Udacity - Programa Nanodegree Engenheiro de Machine Learning

Relatório do projeto de conclusão de machine learning - Capstone

Robson Azevedo Rung
07/08/2019

I - Definição

Visão geral do projeto

Este projeto busca desenvolver uma solução baseada em técnicas de machine learning para prever preços de imóveis.

Imóveis são bens de grande relevância para qualquer sociedade, e têm seus preços formados não de forma única, mas a partir das características que possuem. [1]

Assim sendo, os consumidores potenciais têm a percepção de diferenciar as diversas possibilidades de características em função do que é tido como prioritário. Desta maneira, um determinado consumidor pode escolher seu "pacote" de características disponíveis para cada bem ou serviço em função da percepção de utilidade.

Muitos estudos buscam determinar os atributos intrínsecos e extrínsecos pertencentes a cada residência, a fim de verificar quais deles apresentam maior representatividade para a composição dos instrumentos de demanda e oferta, utilizando-se modelos de preços hedônicos, por meio dos quais é possível analisar a importância relativa de cada atributo em função dos diferentes perfis sociodemográficos. [2]

Uma das atividades de empresas e profissionais do ramo imobiliário é a analisar o valor de imóveis, com objetivo de vendê-los em tempo razoável e maximizar seus lucros.

O conjunto de dados deste trabalho será aquele disponibilizado na competição intitulada *House Prices: Advanced Regression Techniques* da plataforma *Kaggle*. [3]

Constam dos dados diversos atributos, por exemplo:

- Tipo da propriedade;
- Tamanho do lote;
- Recursos públicos (eletricidade, gás, água e saneamento);
- Condições de acesso;
- Acabamento (materiais);
- Estado de conservação;
- Tipo da fundação;
- Tipo de aquecimento;
- Sistema elétrico;
- Características da garagem;
- Idade do imóvel;
- etc.

Os dados são fornecidos em dois grupos, um de treinamento e outro de teste (esse não contém o preço de venda, e deve ser usada para envio à competição).

Declaração do problema

Entender quais características de um imóvel são as mais relevantes e como impactam os preços é tarefa complexa por si só. Além disso, tais fatores podem mudar ao longo do tempo e variam entre regiões geográficas. Podem, ainda, ser afetadas por diferenças culturais e climáticas.

Inicialmente serão usadas técnicas de análise de dados para buscar obter uma compreensão dos atributos existentes na base de dados, considerando seus tipos de dados e suas possíveis correlações.

Como auxílio à análise, estatísticas como média, mediana, desvio padrão, valores máximos e mínimos serão calculadas.

Os atributos que precisarem de tratamento especial, como a aplicação de normalização ou one-hot enconde serão devidamente tratados.

Aplicar-se-á a técnica de seleção de atributos (f*eature selection*) conhecida como PCA (*Principal Component Analysis*), de modo a evitar-se o problema da dimensionalidade (*curse of dimensionality*).

Será conduzida uma uma busca e correção de outliers e dados faltantes, de modo a se evitar distorções nos modelos durante o treinamento.

Após as análises e ajustes relatados, será realizado o treinamento dos modelos. Para o caso da aprendizagem supervisionada, será conduzida uma otimização dos hiperparâmetros usando a técnica de *grid search*, validação com *cross-validation* e regularizção (*regularization*).

Além disso, a mesma tarefa será realizada usando técnicas de *deep learning*. Será criada, treinada e otimizada uma rede neural com os dados fornecidos.

For fim, o coeficiente de determinação será calculado para avaliar a performance dos modelos.

Métricas

Serão usadas duas métricas para análise dos modelos: coeficiente de determinação (*R2 Score*), que é uma métrica bastante usada para análise de regressões e *Root Mean Squared Logarithmic Error – RMSLE*.

O valor de R2 indica o percentual de correlação quadrática entre os valores previstos e reais. Quando o resultado é igual a 0, a regressão se equivale a um modelo que sempre tem como resultado a média amostral (dados de treinamento). Por outro lado, quando R2 é igual a 1, o modelo foi capaz de prever com precisão os valores da variável alvo.

É possível que R2 tenha como resultado um valor negativo, o que significa que o modelo de regressão é pior do que um modelo que sempre prevê a média.

A competição no *Kaggle* utiliza, para comparar as diversas submissões, a métrica *RMSLE*. Essa métrica calcula a raiz quadrada da média do quadrado das diferenças entre o valor original e previsto. O uso de *logs* tem como objetivo evitar que erros nos valores previstos de imóveis caros afete mais o resultado do que erros em imóveis baratos. Essa métrica também será calculada neste projeto.

Além disso, com o objetivo de comparação entre as soluções propostas e uma inicial, do tipo ingênua (*naive*), serão calculadas as métricas acima para dois conjuntos de dados, um que tenha como preços estimados exatamente a média dos preços reais, e outro com o valor da mediana.

II - Análise

Exploração dos Dados

Descrição resumida dos dados (transcrição do original, sem tradução)

```
MSSubClass:         Identifies the type of dwelling involved in the sale.
MSZoning:           Identifies the general zoning classification of the sale.
LotFrontage:        Linear feet of street connected to property
LotArea:            Lot size in square feet
Street:             Type of road access to property
Alley:              Type of alley access to property
LotShape:           General shape of property
LandContour:        Flatness of the property
Utilities:          Type of utilities available
LotConfig:          Lot configuration
LandSlope:          Slope of property
Neighborhood:       Physical locations within Ames city limits
Condition1:         Proximity to various conditions
Condition2:         Proximity to various conditions (if more than one is present)
BldgType:           Type of dwelling
HouseStyle:         Style of dwelling
OverallQual:        Rates the overall material and finish of the house
OverallCond:        Rates the overall condition of the house
YearBuilt:          Original construction date
YearRemodAdd:       Remodel date (same as construction date if no remodeling or additions)
RoofStyle:          Type of roof
RoofMatl:           Roof material
Exterior1st:        Exterior covering on house
Exterior2nd:        Exterior covering on house (if more than one material)
MasVnrType:         Masonry veneer type
MasVnrArea:         Masonry veneer area in square feet
ExterQual:          Evaluates the quality of the material on the exterior
ExterCond:          Evaluates the present condition of the material on the exterior
Foundation:         Type of foundation
BsmtQual:           Evaluates the height of the basement
BsmtCond:           Evaluates the general condition of the basement
BsmtExposure:       Refers to walkout or garden level walls
BsmtFinType1:       Rating of basement finished area
BsmtFinSF1:         Type 1 finished square feet
BsmtFinType2:       Rating of basement finished area (if multiple types)
BsmtFinSF2:         Type 2 finished square feet
BsmtUnfSF:          Unfinished square feet of basement area
TotalBsmtSF:        Total square feet of basement area
Heating:            Type of heating
HeatingQC:          Heating quality and condition
CentralAir:         Central air conditioning
Electrical:         Electrical system
1stFlrSF:           First Floor square feet
2ndFlrSF:           Second floor square feet
LowQualFinSF:       Low quality finished square feet (all floors)
GrLivArea:          Above grade (ground) living area square feet
BsmtFullBath:       Basement full bathrooms
BsmtHalfBath:       Basement half bathrooms
FullBath:           Full bathrooms above grade
HalfBath:           Half baths above grade
Bedroom:            Bedrooms above grade (does NOT include basement bedrooms)
Kitchen:            Kitchens above grade
KitchenQual:        Kitchen quality
TotRmsAbvGrd:       Total rooms above grade (does not include bathrooms)
Functional:         Home functionality (Assume typical unless deductions are warranted)
Fireplaces:         Number of fireplaces
FireplaceQu:        Fireplace quality
```

```
GarageType:         Garage location
GarageYrBlt:        Year garage was built
GarageFinish:       Interior finish of the garage
GarageCars:         Size of garage in car capacity
GarageArea:         Size of garage in square feet
GarageQual:         Garage quality
GarageCond:         Garage condition
PavedDrive:         Paved driveway
WoodDeckSF:         Wood deck area in square feet
OpenPorchSF:        Open porch area in square feet
EnclosedPorch:      Enclosed porch area in square feet
3SsnPorch:          Three season porch area in square feet
ScreenPorch:        Screen porch area in square feet
PoolArea:           Pool area in square feet
PoolQC:             Pool quality
Fence:              Fence quality
MiscFeature:        Miscellaneous feature not covered in other categories
MiscVal:            $Value of miscellaneous feature
MoSold:             Month Sold (MM)
YrSold:             Year Sold (YYYY)
SaleType:           Type of sale
SaleCondition:      Condition of sale
```

A base de dados possui 1460 linhas e 81 colunas, sendo uma delas o preço de venda.

