

**Universidade Federal do Rio de Janeiro
Engenharia de Computação e Informação
Tópicos Especiais em Sistemas de Controle**

**USO DE TÉCNICAS AVANÇADAS DE REGRESSÃO
PARA PREVER PREÇOS DE CASAS NA CIDADE DE
BOSTON**

Lucas Santos de Paula
lucasdepaula@poli.ufrj.br

Orientador: Heraldo L. S. Almeida

2018.1

Sumário

Sumário	2
Introdução	3
Ferramentas Utilizadas	3
Análise Inicial	3
Pré-processamento	10
Outliers	10
Análise da variável de saída	11
Preenchimento de valores faltantes	14
Conversão de colunas numéricas em categóricas	18
Tratamento para variáveis numéricas	19
Tratamento para variáveis categóricas	19
Treinando modelos, realizando testes e submetendo resultados para o Kaggle	19
Lasso	20
ElasticNet	20
KernelRidge	20
Gradient Boosting	20
XGBoost	20
LightGBM	21
Comparações de scores obtidos	21
Buscando melhorias	21
Média simples dos quatro melhores modelos avaliados	22
Média simples de todos os modelos	22
Conclusão	22

Introdução

O objetivo deste trabalho é utilizar técnicas matemáticas de regressão linear e análise de dados para prever preços de casas na cidade de Boston. Se você pedir para alguém descrever a casa dos seus sonhos, sem dúvidas ele dirá o tamanho do quintal, uma casa grande, com um número definido de quartos, entre outras coisas. Contudo, é importante ressaltar que existem diversos fatores envolvidos que influenciam no preço de um imóvel.

Esse trabalho irá fazer análise de um conjunto de dados que conta com 81 variáveis explanatórias que descreve quase todos os aspectos de imóveis nas proximidades da cidade em questão. O objetivo aqui é encontrar resultados factíveis fazendo análise do conjunto de treinamento e de testes. O dataset foi obtido na competição do site Kaggle chamada "House Prices: Advanced Regression Techniques".

Ferramentas Utilizadas

Todo o código desenvolvido ao longo deste trabalho foi utilizando a linguagem Python em sua versão 3 e algumas bibliotecas voltadas para análise, manipulação e visualização de dados, tais como:

pandas - Biblioteca de código aberto que provê ferramentas para uso de estruturas de dados e análise de dados.

Matplotlib - É uma ferramenta de plotagem de gráficos que gera figuras em diversos formatos, o que nos permite visualizar diversos tipos de gráficos durante as análises feitas.

Seaborn - É uma biblioteca de visualização baseada na matplotlib que provê uma interface de plotagem de gráficos estatísticos.

NumPy - Biblioteca matemática que possui diversos módulos de *data science* embutidos.

SciPy - Biblioteca matemática para operações com dados científicos. Neste trabalho foi utilizado principalmente o módulo de estatística, que contém um vasto número de distribuições de probabilidades, além do vasto número de funções estatísticas.

Scikit Learn - Biblioteca para mineração e análise de dados que faz uso de três bibliotecas anteriormente citadas: NumPy, SciPy e matplotlib. Inclui diversos módulos utilizados largamente em *machine learning* como classificação, regressão, clusterização, redução de dimensionalidade, seleção de modelos e pré-processamento de dados.

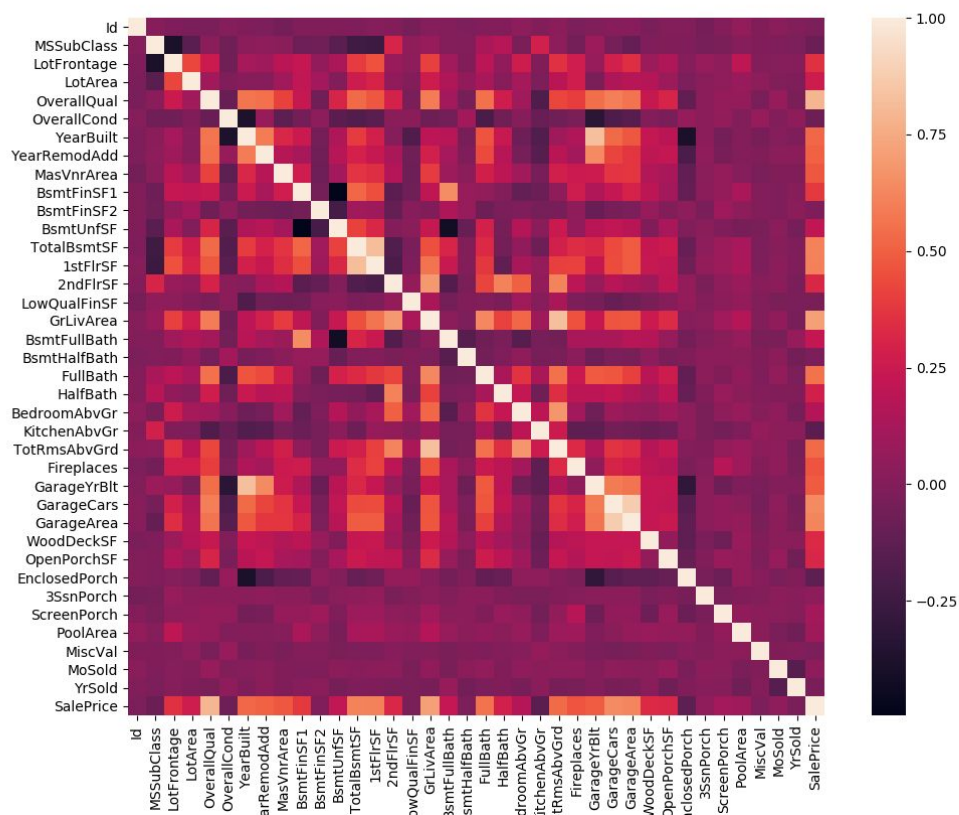
Análise Inicial

Como parte do entendimento do conjunto de dados e também do funcionamento das bibliotecas acima citadas, foi feita uma análise de dados **subjativa** a fim de explorar e entender o comportamento geral do conjunto.

