Prediction of Home Sale Price

By: Leoman Momoh

A detailed report on construction of a Home Sale Price prediction model using python code, machine learning algorithms, and Statistical analysis.

Table of Contents

What's the Right Price?

A peak at the problem

Introduction:

According to Zillow (a leading real estate company), median home prices are projected to rise in the U.S. market. Since home prices are predicted to rise, no buyer wants to pay for a home that is worth less than the listed price. Many buyers rely on realtors to find the home with the best value. Unfortunately, sometimes buyers can be left with an overvalued home once the realtor has claimed his or her commission on the sale.

The problem and data available:

Buyers need a tool to estimate what the actual sales price of a home should be. With an estimation tool, buyers won't have to rely solely on the word of a realtor or a seller that a home is appropriately priced. Using a data set available from Kaggle, I aim to give buyers a resource to determine what the actual price of a home should be based on its features.

The data set I plan to use for my analysis contains data on houses in Ames, Iowa. The data set contains 79 distinct feature variables and missing values that will have to be either omitted or filled. I plan to use statistical analysis, python programming, and machine learning algorithms to identify key features that best explain the variation between sale prices.

Deliverables:

I plan on delivering a housing price prediction model that will be beneficial to buyers, sellers, and investors. In addition to the model, I plan to provide a report that provides details on my process as well as the features that have the greatest influence on price. Buyers, sellers, and real estate investors will find this model useful when determining what the price of a home should be. Buyers and sellers could use this model as a guide to haggle on the price of a home leading to savings or profit on the home depending on the party. Real estate investors could use the insight from the report to focus on investing in homes with valuable features. In conclusion, my aim for this project is to assist all parties by providing a mathematical formula to determine what the starting price for a home should be based on its features.

The Hard Truth

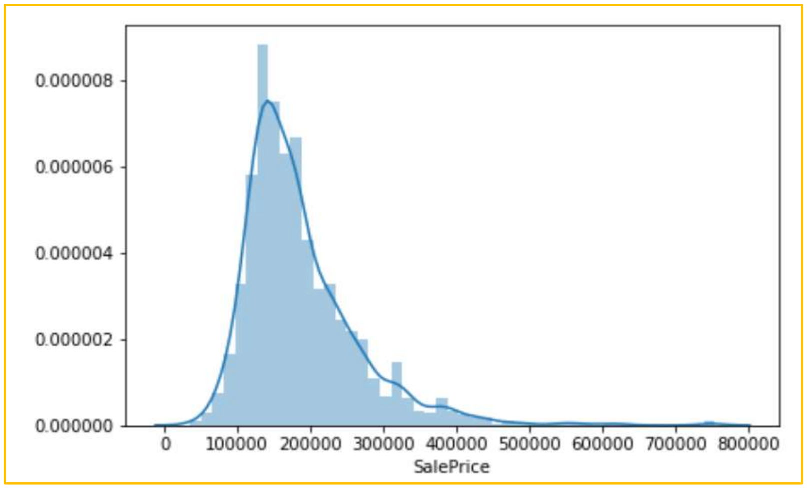
"Any buyer can make a mistake, and if they don't keep a level head, that mistake can end up costing them a lot of money,"

-Herman Chan (San Francisco Real Estate Broker)