Dos 80 atributos, 43 são categóricos e 37 são numéricos.

Atributos numéricos

Id	BsmtFinSF2	HalfBath	EnclosedPorch
MSSubClass	BsmtUnfSF	BedroomAbvGr	3SsnPorch
LotFrontage	TotalBsmtSF	KitchenAbvGr	ScreenPorch
LotArea	1stFlrSF	TotRmsAbvGrd	PoolArea
OverallQual	2ndFlrSF	Fireplaces	MiscVal
OverallCond	LowQualFinSF	GarageYrBlt	MoSold
YearBuilt	GrLivArea	GarageCars	YrSold
YearRemodAdd	BsmtFullBath	GarageArea	SalePrice
MasVnrArea	BsmtHalfBath	WoodDeckSF	
BsmtFinSF1	FullBath	OpenPorchSF	

Atributos categóricos

MSZoning	BldgType	BsmtCond	GarageType
Street	HouseStyle	BsmtExposure	GarageFinish
Alley	RoofStyle	BsmtFinType1	GarageQual
LotShape	RoofMatl	BsmtFinType2	GarageCond
LandContour	Exterior1st	Heating	PavedDrive
Utilities	Exterior2nd	HeatingQC	PoolQC
LotConfig	MasVnrType	CentralAir	Fence
LandSlope	ExterQual	Electrical	MiscFeature
Neighborhood	ExterCond	KitchenQual	SaleType
Condition1	Foundation	Functional	SaleCondition
Condition2	BsmtQual	FireplaceQu	

Estatísticas sobre o preço de venda

```
Valor mínimo do conjunto de treinamento ------>  $34,900.00
Valor máximo do conjunto de treinamento ------> $755,000.00
Valor da média do conjunto de treinamento ----> $180,921.20
Valor da mediana do conjunto de treinamento --> $163,000.00
Valor do desvio padrão ----------------------->  $79,442.50
```

Dados nulos

Listagem de atributos com valores nulos e a quantidade de registros nessas condições:

PoolQC	1453	GarageCond	81
MiscFeature	1406	BsmtFinType2	38
Alley	1369	BsmtExposure	38
Fence	1179	BsmtFinType1	37
FireplaceQu	690	BsmtCond	37
LotFrontage	259	BsmtQual	37
GarageYrBlt	81	MasVnrArea	8
GarageType	81	MasVnrType	8
GarageFinish	81	Electrical	1
GarageQual	81		

Correlação

Atributos que apresentam grau de correlação acima de 50% com o preço de venda:

```
Correlação de OverallQual  com SalePrice: 0.791
Correlação de YearBuilt    com SalePrice: 0.523
Correlação de YearRemodAdd com SalePrice: 0.507
Correlação de TotalBsmtSF  com SalePrice: 0.614
Correlação de 1stFlrSF     com SalePrice: 0.606
Correlação de GrLivArea    com SalePrice: 0.709
Correlação de FullBath     com SalePrice: 0.561
Correlação de TotRmsAbvGrd com SalePrice: 0.534
Correlação de GarageCars   com SalePrice: 0.640
Correlação de GarageArea   com SalePrice: 0.623
```

Outliers

Dentre os atributos selecionados acima (correlação acima de 50% com o preço de venda), existem 87 *outliers*.

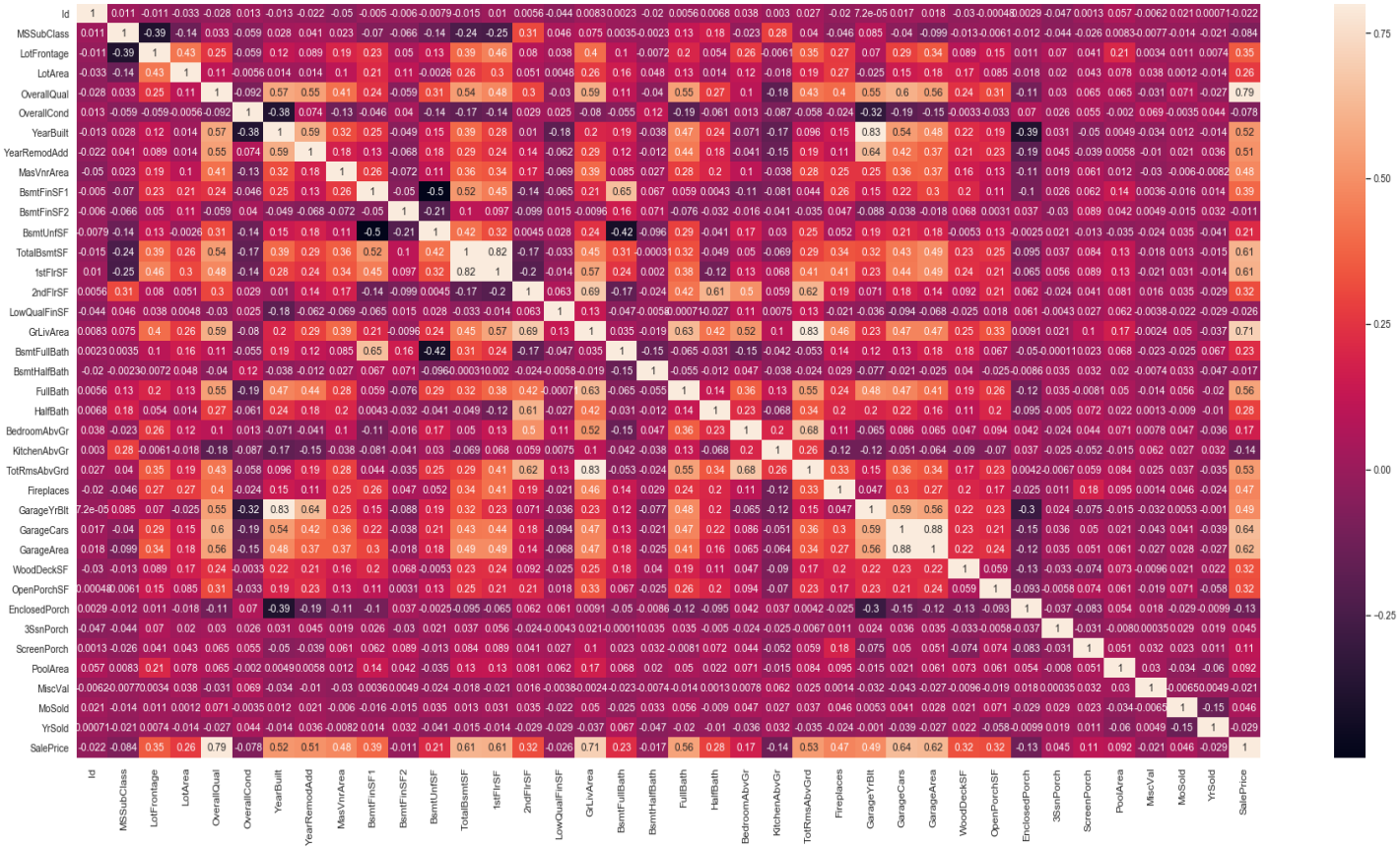
Para definir se um valor deve ser considerado como *outlier*, foi usado como critério o *z-score*, o qual corresponde ao número de desvios padrões de distância da média que um determinado valor está. O limiar usado foi o valor 3.