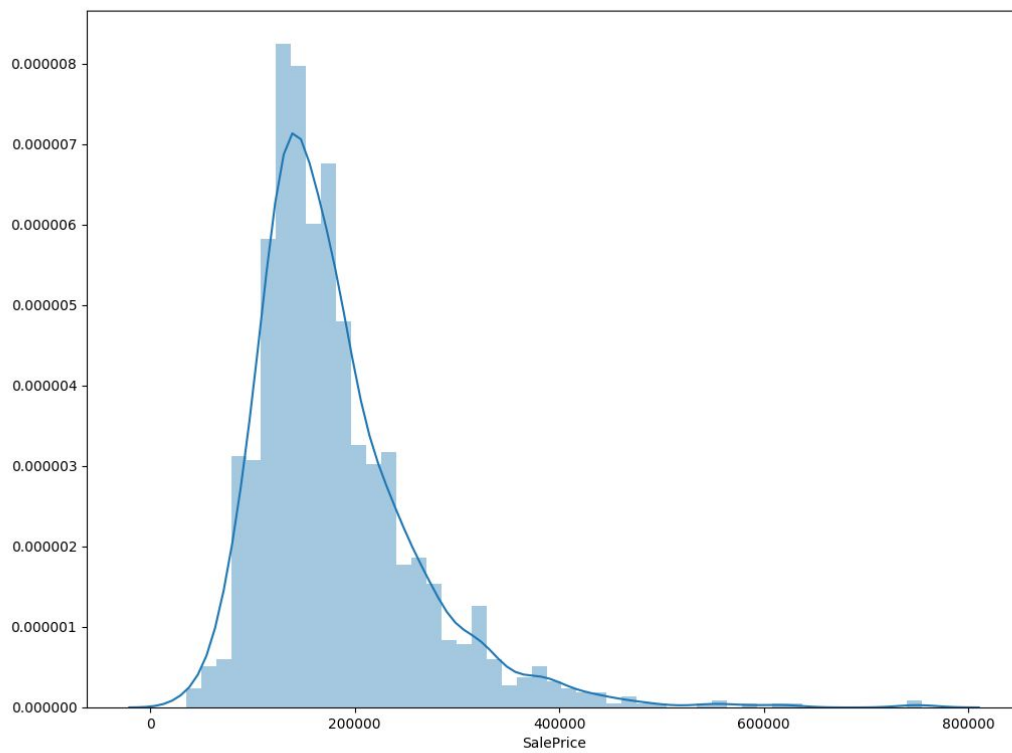
Nessa etapa, foi feita a importação do conjunto de testes totalmente cru e foi gerado uma matriz de correlação de variáveis, um histograma da coluna *SalePrice*, que é a nossa variável de saída no problema, e além disso, baseado em uma opinião totalmente subjetiva, foi feito um levantamento das variáveis com potencial influência no preço final. Essas variáveis foram divididas em variáveis numéricas e categóricas.

As numéricas foram: Área do Lote (LotArea), Área total do porão na medida de pés ao quadrado (TotalBsmtSF), Área Térreo (GrLivArea).

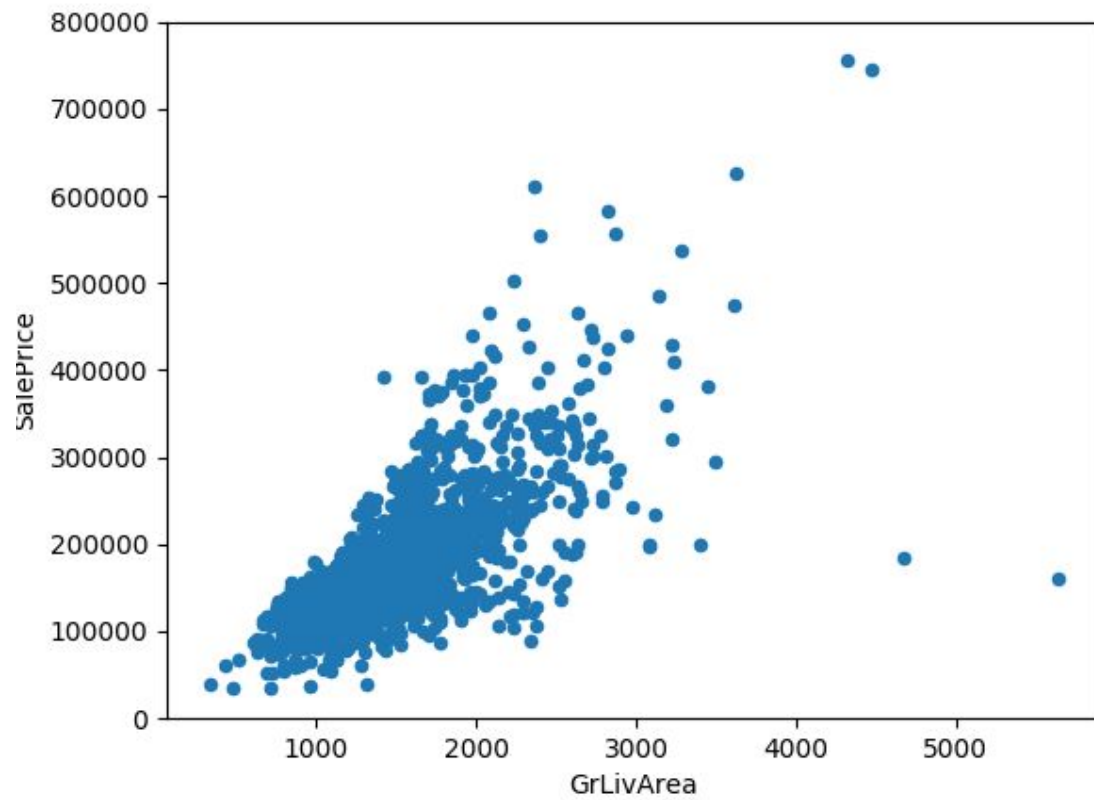
As categóricas foram: Avaliação geral do material e do acabamento da casa (OverallQual), Avaliação geral da condição da casa (OverallCond).



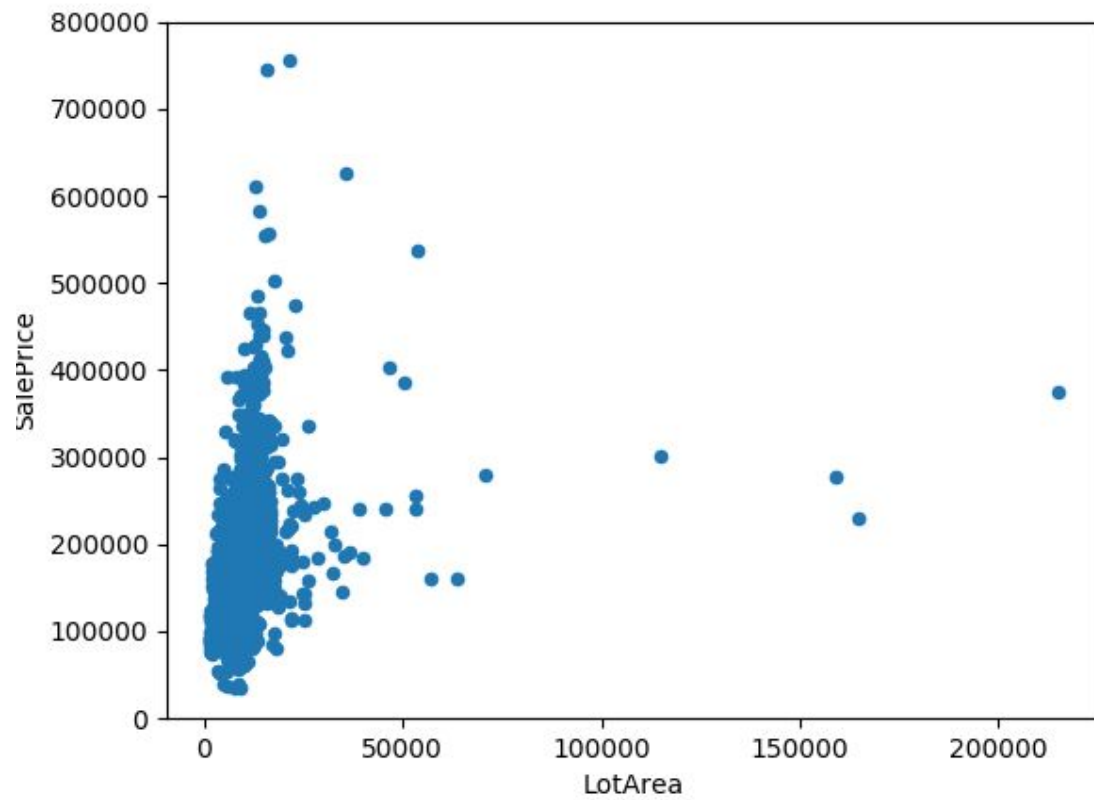
Matriz de Correlação do Conjunto de treinamento



