Data Mining Final Project

House Prices: Advanced Regression Techniques

Sahar Hajiseyednasir

Haolan(Helen) Zhang

Date: 04/30/19

Understanding the Client and their Problem

- Client Housebuyer: This client wants to find their next dream home with a reasonable price tag. They have their locations of interest ready.

- Client Houseseller: This client wants to take advantage of the features that influence a house price the most. They typically want to buy a house at a low price and invest on the features that will give the highest return.

Introduction

- 79 explanatory variables describing every aspect of residential homes
- 2 datasets: train and test sets

- Goal: predicting the final price for each house using advanced regression techniques.

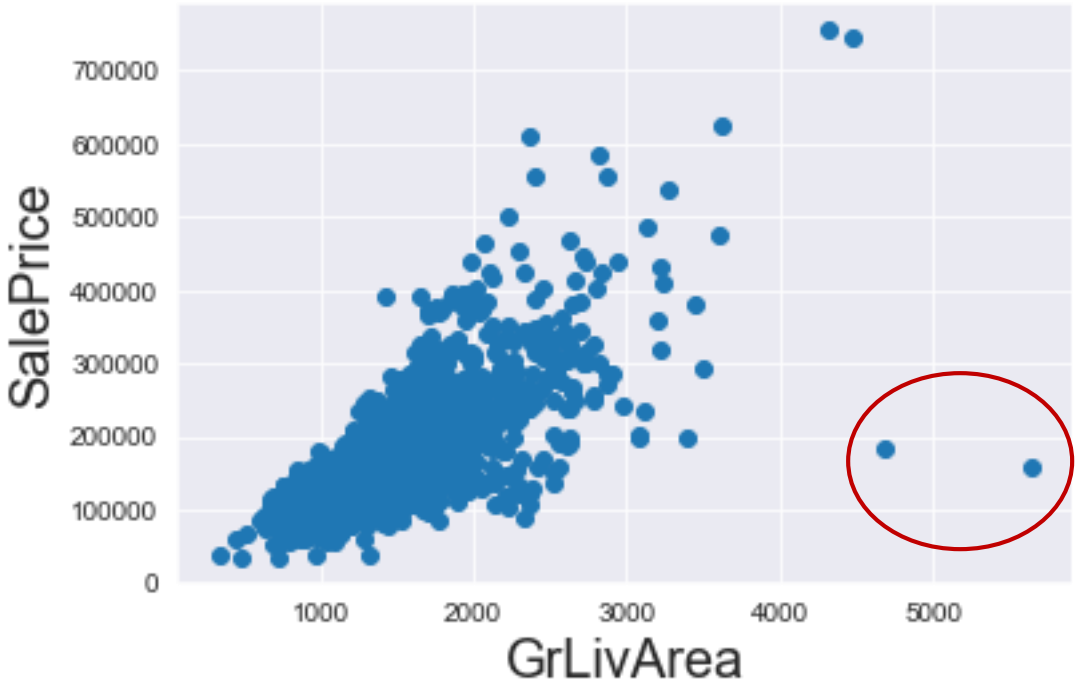
- Data: a Kaggle competition, based on property data in Ames, Iowa from 2006 and 2010.

- Evaluation: Root-Mean-Square-Error (RMSE) (the log price is to reduce the impact of biased higher price).

What is next? Data Preprocessing

- Data exploration
- Missing Value Imputation
- Data Type Transformation

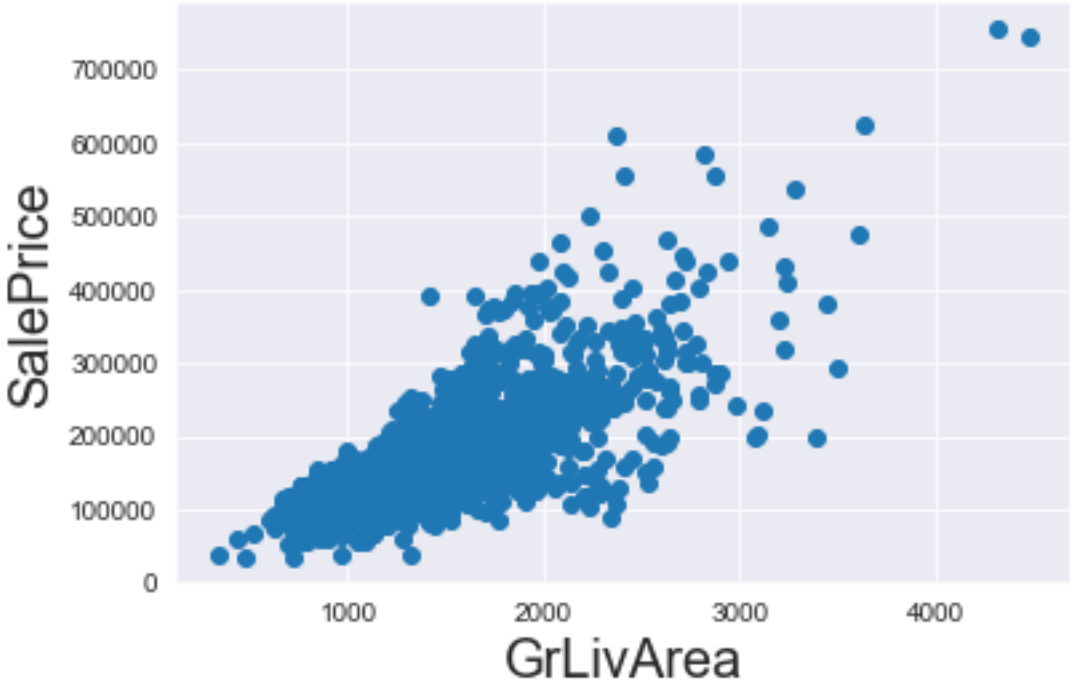
Remove Outliers



It can be seen there are two outliers for GrLivArea variable

Remove Outliers

```
data_train = data_train.drop(data_train[(data_train['GrLivArea']>4000) & (data_train['SalePrice']<300000)].index)
```



The outliers are gone using the code above. This is the way that all the variables have be checked to see if there are outliers that we need to remove.

Analysis of “SalePrice”

```
data_train['SalePrice'].describe()
```

```
count      1458.000000
mean     180932.919067
std       79495.055285
min       34900.000000
25%      129925.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

The “SalePrice” variable in the test set is the label of the dataset so it needs to be analyzed.

Analysis of “SalePrice”

The black line shows the Normal distribution. The “SalePrice” distribution is skewed to the right a little bit according to our Normal distribution graph.

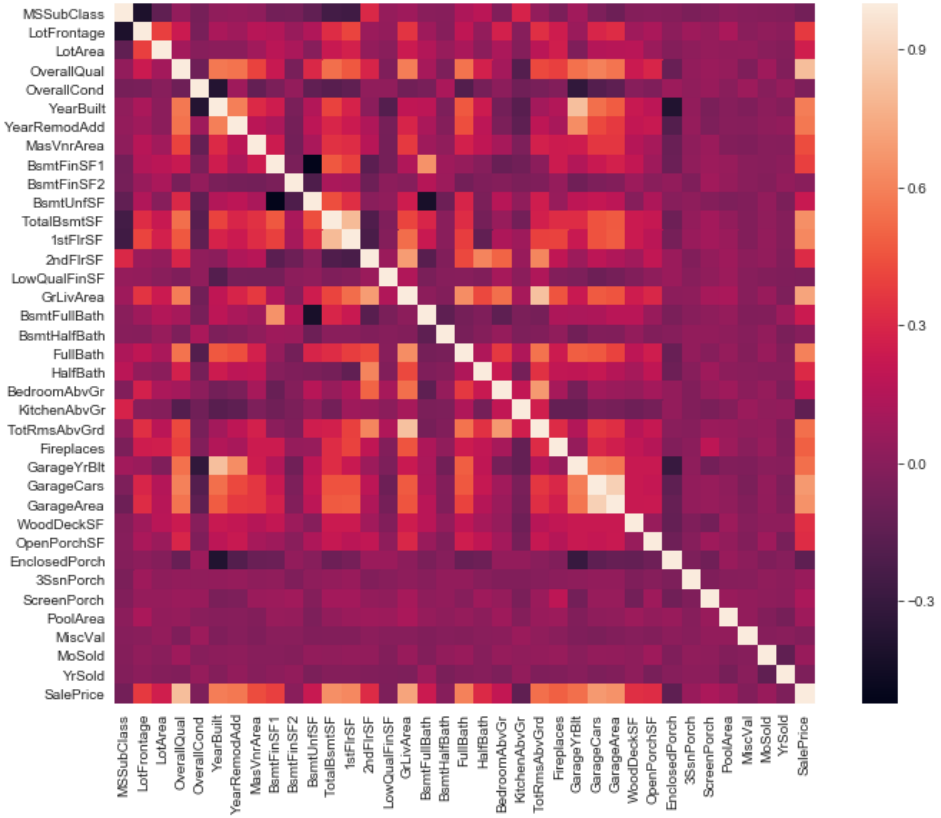
Analysis of “SalePrice”

In order to transfer the data set into normal distribution
In order that linear regression model can be used to fit the data, we used of log function.

For the log function, log(x), x cannot be 0, so we use log1p, which equals to log(x+1).

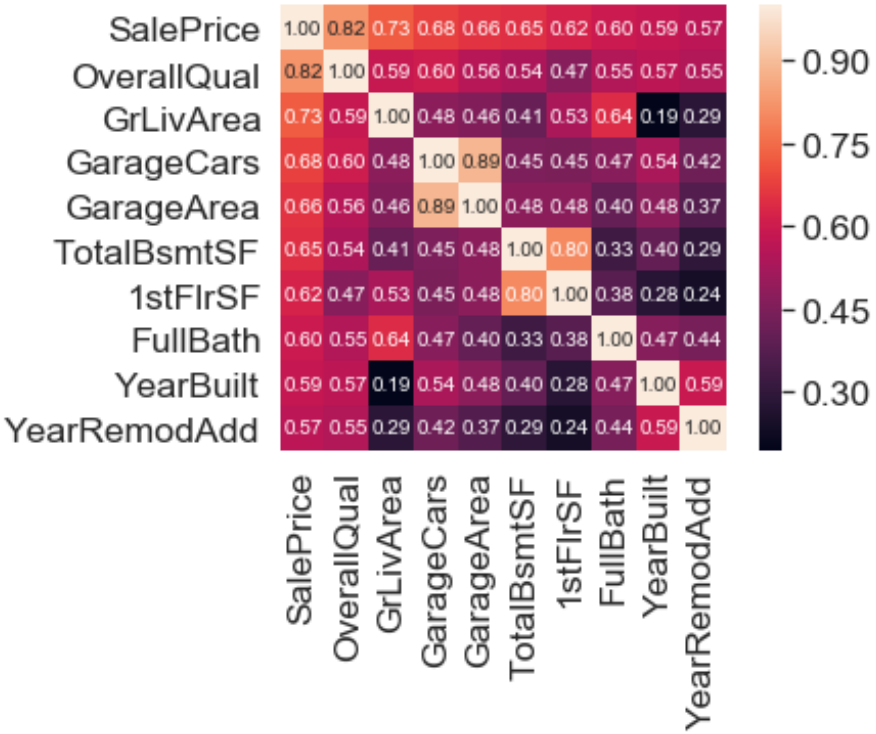
correlation matrix to see correlation between features and SalePrice

It can be seen that in the “SalePrice” column the ligher colors squares have the most correlations.



Ten most correlated features

It can be seen that in the “SalePrice” column the ligher colors squares have the most correlations.



Checking for missing values in both train and test sets

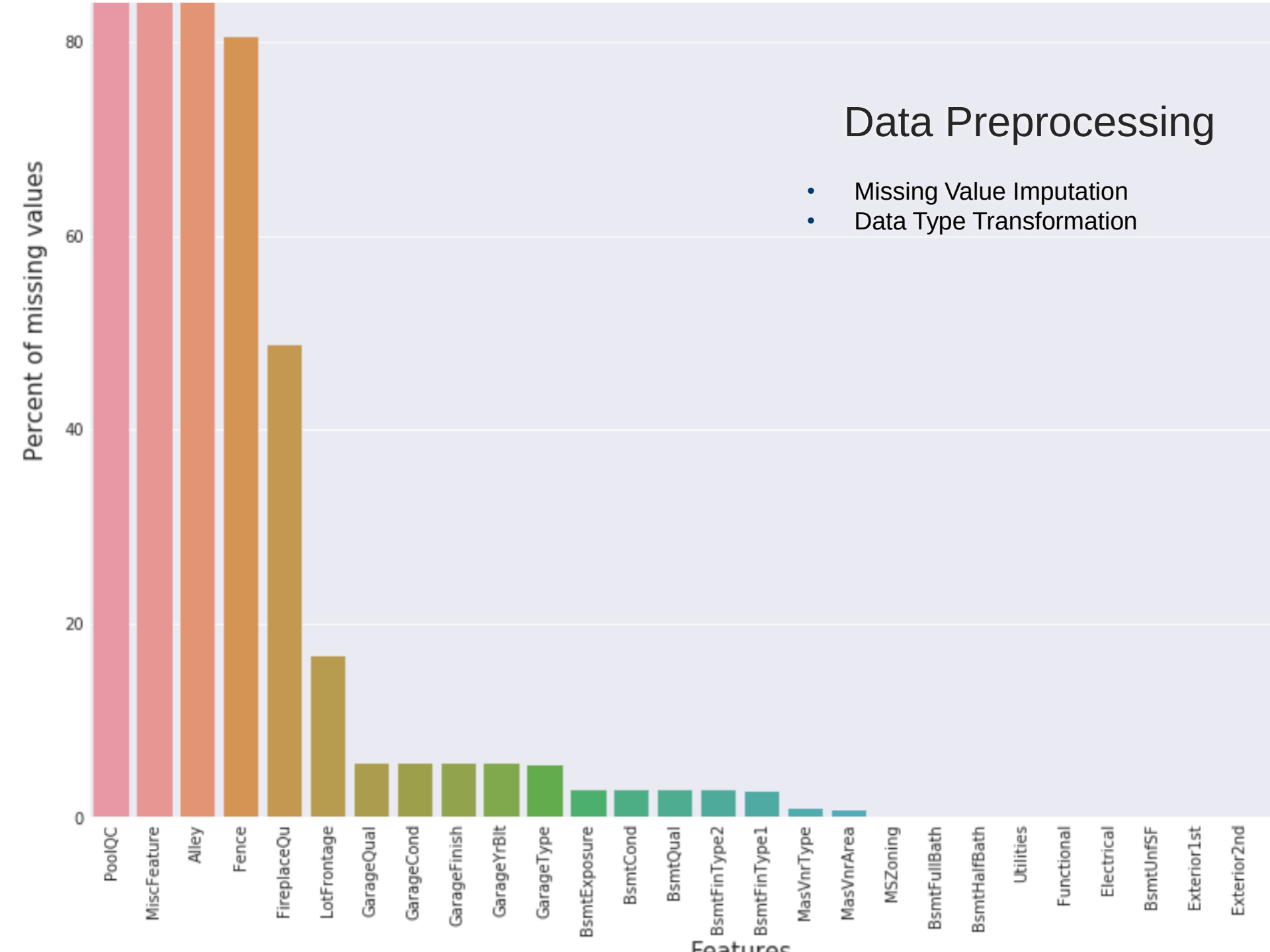
	count	ratio
PoolQC	2908	0.996915
MiscFeature	2812	0.964004
Alley	2719	0.932122
Fence	2346	0.804251
FireplaceQu	1420	0.486802
LotFrontage	486	0.166610
GarageFinish	159	0.054508
GarageQual	159	0.054508
GarageYrBlt	159	0.054508
GarageCond	159	0.054508
GarageType	157	0.053822
BsmtCond	82	0.028111
BsmtExposure	82	0.028111
BsmtQual	81	0.027768
BsmtFinType2	80	0.027425
BsmtFinType1	79	0.027083
MasVnrType	24	0.008228
MasVnrArea	23	0.007885
MSZoning	4	0.001371
BsmtHalfBath	2	0.000686
Utilities	2	0.000686
Functional	2	0.000686
BsmtFullBath	2	0.000686
Electrical	1	0.000343
Exterior2nd	1	0.000343
KitchenQual	1	0.000343
GarageCars	1	0.000343
Exterior1st	1	0.000343
GarageArea	1	0.000343
TotalBsmtSF	1	0.000343
BsmtUnfSF	1	0.000343
BsmtFinSF2	1	0.000343
BsmtFinSF1	1	0.000343
SaleType	1	0.000343
Condition2	0	0.000000

We had different approaches toward missing values:

- For “LotFrontage” variable we calculated the median for missing values
- For “Electrical”, “Exterior2nd”, “Exterior1st”, “SaleType”, “GarageCars”, “BsmtFinSF1”, “BsmtFinSF2”, “BsmtUnfSF”, “GarageCars”, “TotalBsmtSF”, “GarageArea” variables, because the number of missing values are low we just used mod.
- For the rest of them we put “0” for the missing values.

Data Preprocessing

- Missing Value Imputation
- Data Type Transformation



Checking for missing values in both train and test sets

	count	ratio
YrSold_2010	0	0.0
Exterior1st_CBlock	0	0.0
Exterior2nd_BrkFace	0	0.0
Exterior2nd_Brk Cmn	0	0.0
Exterior2nd_AsphShn	0	0.0
Exterior2nd_AsbShng	0	0.0
Exterior1st_WdShing	0	0.0
Exterior1st_Wd Sdng	0	0.0
Exterior1st_VinylSd	0	0.0
Exterior1st_Stucco	0	0.0
Exterior1st_Stone	0	0.0
Exterior1st_Plywood	0	0.0
Exterior1st_MetalSd	0	0.0
Exterior1st_ImStucc	0	0.0
Exterior1st_HdBoard	0	0.0
Exterior1st_CemntBd	0	0.0
Exterior1st_BrkFace	0	0.0
Exterior2nd_CmentBd	0	0.0
Exterior1st_BrkComm	0	0.0
Exterior1st_AsphShn	0	0.0
Exterior1st_AsbShng	0	0.0
Electrical_SBrkr	0	0.0
Electrical_Mix	0	0.0
Electrical_FuseP	0	0.0
Electrical_FuseF	0	0.0
Electrical_FuseA	0	0.0
Condition2_RRNn	0	0.0
Condition2_RRAn	0	0.0
Condition2_RRAe	0	0.0
Condition2_PosN	0	0.0
Condition2_PosA	0	0.0
Condition2_Norm	0	0.0
Exterior2nd_CBlock	0	0.0
Exterior2nd_HdBoard	0	0.0
Condition2_Artery	0	0.0

We have no missing values

What to do with Categorical values?

Some of the features' values are categorical and the kernel cannot understand them and it is not comparable.
Python pandas library has a function get_dummies() named that change the categorical variables to the numeric values.
This function makes a column for each value in each categorical column and puts 1 as that value exist for that row and puts 0 it that value does not exist for that row.

Avoiding Overfitting

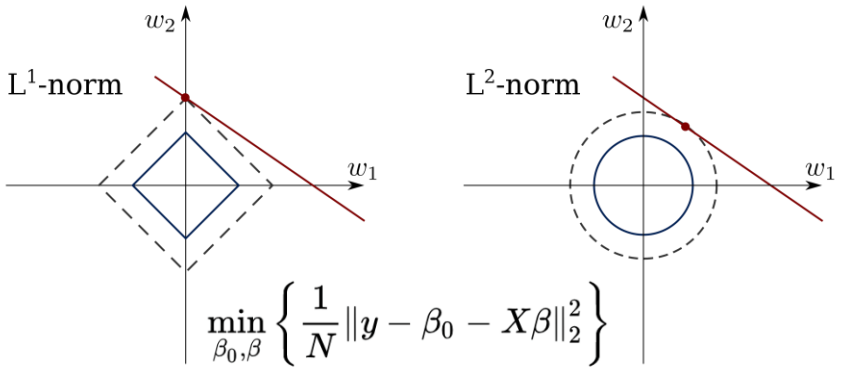
One of the effective ways that the overfitting can be avoid is to use cross-validation.
Here we used cross validation function from sklearn.model_selection library to avoid overfitting and training our model with different training sets

Feature Selection

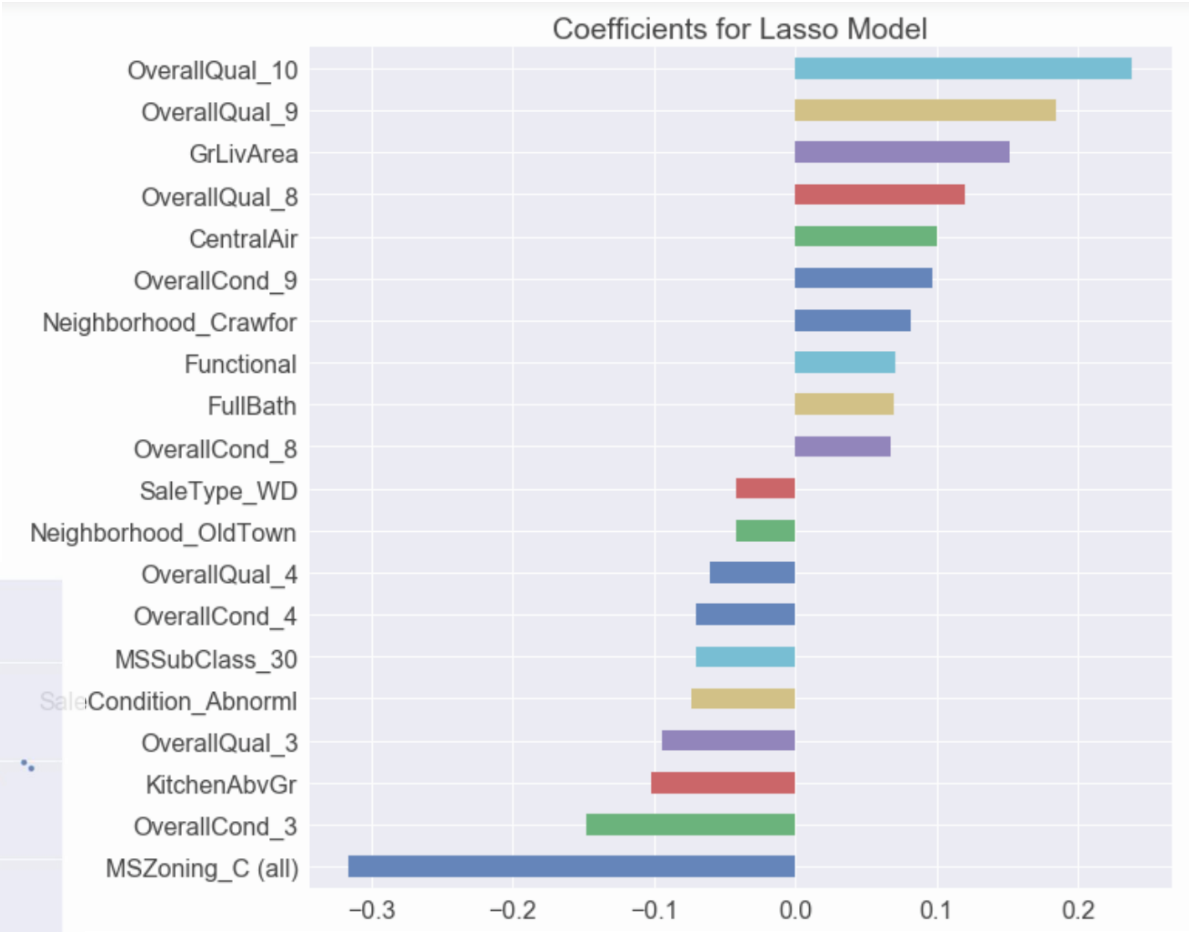
To automatically do feature selection and tune the coefficients of our variables in regression we used Lasso, GBoost, XGBoost and Kernel ridge regression Methods.

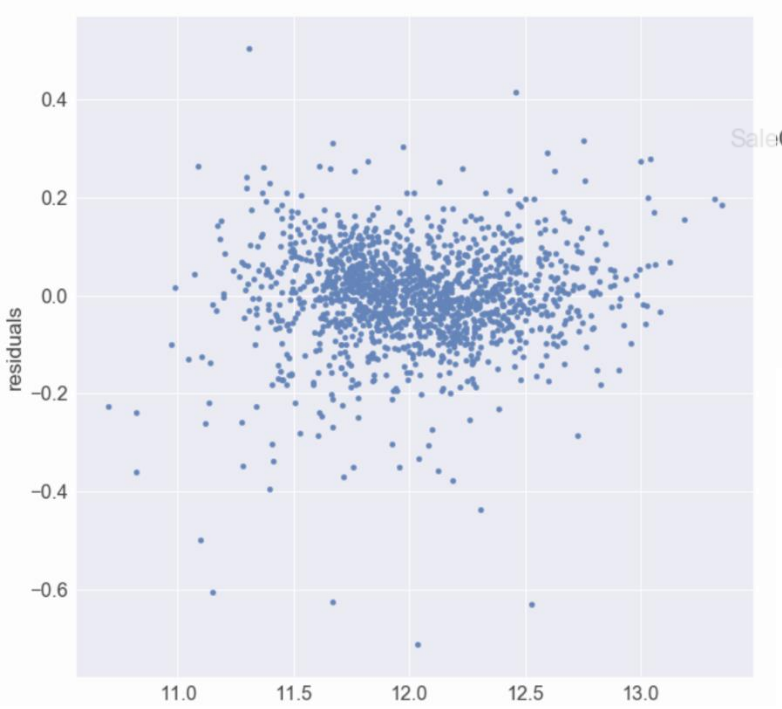
LASSO

In statistics and machine learning, lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a _regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical method it produces.



LASSO





The Error: 0.1131 (0.0071)

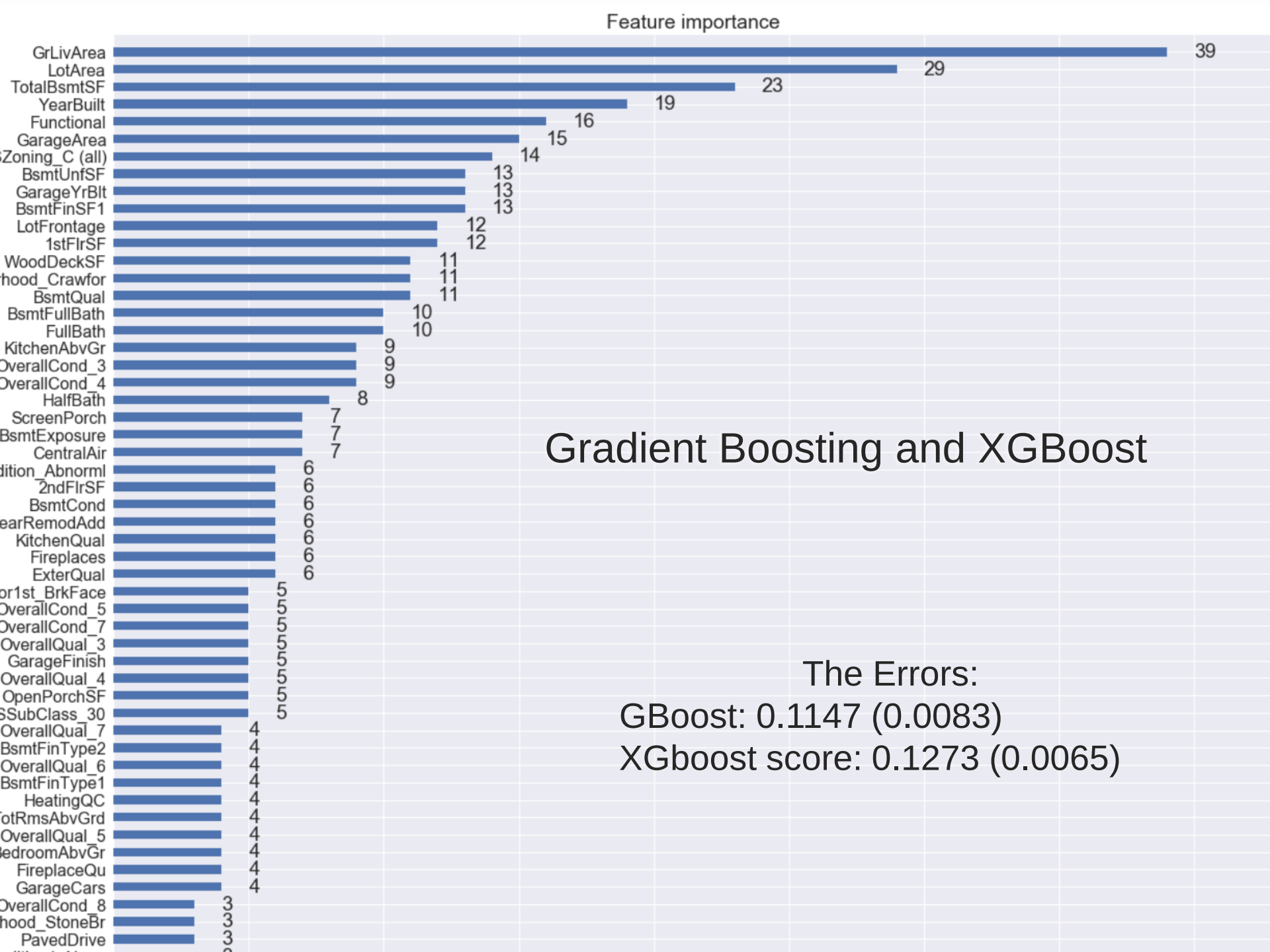
Gradient Boosting

Gradient boosting is a machine learning technique
for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decisions tree. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

XGBOOST

- XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.
- The beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

- It stands for Extreme Gradient Boosting; it is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model. It employs a number of nifty tricks that make it exceptionally successful, particularly with structured data.



Gradient Boosting and XGBoost

The Errors:

GBoost: 0.1147 (0.0083)

XGboost score: 0.1273 (0.0065)

Kernel Ridge Regression

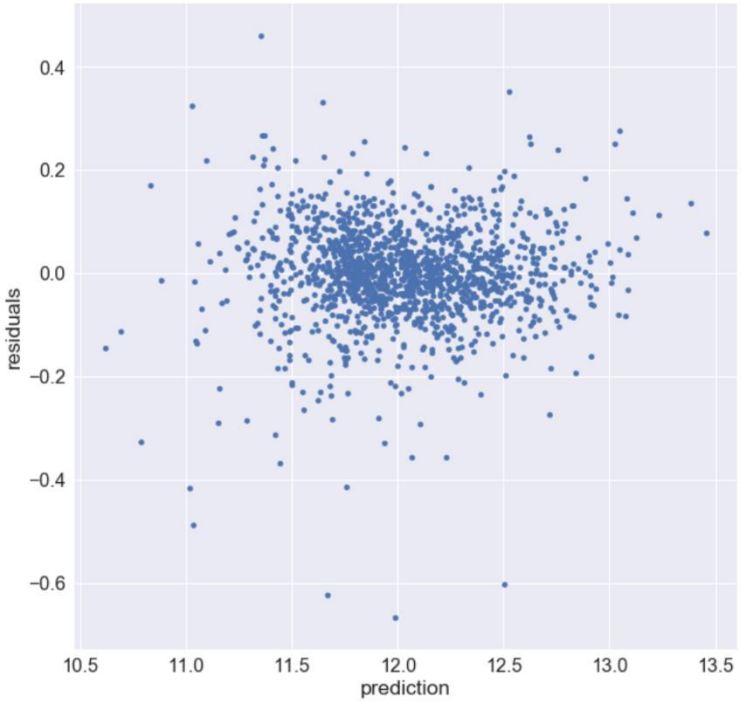
- Kernel ridge regression is a non-parametric form of ridge regression. The aim is to learn a function in the space induced by the respective kernel k by minimizing a squared loss with a squared norm regularization term.
- The solution can be written in closed form as:

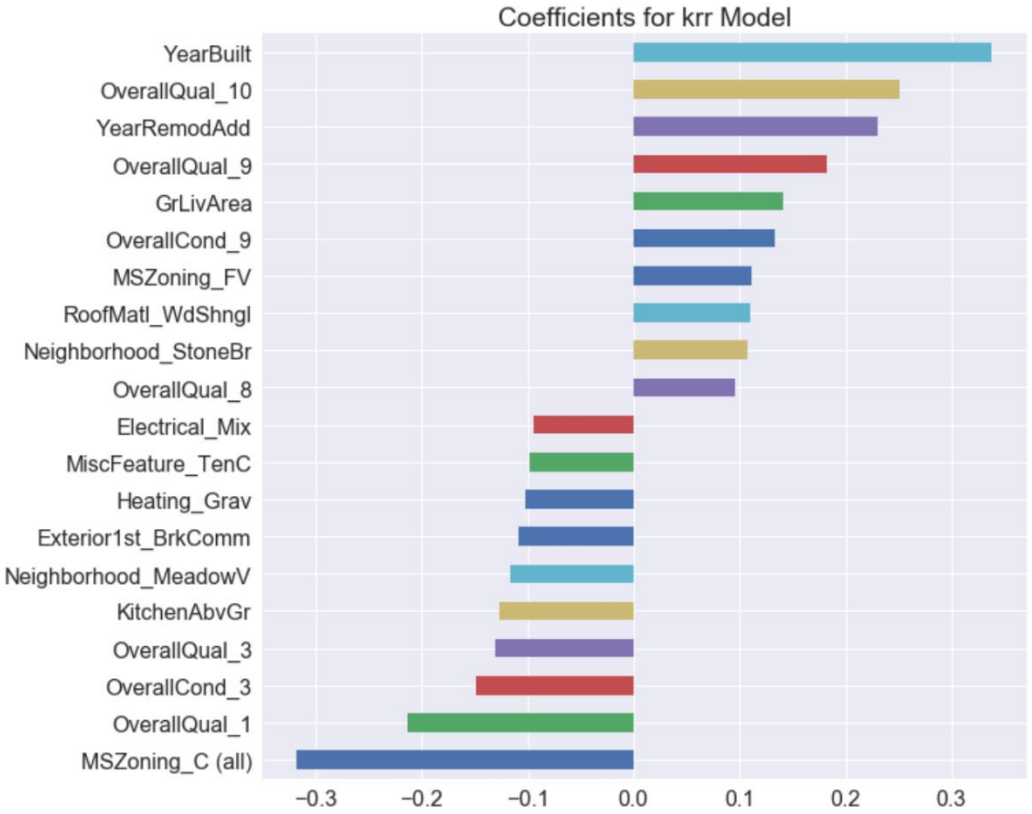
$$\alpha = (\mathbf{K} + \tau\mathbf{I})^{-1}\mathbf{y}$$

- where K is the kernel matrix and αα is the vector of weights in the space induced by the kernel. The learned function can then be evaluated as

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i).$$

KRR





The Error: 0.1163 (0.0056)

Simple Stacking

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model.

```
avg = Averagemodel(mod = (lasso,GBoost,model_xgb,KRR))
score = rmsle_cv(avg)
print(score.mean())
```

```
0.11134820070900386
```

Averaged base models score:
0.1113482

It seems even the simplest stacking approach really improve the score!!

Simple Stacking

After a few experiments, we finally choose LASSO, GBM AND KRR as models

```
avg = Averagemodel(mod = (lasso,GBoost,KRR))
score = rmsle_cv(avg)
print(score.mean())
```

```
0.11057040785293856
```

	A	B
1	Id	SalePrice
2	1461	122183.878
3	1462	160532.617
4	1463	190909.737
5	1464	198970.319
6	1465	197171.35
7	1466	176630.804
8	1467	179998.684
9	1468	162905.988
10	1469	182108.464
11	1470	124520.499
12	1471	194687.125
13	1472	96078.6523
14	1473	98882.8579
15	1474	146078.716
16	1475	107028.748
17	1476	370502.023
18	1477	244267.241
19	1478	287278.874
20	1479	283421.488
21	1480	495604.016
22	1481	315991.889
23	1482	208146.357
24	1483	182364.427
25	1484	163369.088
26	1485	184612.622
27	1486	194643.243
28	1487	342085.316

1616 **helenbb** 0.12747 3 now

Your Best Entry ↑

You advanced 32 places on the leaderboard!

Your submission scored 0.12747, which is an improvement of your previous score of 0.12802. Great job!

Tweet this!