Visualização exploratória

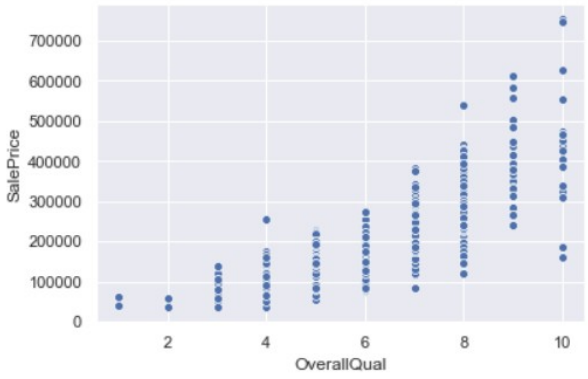
Distribição do preço de venda

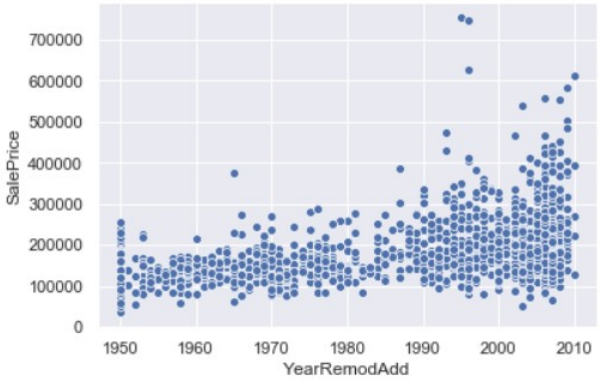
Percebe-se que os preços seguem uma distribuição normal com assimetria à direta, o que demanda ajustes.

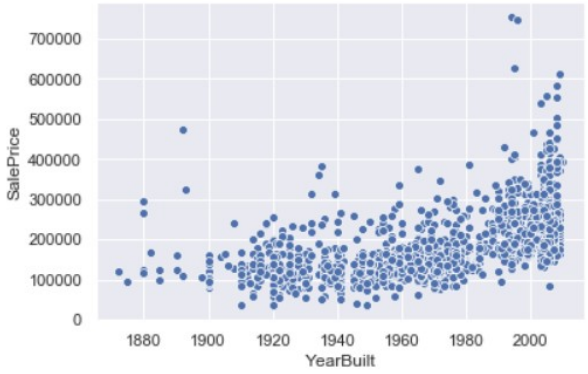
Matriz de correlação

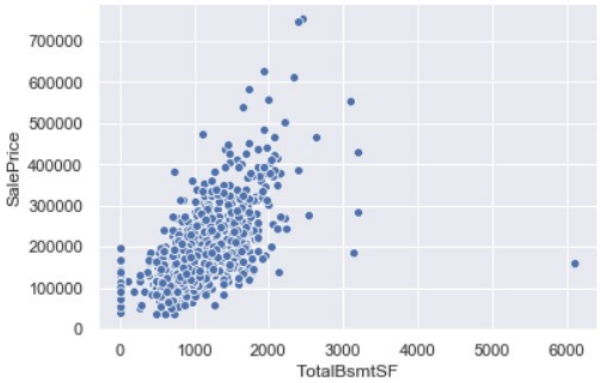


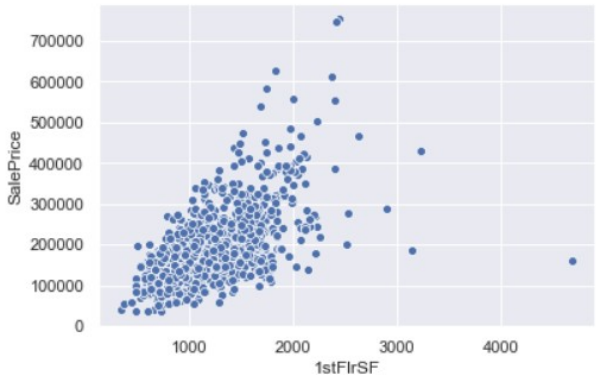
Visualização da correlação dos principais atributos com o preço de venda

