DOI: 10.5748/19CONTECSI/PSE/CID/7007

AVALIAÇÃO IMOBILIÁRIA AUTOMÁTICA: APLICAÇÃO DE REDE NEURAL ARTIFICAL PARA PREDIÇÃO DO VALOR DE VENDA DE APARTAMENTOS

André Klingenfus Antunes ; https://orcid.org/0000-0002-6555-0690
Universidade Federal do Paraná (UFPR)

Rafaela Mantovani Fontana ; https://orcid.org/0000-0001-6350-4167
Universidade Federal do Paraná (UFPR)

AUTOMATIC REAL STATE APPRAISAL: AN ARTIFICIAL NEURAL NETWORK APPLICATION TO PREDICT APARTMENTS SALE VALUE

ABSTRACT

Automatic real state appraisal is a useful endeavor for anyone interested in real state market and, specially, in smart cities, as it contributes to urban management innovation. One of the ways to automate real state appraisal is using artificial intelligence techniques. Regression models can use properties characteristics to estimate their value and, currently, artificial neural networks are intensively used in this field, as they can generate high precision models with low computational cost. Based on this potential, we here present a smartphone application that uses an artificial neural network to predict apartments sale value. The article describes how we created our database, the properties characteristics, as well as models training. The application was developed using Python and React Native framework. 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 appraisal, mobile devices, artificial neural networks, regression models.

AVALIAÇÃO IMOBILIÁRIA AUTOMÁTICA: APLICAÇÃO DE REDE NEURAL ARTIFICAL PARA PREDIÇÃO DO VALOR DE VENDA DE APARTAMENTOS

RESUMO

Avaliar precisa e automaticamente um imóvel é uma atividade importante para interessados no mercado imobiliário e, em cidades inteligentes, é um serviço que contribui para a modernização da gestão urbana. Uma das formas de realizar esta automatização é com inteligência artificial – ou modelos de regressão – que estimam o valor de um imóvel com base em suas características. Atualmente, destacam-se as redes neurais artificiais, que geram modelos com alta precisão e baixo custo computacional. De forma a se beneficiar deste potencial, este estudo tem como objetivo o desenvolvimento de uma aplicação móvel que utiliza uma rede neural artificial para avaliar apartamentos. Descreveu-se em detalhes a criação da base dados, com a constituição das características dos imóveis, bem como o preparo e treinamento do modelo de rede neural criado. A aplicação foi desenvolvida utilizando-se a linguagem Python e o *framework* React Native. A rede neural gerada pelo modelo, do tipo Perceptron com multicamadas de regressão, apresentou R^2 de 81,14% e erro médio absoluto de R$ 63.861,71.

Palavras-chave: apartamentos, avaliação imobiliária, dispositivos móveis, redes neurais artificiais, modelos de regressão.

1.Introdução

Smart Cities – ou cidades inteligentes – são aquelas que permitem uma cidadania participativa e integrada a fenômenos econômicos e sociais, viabilizada principalmente pelo uso das Tecnologias da Informação e Comunicação (TICs) (Cunha, Przeybilovicz, Macaya & Burgos, 2016). Elas utilizam "tecnologia para prestar de forma mais eficiente os serviços urbanos, melhorar a qualidade de vida das pessoas e transformar a relação entre entidades locais, empresas e cidadãos, proporcionando uma nova forma de viver na cidade" (Cunha *et al.*, 2016, p. 28).

Com o crescimento da população nos centros urbanos, o número de imóveis geridos pelas cidades – e as informações a eles relacionadas – cresce proporcionalmente (Akar & Yalpır, 2021). As informações dos imóveis, como características do imóvel e condições econômicas, são a base para determinação do seu valor (Taffese, 2007) e, consequentemente, para os sistemas de avaliação imobiliária. Como fatores determinantes para o valor de imóveis, encontram-se características estruturais e temporais da construção, em conjunto com variáveis de acessibilidade, vizinhança e ambiente. Dentre alguns desses aspectos, podem-se citar: número de cômodos, idade da construção, índices socioeconômicos da região e distância do imóvel para estruturas urbanas (Ceh, Kilibarda, Lisec, & Bajat, 2018).

Tem-se verificado, nos últimos anos, transformações em processos do setor imobiliário, como o desenvolvimento de serviços *online* de avaliação automatizada do preço de imóveis, que utilizam algoritmos e métodos computacionais (Peter, Okagbue, Obasi & Akinola, 2020). Dentre as várias possibilidades de utilização de TICs para avaliação de imóveis, está a aplicação de Inteligência Artificial ou, mais especificamente, de modelos de Rede Neural Artificial (RNA) (Núñez Tabales, Caridad y Ocerin, & Rey Carmona, 2013). Consistindo em um método computacional projetado para realizar tarefas a partir do aprendizado por observação, a RNA gera modelos preditivos com custo computacional reduzido e precisão aprimorada (Peter et al., 2020). Assim, uma RNA pode utilizar grandes bases de dados com características de imóveis para realizar a predição de valores de imóveis que ainda não tenham sido avaliados.