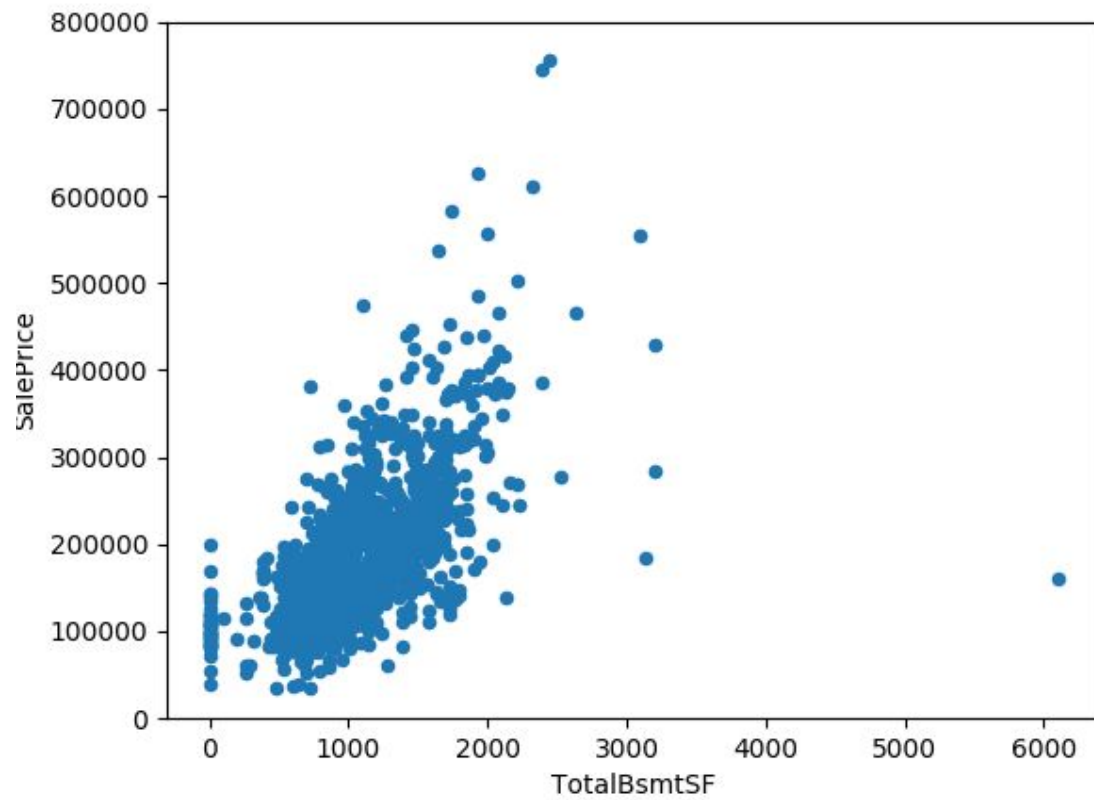
Histograma da variável SalePrice ou Preço de Venda.