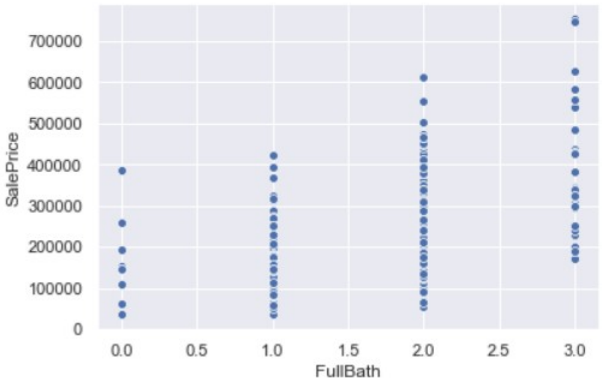


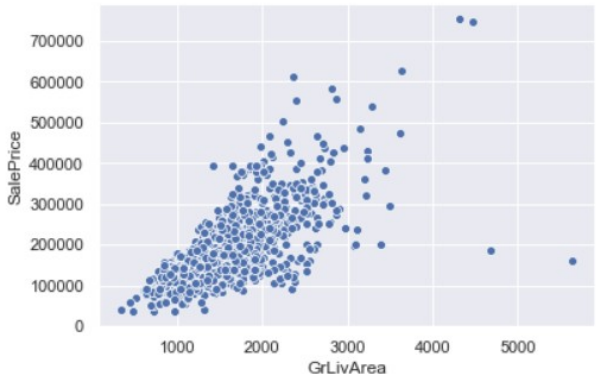


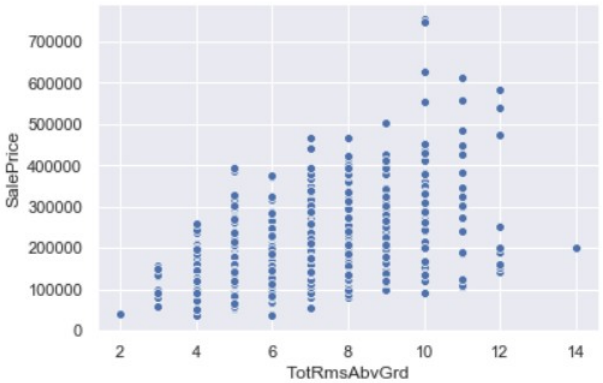


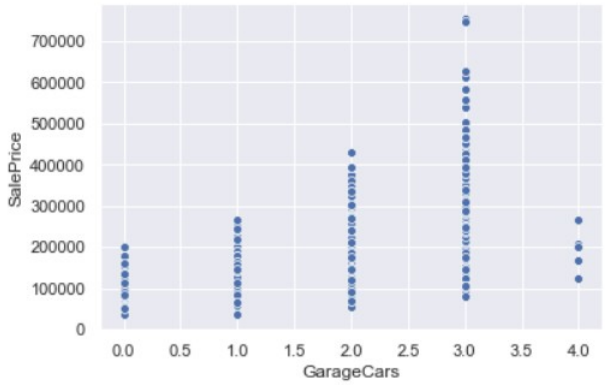


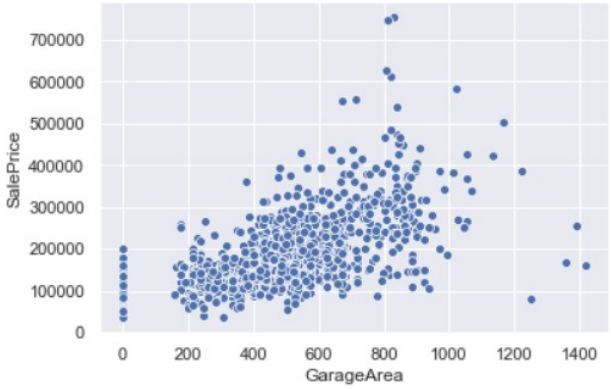




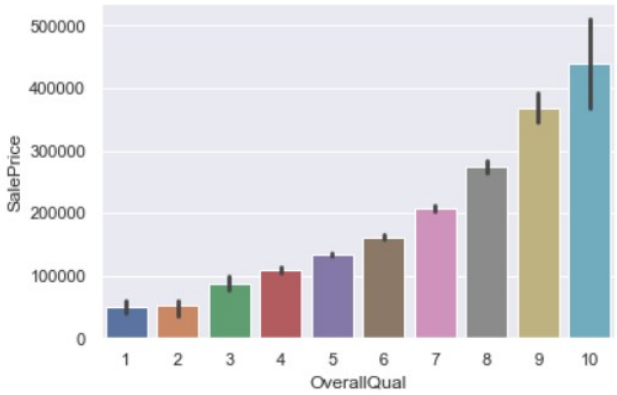


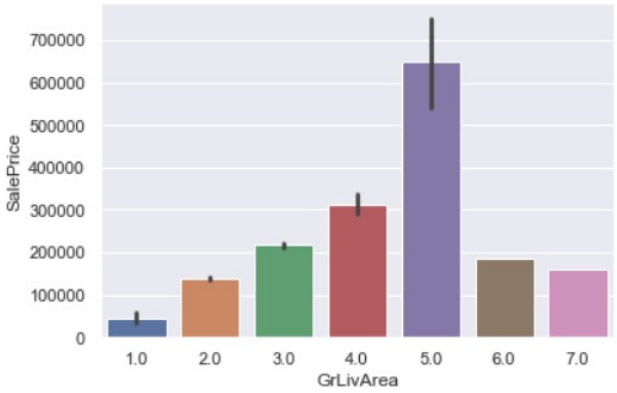


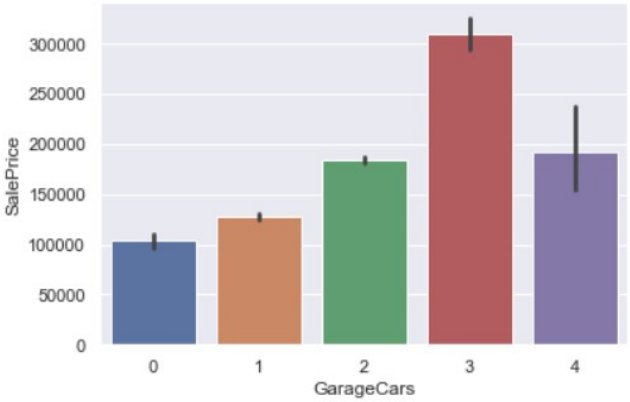




Visualização em barras dos 3 atributos com maior grau de correlação com o preço de venda







Fica claro que a qualidade geral do imóvel tem uma influência direta e linear sobre o preço. Além disso, imóveis com salas de estar com área em torno de 5.000 pés quadrados e 3 vagas de garagem são os mais valorizados.

Algoritmos e técnicas

A estimativa dos valores de venda de um imóvel é um problema estatístico de regressão. Uma regressão busca encontrar valores que não estão disponíveis inicialmente. Para isso, baseia-se em dados existentes e tenta aprender as relações entre as variáveis.

Em *machine learning*, o aprendizado supervisionado (*supervised learning*) consiste de técnicas que buscam encontrar a ligação entre atributos de entrada de várias observações e um valor, também da observação, que depende desses atributos.

Matematicamente, podemos dizer que os atributos das observações formam um matriz X, com dimensões m x n (sendo m a quantidade de observações e n a quantidade de atributos), e os valores dependentes formam um vetor Y, com dimensão m.

Existem basicamente dois tipos de algoritmos preditivos: classificação e regressão. O primeiro busca classificar os dados de entrada em grupos. O segundo tem como objetivo encontrar um valor para os dados de entrada. Como exemplo, classificar e-mails como *spam* ou não é uma tarefa de classificação. Por outro lado, conforme já comentado anteriormente, prever preços de imóveis é um problema de regressão.

A regressão linear é uma classe de algoritmos de regressão capazes de receber um conjunto de observações como entrada (contendo os atributos e os valores dependentes desses atributos) e ser treinado para encontrar uma equação linear que possa estimar os valores dependentes para novas observações contendo apenas os atributos.

Sendo assim, esse tipo de regressão é aplicável ao problema sendo tratado neste projeto. Como existem vários algoritmos para realização de regressões lineares, ao longo da análise vamos buscar encontrar o que funciona melhor para nosso conjunto de dados e, posteriormente, otimizá-lo.

Os algoritmos que testaremos, os quais estão disponíveis na biblioteca *scikit learn*, são:

Linear Regression

Implementação da regressão linear dos mínimos quadrados ordinários. A lógica de funcionamento é tentar minimizar o somatório dos quadrados das diferenças entre os valores originais e estimados.

Ridge Regression

Esse algoritmo resolve alguns dos problemas dos mínimos quadrados ordinários, aplicando uma penalidade ao tamanho dos coeficientes.

Lasso

É um modelo linear que estima coeficientes esparsos, o que é útil em alguns contextos devido a sua tendência de priorizar soluções com poucos coeficientes diferentes de zero, diminuindo a quantidade de atributos dos quais a solução depende.

ElasticNet

É modelo de regressão linear que usa regularização *L1* e *L2* dos coeficientes, sendo útil em bases esparsas. É útil quando existem vários recursos correlacionados entre si.

Bayesian Regression

Impõe parâmetros de regularização não fixos durante processo de estimação. Tais parâmetros são otimizados para os dados utilizados, fazendo com que se adaptem aos dados.

Testaremos também dois modelos baseados em árvores de decisão:

Decision Trees

Trata-se de um modelo não paramétrico usado tanto para classificação quanto para regressão. O algoritmo consiste em construir uma árvore de decisão que seja capaz de estimar o valor desejado com base nos atributos de entrada.

XGBoost (eXtreme Gradient Boosting)

É uma implementação de *gradient boosted decision trees* focada em performance (do próprio modelo) e em velocidade computacional. Algumas características são:

- Habilidade de lidar com valores faltantes;
- Paralelismo;
- Possibilidade continuar a treinar um modelo já treinado usando outros dados.

Para que os algoritmos possam ser usados da melhor forma possível, precisaremos aplicar algumas técnicas de pré-processamento, de modo a ajustar os dados. Já verificamos acima que:

- Os dados nulos precisam ser ajustados;
- Os dados categóricos precisam ser convertidos usando a técnica *one-hot encoding*;
- A assimetria identificada do preço precisa ser ajustada. Usaremos a aplicação da função de *log* para esse fim;
- Os dados numéricos precisam ser normalizados para afetarem de forma igual o processo de treinamento.