No contexto das cidades inteligentes, sistemas de avaliação imobiliária, que incorporem automatizações como estas, permitem uma tomada de decisão consciente por parte de interessados, como investidores, avaliadores, agentes tributários, urbanistas e demais participantes do mercado imobiliário (Renigier-Bilozor, Bilozor & Wisniewski, 2017). Ainda, com a utilização de dispositivos móveis, que permitem a utilização de serviços de TIC de forma distribuída (Dehlinger & Dixon, 2011), o potencial de serviços imobiliários automatizados pode colaborar ainda mais para o portfolio de serviços de cidades inteligentes.

É neste contexto que este estudo propõe a criação de uma aplicação móvel que utiliza um modelo de aprendizado de máquina para avaliação do preço de apartamentos. O objetivo principal foi desenvolver uma ferramenta que avalia imóveis para auxiliar a tomada de decisão de qualquer interessado no setor imobiliário. Especificamente, um aplicativo móvel que emprega um modelo de Rede Neural de Regressão do tipo *Perceptron* Multicamadas treinada com uma base de dados gerada pelos autores, utilizando-se a linguagem Python.

Os resultados aqui apresentados podem ser úteis para profissionais e acadêmicos do ramo imobiliário, interessados em aplicações inovadoras para sistemas de avaliação de imóveis; e também para profissionais da área de TIC, pois apresenta uma aplicação de inteligência artificial no contexto de cidades inteligentes.

Este artigo está organizado da seguinte forma: a seção a seguir apresenta trabalhos relacionados. A Seção 3 descreve os materiais e métodos utilizados no estudo. A Seção 4,

como resultados, apresenta o aplicativo móvel e as métricas do modelo de RNA utilizado e, por fim, a Seção 5 apresenta as considerações finais do estudo.

2.Trabalhos relacionados

Vários trabalhos têm sido propostos na área de avaliação automática de imóveis. Para que esta importante atividade dentro do contexto de informatização das cidades (Arcuri, De Ruggiero, Salvo & Zinno, 2020) possa ser realizada, duas abordagens principais são utilizadas: utilizando-se dados do setor imobiliário, em que estima-se valores por meio da comparação entre atributos de imóveis, ou a abordagem que realiza a estimativa sem uso de tais dados.

Na situação em que não há dados suficientes para realizar as avaliações baseadas no mercado imobiliário, Arcuri, De Ruggiero, Salvo & Zinno (2020) propõem uma solução orientada ao custo, que calcula o preço para reconstruir o imóvel. Considerando tanto atributos intrínsecos do imóvel quanto fatores externos referentes à localização, a proposta dos autores combina Sistemas de Informação Geográfica com Modelos de Informação de Construção (*Ibid.*). Dessa forma, o preço estimado do imóvel é fundamentado pelo valor estimado de sua reconstrução, levando em conta custos estruturais, técnicos e financeiros combinados com fatores geográficos.

Por outro lado, em situações com grande quantidade de dados disponíveis, o problema de avaliar bens pode ser abordado com análises preditivas. Contudo, como dados imobiliários são tipicamente multidimensionais e não lineares, modelos convencionais de regressão são ineficientes para modelar funções de avaliação (Xu & Gade, 2017). Assim, não há solução trivial para predizer o valor de propriedades.

Para estimar preços de apartamentos, por exemplo, Núñez Tabales et al. (2013) indicam o modelo de Redes Neurais Artificiais (RNA). Sua proposta consiste em utilizar uma rede do tipo *Perceptron* Multicamadas, na qual destinam-se as características do imóvel à camada de entrada e o preço à camada de saída. Entre essas características, encontram-se variáveis referentes aos cômodos do apartamento, à estrutura externa do edifício e à localização.

O modelo, que foi estruturado com dados de transações imobiliárias efetivamente registradas em uma única cidade, apresentou melhores resultados que métodos clássicos de avaliação imobiliária. Ainda, dentre vantagens do modelo, destaca-se a capacidade das redes neurais de trabalhar com relações não lineares e multicolinearidade entre variáveis (Núñez Tabales et al., 2013).

Já o trabalho de Xu & Gade (2017) propõe o uso de Redes Neurais Profundas estruturadas, que têm capacidade de aprender em tempo real e se adaptar às tendências do mercado. Configuradas como redes neurais com mais de uma camada oculta, as redes profundas aprendem relações ocultas entre dados por meio de treinamento supervisionado e não supervisionado (*Ibid.*).

Para esse treinamento, são necessárias as etapas de coleta e pré-processamento, na qual dados considerados como anomalias são removidos do modelo. Em relação a arquitetura da rede profunda, Xu & Gade (2017) propõem o uso de redes estruturadas, em que as conexões entre neurônios são organizadas por afinidade de atributos do imóvel. Em comparação com redes totalmente conectadas, nas quais todos os neurônios se encontram conectados com os demais, as redes estruturadas se apresentaram como método de menor erro percentual.

A determinação do valor de imóveis também pode ser abordada por outros métodos computacionais, como: máquinas de vetores de suporte ou análise de regressão múltipla. Por

meio da observação de uma base de dados contendo valor de mercado e atributos, esses métodos podem ser utilizados para estimar preços de imóveis e determinar o grau de influência que cada atributo tem sobre o valor final (Akar & Yalpır, 2021).

