

# Análise e uso de Classificação e Regressão em conjunto de dados de imóveis para alugar no Brasil

Autor: Pedro Avellar Machado

Instituto de Ciências Matemáticas e de Computação - USP

15 de Maio de 2020

## 1 Introdução

O mercado imobiliário é um setor de suma importância ao redor do mundo, afinal, as pessoas necessitam de casas para residir. No Brasil, existem cerca de 13,3 milhões de imóveis alugados. Deste número surgem fontes de renda, oportunidades e possibilidade de conforto. Os valores de aluguel de imóveis são extremamente variados e são influenciados por diversos fatores, como localização, tamanho, serviços etc.

Através de um conjunto de dados de imóveis para alugar no Brasil, será feita uma análise em busca de informações e usando modelos de aprendizado de máquina, vamos tentar descobrir de que cidade é algum imóvel e também faremos um estimador de preço de aluguel.

Será apresentado o conjunto de dados utilizado e seus atributos, depois uma análise exploratória contando como o conjunto foi pré-processado, as abordagens serão explicadas detalhadamente e finalmente os resultados dos experimentos serão mostrados.

Este artigo faz parte de um trabalho avaliativo da disciplina de Aprendizado de Máquina oferecida no primeiro semestre de 2020 no ICMC-USP. O código utilizado para desenvolvimento deste artigo, foi feito no Google Colab e está disponibilizado na página <https://github.com/pedroavellar/HouseRentML>.

## 2 Trabalhos relacionados

Este conjunto de dados não é muito conhecido, portanto existem poucos trabalhos em cima deste. Algumas abordagens são encontradas na página do *dataset* dentro da seção *Kernels*, onde outros usuários expõe seus trabalhos. Os principais destes são focados na análise dos dados, usando gráficos e métricas. Este trabalho também faz análise exploratória dos dados, mas é voltado para o uso de modelos de aprendizado de máquina.

## 3 Material e Métodos

### 3.1 Apresentando o Dataset

O Dataset utilizado chama-se `brazilian_houses_to_rent`, de Rubens Junior. Este Dataset está disponível para visualização e download no Kaggle. (Link nas referências)

Existem dois arquivos no repositório, foi utilizada a versão mais atual, a `houses_to_rent_v2.csv`. Este conjunto contém dados de casas para alugar, obtidos em março de 2020, no Brasil, especificamente em cinco cidades: Belo Horizonte, Campinas, Porto Alegre, Rio de Janeiro e São Paulo. Originalmente, contém 10692 residências, com 13 atributos.

- city: Cidade onde o imóvel está localizado
  - String  $\bar{\{}$ Belo Horizonte, Campinas, Porto Alegre, Rio de Janeiro, São Paulo $\}$
- area: Área do imóvel
  - Real  $\bar{\{}$ (11, 46.300) $\}$
- rooms: Número de quartos
  - Inteiro  $\bar{\{}$ (1, 13) $\}$
- bathroom: Número de banheiros
  - Inteiro  $\bar{\{}$ (1, 10) $\}$
- parking spaces: Número de vagas
  - Inteiro  $\bar{\{}$ (0, 12) $\}$
- floor: Andar
  - Inteiro  $\bar{\{}$ (1, 301) $\}$  +  $\{-\}$
- animal: Permissão de animais
  - String(booleano)  $\bar{\{}$ accept (sic), not accept (sic) $\}$
- furniture: Mobiliado
  - String(booleano)  $\bar{\{}$ not furnished, furnished $\}$
- hoa (R\$): Valor de condomínio, em reais
  - Real  $\bar{\{}$ (0, 1.12M) $\}$
- rent amount (R\$): Valor do aluguel, em reais
  - Real  $\bar{\{}$ (450, 45000) $\}$
- property tax (R\$): IPTU, em reais
  - Real  $\bar{\{}$ (0, 314000) $\}$
- fire insurance (R\$): Seguro de incêndio, em reais
  - Real  $\bar{\{}$ (3, 677) $\}$
- total (R\$): Valor total, em reais
  - Real  $\bar{\{}$ (499, 1.12M) $\}$

### 3.2 Exploração e Pré-processamento

Durante esta etapa, investigamos o dataset e fizemos alterações para deixá-lo adequado para os próximos passos.

Foram encontradas instâncias duplicadas, que foram removidas, restando 10334 registros.

Como mostrado acima, no domínio dos atributos, algumas variáveis não estão em formatos ideais para serem utilizadas em modelos de aprendizado de máquina. Para resolver isso, atribuímos valores numéricos para as variáveis strings.