Usaremos, também, a técnica conhecida como PCA (*Principal Component Analysis*) para evitar o problema da dimensionalidade. Tal técnica busca formar componentes que representem de forma agregada a variância dos dados, o que nos dá, também, o efeito de selecionar atributos.

Para realizarmos as primeiras regressões que servirão como *benchmark* para as demais, será usada a correlação dos atributos com o preço do imóvel, de modo a serem usados apenas os com maior grau de correlação.

Quando os dados estiverem totalmente trabalhados, vamos otimizar o algoritmo de melhor performance, usando a técnica de *Grid Search*, a qual é capaz de receber uma lista de parâmetros aceitos pelo modelo acompanhados de um conjunto de valores que se deseja testar. Com base nisso são executadas simulações do modelo com todas as possíveis combinações dos valores dos parâmetros. Por exemplo, se o modelo aceitar dois parâmetros numéricos Ae B, e informarmos que queremos testar os valores 1 e 2 para A, e 10 e 30 para B, o modelo será executado com as 4 possíveis combinações: A=1 e B=10; A=1 e B=30; A=2 e B=10; A=2 e B=30. Ao final é possível verificar qual combinação obteve o melhor resultado.

Outra técnica que será usada como alternativa para solucionar o problema de prever os preços é a aplicação de redes neurais como ferramenta de regressão.

Redes neurais simulam o modo de funcionamento do cérebro humano. São formadas por camadas estruturadas constituídas por unidades de processamento. Pesos são atribuídos a cada uma dessas unidades e vão sendo ajustados durante o treinamento, a fim de se atingir o objetivo pretendido, que é a capacidade de fornecer um resultado com base nas entradas. A determinação de quais são os dados de entrada e de saída depende do problema que está se buscando resolver. No nosso caso, a entrada será formanda pelos atributos dos imóveis, enquanto a saída será a estimativa do preço de venda.

Benchmark

Uma boa prática quando se busca treinar modelos de aprendizagem supervisionada é calcular, como *benchmark* inicial, o índice de performance escolhido com base em uma previsão ingênua. Usaremos o valor da média e da mediana. Ou seja, criaremos um vetor de resultado totalmente preenchido com o valor da média, e, depois, outro com o valor da mediana.

Conforme comentado anteriormente, usaremos duas métricas: *R2 Score* e *RMSLE*.

Evoluiremos o *benchmark* inicial com as métricas calculadas com base em previsões que usarão apenas os atributos mais correlacionados com o preço de venda, e modelos sem otimizações.

III - Metodologia

Pré-processamento

Ajuste dos dados nulos - dados categóricos

Começamos com a atribuição do valor *NA* para os atributos `PoolQC`, `MiscFeature`, `Alley`, `Fence`, `FireplaceQu`, `GarageCond`, `GarageQual`, `GarageFinish`, `GarageType`, `BsmtFinType2`, `BsmtExposure`, `BsmtFinType1`, `BsmtCond` e `BsmtQual`, indicando que não existe tal característica no imóvel, ou ela não se aplica devido ao fato de ser dependente de outro atributo que não consta da propriedade. Um exemplo disso são os atributos sobre a garagem (condição, tipo, etc.). Se não existir garagem, esses atributos não fazem sentido. Por outro lado, caso exista garagem e esses dados pudessem estar preenchidos, não teríamos como descobrir seus valores, de modo que sempre colocar o valor NA faz sentido.

O atributo `MasVnrType` indica o tipo de alvenaria, e o colocamos como `None`, quando era nulo.

O atributo `Electrical` indica o tipo de ligação elétrica. O tipo mais genérico seria o misturado `Mixed`, de modo que usamos o valor `Mix`.

Os atributos `LotFrontage` e `MasVnrArea` dizem respeito a medidas numéricas. A opção foi atribuirmos o valor zero.

O atributo `GarageYrBlt` indica o ano de construção da garagem. Considerando que há o mesmo número de linhas com esse atributo nulo, quando comparado ao número de linhas com os demais atributos da garagem também nulos, provavelmente são imóveis sem garagem, de forma que foi atribuído o valor zero.

Cálculo das métricas a serem usadas como base

Foram calculadas as métricas para um conjunto de dados que tinha como valores previstos a média dos valores originais e também para um conjunto de dados que tinha como valores previstos a mediana.

```
Resultados para o modelo Média:
R2 Score:      0.000
RMSLE:         0.408

Resultados para o modelo Mediana:
R2 Score:     -0.051
RMSLE:         0.400
```

Com base nos resultados acima, o objetivo passou a ser buscar projeções que apresentassem, pelo menos, *R2 Score* maior do que 0.0 e *RMSLE m*enor do que 0.4.

Regressões com atributos que possuem correlação maior do que 50% com o preço de venda

Foram executadas regressões para os modelos propostos com base nos atributos que possuem correlação maior do que 50% com o preço de venda, sem nenhum tipo de otimização.

Foi feita a a divisão dos dados em dois conjuntos, um para treinamento e outro para testes.

Resultados:

```
LinearRegression:
------------------------
R2 Score:      0.671
RMSLE:         0.222

Ridge:
------------------------
R2 Score:      0.666
RMSLE:         0.208

Lasso:
------------------------
R2 Score:      0.671
RMSLE:         0.221
```

```
ElasticNet:
------------------------
R2 Score:    -16.328
RMSLE:         0.330

DecisionTreeClassifier:
------------------------
R2 Score:      0.695
RMSLE:         0.236

XGBoost:
------------------------
R2 Score:      0.853
RMSLE:         0.148
```

Foi realizado o treinamento e execução da rede neural. Como durante o treinamento é usado tanto um conjunto de treinamento quanto um de validação, foi feita uma nova divisão do conjunto de testes em dois (validação e testes). O objetivo é sempre realizar testes em um conjunto de dados que nunca foram vistos pelo modelo.

```
RedeNeural:
------------------------
R2 Score:      0.546
RMSLE:         0.170
```

Os melhores índices alcançados (*R2 Score* = 0.853 e *RMSLE* = 0.148) são melhores do que os anteriores, portanto serão usados como novo *benchmark* para as futuras tentativas de melhorar os resultados.

Remoção de outliers

*Outliers podem ter um impac*to negativo em vários modelos. Por isso foram realizados ajustes nos *outliers* para o preço e para os atributos com correlação maior do que 50% com o preço. Após a remoção de *outliers*, restaram 1.391 linhas de observações.