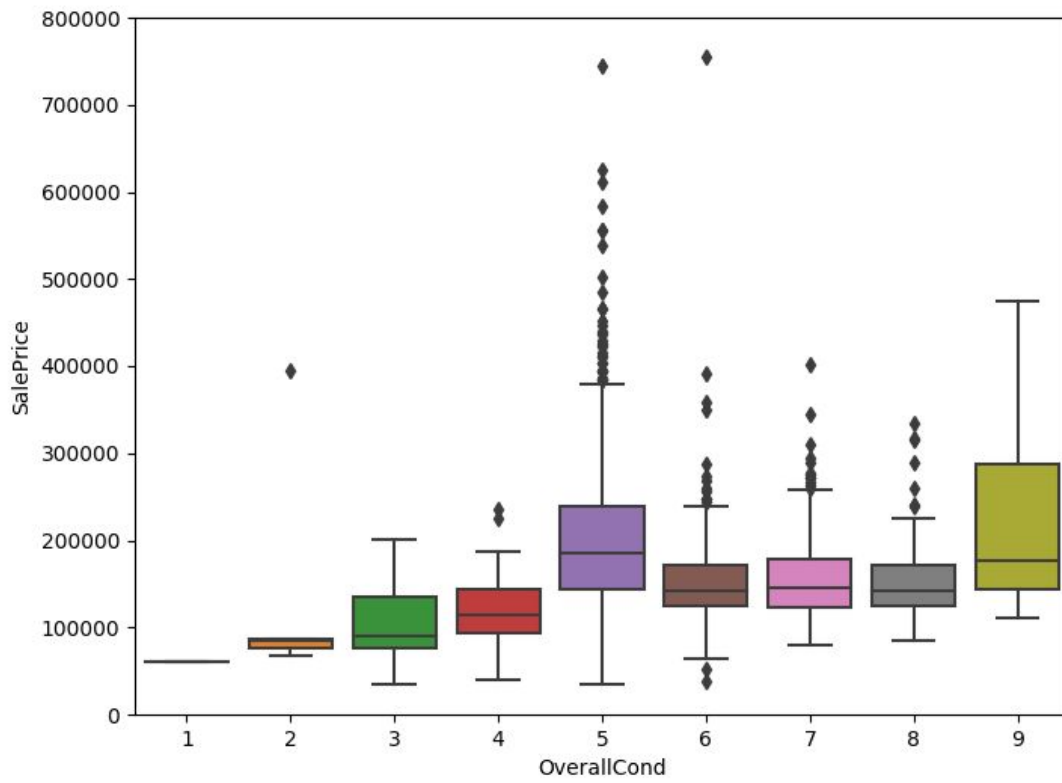
GrLivArea vs SalePrice



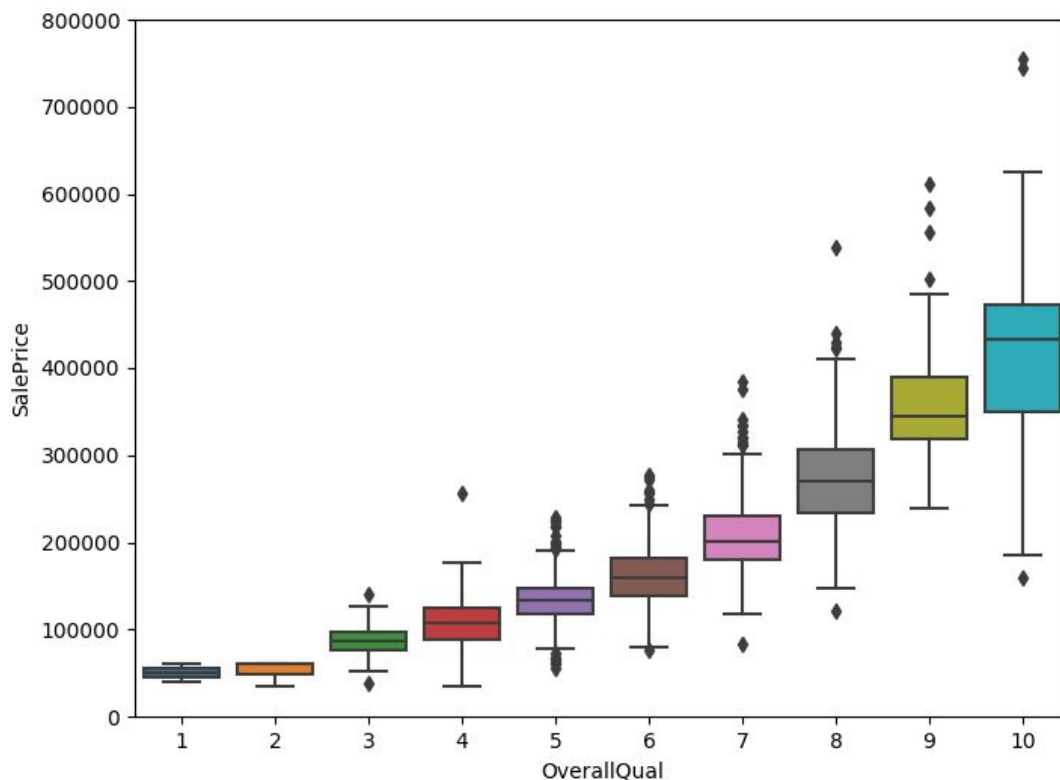
LotArea vs SalePrice



TotalBsmtSF vs SalePrice



Boxplot de OverallCond vs SalePrice



Boxplot de OverallQual vs SalePrice

Baseado nos gráficos iniciais da análise subjetiva, quanto as variáveis numéricas podemos concluir que as variáveis GrLivArea e TotalBsmtSF tem um relacionamento linear, sendo TotalBsmtSF um relacionamento mais forte. Já as variáveis categóricas podemos ver que quanto melhor a OverallQual, maior será o preço da casa, enquanto a OverallCond não diz tanto assim.

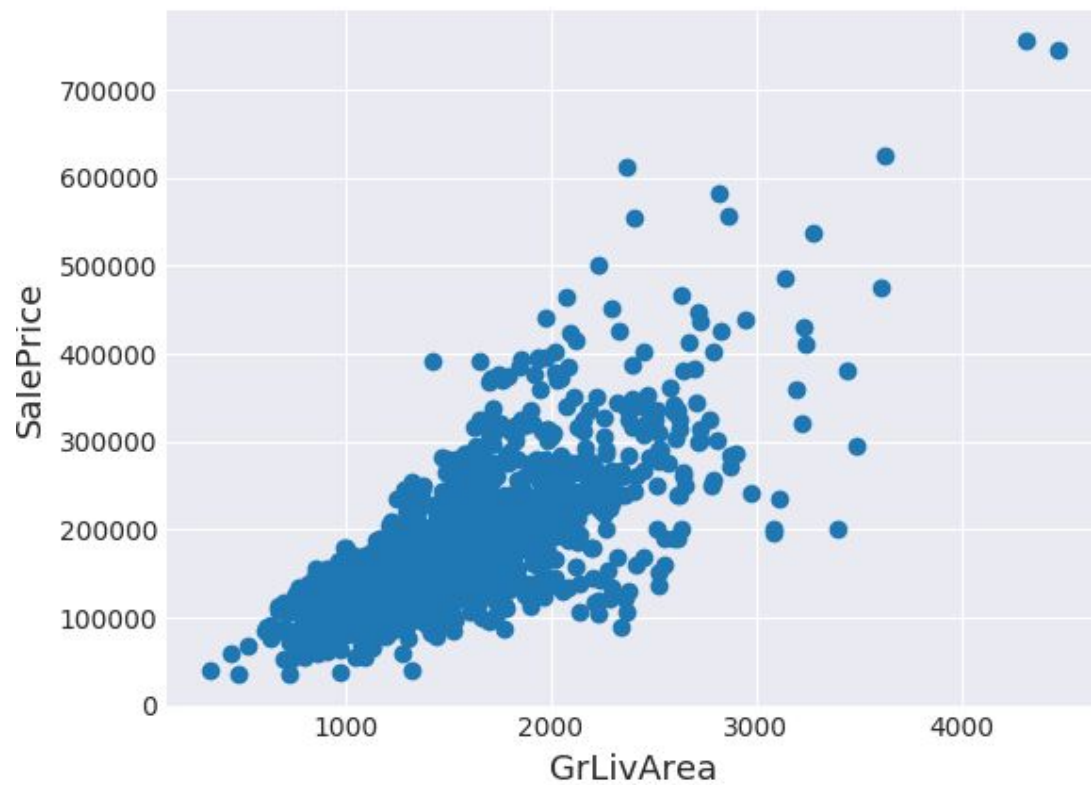
Pré-processamento

Passado a etapa de aprendizagem das bibliotecas e entendimento do dataset e seu comportamento de maneira geral, foi continuado o desenvolvimento do trabalho. Os próximos passos eram fazer uma análise completamente **objetiva** acerca do problema.

Outliers

O primeiro passo dado, visto que já foram feitas algumas análises, é remover outliers. Existem dois bem definidos quando olhamos para GrLivArea e SalePrice.

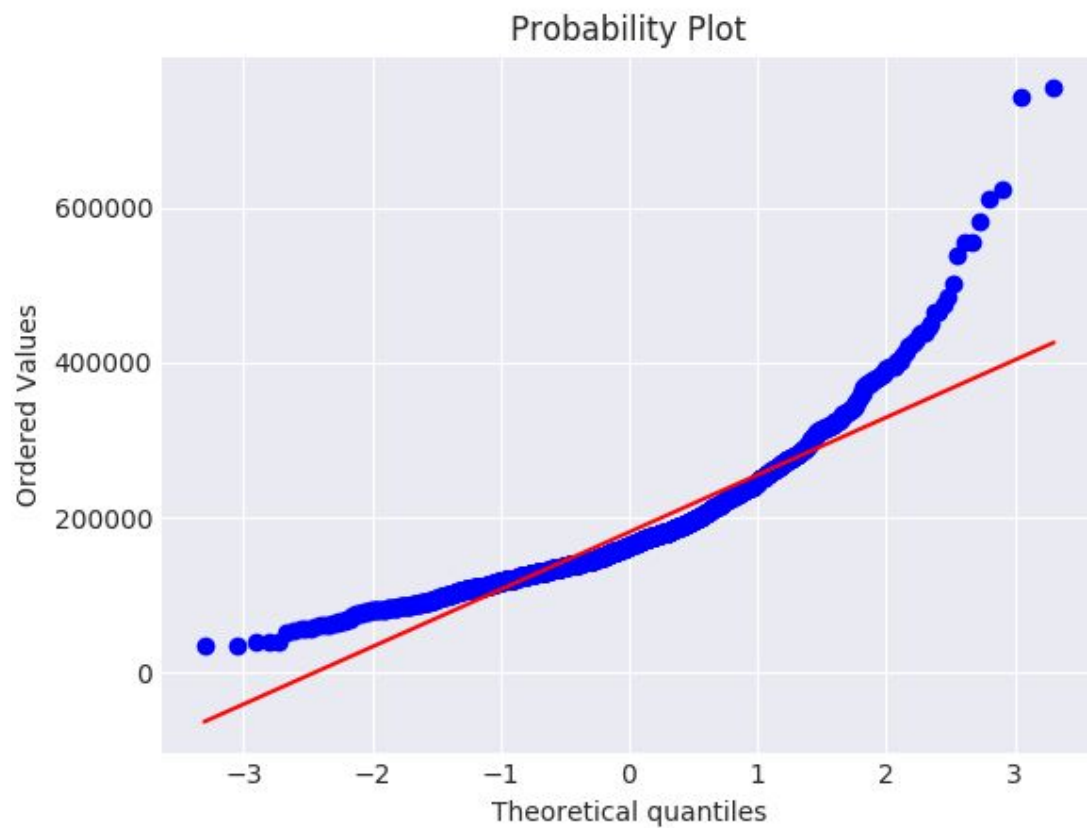
Após a remoção, o scatterplot dessas variáveis fica disposto da seguinte maneira:



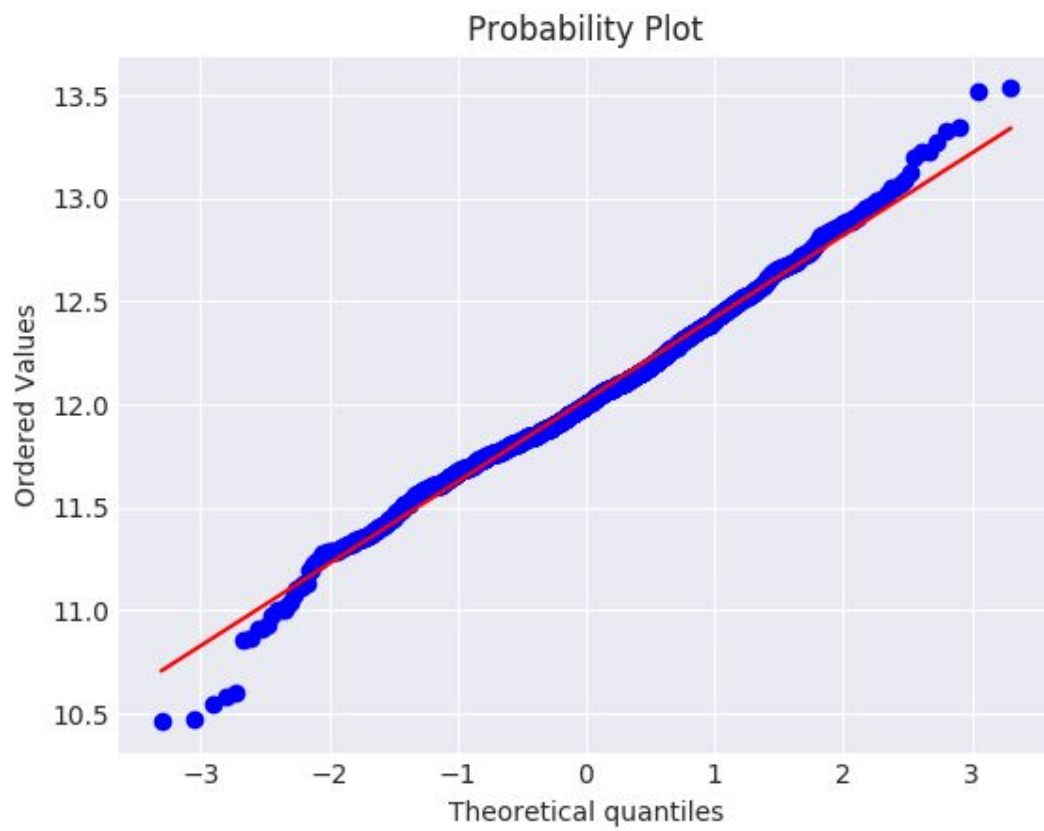
Que é bem mais linearmente definido que a versão indicada na figura X

Análise da variável de saída

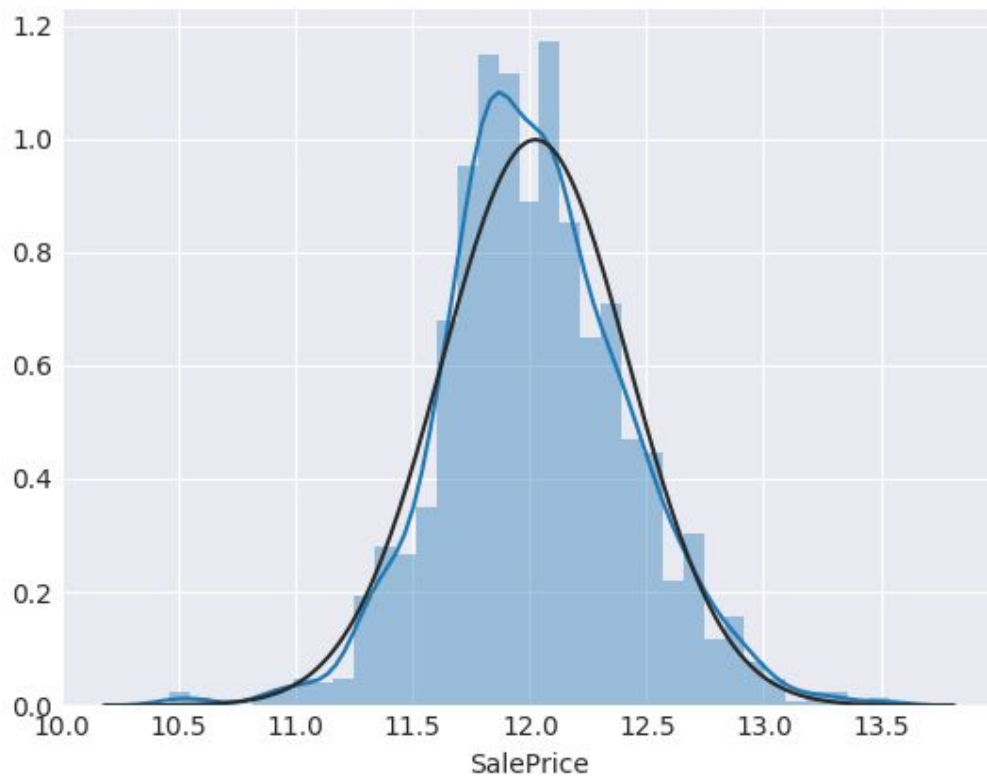
A próxima etapa foi plotar um gráfico de probabilidades da variável de saída, cujo resultado foi o seguinte:



O que nos permite concluir que a variável não é normalmente distribuída. Para melhorar esse gráfico, passamos a função `log1p` do `numpy` nos dados dessa coluna para ver se ocorria algum progresso.