Exploratory Data Analysis

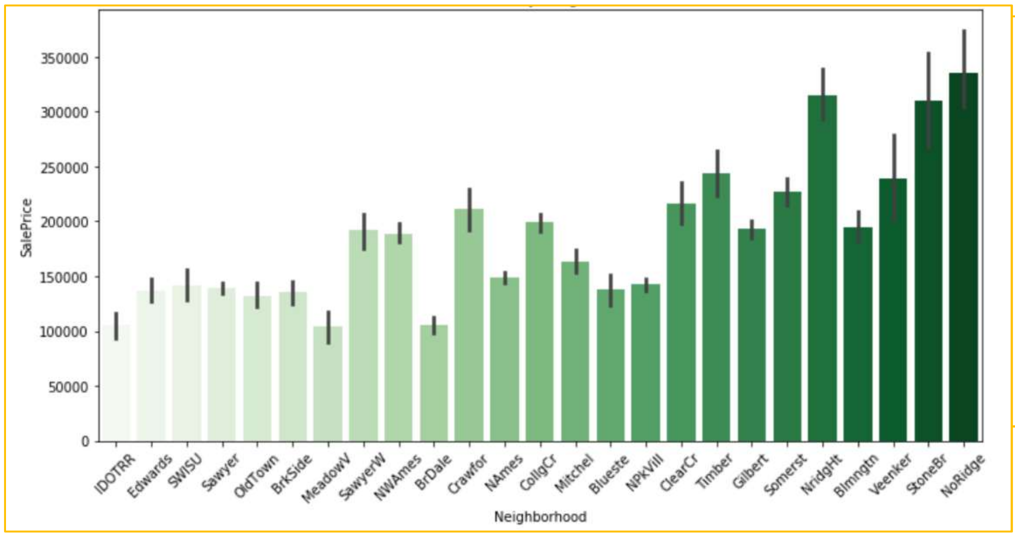
Our main goal in this project is to build a model to predict sales price. Before we start variable selection, we need to explore our data to gain a greater understanding of how variables interact with one another. We will start by looking at the target variable 'Sale Price'. It seems that our data set has a mean sale price of $186,761.78 with a Standard Deviation of $78,913.84. The plot below shows that the distribution of sales price is skewed to the right. We can see that most sales prices lie between $100,000 and $250,000 in the data set.

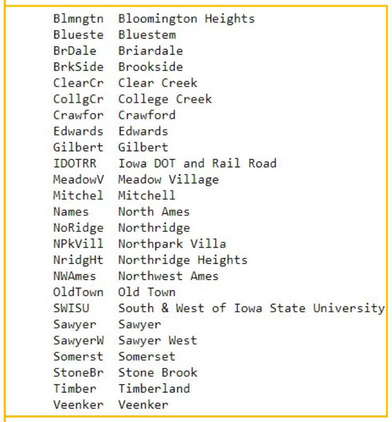
Distribution of Sale Price



The First relationship I wanted to explore in the data set was the relationship between neighborhood and Sale Price. My initial thought is that there should be a strong relationship between neighborhood and Sale Price.

Sale by Neighborhood

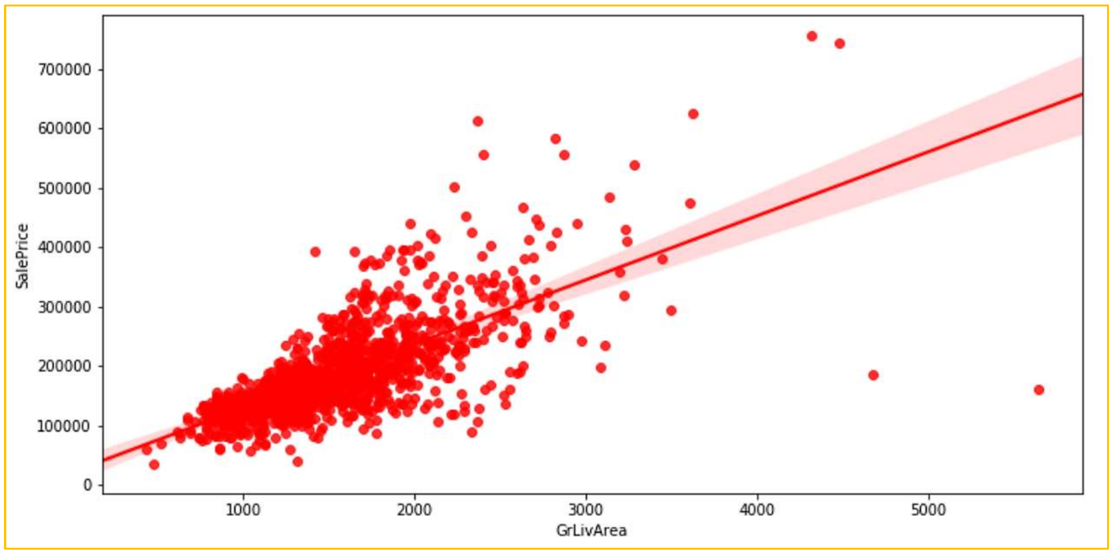




```
Blmngtn  Bloomington Heights
Blueste  Bluestem
BrDale   Briardale
BrkSide  Brookside
ClearCr  Clear Creek
CollgCr  College Creek
Crawfor  Crawford
Edwards  Edwards
Gilbert  Gilbert
IDOTRR   Iowa DOT and Rail Road
MeadowV  Meadow Village
Mitchel  Mitchell
Names    North Ames
NoRidge  Northridge
NPkVill  Northpark Villa
NridgHt  Northridge Heights
NWAmes   Northwest Ames
OldTown  Old Town
SWISU    South & West of Iowa State University
Sawyer   Sawyer
SawyerW  Sawyer West
Somerst  Somerset
StoneBr  Stone Brook
Timber   Timberland
Veenker  Veenker
```