A execução da regressão após a remoção dos *outliers* resultou nos seguintes números:

```
LinearRegression:
------------------------
R2 Score:      0.779
RMSLE:         0.230

Ridge:
------------------------
R2 Score:      0.768
RMSLE:         0.198

Lasso:
------------------------
R2 Score:      0.778
RMSLE:         0.229

ElasticNet:
------------------------
R2 Score:     -8.515
RMSLE:         0.284
```

```
DecisionTreeClassifier:
------------------------
R2 Score:      0.579
RMSLE:         0.255

XGBoost:
------------------------
R2 Score:      0.805
RMSLE:         0.147

Rede Neural:
------------------------
R2 Score:      0.684
RMSLE:         0.171
```

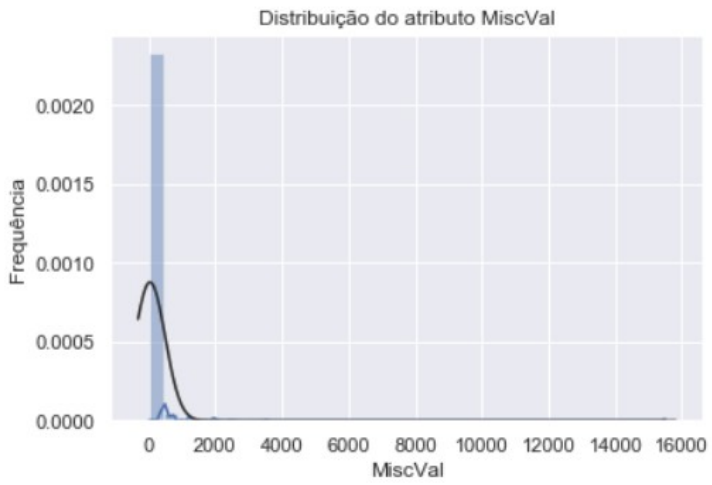
Tratamento das assimetrias

Exibimos abaixo, para os os atributos numéricos, suas assimetrias, em ordem decrescente.

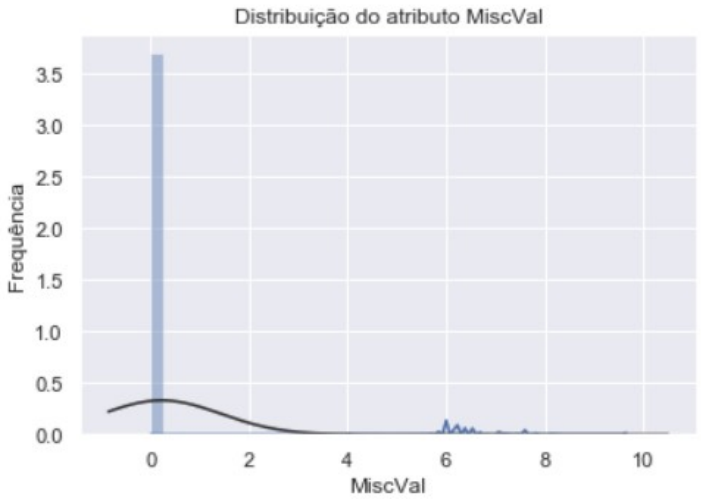
```
MiscVal          29.115699
PoolArea         21.793020
LotArea          13.109708
3SsnPorch        10.667415
LowQualFinSF     10.438691
KitchenAbvGr      4.657103
BsmtFinSF2        4.303674
BsmtHalfBath      4.113736
ScreenPorch       4.029569
EnclosedPorch     2.841887
MasVnrArea        2.684079
OpenPorchSF       2.397688
WoodDeckSF        1.600833
MSSubClass        1.390503
BsmtUnfSF         0.863872
OverallCond       0.739404
2ndFlrSF          0.730514
HalfBath          0.698389
BsmtFinSF1        0.681109
```

```
Fireplaces        0.667579
BsmtFullBath      0.628193
1stFlrSF          0.620751
GrLivArea         0.512476
TotRmsAbvGrd      0.412386
MoSold            0.221973
OverallQual       0.135151
TotalBsmtSF       0.109683
YrSold            0.097245
Id               -0.001253
LotFrontage      -0.020480
BedroomAbvGr     -0.033718
FullBath         -0.041668
GarageArea       -0.062739
GarageCars       -0.448420
YearRemodAdd     -0.471064
YearBuilt        -0.537706
GarageYrBlt      -3.931213
```

O atributo com maior assimetria (`MiscVal`) possui a seguinte distribuição:



Todos os atributos com assimetria maior do que 0.5 foram ajustados com a aplicação da função de logaritmo neperiano. Após esse ajuste, o atributo ficou com a seguinte distribuição:



Normalização dos dados numéricos

Para evitar que atributos com valores numéricos mais altos afetassem mais os modelos do que atributos com escalas menores, eles foram normalizados com o uso da classe *MinMaxScaler* da biblioteca *Scikit Learn*.

One-hot encode dos dados categóricos

Muitos modelos não lidam bem com dados categóricos. Por isso as categorias foram transformadas em novas colunas usando a técnica de *one-hot encoding*, por meio do uso da função *get_dummies* da biblioteca *Pandas*. Antes da transformação havia 43 colunas. Depois da transformações, passou-se a ter 261 colunas.

Os resultados das regressões passou a ser:

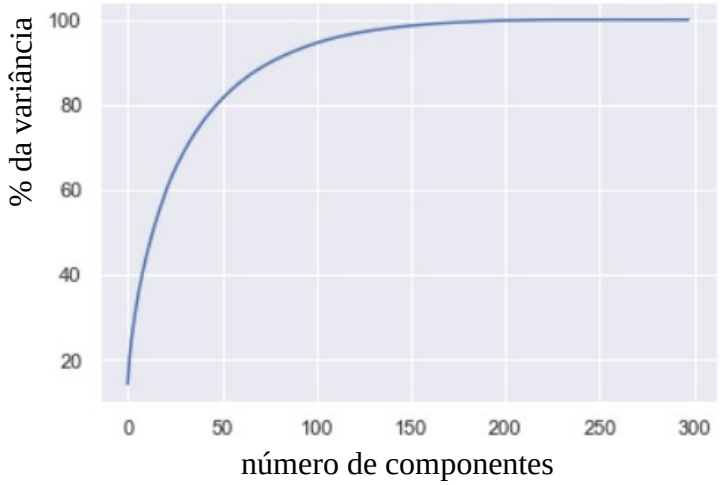
```
LinearRegression:
------------------------
R2 Score:     -0.013
RMSLE:         3.147

Ridge:
------------------------
R2 Score:      0.885
RMSLE:         0.132

Lasso:
------------------------
R2 Score:      0.873
RMSLE:         0.149

ElasticNet:
------------------------
R2 Score:      0.293
RMSLE:         0.189
```

```
DecisionTreeClassifier:
------------------------
R2 Score:      0.480
RMSLE:         0.263

XGBoost:
------------------------
R2 Score:      0.883
RMSLE:         0.117

RedeNeural:
------------------------
R2 Score:      0.866
RMSLE:         0.127
```

PCA

Para tentar conseguir mais melhorias nos resultados, aplicamos a técnica PCA (*Principal Component Analysis*), que nos dá a redução das dimensões dos dados por meio da criação de componentes que melhor representam a variância contida nos dados.

Inicialmente aplicamos a técnica usando como quantidade de componentes a serem criados o número de colunas existentes (298), de modo que foi possível observar como os componentes foram explicando, cumulativamente, a variância dos dados:



Percebe-se que com cerca da metade dos componentes já é possível explicar quase que a totalidade da variância observada nos dados.

Por esse motivo foi feito teste de regressão com 150 componentes.

