.224.583,79
1	0,49	0,46	1.205.737,90	1.224.583,88
5	0,49	0,46	1.205.737,94	1.224.584,34

Fonte: Autor.

A regularização de Lasso não apresentou vantagens em relação à regressão linear múltipla.

**Figura 23** – Regularização de Lasso.

```

from sklearn.linear_model import Lasso

X_, y_ = drop_outliers(X, y)

r_columns = [
    'alpha', 'R2 treino', 'R2 teste', 'REQM Treino', 'REQM Teste', 'Algoritmo'
]
r_rows = []

for alpha in (0.001, 0.005, 0.01, 0.1, 0.5, 1, 5):
    results = Results('Lasso')

    for X_train, y_train, X_test, y_test in shuffled_split_gen(X_, y_):
        model = Lasso(alpha=alpha)

        model.fit(X_train, y_train)

        results.add(
            model,
            X_train,
            y_train,
            X_test,
            y_test
        )

    r2_train, r2_test = results.get_mean_r2()
    rmse_train, rmse_test = results.get_mean_rmse()
    r_rows.append([
        alpha, r2_train, r2_test, rmse_train, rmse_test, results.base_model_name
    ])

r_df = pd.DataFrame(r_rows, columns=r_columns)

```

Fonte: Autor.

## 5.7. Regularização de Ridge

A regularização de Ridge (L2) possui o encolhimento dos coeficientes como penalidade, o que inclusive diminui o impacto de multicolinearidade.

**Tabela 11** – Resultados dos modelos de regularização de Ridge.

Hiperparâmetro	R <sup>2</sup> médio		REQM média	
alpha	Treino	Teste	Treino	Teste
0,001	0,49	0,46	1.205.737,89	1.224.583,68
0,005	0,49	0,46	1.205.737,89	1.224.583,68
0,01	0,49	0,46	1.205.737,89	1.224.583,68
0,1	0,49	0,46	1.205.737,90	1.224.583,71
0,5	0,49	0,46	1.205.738,05	1.224.583,91

1	0,49	0,46	1.205.738,51	1.224.584,42
5	0,49	0,46	1.205.752,54	1.224.598,15

Fonte: Autor.

Assim como a regularização de Lasso, também não apresentou vantagens em relação à regressão linear múltipla.

**Figura 24** – Regularização de Ridge.



```

● ● ●

from sklearn.linear_model import Ridge
X_, y_ = drop_outliers(X, y)

r_columns = [
    'alpha', 'R² treino', 'R² teste', 'REQM Treino', 'REQM Teste', 'Algoritmo'
]
r_rows = []

for alpha in (0.001, 0.005, 0.01, 0.1, 0.5, 1, 5):
    results = Results('Ridge')

    for X_train, y_train, X_test, y_test in shuffled_split_gen(X_, y_):
        model = Ridge(alpha=alpha)

        model.fit(X_train, y_train)

        results.add(
            model,
            X_train,
            y_train,
            X_test,
            y_test
        )

    r2_train, r2_test = results.get_mean_r2()
    rmse_train, rmse_test = results.get_mean_rmse()
    r_rows.append([
        alpha, r2_train, r2_test, rmse_train, rmse_test, results.base_model_name
    ])

r_df = pd.DataFrame(r_rows, columns=r_columns)

```

Fonte: Autor.

## 5.8. Regularização Elastic-Net

A regularização Elastic-Net é uma combinação de Ridge com Lasso, aplicando ambas as penalidades, isto é, seleção e encolhimento de atributos, em proporção definida por hiperparâmetro.

**Figura 25** – Regularização Elastic-Net.

```

from sklearn.linear_model import ElasticNet

X_, y_ = drop_outliers(X, y)

r_columns = [
    'alpha', 'l1_ratio', 'R2 treino', 'R2 teste', 'REQM Treino', 'REQM Teste', 'Algoritmo'
]
r_rows = []

for l1_ratio in (0.3, 0.5, 0.7, 1):
    for alpha in (0.001, 0.005, 0.01, 0.1, 0.5, 1, 5):
        results = Results('Elastic-Net')

        for X_train, y_train, X_test, y_test in shuffled_split_gen(X_, y_):
            model = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=10)

            model.fit(X_train, y_train)

            results.add(
                model,
                X_train,
                y_train,
                X_test,
                y_test
            )

            r2_train, r2_test = results.get_mean_r2()
            rmse_train, rmse_test = results.get_mean_rmse()
            r_rows.append([alpha, l1_ratio, r2_train, r2_test, rmse_train, rmse_test,
                           results.base_model_name])

r_df = pd.DataFrame(r_rows, columns=r_columns)

```

Fonte: Autor.

**Tabela 12** – Resultados dos modelos de regularização Elastic-Net.

Hiperparâmetros	R <sup>2</sup> médio		REQM média	
	l1_ratio	alpha	Treino	Teste
0,3	0,001	0,49	0,46	1.206.008,29 1.224.837,84
0,3	0,01	0,48	0,46	1.214.111,46 1.232.270,59
0,3	0,1	0,44	0,41	1.268.624,03 1.283.630,55
0,3	1	0,37	0,35	1.343.241,02 1.354.224,01
0,5	0,001	0,49	0,46	1.205.884,92 1.224.722,74
0,5	0,01	0,49	0,46	1.211.314,89 1.229.709,85
0,5	0,1	0,45	0,42	1.256.633,28 1.272.161,04
0,5	1	0,38	0,36	1.333.894,79 1.345.686,66
0,7	0,001	0,49	0,46	1.205.794,49 1.224.637,91
0,7	0,01	0,49	0,46	1.208.562,10 1.227.189,86
0,7	0,1	0,46	0,43	1.240.605,40 1.256.939,07
0,7	1	0,39	0,37	1.319.379,74 1.332.163,38

1	<b>0,001</b>	<b>0,49</b>	<b>0,46</b>	<b>1.205.737,89</b>	<b>1.224.583,68</b>
1	0,01	0,49	0,46	1.205.737,89	1.224.583,68
1	0,1	0,49	0,46	1.205.737,89	1.224.583,70
1	1	0,49	0,46	1.205.737,90	1.224.583,88

Fonte: Autor.

A regularização Elastic-Net também não apresentou vantagens em relação à regressão linear múltipla.

## 5.9. Árvore de Decisão

Árvore de decisão (*decision tree*) é a aplicação de um algoritmo não-paramétrico que se utiliza de um conjunto de dados supervisionado e gera um fluxograma de nós condicionais para inferência indutiva. Esse algoritmo também pode ser utilizado para tarefas de regressão (árvore de regressão), e é bem aplicado para casos em que não se observa uma correlação linear.

**Tabela 13** – Resultados dos modelos de árvore de regressão.

<b>Hiperparâmetro</b>	<b>R<sup>2</sup> médio</b>		<b>REQM média</b>	
	<i>Treino</i>	<i>Teste</i>	<i>Treino</i>	<i>Teste</i>
<i>max_depth</i>				
	0,99	0,55	141.314,33	956.999,68
1	0,26	0,25	1.457.713,79	1.478.480,53
2	0,45	0,43	1.259.236,60	1.287.415,91
<b>3</b>	<b>0,58</b>	<b>0,54</b>	<b>1.101.738,84</b>	<b>1.124.459,87</b>
4	0,64	0,56	1.003.913,00	1.044.569,73
5	0,69	0,56	924.485,71	984.008,71
10	0,86	0,60	636.214,20	898.531,16
20	0,97	0,57	287.593,56	945.460,54
50	0,99	0,56	141.343,97	975.221,70
100	0,99	0,55	141.314,33	956.999,68
200	0,99	0,55	141.314,33	956.999,68
500	0,99	0,55	141.314,33	956.999,68
1000	0,99	0,55	141.314,33	956.999,68

Fonte: Autor.

**Figura 26** – Árvore de regressão.

```
from sklearn.tree import DecisionTreeRegressor

X_, y_ = drop_outliers(X, y)

r_columns = [
    'max_depth', 'R2 treino', 'R2 teste', 'REQM Treino', 'REQM Teste', 'Algoritmo'
]
r_rows = []

for max_depth in (None, 1, 2, 3, 4, 5, 10, 20, 50, 100, 200, 500, 1000):
    results = Results('Árvore de decisão')

    for X_train, y_train, X_test, y_test in shuffled_split_gen(X_, y_):
        model = DecisionTreeRegressor(
            max_depth=max_depth, criterion='mse', random_state=10
        )

        model.fit(X_train, y_train)

        results.add(
            model,
            X_train,
            y_train,
            X_test,
            y_test
        )

        r2_train, r2_test = results.get_mean_r2()
        rmse_train, rmse_test = results.get_mean_rmse()
        r_rows.append([
            max_depth, r2_train, r2_test, rmse_train, rmse_test, results.base_model_name
        ])

    r_df = pd.DataFrame(r_rows, columns=r_columns)
    display(r_df)
```

Fonte: Autor.

