Advanced Regression Techniques To Predict Housing Prices

By

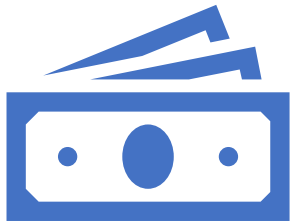
Daniel Bradley, Nate Berman and Peter Shapiro

The Business Problem

Current User State: Inexperienced home buyers can be limited in resources to accurately predict its true value

- Real Estate Agents
- Houses in close proximity
- Online information

The Goal: create a tool that can accurately predict the price of homes based off of the most important features



Benefit: An accurate model will allow home buyers to identify undervalued homes and save more money overall

The Data

- Kaggle DataSet(https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data)
- 1,460 homes in Ames, Iowa
- 80 explanatory variables for almost every home
- Categorical and numerical data features
- 6,965 null values

Frame The Problem

01

SUPERVISED LEARNING TASK-

WE KNOW THE OUTCOME (EACH INSTANCE COMES WITH HOUSE PRICES)

02

MULTIPLE REGRESSION-

MULTIPLE ATTRIBUTES/FEATURES- USES MULTIPLE FEATURES TO MAKE THE PREDICTION

03

MULTIVARIATE REGRESSION-

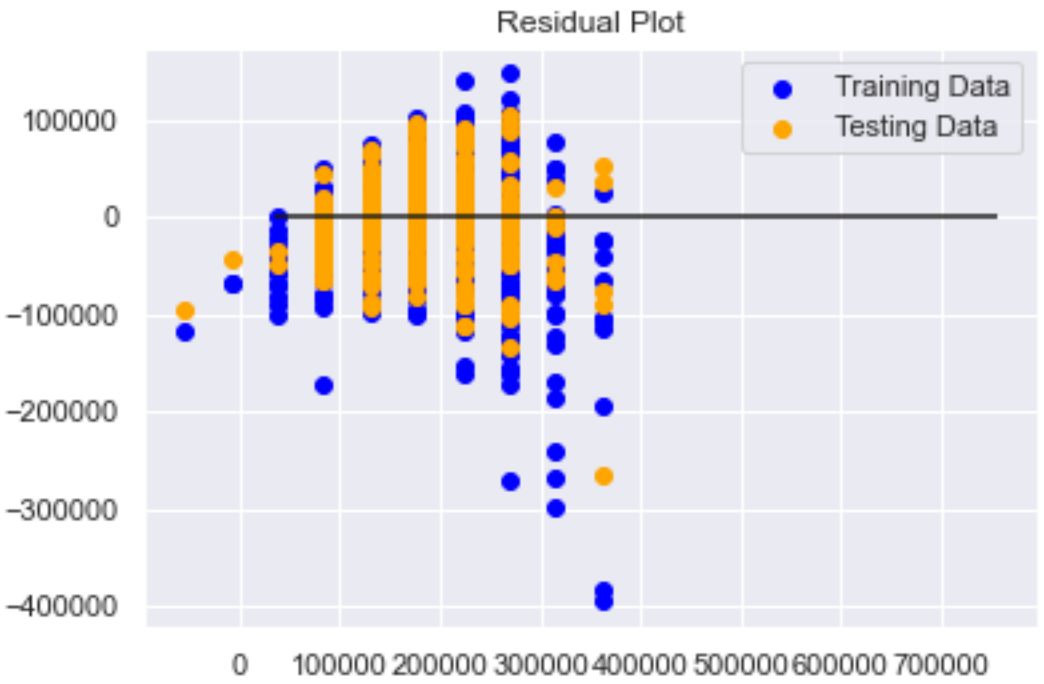