Resultados:

```
LinearRegression:
-------------------------
R2 Score:      0.881
RMSLE:         0.147

Ridge:
-------------------------
R2 Score:      0.872
RMSLE:         0.132

Lasso:
-------------------------
R2 Score:      0.881
RMSLE:         0.146

ElasticNet:
-------------------------
R2 Score:    -80.101
RMSLE:         0.334
```

```
DecisionTreeClassifier:
-------------------------
R2 Score:      0.050
RMSLE:         0.340

XGBoost:
-------------------------
R2 Score:      0.676
RMSLE:         0.169

RedeNeural:
-------------------------
R2 Score:      0.763
RMSLE:         0.152
```

Otimizações

Inicialmente as otimização no *XGBoost* foram realizadas parâmetro a parâmetro, chegando-se aos seguintes valores:

- max_depth=3
- colsample_bytree=0.3
- min_child_weight=0.3
- gamma=0
- learning_rate=0.1
- n_estimators=300
- reg_alpha=1e-5
- reg_lambda=0.4
- subsample=0.5

Depois variou-se cada parâmetro para mais e para menos e usou-se *GridSearch* para encontrar a melhor combinação. As opções para cada parâmetro foram as seguintes:

- max_depth:[3,4]
- colsample_bytree:[0.2,0.3,0.4]
- min_child_weight:[0.2,0.3,0.4]
- gamma:[0,0.01]
- learning_rate:[0.08, 0.1, 0.12]
- n_estimators:[200, 300, 400]
- reg_alpha:[1e-4, 1e-5, 1e-6]
- reg_lambda:[0.35,0.4,0.45]
- subsample:[0.4,0.5,0.6]

Após 26.244 execuções com todas as possíveis combinações de parâmetros, o melhor resultado foi obtido com a seguinte configuração:

- colsample_bytree: 0.2
- gamma: 0
- learning_rate: 0.08
- max_depth: 3
- min_child_weight: 0.2
- n_estimators: 400
- reg_alpha: 0.0001
- reg_lambda: 0.35
- subsample: 0.6

Com esses parâmetros, o resultado da regressão passou a ser:

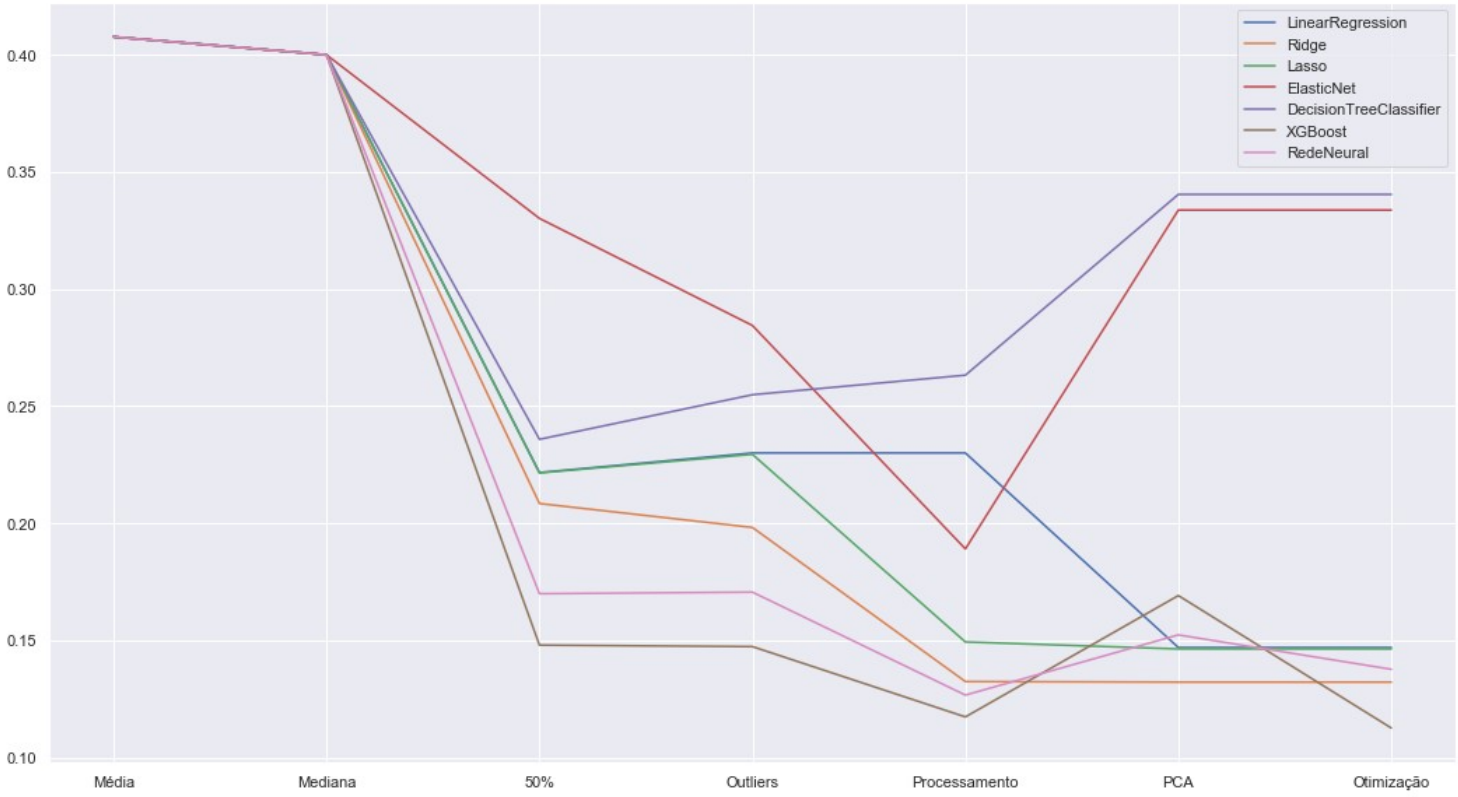
```
XGBoost:
-------------------------
R2 Score:      0.890
RMSLE:         0.113
```

Foram feitas várias tentativas de otimização da rede neural, mas sem sucesso.

IV - Resultados

Avaliação e validação do modelo

Visualização dos resultados obtidos pelos diversos modelos para *RMSLE*:



Como pode ser visto, os melhores modelos foram o *XGBoost* (fase de otimização) e a Rede Neural (fase de processamento dos dados). Vamos analisar a evolução de cada um deles.

Análise do XGBoost

	R2	RMSLE
Média	0.000000	0.407598
Mediana	-0.050924	0.399948
50%	0.852617	0.147940
Outliers	0.804706	0.147305
Processamento	0.882630	0.117279
PCA	0.675795	0.169081
Otimização	0.889892	0.112534

Esse algoritmo é altamente eficiente e otimizado. Ele é capaz de lidar por conta própria com dados nulos e fazer regularização e *cross validation*. Por esse motivo, percebe-se que apresentou excelentes resultados desde o início (bem melhores do que nossa métrica *naive*) e foi tendo melhorias

consecutivas, exceto quando fizemos o PCA, o que mostra sua grande capacidade de lidar com bases esparsas e ser sensível às nuances existentes.

O melhor resultado obtido foi ao final da otimização usando *Grid Search.*

Análise da Rede Neural

	R2	RMSLE
Média	0.000000	0.407598
Mediana	-0.050924	0.399948
50%	0.546029	0.169877
Outliers	0.684490	0.170557
Processamento	0.865888	0.126594
PCA	0.762621	0.152301
Otimização	0.691450	0.137607

A rede neural usada foi apresentando ganhos até o processamento da base de dados. Quando foi aplicado o PCA, o resultado piorou. Por algum motivo que não consegui identificar, a otimização também não foi capaz de melhorar a performance do modelo. Em algumas situação parecia que a rede estava travada, ou carregada com os pesos da última execução, apesar de sempre ser criada uma nova rede a cada treinamento. Para tentar evitar esse comportamento, passei a chamar as funções `backend.clear_session` e `model.reset_states`, mas não obtive sucesso.

Análise de sensibilidade

Durante todo o processo, sempre que treinamos os modelos e depois testamos, fizemos a separação de dados entre conjunto de treinamento e testes. No caso da rede neural, ainda usamos um conjunto de validação. Isso significa que os testes sempre foram realizados com dados ainda não vistos pelo modelo. Isso dá robustez ao nosso processo.