Conforme a profundidade máxima da árvore aumenta, menos generalizável se torna o modelo, mas apresenta resultados mais promissores do que a regressão linear múltipla.

## 5.10. Floresta Aleatória

A floresta aleatória (*random forest*) é uma combinação de árvores de decisão através do método de agrupamento (*ensemble*) do tipo *bagging*, no qual cada modelo é treinado de maneira paralela com um subconjunto aleatório dos dados.

**Tabela 14** – Resultados dos modelos de floresta aleatória.

Hiperparâmetros		R <sup>2</sup> médio		REQM média	
<i>max_depth</i>	<i>n_estimators</i>	<i>Treino</i>	<i>Teste</i>	<i>Treino</i>	<i>Teste</i>
-	50	0,96	0,75	318.335,42	752.256,61
-	100	0,97	0,75	311.778,66	748.072,21
-	200	0,97	0,76	309.986,86	746.370,37
-	500	0,97	0,75	305.319,15	748.417,44
-	1000	0,97	0,75	305.470,34	747.769,34
3	50	0,61	0,58	1.045.192,07	1.074.371,54
3	100	0,61	0,59	1.043.587,40	1.071.733,04
3	200	0,61	0,59	1.043.283,74	1.070.718,44
3	500	0,62	0,59	1.042.571,19	1.070.798,26
3	1000	0,62	0,59	1.042.759,01	1.070.686,05
4	50	0,67	0,60	967.135,85	1.005.593,65
4	100	0,67	0,60	966.379,08	1.005.197,70
4	200	0,67	0,61	965.919,29	1.000.724,85
4	500	0,67	0,61	965.749,80	1.002.223,63
4	1000	0,67	0,61	965.585,23	1.001.479,58
5	50	0,72	0,65	883.650,12	930.166,11
5	100	0,73	0,65	881.898,57	929.819,35
5	200	0,73	0,64	880.394,15	930.410,02
5	500	0,73	0,64	881.380,33	930.507,80
5	1000	0,73	0,65	880.479,90	930.381,07
10	50	0,87	0,73	602.926,24	789.121,24
10	100	0,87	0,73	601.265,51	782.792,85
10	200	0,87	0,74	600.462,93	780.413,29
10	500	0,87	0,73	600.293,91	783.951,60
10	1000	0,87	0,74	600.155,01	783.297,92
20	50	0,95	0,76	362.160,85	752.949,81
20	100	0,95	0,75	361.039,24	750.220,82
20	200	0,95	0,75	358.234,21	747.028,84
20	500	0,95	0,75	358.415,66	748.926,29
20	1000	0,95	0,75	357.226,59	749.186,81
30	50	0,96	0,75	325.835,01	755.192,48
30	100	0,96	0,75	320.344,73	745.876,06
30	200	0,96	0,75	314.293,17	747.007,26
30	500	0,96	0,75	313.797,66	748.137,86
30	1000	0,96	0,75	311.892,03	747.203,97

Fonte: Autor.

A floresta aleatória de árvores de regressão aumentou o coeficiente de determinação, reduziu a REQM, e ao mesmo tempo estreitou a diferença de desempenho entre os subconjuntos de treino e teste.

**Figura 27** – Floresta aleatória de árvores de regressão.

```
from sklearn.ensemble import RandomForestRegressor

X_, y_ = drop_outliers(X, y)

r_columns = [
    'max_depth', 'n_estimators', 'R2 treino', 'R2 teste', 'REQM Treino', 'REQM Teste',
    'Algoritmo'
]
r_rows = []

for max_depth in (None, 3, 4, 5, 10, 20, 30):
    for n_estimators in (50, 100, 200, 500, 1000):
        results = Results('Floresta aleatória')

        for X_train, y_train, X_test, y_test in shuffled_split_gen(X_, y_):
            model = RandomForestRegressor(
                n_estimators=n_estimators,
                max_depth=max_depth
            )

            model.fit(X_train, y_train.values.ravel())

            results.add(
                model,
                X_train,
                y_train,
                X_test,
                y_test
            )

            r2_train, r2_test = results.get_mean_r2()
            rmse_train, rmse_test = results.get_mean_rmse()
            r_rows.append([
                max_depth, n_estimators, r2_train, r2_test, rmse_train, rmse_test,
                results.base_model_name
            ])

r_df = pd.DataFrame(r_rows, columns=r_columns)
display(r_df)
```

Fonte: Autor.

## 5.11. Árvore de Decisão Aumentada

A árvore de decisão aumentada é uma combinação de árvores de decisão através do método de agrupamento (*ensemble*) do tipo aceleração (*boosting*), no

qual cada modelo é treinado de maneira sequencial e o sucessor aprende com os erros do antecessor.

**Tabela 15** – Resultados dos modelos de árvore de decisão aumentada.

max_depth	Hiperparâmetros			R <sup>2</sup> médio		REQM média	
	n_estimators	reg_lambda	learning_ratio	Treino	Teste	Treino	Teste
3	50	0	0,001	-0,50	-0,52	2.071.770,21	2.084.539,12
3	50	0	0,01	0,15	0,15	1.558.686,14	1572.863,96
3	50	0	0,1	0,75	0,72	840.593,94	862.014,09
3	50	1	0,001	-0,50	-0,52	2.071.933,19	2084.709,70
3	50	1	0,01	0,15	0,15	1.560.189,74	1574.606,61
3	50	1	0,1	0,75	0,71	843.344,36	864.986,66
3	50	2	0,001	-0,50	-0,52	2.072.113,76	2084.890,32
3	50	2	0,01	0,15	0,14	1.561.470,59	1576.105,60
3	50	2	0,1	0,75	0,71	846.522,93	870.340,86
3	50	3	0,001	-0,50	-0,52	2.072.282,09	208.5064,28
3	50	3	0,01	0,15	0,14	1.562.607,70	1577.168,38
3	50	3	0,1	0,75	0,72	849.295,03	868.839,56
3	50	4	0,001	-0,50	-0,52	2.072.452,83	2085.243,17
3	50	4	0,01	0,15	0,14	1.563.742,08	1578.297,43
3	50	4	0,1	0,75	0,71	849.205,88	871.914,20
3	100	0	0,001	-0,39	-0,42	2.000.301,76	2013.013,51
3	100	0	0,01	0,45	0,44	1.245.791,11	1263.083,54
3	100	0	0,1	0,78	0,74	790.443,51	824.923,65
3	100	1	0,001	-0,39	-0,42	2.000.630,94	2013.345,90
3	100	1	0,01	0,45	0,44	1.248.045,92	1265.931,43
3	100	1	0,1	0,78	0,74	795.328,85	825.609,91
3	100	2	0,001	-0,39	-0,42	2.000.953,14	2013.673,55
3	100	2	0,01	0,45	0,44	1.250.236,08	1268.217,43
3	100	2	0,1	0,78	0,73	799.059,59	830.380,69
3	100	3	0,001	-0,40	-0,42	2.001.263,47	2013.996,18
3	100	3	0,01	0,45	0,44	1.252.313,54	1270.313,31
3	100	3	0,1	0,77	0,74	802.136,22	828.707,41
3	100	4	0,001	-0,40	-0,42	2.001.575,18	2.014.319,00
3	100	4	0,01	0,45	0,44	1.254.070,21	1.272.381,76
3	100	4	0,1	0,77	0,73	803.519,27	831.435,64
4	50	0	0,001	-0,49	-0,51	2.066.858,78	2.078.715,12
4	50	0	0,01	0,19	0,19	1.520.189,37	1.531.190,82
4	50	0	0,1	0,79	0,74	769.610,54	819.351,29
4	50	1	0,001	-0,49	-0,51	2.067.559,46	2.079.993,86
4	50	1	0,01	0,19	0,18	1.524.382,45	1.536.960,14
4	50	1	0,1	0,78	0,73	778.456,13	824.621,08