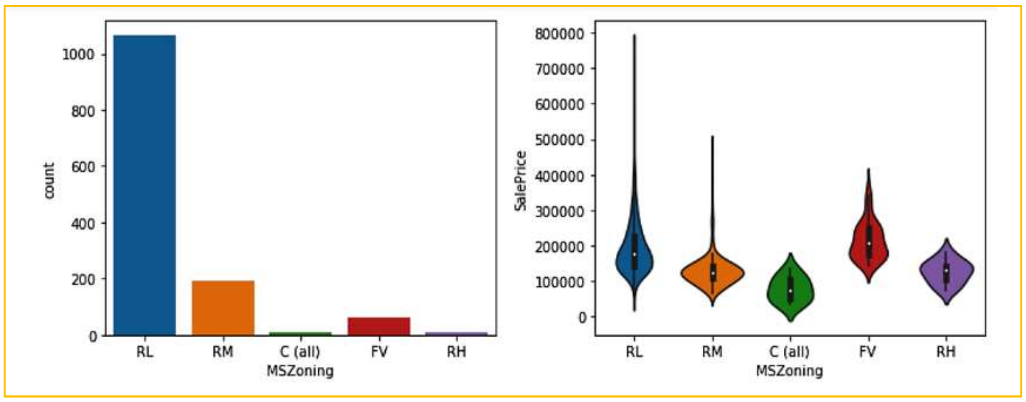
The 'Sales Price by Neighborhood' Plot shows 'NoRidge', 'StoneBr', and 'NridgeHt' are the neighborhoods with the highest average sales price. It seems that 'BrDale', 'MeadowV', and 'IDOTRR' are neighborhoods with the lowest average sales price. Upon further investigation of homes in these neighborhoods, it seems like these homes have lower than average 'GrLivArea' (Above ground living area in square feet) which could have contributed to the lower sale price of homes in this neighborhood. Looking at the Plot of GrLivArea and Sale Price, the plot reveals a high correlation between these 2 variables. The strong positive relationship leads me to believe that GrLivArea may play a crucial role in prediction model.

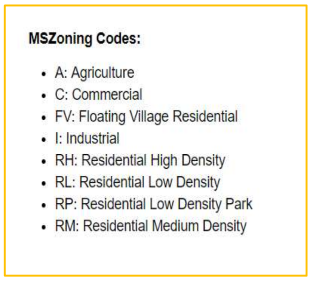
Sale Price vs GrLivArea



Moving on, I wanted to see if MS Zoning codes played a part in Sale Price. The chart below displays the MS Zoning codes present in the data set. The plot shows that most of our observations are homes that are in Residential Low-Density Areas. Homes with the MS Zoning code of 'RL' have an average sale price just under 200k and have the greatest variance of all MS Zoning codes. Floating Village homes are priced higher on average than any other code.

Sale Price vs MS Zoning code

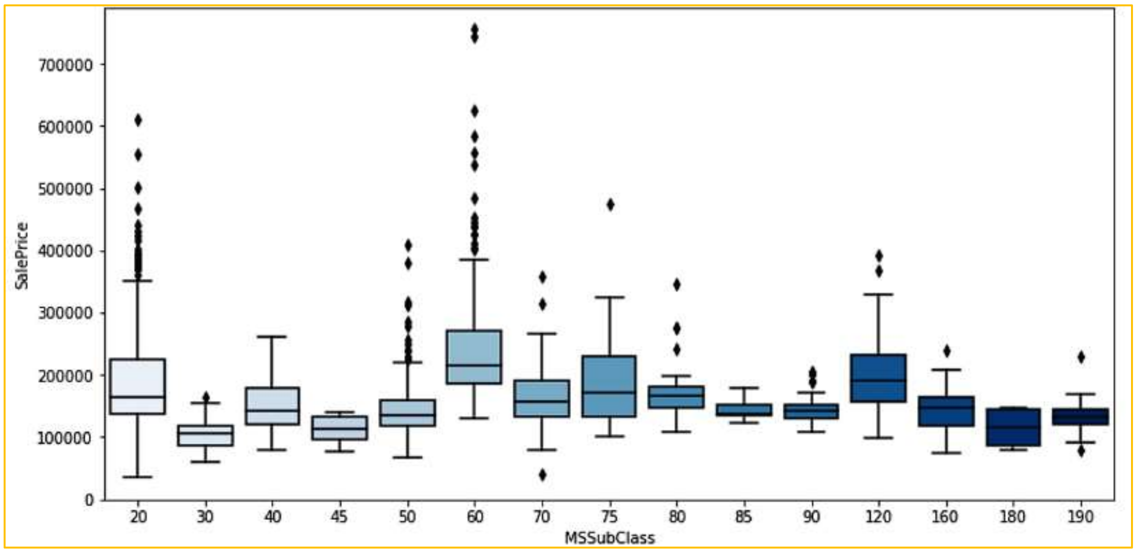


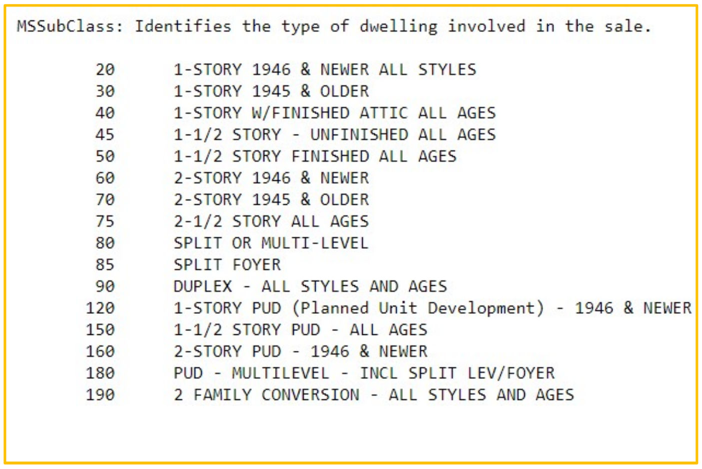

MSZoning Codes:

- A: Agriculture
- C: Commercial
- FV: Floating Village Residential
- I: Industrial
- RH: Residential High Density
- RL: Residential Low Density
- RP: Residential Low Density Park
- RM: Residential Medium Density

The chart below displays the sales price by dwelling code. The plot displays multiple outlier for each subclass. Notice that the code 60, which represents 2 story homes 1946 & newer, has the highest average Sale Price among all other dwelling code. The chart shows almost all dwelling codes have extreme outliers. Split Foyer homes (85) have the lowest variance in price when compared to other subclass types.

Sale Price by MS Subclass codes

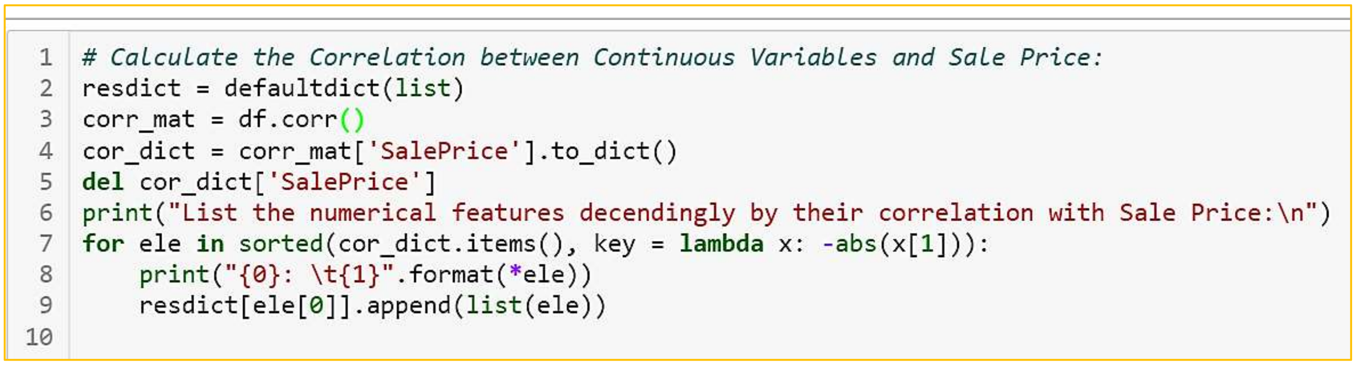




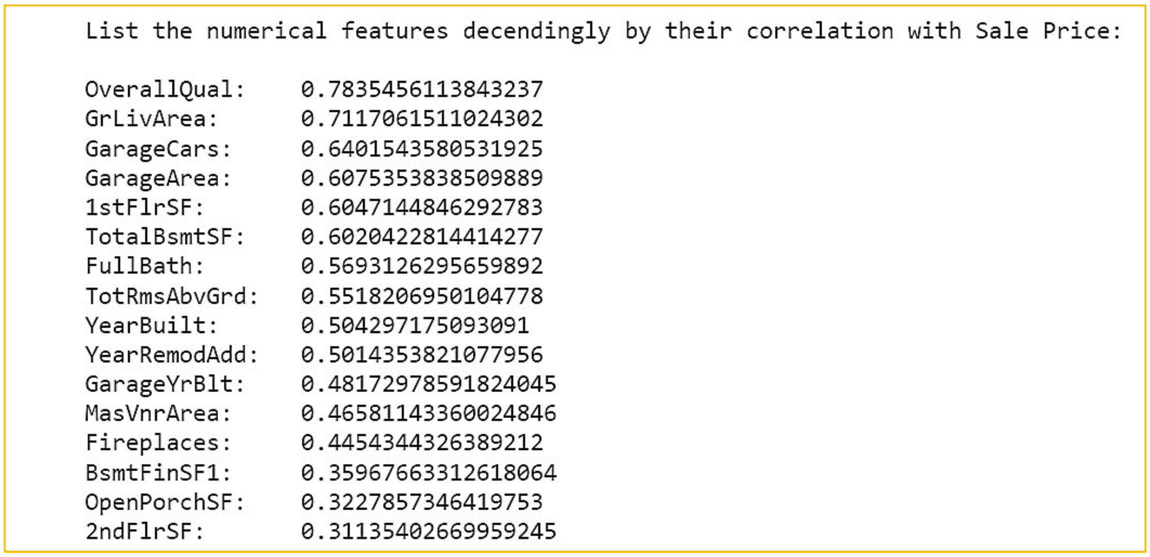
```
MSSubClass: Identifies the type of dwelling involved in the sale.

        20      1-STORY 1946 & NEWER ALL STYLES
        30      1-STORY 1945 & OLDER
        40      1-STORY W/FINISHED ATTIC ALL AGES
        45      1-1/2 STORY - UNFINISHED ALL AGES
        50      1-1/2 STORY FINISHED ALL AGES
        60      2-STORY 1946 & NEWER
        70      2-STORY 1945 & OLDER
        75      2-1/2 STORY ALL AGES
        80      SPLIT OR MULTI-LEVEL
        85      SPLIT FOYER
        90      DUPLEX - ALL STYLES AND AGES
       120      1-STORY PUD (Planned Unit Development) - 1946 & NEWER
       150      1-1/2 STORY PUD - ALL AGES
       160      2-STORY PUD - 1946 & NEWER
       180      PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
       190      2 FAMILY CONVERSION - ALL STYLES AND AGES
```

Statistical Inference

Our data set contains over 70 different predictor variables that can be used to predict sales price. Before the model is trained, we need to verify that all these variables have a statistically significant relationship with Sale Price. We start by looking at the continuous variables in the data set and looking at their relationship with sales price. I wrote a function that went through each continuous variable in the data set and calculates the correlation between that variable and 'Sale Price'.



```python
# Calculate the Correlation between Continuous Variables and Sale Price:
resdict = defaultdict(list)
corr_mat = df.corr()
cor_dict = corr_mat['SalePrice'].to_dict()
del cor_dict['SalePrice']
print("List the numerical features decendingly by their correlation with Sale Price:\n")
for ele in sorted(cor_dict.items(), key = lambda x: -abs(x[1])):
    print("{0}: \t{1}".format(*ele))
    resdict[ele[0]].append(list(ele))

```



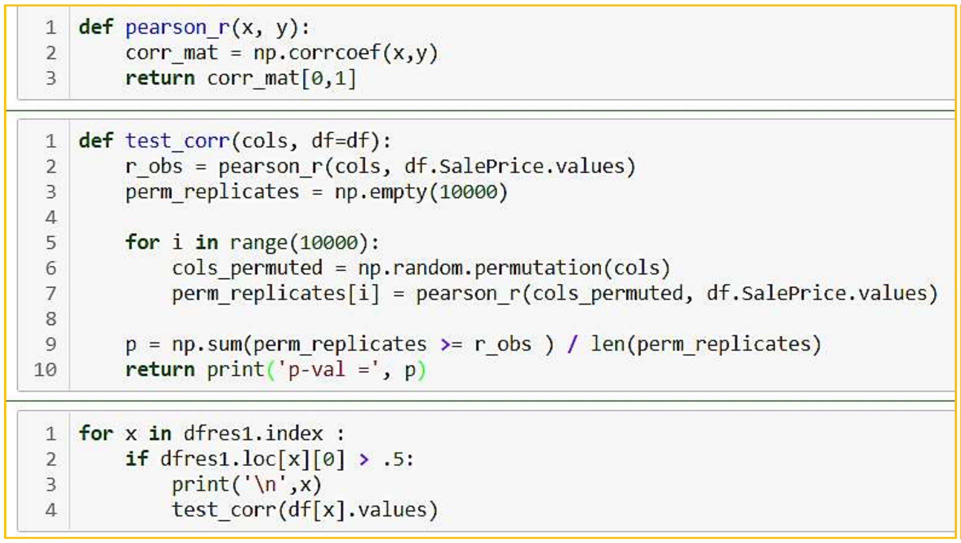
```
List the numerical features decendingly by their correlation with Sale Price:

OverallQual:    0.7835456113843237
GrLivArea:      0.7117061511024302
GarageCars:     0.6401543580531925
GarageArea:     0.6075353838509889
1stFlrSF:       0.6047144846292783
TotalBsmtSF:    0.6020422814414277
FullBath:       0.5693126295659892
TotRmsAbvGrd:   0.5518206950104778
YearBuilt:      0.504297175093091
YearRemodAdd:   0.5014353821077956
GarageYrBlt:    0.48172978591824045
MasVnrArea:     0.46581143360024846
Fireplaces:     0.4454344326389212
BsmtFinSF1:     0.35967663312618064
OpenPorchSF:    0.3227857346419753
2ndFlrSF:       0.31135402669959245
```

Overall Quality, GrLivArea, and Garage Cars have the strongest correlation with Sale Price. Before we include these variables in our model, we need to verify the validity of this relationship with a correlation test. The correlation test will verify that the relationship between Sale Price and a numerical variable isn't by chance. The null and alternative hypothesis is as follows:

H_0: There is no correlation between Sale Price and tested numerical variable

H_A: There is a correlation between Sale Price and tested numerical variable

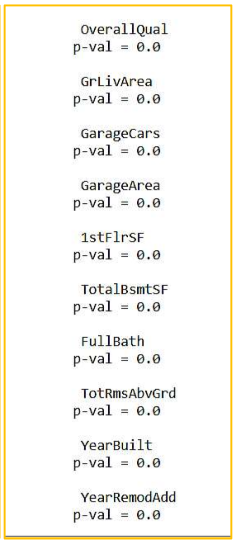
I used the following code to write a function that finds numerical features that have a correlation with Sale Price Higher than .50. Then I used a self-defined correlation function to iterate through the column names and test each one then lists the resulting p value.



```
def pearson_r(x, y):
    corr_mat = np.corrcoef(x,y)
    return corr_mat[0,1]
```

```
def test_corr(cols, df=df):
    r_obs = pearson_r(cols, df.SalePrice.values)
    perm_replicates = np.empty(10000)

    for i in range(10000):
        cols_permuted = np.random.permutation(cols)
        perm_replicates[i] = pearson_r(cols_permuted, df.SalePrice.values)

    p = np.sum(perm_replicates >= r_obs ) / len(perm_replicates)
    return print('p-val =', p)
```

```
for x in dfres1.index :
    if dfres1.loc[x][0] > .5:
        print('\n',x)
        test_corr(df[x].values)
```



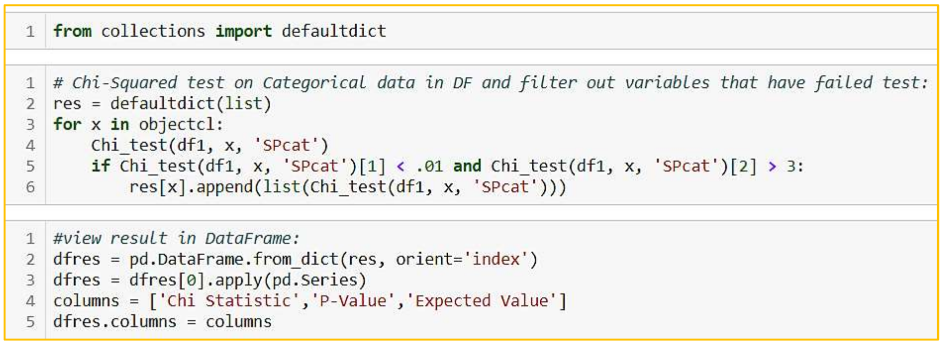
```
 OverallQual
p-val = 0.0

 GrLivArea
p-val = 0.0

 GarageCars
p-val = 0.0

 GarageArea
p-val = 0.0

 1stFlrSF
p-val = 0.0

 TotalBsmtSF
p-val = 0.0

 FullBath
p-val = 0.0

 TotRmsAbvGrd
p-val = 0.0

 YearBuilt
p-val = 0.0

 YearRemodAdd
p-val = 0.0
```

The results from our hypothesis testing shows that the above variables' correlation with sales price are true and not by chance. We can be confident that these feature variables will play a crucial part in our model's ability to predict Sale Price. These variables will be included in our initial model.