Durante a fase de otimização, como usamos o recurso *GridSearchCV* do *scikit-learn*, e ele faz validação cruzada, também estamos seguros de não termos sofrido de *overfitting*.

Fizemos algumas modificações aleatórias nos preços para ver como o modelo *XGBoost* reagiria.

As alterações consistiram em sortear 10% das observações e aplicar uma modificação percentual aleatória nos preços de, no máximo, 10%.

Os resultados da regressão usando *XGBoost* foi:

```
XGBoost:
-----------------------
R2 Score:      0.874
RMSLE:         0.121
```

A perda que tivemos com as alterações efetuadas nos dados foi pequena, o que indica que o modelo não é muito sensível, ou seja, é capaz de lidar bem com ruídos nos dados.

Justificativa

Como já demonstrado, a escolha do modelo *XGBoost* pode ser considerada boa devido à grande melhoria obtida em relação ao *benchmark inicial*, que foi a projeção ingênua usando média e mediana. Além disso, o modelo foi evoluindo bem com os ajustes que fizemos nos dados e, por fim, com as otimizações dos hiperparâmetros. Além disso, foi capaz de suportar a introdução de ruído nos preços sem grandes impactos.

Por fim, comparando o melhor índice obtido, que foi *RMSLE* igual a 0.112534, com o ranking da competição no *Kaggle*, ficaríamos na posição 251 de 4.267 competidores, o que é um bom resultado inicial.

V - Conclusão

Visualização livre

Vamos visualizar como ficou o valor previsto em comparação com o valor original para a regressão usando o melhor modelo com os melhores parâmetros.

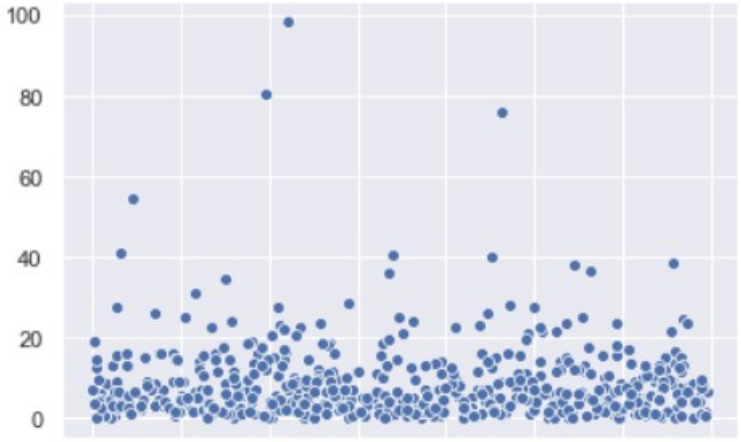
Menores diferenças entre a previsão e o valor real:

	Previsto	Valor Real	Diferença	Percentual
580	118461.359375	118500	38.640625	0.032608
1361	183004.187500	182900	104.187500	0.056964
481	139865.140625	140000	134.859375	0.096328
10	129357.718750	129500	142.281250	0.109870
1047	170286.562500	170000	286.562500	0.168566
839	157270.531250	157000	270.531250	0.172313
1078	214577.656250	215000	422.343750	0.196439
1195	127244.132812	127500	255.867188	0.200680
500	159327.906250	159000	327.906250	0.206230
261	266550.000000	266000	550.000000	0.206767

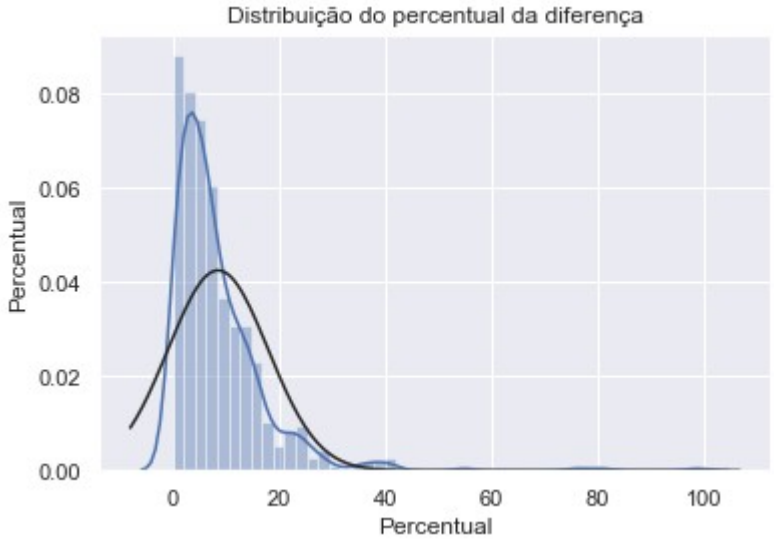
Maiores diferenças entre a previsão e o valor real:

	Previsto	Valor Real	Diferença	Percentual
443	123944.796875	62383	61561.796875	98.683611
393	108356.015625	60000	48356.015625	80.593359
924	66680.406250	37900	28780.406250	75.937747
92	207284.718750	133900	73384.718750	54.805615
65	253847.718750	180000	73847.718750	41.026510
679	183612.984375	130500	53112.984375	40.699605
901	175359.562500	124900	50459.562500	40.399970
1316	155364.796875	112000	43364.796875	38.718569
1090	110341.281250	80000	30341.281250	37.926602
1129	129961.445312	95000	34961.445312	36.801521

Distribuição dos percentuais da diferença entre valor real e valor previsto.



Visualização em forma de distribuição normal dos percentuais de diferença entre valor real e valor previsto.



Pode-se afirmar que o modelo fez, na grande maioria das vezes, boas previsões.

A distribuição dos percentuais de diferença é altamente assimétrica à direita, o que é um ótimo sinal.

A grande maioria das projeções apresenta taxa de erro inferior a 10%.

Reflexões

Este projeto começou carregando os dados a partir da base de dados do *Kaggle* e foi evoluindo no tratamento dos dados, com ajustes de dados nulos e assimetrias, transformação de dados categóricos em novas colunas, remoção de *outliers*, normalização de dados e seleção de atributos.

Depois foram realizadas otimizações nos melhores modelos.

A parte mais desafiadora foi a otimização. É complexo definir quais parâmetros devem ser otimizados, e cada opção adicionada à lista de parâmetros a serem combinados duplica o tempo de execução do *GridSearch*. Ou seja, trata-se de um crescimento exponencial. Sendo assim, uma otimização pode, facilmente, acabar levando semanas para ser concluída.

Acredito ter chegado a um modelo bastante satisfatório, que poderia ser usado em negócios de verdade.

Melhorias

Uma possível melhoria seria a execução de otimizações mais completas usando uma maior combinação de parâmetros.

Outra possível tentativa de melhoria seria usar *embedings* no tratamento de dados categóricos na rede neural, no lugar de *one-hot encoding*.

Referências

[1] Almeida, Pedro Henrique Ramos. Fatores determinantes para a formação de preço no mercado imobiliário de Brasília. Brasília. 2001. Universidade de Brasília. Disponível em http://bdm.unb.br/bitstream/10483/2122/1/2011_PedroHenriqueRamosdeAlmeida.pdf

[2] Belfiore, Patrícia Prado. Fávero, Luiz Paulo Lopes. Lima, Gerlando A. S. Franco. Modelos de precificação hedônica de imóveis residenciais na região metropolitana de São Paulo: uma abordagem sob as perspectivas da demanda e da oferta. São Paulo Jan./Mar. 2008. Estud. Econ. vol.38. Disponível em http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-41612008000100004

[3] Kaggle. House Prices: Advanced Regression Techniques. Disponível em https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview