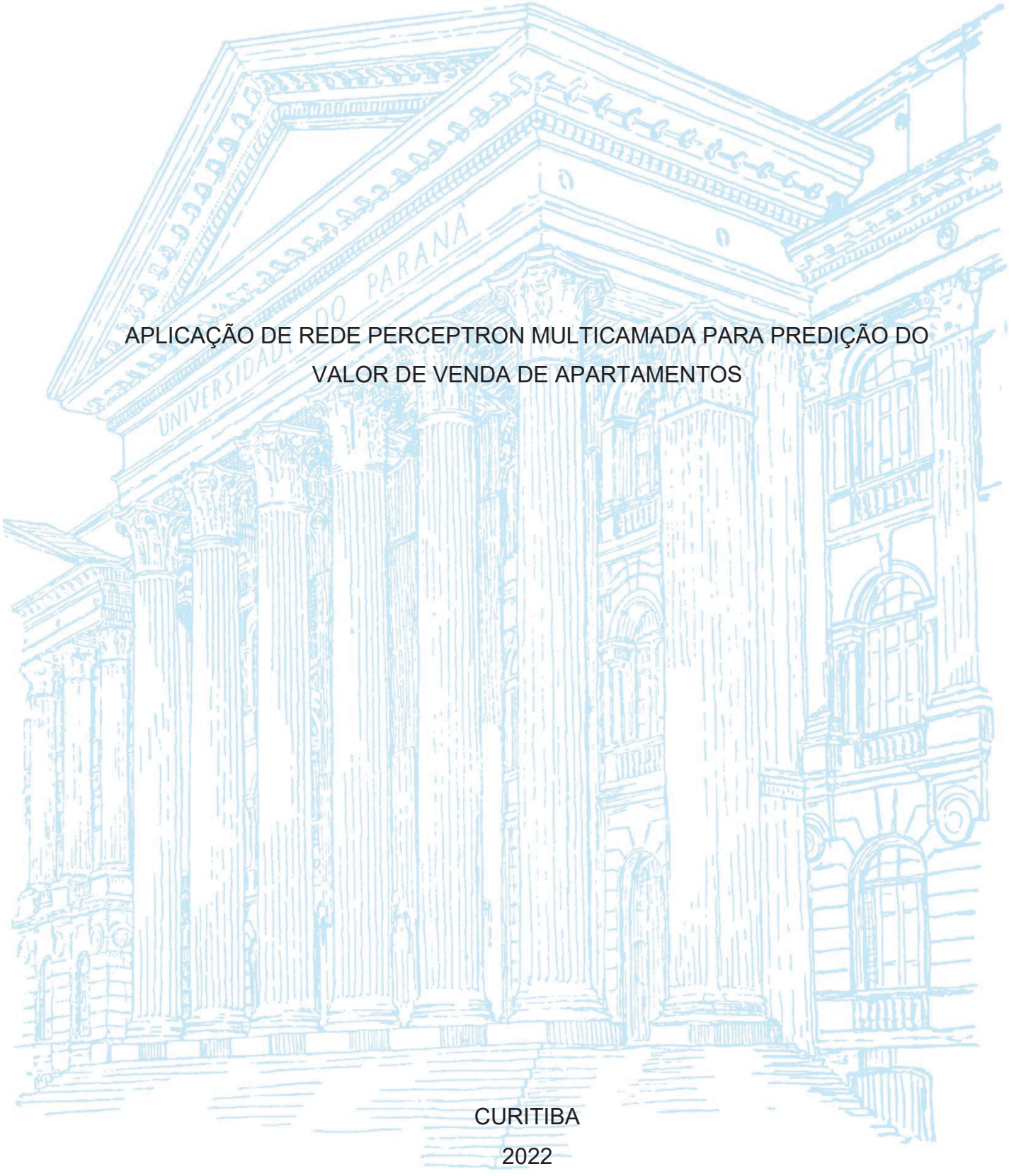
UNIVERSIDADE FEDERAL DO PARANÁ

ANDRÉ KLINGENFUS ANTUNES



APLICAÇÃO DE REDE PERCEPTRON MULTICAMADA PARA PREDIÇÃO DO VALOR DE VENDA DE APARTAMENTOS

CURITIBA

2022

ANDRÉ KLINGENFUS ANTUNES

APLICAÇÃO DE REDE PERCEPTRON MULTICAMADA PARA PREDIÇÃO DO VALOR DE VENDA DE APARTAMENTOS

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

CURITIBA

2022

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **ANDRÉ KLINGENFUS ANTUNES** intitulada: **APLICAÇÃO DE REDE PERCEPTRON MULTICAMADA PARA PREDIÇÃO DO VALOR DE VENDA DE APARTAMENTOS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 13 de Outubro de 2022.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

RAZER ANTHOM NIZER ROJAS MONTAÑO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

APLICAÇÃO DE REDE PERCEPTRON MULTICAMADA PARA PREDIÇÃO DO VALOR DE VENDA DE APARTAMENTOS

André Klingenfus Antunes
SEPT - Setor de Educação Profissional e Tecnológica
UFPR - Universidade Federal do Paraná
Curitiba, Brasil
andre.antunes@ufpr.br

Rafaela Mantovani Fontana
SEPT - Setor de Educação Profissional e Tecnológica
UFPR - Universidade Federal do Paraná
Curitiba, Brasil
rafaela.fontana@ufpr.br

***Resumo* — Avaliar precisamente um imóvel é uma atividade importante para interessados no mercado imobiliário. Para isso, utilizam-se modelos de regressão que estimam o valor de um imóvel com base em suas características. Nesse contexto, atualmente, destacam-se as redes neurais artificiais, que geram modelos com alta precisão e baixo custo computacional. No quadro da informatização do setor imobiliário, nota-se a existência de aplicações que atuam como serviços automatizados de avaliação de imóveis. Ainda, levando em conta as tendências no setor da tecnologia da informação, destaca-se o crescimento da utilização de dispositivos móveis. Por meio desses dispositivos, usuários podem interagir de forma rápida com aplicações móveis de alta precisão. Nesse sentido, este artigo tem como objetivo apresentar o desenvolvimento de uma aplicação *mobile* que utiliza uma rede neural artificial para avaliar apartamentos. Assim, espera-se auxiliar decisões de compradores, vendedores ou quaisquer interessados com estimativas precisas do preço de venda de apartamentos. A rede neural gerada pelo modelo, do tipo Perceptron multicamadas de regressão, apresentou R2 de 81,14% e erro médio absoluto de R$ 63.861,71.**

***Palavras chave* —apartamentos, avaliação imobiliária, dispositivos móveis, redes neurais artificiais, regressão**

***Abstract* — Accurately evaluating a property is an important operation for parts concerned in the real estate market. For this purpose, regression models are used to assess property's value based on its features. In this context, currently, artificial neural networks stand out by generating models with high precision and low computational cost. With the informatization of the real estate sector, application softwares that provide automated real estate appraisal services can be noted. In addition, considering trends of the information technology, the growth of mobile devices emerges. Through these devices, users can quickly interact with high-precision mobile applications. In this regard, this paper presents the development of a mobile application that uses a artificial neural network to estimate the selling price of apartments. Thus, it is expected to assist decisions of buyers, sellers or any other interested parts with accurate apartment evaluation. The resulting artificial neural network was a Multi Layer Perceptron network that resulted in R2 of 81,14% and mean absolute error of R$ 63.861,71.**

***Keywords* — apartments, real estate valuation, mobile devices, artificial neural networks, regression**

I. DESENVOLVIMENTO

Uma atividade recorrente no mercado imobiliário é a avaliação de apartamentos, que consiste em estimar valores levando em conta características do imóvel e condições econômicas. Uma avaliação rápida e precisa é de grande interesse para clientes, como: compradores, proprietários, investidores e seguradoras [1].

Determinar preços imobiliários é objeto de estudo desde a década de 1950. Contudo, somente em 1974, Rosen propôs uma solução utilizando modelos de regressão, denominados modelos hedônicos ou econométricos [2].