4	50	2	0,001	-0,49	-0,52	2.068.139,29	2.080.624,92
4	50	2	0,01	0,18	0,18	1.527.934,86	1.541.289,32
4	50	2	0,1	0,79	0,73	781.414,26	826.638,34
4	50	3	0,001	-0,49	-0,52	2.068.615,45	2.081.177,44
4	50	3	0,01	0,18	0,18	1.530.921,32	1.544.672,32
4	50	3	0,1	0,78	0,73	785.028,76	825.794,14
4	50	4	0,001	-0,49	-0,52	2.068.999,05	2.081.577,80
4	50	4	0,01	0,18	0,18	1.533.623,23	1.547.502,88
4	50	4	0,1	0,78	0,74	784.084,94	822.206,62
4	100	0	0,001	-0,38	-0,40	1.990.333,98	2.001.511,63
4	100	0	0,01	0,51	0,49	1.186.329,32	1.206.106,45
4	100	0	0,1	0,82	0,76	720.061,75	781.781,90
4	100	1	0,001	-0,38	-0,40	1.991.647,38	2.003.707,70
4	100	1	0,01	0,50	0,48	1.192.177,07	1.210.994,58
4	100	1	0,1	0,81	0,75	730.858,74	788.265,04
4	100	2	0,001	-0,38	-0,40	1.992.719,95	2.005.005,68
4	100	2	0,01	0,50	0,48	1.196.843,91	1.215.887,16
4	100	2	0,1	0,81	0,75	738.052,22	794.948,95
4	100	3	0,001	-0,39	-0,41	1.993.632,45	2.005.946,96
4	100	3	0,01	0,49	0,48	1.200.944,93	1.219.506,64
4	100	3	0,1	0,81	0,75	742.151,82	792.754,83
4	100	4	0,001	-0,39	-0,41	1.994.363,28	2.006.816,18
4	100	4	0,01	0,49	0,48	1.204.807,98	1.223.386,81
4	100	4	0,1	0,81	0,75	740.204,11	790.230,06
5	50	0	0,001	-0,48	-0,51	2.063.358,16	2.075.516,70
5	50	0	0,01	0,23	0,20	1.485.585,51	1.501.893,77
5	50	0	0,1	0,82	0,74	703.548,33	798.259,58
5	50	1	0,001	-0,49	-0,51	2.064.841,62	2.077.695,08
5	50	1	0,01	0,22	0,20	1.495.703,75	1.511.777,84
5	50	1	0,1	0,82	0,73	713.551,53	794.549,63
5	50	2	0,001	-0,49	-0,51	2.065.861,08	2.078.409,27
5	50	2	0,01	0,21	0,20	1.503.414,79	1.520.714,87
5	50	2	0,1	0,82	0,74	718.567,10	792.786,34
5	50	3	0,001	-0,49	-0,51	2.066.623,42	2.079.100,75
5	50	3	0,01	0,20	0,19	1.509.436,33	1.526.653,49
5	50	3	0,1	0,82	0,74	721.834,53	791.605,46
5	50	4	0,001	-0,49	-0,52	2.067.273,37	2.080.017,64
5	50	4	0,01	0,20	0,19	1.514.425,66	1.531.345,57
5	50	4	0,1	0,81	0,73	726.041,06	791.232,51
5	100	0	0,001	-0,37	-0,40	1.983.351,02	1.995.024,46
5	100	0	0,01	0,55	0,50	1.128.142,10	1.157.244,06
5	100	0	0,1	0,84	0,76	659.627,76	773.452,53
5	100	1	0,001	-0,38	-0,40	1.986.175,96	1.999.283,72
5	100	1	0,01	0,54	0,50	1.142.414,57	1.169.298,33
5	100	1	0,1	0,84	0,75	670.404,05	769.876,07

5	100	2	0,001	-0,38	-0,40	1.988.125,61	2.000.790,35
5	100	2	0,01	0,53	0,49	1.152.904,83	1.180.276,95
5	100	2	0,1	0,84	0,75	676.201,90	768.179,62
5	100	3	0,001	-0,38	-0,40	1.989.582,75	2.001.827,63
5	100	3	0,01	0,53	0,49	1.161.043,18	1.188.458,15
5	100	3	0,1	0,83	0,75	682.027,85	766.478,80
5	100	4	0,001	-0,38	-0,40	1.990.848,47	2.003.601,38
5	100	4	0,01	0,52	0,49	1.168.339,65	1.195.206,67
5	100	4	0,1	0,83	0,75	686.360,75	764.587,05
10	50	0	0,001	-0,47	-0,50	2.052.832,84	2.068.356,92
10	50	0	0,01	0,33	0,26	1.388.154,59	1.440.226,24
10	50	0	0,1	0,94	0,70	413.433,82	767.176,56
10	50	1	0,001	-0,48	-0,51	2.059.514,75	2.073.450,32
10	50	1	0,01	0,28	0,23	1.439.263,40	1.476.106,34
10	50	1	0,1	0,93	0,72	458.154,08	765.888,21
10	50	2	0,001	-0,48	-0,51	2.061.822,06	2.075.283,63
10	50	2	0,01	0,26	0,23	1.460.585,90	1.489.747,46
10	50	2	0,1	0,92	0,71	484.568,48	774.720,91
10	50	3	0,001	-0,48	-0,51	2.063.271,42	2.076.357,33
10	50	3	0,01	0,24	0,22	1.472.771,84	1.499.305,39
10	50	3	0,1	0,91	0,71	505.061,33	763.446,75
10	50	4	0,001	-0,49	-0,51	2.064.363,23	2.077.671,22
10	50	4	0,01	0,23	0,22	1.482.257,87	1.507.182,34
10	50	4	0,1	0,90	0,72	518.442,75	760.071,20
10	100	0	0,001	-0,34	-0,37	1.962.373,75	1.980.818,67
10	100	0	0,01	0,68	0,56	951.447,12	1.067.703,70
10	100	0	0,1	0,95	0,70	368.011,01	761.658,99
10	100	1	0,001	-0,36	-0,39	1.975.357,41	1.990.599,90
10	100	1	0,01	0,63	0,53	1.024.221,81	1.105.944,76
10	100	1	0,1	0,94	0,73	413.424,96	758.238,00
10	100	2	0,001	-0,37	-0,39	1.979.943,75	1.994.384,63
10	100	2	0,01	0,61	0,53	1.060.707,82	1.125.275,29
10	100	2	0,1	0,93	0,72	437.448,68	768.031,47
10	100	3	0,001	-0,37	-0,39	1.982.720,74	1.996.543,88
10	100	3	0,01	0,59	0,53	1.082.343,52	1.136.929,38
10	100	3	0,1	0,93	0,72	451.734,58	757.198,67
10	100	4	0,001	-0,37	-0,40	1.984.938,70	1.998.827,82
10	100	4	0,01	0,58	0,53	1.097.730,36	1.148.388,37
10	100	4	0,1	0,92	0,73	462.723,99	752.368,11

Fonte: Autor.

Árvore de decisão aumentada melhorou o coeficiente de determinação e reduziu a REQM.

**Figura 28** – Árvore de decisão aumentada.

```

● ● ●

from xgboost import XGBRegressor

X_, y_ = drop_outliers(X, y)

r_columns = [
    'max_depth',
    'n_estimators',
    'reg_lambda',
    'learning_rate',
    'R2 treino',
    'R2 teste',
    'REQM Treino',
    'REQM Teste',
    'Algoritmo'
]
r_rows = []

for max_depth in (None, 3, 4, 5, 10, 20):
    for n_estimators in (50, 100, 200):
        for reg_lambda in (0, 1, 2, 3, 4):
            for learning_rate in (0.001, 0.01, 0.1):
                results = Results('Árvore de decisão aumentada')

                for X_train, y_train, X_test, y_test in shuffled_split_gen(X_, y_):
                    model = XGBRegressor(
                        n_estimators=n_estimators,
                        learning_rate=learning_rate,
                        objective='reg:squarederror',
                        booster='gbtree',
                        max_depth=max_depth,
                        reg_lambda=reg_lambda
                    )

                    model.fit(X_train, y_train)

                    results.add(
                        model,
                        X_train,
                        y_train,
                        X_test,
                        y_test
                    )

                r2_train, r2_test = results.get_mean_r2()
                rmse_train, rmse_test = results.get_mean_rmse()
                r_rows.append([
                    max_depth,
                    n_estimators,
                    reg_lambda,
                    learning_rate,
                    r2_train,
                    r2_test,
                    rmse_train,
                    rmse_test,
                    results.base_model_name
                ])
]

r_df = pd.DataFrame(r_rows, columns=r_columns)
display(r_df)

```

Fonte: Autor.

## 6. Apresentação dos Resultados

### 6.1. Fluxo de Trabalho

O **Quadro 3** apresenta um *canvas* nos moldes propostos por Vasandani (2019) para sumarizar o fluxo de trabalho que levou aos resultados apresentados a seguir.

**Quadro 3 – Canvas do projeto.**

Predição de Preços de Imóveis Residenciais		
Problema	Desfechos e Predições	Aquisição de Dados
<ul style="list-style-type: none"> <li>• <i>Why</i>: A estratégia de preços é fundamental para investir em imóveis.</li> <li>• <i>Who</i>: Anúncios de imóveis.</li> <li>• <i>What</i>: Análise dos atributos que interferem no preço.</li> <li>• <i>Where</i>: Florianópolis – SC.</li> <li>• <i>When</i>: 31/01/2021.</li> </ul>	<ul style="list-style-type: none"> <li>• Atributos: tipo, bairro, área (<math>m^2</math>), quartos, banheiros, vagas de garagem</li> <li>• Variável de saída: preço</li> </ul>	<ul style="list-style-type: none"> <li>• <i>Web scraping</i> vivareal.com.br</li> <li>• <i>Web scraping</i> imovelweb.com.br</li> </ul>
Modelagem	Avaliação de Modelo	Preparo dos Dados
<ul style="list-style-type: none"> <li>• Regressão linear múltipla</li> <li>• Regularizações</li> <li>• Árvore de decisão</li> <li>• Floresta aleatória</li> <li>• Árvore de decisão aumentada</li> </ul>	<ul style="list-style-type: none"> <li>• Validação cruzada</li> <li>• Ajuste de hiperparâmetros</li> <li>• <math>R^2</math> médio</li> <li>• REQM média</li> </ul>	<ul style="list-style-type: none"> <li>• Remoção de duplicidades</li> <li>• Imputação para ausências NMAR</li> <li>• Resolução das variações</li> <li>• Remoção de <i>outliers</i></li> </ul>

Fonte: Autor.