WE ARE ONLY TRYING TO PREDICT ONE OUTCOME WITH MULTIPLE FEATURES

04

BATCH LEARNING

BECAUSE IT IS SMALL ENOUGH TO FIT INTO LOCAL MEMORY

Simple Model

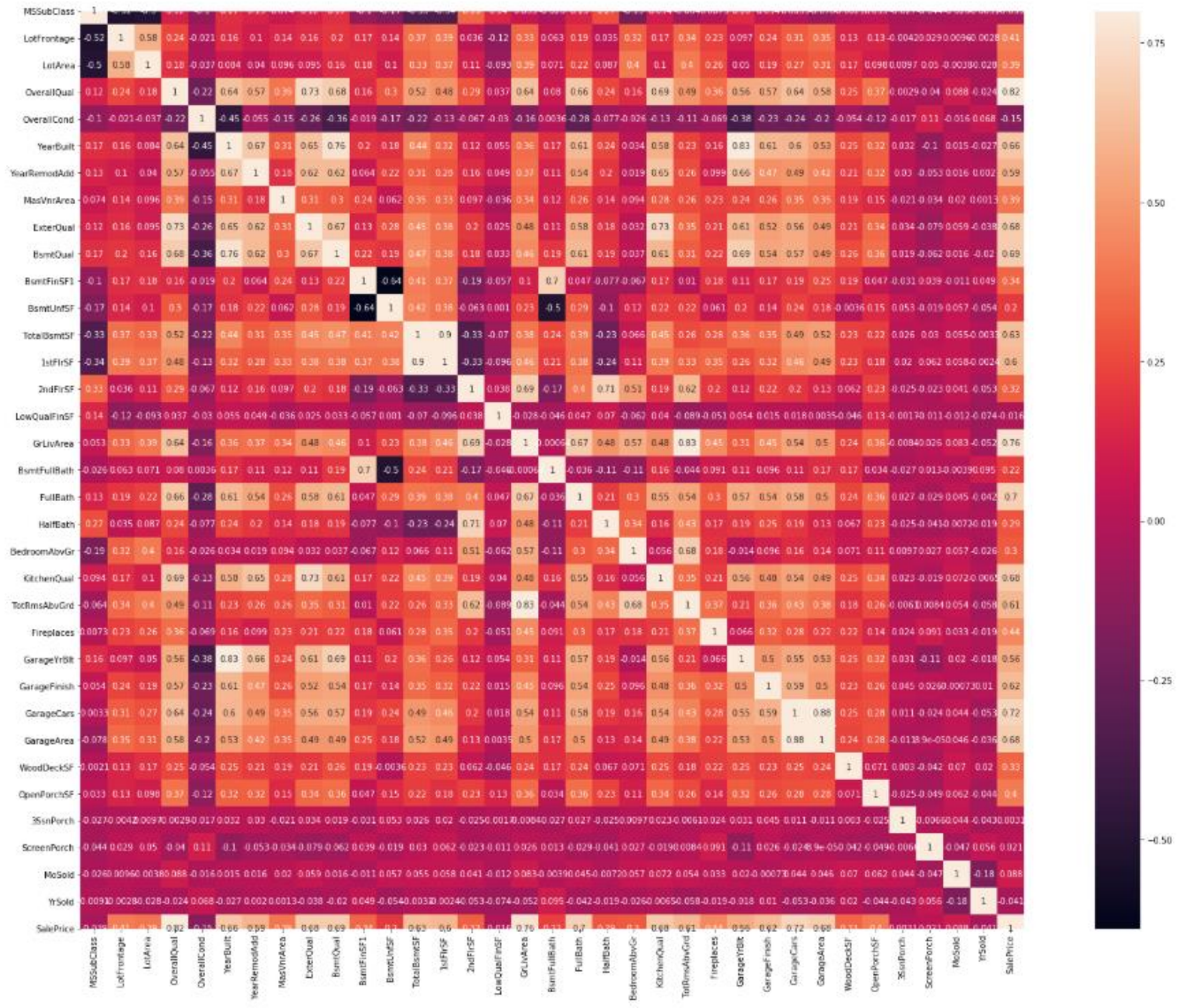


Model Type	Training R^2	Testing R^2
Single Feature OLS	0.618	0.686

Ordinary Least Squares Multiple Regression

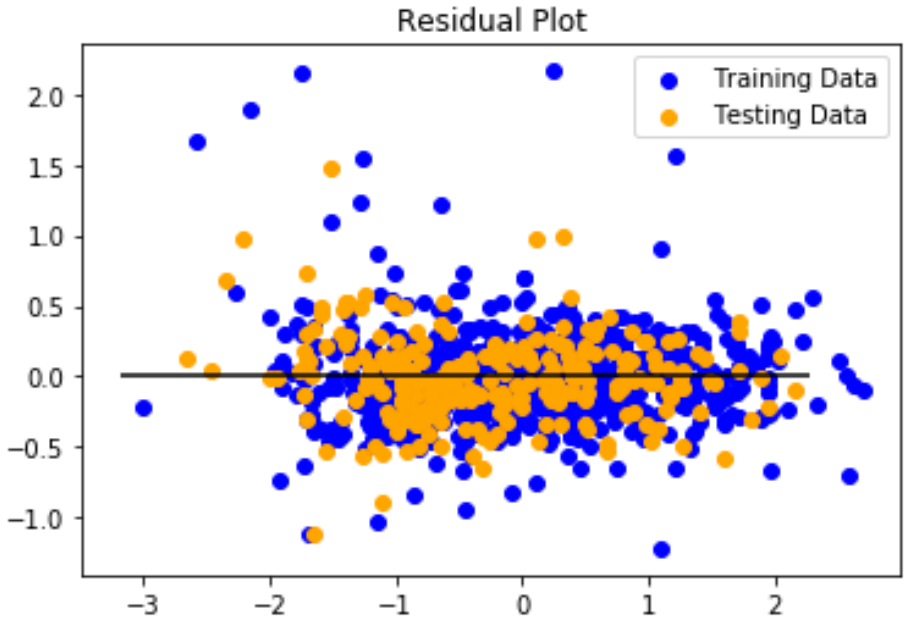
Goal: Minimize the sum of square differences between observed and predicted values

- Normalized the price feature with the natural log (originally skewed right)
- Eliminated all categorical data and initially focused on numeric
  - Median()
  - Mean()
- Testing score was in the mid 80's
- Used seaborn correlation heat map to eliminate clear, uncorrelated features



Exploring The Data

- Converted categorical data to numeric and included the highest correlated features in the model
- Resulted in 33 features and 91.3% testing accuracy
- Utilized recursive feature elimination to find the optimal number of features
  - Resulted in 13 features
  - However, when those features were used and tested, the accuracy of the testing model dropped