O objetivo dos modelos hedônicos é avaliar o preço de um bem em função de suas características. Para o caso, esta análise envolve atributos internos do imóvel e variáveis do ambiente em que está localizado [2].

A metodologia baseada em modelos econométricos, que usualmente utiliza funções lineares, apresenta certas falhas. Uma dessas imperfeições é desconsiderar possíveis relações não lineares entre parâmetros. Outras falhas são a heterocedasticidade e a multicolinearidade, que ocorrem quando há, respectivamente, variância inconstante do erro na amostra e relações entre variáveis independentes do modelo [2].

Como determinantes para o valor de imóveis, Čeh *et al.* [3] citam: variáveis de acessibilidade, vizinhança e ambiente, em conjunto com características estruturais e temporais.

Com relação às variáveis estruturais e temporais, podem-se listar quantidade de apartamentos e andares, tipo e idade da construção, presença de elevador, entre outros. Ademais, atributos do apartamento, como posição e andar, número de cômodos e características da área privativa, também são determinantes para a estimativa [3].

Sobre o ambiente, encontram-se parâmetros como poluição sonora e visual, distância de margens de rios e altitude em relação ao nível do mar. A respeito da vizinhança, destacam-se atributos específicos da localização e arredores, como: índices de desemprego, região imobiliária e posicionamento geográfico [3].

Por fim, as variáveis de acessibilidade estão relacionadas à distância entre o imóvel e estruturas urbanas, como proximidade de escolas, rodovias, ferrovias, aeroportos ou lojas [3].

Considerando as falhas dos modelos econométricos, Tabales *et al.* [2] propõem o modelo de Redel Neural Artificial (RNA) como uma solução precisa e de fácil implementação.

A RNA é um método computacional projetado para realizar tarefas a partir do aprendizado por observação. Assim como os neurônios do sistema nervoso, esse método gera resultados a partir de exemplos observados [4].

Em relação ao setor imobiliário, uma das aplicações da RNA é a predição de preços de imóveis. Pela captura de incertezas e exploração da correlação entre variáveis, essas redes geram modelos preditivos com custo computacional reduzido e precisão aprimorada [4].

Em decorrência dos diversos fatores que influenciam o preço de um imóvel, a avaliação imobiliária é um processo complexo. Contudo, por administrar bem esses fatores, recentemente, usam-se amplamente redes neurais para estimar o valor de imóveis [5].

Pela estimativa precisa, essas redes auxiliam na tomada de decisão de compradores, vendedores, investidores, governos e instituições financeiras [4].

Atualmente, o mercado imobiliário está mais informatizado. Como exemplo, percebe-se o desenvolvimento de serviços *online* de avaliação automatizada do preço de casas, que utilizam algoritmos e métodos computacionais [4].

Como plataformas de rápido crescimento no quadro da informatização, destacam-se os dispositivos móveis. Fazendo uso de características como conexão sem fio, mobilidade e capacidade de processamento similar a computadores, esses dispositivos alteraram a forma de utilização de aplicativos [6].

Associada a esses atributos, a fácil comercialização em *marketplaces* contribuiu na consolidação de aplicativos móveis no campo da tecnologia da informação [6].

Nesse contexto, o objetivo do trabalho é desenvolver uma aplicação *mobile* que utiliza um modelo de aprendizado de máquina para avaliação do preço de apartamentos. Especificamente, o modelo emprega uma Rede Neural de Regressão do tipo Perceptron Multicamadas. A técnica foi aplicada sobre uma base de dados gerada pelo autor, utilizando-se a linguagem Python, como detalhado em sequência.

A. Descrição dos dados

Para geração da base de dados, utiliza-se um *Web Scraper* que obtém informações relevantes de anúncios de apartamentos listados para venda no site Imovelweb[1].

A *Web Scraping* é uma técnica que consiste em extrair dados não estruturados de sites e estruturá-los em forma de tabela. Estes podem ser armazenados em arquivos ou bancos de dados para futura análise [7].

No presente estudo, o *Web Scraper* utiliza a combinação dos métodos *Hyper Text Markup Language* (HTML) *Parsing* e *Hypertext Transfer Protocol* (HTTP) *Programming*. Enquanto este consiste em obter páginas na forma HTML a partir de requisições HTTP, aquele compreende analisar e extrair dados de interesse de páginas HTML [7].

O *Web Scraper*, desenvolvido em R, foi estruturado principalmente sobre os pacotes *httr*[2], *rvest*[3] e *RPostgres*[4], para realizar, respectivamente, *HTTP Programming*, *HTML Parsing* e armazenamento de informações em banco de dados PostgreSQL.

A extração de dados tem como foco as dez maiores cidades em população de cada estado das regiões Sul e Sudeste do Brasil, conforme estimativa para 2021 do Instituto Brasileiro de Geografia e Estatística (IBGE) [8]. Na Tabela I, como exemplo, estão ordenadas as cidades de maior população presentes no estudo.

Tabela I
MAIORES CIDADES EM POPULAÇÃO

Cidade	População [2021]
São Paulo	12.396.372
Rio de Janeiro	6.775.561
Belo Horizonte	2.530.701
Curitiba	1.963.726
Porto Alegre	1.492.530
Guarulhos	1.404.694
Campinas	1.223.237
São Gonçalo	1.098.357
Duque de Caxias	929.449
São Bernardo do Campo	849.874

A base de dados consolidada registra 383.354 anúncios de apartamentos, que, classificados por estado, ficam distribuídos na forma da Figura 1.

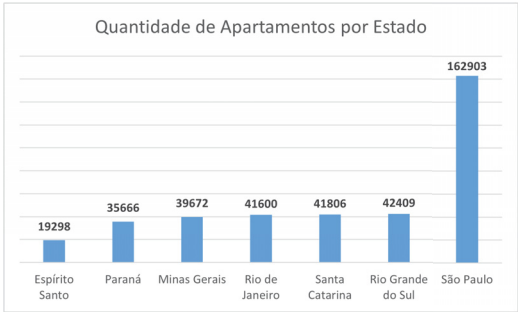


Figura 1. Apartamentos registrados por estado

As características relevantes para determinação do preço dos imóveis foram armazenadas conforme a Tabela II. Dessa forma, a base gerada é distribuída na forma de tabela com 383.354 linhas e 19 colunas, que representam, respectivamente, imóveis e atributos.

B. Métodos

Esta seção apresenta os métodos utilizados para análise dos dados, treinamento do modelo de Inteligência Artificial (IA)

[1]https://www.imovelweb.com.br/

[2]https://cran.r-project.org/web/packages/httr/httr.pdf
[3]https://cran.r-project.org/web/packages/rvest/rvest.pdf
[4]https://cran.r-project.org/web/packages/RPostgres/RPostgres.pdf

Tabela II
ATRIBUTOS DA BASE DE DADOS

Coluna	Descrição	Tipo do Dado
cod	Identificador único	Inteiro
category	Categoria do imóvel	Texto
address	Endereço	Texto
neighborhood	Bairro	Texto
city	Cidade	Texto
estate	Estado	Texto
latitude	Latitude	Real
longitude	Longitude	Real
description	Descrição detalhada do imóvel	Texto
sale	Valor de venda	Real
rent	Valor de aluguel	Real
area_total	Área total	Real
area_util	Área útil	Real
rooms	Número de quartos	Inteiro
suites	Número de suítes	Inteiro
bathrooms	Número de banheiros	Inteiro
garage	Número de vagas de garagem	Inteiro
age	Idade do imóvel	Inteiro
private	Atributos internos e externos	Texto

e desenvolvimento da aplicação. Na Figura 2, observam-se as etapas da análise de dados até a geração do modelo, as quais serão descritas a seguir.