### 6.2. Melhor Modelo

**Tabela 16 – Resultados dos melhores modelos de cada algoritmo.**

<b>Algoritmo</b>	<b><math>R^2</math> médio</b>		<b>REQM média</b>	
	<i>Treino</i>	<i>Teste</i>	<i>Treino</i>	<i>Teste</i>
Regressão linear múltipla	0,50	0,49	1.205.737,89	1.224.583,68
Regularização de Lasso	0,49	0,46	1.205.737,89	1.224.583,68
Regularização de Ridge	0,49	0,46	1.205.737,89	1.224.583,68

Regularização Elastic-Net	0,49	0,46	1.205.737,89	1.224.583,68
Árvore de decisão	0,58	0,54	1.101.738,84	1.124.459,87
Floresta aleatória	0,61	0,59	1.043.587,40	1.071.733,04
<b>Árvore de decisão aumentada</b>	<b>0,77</b>	<b>0,74</b>	<b>802.136,22</b>	<b>828.707,41</b>

Fonte: Autor.

Os algoritmos que envolvem árvore de decisão apresentaram melhor desempenho provavelmente pelo fato de a relação dos atributos com a variável de saída não ser linear. Além disso, os métodos de agrupamento foram eficazes, com destaque para o agrupamento do tipo aceleração.

O algoritmo de árvore de decisão aumentada aplicado foi o Extreme Gradient Boosting (XGBoost), e apresentou o melhor desempenho. O melhor modelo para esse algoritmo foi treinado com os seguintes hiperparâmetros:

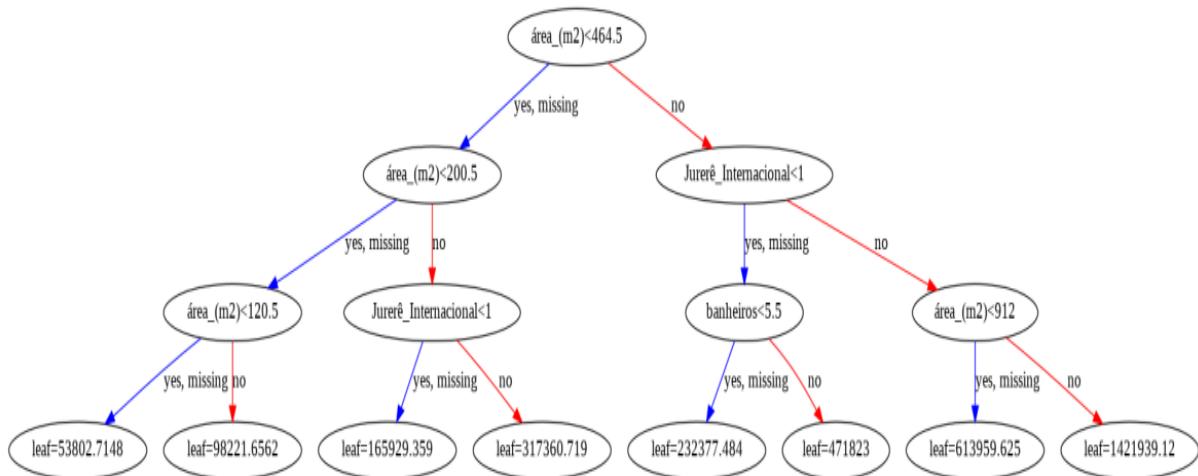
- *n\_estimators* = 100: número de árvores no agrupamento.
- *max\_depth* = 3: maior profundidade permitida para cada árvore.
- *reg\_lambda* = 3: parâmetro de regularização – favorece a generalização da predição.
- *learning\_rate* = 0,1: taxa de encolhimento da função de perda a cada iteração – quanto menor, maior a chance de convergir a um ponto ótimo da função de perda, mas mais custosa a computação.

A **Figura 29** apresenta a primeira árvore gerada para o agrupamento com o melhor modelo. Conforme configurado, apresenta uma profundidade de três camadas de nós a partir da raiz.

Os atributos selecionados para compor os nós foram aqueles que apresentaram melhor escore de similaridade, levando-se em consideração os resíduos de cada predição.

A partir dos erros cometidos pelas previsões da primeira árvore, outra árvore é gerada, priorizando as observações com maior peso por terem sido incorretamente previstas, e em direção ao ponto ótimo da função de perda – neste caso, a REQM. E assim segue a sequência para a geração de todas as outras árvores no agrupamento.

**Figura 29** – Primeira árvore no agrupamento de árvore de decisão aumentada.



Fonte: Autor.

### 6.3. Principais Atributos

A **Tabela 17** contém os principais atributos, em ordem de importância para a predição. A escala para comparação é mais evidente no gráfico do **Apêndice E**.

**Tabela 17** – Escore de importância dos atributos.

Atributo	Escore de Importância
Área (m <sup>2</sup> )	259
Vagas de garagem	76
Banheiros	62
Jurerê Internacional	42

Casa	40
Quartos	25
Jurerê	22
Agronômica	19
Cacupé	17
Beira Mar Norte	15
Cobertura	15
Ingleses do Rio Vermelho	15
Centro	13
São João do Rio Vermelho	11
Lagoa da Conceição	9
Ingleses	9
Capoeiras	7
João Paulo	7
Praia Brava	6
Jardim Atlântico	4
Estreito	3
Campeche	3
Casa de condomínio	2
Ribeirão da Ilha	2
Carianos	1
Sambaqui	1

Fonte: Autor.

O escore de importância é calculado a partir da frequência de aparecimento de cada atributo nos nós das árvores de regressão de todo o agrupamento.

A área ( $m^2$ ) se destaca consideravelmente dos outros atributos, sendo o que mais contribui para a predição do preço com esse modelo. Vagas de garagem contribuiu mais do que banheiros e quartos, e algumas variáveis fictícias de bairros se destacaram, como “Jurerê Internacional”.

#### 6.4. Métricas e Interpretação

**Tabela 18** – Resultados da regressão aplicada pelo melhor modelo para cada subconjunto.

<b>Métrica</b>	<b>Treino</b>			<b>Teste</b>		
	<i>Mínimo</i>	<i>Máximo</i>	<i>Média</i>	<i>Mínimo</i>	<i>Máximo</i>	<i>Média</i>
R <sup>2</sup>	0,77	0,78	0,78	0,74	0,78	0,77
EAM	402.346,76	405.666,44	403.702,7	404.077,13	416.226,51	409.649,42
EQM	622.429.061.297,62	656.456.351.417,17	643.473.250.758,68	620.302.651.696,82	788.997.761.062,9	688.221.547.640,21
REQM	788.941,74	788.941,74	802.136,22	787.592,95	888.255,46	828.707,41

Fonte: Autor.

As médias dos valores dos coeficientes de determinação e dos erros estão próximas, o que indica uma boa capacidade de generalização do modelo.

O valor do R<sup>2</sup> possui um valor que indica magnitude estatística razoável, mas especialistas no domínio financeiro indicam que um coeficiente de determinação seja de pelo menos 0,85 para apoiar investimentos (FERNANDO, 2020).

A REQM apresenta um valor também estatisticamente razoável, equivalendo a apenas 45% do desvio padrão da distribuição do preço. No entanto, os imóveis de preço nos menores quantis da distribuição são penalizados pelos imóveis de preços nos quantis superiores, por serem muito discrepantes, uma vez que a distribuição do preço possui alto grau de dispersão, e isso reflete em valores absolutos inadmissíveis para o domínio financeiro. Um EAM de R\$ 409.649,42 possui magnitude muito alta para o domínio, que pode refletir um risco considerável advindo de prejuízo monetário decorrente de precificação incorreta.

## 6.5. Deploy do Modelo

O *deploy* do modelo foi realizado através da criação de uma página com um formulário para alimentação dos atributos e um botão para obter a predição.

**Figura 30** – Página de predição.

The screenshot shows a web application interface for predicting real estate prices. The left side has a green header with the PUC Minas logo and the title "Predição de Preço de Imóveis" (Real Estate Price Prediction). Below this, the author's name "Fábio de Oliveira Tabalipa" is displayed. A note at the bottom states: "Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data". The right side contains a form with the following fields:

- Tipo do imóvel: Apartamento
- Bairro em Florianópolis-SC: Córrego Grande
- Área (m<sup>2</sup>): 100
- Número de quartos: 3
- Número de banheiros: 2
- Vagas de garagem: 2

A "Consultar" (Search) button is located below these fields. At the bottom, the predicted price is shown as "R\$ 673.842,38" in a large white box, with a smaller note below it stating: "Esse resultado não representa um aconselhamento de investimento".

Fonte: Autor.

A página envia requisições para um pequeno serviço desenvolvido com o framework Flask em Python, que carrega o modelo e insere os atributos para responder com a predição, que é renderizada como resultado na página.

O código-fonte da página encontra-se no **Apêndice F** e do serviço no **Apêndice G**.

O **Quadro 4** contém um passo-a-passo para executar o serviço em máquina local e testar a predição com a página.

**Quadro 4 – Passo-a-passo para executar o serviço de predição em máquina local.**

1	Instalar os módulos Python do requirements.txt <code>pip install -r requirements.txt</code>
2	Executar o Flask <code>flask run</code>
3	Acessar a seguinte URL com o navegador web: <code>http://127.0.0.1:5000/</code>

Fonte: Autor.

## 6.6. Conclusões

Os modelos gerados pela árvore de decisão aumentada apresentaram um desempenho superior em relação aos de outros algoritmos.

A área ( $m^2$ ) se destacou como preditor, e sua distância em escore de importância para os demais atributos pode indicar a necessidade de descoberta de mais atributos. Por exemplo, a obtenção de mais detalhes em relação à geolocalização, como proximidade ao litoral, e análises de processamento de linguagem natural das descrições do anúncio, obtendo informações como vista livre, sol da manhã ou portaria 24 horas, poderiam melhorar o desempenho do modelo.

Os resultados da regressão poderiam ser satisfatórios caso a distribuição não fosse tão dispersa, o que indica que o melhor modelo está na direção correta, mas necessitará de ajustes mais avançados para obter de fato a utilidade prática.

## 7. Links

- **Link para o vídeo:** <https://youtu.be/Kc9P7FQx2ME>
- **Link para o repositório:** <https://github.com/fabiotabalipa/ml-house-pricing>

## REFERÊNCIAS

Atlas do Desenvolvimento Humano no Brasil 2013. Programa das Nações Unidas para o Desenvolvimento, 2010. Disponível em: <<https://www.br.undp.org/content/brazil/pt/home/idh0/rankings/idhm-municipios-2010.html>>. Acesso em: 31 de jan. de 2021.

FERNANDO, Jason. R-Squared Definition, 18 de nov. de 2020. Disponível em: <<https://www.investopedia.com/terms/r/r-squared.asp>>. Acesso em: 31 de jan. de 2021.

Informe de dezembro/2020. Grupo ZAP e fipe, dez. de 2020. Disponível em: <<https://fipezap.zapimoveis.com.br/wp-content/uploads/2021/01/fipezap-202012-residencial-venda.pdf>>. Acesso em: 31 de jan. de 2021.

Panorama do Setor de Tecnologia de Santa Catarina 2018. Associação Catarinense de Tecnologia, 2018. Disponível em: <<https://www.acate.com.br/wp-content/uploads/2018/11/ACATE-Observat%C3%B3rio-2018.pdf>>. Acesso em: 31 de jan. de 2021.

VASANDANI, Jasmine. A Data Science Workflow Canvas to Kickstart Your Projects, 5 de mai. de 2019. Disponível em: <<https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0>>. Acesso em: 31 de jan. de 2021.

## APÊNDICE A – WEB SCRAPING DO VIVAREAL.COM.BR

```

import json
import re
import time

from bs4 import BeautifulSoup
from google.colab import files
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
import pandas as pd

options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')

driver = webdriver.Chrome('chromedriver', options=options)

BASE_URL = 'https://www.vivareal.com.br/venda/santa-catarina/florianopolis'

vr_endpoint_fpolis = BASE_URL + '/{}{/}'

vr_dict = {
    'tipo': [],
    'bairro': [],
    'área (m²)': [],
    'quartos': [],
    'banheiros': [],
    'vagas de garagem': [],
    'preço': []
}

for req_type in (
    'condominio_residencial',
    'cobertura_residencial',
    'apartamento_residencial',
    'casa_residencial'
):
    print(req_type)
    driver.get(vr_endpoint_fpolis.format(req_type))

    pg, last_page = 1, False
    while (not last_page):
        vr_listing_soup = BeautifulSoup(driver.page_source, 'html.parser')

        listings = vr_listing_soup.find_all('article', class_='js-property-card')
        for listing in listings:
            # ignorar imóveis na planta ou em construção
            if listing.find('div', class_='js-construction-status'):
                continue

            type_ = ''
            if req_type == 'apartamento_residencial':
                type_ = 'Apartamento'
            elif req_type == 'cobertura_residencial':
                type_ = 'Cobertura'
            elif req_type == 'casa_residencial':
                type_ = 'Casa'
            elif req_type == 'condominio_residencial':
                type_ = 'Casa de condomínio'

```

```

neighborhood_str = listing.find('span', class_='property-card__address').string.strip()
if ', Florianópolis - SC' not in neighborhood_str:
    continue

if neighborhood_str.count('-') == 2:
    neighborhood = re.search(r'- (.+), Florianópolis - SC', neighborhood_str).group(1)
else:
    neighborhood = re.search(r'^(.+), Florianópolis - SC', neighborhood_str).group(1)

area = None
area_tag = listing.find('li', class_='property-card__detail-area')
if area_tag:
    area = area_tag.find('span', class_='property-card__detail-value').string.strip()

bedrooms = None
bedrooms_tag = listing.find('li', class_='property-card__detail-room')
if bedrooms_tag:
    bedrooms = bedrooms_tag.find('span', class_='property-card__detail-value').string.strip()

bathrooms = None
bathrooms_tag = listing.find('li', class_='property-card__detail-bathroom')
if bathrooms_tag:
    bathrooms = bathrooms_tag.find('span', class_='property-card__detail-value').string.strip()

parking_spaces = None
parking_spaces_tag = listing.find('li', class_='property-card__detail-garage')
if parking_spaces_tag:
    parking_spaces = parking_spaces_tag.find('span', class_='property-card__detail-value').string.strip()

price_tag = listing.find('div', class_='js-property-card-prices').find('p',
style='display: block;')
if price_tag is None:
    continue
price = price_tag.string.strip()

vr_dict['tipo'].append(type_)
vr_dict['bairro'].append(neighborhood)
vr_dict['área (m²)'].append(area)
vr_dict['quartos'].append(bedrooms)
vr_dict['banheiros'].append(bathrooms)
vr_dict['vagas de garagem'].append(parking_spaces)
vr_dict['preço'].append(re.sub(r'\D', '', price))

print('pg {} ok'.format(pg))

try:
    next_page_btn = driver.find_element_by_xpath("//a[@class='js-change-page']"
[@title='Próxima página']")
    next_page_btn.click()
except:
    print('não há mais páginas de {}'.format(req_type))
    last_page = True
finally:
    time.sleep(5)

pg += 1

vr_df = pd.DataFrame(vr_dict)
vr_df.to_csv('/content/vivareal.tsv', sep='\t', index=False)
files.download('/content/vivareal.tsv')

```

## APÊNDICE B – WEB SCRAPING DO IMOVELWEB.COM.BR

```

import json
import re
import time

from bs4 import BeautifulSoup
from google.colab import files
import cloudscraper
import pandas as pd

scraper = cloudscraper.create_scraper()
BASE_URL = 'https://www.imovelweb.com.br'

iw_endpoint_fpolis = BASE_URL + '/{}-venda-florianopolis-sc-pagina-{}.html'

iw_dict = {
    'tipo': [],
    'bairro': [],
    'área (m²)': [],
    'quartos': [],
    'banheiros': [],
    'vagas de garagem': [],
    'preço': []
}

for req_type in ('apartamentos', 'casas'):
    print(req_type)

    pg, last_page, results, count = 1, False, None, 0
    while (not last_page):
        iw_listing_res = scraper.get(iw_endpoint_fpolis.format(req_type, pg))

        iw_listing_soup = BeautifulSoup(iw_listing_res.content, 'html.parser')

        if results is None:
            results_str = iw_listing_soup.find('h1', class_='list-result-title').string
            results = int(re.sub(r'\D', '', results_str))

        for listing in iw_listing_soup.find_all('div', class_='postingCard'):
            count += 1

            # ignorar imóveis na planta ou em construção
            if (
                listing.find('span', class_='fromPrice') or
                listing.find('i', class_='iconUnits')
            ):
                continue

            description = listing.find('div', class_='postingCardDescription').text.strip()

            if req_type == 'apartamentos':
                if 'cobertura' in description.lower():
                    type_ = 'Cobertura'
                else:
                    type_ = 'Apartamento'
            elif req_type == 'casas':
                type_ = 'Casa'
                condominium_tag = listing.find('span', class_='postingCardExpenses')
                if condominium_tag:
                    if condominium_tag.string:
                        if 'condomínio' in condominium_tag.string.lower():
                            type_ = type_ + ' de condomínio'

            iw_dict[type_].append({
                'pagina': pg,
                'url': iw_listing_res.url
            })

```

```

    iw_dict['tipo'].append(type_)

    neighborhood = listing.find('span',
                                class_='postingCardLocation').span.text.strip().split(',')[0]
    iw_dict['bairro'].append(neighborhood)

    area = None
    area_tag = listing.find('i', class_=['iconArea'])
    if area_tag:
        area = [s for s in area_tag.parent.stripped_strings][0]
        area = re.search(r'(\d+)\D+', area).group(1)
    iw_dict['área (m²)'].append(area)

    bedrooms = None
    bedrooms_tag = listing.find('i', class_='iconBedrooms')
    if bedrooms_tag:
        bedrooms = [s for s in bedrooms_tag.parent.stripped_strings][0]
        bedrooms = re.search(r'(\d+)\D+', bedrooms).group(1)
    iw_dict['quartos'].append(bedrooms)

    bathrooms = None
    bathrooms_tag = listing.find('i', class_='iconBathrooms')
    if bathrooms_tag:
        bathrooms = [s for s in bathrooms_tag.parent.stripped_strings][0]
        bathrooms = re.search(r'(\d+)\D+', bathrooms).group(1)
    iw_dict['banheiros'].append(bathrooms)

    parking_spaces = None
    parking_spaces_tag = listing.find('i', class_='iconGarage')
    if parking_spaces_tag:
        parking_spaces = [s for s in parking_spaces_tag.parent.stripped_strings][0]
        parking_spaces = re.search(r'(\d+)\D+', parking_spaces).group(1)
    iw_dict['vagas de garagem'].append(parking_spaces)

    price = listing.find('span', class_='firstPrice').text.strip()
    iw_dict['preço'].append(re.sub(r'\D', '', price))

    print('pg {}'.format(pg))
    time.sleep(2)
    pg += 1

    if count >= results:
        last_page = True

    print('count', count, 'results', results)

    iw_df = pd.DataFrame(iw_dict)
    iw_df.to_csv('/content/imovelweb.tsv', sep='\t', index=False)
    files.download('/content/imovelweb.tsv')

```

## APÊNDICE C – CORREÇÃO DAS VARIAÇÕES DE BAIRROS

```
● ● ●

def fix_var_neighborhood(neighborhood):
    global var_count

    if neighborhood in (
        'Alto Ribeirão Leste'
    ):
        var_count += 1
        return 'Alto Ribeirão'

    if neighborhood in (
        'Armacao do Pântano do Sul',
        'Armação',
        'Armação do Pântano do Sul'
    ):
        var_count += 1
        return 'Pântano do Sul'

    if neighborhood in (
        'Avenida do Ipê-Amarelo - Lagoa da Conceição',
        'Avenida do Ipê-Amarelo, 333 - Lagoa da Conceição',
        'Rua do Beija-Flor - Lagoa da Conceição',
        'Rua do Beija-Flor, 219 - Lagoa da Conceição',
        'Rua do Bem-Te-Vi - Lagoa da Conceição',
        'Rua do Bem-Te-Vi, 100 - Lagoa da Conceição',
        'Rua do Ipê-Roxo - Lagoa da Conceição',
        'Rua do Ipê-Roxo, 305 - Lagoa da Conceição',
        'Rua do Ipê-Roxo, 50 - Lagoa da Conceição'
    ):
        var_count += 1
        return 'Lagoa da Conceição'

    if neighborhood in (
        'Balneario',
        'Balneário'
    ):
        var_count += 1
        return 'Estreito'

    if neighborhood in (
        'Servidão Jacatirão-Açu - Campeche', 'Servidão Jacatirão-Açu, 101 - Campeche',
        'Campeche Central',
        'Campeche Leste',
        'Servidão Araçá-da-Praia - Campeche'
    ):
        var_count += 1
        return 'Campeche'

    if neighborhood in (
        'Jurere Leste'
    ):
        var_count += 1
        return 'Jurerê'

    if neighborhood in (
        'P das Canas',
        'Ponta das Canas'
    ):
        var_count += 1
        return 'Ponta das Canas'
```

```

if neighborhood in (
    'Praia Acores'
):
    var_count += 1
    return 'Praia dos Acores'

if neighborhood in (
    'Rua Arco-Íris - Carianos',
    'Rua Arco-Íris, 172 - Carianos',
    'Rua Arco-Íris, 61 - Carianos'
):
    var_count += 1
    return 'Carianos'

if neighborhood in (
    'Rua Maestro Villa-Lobos - João Paulo'
):
    var_count += 1
    return 'João Paulo'

if neighborhood in (
    'Rua Bem-Vi - Ingleses do Rio Vermelho',
    'Rua Bem-Vi, 1050 - Ingleses do Rio Vermelho',
    'Rua Tico-Tico, 338 - Ingleses do Rio Vermelho',
    'Rua do Trinta - Réis - Ingleses do Rio Vermelho',
    'Rua do Trinta - Réis, 64 - Ingleses do Rio Vermelho'
):
    var_count += 1
    return 'Ingleses do Rio Vermelho'

if neighborhood in (
    'Rua dos Lambari-Guaçu - Jurerê Internacional',
    'Rua dos Lambari-Guaçu, 112 - Jurerê Internacional',
    'Rua dos Lambari-Guaçu, 397 - Jurerê Internacional',
    'Rua dos Lambari-Guaçu, 400 - Jurerê Internacional',
    'Rua dos Lambari-Guaçu, 417 - Jurerê Internacional',
    'Rua dos Peixes-Elétricos - Jurerê Internacional',
    'Rua dos Peixes-Espada - Jurerê Internacional',
    'Rua dos Peixes-Espada, 49 - Jurerê Internacional',
    'Rua dos Peixes-Lua - Jurerê',
    'Rua dos Peixes-Lua - Jurerê Internacional',
    'Rua dos Peixes-Lua, 121 - Jurerê Internacional',
    'Rua dos Peixes-Lua, 4 - Jurerê',
    'Rua dos Peixes-Lua, 49 - Jurerê Internacional',
    'Rua dos Peixes-Lua, 70 - Jurerê Internacional',
    'Rua dos Peixes-Serra - Jurerê Internacional',
    'Rua dos Peixes-Serra, 103 - Jurerê Internacional',
    'Rua dos Peixes-Serra, 133 - Jurerê Internacional'
):
    var_count += 1
    return 'Jurerê Internacional'

if neighborhood in (
    'Jd Anchieta'
):
    var_count += 1
    return 'Jardim Anchieta'

if neighborhood in (
    'ingleses',
    'Ingleses Sul',
    'Rua do Trinta - Réis, 102 - Ingleses Sul',
    'Ingleses Centro', 'Ingleses Norte'
):
    var_count += 1
    return 'Ingleses'

```

```
if neighborhood in (
    'SC-405 - Moenda'
):
    var_count += 1
    return 'Moenda'

if neighborhood in (
    'Tapera da Base'
):
    var_count += 1
    return 'Tapera'

if neighborhood in (
    'Servidão Ipê-Branco, 350 - São João do Rio Vermelho',
    'Rio Vermelho'
):
    var_count += 1
    return 'São João do Rio Vermelho'

if neighborhood in (
    'Santa Monica'
):
    var_count += 1
    return 'Santa Mônica'

if neighborhood in (
    'Santo Antonio de Lisboa'
):
    var_count += 1
    return 'Santo Antônio de Lisboa'

return neighborhood

var_count = 0
df['bairro'] = df.apply(lambda row: fix_var_neighborhood(row['bairro']), axis=1)
print('variações encontradas: ', var_count)
```

## APÊNDICE D – AUXILIARES PARA OS ALGORITMOS DE REGRESSÃO

```

● ● ●

import numpy as np

# Função auxiliar para transformação logarítmica

def log_t(input):
    return np.log(input, where=(input != 0))

def exp_t(input):
    return np.exp(input, where=(input != 0))

# Funções auxiliares para detecção e remoção de outliers

def is_pos_def(A):
    if np.allclose(A, A.T):
        try:
            np.linalg.cholesky(A)

            return True
        except np.linalg.LinAlgError:
            return False

    return False

def mahalanobis_distance(data):
    covariance_matrix = np.cov(data, rowvar=False)

    if is_pos_def(covariance_matrix):
        inv_covariance_matrix = np.linalg.inv(covariance_matrix)

    if is_pos_def(inv_covariance_matrix):
        vars_mean = []

        for i in range(data.shape[0]):
            vars_mean.append(list(data.mean(axis=0)))

        diff = data - vars_mean

        md = []
        for i in range(len(diff)):
            md.append(np.sqrt(diff[i].dot(inv_covariance_matrix).dot(diff[i])))

        return md
    else:
        raise Exception('A matriz inversa de covariância não é positivo definida!')
else:
    raise Exception('A matriz de covariância não é positivo definida!')

def get_outliers_idx(data):
    md = mahalanobis_distance(data)
    std = np.std(md)

    k = 2. * std

    m = np.mean(md)

    up_t = m + k
    low_t = m - k

```

```

outliers_idx = []
for i in range(len(md)):
    if (md[i] >= up_t) or (md[i] <= low_t):
        outliers_idx.append(i)

return outliers_idx

def drop_outliers(X, y):
    outliers_idx = get_outliers_idx(X.select_dtypes(['number']).values)

    return X.drop(outliers_idx), y.drop(outliers_idx)

# Classe auxiliar para os resultados

class Results:
    def __init__(self, base_model_name):
        self.base_model_name = base_model_name

        self.r2_dict = {'Treino': np.array([]), 'Teste': np.array([])}
        self.mae_dict = {'Treino': np.array([]), 'Teste': np.array([])}
        self.mse_dict = {'Treino': np.array([]), 'Teste': np.array([])}
        self.rmse_dict = {'Treino': np.array([]), 'Teste': np.array([])}

    def add(
        self, model, X_train, y_train, X_test, y_test,
        inverse_log=False, inverse_normal=False
    ):
        sub_datasets = {
            'Treino': (X_train, y_train),
            'Teste': (X_test, y_test)
        }

        for k in sub_datasets.keys():
            X = sub_datasets[k][0]
            y = sub_datasets[k][1]

            y_hat = model.predict(X)

            r2 = r2_score(y, y_hat)
            mae = mean_absolute_error(y, y_hat)
            mse = mean_squared_error(y, y_hat)

            rmse = math.sqrt(mse)

            self.r2_dict[k] = np.append(self.r2_dict[k], r2)
            self.mae_dict[k] = np.append(self.mae_dict[k], mae)
            self.mse_dict[k] = np.append(self.mse_dict[k], mse)
            self.rmse_dict[k] = np.append(self.rmse_dict[k], rmse)

    def get_df(self):
        data = {
            'Métrica': ['R2', 'EAM', 'EQM', 'REQM', 'IP 95%']
        }

        for x in ('Treino', 'Teste'):
            (
                key_min, key_max, key_avg
            ) = (
                'Mínimo - {}'.format(x),
                'Máximo - {}'.format(x),
                'Média - {}'.format(x)
            )

            if key_min not in data.keys():
                data[key_min] = []

            if key_max not in data.keys():
                data[key_max] = []

        return pd.DataFrame(data)

```

```

if key_avg not in data.keys():
    data[key_avg] = []

data[key_min].append(str(round(self.r2_dict[x].min(), 2)))
data[key_min].append(str(round(self.mae_dict[x].min(), 2)))
data[key_min].append(str(round(self.mse_dict[x].min(), 2)))
data[key_min].append(str(round(self.rmse_dict[x].min(), 2)))
data[key_min].append(str(round(self.rmse_dict[x].mean()*2-1, 2)))

data[key_max].append(str(round(self.r2_dict[x].max(), 2)))
data[key_max].append(str(round(self.mae_dict[x].max(), 2)))
data[key_max].append(str(round(self.mse_dict[x].max(), 2)))
data[key_max].append(str(round(self.rmse_dict[x].max(), 2)))
data[key_max].append(str(round(self.rmse_dict[x].mean()*2, 2)))

data[key_avg].append(str(round(self.r2_dict[x].mean(), 2)))
data[key_avg].append(str(round(self.mae_dict[x].mean(), 2)))
data[key_avg].append(str(round(self.mse_dict[x].mean(), 2)))
data[key_avg].append(str(round(self.rmse_dict[x].mean(), 2)))
data[key_avg].append(' - ')

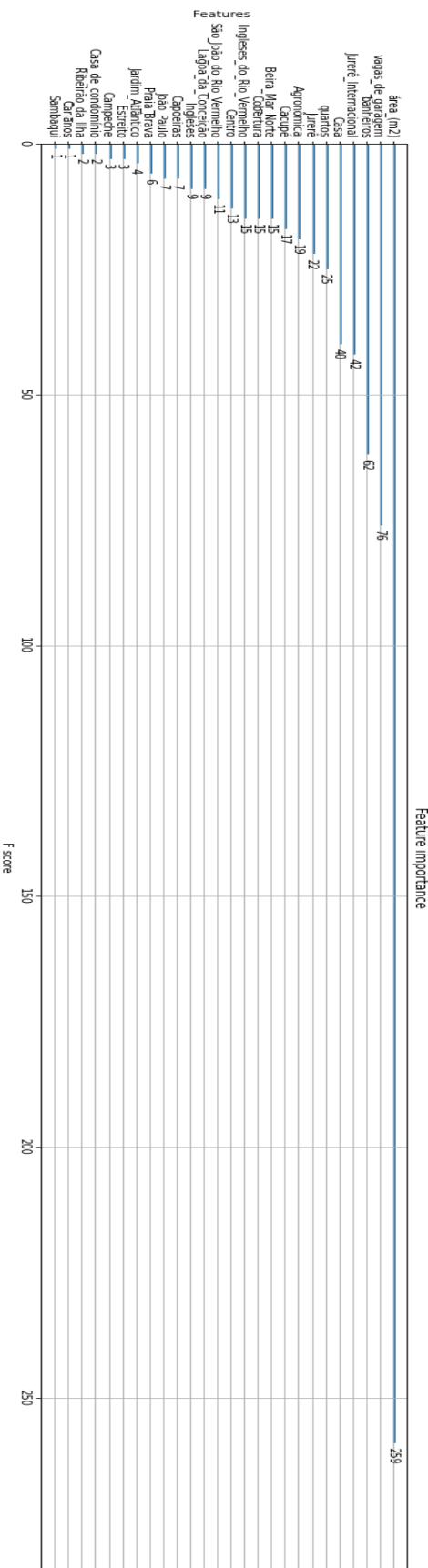
return pd.DataFrame(data)

def get_mean_r2(self):
    return (
        round(self.r2_dict['Treino'].min(), 2),
        round(self.r2_dict['Teste'].min(), 2)
    )

def get_mean_rmse(self):
    return (
        round(self.rmse_dict['Treino'].mean(), 2),
        round(self.rmse_dict['Teste'].mean(), 2)
    )

```

## APÊNDICE E – ESCORE DE IMPORTÂNCIA DOS ATRIBUTOS



## APÊNDICE F – CÓDIGO-FONTE DA PÁGINA DE PREDIÇÃO

```

● ● ●

<!doctype html>
<html lang="pt-br">
    <head>
        <meta charset="utf-8"/>
        <title>TCC Ciência de Dados - Fábio Tabalipa</title>

        <link href='http://fonts.googleapis.com/css?family=Roboto' rel='stylesheet'
type='text/css'>
        <link rel="stylesheet" href="/static/style.css">
    </head>
    <body>
        <div id="background">
            {% if result %}
                <div id="result-frame">
                    <span id="result-price">
                        {{ result }}
                    </span>
                    <span id="result-disclaimer">
                        Esse resultado não representa um aconselhamento de investimento
                    </span>
                </div>
            {% endif %}
            <div id="content-frame">
                <div id="header-frame">
                    
                    <p id="title" class="header-text">
                        Predição de Preço de Imóveis
                    </p>
                    <p id="author" class="header-text">
                        Fábio de Oliveira Tabalipa
                    </p>
                    <p id="description" class="header-text">
                        Trabalho de Conclusão de Curso apresentado ao Curso de Especialização
em Ciência de Dados e Big Data
                    </p>
                </div>
                <div id="form-frame">
                    <form action="{{ url_for('main') }}" method="POST">
                        <label>
                            Tipo do imóvel
                            <select name="type" class="form-element-box">
                                <option value="Apartamento">Apartamento</option>
                                <option value="Cobertura">Cobertura</option>
                                <option value="Casa">Casa</option>
                                <option value="Casa_de_Condomínio">Casa de
Condomínio</option>
                            </select>
                        </label>
                        <label>
                            Bairro em Florianópolis-SC
                            <select name="neighborhood" class="form-element-box">
                                <option value="Abraão">Abraão</option>
                                <option value="Agronômica">Agronômica</option>
                                <option value="Alto_Ribeirão">Alto Ribeirão</option>
                                <option value="Autódromo">Autódromo</option>
                                <option value="Barra_da_Lagoa">Barra da Lagoa</option>
                                <option value="Barreiros">Barreiros</option>
                                <option value="Beira_Mar_Norte">Beira Mar Norte</option>
                                <option value="Bom_Abrigo">Bom Abrigo</option>
                                <option value="Cachoeira_do_Bom_Jesus">Cachoeira do Bom
Jesus</option>
                                <option value="Cacupé">Cacupé</option>
                                <option value="Caiacanga">Caiacanga</option>
                            </select>
                        </label>
                    </form>
                </div>
            </div>
        </div>
    </body>
</html>

```

```

        <option value="Campeche">Campeche</option>
        <option value="Campinas">Campinas</option>
        <option value="Canajurê">Canajurê</option>
        <option value="Canasvieiras">Canasvieiras</option>
        <option value="Canto">Canto</option>
        <option value="Canto_da_Lagoa">Canto da Lagoa</option>
        <option value="Canto_dos_Araçás">Canto dos Araçás</option>
        <option value="Capoeiras">Capoeiras</option>
        <option value="Carianos">Carianos</option>
        <option value="Carvoeira">Carvoeira</option>
        <option value="Centro">Centro</option>
        <option value="Coloninha">Coloninha</option>
        <option value="Coqueiros">Coqueiros</option>
        <option value="Costa_da_Lagoa">Costa_da_Lagoa</option>
        <option value="Costeira_Do_Ribeirão_Da_Ilha">Costeira Do
Ribeirao Da Ilha</option>
        <option value="Costeira_do_Pirajubaé">Costeira do
Pirajubaé</option>
        <option value="Costeira_do_Ribeirão">Costeira do
Ribeirão</option>
        <option value="Córrego_Grande">Córrego Grande</option>
        <option value="Daniela">Daniela</option>
        <option value="Estreito">Estreito</option>
        <option value="Ingleses">Ingleses</option>
        <option value="Ingleses_do_Rio_Vermelho">Ingleses do Rio
Vermelho</option>
        <option value="Itacorubi">Itacorubi</option>
        <option value="Itaguaçu">Itaguaçu</option>
        <option value="Jardim_Anchieta">Jardim Anchieta</option>
        <option value="Jardim_Atlântico">Jardim Atlântico</option>
        <option value="José_Mendes">José Mendes</option>
        <option value="João_Paulo">João Paulo</option>
        <option value="Jurerê">Jurerê</option>
        <option value="Jurerê_Internacional">Jurerê
Internacional</option>
        <option value="Lagoa_Pequena">Lagoa Pequena</option>
        <option value="Lagoa_da_Conceição">Lagoa da
Conceição</option>
        <option value="Lagoinha">Lagoinha</option>
        <option value="Moenda">Moenda</option>
        <option value="Monte_Cristo">Monte Cristo</option>
        <option value="Monte_Verde">Monte Verde</option>
        <option value="Morro_da_Cruz">Morro da Cruz</option>
        <option value="Morro_das_Pedras">Morro das Pedras</option>
        <option value="Muquém">Muquém</option>
        <option value="Pantanal">Pantanal</option>
        <option value="Parque_São_Jorge">Parque São Jorge</option>
        <option value="Pedrita">Pedrita</option>
        <option value="Ponta_das_Canas">Ponta das Canas</option>
        <option value="Porto_da_Lagoa">Porto da Lagoa</option>
        <option value="Praia_Brava">Praia Brava</option>
        <option value="Praia_Mole">Praia Mole</option>
        <option value="Praia_da_Lagoinha">Praia da Lagoinha</option>
        <option value="Praia_dos_Açores">Praia dos Açores</option>
        <option value="Pântano_do_Sul">Pântano do Sul</option>
        <option value="Ratones">Ratones</option>
        <option value="Ribeirão_da_Ilha">Ribeirão da Ilha</option>
        <option value="Rio_Tavares">Rio Tavares</option>
        <option value="Saco_Grande">Saco Grande</option>
        <option value="Saco_dos_Limões">Saco dos Limões</option>
        <option value="Sambaqui">Sambaqui</option>
        <option value="Santa_Mônica">Santa Mônica</option>
        <option value="Santinho">Santinho</option>
        <option value="Santo_Antônio_de_Lisboa">Santo Antônio de
Lisboa</option>
        <option value="Serrinha">Serrinha</option>
        <option value="São_João_do_Rio_Vermelho">São João do Rio
Vermelho</option>
    
```

```

                <option value="Trindade">Trindade</option>
                <option value="Vargem_Grande">Vargem Grande</option>
                <option value="Vargem_Pequena">Vargem Pequena</option>
                <option value="Vargem_do_Bom_Jesus">Vargem do Bom
Jesus</option>
            </select>
        </label>
        <label>
            Área (m2)
            <input name="area" class="form-element-box" type="text" />
        </label>
        <label>
            Número de quartos
            <select name="bedrooms" class="form-element-box">
                <option value="0">0</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
                <option value="8">8</option>
                <option value="9">9</option>
                <option value="10">10</option>
            </select>
        </label>
        <label>
            Número de banheiros
            <select name="bathrooms" class="form-element-box">
                <option value="0">0</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
                <option value="8">8</option>
                <option value="9">9</option>
                <option value="10">10</option>
            </select>
        </label>
        <label>
            Vagas de garagem
            <select name="parking_spaces" class="form-element-box">
                <option value="0">0</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
                <option value="8">8</option>
                <option value="9">9</option>
                <option value="10">10</option>
            </select>
        </label>
        <input id="form-button" type="submit" value="Consultar">
    </form>
</div>
</div>
</body>
</html>

```

```
● ● ●

html, body {
    height: 100%;
    width: 100%;

    margin: 0;
    padding: 0;

    border: 0;

    font-size: 100%;
    font: inherit;
    font-family: 'Roboto', sans-serif;

    vertical-align: baseline;
}

#background {
    display: flex;
    justify-content: center;

    height: 100%;
    width: 100%;

    background-color: #272727;
}

#content-frame {
    display: flex;
    flex: 1;

    height: 100%;
    max-width: 1024px;
}

#header-frame {
    align-items: center;
    display: flex;
    flex: 1;
    flex-direction: column;
    justify-content: center;
    z-index: 1;

    margin: 64px 0px;
    padding: 40px;

    background-color: #48CA93;

    box-shadow: 0 2px 11px 0 rgba(0,0,0,0.50);
}

.header-text {
    text-align: center;
}

#title {
    font-size: 32px;
    font-weight: 600;
}

#author {
    font-size: 28px;
    font-weight: 500;
}
```

```
#description {
    color: #4F5858;

    font-size: 18px;
    font-weight: 500;
}

#form-frame {
    align-items: center;
    display: flex;
    flex: 0.9;
    justify-content: center;

    margin: 64px 0px;
    background-color: #D6D6D6;
}

input, select, label {
    display: block;

    -webkit-box-sizing: border-box; /* Safari/Chrome, other WebKit */
    -moz-box-sizing: border-box;    /* Firefox, other Gecko */
    box-sizing: border-box;        /* Opera/IE 8+ */
}

input:focus, select:focus{
    outline: none;
}

label {
    font-weight: 600;
    margin-bottom: 32px;
}

.form-element-box {
    height: 48px;
    width: 248px;
    max-width: 248px;

    margin-top: 4px;
    padding: 0px 8px;

    border: transparent;
    border-radius: 8px;
}

#form-button {
    height: 48px;
    width: 248px;
}

#result-frame {
    align-items: center;
    bottom: 132px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    position: absolute;
    z-index: 3;

    height: 96px;
    width: 400px;

    background-color: rgba(0,0,0,0.65);

    text-align: center;

    box-shadow: 0 2px 11px 0 rgba(0,0,0,0.50);
}
```

```
#result-price {  
    font-size: 22px;  
    font-weight: 600;  
  
    margin-bottom: 8px;  
  
    color: #FFFFFF;  
}  
  
#result-disclaimer {  
    font-size: 14px;  
    font-weight: 500;  
  
    color: #D6D6D6;  
}
```

## APÊNDICE G – CÓDIGO-FONTE DO SERVIÇO DE PREDIÇÃO

```

● ● ●

from xgboost import XGBRegressor
import flask
import locale
import pandas as pd
from df_schema import df_dict

model = XGBRegressor()
model.load_model('model/best_model.json')

app = flask.Flask(__name__, template_folder='templates')

def brl(value):
    locale.setlocale(locale.LC_ALL, 'pt_BR.UTF-8')
    return 'R$ {}'.format(locale.currency(value, grouping=True, symbol=False))

@app.route('/', methods=['GET', 'POST'])
def main():
    if flask.request.method == 'GET':
        return flask.render_template('main.html')

    if flask.request.method == 'POST':
        type_ = flask.request.form['type']
        neighborhood = flask.request.form['neighborhood']

        if type_ != 'Apartamento':
            df_dict[type_] = 1.

        if neighborhood != 'Tapera':
            df_dict[neighborhood] = 1.

        df_dict['área_(m2)'] = float(flask.request.form['area'])
        df_dict['quartos'] = float(flask.request.form['bedrooms'])
        df_dict['banheiros'] = float(flask.request.form['bathrooms'])
        df_dict['vagas_de_garagem'] = float(flask.request.form['parking_spaces'])

        X = pd.DataFrame(df_dict)
        y_hat = model.predict(X)

        return flask.render_template('main.html', result=brl(y_hat[0]))

if __name__ == '__main__':
    app.run()

```