No caso de city: As cidades foram mapeadas de 1 a 5, seguindo ordem alfabética.

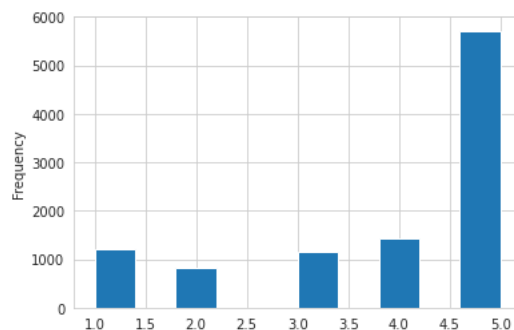
No caso de animal: accept  $\rightarrow$  1, not accept  $\rightarrow$  0

No caso de furniture: furnished  $\rightarrow$  1, not furnished  $\rightarrow$  0

Em seguida, cada um dos atributos foi analisado para encontrar situações incomum.

Usando matplotlib, gráficos foram gerados para auxiliar na visualização dos dados.

**city:**

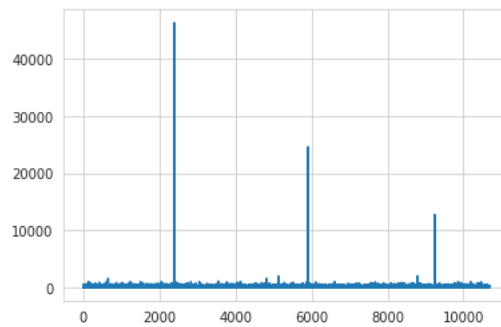


**Fig. 1.** Plot do número de registros por cidade. Lembrando que 1 é Belo Horizonte, 2 é Campinas, 3 é Porto Alegre, 4 é Rio de Janeiro e 5 é São Paulo

É possível visualizar que existem bem mais registros na cidade 5, São Paulo. Por enquanto não vamos fazer alterações nisso, mas esta característica será lembrada.

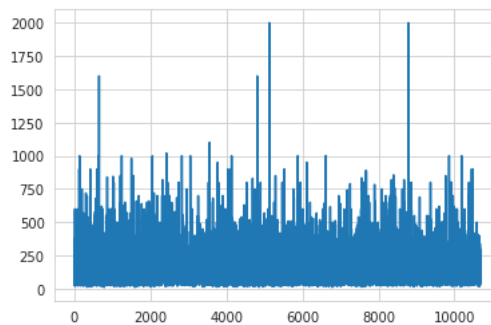
**area:**

Aqui foram encontrados nossos primeiros *outliers*. São esses picos gigantes no gráfico. *Outliers* são "pontos fora da curva", que apresentam grande distanciamento do restante dos dados, em casos como o deste artigo, podem



**Fig. 2.** Plot da área de cada registro

causar um viés negativo na análise, por isso estes registros foram removidos.



**Fig. 3.** Plot da área de cada registro após remoção dos *outliers*. Agora temos 10331 registros.

#### **rooms, bathroom, animal, furniture:**

Nestes atributos não foi encontrada necessidade de alteração.

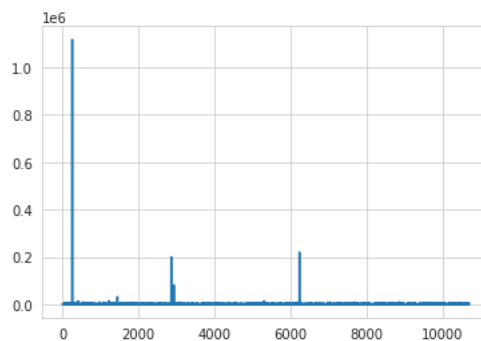
O valor mais frequentes de quartos e de banheiros é 1.

A proporção de registros que permitem animais é 78%. E a proporção de registros que são mobiliados é 24%.

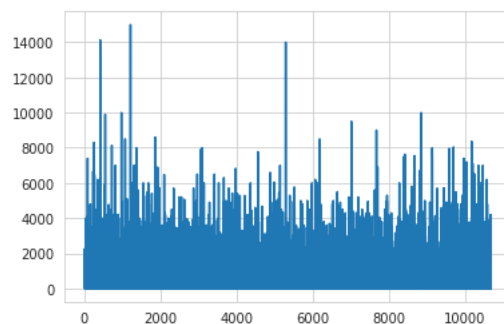
#### **hoa (condomínio):**

Um dos registros tem valor d condomínio superior a um milhão! Este condomínio deve ter tantas regalias... Ou este valor está incorreto. Também foram retiradas os registros com condomínio superior a R\$30.000.