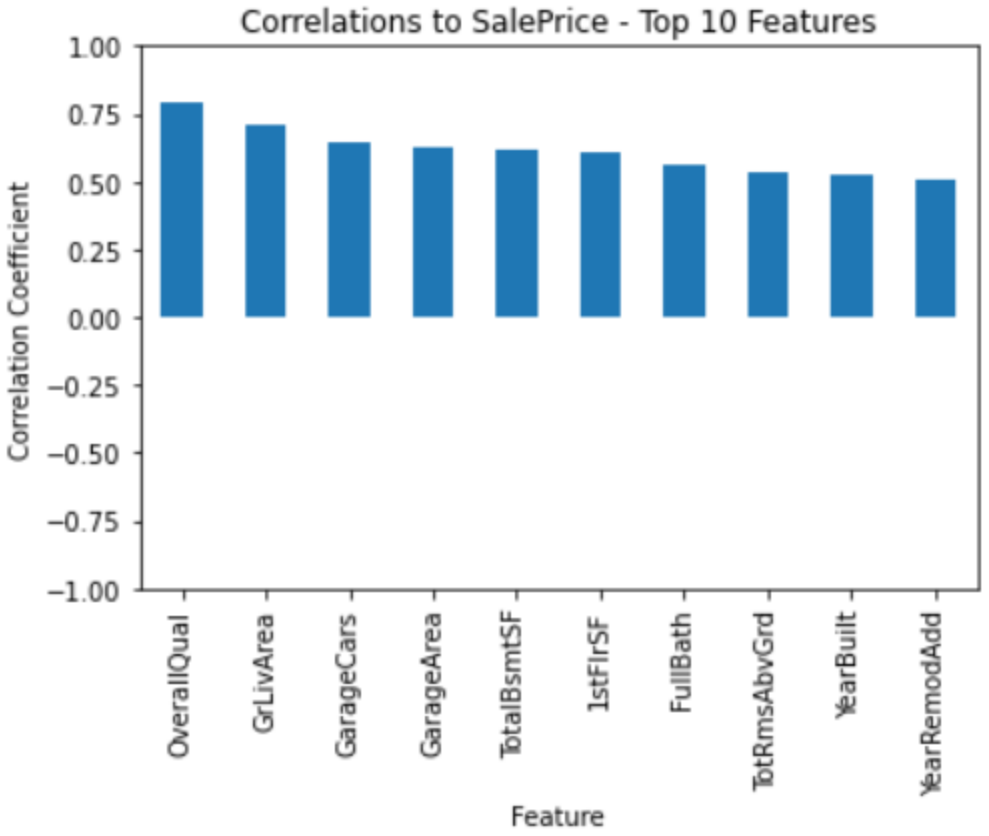


Mean Squared Error: 0.0945, R2: 0.9128

Training Model Score: 90.0%
Testing Model Score: 91.3%

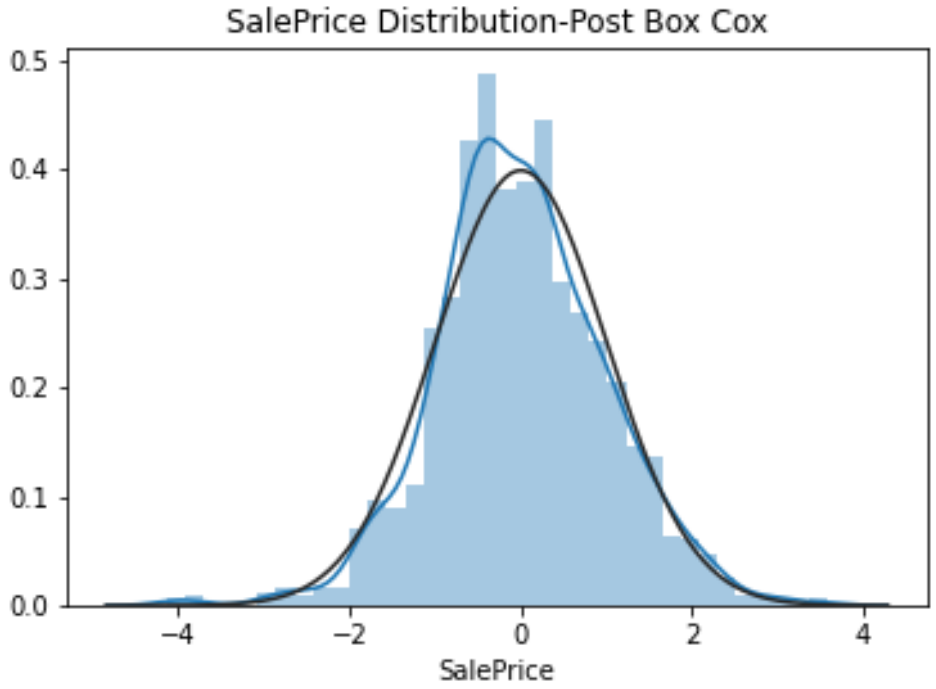
Nate-Lasso Regression

- I first wanted to review all the numerical features and their correlation to SalePrice. This provided me with the features that I will use for the train set.




```