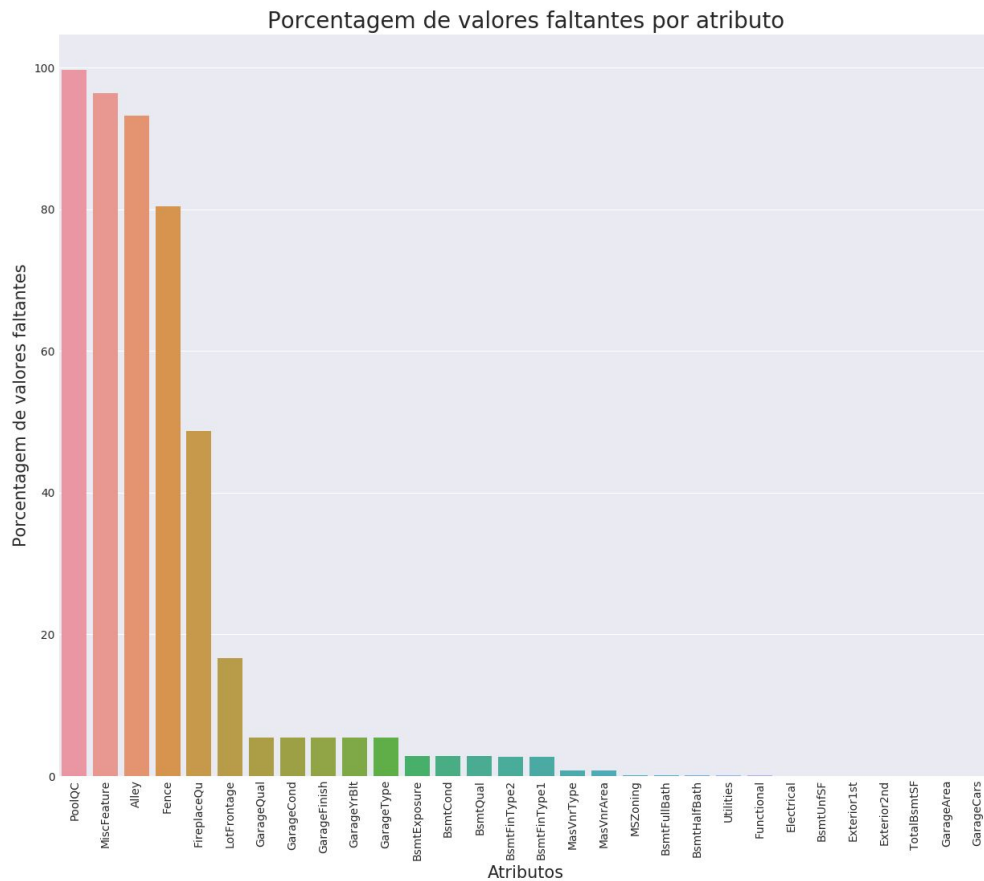
E nitidamente houve. Então mantive a coluna na escala logarítmica e gerei um novo histograma para ela:



Podemos ver que o comportamento do histograma está bem próximo de um modelo normalizado.

Preenchimento de valores faltantes

O próximo passo foi analisar os dados faltantes do dataset. Para tal, guardei a coluna SalePrice e a removi do dataset de treino. Em seguida, concatenei os datasets de treino e de testes, a fim de efetuar o pré-processamento de ambos. Após a concatenação foi feita uma varredura por valores faltantes, e um estudo de caso para cada variável, a fim de identificar o sentido daquele valor estar em falta.



PoolQC é o atributo com a maior porcentagem de dados faltantes em ambos os datasets. Ele indica o grau de qualidade da piscina da casa. "NA" nesse campo significa que não há piscina na casa em questão; Então, seus valores foram preenchidos com um dado não significativo para o problema: "None".

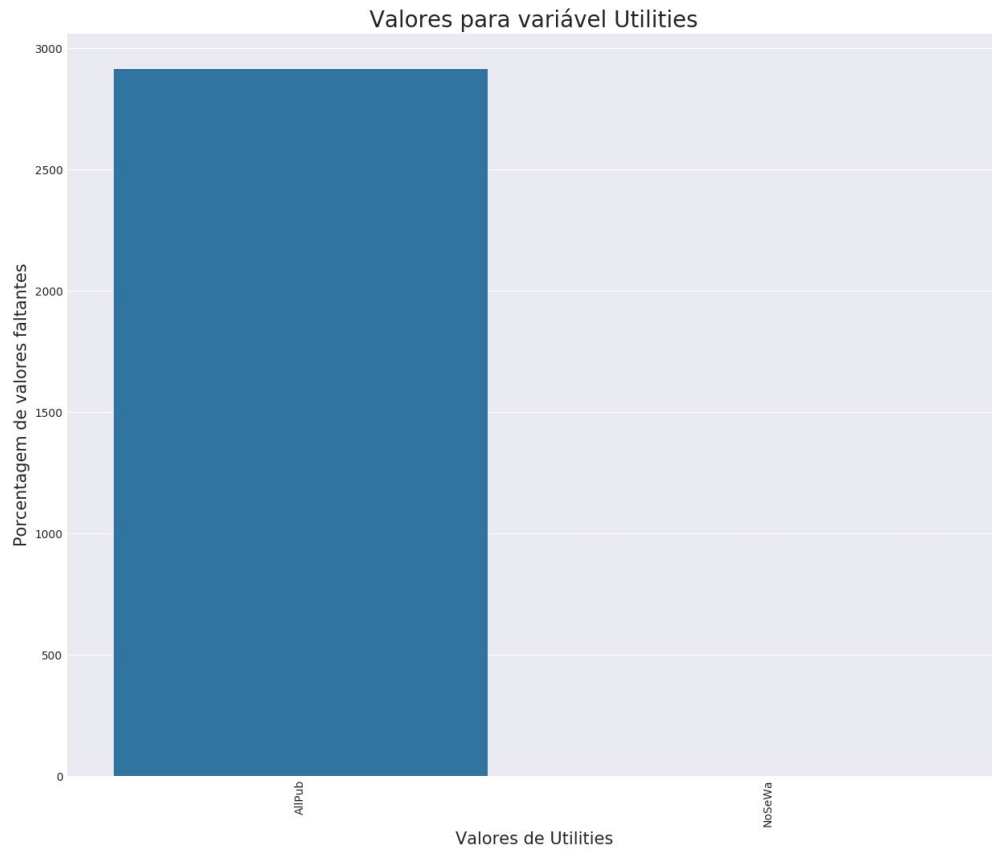
"NA" em MiscFeature significa que não há nenhuma MiscFeature na casa em questão, então foi feito o mesmo tratamento da variável anterior.

Já o valor NA em Alley, significa que não há acessos por "becos", seguindo então, o mesmo processo de PoolQC e MiscFeature. O mesmo procedimento também foi repetido para as variáveis Fence (cerca) e Fireplace(lareira), e para as variáveis categóricas relacionadas a garagem e porão (GarageType, GarageFinish, GarageCond, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, MasVnrType, MSSubClass).