Especificamente na avaliação de terrenos, destacam-se diversos atributos externos que impactam no preço de venda, como: índices socioeconômicos, condições ambientais e estrutura urbana local. No estudo proposto por Akar & Yalpır (2021), ambos os métodos identificaram a área do lote como o atributo de maior impacto no valor do terreno, contudo as máquinas de vetores de suporte apresentaram desempenho ligeiramente superior. Em relação a modelagem, os autores ainda destacam que, embora acrescentar atributos favoreça à eficiência do modelo, é importante determinar quais são os apropriados para a construção do modelo.

De forma a contribuir com uma aplicação prática de Redes Neurais Artificiais, este estudo criou um modelo semelhante ao apresentado em Núñez Tabales et al. (2013). Os autores utilizaram uma base de dados de registros de vendas de imóveis realizadas. Neste estudo, não se utiliza dados de vendas, mas sim de anúncios realizados por imobiliárias em plataformas web, o que facilita o acesso aos dados, já que são públicos e contém características relevantes dos imóveis publicados.

3. Materiais e métodos

O objetivo deste trabalho foi desenvolver uma ferramenta que avalia imóveis para auxiliar a tomada de decisão de qualquer interessado no setor imobiliário. Especificamente, um aplicativo móvel que emprega um modelo de Rede Neural de Regressão do tipo *Perceptron* Multicamadas treinada com uma base de dados gerada pelos autores, utilizando-se a linguagem Python.

O estudo deu-se em seis etapas: 1) criação da base de dados, para a qual utilizou-se *web scraping*; 2) tratamento da base de dados, em que se corrigiu dados incorretos; em seguida, 3) o pré-processamento dos dados, em que se removeram dados *Outliers* e ampliou-se o *Dataframe*; 4) preparação do treinamento da rede neural; 5) o treinamento da rede em si e, por fim, 6) o desenvolvimento do aplicativo móvel. Estas etapas são detalhadas nas subseções a seguir.

3.1 Base de dados

Para geração da base de dados, utilizou-se um *web scraper* que obtém informações relevantes de anúncios de apartamentos listados para venda no site Imovelweb https://www.imovelweb.com.br. A *web scraping* é uma técnica que consiste em extrair dados não estruturados de *sites web* e estruturá-los em forma de tabela. Estes podem ser armazenados em arquivos ou bancos de dados para futura análise (Sirisuriya, 2015).

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 (Sirisuriya, 2015).

O *web scraper*, desenvolvido em R, foi estruturado principalmente com os pacotes *httr*, *rvest* e *RPostgres*, 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 Portal Cidades@ do Instituto Brasileiro de Geografia e Estatística (IBGE) em

https://cidades.ibge.gov.br/brasil/panorama. A base de dados consolidada registra 383.354 anúncios de apartamentos, que, classificados por estado, ficam distribuídos na forma da Figura 1.

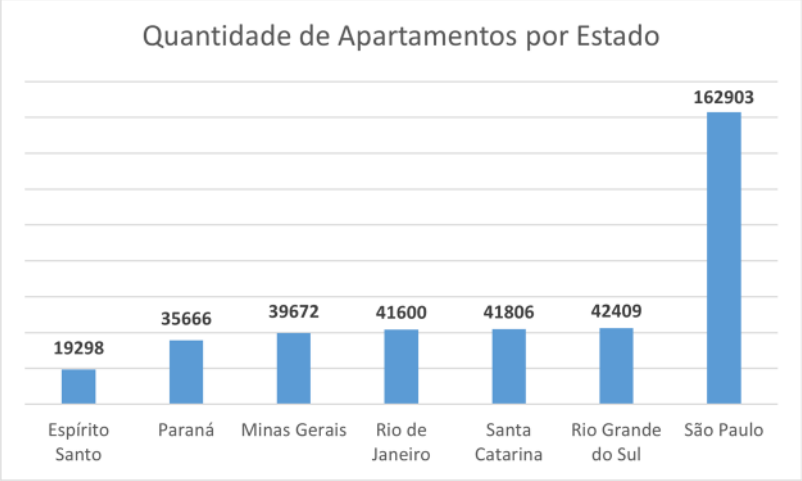

Figura 1. Apartamentos registrados por estado
Fonte: Os autores (2022)

Na base de dados, as características relevantes para a determinação do preço de imóveis foram armazenadas em tabela com 383.354 linhas e 19 colunas, que representam, respectivamente, imóveis e atributos, conforme Quadro 1.

Quadro 1 - 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

Fonte: Os Autores (2022)

3.2 Tratamento da base de dados

A etapa de tratamento, que preparou a base para o pré-processamento, fundamentou-se em um princípio de *Traditional Data Cleaning* citado por Tae, Roh, Oh, Kim, & Whang (2019), que consiste em checar se um registro está presente em um conjunto de valores considerados corretos. Caso um registo não esteja contido no conjunto de valores corretos, é necessária sua retificação com um dos valores desse conjunto (Tae et al., 2019).

Nesse estudo, o procedimento foi necessário para correção dos nomes de bairros, uma vez que havia 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 Portal Correios: Busca CEP da Empresa Brasileira de Correios e Telégrafos (Correios) em https://buscacepinter.correios.com.br. Dessa forma, para cada cidade, empregou-se uma lista de bairros na forma da Figura 2.

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

Figura 2. Algoritmo da lista de bairros de uma cidade
Fonte: Os Autores (2022)

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, seguindo as instruções em https://geopy.readthedocs.io/en/stable, empregou-se a ferramenta GeoPy, 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, a Figura 3 demonstra a utilização da GeoPy para a latitude -25,4064869 e longitude -49,2502621. Como resposta, a função reverse retorna um objeto do tipo Location contendo informações sobre a coordenada.

```
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))
```

Figura 3. Execução da função reverse do GeoPy
Fonte: Os Autores (2022)

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, identificaram-se 22.236 registros com nomes de bairros inconsistentes. Após a correção, com a 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).

3.3 Pré-processamento dos dados

A etapa de pré-processamento foi dividida em duas etapas: remoção de *Outliers* e ampliação do *Dataframe*. Para isso, empregou-se Python com implementação de funções das bibliotecas Pandas e Scikit-learn, conforme instruções em https://pandas.pydata.org e https://scikit-learn.org/stable, respectivamente.

3.3.1 Remoção de *Outliers*

Inicialmente, utilizou-se a biblioteca 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), citada por Matos, Netto, Lage, Segundo & Braga (2019).

$$IQR = Q_3 - Q_1 \quad (1)$$

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 Rousseeuw e Leroy (1987). Depois de obtido o IQR, calcularam-se os limites inferior e superior, em acordo com a Equação 2 (Matos *et al.*, 2019).

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

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 valor mínimo de R$ 0,00 e máximo de R$ 7.670.000,00.

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*, na Figura 4, pode-se perceber a alteração da distribuição dos valores de venda. No diagrama anterior a exclusão (à esquerda), percebem-se diversos *Outliers* que se distanciam da mediana da amostra, retratada pela linha vermelha.

Já no diagrama posterior à exclusão (à direita), verifica-se a redução *Outliers*, assim como diminuição da distância entre os valores de *Outliers* remanescentes para a mediana.

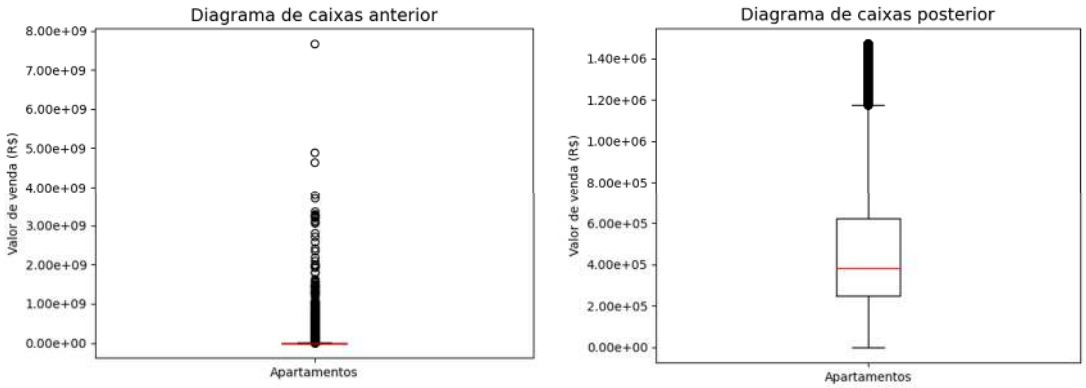

Figura 4. Diagrama de caixas para a variável valor de venda anterior e posterior a remoção de *outliers*

Fonte: Os autores (2022)

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 no Quadro 2.

Quadro 2 - 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

Fonte: Os Autores (2022)

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 (Ding & Fei, 2013).

Para aplicação da técnica, utilizou-se a função IsolationForest do *framework* Scikit-learn sobre o *Dataframe*. 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.

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*.

3.3.2 Ampliação do *Dataframe*

A etapa de ampliação do *Dataframe* consistiu em agregar informações aos registros da tabela, de forma a enriquecer os dados para gerar o modelo de predição. Para tal fim, realizou-se identificação de outros atributos do 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.* (2018), 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. Esta é uma informação normalmente apresentada de forma textual livre nos anúncios de imóveis. Assim, identificou-se a presença ou ausência de determinadas características no imóvel no texto extraído. Como regra de negócio, os atributos internos e externos de interesse se apresentam na forma do Quadro 3.

Quadro 3 - Atributos dos apartamentos

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

Fonte: Os Autores (2022)

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 Portal Cidades@ do IBGE em https://cidades.ibge.gov.br/brasil/panorama e do Sistema Ipeadata do Instituto de Pesquisa Econômica Aplicada (IPEA) em http://www.ipeadata.gov.br, conforme Quadro 4.

Para cada registro, inseriu-se na tabela o valor do índice referente à 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 a localização.

Primeiramente, definiram-se limites de quantidade mínima para manter cidades e bairros na análise. Para cidades, definiu-se esse limite em 250 registros, enquanto 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.

Quadro 4 - Índices socioeconômicos das cidades

Dados do IBGE
Densidade Demográfica (2010)
Salário médio mensal dos trabalhadores formais (2019)
População ocupada (2019)
Taxa de escolarização de 6 a 14 anos de idade (2010)
IDEB – anos iniciais do ensino fundamental (rede pública) (2019)
IDEB – anos finais do ensino fundamental (rede pública) (2019)
PIB per capita (2019)
Índice de desenvolvimento humano municipal (idhm) (2010)
Mortalidade infantil (2020)
Internações por diarreia (2016)
Esgotamento sanitário adequado (2010)
Arborização de vias públicas [2010]
Urbanização de vias públicas [2010]
Dados do IPEA
Taxa de homicidios [2019]
Taxa de suicidios [2019]
Taxa de acidentes de trânsito com óbito [2019]

Fonte: Os Autores (2022)

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 Quadro 5.

3.4 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 (Haykin, 2001).

Na Figura 5, 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 (Haykin, 2001).

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 (Haykin, 2001).

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 (Wang, 2003).

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.

Quadro 5 - Distribuição de dados pré-processados

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

Fonte: Os autores (2022)

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 (Salazar, Garland, Ochoa & Pyrcz, 2022).

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.

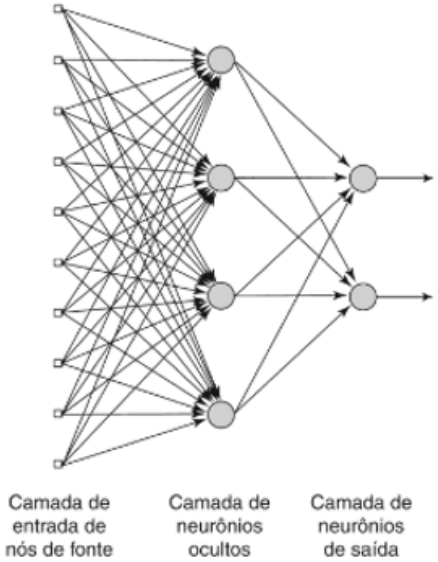

Figura 5. Estrutura de uma Rede Neural Multicamadas do tipo *feed-foward*
Fonte: Adaptado de Haykin (2001)

A aplicação dessas técnicas justifica-se pelo aumento da precisão de algoritmos do *framework* Scikit-learn em decorrência da normalização da distribuição de dados (Pedregosa *et al.*, 2011).

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) (Pedregosa *et al.*, 2011). Para isso, aplicou-se a Equação 3 sobre cada coluna do *Dataframe*.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3)$$

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 (Pedregosa et al., 2011).

3.5 Treinamento

Para a predição dos resultados, utilizou-se uma rede neural de regressão do tipo *Perceptron* multicamadas implementada pela classe MLPRegressor do *framework* Scikitlearn.

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 (Pedregosa *et al.*, 2011). Para a camada de entrada, destinaram-se as 33 variáveis independentes presentes no Quadro 5. 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*) (Pedregosa *et al.*, 2011).

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 Quadro 6.

Quadro 6 - Hiperparâmetros para o GridSearchCV

Parâmetros	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

Fonte: Os Autores (2022)

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. 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 no Quadro 7.

Quadro 7 - Hiperparâmetros ideais para o modelo

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

Fonte: Os Autores (2022)

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.

3.6 Desenvolvimento da Aplicação

A aplicação móvel tem como finalidade permitir que partes interessadas do ramo imobiliário 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 6.

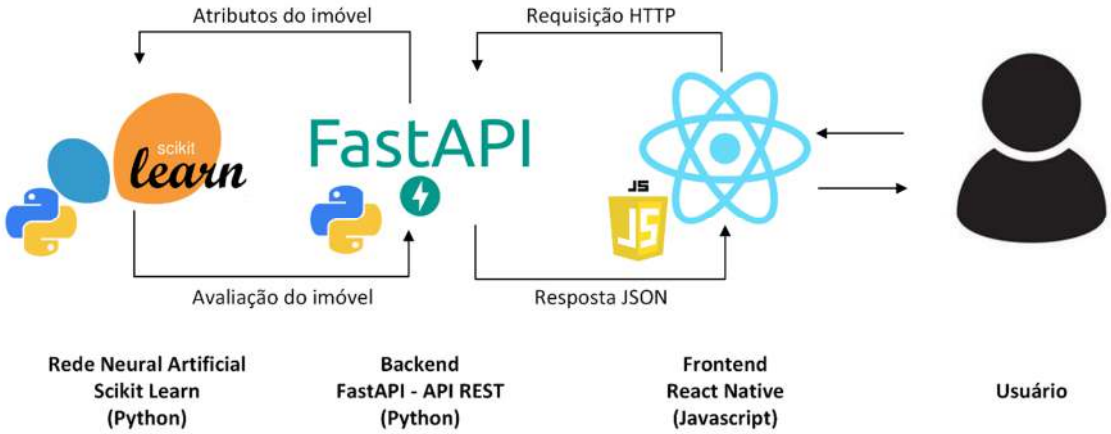

Figura 6. Arquitetura da Aplicação *Mobile*
Fonte: Os Autores (2022)

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.

4.Resultados e discussão

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

4.1 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 Núñez 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, Núñez 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.

4.2 Aplicativo Móvel

Os requisitos da aplicação dividem-se em funcionais e não funcionais. No Quadro 8, apresentam-se os requisitos funcionais, que especificam as funções que a aplicação executa durante a operação por um usuário. Por outro lado, no Quadro 9, mostram-se os requisitos não funcionais, que definem a forma com que a aplicação realizará suas funções.

Quadro 8 - 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.

Fonte: Os autores (2022)

Quadro 9 - 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.

Fonte: Os autores (2022)

Na Figura 7, 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.

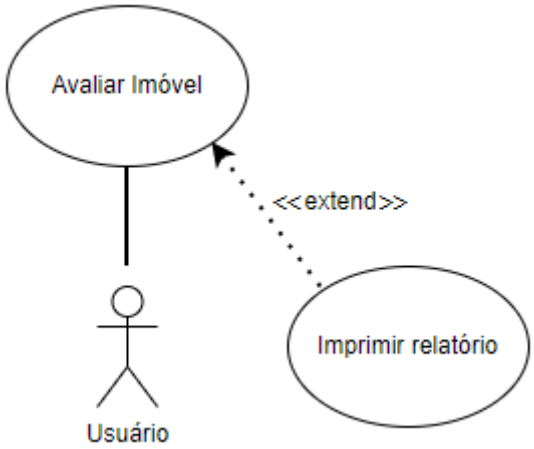

Figura 7. Diagrama de casos de uso
Fonte: Os autores (2022)

As telas do sistema referentes ao caso de uso avaliar imóvel, em conjunto com o relatório gerado, estão apresentadas na Figura 8. Por meio delas, o usuário informa localização, características e atributos do imóvel. Após, o sistema avalia as informações e apresenta a avaliação do apartamento.

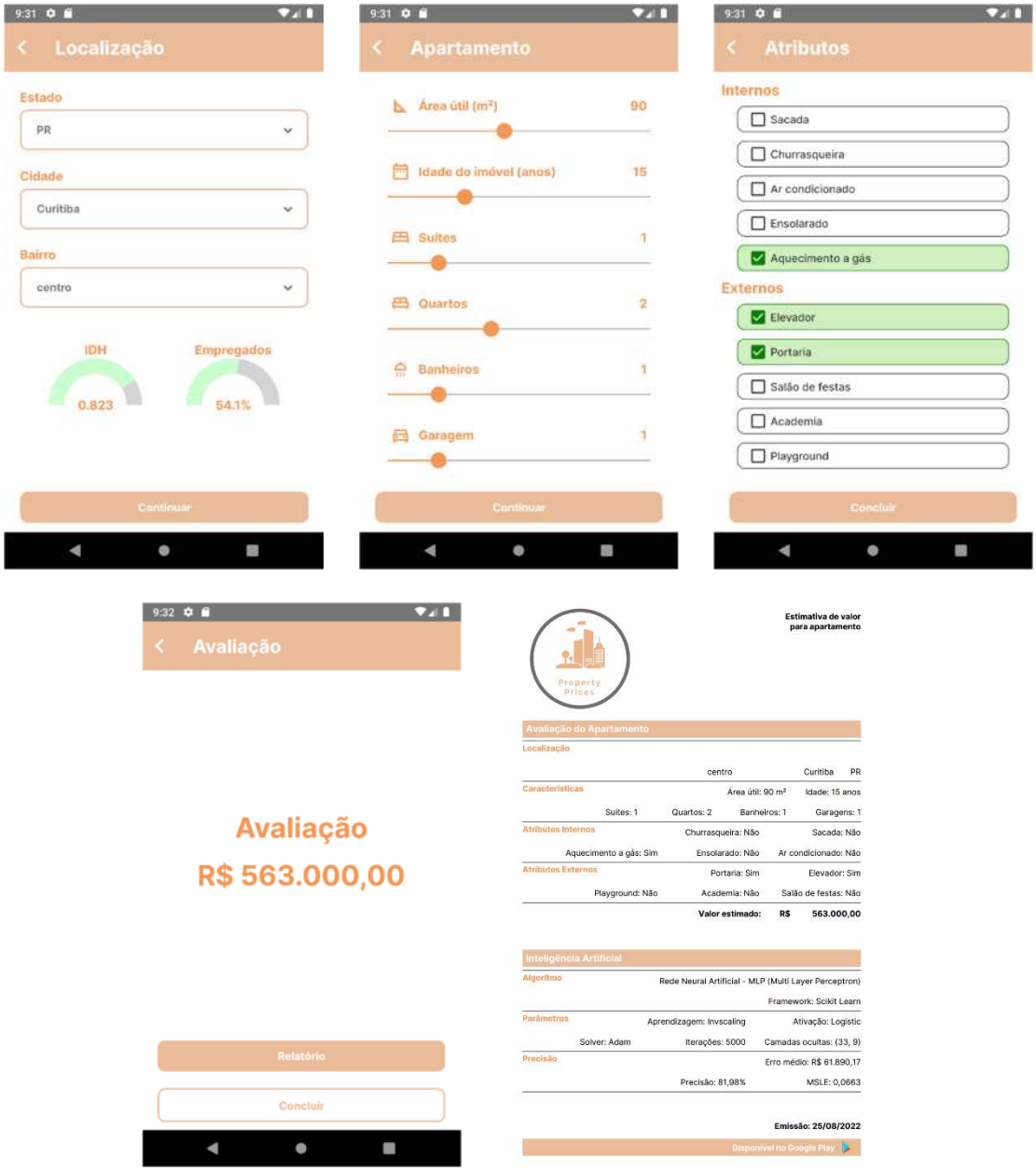

Figura 8. Telas da Aplicação *Mobile*
Fonte: Os autores (2022)

Na Figura 9, 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. Por fim, a Figura 10 apresenta 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 ao sistema até a avaliação do apartamento, que se utiliza da rede neural para predição do valor do imóvel.

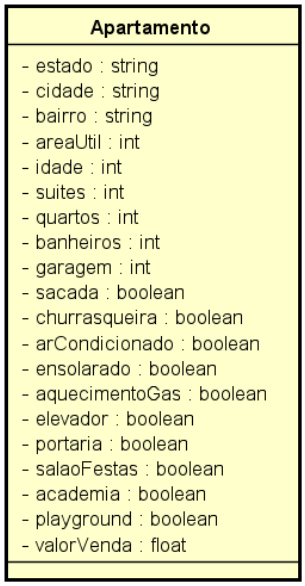

Figura 9. Diagrama de Classes
Fonte: Os autores (2022)

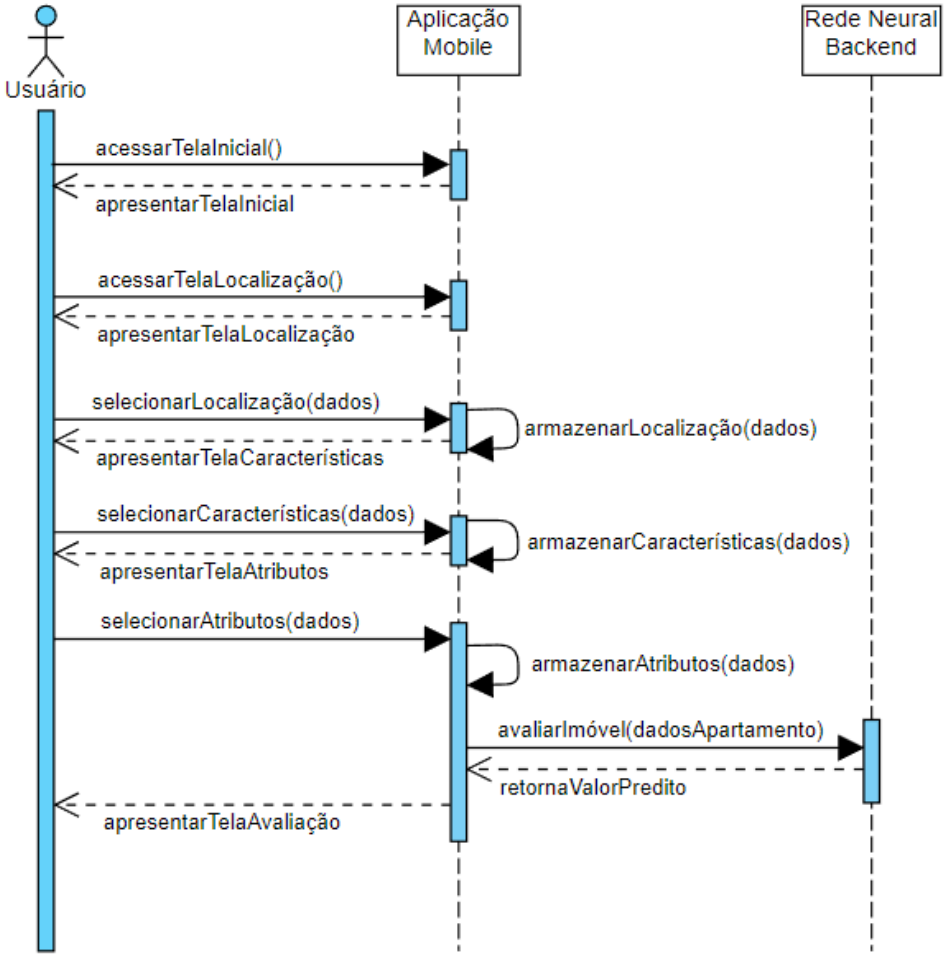

Figura 10. Diagrama de Sequência
Fonte: Os autores (2022)

5. Considerações finais

Este estudo 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.

Os resultados aqui apresentados contribuem para profissionais e acadêmicos do mercado imobiliário e da área de cidades inteligentes, interessados em aplicações com inteligência artificial. Descreveu-se em detalhes a criação da base dados, com a constituição das características dos imóveis, bem como o preparo e treinamento do modelo de RNA criado.

Na aplicação resultante, a RNA criada ficou localizada no *Backend*, desenvolvido com o framework FastApi em Python e executado em servidor. Já o *Frontend*, elaborado com o framework React Native em Javascript, constitui a interface gráfica da aplicação móvel ao usuário. 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.

É relevante mencionar que a aplicação aqui proposta é limitada à 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.

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

Akar, A. U., & Yalpır, S. (2021). Using svr and mra methods for real estate valuation in the smart cities. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W5-2021, 21-26. Recuperado de: https://doi.org/10.5194/isprs-archives-XLVI-4-W5-2021-21-2021

Arcuri, N., De Ruggiero, M., Salvo, F., & Zinno, R. (2020). Automated valuation methods through the cost approach in a BIM and GIS integration framework for smart city appraisals. *Sustainability*, 12(18), 7546. Recuperado de: https://doi.org/10.3390/su12187546

Ceh, M., Kilibarda, M., Lisec, A., & Bajat, B. (2018) Estimating the performance of random forest versus multiple regression for predicting prices of the apartments. *International Journal of Geo-Information*, 7(5), 168. Recuperado de: https://doi.org/10.3390/ijgi7050168

Cunha, M. A., Przeybilovicz, E., Macaya, J. F. M., & Burgos, F. (2016). *Smart cities: transformação digital de cidades*. São Paulo, SP: Programa Gestão Pública e Cidadania – PGPC

Dehlinger, J., & Dixon, J. (2011). Mobile application software engineering: Challenges and research directions. In *Proceedings of the second annual workshop on Software Engineering for mobile application development*, 27–30. Santa Monica, CA. Recuperado de: https://web.archive.org/web/20120611212356id_/http://www.mobileseworkshop.org:80/papers/MSEWorkshopProceedings.pdf#page=29

Ding, Z., & Fei, M. (2013). An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. In 3rd IFAC International Conference on Intelligent Control and Automation Science, 46(20), 12-17. Chengdu, China.

Haykin, S. (2001) Redes neurais: princípios e prática. Porto Alegre, RS: Bookman.

Matos, S. N., Netto, G. G., Lage, V. N., Segundo, A. K. R., & Braga, M. F. (2019) Tratamento de dados: uma abordagem prática para aprendizagem de atenuaçao de ruıdos e eliminaçao de outliers. Simpósio Brasileiro de Automação Inteligente, 2827-2834. Recuperado de: http://doi.org/10.17648/sbai-2019-111570

Núñez Tabales, J. M., Caridad y Ocerin, J. M., & Rey Carmona, F. J. (2013). Artificial neural networks for predicting real estate prices. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 15, 29-44. Recuperado de: https://upo.es/revistas/index.php/RevMetCuant/article/view/2218

Peter, N. J., Okagbue, H. I., Obasi, E. C., & Akinola, A. O. (2020). Review on the application of artificial neural networks in real estate valuation. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(3), 2918-2925. Recuperado de: https://doi.org/10.30534/ijatcse/2020/66932020

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python, *Journal of Machine Learning*, 12, 2825-2830. Recuperado de: https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf

Renigier-Bilozor, M., Bilozor, A., & Wisniewski, R. (2017) Rating engineering of real estate markets as the condition of urban areas assessment. *Land Use Policy*, 61, 511-525. Recuperado de: https://doi.org/10.1016/j.landusepol.2016.11.040

Rousseeuw, P. J., & Leroy, A. M. (1987). *Robust regression and outlier detection*. United States of America: John Wiley & Sons.

Salazar, J. J., Garland, L., Ochoa, J., & Pyrcz, M. J. (2022). Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy. Journal of Petroleum Science and Engineering, 209, 109885. Recuperado de: https://doi.org/10.1016/j.petrol.2021.109885

Sirisuriya, D. S. (2015). A comparative study on web scraping. In Proceedings of 8th International Research Conference KDU, 135-140. Recuperado de: http://ir.kdu.ac.lk/handle/345/1051

Tae, K. H., Roh, Y., Oh, Y. H., Kim, H., & Whang, S. E. (2019). Data cleaning for accurate, fair, and robust models: A big data-AI integration approach. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, 1-4. Amsterdam, NL. Recuperado de: https://doi.org/10.1145/3329486.3329493

Taffese, W. Z. (2007), Case-based reasoning and neural networks for real estate valuation. *Artifcial Intelligence and Applications*, p. 98-104. Recuperado de: https://www.researchgate.net/profile/Woubishet-Taffese/publication/221173735_Case-based_reasoning_and_neural_networks_for_real_estate_valuation

Wang, S. C. (2003). Artificial Neural Network. In S. C. Wang, *Interdisciplinary computing in java programming*, (1a. ed., Cap. 5, pp. 81-100)., Boston, MA: Springer

Xu, H., & Gade, A. (2017). Smart real estate assessments using structured deep neural networks. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, 1-7. San Francisco, CA. Recuperado de: https://doi.org/10.1109/UIC-ATC.2017.8397560