top10 = pd.concat([train.corr()['SalePrice'].sort_values(ascending=False)[1:11]])
```

- After cleaning data, I looked to make sure the data followed a normal distribution. It was not.
  - I used a box-cox to transform the data to a normal distribution.



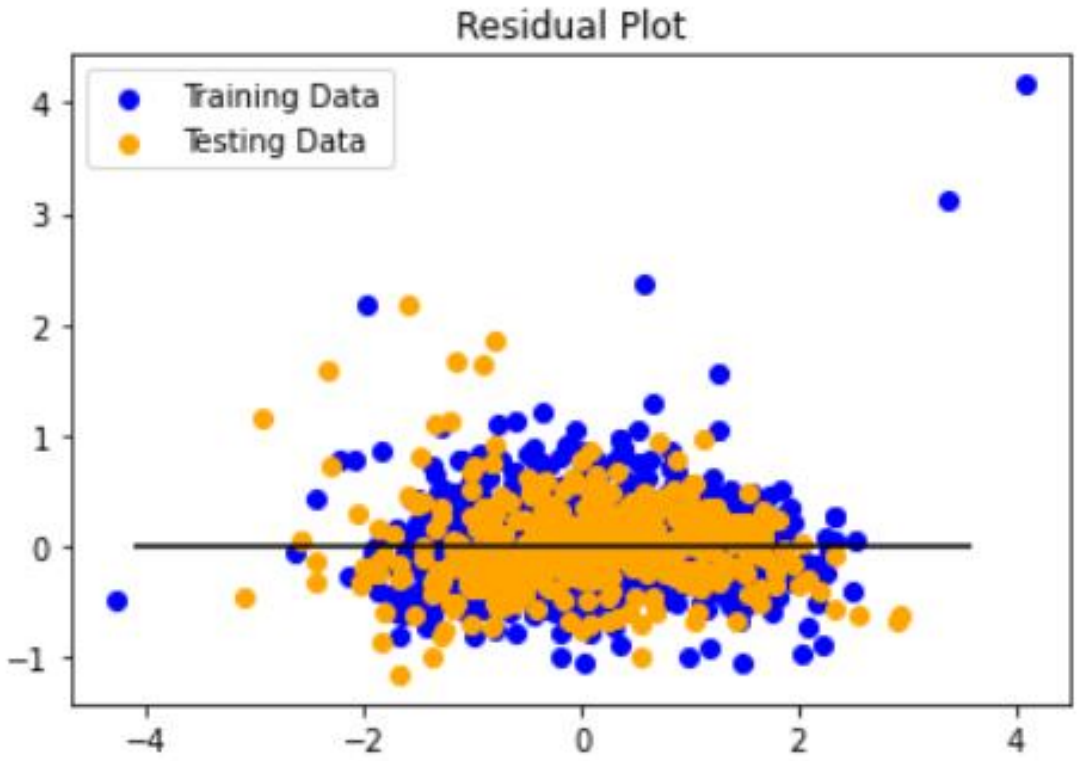

```
pd.DataFrame((power_transform(train,method='box-cox')),columns=train.columns)
```