#### **rent amount (aluguel):**



**Fig. 4.** Plot do valor de condomínio de cada registro



**Fig. 5.** Plot do valor de condomínio de cada registro após a remoção de valores muito elevados. Agora temos 10326 registros.

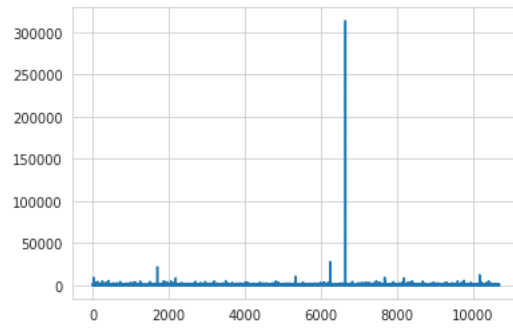
Neste item, existem dados que são bem maiores que a média, mas são poucos e parecem plausíveis. Nada foi alterado.

#### **property tax (IPTU):**

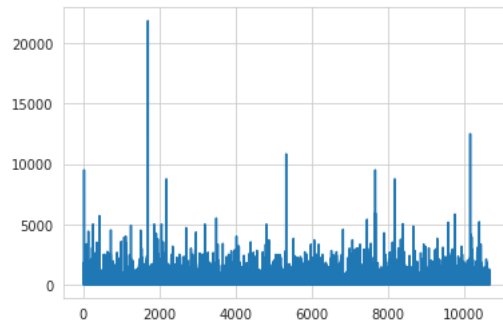
Mais *outliers*! Alguém está pagando 300000 de IPTU numa residência de 1 quarto... Removendo registros com IPTU superior a 25000.

#### **fire insurance, total:**

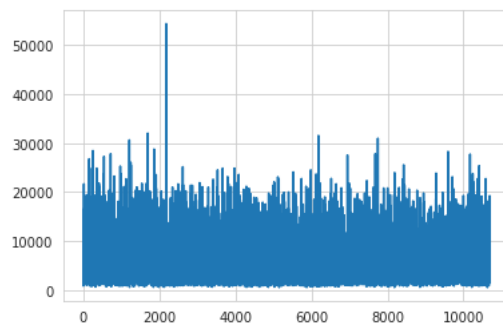
Estes atributos apresentam dados razoavelmente bem distribuídos após as alterações anteriores. Segue o gráfico final do total:



**Fig. 6.** Plot do valor de IPTU de cada registro



**Fig. 7.** Plot do valor de IPTU de cada registro atualizado. Agora temos 10324 registros.



**Fig. 8.** Plot do valor total em 10324 registros.

Finalizada esta limpeza do conjunto, foi feita uma análise da média dos atributos por cidade:

Pode-se observar algumas características das residências de cada cidade nessa amostra. Como o fato de registros de Belo Horizonte terem área maior, Porto

	area	rooms	animal	hoa (R\$)	total (R\$)
city					
1	174.768848	3.064623	0.744822	562.243579	4636.316487
2	124.684915	2.360097	0.809002	638.180049	3216.059611
3	105.464007	2.172593	0.856895	469.463140	2983.770165
4	107.622113	2.277117	0.800560	1057.217635	4668.101470
5	157.852792	2.594609	0.765272	1114.277963	6351.900228

**Fig. 9.** Média por cidade de alguns dos atributos.

Alegre permitir mais animais e São Paulo tendo valores maiores que as outras cidades. Obviamente que isso é válido especificamente para esse conjunto de dados, mas pode ser que reflita a realidade de alguma forma.

Finalmente, foi feita a substituição dos valores '-' em *floor*, que devem representar casas, pelo valor zero, permitindo transformar todos os valores desse atributo em inteiro.

### 3.3 Feature selection

Não foi utilizada técnica para seleção de atributos, já que não se mostrou necessário por haver poucos atributos no conjunto.

Mais pra frente, será comentado sobre os atributos utilizados nas duas abordagens escolhidas.

### 3.4 Abordagens

A ideia inicial do projeto era realizar um exercício de classificação, assim foi concebida a primeira abordagem: A partir de todos os outros atributos, classificar uma residência por cidade. Exemplo, dado número de quartos, área, preço do aluguel etc, classificar como sendo Belo Horizonte, Campinas etc.

Esta abordagem é factível, porém surgiram dúvidas quanto a capacidade de se obter um bom resultado, levando em conta são dados bem gerais e as cidades tem certo nível de similaridade. Além disso, esta tarefa apesar de interessante, não apresenta explicitamente uma aplicação que pode ser útil na vida real.

Então surgiu uma abordagem complementar. Desta vez um exercício de regressão: Através de outros atributos, estimar o preço do aluguel de um imóvel. Exemplo, sabendo qual a cidade, o número de quartos, mobília etc, estimar um valor de aluguel da residência.

Esta última poderia ser uma solução prática. Permitindo que um proprietário tenha uma estimativa do preço que pode alugar seu imóvel, através de seus atributos. Ou um locatário estimar se o aluguel é condizente.

### 3.5 Modelos e Implementação

Os modelos, tanto de classificação quanto de regressão, utilizados são da biblioteca *scikit-learn*.

Os modelos de classificação testados foram: SVC, MLPClassifier, Perceptron, GaussianNB, DecisionTreeClassifier, KNeighborsClassifier, RandomForestClassifier.

Os melhores modelos foram testados com diferentes hiperparâmetros para buscar melhor desempenho. Para avaliação desses modelos foi utilizada validação-cruzada *kfold*, com 10 *folds*. E as métricas coletadas foram acurácia, precisão, recall e F1-score.

Os modelos de regressão testados foram: SVR, MLPRegressor, PassiveAggressiveRegressor, LinearRegression, LogisticRegression, DecisionTreeRegressor, KNeighborsRegressor, RandomForestRegressor, GradientBoostingRegressor.

Para avaliação desses modelos o conjunto foi dividido em treino/teste e a métrica usada foi o erro quadrático médio (MSE).

## 4 Experimentos

Nesta seção as técnicas e resultados serão discutidos mais a fundo. Por existirem duas abordagens, analisaremos uma de cada vez.

### 4.1 Classificação

Relembrando, o objetivo aqui é classificar um registro em uma das cinco cidades do conjunto de dados. Para isso, vamos dividir o *dataframe*, a coluna das cidades (city) terá nossas classes, será nosso alvo. As outras colunas serão nossos dados utilizados pelo modelo pra fazer a predição, chamaremos de *features*. É feita uma normalização nas *features*, isso é feito para obtermos uma escala comum para que os modelos tenham uma convergência mais eficiente.

### 4.2 Resultados

Executando todos os modelos de classificação citados anteriormente, com hiperparâmetros padrões, foi obtido:

Os modelos que apresentaram melhor resultado foram o MLP e Random Forest. Portanto, vamos trabalhar com eles para obter o melhor modelo.

Com o MLP, obtivemos melhor resultado geral com taxa de aprendizado = 0.01. Fazendo uma análise do número de camadas e neurônios, foram testadas



	accuracy	precision	recall	f1
Perceptron	50,07%	34,18%	32,99%	30,45%
Naive Bayes	35,06%	32,41%	32,12%	26,32%
Decision Tree	61,01%	50,05%	50,41%	50,17%
Random Forest	72,24%	69,55%	55,75%	60,53%
KNN	59,31%	46,29%	43,50%	44,56%
SVC	59,89%	47,66%	29,64%	29,61%
MLP	76,67%	71,85%	63,82%	66,55%

as seguintes configurações: (50,), (50,5), (100,10), (100,20), (200,10), (200,20). O resultado foi muito parecido em todas elas, o melhor foi o (100,20).

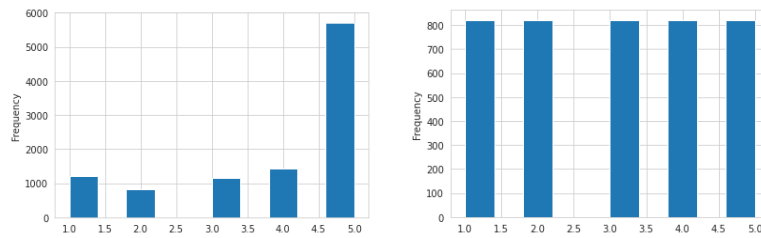
Com o Random Forest, as configurações testadas foram: [80 est, entropy], [100 est, entropy], [120 est, entropy], [80 est], [100 est], [120 est]. Novamente resultados bem próximos. O melhor foi [120 est, entropy].

	accuracy	precision	recall	f1
MLP (100,20)	78,99%	74,27%	69,30%	71,07%
Random Forest [120 est, entropy]	72,38%	69,95%	55,81%	60,70%

Portanto, o melhor modelo para esta tarefa foi o MLP, que sabemos que é um bom classificador para múltiplas classes. O resultado foi surpreendente, quase 80% de acurácia, basicamente o modelo classifica corretamente entre as cinco cidades, 4 a cada 5 vezes!

Porém temos uma questão pra responder... Se você tem uma boa memória lembra que existia um desbalanceamento nos dados, mais precisamente, existe muito mais instâncias de uma classe (São Paulo ou 5) do que das outras neste conjunto. Existe uma chance de que os modelos tenham aprendido isto, ficando enviesados. Pense, caso você queira classificar uma residência mas não tem ideia de como fazer, chutando São Paulo você tem uma boa probabilidade de acertar.

Deixaremos nosso conjunto equilibrado fazendo um balanceamento. O mais simples, vamos pegar uma amostra com o tamanho da menor frequência de uma classe, para cada classe, desta forma todas as amostras terão o mesmo tamanho.



**Fig. 10.** Plot da quantidade de registros por cidade antes e depois do balanceamento.

O impacto negativo disso é que perdemos informação e agora temos menos dados para treinar nossos modelos. Os testes serão refeitos para ver se existe alguma variação no resultado.

BALANCEADO	accuracy	precision	recall	f1
MLP (100,20)	70,83%	71,25%	70,83%	70,80%
Random Forest [120 est, entropy]	61,51%	61,82%	61,51%	61,52%

Como foi sugerido, o balanceamento alterou os resultados dos modelos, diminuindo a acurácia, mas não o suficiente para invalidar o modelo. Pode ser percebido também que as quatro métricas agora apresentam valores bem próximos.

### 4.3 Regressão

Relembrando, o objetivo aqui é estimar o valor de um aluguel baseado nos outros atributos da residência. Para isso também vamos dividir nosso *dataset*, porém dessa vez da seguinte forma: a coluna do valor do aluguel (rent value (R\$)) será nosso alvo, já as outras colunas serão nossas *features*, com exceção da coluna do valor total, já que ela é diretamente ligada ao valor do aluguel e seria desnecessário estimar algo que poderia ser calculado de forma simples. O mesmo processo de normalização da abordagem anterior é realizado.

### 4.4 Resultados


Executando todos os modelos de regressão citados anteriormente, com hiperparâmetros padrões, foi obtido:

Neste caso, quanto menos o MSE melhor, então o melhor modelo foi o GradientBoosting. Vamos brincar com os parâmetros pra ver se conseguimos aprimorá-lo.

Depois de muitas execuções, os melhores hiperparâmetros foram [learning\_rate=0.18, n\_estimators=120], apresentando um MSE próximo de 98000.

	MSE		MSE	
Random Forest	128267		158912	Linear
KNN	824970		151696	Decision Tree
SVR	10313908		104466	GradientBooster
PassiveAgressive	171459		248917	MLP

Este número parece assustar, vamos mostrar as previsões ao lado dos valores reais pra ver como o regressor se sai:



Predict	Value
4085	4000
1147	1090
15017	15000
3781	3800
4327	4350
15041	15000
812	700
1026	1000
1898	1900
1200	1200
706	750
1208	1584
2584	2500
2852	2900
677	550
863	900
1257	1300
1246	1300
992	1000
7667	7609
4434	4200
3743	3870
677	800
2872	2896
2097	2000
14898	15000
1228	1250
2025	2060
5426	5556
966	1000

**Fig. 11.** Os primeiros 30 valores preditos e os valores reais do conjunto de dados.

Agora esses números dão uma aliviada. A estimativa está muito próxima da realidade e acerta praticamente em cheio algumas instâncias. Considerando a limitação na amostra de dados e sem atributos importantes na definição do valor, como localização, estado do imóvel etc, podemos dizer que temos um bom estimador, que ainda pode ser melhorado.

## 5 Conclusão

Vimos neste artigo a importância e como fazer uma análise exploratória para identificar inconsistências, vieses e relações escondidas. O pré-processamento é vital para obter um desempenho bom e confiável em um modelo de aprendizado de máquina. A escolha de um modelo e hiperparâmetros bons depende do problema a ser trabalhado e exige uma tarefa exaustiva de busca. Na primeira abordagem, o Multi-Layer Perceptron obteve vantagem para classificar corretamente, já na segunda, o Gradient Boosting, um ensemble, se saiu melhor estimando os valores bem próximos dos reais.

## References

1. Conjunto de dados de casas para alugar ,  
<https://www.kaggle.com/rubenssjr/brasilian-houses-to-rent>
2. Notebook utilizado para implementação ,  
<https://github.com/pedroavellar/HouseRentML/blob/master/ArtigoAMouseRent.ipynb>
3. Scikit-learn ,  
<https://scikit-learn.org/stable/index.html>