Machine Learning Engineer Nanodegree

Capstone Project

Armen Donigian
December 6th, 2016

I. Definition

Project Overview

This project is in response to a Kaggle competition to build a predictive model to determine the sale price of a property within the Ames, Iowa region. See **House Prices: Advanced Regression Techniques** (https://www.kaggle.com/c/house-prices-advanced-regression-techniques) for further details.

Besides being a home, real estate has been a very popular investment. While most of us realize the number of bedrooms or bathrooms would influence the price of a home, it turns out that other features such as the size of the garage or proximity to a particular landmark may influence the price just as much.

The gain from an investment (such as a house in our case) is significantly impacted by the purchase price, which is why the answer for this competition matters.

The competition began on August 30, 2016 while the final submission must be made by March 1st, 2017.

The source code for this project is available via a Jupyter notebook located here (./Capstone-Project-Ames-Housing-Data.ipynb).

Problem Statement

With 79 explanatory variables describing (almost) every aspect of residential homes in **Ames, Iowa**, this competition challenges you to predict the final price of each home.

Here's an end-to-end workflow for building a regressor using scikit-learn.

1. Problem Definition (Ames house price data).
   - Experimental Design: identify data sources, formats, data dictionary, features and target.
2. Loading the Dataset
   - Load and pre-process the data into a representation which is ready for model training
3. Exploratory Data Analysis
   - Gather insights by using exploratory methods (univariate feature distributions, skew and correlated attributes to name a few)
4. Evaluate Algorithms
5. Evaluate Algorithms with Standardization
6. Algorithm Tuning
   - Use GridSearch to search & tune hyper-parameters

7. Ensemble Methods (such as Bagging and Boosting, Gradient Boosting looked good).
8. Finalize Model (use all training data and confirm using validation dataset).

Metrics

Given this is a Kaggle competition, the evaluation metric has been specified as the **Root-Mean-Squared-Error (RMSE)** (https://www.kaggle.com/c/outbrain-click-prediction/details/evaluation).

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (x_{1,t} - x_{2,t})^2}{n}}.$$

The root mean squared error metric intutively is suitable for house price prediction since:

- a regressor outputs numerical predictions which won't exactly match the target values in the data set (due to irreducible error)
- can use the outputs of the regressor to measure difference between values in training dataset vs predicted values by my model

II. Analysis

Data Exploration

The following files have been provided for this competition:

File	Dimensions (rows, columns)	Text or Numeric
train	1459, 81	Alpha Numeric
test (public hold out)	1459, 80	Alpha Numeric

There are 79 explanatory variables describing (almost) every aspect of residential homes in Ames.

```
SalePrice - the property's sale price in dollars. This is the target variabl
e that you're trying to predict.
MSSubClass: The building class
MSZoning: The general zoning classification
LotFrontage: Linear feet of street connected to property
LotArea: Lot size in square feet
Street: Type of road access
Alley: Type of alley access
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 main road or railroad
Condition2: Proximity to main road or railroad (if a second is present)
BldgType: Type of dwelling
HouseStyle: Style of dwelling
OverallQual: Overall material and finish quality
OverallCond: Overall condition rating
YearBuilt: Original construction date
YearRemodAdd: Remodel date
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: Exterior material quality
ExterCond: Present condition of the material on the exterior
Foundation: Type of foundation
BsmtQual: Height of the basement
BsmtCond: General condition of the basement
BsmtExposure: Walkout or garden level basement walls
BsmtFinType1: Quality of basement finished area
BsmtFinSF1: Type 1 finished square feet
BsmtFinType2: Quality of second finished area (if present)
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: Number of bedrooms above basement level
Kitchen: Number of kitchens
KitchenQual: Kitchen quality
TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
Functional: Home functionality rating
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
YrSold: Year Sold
SaleType: Type of sale
SaleCondition: Condition of sale
```

Kaggle maintains a portion of the data for each competition as a hold out set which is used to evaluate the generalizability of our model to new unseen data (aka private leaderboard). The Public Score is being determined from only a fraction of the test data set (25-33%).

Sample data:

```
Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape LandCont
our Utilities  \
0        1          60       RL          65.0     8450   Pave   NaN      Reg
        Lvl    AllPub
1        2          20       RL          80.0     9600   Pave   NaN      Reg
        Lvl    AllPub
2        3          60       RL          68.0    11250   Pave   NaN      IR1
        Lvl    AllPub
3        4          70       RL          60.0     9550   Pave   NaN      IR1
        Lvl    AllPub
4        5          60       RL          84.0    14260   Pave   NaN      IR1
        Lvl    AllPub
5        6          50       RL          85.0    14115   Pave   NaN      IR1
        Lvl    AllPub
6        7          20       RL          75.0    10084   Pave   NaN      Reg
        Lvl    AllPub
7        8          60       RL           NaN    10382   Pave   NaN      IR1
        Lvl    AllPub

 PoolArea PoolQC  Fence MiscFeature MiscVal MoSold YrSold  SaleType  SaleCon
dition  \
0       ...              0    NaN    NaN          NaN       0       2   2008
    WD         Normal
1       ...              0    NaN    NaN          NaN       0       5   2007
    WD         Normal
2       ...              0    NaN    NaN          NaN       0       9   2008
    WD         Normal
3       ...              0    NaN    NaN          NaN       0       2   2006
    WD        Abnorml
4       ...              0    NaN    NaN          NaN       0      12   2008
    WD         Normal
5       ...              0    NaN  MnPrv        Shed     700      10   2009
    WD         Normal
6       ...              0    NaN    NaN          NaN       0       8   2007
    WD         Normal
7       ...              0    NaN    NaN        Shed     350      11   2009
    WD         Normal   \


      SalePrice
0        12.248
1        12.109
2        12.317
3        11.849
4        12.429
5        11.871
6        12.635
7        12.206
8        11.775
```

Here's a summary of the missing values in the train dataset file provided:

File	# of missing
LotFrontage	259
Alley	1368
MasVnrType	8
MasVnrArea	8
BsmtQual	37
BsmtCond	37
BsmtExposure	38
BsmtFinType1	37
BsmtFinType2	38
Electrical	1
FireplaceQu	689
GarageType	81
GarageYrBlt	81
GarageFinish	81
GarageQual	81
GarageCond	81
PoolQC	1452
Fence	1178
MiscFeature	1405

Descriptive Statistics

It's important to note that Id column should not be used as a feature in our training set and will hence be removed.

The missing values (NAN) impact the results of our descriptive stats. I'll be using the imputer module from scikit-learn to impute mean values for those missing.

1459 rows of training data (test dataset same dimensions) may not be enough to build a predictive model due to it's small size, but let's see what we can do with what we have since gathering new data takes more time, money or might just not be possible.

Since MSSubClass is a type identifier for type of dwelling, it wouldn't make sense to look at the summary statistics.

OverallQual with a mean of 6.0, OverallCond with mean of 5.5 seems reasonable and nice to see the data confirm what we would intuitively expect.

Oldest property built in 1879 while most recent in 2010. Since price of a property is determined not only by where it's located but also when it's purchased. This is something we want to keep in mind since we know the real estate market is susceptible to bubbles followed by crashes.

Which brings us to our target variable SalePrice. Cheapest property sold for 39K, while most expensive 755K with a mean of 180,944. I will create candle stick plots to do outlier analysis.

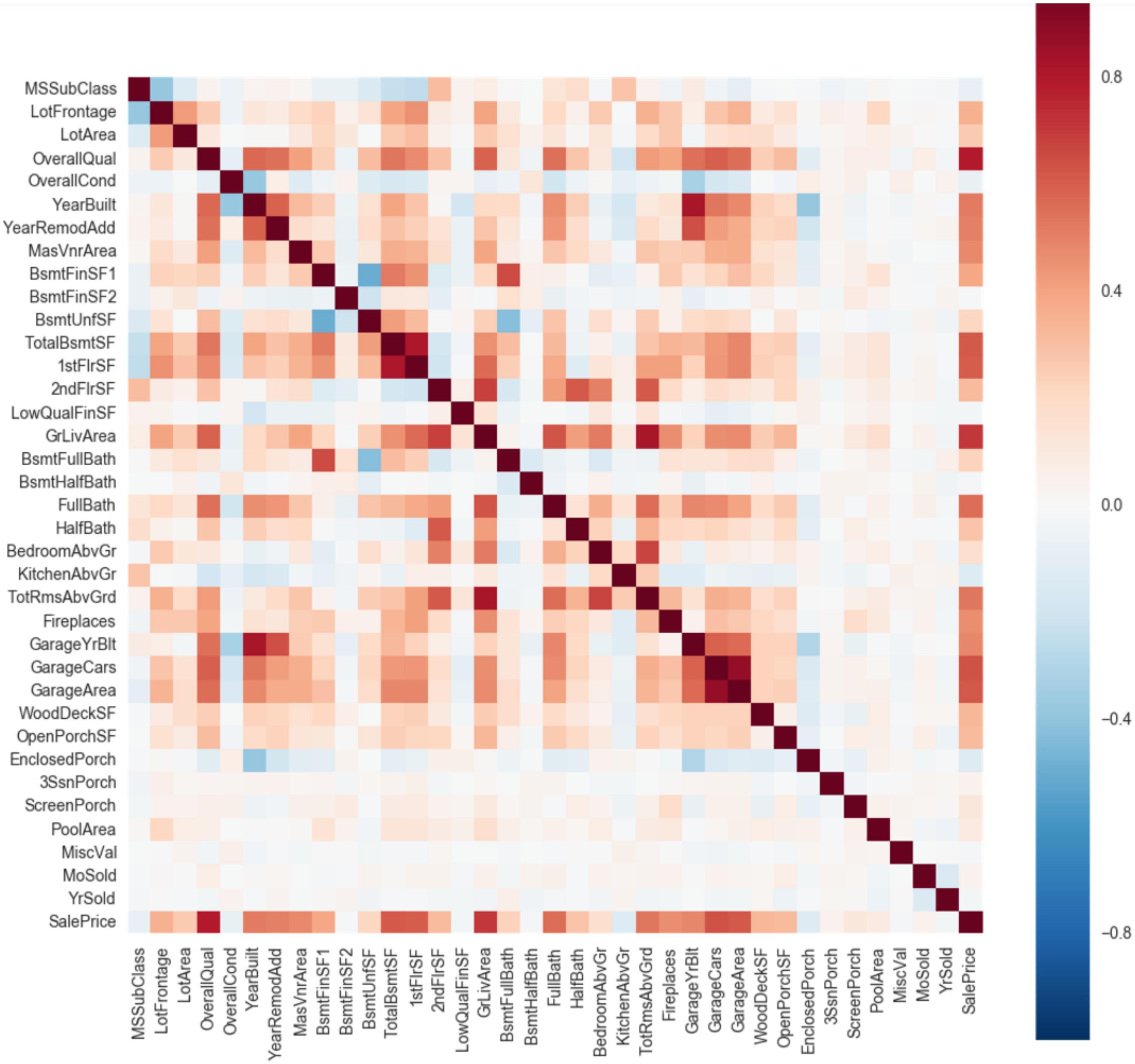
```
Id  MSSubClass  LotFrontage     LotArea  OverallQual  OverallCond  YearBuilt
  \
count  1459.000    1459.000     1200.000    1459.000     1459.000     1459.0
00   1459.000
mean    730.000      56.923       70.046   10517.225        6.100        5.5
75   1971.272
std     421.321      42.304       24.294    9984.676        1.383        1.1
13     30.213
min       1.000      20.000       21.000    1300.000        1.000        1.0
00   1872.000
25%     365.500      20.000          NaN    7549.000        5.000        5.0
00   1954.000
50%     730.000      50.000          NaN    9477.000        6.000        5.0
00   1973.000
75%    1094.500      70.000          NaN   11603.000        7.000        6.0
00   2000.000
max    1459.000     190.000      313.000  215245.000       10.000        9.0
00   2010.000

       YearRemodAdd  MasVnrArea  BsmtFinSF1     ...      WoodDeckSF  OpenPor
chSF  EnclosedPorch  \
count      1459.000    1451.000    1459.000     ...        1459.000     145
9.000       1459.000
mean       1984.879     103.757     443.375     ...          93.805       4
6.646         21.969
std          20.646     181.108     456.142     ...         124.249       6
6.276         61.137
min        1950.000       0.000       0.000     ...           0.000
 0.000          0.000
25%        1967.000         NaN       0.000     ...           0.000
 0.000          0.000
50%        1994.000         NaN     383.000     ...           0.000       2
5.000          0.000
75%        2004.000         NaN     712.000     ...         168.000       6
8.000          0.000
max        2010.000    1600.000    5644.000     ...         857.000      54
7.000        552.000

       3SsnPorch  ScreenPorch  PoolArea    MiscVal    MoSold    YrSold   Sal
ePrice
count   1459.000     1459.000  1459.000   1459.000  1459.000  1459.000    14
59.000
mean       3.412       15.071     2.761     43.519     6.322  2007.816  1809
44.103
std       29.327       55.775    40.191    496.292     2.705     1.329   794
64.918
min        0.000        0.000     0.000      0.000     1.000  2006.000   349
00.000
25%        0.000        0.000     0.000      0.000     5.000  2007.000   1299
```

```
50.000
50%        0.000        0.000     0.000      0.000     6.000  2008.000  1630
00.000
75%        0.000        0.000     0.000      0.000     8.000  2009.000  2140
00.000
max      508.000      480.000   738.000  15500.000    12.000  2010.000  7550
00.000
```

Exploratory Visualization

Correlations Between Attributes

Correlation refers to the relationship between two variables and how they may or may not change together. The most common method for calculating correlation is Pearson’s Correlation Coefficient (assumes normal distribution). A correlation of -1 or 1 shows a full negative or positive correlation respectively, while a value of 0 shows no correlation at all. Some machine learning algorithms like linear and logistic regression can suffer poor performance if there are highly correlated attributes in your dataset. Let's review all of the pairwise correlations of the attributes in your dataset.



Here are the top feature correlations:

- YearBuilt & GarageYrBlt
- BsmtFullBath & BsmtFinSF1
- TotalBsmtSF & 1stFlrSF
- TotRmsAbvGrd & GrLivArea
- GarageCars & GarageArea

There are some obvious correlations with the SalePrice (target):

- Rates the overall material and finish of the house
- Basement square footage
- Year built and remodeled
- 1st Floor square footage
- Living area square footage
- Number of full bathrooms
- Total rooms above ground
- Size of garage in car capacity

- Size of garage in square feet

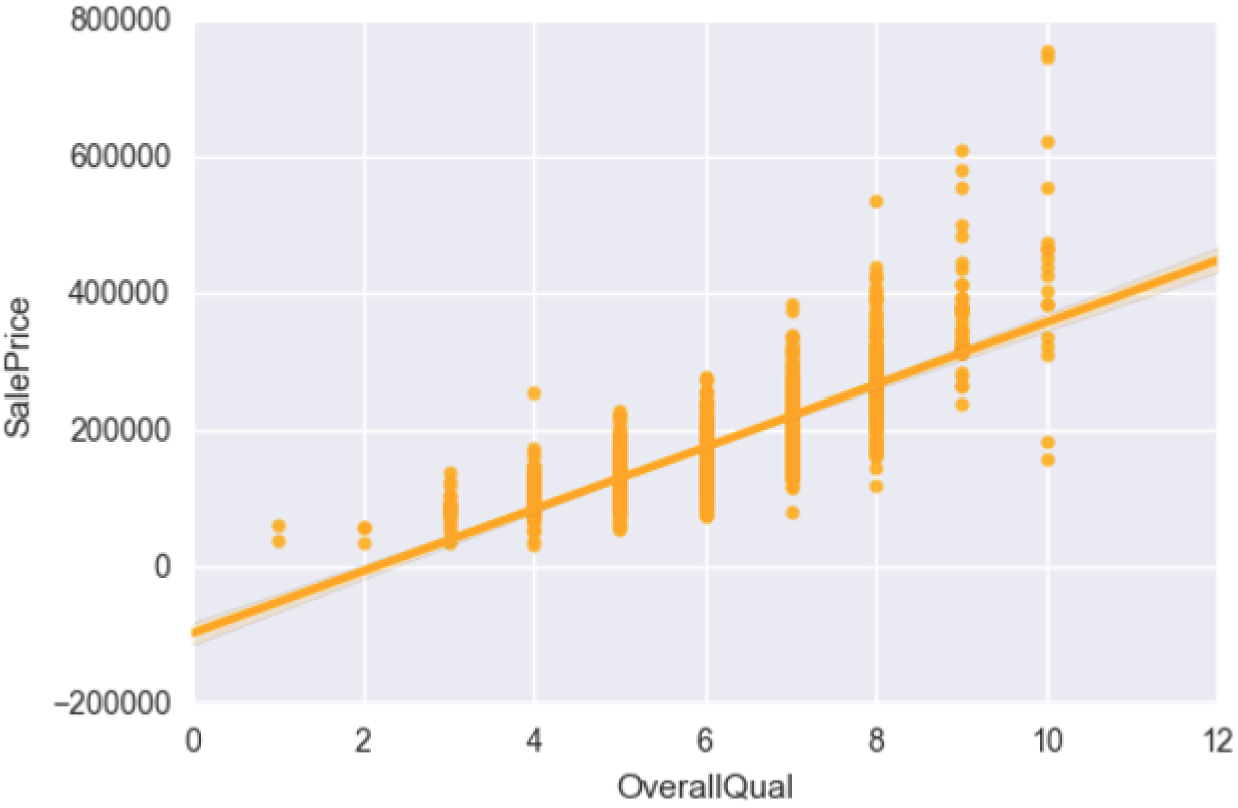
Some not so obvious features which didn't show strong correlation with target:

- Year sold (housing bubbles)
- Month sold (summer $$$)
- Kitchen quality
- Overall condition of the house
- Type of dwelling involved in the sale

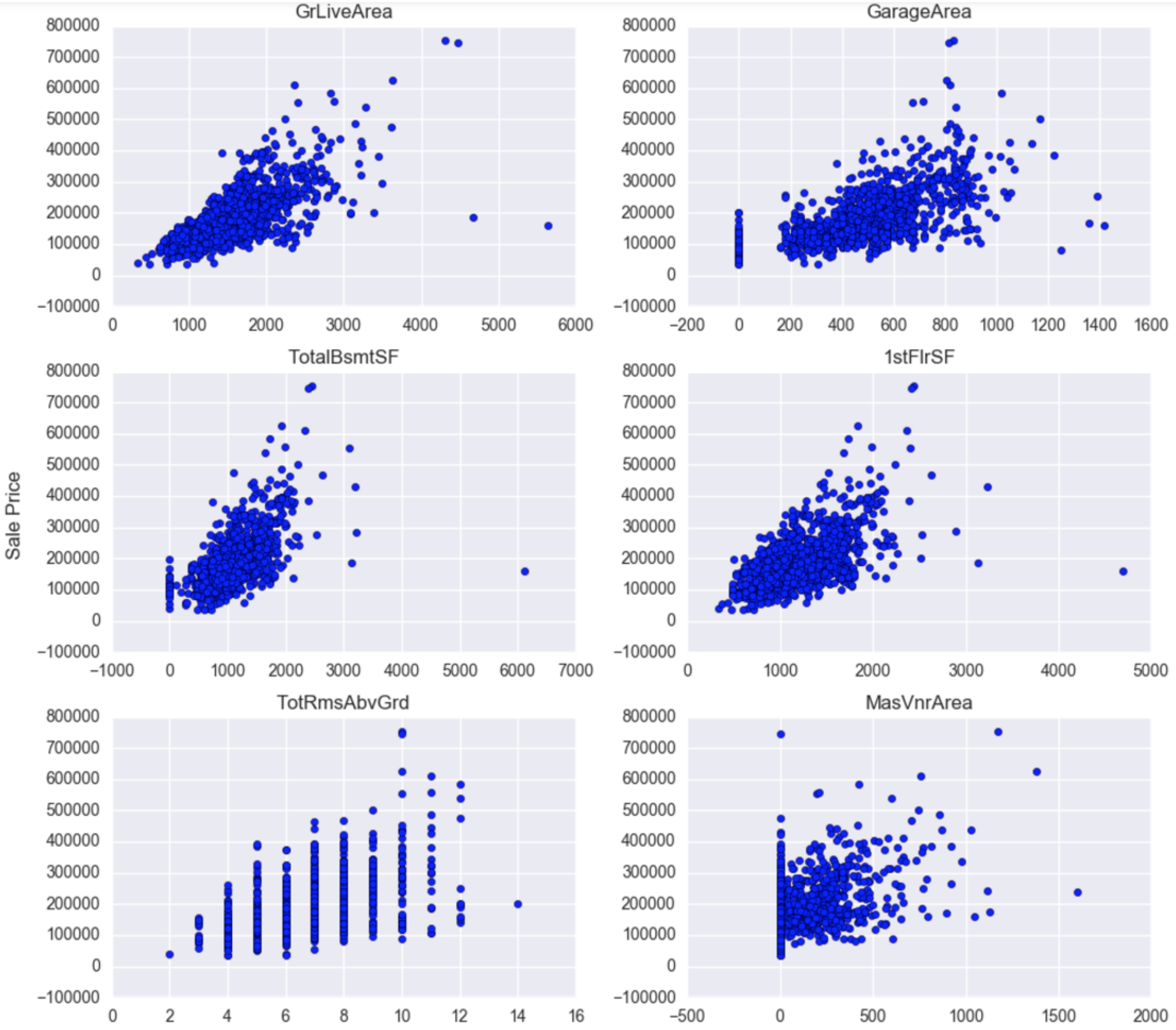
List the numerical features decendingly by their correlation with Sale Price:

```
OverallQual:     0.790971646739
GrLivArea:     0.708584256389
GarageCars:     0.640383308726
GarageArea:     0.623384913038
TotalBsmtSF:     0.613791532285
1stFlrSF:     0.605970779941
FullBath:     0.560604113108
TotRmsAbvGrd:     0.533682199367
YearBuilt:     0.522876912022
YearRemodAdd:     0.507015099034
GarageYrBlt:     0.486264369585
MasVnrArea:     0.477410802037
Fireplaces:     0.466827565809
BsmtFinSF1:     0.386782893996
LotFrontage:     0.351896380432
WoodDeckSF:     0.32888080889
2ndFlrSF:     0.319192983491
OpenPorchSF:     0.315979580344
HalfBath:     0.284626062633
LotArea:     0.263842911565
BsmtFullBath:     0.227551303799
BsmtUnfSF:     0.214280506904
BedroomAbvGr:     0.168272155796
KitchenAbvGr:     -0.135978700432
EnclosedPorch:     -0.128695108854
ScreenPorch:     0.111378177949
PoolArea:     0.0923894928294
MSSubClass:     -0.0845630196621
OverallCond:     -0.0777543847898
MoSold:     0.046400930621
3SsnPorch:     0.04455301874
YrSold:     -0.0288844993565
LowQualFinSF:     -0.0256424965435
MiscVal:     -0.0212162163774
BsmtHalfBath:     -0.016915050596
BsmtFinSF2:     -0.0109518948727
```

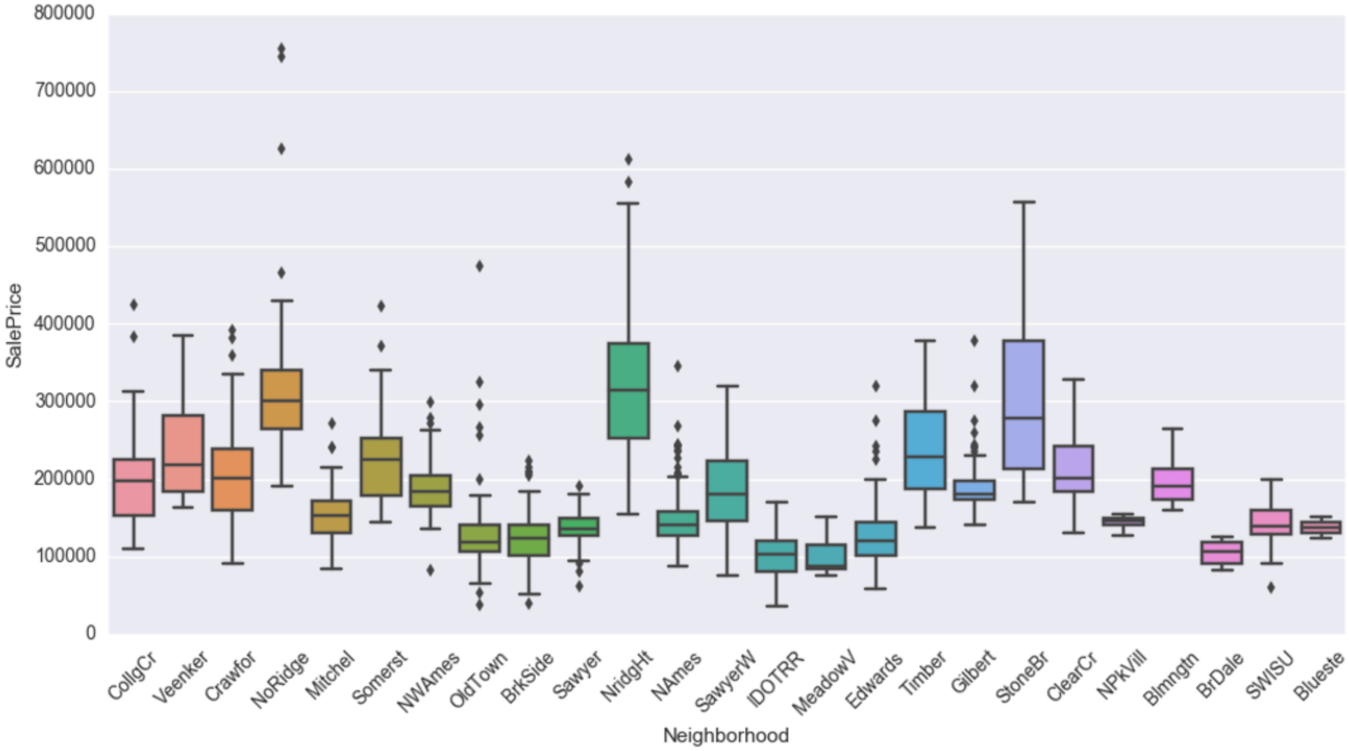
Here's an examle of the correlation between quality of property & sale price. You can clearly see the linear relationship with a positive slope.



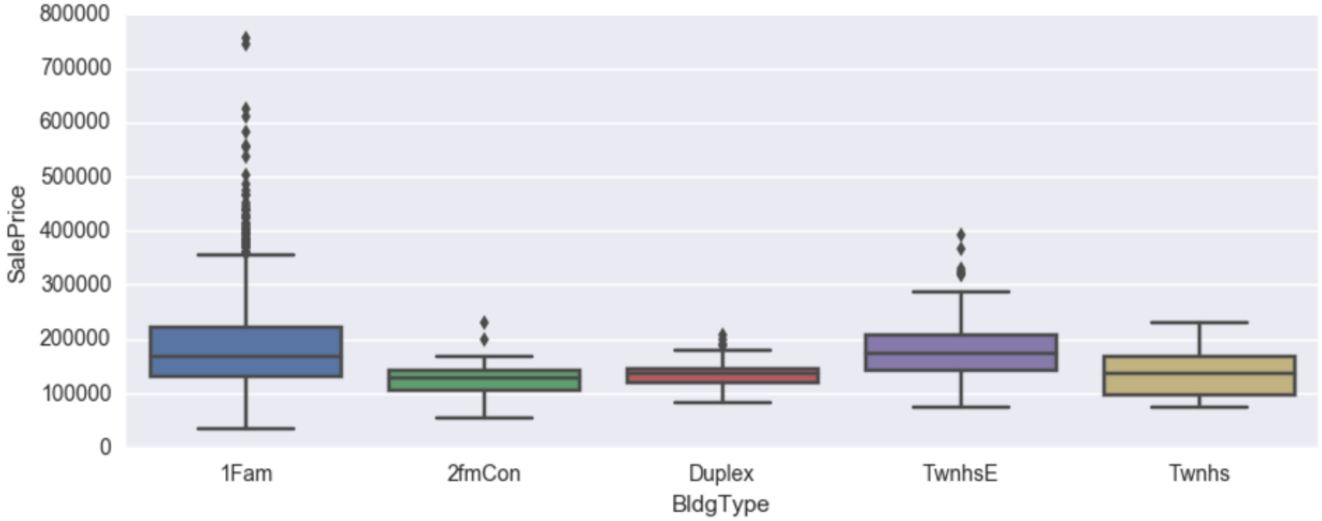
Here are a few more plots of the highest correlated features with target and the linear relationship between them:

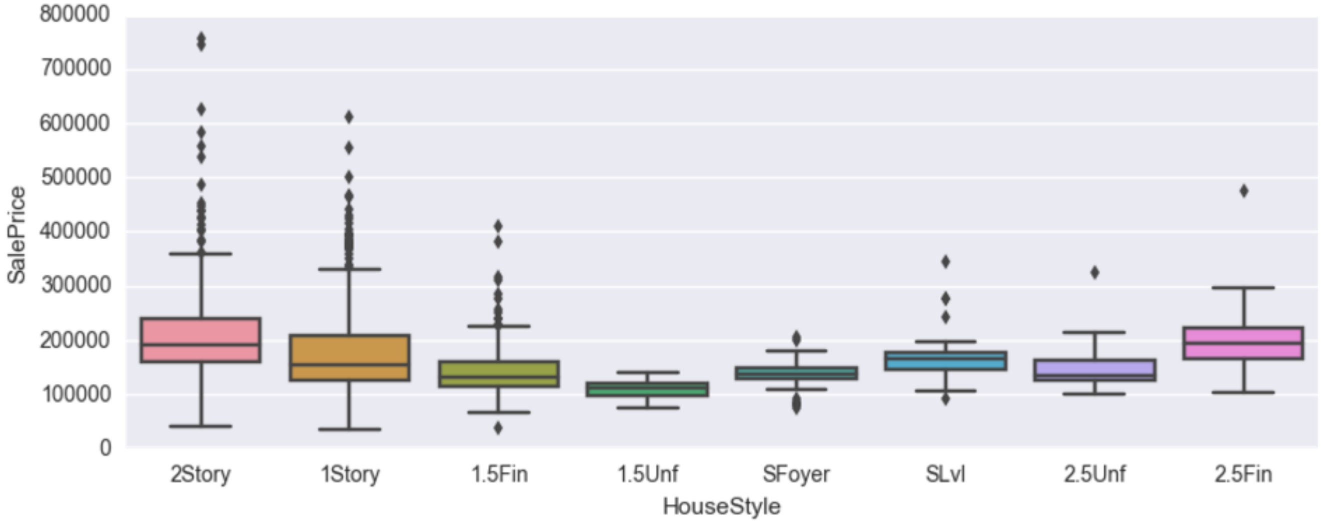


Real estate prices are strongly correlated with location (desirable vs un-desirable). Here's a candle stick plot of Neighboorhood to Sale Price:

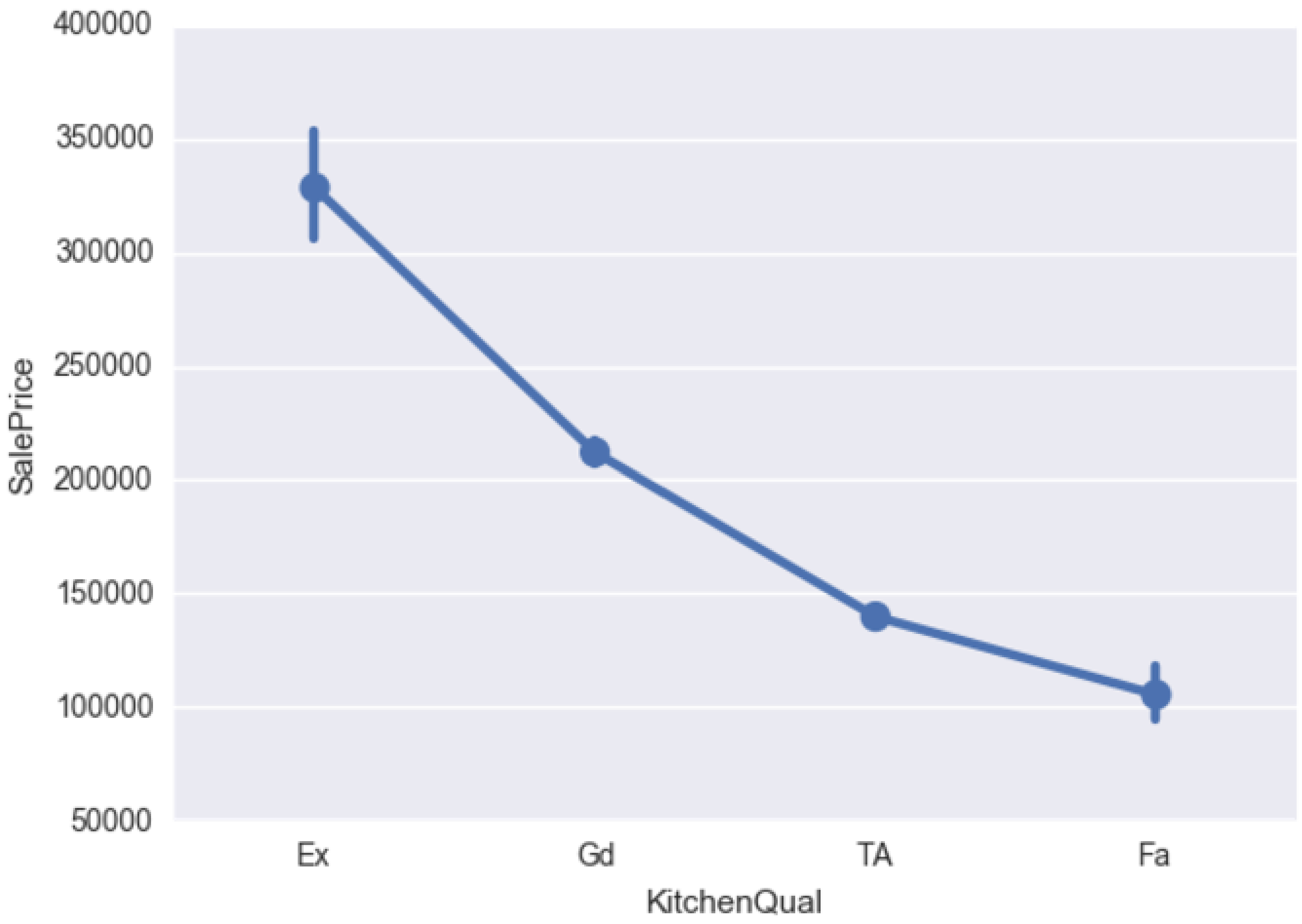


Property type to Sale Price:





From experience we know that the kitchen is an important part of making a decision to purcahse a property for alot of families. The data supports our experience:



Skew of Univariate Distributions

Skew refers to a distribution that is assumed Gaussian (normal or bell curve) that is shifted or squashed in one direction or another. Many machine learning algorithms assume a Gaussian distribution. Knowing that an attribute has a skew may allow you to perform data preparation to correct the skew and later improve the accuracy of your models.

Below is a summary of the skewness in the training dataset provided.

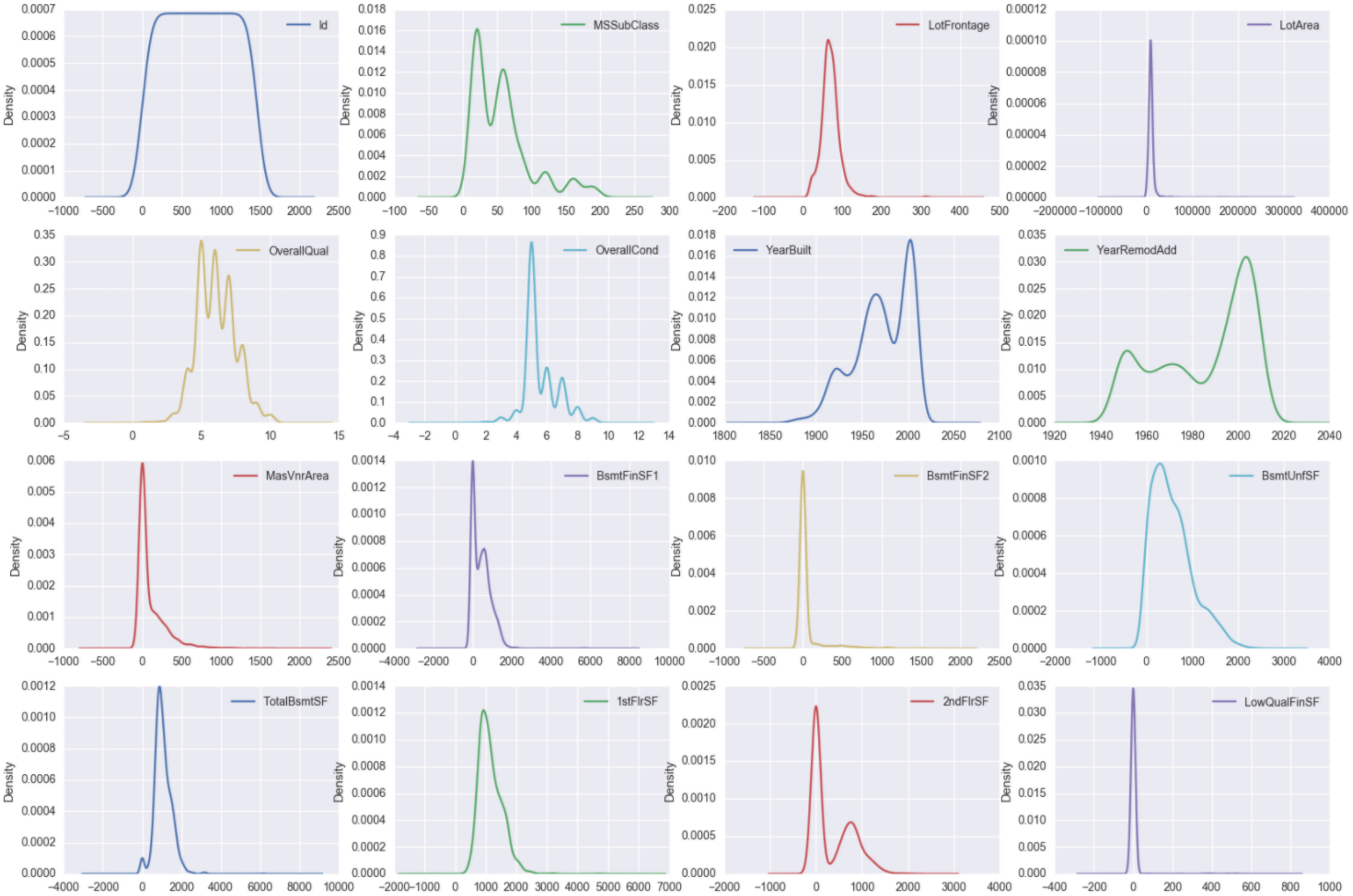
Note: Positive (right), zero show less skew and negative (left) skew. Both in & out of time datasets seem to have similar skew properties. I will be applying a log transformation to reduce the impact of skew on the overall model selection & evaluation process.

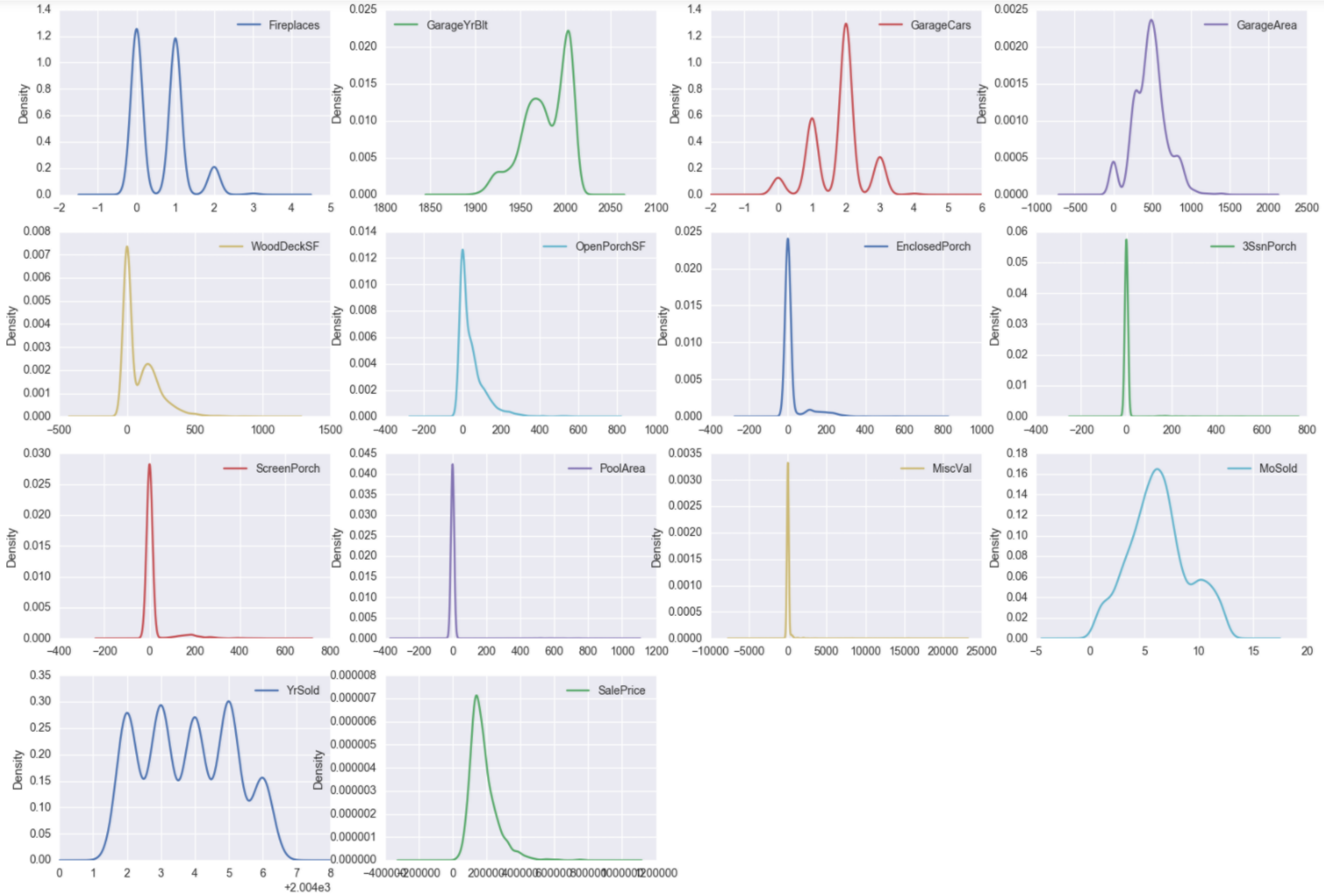
```
Id                0.000
MSSubClass        1.407
LotFrontage       2.163
LotArea          12.203
OverallQual       0.216
OverallCond       0.694
YearBuilt        -0.614
YearRemodAdd     -0.505
MasVnrArea        2.668
BsmtFinSF1        1.687
BsmtFinSF2        4.265
BsmtUnfSF         0.919
TotalBsmtSF       1.525
1stFlrSF          1.377
2ndFlrSF          0.812
LowQualFinSF      9.008
GrLivArea         1.366
BsmtFullBath      0.598
BsmtHalfBath      4.102
FullBath          0.035
HalfBath          0.678
BedroomAbvGr      0.212
KitchenAbvGr      4.487
TotRmsAbvGrd      0.676
Fireplaces        0.649
GarageYrBlt      -0.650
GarageCars       -0.344
GarageArea        0.179
WoodDeckSF        1.499
OpenPorchSF       2.364
EnclosedPorch     3.089
3SsnPorch        10.301
ScreenPorch       4.121
PoolArea         14.823
MiscVal          24.468
MoSold            0.212
YrSold            0.097
SalePrice         1.882
```

Density Distrubtion Plots

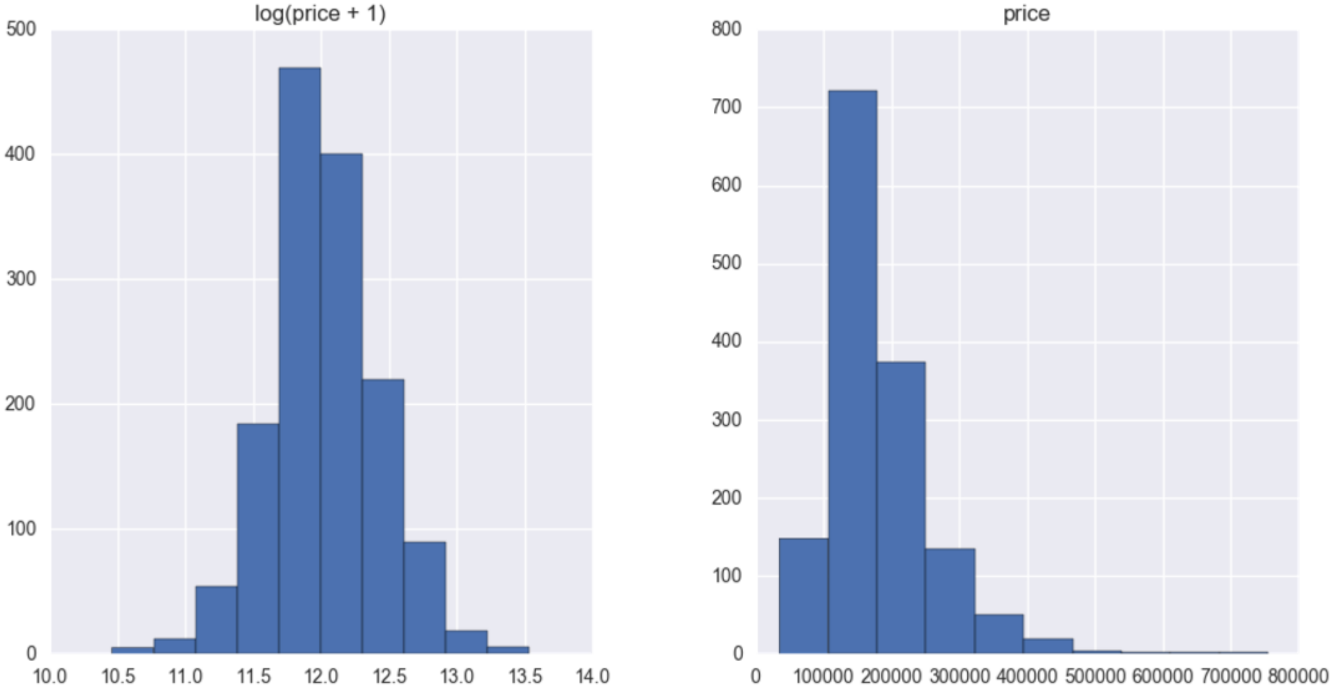
Density plots are another way of getting a quick idea of the distribution of each attribute. The plots look like an abstracted histogram with a smooth curve drawn through the top of each bin, much like your eye tried to do with the histograms.

Notice that some features like YearSold and Fireplaces are not normally distributed...





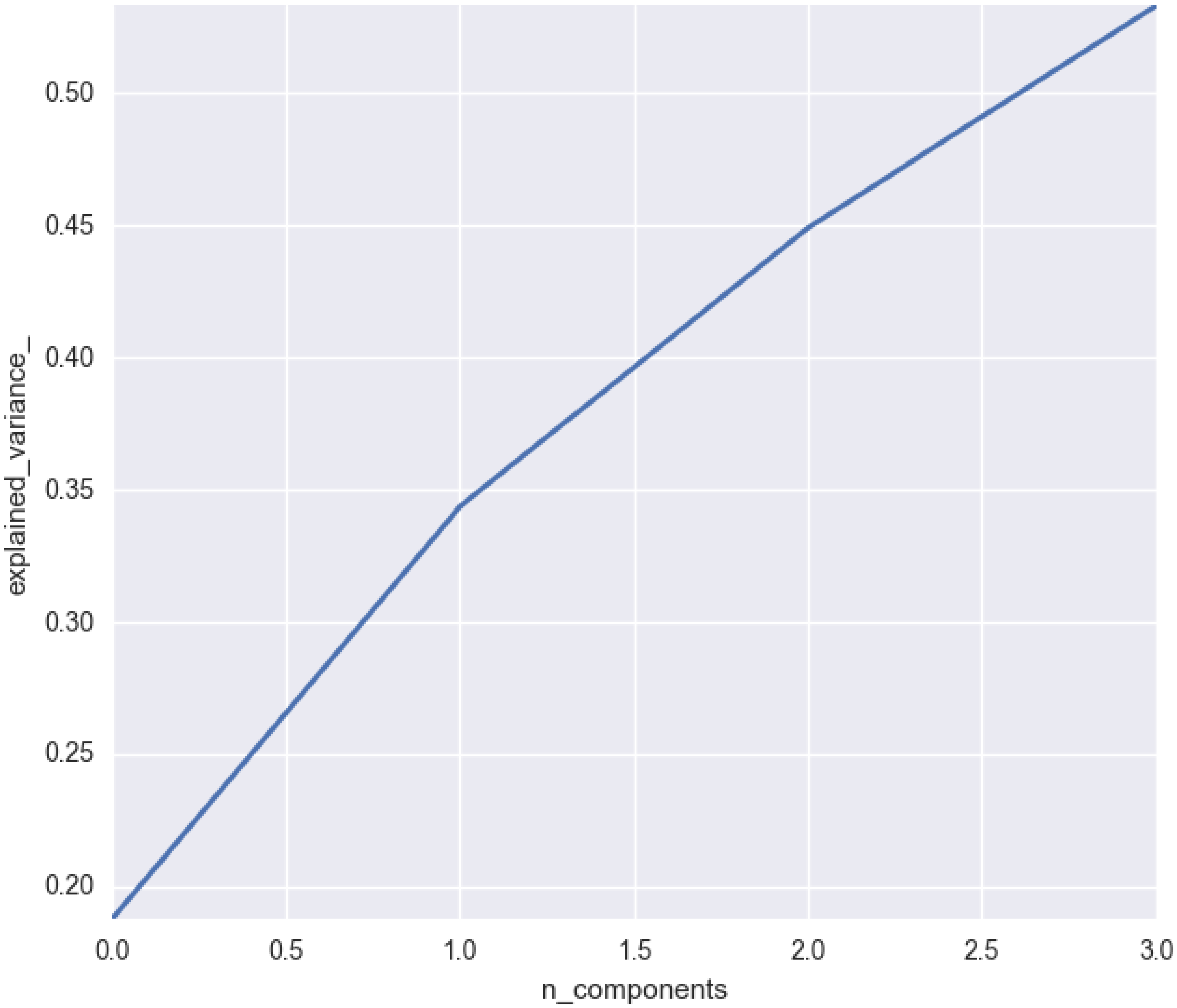
We can take the log of SalePrice to make it normally distributed...



Principle Component Analysis

Principal Component Analysis (PCA) uses linear algebra to transform the dataset into a compressed form. Generally this is called a data reduction technique. A property of PCA is that you can choose the number of dimensions or principal components in the transformed result. In the example below, we use PCA and select 3 principal components.

Looks like most of the variance can be explained within first 3 principe components.



Algorithms and Techniques

This supervized learning project requires I build a regressor. Since I have no idea which algorithms will work well, I will build a test harness aka (see Spot Check below) to evaluate several algorithms based on the Root Meas Squared Error (RMSE) metric. I have implemented a function called `rmse_cv` to perform this evaluation locally using cross validation. This is the metric I will be using and comparing against the Kaggle Public Leaderboard (LB).

You can find more details about the test harness in the implementation section. The overall idea is to iterate over a list of regression algorithms listed below and analyze how each performs. I will then take the top 3 performing ones and tune further.

Candidate Regression Algorithms (using default parameters documented in scikit-learn docs)

- LASSO
  - During EDA, noticed several features were linearly correlated to target.
  - LASSO is a regression method that involves penalizing the absolute size of the regression coefficients. By penalizing (or equivalently constraining the sum of the absolute values of the estimates) you end up in a situation where some of the parameter estimates may be exactly zero. Automatic feature/variable selection
- CART in the case where our model can be learned via decision tree splits

- KNeighborsRegressor since home prices vary by geography, this is why the real estate profession uses comparables to determine sale prices
- XGBRegressor since 70% of Kaggle competitions are won by XGBoost.
    - Similarily AdaBoostRegressor & GradientBoostingRegressor since boosting algorithms use future weak learners to correct what past weak learners got wrong.
- RandomForestRegressor since an esemble of weak learners each of which contains a subset of features with bagging might pick up on the structure of the problem to be learned.
- ExtraTreesRegressor fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.
- Support Vector Regressor since PCA showed that most of the variance could be explained by the first 3 components. Suppor vectors may prove similar value.
- Deep Learning using Keras & TensforFlow since our time for feature engineering is limited and it's more efficient to have an algorithm try all the possible higher dimensional relationship analysis

Benchmark

Kaggle maintains a portion of the data for each competition (including this one) as a hold out set which is used to evaluate the generalizability of our model to new unseen data (aka private leaderboard).

Kaggle competitions are decided by a model's performance on a test data set. Kaggle maintains the answers for the test dataset, but withholds them to compare with the submitted predictions. The Public Score is what you receive after each submission (calculated using a statistical evaluation metric described on the Evaluation page). The Public Score is being determined from only a fraction of the test data set (25-33%). This is the Public Leaderboard, and it shows some the performance of your submission relative to others during the competition.

When the competition ends, Kaggle takes the selected submissions and score the predictions against the REMAINING FRACTION of the test set. You never receive ongoing feedback about your score on this portion; hence the name (Private leaderboard). Final competition results are based on the Private leaderboard, and the Winner is the person(s) at the top of the Private Leaderboard. This separation of the test set into public and private portions is what ensures that the most accurate but generalized model is the one that wins the challenge. If you based your model solely on the public data which gives you constant feedback, you run the danger of a model that overfits to the specific noise in that data. This addresses one of the hard challenges in data science is to avoid overfitting, by leaving your model flexible to out-of-sample data.

My goal for this project was to place among top 20% of the Kaggle public leaderboard.

In order to perform well, it's critical to define a benchmark and cross validation strategy locally. I created a method to calculate the Root Mean Squared Error and used it locally as I experimented with various algorithms.

Since scikit learn has a convention convention to maximize scoring functions, while RSME is minimized. By multiplying by negative one, higher values of negative mean squared error are better than lower values and thus conform to the scikit learn convention.

```
def rmse_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train, y, scoring="neg_mean_squa
red_error", cv = 10))
    return(rmse)
```

III. Methodology

Data Preprocessing & Feature Engineering

For most real world data science projects, data preprocessing often ends up taking a large portion of the effort and time. Fortuantely, this is less of an issue for Kaggle competitions since the data is already represented in a tabular format.

Since some learning algorithms are negatively impacted by differing scales of the raw data, I performed a log transformation on the numeric features as well as the target.

Additionally, due to the large number of missing values (see EDA above), I used the scikit-learn imputer module to impute the mean for missing values.

Both our training & test data sets contain both numerical (quantitative) and categorical (qualitative) features. Since scikit-learn expects a tabular numerical training data set, I must convert all categoricals using one-hot encoding (aka binary features).

Features to engineer include but not limited to:

- calculate age of property (relative to oldest)
- years till remodel
- linear combinations of square footage attributes (DID NOT HELP!)
- one hot encoding of categorical features

Feature selection will be done via:

- dropping highly correlated features (see correlation section above)
- PCA
- Recusive Feature Elimination (DID NOT HELP!)
- LASSO

Implementation

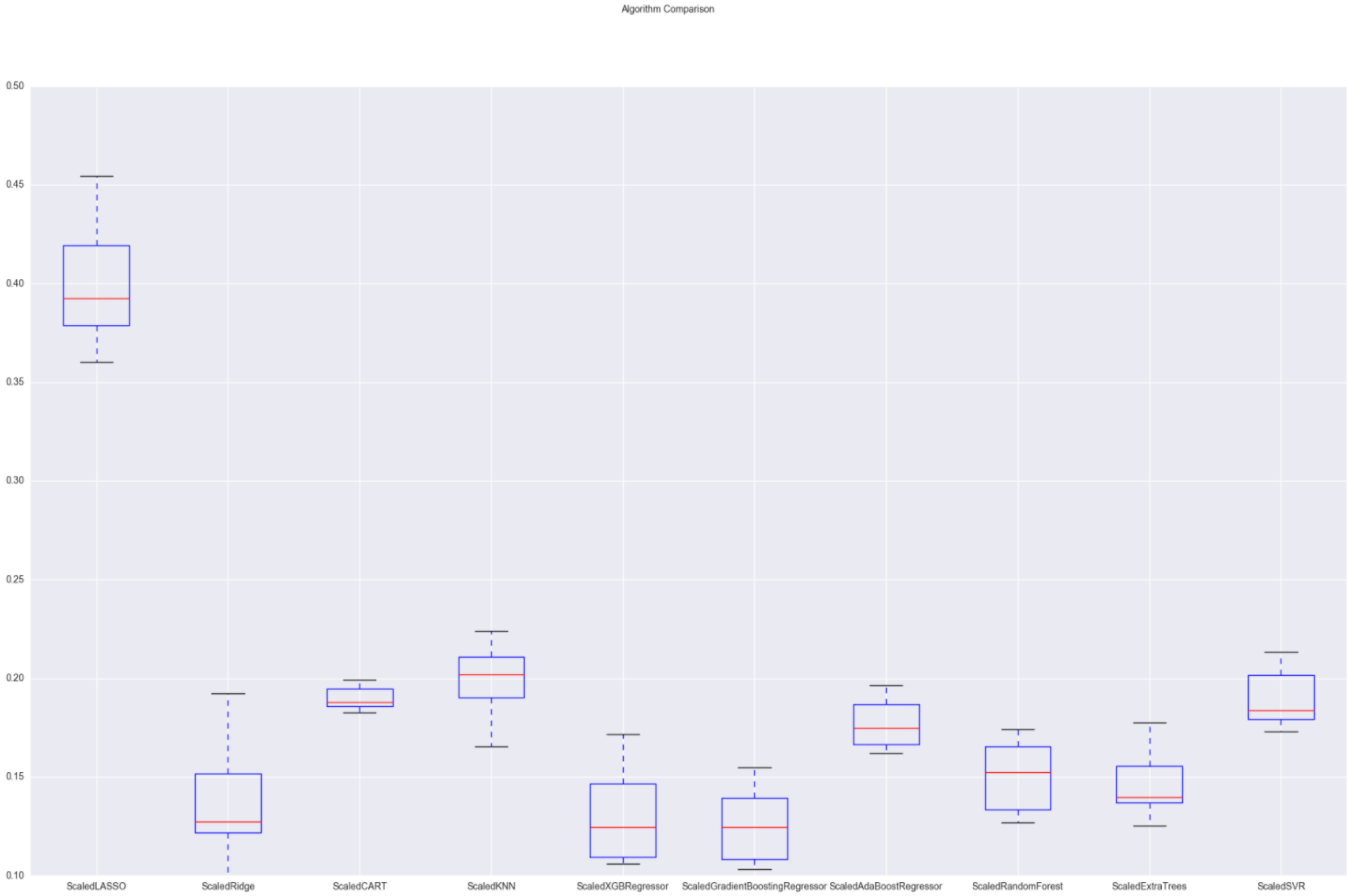
Evaluate Algorithms (Spot Check) with Standardization

I suspect that differing scales of the raw data may be negatively impacting the skill of some of the algorithms. I have performed a standardization in which data is transformed such that each attribute has a mean value of zero and a standard deviation of 1.

Please look at the Jupyter notebook for implementation details including parameter values.

According to the figure below, here are the results (lower is better):

```
ScaledLASSO: 0.398762 (0.027284)
ScaledRidge: 0.135862 (0.026519)
ScaledCART: 0.189709 (0.022052)
ScaledKNN: 0.197759 (0.018980)
ScaledXGBRegressor: 0.129797 (0.021415) ## 1st Best
ScaledGradientBoostingRegressor: 0.125931 (0.018366) ## 2nd Best
ScaledAdaBoostRegressor: 0.176386 (0.011455)
ScaledRandomForest: 0.150491 (0.016641)
ScaledExtraTrees: 0.148834 (0.019462)
ScaledSVR: 0.189114 (0.025751)
```



Next, I tuned the hyper-parameters for several of these algorithms above and created a final ensemble model with `.2 * xgboost_predictions + .8 * lasso_predictions`.

Refinement

The process of improvement upon the algorithms and techniques is iterative and experiemental in nature. you used in your implementation. I used the `rmse_cv` (Root Mean Squared Error Cross Validation) to help me decide whether a certain change was to be persisted or not.

I first started with the best performing model (XGBoost) according to the test harness for model selection mentioned earlier. I then performed feature engineering to create a whole new bunch of signals which I thought would add value including:

- Discretization
  - Tranform Neighborhoods into {Poor, MiddleIncome, Affluent} based on the SalePrice in training data
- Linear construction
  - create new features to combine numeric features with various operators (x_i + x_j, x_i * x_j, x_i ^ 2, log x_i)

These features didn't help my local cross validation score for XGBoost but did help for LASSO.

I've documented the series of refinements I've made along with Kaggle submissions for promising refinements.

Refinement Workflow Results

Model Description	Local CV	LeaderBoard
baseline: forgot to tranform predicted results using exponential	0.132	Bottom 90%
baseline: np.expm1(target)	0.132	0.1335
Dropped 10 least important features (according to XGBoost feature importances)	0.145	-
Feature construction using strategies mentioned above	0.132	-
XGBoost Hyper-parameter tuning colsample_bylevel	0.128	0.1354
LASSO	0.1226	-
LASSO w/ YearBuilt & YearRemodAdd engineered features	0.12243	-
Ensemble XGBoost (60%) with ExtraTrees (40%)	0.131	0.1364
Ensemble XGBoost (30%) with LASSO (70%)	0.128	0.1202 (top 17%)
Ensemble XGBoost (20%) with LASSO (80%)	0.128	0.1203 (top 15%)
Ensemble XGBoost (20%) with LASSO (80%) w/ YearBuilt & YearRemodAdd engineered features	0.12243	0.11968 (top 14%)
Ensemble XGBoost (20%) with LASSO (80%) w/ linear combination of engineered features	0.12255	-
Ensemble XGBoost (20%) with LASSO (80%) w/ highly correlated features dropped	0.123962	0.11965

IV. Results

Model Evaluation and Validation

The iterative refinement workflow documented above led me to the final model which is an esemble of two submodels (XGBoost & LASSO). I've validated the model's performance with the public leaderboard and am currently among top 14%. One of the crucial aspects for my model to work is to perform the same transformations on training vs production (hold out) test set. I've tested the model against the test set provided by Kaggle. I've made sure this is done in a consistent manner which prevents leakage from occuring as well as well as being robust to small perturbations (log feature transformations). The generalizability of the model will be evident at the end of competition. The results can be trusted since it takes care of the precautions mentioned above.

Justification

As you can see from the refinement result table, the final results found are stronger than the benchmark result reported earlier. The final model is a linear combination (80%) LASSO and (20%) XGBoost. Due to the high ranking on the public leaderboard, I conclude the model has sufficient predictive power to predict Sale Price.

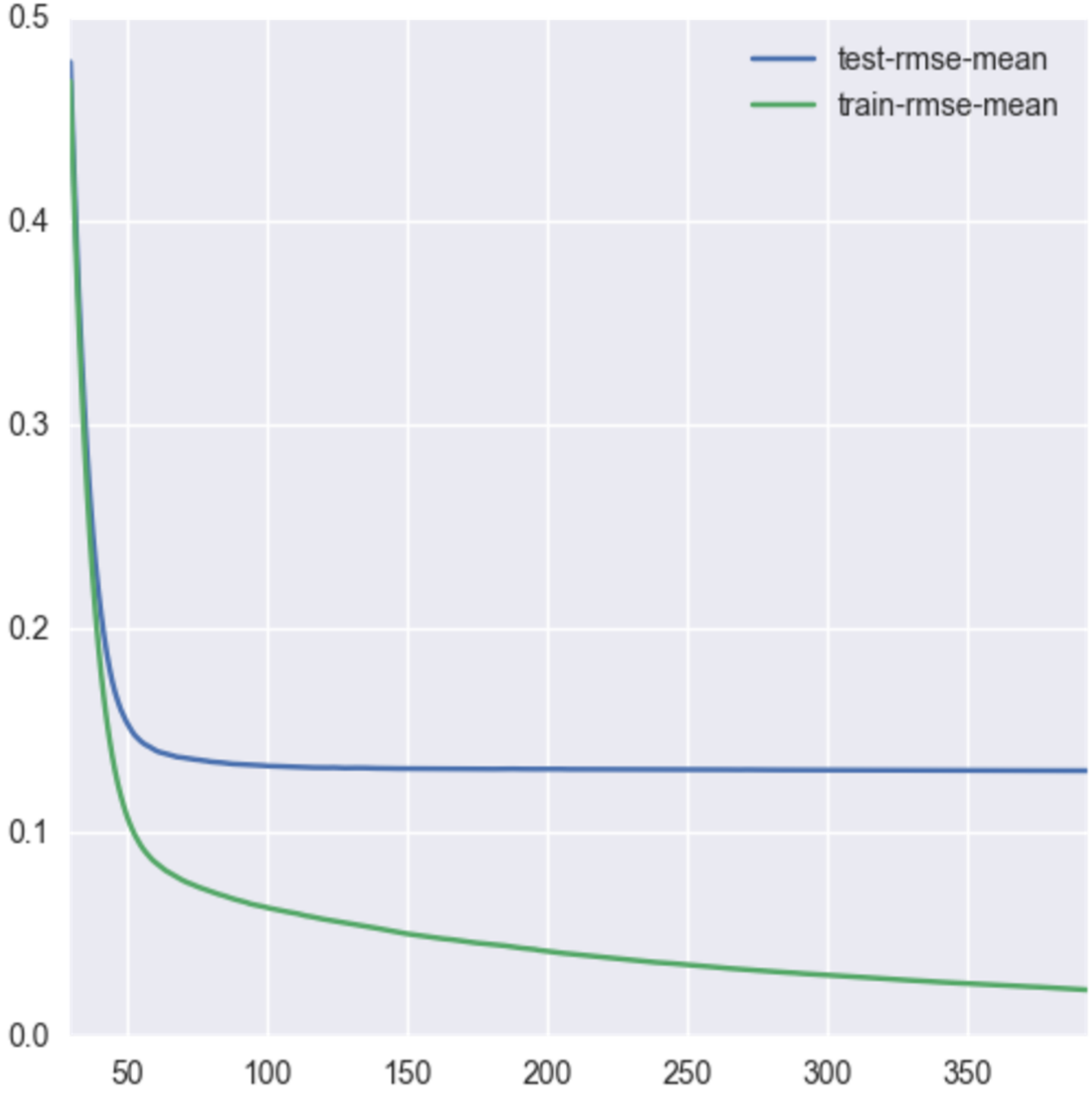
Furthermore, the output of the feature importance tables provided additional support as it's consistent with our domain knowledge and our intution.

V. Conclusion

Free-Form Visualization

RSME (Train vs Test)

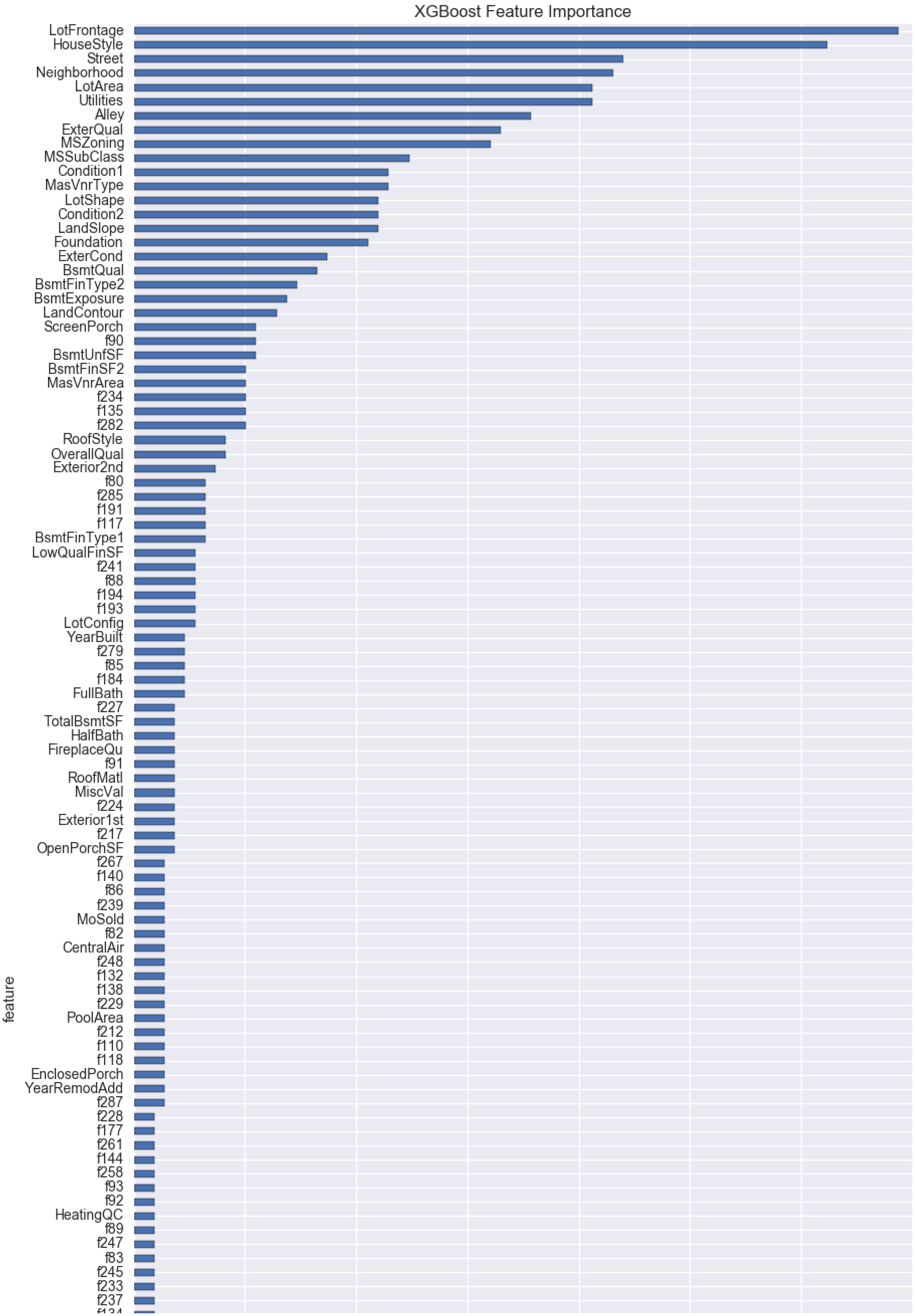
You can observe that the RSME for training is much lower than test which expected since we trained using the training data set and our test set doesn't contain targets. You can also see that the model doesn't perform any better (flat line) after 50 estimators on the test set.

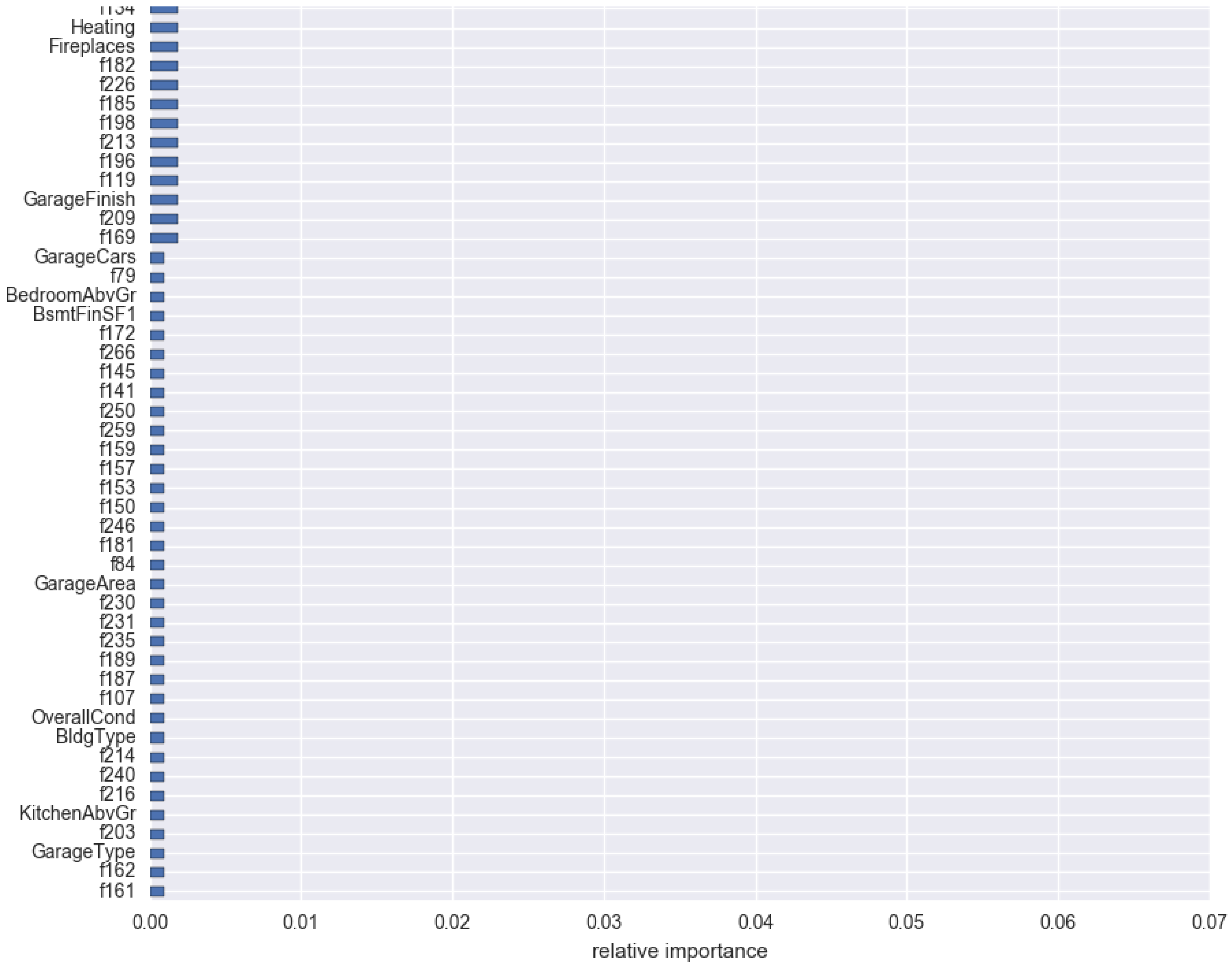


Feature Importances

Model interpretability is key to build trust in the output of the model. Feature importance table provides the the contributions of each feature to the final model in sorted order.

XGBoost Feature Importances


XGBoost Feature Importance
feature
LotFrontage
HouseStyle
Street
Neighborhood
LotArea
Utilities
Alley
ExterQual
MSZoning
MSSubClass
Condition1
MasVnrType
LotShape
Condition2
LandSlope
Foundation
ExterCond
BsmtQual
BsmtFinType2
BsmtExposure
LandContour
ScreenPorch
f90
BsmtUnfSF
BsmtFinSF2
MasVnrArea
f234
f135
f282
RoofStyle
OverallQual
Exterior2nd
f80
f285
f191
f117
BsmtFinType1
LowQualFinSF
f241
f88
f194
f193
LotConfig
YearBuilt
f279
f85
f184
FullBath
f227
TotalBsmtSF
HalfBath
FireplaceQu
f91
RoofMatl
MiscVal
f224
Exterior1st
f217
OpenPorchSF
f267
f140
f86
f239
MoSold
f82
CentralAir
f248
f132
f138
f229
PoolArea
f212
f110
f118
EnclosedPorch
YearRemodAdd
f287
f228
f177
f261
f144
f258
f93
f92
HeatingQC
f89
f247
f83
f245
f233
f237
f134



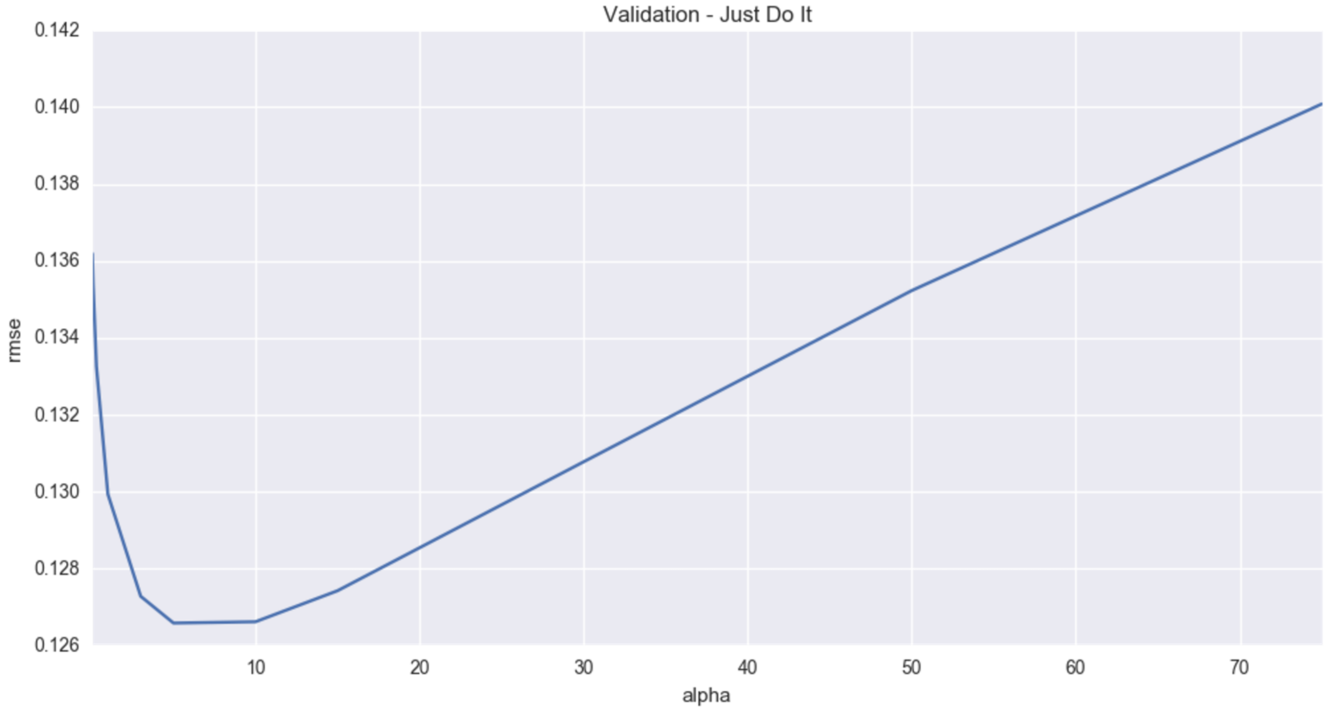
LASSO Feature Importances

```
0.43238483795 HouseStyle
0.306658079877 LotArea
0.232434529655 Street
0.123212466816 MasVnrArea
0.0971238714301 SaleCondition
0.0808617139641 ScreenPorch
0.0661590211202 LotFrontage
0.049453316714 YearBuilt
0.0334307485067 OverallQual
0.0325227384018 Exterior2nd
0.0319212149676 YearRemodAdd
0.0281572640839 FireplaceQu
0.023089079368 Neighborhood
0.0226932484255 Condition1
0.0137705292927 LowQualFinSF
0.0118515554231 EnclosedPorch
0.0106919987952 OpenPorchSF
0.00847464379466 Utilities
0.00831218227088 BsmtFullBath
0.008269826877 BsmtExposure
0.00729119609035 Fireplaces
0.00547443378409 GarageArea
0.00483136123582 ExterCond
0.00370262728343 BsmtFinType1
0.00286553088743 BedroomAbvGr
0.00276529252337 TotalBsmtSF
0.00268664585463 BsmtCond
0.00232346460797 Foundation
9.49548227015e-05 LandContour
0.0 YrSold
0.0 WoodDeckSF
0.0 TotRmsAbvGrd
-0.0 RoofStyle
-0.0 PoolQC
0.0 PavedDrive
0.0 OverallCond
0.0 MasVnrType
0.0 MSZoning
0.0 LotShape
-0.0 KitchenQual
0.0 KitchenAbvGr
0.0 HeatingQC
0.0 Heating
-0.0 GarageYrBlt
-0.0 GarageQual
-0.0 GarageFinish
0.0 GarageCond
-0.0 GarageCars
-0.0 Functional
```

```
0.0 FullBath
0.0 Exterior1st
-0.0 ExterQual
-0.0 Electrical
-0.0 BsmtQual
-0.0 BsmtHalfBath
0.0 BsmtFinType2
-0.0 BsmtFinSF2
0.0 Alley
0.0 3SsnPorch
0.0 1stFlrSF
-0.000413127896622 GrLivArea
-0.00082409880122 2ndFlrSF
-0.00142257874049 LotConfig
-0.0029507208516 BsmtFinSF1
-0.00498385472283 LandSlope
-0.00569681578958 Condition2
-0.00683008097597 BldgType
-0.00715378950237 SaleType
-0.00869858413963 MiscFeature
-0.0136377572739 MSSubClass
-0.0158043933627 MiscVal
-0.0172049027349 MoSold
-0.0175369827903 GarageType
-0.0236894322544 RoofMatl
-0.0325322135113 Fence
-0.0333092460561 CentralAir
-0.0397138716501 HalfBath
-0.0670340111069 PoolArea
-0.292551006308 BsmtUnfSF
```

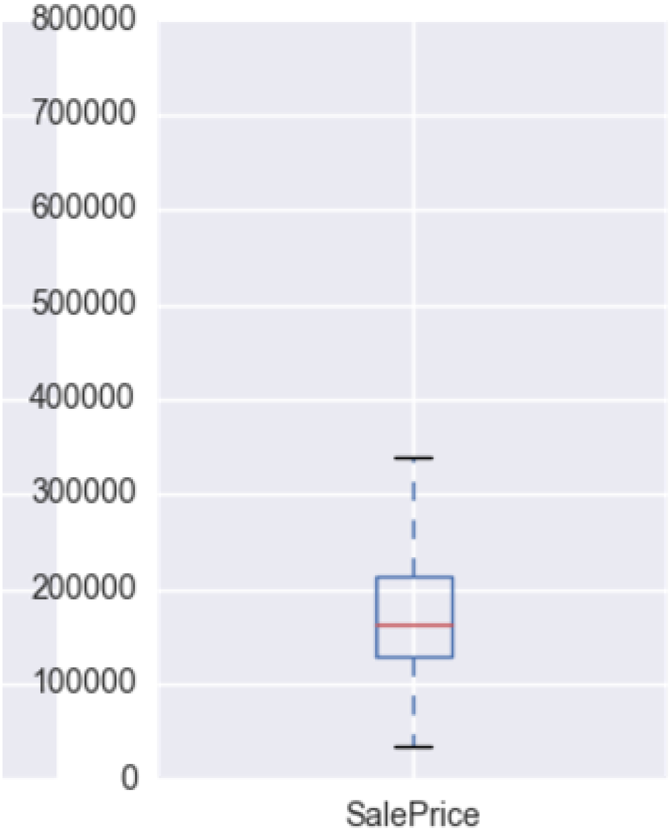
Hyperparameter vs Cross Validation Score

This plot illustrates the RMSE relative to various values for `alpha`. You can see that the minimum RMSE is achieved when `alpha=10` and greater values of alpha results in larger (undesirable) values for RMSE.



SalePrice Outlier Analysis

It's important to spot outliers in our target. The IQR range seems to be from 120K - 205K.



Target Variable Outlier Analysis

There were a number of values in the target variable which were outliers. It's important to note the IQR range and to spot check them against our predicted results.

Eval Equation

Reflection

This project was a great opportunity to define & implement my end-to-end data science workflow using a results oriented approach.

The workflow is as follows:

1. Problem Definition (Ames house price data).

   - Experimental Design: identify data sources, formats, data dictionary, features and target.
2. Loading the Dataset

   - Load and pre-process the data into a representation which is ready for model training
3. Exploratory Data Analysis

   - Gather insights by using exploratory methods (univariate feature distributions, skew and correlated attributes to name a few)
4. Evaluate Algorithms
5. Evaluate Algorithms with Standardization
6. Algorithm Tuning

   - Use GridSearch to search & tune hyper-parameters
7. Ensemble Methods (such as Bagging and Boosting, Gradient Boosting looked good).
8. Finalize Model (use all training data and confirm using validation dataset).

Interestings

Having gone through the iterative process of model selection & evaluation with the public leaderboard, I'm confident I'll be able to do this in the future for more comprehensive projects.

I was also quite pleased with the support Kaggle users provide via the forums. Taking the time to go through the questions and kernels is a great learning resource for practitioners. I learned some new techniques which I'm looking forward to using in the future such as identifying how the private and public data sets were split as this information will help me build a more generalizable model.

Challenges One of the challenges with working with Kaggle is to make sure you're not overfitting to the public leaderboard. It's tempting to see your score rise and sometimes there are quick shortcuts which may raise your score but may not necessarily make sense. It's important to keep a disciplined approach during model selection and evaluation.

It's also critical to structure your code in a way which produces repeatable & reproducible results. Writing modular code which is easy to communicate with others,

Improvement

I plan to work on this Kaggle competition well after my capstone project submission. I believe that engineering new features from existing ones will be the key to building a generalizable model. For example, experimenting with newly created features which are linear combinations of existing ones (BsmtUnfSF/TotalBsmtSF) may provide some lift in the score.

One of the activities I found tremendously useful is reading through the forums. Not only do you find code refactoring opportunitties, but you also gain insight into how others decide to approach this learning problem. Data science often involves experimentation and contributing my EDA as well as baseline model is an improvement both personally and to the community at large.

Another improvement I could do is define a more aggresive strategy for dealing with outliers. The most obvious is to remove them.