Lasso Regression

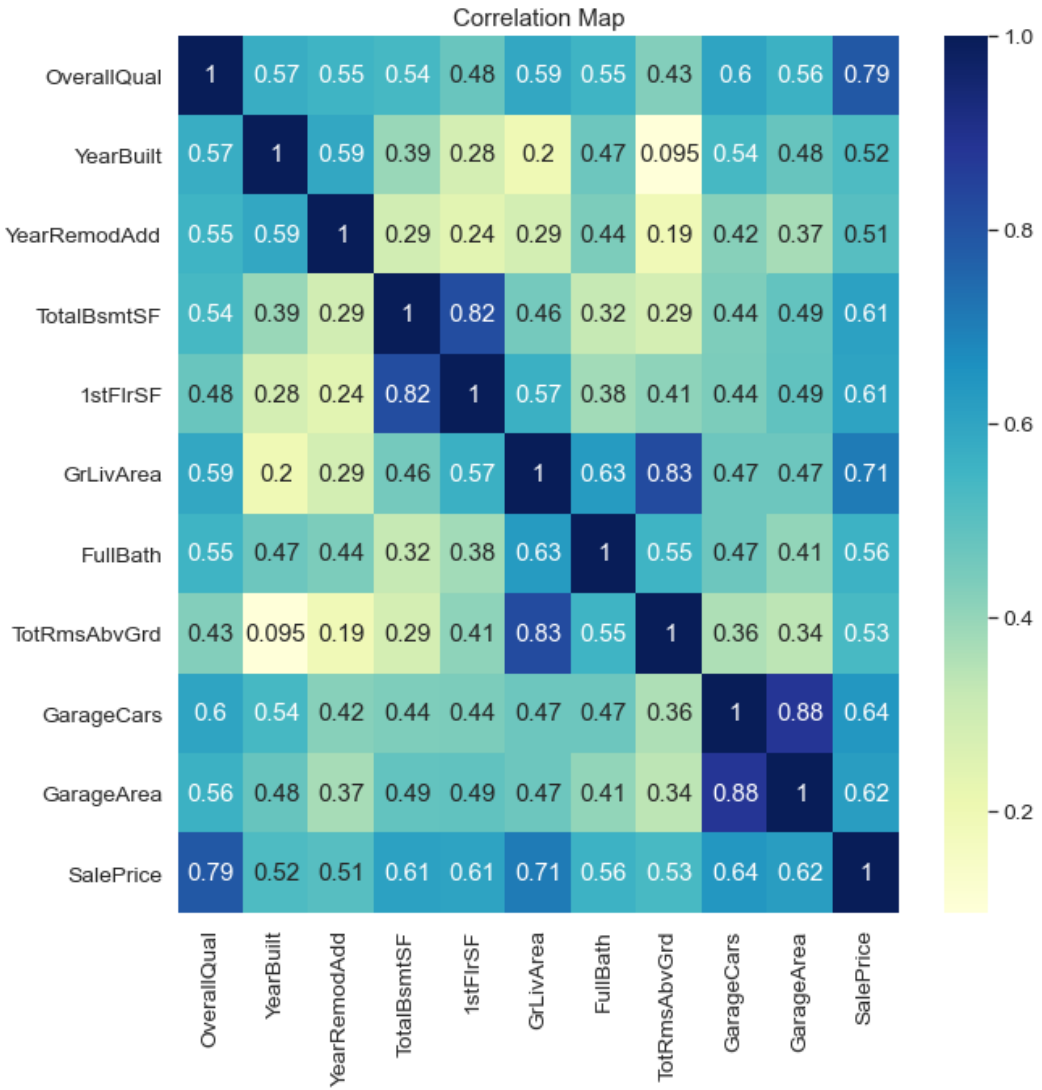
- The data set was divided into a train and test set where GridSearchCV was used to get the best alpha and train and test the data with a Lasso Regression.
  - The best alpha was 0.001
  - Train set had 977 records and 15 features
  - Test set had 977 records to predict SalePrice
- Compared to a standard linear regression it had slightly improved results, 0.05 and a lower MSE, 0.017

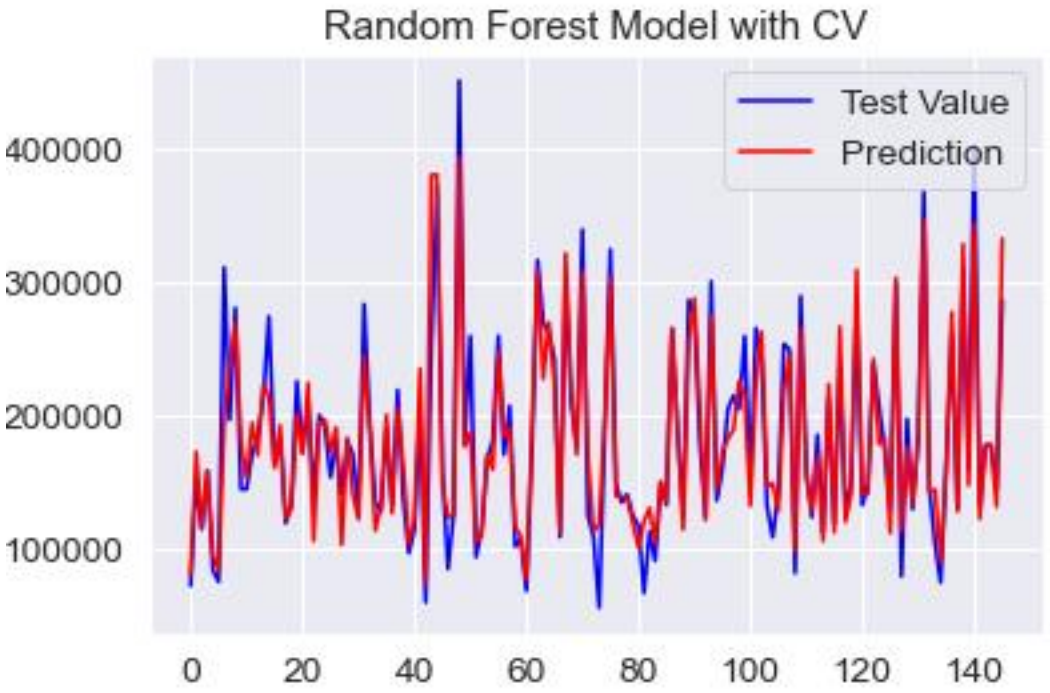
```
Linear Regression training score: 0.8462703454590321
Linear Regression test score: 0.8598790679349718
Linear Regression MSE: 0.1628601803777612
Best Alpha: 0.001
Lasso training score for alpha=0.001: 0.8338811135194252
Lasso testing score for alpha=0.001: 0.8644782026844766
Lasso Regression MSE: 0.14476825932906878
```

- Based on the density of the residual plot, we can see high levels of accuracy however we are able to identify some outliers in the training set.



Daniel

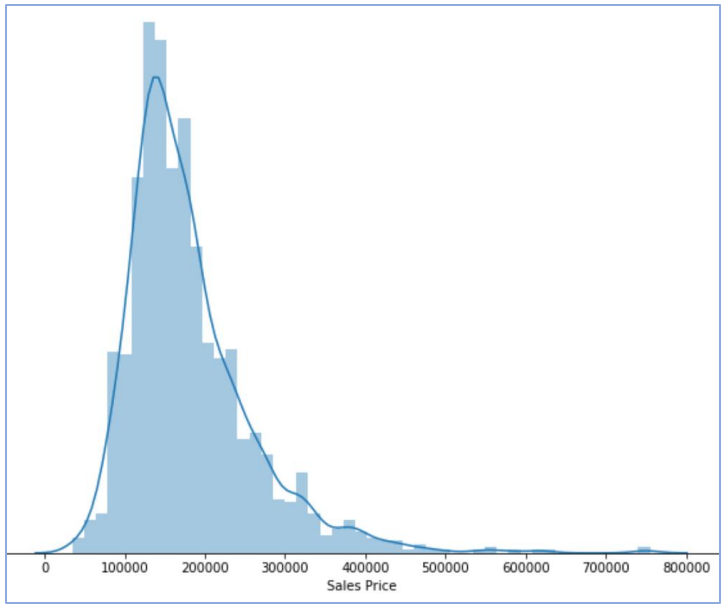

Correlation Map
OverallQual 1 0.57 0.55 0.54 0.48 0.59 0.55 0.43 0.6 0.56 0.79
YearBuilt 0.57 1 0.59 0.39 0.28 0.2 0.47 0.095 0.54 0.48 0.52
YearRemodAdd 0.55 0.59 1 0.29 0.24 0.29 0.44 0.19 0.42 0.37 0.51
TotalBsmtSF 0.54 0.39 0.29 1 0.82 0.46 0.32 0.29 0.44 0.49 0.61
1stFlrSF 0.48 0.28 0.24 0.82 1 0.57 0.38 0.41 0.44 0.49 0.61
GrLivArea 0.59 0.2 0.29 0.46 0.57 1 0.63 0.83 0.47 0.47 0.71
FullBath 0.55 0.47 0.44 0.32 0.38 0.63 1 0.55 0.47 0.41 0.56
TotRmsAbvGrd 0.43 0.095 0.19 0.29 0.41 0.83 0.55 1 0.36 0.34 0.53
GarageCars 0.6 0.54 0.42 0.44 0.44 0.47 0.47 0.36 1 0.88 0.64
GarageArea 0.56 0.48 0.37 0.49 0.49 0.47 0.41 0.34 0.88 1 0.62
SalePrice 0.79 0.52 0.51 0.61 0.61 0.71 0.56 0.53 0.64 0.62 1
OverallQual
YearBuilt
YearRemodAdd
TotalBsmtSF
1stFlrSF
GrLivArea
FullBath
TotRmsAbvGrd
GarageCars
GarageArea
SalePrice
1.0
0.8
0.6
0.4
0.2


Random Forest Model with CV
Test Value
Prediction
400000
300000
200000
100000
0
20
40
60
80
100
120
140

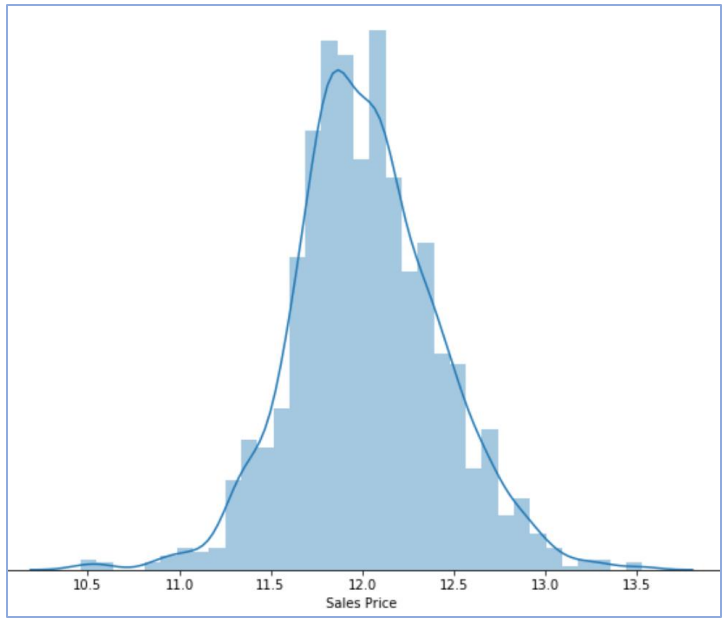
Comparison

Model	No. Features	Training R^2	Testing R^2
Simple Model	1	0.618	0.686
Random Forest	10	0.978	0.867
Random Forest CV	10	0.917	0.883
Lasso	15	0.834	0.865
OLS	33	0.913	0.900

Exploring The Data



Natural Log to normalize the data



Null Values

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

All null values reduced to 0