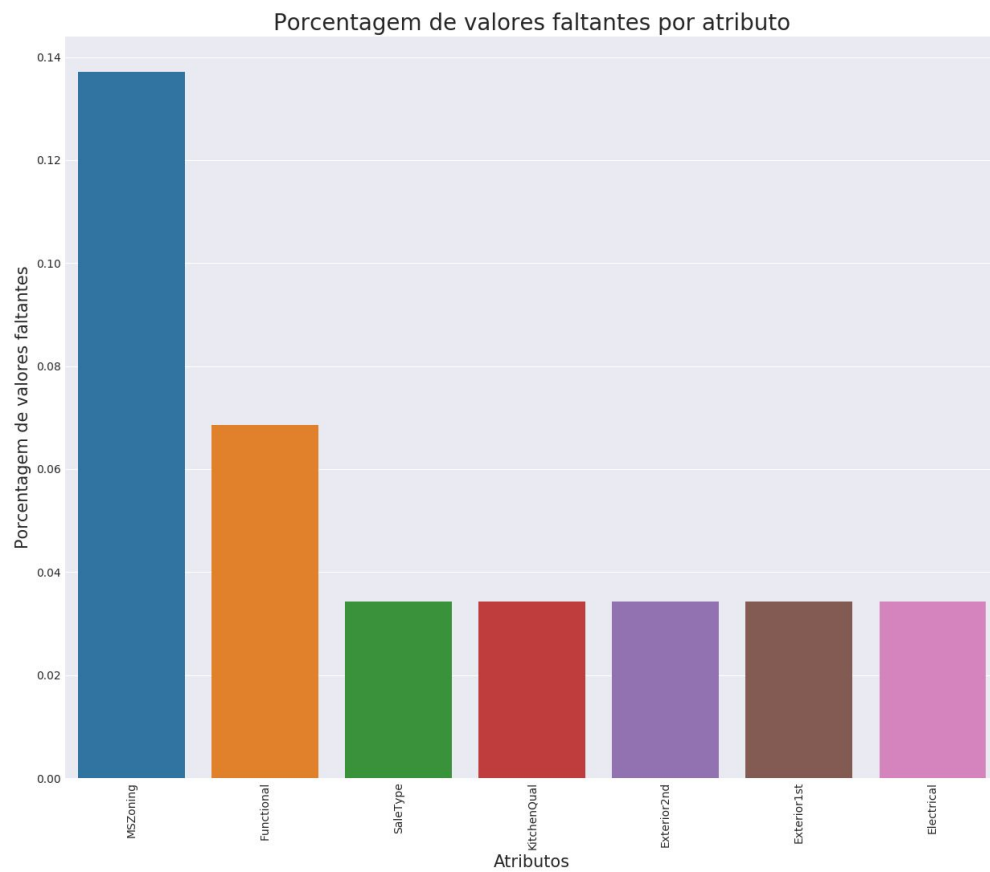
E, se não temos garagem, então o ano da construção, a área da garagem, quantidade de carros suportada, e atributos espaciais do porão também foram removidos tratados, sendo preenchidos com zero. ('GarageYrBlt', 'GarageArea', 'GarageCars', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath', 'MasVnrArea').

Uma terceira metodologia foi aplicada para a variável LotFrontage: ela significa a distância entre a rua e a casa. Se assumirmos a similaridade dos lotes de um mesmo bairro (Neighborhood), podemos assumir que o LotFrontage da vizinhança é a média dos seus valores para aquela vizinhança.

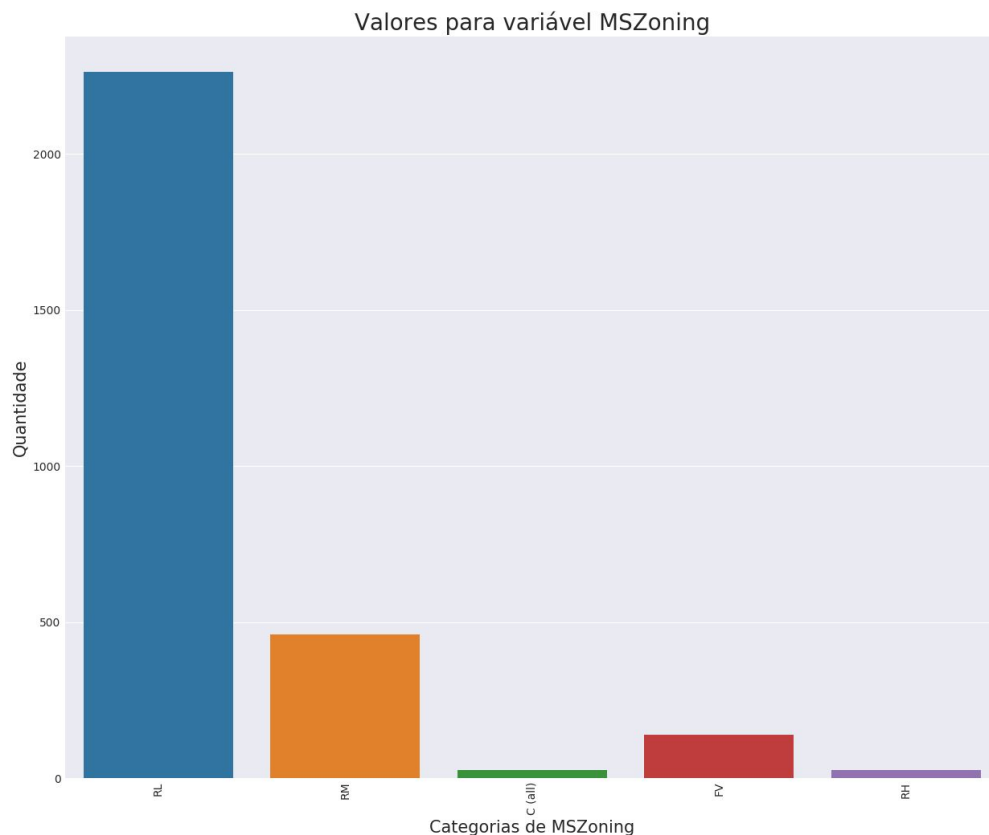
A coluna Utilities foi removida porque praticamente todos os seus valores são AllPub como ilustrado no gráfico abaixo. Por essa razão, ela não ajudaria nosso modelo a predizer um resultado.



Depois de tratadas todas essas variáveis, foi hora de dar uma olhada no panorama de dados faltantes do dataset e acompanhar as variáveis que estavam pendentes de tratamento.



Para decidir o que fazer com a variável MSZoning foi gerado um gráfico que exibía seus valores.



É possível ver que mesmo com diversas categorias presentes no dataset, uma em especial é dominante. O mesmo comportamento foi identificado nas variáveis KitchenQual, SaleType, Exterior1st, Exterior2nd e Electrical. Nesse caso, para preencher os valores faltantes foi utilizada a moda da variável em questão, isto é, o valor mais presente.

Restou somente uma variável a ser tratada: Functional. Nesse caso, NA significa Typical. Apenas foi feita uma substituição simples de NA para "Typ".

Conversão de colunas numéricas em categóricas

Feitas as substituições de valores faltantes no dataset, os próximos passos envolvem tratamentos diferentes para variáveis que são categóricas e variáveis que são numéricas. Contudo, algumas variáveis desse dataset são reconhecidas como números, quando, na verdade, representam categorias, dado o seu significado no problema proposto para esse estudo.

Feito isso, foram convertidas para string as variáveis MSSubClass, OverallCond, YrSold e MoSold.

Tratamento para variáveis numéricas

Separadas as colunas numéricas das colunas categóricas, o momento é de avaliar a assimetria (skewness) das variáveis, deixando-as o mais simétricas possível.

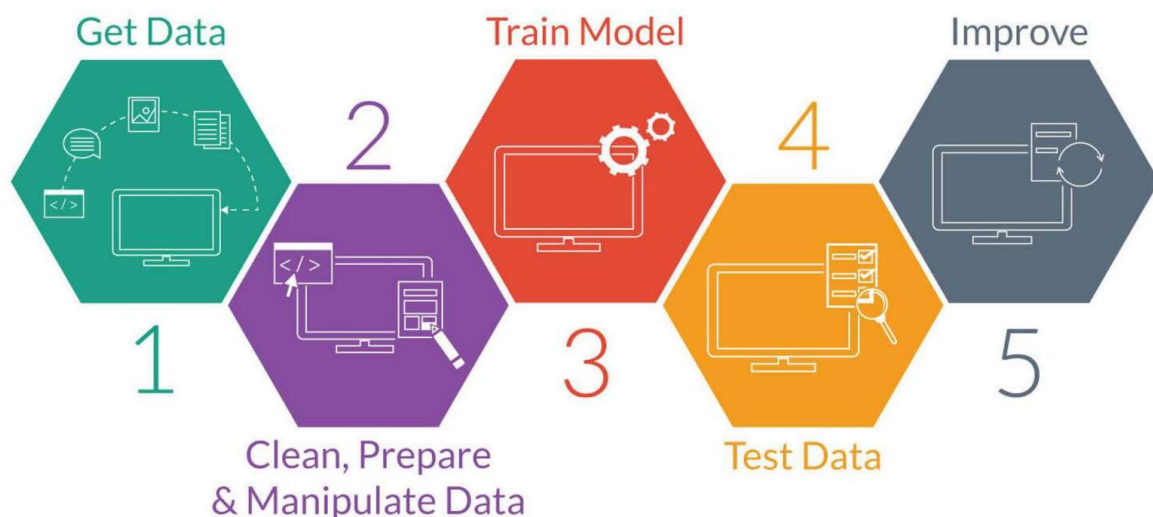
As colunas cuja skewness foi maior que 0,5 foram transformadas para escala logarítmica, a fim de melhorar a assimetria.

Tratamento para variáveis categóricas

O tratamento dado às variáveis categóricas foi de criar indicadores booleanos para as colunas, separando-as utilizando a função `get_dummies` do pandas.

Com isso, houve um aumento notório no número de variáveis do problema. No início tínhamos, contando todas as variáveis (inclusive Utilities e SalePrice), um total de 81 colunas no dataset. Após as devidas remoções e a aplicação do `get_dummies` nas variáveis categóricas, saltamos para um problema de 337 variáveis!

Treinando modelos, realizando testes e submetendo resultados para o Kaggle



A próxima etapa do workflow de aprendizado supervisionado é treinar modelos para fazer previsões e em seguida realizar testes.



Foram utilizados os modelos: Lasso, ElasticNet, KernelRidge, Gradient Boosting, XGBoost e LightGBM. O validador, nesse caso, foi o próprio validador de erro do Kaggle, que nos é dado ao submetermos um arquivo.

Lasso

O primeiro regressor testado foi o Lasso. Lasso é um modelo que é extremamente sensível a outliers. Em um primeiro momento foram removidos alguns outliers, mas não há garantias de que não existam outros.

Portanto, foram utilizadas as funções `make_pipeline` e `RobustScaler` da biblioteca `scikit learn`, na hora da instanciação do modelo, para torná-lo mais robusto.

O arquivo de saída gerado foi submetido ao Kaggle, com um score de 0.12648, rendendo no momento a posição 1326 do ranking.

1326	new	Lucas de Paula		0.12648	1	1d
Your Best Entry 						
Your submission scored 0.12648, which is not an improvement of your best score. Keep trying!						

ElasticNet

ElasticNet foi o segundo regressor utilizado. Também foram utilizadas as funções `make_pipeline` e `RobustScaler` por sua suscetibilidade a outliers. Contudo, o score gerado ficou longe do Lasso, pontuando 0,14146.

KernelRidge

O terceiro regressor foi o Kernel Ridge. As configurações dos seus parâmetros foram colhidas após uma pesquisa nos kernels do Kaggle, porém os resultados não foram bons e conseguiu ser pior que o ElasticNet, com um score de 0,16792.

Gradient Boosting

Gradient Boosting Regressor foi o quarto regressor testado, e seu resultado foi superior ao Lasso, com um score de 0,12524 talvez graças aos parâmetros escolhidos na hora de sua instanciação. Os parâmetros também vieram de um kernel do Kaggle, que será referenciado ao fim deste documento. Ao final do cálculo do score, houve um salto para a posição 1233 do ranking.

XGBoost

O quinto modelo, XGBoost, teve seu score avaliado em 0,12688, o que não foi uma melhora se comparado ao Gradient Boosting.


LightGBM

Com o sexto e último modelo testado, obtivemos o melhor score de todos: 0,12437, gerando um novo salto de 46 posições no ranking, ficando, no momento da submissão, em 1187.

1187

new


Lucas de Paula



0.12437


6

now

Your Best Entry 

You advanced 46 places on the leaderboard!

Your submission scored 0.12437, which is an improvement of your previous score of 0.12524. Great job!

 Tweet this!

Comparações de scores obtidos

Abaixo, o compilado de todos os scores obtidos na primeira fase de testes.

Submission and Description	Public Score
lgbm_submission.csv 5 minutes ago by Lucas de Paula lgbm	0.12437
xgboost_submission.csv 7 minutes ago by Lucas de Paula XGBoost	0.12688
gboost_submission.csv 31 minutes ago by Lucas de Paula Gradient Boosting Regressor Submission	0.12524
ker_ridge_submission.csv 32 minutes ago by Lucas de Paula Kernel Ridge submission	0.16792
elastic_submission.csv 34 minutes ago by Lucas de Paula ElasticNet submission	0.14146
lasso_submission.csv 2 days ago by Lucas de Paula Simple Lasso SUBmission	0.12648

Buscando melhorias

Feitas as análises próprias, era o momento de buscar melhorar o score final. Para tal, poderiam ser feitas mudanças no pré-processamento, ou então utilizar uma técnica chamada



Stacked Models, ou empilhamento de modelos. Essa técnica envolve fazer um "wrapper" para os modelos, e realizar previsões com todos eles. Feitas as previsões, é feita uma média simples ou ponderada, que podem resultar em um score melhor.

Para esse estudo, foi realizada a técnica de stacked models na tentativa de encontrar um score melhor.

Média simples dos quatro melhores modelos avaliados

No primeiro teste foram utilizados os quatro melhores modelos mais bem avaliados nos testes realizados anteriormente: Lasso, Gradient Boost, XGradiente Boost e LightGBM. Cada modelo faz sua previsão, e o resultado final é a média aritmética das saídas de cada modelo.

Submetendo ao Kaggle para análise, foi obtido o score de 0,12106 dando um salto de 206 posições para o 982 lugar.

982	new	Lucas de Paula		0.12106	7	now
Your Best Entry ↑						
You advanced 208 places on the leaderboard!						
Your submission scored 0.12106, which is an improvement of your previous score of 0.12437. Great job!						
						 Tweet this!

Média simples de todos os modelos

O segundo teste foi incluir todos os modelos já utilizados e aplicar a mesma dinâmica do item anterior. Então, além dos modelos já citados, participarão desse teste os modelos ElasticNet e Kernel Ridge.

Contudo, o resultado não foi melhor que o obtido no teste anterior. O score foi de 0,12299.

982	new	Lucas de Paula		0.12106		
Your Best Entry ↑						
Your submission scored 0.12299, which is not an improvement of your best score. Keep trying!						

Conclusão

Em todos os testes feitos, o melhor resultado foi aplicando médias entre os resultados dos quatro melhores modelos avaliados.

Para futuros trabalhos, há a sugestão de ponderar a média, aplicando pesos maiores para alguns modelos, podendo resultar assim em resultados melhores.