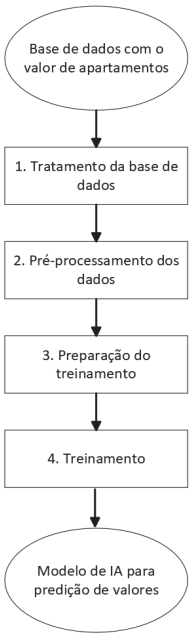


Figura 2. Etapas da Análise de Dados

1) Tratamento da base de dados: A etapa de tratamento, que prepara a base para o pré-processamento, fundamenta-se em um princípio de *Traditional Data Cleaning* citado por Tae *et al.* [9]. Esse princípio consiste em checar se um registro está presente em um conjunto de valores considerados corretos. Em caso de ausência, retifica-se o registro com um valor desse conjunto [9].

O procedimento é necessário para correção dos nomes de bairros, uma vez que há informações incorretas na base. Como exemplo dessas inconsistências, pode-se citar: textos incompletos, incoerentes e bairros inexistentes.

Assim, compararam-se os registros da base de dados com um conjunto de valores de referência. Após identificadas as informações inconsistentes, utilizou-se Python para atualizar os dados com a indicação correta dos bairros.

Como referência, utilizou-se a lista de bairros do site da Empresa Brasileira de Correios e Telégrafos (Correios)[5]. Dessa forma, para cada cidade, empregou-se uma lista de bairros na forma do Algoritmo 1.

Algoritmo 1: Lista de bairros de uma cidade

```
1 curitiba = ['abranches', 'agua verde', 'ahu', [...], 'vila
    izabel', 'vista alegre', 'xaxim']
```

Com a comparação entre referência e registros da base, criou-se uma lista de bairros com nomes inconsistentes para tratamento.

Em sequência, buscou-se a indicação correta dos bairros. Para isso, empregou-se a ferramenta *GeoPy*[6], um serviço de geolocalização que fornece informações sobre lugares com base em suas coordenadas geográficas.

Por meio da função *reverse*, a *GeoPy* apresenta informações sobre uma localização a partir de sua latitude e longitude. Dessa forma, submeteram-se à essa função as coordenadas geográficas dos registros inconsistentes e atualizaram-se os nomes dos bairros com a indicação da ferramenta.

Como exemplo, o Algoritmo 2 demonstra a utilização da *GeoPy* para a latitude -25,4064869 e longitude -49,2502621. Como resposta, a função *reverse* retorna um objeto *Location* contendo informações sobre a coordenada.

Algoritmo 2: Execução da função *reverse* do *GeoPy*

```
1 from geopy.geocoders import Nominatim
2 geolocator = Nominatim(user_agent="application")
3 geolocator.reverse((-25.4064869, -49.2502621))
```

Resultado:

```
Location(
    131, Rua Quintino Bocaiúva, Cabral, Curitiba, Região
    Geográfica Imediata de Curitiba, Região Metropolitana
    de Curitiba, Região Geográfica Intermediária de
    Curitiba, Paraná, Região Sul, 80035-060, Brasil,
    (-25.4064869, -49.2502621, 0.0))
```

Após atualização dos registros, analisou-se novamente a base para verificação de bairros indicados incorretamente. Para identificar os demais dados inconsistentes como valores indisponíveis, atribuiu-se o valor 'na'.

Inicialmente, identificou-se a presença de 22.236 registros com nomes de bairros inconsistentes. Após a correção, com a

[5]https://buscacepinter.correios.com.br/
[6]https://geopy.readthedocs.io/en/stable/

aplicação da ferramenta *GeoPy*, esse valor foi reduzido para 12.050 registros.

Como os registros com valor indisponível para bairros foram desconsiderados do modelo, o tratamento proporcionou um acréscimo de 10.186 registros na base de dados.

Após o tratamento, exportou-se a base de dados atualizada em formato *Comma Separated Values* (CSV).

2) Pré-processamento dos dados: O pré-processamento foi dividido em duas etapas: remoção de *Outliers* e ampliação do *Dataframe*. Para isso, empregou-se Python com implementação de funções das bibliotecas *Pandas*[7] e *Scikit-learn*[8].

Inicialmente, utilizou-se a *Pandas* para importação do arquivo CSV em formato *Dataframe*, no qual os dados são distribuídos em forma de tabela.

Após a importação, realizou-se a exclusão de 9 colunas insignificantes para o estudo, sendo elas: código, categoria, endereço, latitude, longitude, descrição, valor do aluguel e área total. Dessa forma, o *Dataframe* ficou configurado com 10 colunas.

A etapa de remoção de *Outliers* compreendeu identificar e excluir dados considerados como atípicos. Com esse fim, para cada coluna numérica da tabela, aplicou-se a detecção por intervalo interquartil (IQR) conforme a Equação 1 citada por Matos *et al.* [10].

$$IQR = Q_3 - Q_1 \tag{1}$$

Na Equação 1, IQR é o intervalo interquartil; Q_3 é o terceiro quartil; e Q_1 é o primeiro quartil.

Definiram-se o primeiro e terceiro quartis aplicando, respectivamente, os percentuais 25% e 75% sobre os dados ordenados, conforme Leroy e Rousseeuw [11]. Depois de obtido o IQR, calcularam-se os limites inferior e superior, em acordo com a Equação 2 [10].

$$Q_1 - 1,5 \times IQR \leq x \leq Q_3 + 1,5 \times IQR \tag{2}$$

Em sequência, para cada coluna, procedeu-se com a exclusão de dados externos aos limites de aceitação determinados pelo intervalo interquartil.

Como exemplo, demonstra-se a aplicação sobre os preços de venda, que, no primeiro momento, apresentam o valor mínimo de R$ 0,00 e o máximo de R$ 7.670.000,00. Conforme o Diagrama de Caixas apresentado na Figura 3, percebem-se *Outliers*, representados por círculos, que se distanciam da mediana da amostra, retratada pela linha vermelha.

Para determinar o primeiro e terceiro quartil da variável, utilizou-se a função *quantile* da biblioteca *Pandas* com o parâmetro "q" igual a 0,25 e 0,75. Assim, obtiveram-se os valores Q_1 igual a R$ 260.000,00 e Q_3 igual a R$ 745.000,00. Em sequência, por meio da Equação 1, obteve-se o resultado de IQR igual a R$ 485.000,00.

Com o auxílio da Equação 2, definiram-se os limites inferior e superior do valor em R$ -467.500,00 e R$ 1.472.500,00. Por fim, excluíram-se do *Dataframe* os registros com valores de venda externos a esses limites.

Após a exclusão dos *Outliers*, no Diagrama de Caixas da Figura 4, pode-se verificar a distribuição do valor de venda. Em comparação com a Figura 3, percebe-se que os *Outliers* remanescentes não se distanciam tanto da mediana.

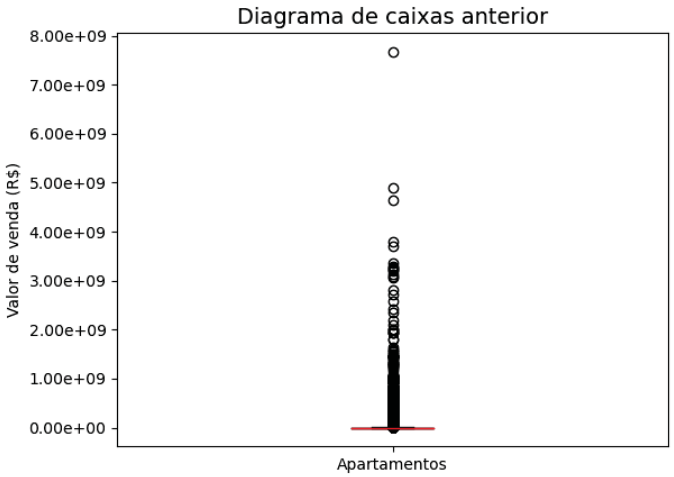


Figura 3. Diagrama de Caixas para a variável valor de venda, anterior a remoção de *Outliers*

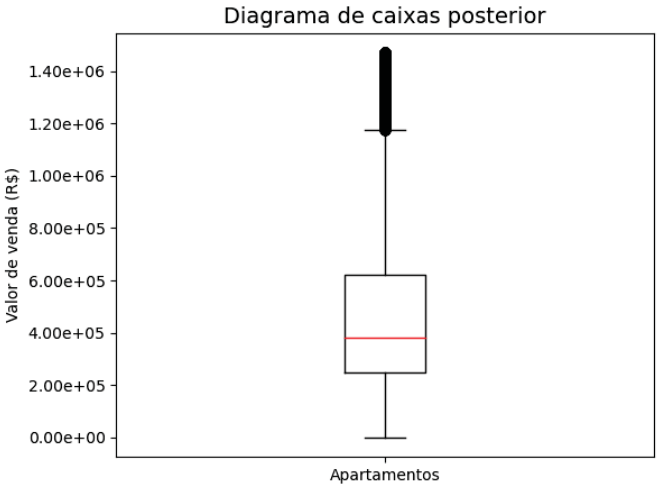


Figura 4. Diagrama de Caixas para a variável valor de venda, posterior a remoção de *Outliers*

Dessa forma, definiram-se os limites de aceitação para os valores das colunas do *Dataframe*. Esses limites, assim como os dados remanescentes no *Dataframe* após a remoção dos *Outliers*, estão representados na Tabela III.

Adicionalmente, para detecção de outras inconformidades no *Dataframe*, aplicou-se a técnica *Isolation Forest*. Baseada na premissa de que anomalias são sempre raras e distantes de aglomerações de dados normais, essa técnica analisa padrões e identifica inconformidades na amostra [12].

[7]https://pandas.pydata.org/

[8]https://scikit-learn.org/stable/index.html

Tabela III
REMOÇÃO DE OUTLIERS

Coluna	Limite Inferior	Limite Superior	Dados Restantes
Valor	-467.500,0	1.472.500,0	344.830
Área útil	-12,0	164,0	314.576
Quartos	0,5	4,5	286.059
Suítes	-1,5	2,5	277.052
Banheiros	-0,5	3,5	272.948
Garagens	-0,5	3,5	272.361
Idade	-25,0	47,0	89.147

Para aplicação da técnica, utilizou-se a função *IsolationForest* do *framework Scikit-learn* sobre o *Dataframe*. Na Figura 5, verifica-se um exemplo da aplicação dessa técnica na análise de uma amostra. No caso, os dados distantes das aglomerações, representados por círculos vermelhos, são conisderados anomalias na amostra.

Após execução da função, como resultado, obteve-se um vetor que classifica os dados em normais ou anormais. Após identificadas as anomalias, mantiveram-se na tabela apenas os dados classificados como normais.

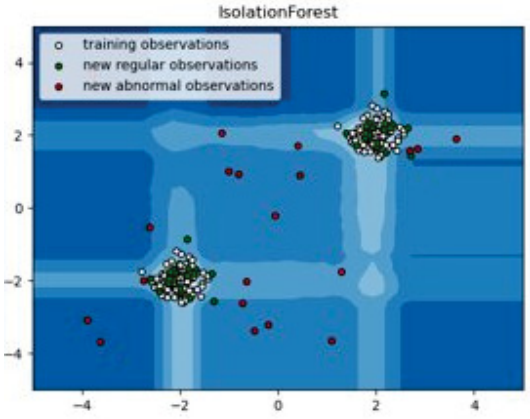


Figura 5. Exemplo da aplicação do *IsolationForest* [17]

No estudo, aplicou-se a *Isolation Forest* sobre as colunas: valor de venda, área útil, quartos, suítes, banheiros, garagem e idade do imóvel. Após a remoção dos *Outliers*, restaram 66.426 registros no *Dataframe*.

De outro modo, a etapa de ampliação do *Dataframe* consistiu em agregar informações aos registros da tabela. Para tal fim, realizou-se identificação de atributos de imóvel, inclusão de índices socioeconômicos e avaliação da área de cada bairro.

a) Atributos do imóvel: Considerando o impacto de atributos sobre o valor dos imóveis mencionado por Ceh *et al.* [3], efetuou-se análise para identificação dessas características.

Para tal fim, realizou-se a análise textual da coluna *private*, que contém atributos internos e externos do apartamento. Assim, identificou-se a presença ou ausência de determinadas características no imóvel.

Como regra de negócio, os atributos internos e externos de interesse se apresentam na forma da Tabela IV.

Tabela IV
ATRIBUTOS DOS APARTAMENTOS

Internos	Externos
Sacada (Varanda)	Elevador
Churrasqueira	Portaria
Ar condicionado	Salão de festas
Ensolorado	Academia
Aquecimento à gás	Playground

A análise textual compreendeu a verificação de cada elemento da lista de atributos do imóvel. Caso um atributo de interesse estivesse contido nessa lista, registrou-se a presença desse parâmetro para o apartamento.

Trataram-se esses parâmetros qualitativamente na forma de variáveis categóricas. Para isso, criou-se uma coluna para cada variável com valores 0 ou 1, os quais indicaram ausência ou presença do atributo.

Após a ampliação com atributos dos imóveis, criaram-se 10 colunas na tabela de dados. Portanto, ao final da etapa, a tabela apresentou o total de 20 colunas.

b) Índices socioeconômicos: Em sequência, procedeu-se com a ampliação do *Dataframe* com a inclusão de índices socioeconômicos referentes a cada cidade. Com esse intuito, utilizaram-se dados provenientes do IBGE [8] e do Instituto de Pesquisa Econômica Aplicada (IPEA) [13], conforme Tabela IV.

Tabela V
ÍNDICES SOCIOECONÔMICOS DAS CIDADES

Dados do IBGE
Densidade Demográfica (2010)
Salario medio mensal dos trabalhadores formais (2019)
Populacao ocupada (2019)
Taxa de escolarizacao de 6 a 14 anos de idade (2010)
IDEB – anos iniciais do ensino fundamental (rede publica) (2019)
IDEB – anos finais do ensino fundamental (rede publica) (2019)
PIB per capita (2019)
Índice de desenvolvimento humano municipal (idhm) (2010)
Mortalidade infantil (2020)
Internacoes por diarreia (2016)
Esgotamento sanitario adequado (2010)
Arborizacao de vias publicas [2010]
Urbanizacao de vias publicas [2010]
Dados do IPEA
Taxa de homicidios [2019]
Taxa de suicidios [2019]
Taxa de acidentes de transito com obito [2019]

Para cada registro, inseriu-se na tabela o valor do índice referente a cidade em que o imóvel está localizado. Após essa ampliação, a dimensão do *Dataframe* ficou alterada para 36 colunas.

c) Avaliação do bairro: Para avaliação da área por bairro, após a remoção de registros com bairros indisponíveis e dos *Outliers* das colunas valor e área útil, trataram-se os dados referentes à localização.

Primeiramente, definiram-se os limites de quantidade mínima para manter cidades e bairros na análise. Para cidades,

definiu-se esse limite em 250 registros, enquanto que para bairros o limite foi de 25 registros. Em seguida, descartaram-se as localidades que apresentaram o total de registros inferior a essas quantidades.

Em sequência, por meio do agrupamento da tabela por bairros, gerou-se uma lista de cidades e bairros presentes no estudo. Dessa forma, a partir do valor de venda e da área útil, pode-se calcular o valor médio da área de cada bairro.

Para isso, criou-se na tabela uma coluna contendo a divisão do valor pela área útil do imóvel. Após, procedeu-se com o agrupamento desses valores por bairro utilizando a mediana.

O agrupamento contendo a mediana do valor por área de cada bairro foi armazenado em uma nova coluna no *Dataframe*. Assim, a dimensão da tabela foi incrementada em 1 coluna.

Após a inclusão de atributos do imóvel, índices socioeconômicos e valor do bairro, o *Dataframe* ficou configurado com 34 colunas, conforme a Tabela VI.

3) Preparação do treinamento: Na etapa de preparação, adequou-se a forma do *Dataframe* ao treinamento de uma rede neural. Para isso, separaram-se as variáveis independentes da variável dependente e dividiu-se a base em treino e teste.

Uma rede neural simples apresenta uma camada de entrada, que contém neurônios que se projetam adiante para uma camada de saída. Em redes com múltiplas camadas, há presença de uma ou mais camadas ocultas, cujos nós são intitulados neurônios ocultos. Esses neurônios atuam entre as camadas de entrada e saída, extraindo estatísticas de ordem elevada [14].

Na Figura 6, observa-se a estrutura de uma rede, em que a camada de entrada fornece os sinais de entrada, em forma de vetor, a uma primeira camada oculta. Os sinais de saída dessa camada oculta são encaminhados adiante a uma terceira, e assim até o fim da rede. Por fim, a resposta global da rede é obtida nos sinais de saída da última camada [14].

Essa rede, também conhecida como *perceptron* de múltiplas camadas, tem como destaque o treinamento supervisionado com algoritmo de retropropagação de erro. Por meio desse algoritmo, a rede produz um sinal de erro entre a resposta real e a desejada, que é propagado para trás na rede. Dessa forma, a rede ajusta sua configuração para aproximar a resposta desejada da resposta real [14].

Para adequar o *Dataframe* à estrutura da rede neural, procedeu-se com a partição dos dados em variáveis dependentes e independentes. Esse ajuste é necessário para que os neurônios de entrada recebam somente as variáveis independentes, também denominadas preditoras. Por outro lado, a variável dependente é resultante da saída na última camada da rede [15].

Nessa partição, separou-se a coluna referente ao valor de venda (variável dependente) das demais colunas do *Dataframe*. Assim, essas colunas se configuram como variáveis preditoras do valor de venda dos imóveis.

Em sequência, dividiram-se os dados em conjuntos de treino e teste. Assim, na etapa de treinamento, submete-se o conjunto de treino aos parâmetros da rede neural, que se ajusta aos dados. Ainda no treinamento, oculta-se o conjunto de teste, que é utilizado posteriormente para avaliar o erro e aprimorar hiperparâmetros [16].

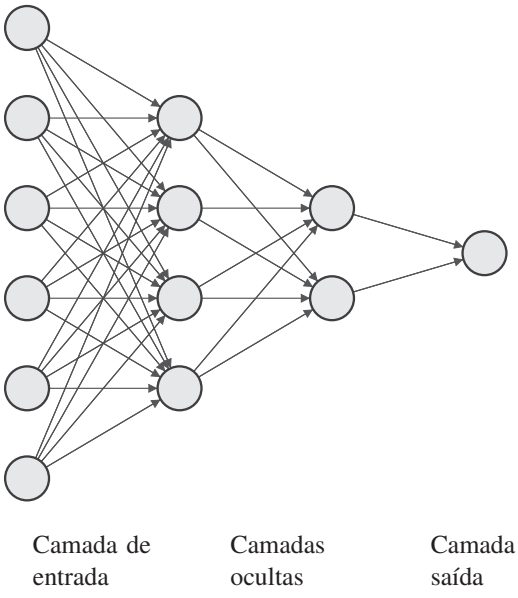


Figura 6. Estrutura de uma Rede Neural Multicamadas do tipo *feed-foward*

Com esse intuito, utilizou-se a função *train_test_split* do *framework Scikit-learn* para dividir a base na proporção de 65% para treino e 35% para teste. Sob a base de treino, após análise da distribuição de dados, aplicaram-se técnicas de normalização e transformação.

A aplicação dessas técnicas justifica-se pelo aumento da precisão de algoritmos do *framework Scikit-learn* em decorrência da normalização da distribuição de dados [17].

Como os atributos dos imóveis são representados por valores de 0 a 1, optou-se pela redução da amplitude das demais variáveis dependentes para essa mesma escala.

Com esse fim, utilizou-se a função *MinMaxScaler* do *framework Scikit-learn*, que transforma os dados para uma escala de 0 (valor mínimo) a 1 (valor máximo) [17]. Para isso, aplicou-se a Equação 3 sobre cada coluna do *Dataframe*.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3}$$

Na Equação 3, X_{scaled} é o dado transformado; X é o dado atual; X_{min} é o valor mínimo da coluna; e X_{max} é o valor máximo da coluna.

Em sequência, para ampliar a exatidão do modelo, realizaram-se a transformação e normalização da variável independente. Para isso, utilizou-se a técnica de transformação seguida de padronização através da função *PowerTransformer* do *framework Scikit-learn*.

Como todos os valores da amostra são positivos, selecionou-se o método Box-Cox para transformar os dados. Já para normalizar a distribuição após a transformação, definiu-se o parâmetro *standardize* como *True* na função *PowerTransformer* [17].

Tabela VI
DISTRIBUIÇÃO DE DADOS PRÉ-PROCESSADOS

Índice	Coluna	Tipo	Amplitude	Unidade
1	Valor de venda	Real	1 – 1.455.230,00	R$
2	Área útil	Real	1 – 155	m²
3	Quartos	Inteiro	1 – 4	Unidade
4	Suítes	Inteiro	0 – 2	Unidade
5	Banheiros	Inteiro	0 – 3	Unidade
6	Garagens	Inteiro	0 – 3	Unidade
7	Idade	Inteiro	0 - 47	Anos
8	Valor do bairro	Real	2100,00 – 19.800,00	R$/m²
9	Varanda	Booleano	0 - 1	
10	Churrasqueira	Booleano	0 - 1	
11	Ar-condicionado	Booleano	0 - 1	
12	Ensolarado	Booleano	0 - 1	
13	Aquecimento a gás	Booleano	0 - 1	
14	Elevador	Booleano	0 - 1	
15	Portaria	Booleano	0 - 1	
16	Salão de festas	Booleano	0 - 1	
17	Academia	Booleano	0 - 1	
18	Playground	Booleano	0 - 1	
19	Densidade Demográfica	Real	65,43 – 10.264,80	Pessoas/m²
20	Salário médio de trabalhadores	Real	1,90 - 4,50	Salários Mínimos/mês
21	População ocupada	Real	9,20 – 67,50	Pessoal ocupado/População estimada
22	Taxa de escolarização (6 a 14 anos)	Real	95,60 – 98,50	Percentual
23	IDEB – Anos iniciais	Real	0,00 – 7,20	IDEB
24	IDEB – anos finais	Real	3,60 – 5,80	IDEB
25	PIB per capita	Real	12.976,50 – 130.034,00	R$
26	Índice de Desenvolvimento Humano	Real	0,684 – 0,847	IDH
27	Mortalidade infantil	Real	5,89 – 18,10	Óbitos/1.000 nascidos vivos
28	Internações por diarreia	Real	0 – 0,8	Internações/1.000 habitantes
29	Esgotamento sanitário adequado	Real	57,1 – 98,40	Percentual
30	Arborização de vias publicas	Real	11,30 – 97,30	Percentual
31	Urbanização de vias publicas	Real	5,30 – 90,60	Percentual
32	Taxa de homicídios	Real	3,71 – 36,21	Homicídios/100.000 habitantes
33	Taxa de suicídios	Real	1,22 – 16,06	Suicídios/100.000 habitantes
34	Taxa de acidentes de trânsito com óbito	Real	4,46 – 26,51	Acidentes/100.000 habitantes

4) Treinamento: Para a predição dos resultados, utilizou-se uma rede neural de regressão do tipo perceptron multicamadas implementado pela classe *MLPRegressor* do *framework Scikit-learn*.

A rede *MLPRegressor* aplica a técnica de aprendizado supervisionado em uma rede neural do tipo *feed-forward*, caracterizada por múltiplas entradas, uma ou mais camadas ocultas e uma única saída [17].

Para a camada de entrada da rede, destinaram-se as 33 variáveis independentes presentes na Tabela VI. Assim, a primeira camada do algoritmo é composta por 33 neurônios. Como saída da rede, obteve-se o valor estimado da variável dependente, no caso, o preço do imóvel.

Para a classe *MLPRegressor*, os principais hiperparâmetros disponíveis são: dimensões da camada oculta (*hidden_layer_sizes*), função de ativação da camada oculta (*activation*), solucionador (*solver*), grau de aprendizado (*learning_rate*), tamanho de lote (*batch_size*), limite de iterações até convergência (*max_iter*), tolerância para melhoria (*tol*), limite de iterações sem melhoria (*n_iter_no_change*) [17].

Para determinar a combinação de hiperparâmetros ideal para o modelo, utilizou-se uma busca em *grid* através da classe *GridSearchCV* do *framework Scikit-learn*. A faixa de valores testados pode ser observada na Tabela VII.

Tabela VII
HIPERPARÂMETROS DO GRIDSEARCHCV

Parâmetro	Valores
hidden_layer_sizes	(9), (17), (33), (9,9), (17,17), (33,33), (33,17), (33,9), (17,9)
activation	logistic, relu
solver	lbfgs, sgd, adam
learing_rate	constant, invscaling, adaptative
batch_size	50, 100, 150, 200
max_iter	5.000
tol	0.00001
n_iter_no_change	50

Como parâmetros para a dimensão da camada, utilizaram-se combinações de um quarto, metade e total de variáveis independentes, para uma e duas camadas.

Para a função de ativação, testaram-se as funções logística e retificação linear unitária (relu). Já em relação a otimização de pesos para os neurônios, avaliaram-se todos os solucionadores disponíveis: *Limited-memory Broyden–Fletcher–Goldfarb–Shanno* (LBFGS), *Stochastic Gradient Descent* (SGD) e *Adaptive Moment Estimation* (ADAM).

Em referência ao grau de aprendizado, consideraram-se todos os métodos disponíveis: constante, adaptativo e invscaling. Por fim, a respeito do tamanho dos lotes, testaram-se os valores: 50, 100, 150 e 200.

Como estratégia de avaliação das combinações de parâmetros, utilizou-se o erro médio absoluto. Para isso, indicou-se o valor *neg_mean_absolute_error* ao parâmetro *scoring* do *GridSearchCV*.

Para encontrar a melhor combinação de parâmetros, a busca em *grid* foi executada por 14 horas e 20 minutos. Após, o *GridSearchCV* definiu como ideais os parâmetros apresentados na Tabela VIII, enquanto os demais melhores resultados podem ser observados na Figura 7.

Tabela VIII
HIPERPARÂMETROS DO GRIDSEARCHCV

Parâmetro	Valores
hidden_layer_sizes	(33, 9)
activation	logistic
solver	adam
learing_rate	invscaling
batch_size	200
max_iter	5.000
tol	0.00001
n_iter_no_change	50

Após a definição dos parâmetros ótimos para o modelo, treinou-se a rede com toda a base de dados. Em sequência, para realizar a predição de valores, alocou-se a rede neural gerada na aplicação.

5) Desenvolvimento da Aplicação: A Aplicação *Mobile* tem como finalidade permitir que usuários avaliem apartamentos. Para isso, o usuário submete atributos do imóvel à aplicação, que retorna um valor estimado pela rede neural, conforme arquitetura apresentada na Figura 8.

Pode-se dividir a arquitetura em duas estruturas, *Frontend* e *Backend*. Enquanto essa é executada em servidor e intermedia funcionalidades com a rede neural, aquela é executada no cliente e apresenta a interface gráfica ao usuário final.

Quanto à comunicação entre essas estruturas, mediante requisições HTTP, enviam-se as informações do *Frontend* ao *Backend*, que as submete a rede neural. Essa rede avalia as informações e gera o valor predito do imóvel, que é encaminhado ao *Frontend* e apresentado ao usuário.

Para o desenvolvimento do *Frontend*, utilizou-se a *Javascript* e o *framework React Native*. Já para o *Backend*, empregou-se *Python* e o *framework FastApi*.

C. Tecnologias

A aplicação e os scripts foram desenvolvidos em computador pessoal com processador i7 de 6-núcleos a 3,7 GHz, com sistema operacional Microsoft Windows 10 Home.

Foram utilizadas as seguintes aplicações e bibliotecas:

- Linguagem Javascript;
- Linguagem Python, versão 3.9.2;
- Linguagem R, versão 4.0.2;
- pgAdmin 4, versão 5.0;
- Anaconda Spyder, versão 4.1.5;
- RStudio, versão 1.4.1106;
- Visual Studio Code, versão 1.69.2;
- Pacote httr, versão 1.4.2;
- Pacote RPostgres, versão 1.4.1;
- Pacote rvest, versão 1.0.1;
- Pacote stringr, versão 1.4.0;
- Pacote stringi, versão 1.5.3;
- Pacote geopy, versão 2.2.0;
- Pacote psycopg2, versão 2.8.6;
- Pacote joblib, versão 1.0.1;
- Pacote matplotlib, versão 3.4.1;
- Pacote numpy, versão 1.19.5;
- Pacote pandas, versão 1.4.1;
- Pacote scikit-learn, versão 0.24.1;
- Framework FastApi, versão 0.75.1;
- Framework React Native, versão 0.68.0;

II. RESULTADOS E DISCUSSÕES

Esta seção apresenta os resultados obtidos no desenvolvimento do modelo de inteligência artificial, assim como o levantamento de requisitos e os diagramas referentes a elaboração da aplicação.

A. Métricas

A precisão do modelo apresentado neste trabalho foi medida por meio de três métricas: *Mean Absolut Error* (MAE), R^2 *Score* (R^2) e *Mean Squared Logarithmic Error* (MSLE). O R^2, ou coeficiente de determinação, mensura a proporção que o modelo se ajusta aos dados, no caso, a correlação entre as características do imóvel e o preço. Por outro lado, MAE e MSLE são medidas estatísticas referentes ao erro contido nas estimativas produzidas pelo modelo.

Levando em conta os hiperparâmetros ideais resultantes da busca em grid, o melhor resultado do modelo apresentou: MAE igual a R$63.861,71, R^2 igual a 81,14% e MSLE igual a 0,0571. Ainda assim, esse resultado é inferior ao registrado por Tabales *et al.* (2013), que apresentou MAE igual a €28.551,34 e R^2 igual a 86,05

Embora ambos os estudos usem o modelo Perceptron multicamadas para a avaliação de apartamentos, diferenças na base e pré-processamento de dados podem justificar a desigualdade de resultados.

Quanto à base de dados, Tabales *et al.* (2013) utilizam transações imobiliárias registradas em uma única cidade na Espanha, que apontam o preço efetivo da venda de imóveis. Já em relação ao pré-processamento, os autores organizam as variáveis preditoras do modelo em grupos, relativos ao ambiente interno, estrutura externa, aspectos da construção e localização. Dessa forma, as características dos apartamentos não são submetidas de forma individual à avaliação da rede neural.

B. Requisitos

Os requisitos da aplicação dividem-se em funcionais e não funcionais. Na Tabela IX, apresentam-se os requisitos

Rank	mean test score	std test score	Activation	Batch Size	Hidden Layers	Learning Rate	Solver	mean fit time	std fit time	mean score time	std score time
1	-0,30425	0,00156	logistic	200	(33, 9)	invscaling	adam	357,10393	48,97820	0,01250	0,00625
2	-0,30449	0,00226	logistic	150	(17, 9)	constant	lbfgs	294,41936	9,30578	0,00625	0,00765
3	-0,30450	0,00162	logistic	50	(17, 9)	constant	adam	505,95597	41,92247	0,01250	0,00625
4	-0,30450	0,00178	logistic	200	(17, 17)	invscaling	lbfgs	368,85613	16,82088	0,01250	0,01169
5	-0,30462	0,00262	logistic	100	(17, 9)	constant	lbfgs	261,40238	8,43723	0,00938	0,00766
6	-0,30474	0,00256	logistic	150	(17, 9)	invscaling	lbfgs	248,24157	14,35281	0,00313	0,00625
7	-0,30488	0,00261	logistic	100	(33, 9)	constant	adam	452,04935	47,30680	0,00937	0,00765
8	-0,30490	0,00297	logistic	100	(33, 17)	constant	adam	575,93454	133,34863	0,00625	0,00765
9	-0,30495	0,00142	logistic	100	(33, 9)	invscaling	adam	433,24835	61,08714	0,01250	0,00625
10	-0,30497	0,00264	logistic	200	(17, 9)	constant	lbfgs	310,80861	9,46048	0,00938	0,00766

Figura 7. Melhores resultados apresentados pelo *GridSearchCV*

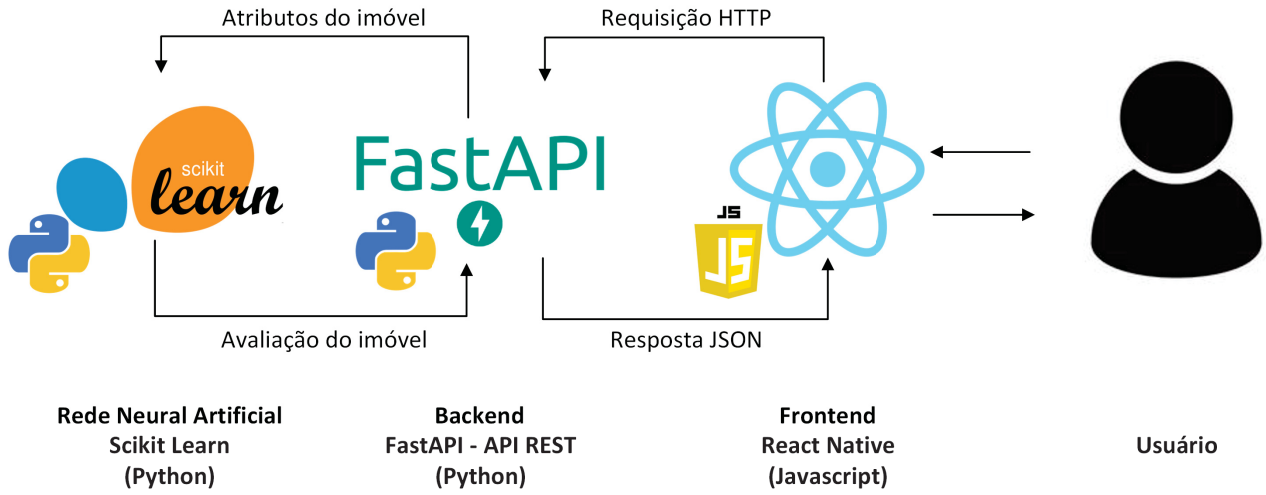


Figura 8. Arquitetura da Aplicação *Mobile*

funcionais, que especificam as funções que a aplicação executa durante a operação por um usuário.

Por outro lado, na Tabela X, mostram-se os requisitos não funcionais, que definem a forma com que a aplicação realizará suas funções.

C. Diagrama de Casos de Uso

Na Figura 9, verifica-se o diagrama de casos de uso, que descreve as funcionalidades presentes na aplicação. Dentre elas, estão: avaliar um imóvel com base em suas características e imprimir um relatório com detalhes sobre a avaliação e a rede neural artificial.

As telas do sistema referentes ao caso de uso avaliar imóvel estão apresentadas na Figura 10. Por meio delas, o usuário informa localização, características e atributos do imóvel. Após, o sistema avalia as informações e apresenta a avaliação do apartamento.

Em sequência, o usuário pode emitir um relatório detalhado na forma da Figura 11. Esse relatório consolida informações sobre o apartamento avaliado e características da rede neural artificial, como: algoritmo utilizado, hiperparâmetros e métricas de precisão.

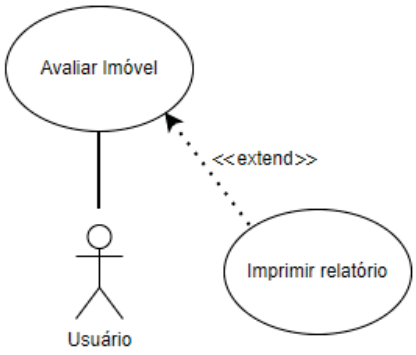


Figura 9. Diagrama de Casos de Uso

Tabela IX
REQUISITOS FUNCIONAIS

Requisito	Nome	Descrição
RF01	Selecionar local do apartamento	O usuário seleciona o local do imóvel, informando estado, cidade e bairro.
RF02	Selecionar características do apartamento	O usuário seleciona as características do imóvel, informando área útil e idade do imóvel, assim como a quantidade de suítes, quartos, banheiros e garagens.
RF03	Selecionar atributos do apartamento	O usuário seleciona os atributos internos e externos do imóvel, informando presença ou ausência dos atributos da Tabela IV.
RF04	Listar locais disponíveis	O sistema apresenta os locais (Estado, Cidade e Bairro) disponíveis para avaliação imobiliária em forma de lista.
RF05	Disponibilizar informações sobre cidades	O sistema apresenta informações do IBGE sobre as cidades, como Índice de Desenvolvimento Humano (IDH) e Percentual da População Ocupada.
RF06	Avaliar apartamento	O sistema avalia as informações fornecidas pelo usuário e apresentará uma avaliação do imóvel.
RF07	Imprimir relatório	O usuário pode imprimir um relatório detalhado com a avaliação o imóvel e informações sobre o modelo de Inteligência Artificial utilizado.

Tabela X
REQUISITOS NÃO FUNCIONAIS

Requisito	Nome	Descrição
RNF01	Mobile	O sistema deve ser executado em dispositivos móveis (plataforma *Mobile*)
RNF02	Python	O backend do sistema deve ser desenvolvido em Python para integração com o modelo de Inteligência Artificial.
RNF03	JavaScript com React Native	O frontend do sistema deve ser desenvolvido em Javascript com o framework React Native.
RNF04	Selecionar valor de características em Slider	O sistema permitirá ao usuário atribuir valor às características do imóvel conforme limite predeterminado.
RNF05	Selecionar atributos em Checklist	O sistema permitirá ao usuário preencher atributos do imóvel na forma de CheckList.
RNF06	Relatório em formato PDF	O sistema criará relatório da avaliação em formato Portable Document File (PDF) para consolidação e compartilhamento de informação.

D. Diagrama de Classes

Na Figura 12, verifica-se o diagrama de classes, que representa o objeto da aplicação com seus atributos. Assim, percebem-se os atributos da classe apartamento a serem avaliados pela rede neural artificial.

E. Diagrama de Sequência

Na Figura 13, apresenta-se o diagrama de sequência referente ao caso de uso avaliar imóvel. Por meio do diagrama, pode-se perceber a sequência de interações entre usuário e sistema, desde o acesso e as interações com as telas do sistema até a avaliação do apartamento.

Figura 10. Telas da aplicação referentes ao caso de uso avaliar imóveis

F. Considerações Finais

O presente trabalho teve como objetivo desenvolver uma aplicação *mobile* que prediz preços de apartamentos com base em suas características. Para isso, treinou-se uma Rede Neural Artificial do tipo perceptron multicamadas com base em anúncios de imóveis extraídos do site Imovelweb.

A Rede Neural treinada fica localizada no *Backend* da aplicação, desenvolvido com o *framework FastApi* em *Python* e executado em servidor. Já o *Frontend*, elaborado com o *framework React Native* em *Javascript*, apresenta a interface gráfica da aplicação *mobile* ao usuário.

Com base no estudo proposto, o modelo de inteligência artificial alcançou precisão de R^2 igual a 81,14%, MAE de R$63.861,71 e MSLE igual a 0,0571. Quanto à precisão, aplicou-se a métrica R^2 para determinar a proporção com que a variação na estimativa do preço é explicada por características do imóvel. Com relação à predição, em amostras com valores contínuos de grande amplitude, o MAE permite avaliar o erro médio em unidade absoluta e o MSLE o erro percentual entre valor real e predito.

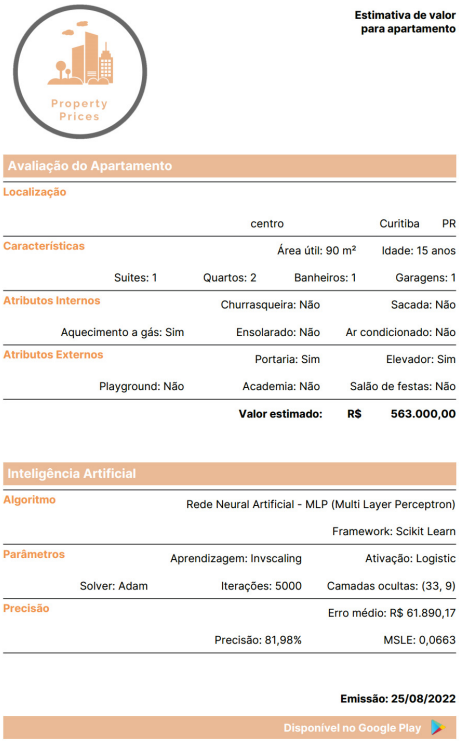


Figura 11. Relatório de avaliação gerado pela aplicação

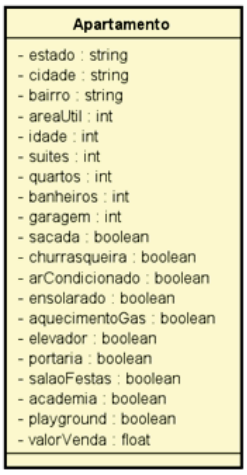


Figura 12. Diagrama de Classes

Além disso, a aplicação é limitada a avaliação de apartamentos situados nas dez maiores cidades de cada estado das regiões Sul e Sudeste do Brasil. Outra limitação é relativa às características definidas como variáveis independentes, já que há restrição da avaliação a seis atributos quantitativos e dez qualitativos.

Recentemente, no cenário do aprendizado de máquina, destaca-se o aprendizado profundo. Por aprender relações de múltiplos níveis em dados estruturados de forma supervisionada e não supervisionada, o aprendizado profundo pode ser aplicado a avaliação imobiliária [19].

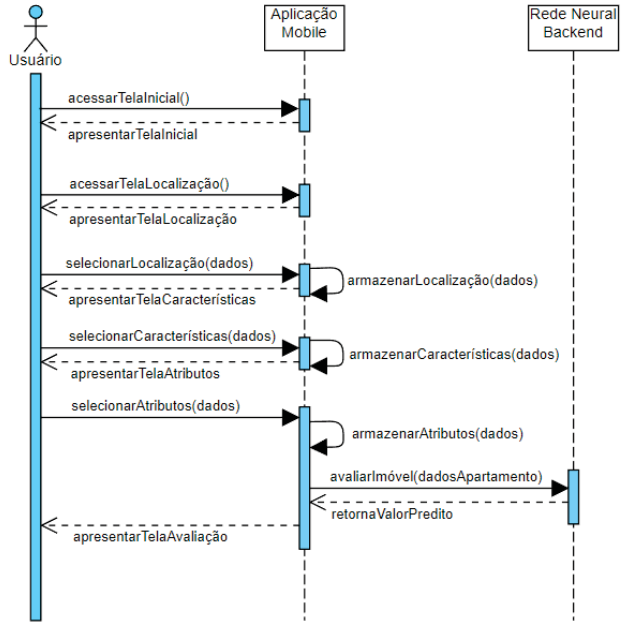


Figura 13. Diagrama de Sequência do Caso de Uso Avaliar imóvel

Em termos de desempenho e precisão, modelos de aprendizado profundo, como redes neurais profundas, podem superar os modelos convencionais. Contudo, além de requisitar maior base de dados, o treinamento desses modelos exige maior tempo de processamento [19].

Como trabalhos futuros, iniciativas podem ser tomadas para expansão de locais englobados pelo estudo e melhoria de resultados. Uma delas seria expandir a base de dados, com apartamentos de cidades menores ou de cidades de outros estados. Além disso, pode-se também ampliar a base com dados provenientes de outros sites de anúncios de apartamentos.

Outra iniciativa seria ampliar ou alterar a combinação de atributos dos apartamentos submetidos à análise da rede neural. Para isso, podem ser extraídas diferentes características da descrição detalhada do imóvel, presente na base de dados.

Referências

[1] W. Z. Taffese, “Case-based reasoning and neural networks for real estate valuation”. Artificial Intelligence and Applications, 98-104, 2007.

[2] J. M. N. Tabales, J. M. C. y Ocerin, F. J. R. Carmona, “Artificial neural networks for predicting real estate price”. Revista de Métodos Cuantitativos para la Economía y la Empresa, vol. 15, 29-44, 2013.

[3] M. Čeh, M. Kilibarda, A. Lisec, B. Bajat, “Estimating the performance of random forest versus multiple regression for predicting prices of the apartments.” International Journal of Geo-Information, vol. 7, n. 5, 168, 2018. DOI: 10.3390/ijgi7050168

[4] P. J. Nkolika, H. I. Okagbue, E. C. M. Obasi, A. O. Akinola, “Review on the application of artificial neural networks in real estate valuation”. International Journal, vol. 9, n. 3, 2020.

[5] J. Cetkovic, S. Lakic, M. Lazarevska, M. Zarkovic, S. Vujosevic, J. Cvijovic, M. Gogic. “Assessment of the real estate market value in the European market by artificial neural networks application”. Complexity, vol. 2018, 2018. DOI: 10.1155/2018/1472957

[6] J. Dehlinger, J. Dixon. “Mobile application software engineering: Challenges and research directions”. Workshop on mobile software engineering, vol. 2, 29-32, 2011.
[7] S. C. M. de S. Sirisuriya, “A comparative study on web scraping”. Proceedings of 8th International Research Conference, 2015.
[8] Instituto Brasileiro de Geografia e Estatística - Portal Cidades@. Disponível em: https://cidades.ibge.gov.br/brasil/panorama. Acesso em julho de 2022
[9] K. H. Tae, Y. Roh, Y. H. Oh, H. Kim, S. E. Whang, “Data cleaning for accurate, fair, and robust models: A big data-AI integration approach”. Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning, 1-4, 2019. DOI: 10.1145/3329486.3329493
[10] S. N. Matos, G. G. Netto, V. N. Lage, A. K. R. Segundo, M. F. Braga, “Tratamento de dados: uma abordagem prática para aprendizagem de atenuaçao de ruıdos e eliminaçao de outliers.” 14º Simpósio Brasileiro de Automação Inteligente, 2019.
[11] P. J. Rousseeuw, A. M. Leroy, “Robust regression and outlier detection.” John wiley e sons, 2005.
[12] Z. Ding, M. Fei, “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window.” 3rd IFAC International Conference on Intelligent Control and Automation Science, vol. 46, n. 20, 12-17, 2013. DOI: 10.3182/20130902-3-CN-3020.00044
[13] Instituto de Pesquisa Econômica Aplicada - Sistema Ipeadata. Disponível em: http://www.ipeadata.gov.br. Acesso em julho de 2022
[14] S. Haykin. “Redes neurais: princípios e prática”. Bookman Editora, 2001.
[15] SC. Wang, “Artificial Neural Network. In: Interdisciplinary Computing in Java Programming”. The Springer International Series in Engineering and Computer Science, vol 743, 2003. DOI: 10.1007/978-1-4615-0377-4_5
[16] J. J. Salazar, L. Garland, J. Ochoa, M. J. Pyrcz (2022). “Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy”. Journal of Petroleum Science and Engineering, vol. 209, p. 109885, 2022.
[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning, vol. 12, p. 2825-2830, 2011. Disponível em: https://scikit-learn.org/stable/index. Acesso em julho de 2022
[18] D. Radulovic, D. Negovanovic, “Gait speed prediction based on walking parameters using MLPRegressor”. RISTEM-2021 Student Scientific Conference, 13-18, 2021. ISBN: 978-953-8246-22-7
[19] H. Xu, A. Gade, “Smart real estate assessments using structured deep neural networks”. IEEE SmartWorld, Ubiquitous Intelligence e Computing, Advanced e Trusted Computed, Scalable Computing e Communications, Cloud e Big Data Computing, Internet of People and Smart City Innovation, p. 1-7, 2017.