Moving on to the categorical variables, we will need to test the relationship with Sale Price as well. We can test this relationship using a chi-squared test of independence. To use this test, we need to categorize our sale price. Using the code below, I split Sale Price into 2 distinct groups. The Null and Alternate hypothesis for our testing is as follows:

H_0: In the population, the tested categorical variables and categorized sale price are independent.

H_A: In the population, the tested categorical variables and categorized sale price are dependent.

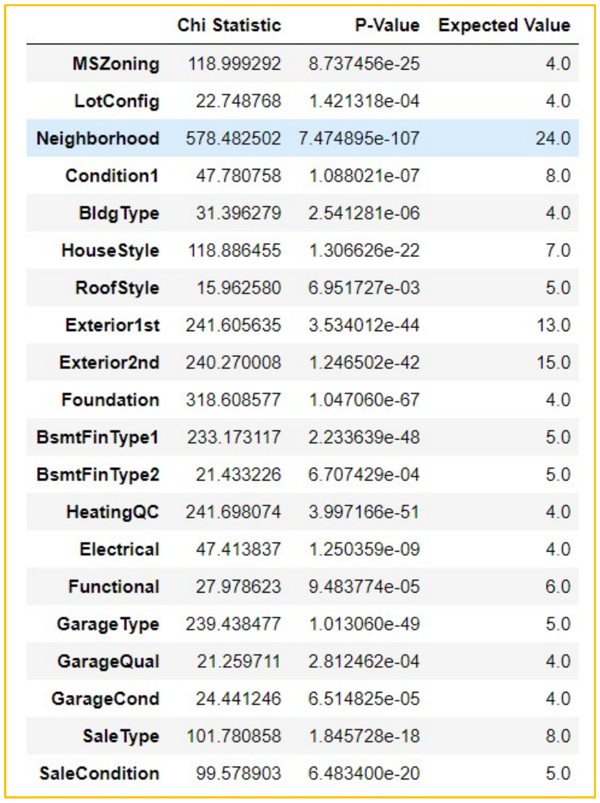


```
from collections import defaultdict
```

```
# Chi-Squared test on Categorical data in DF and filter out variables that have failed test:
res = defaultdict(list)
for x in objectcl:
    Chi_test(df1, x, 'SPcat')
    if Chi_test(df1, x, 'SPcat')[1] < .01 and Chi_test(df1, x, 'SPcat')[2] > 3:
        res[x].append(list(Chi_test(df1, x, 'SPcat')))
```

```
#view result in DataFrame:
dfres = pd.DataFrame.from_dict(res, orient='index')
dfres = dfres[0].apply(pd.Series)
columns = ['Chi Statistic','P-Value','Expected Value']
dfres.columns = columns
```

Next, I conducted the chi squared test of independence on each variable using the code below and append the results to a data frame that can be seen on the next page.

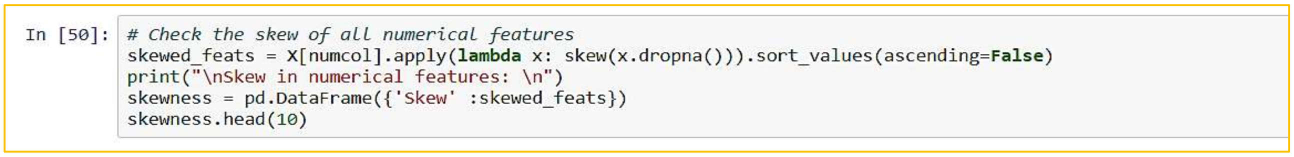


	Chi Statistic	P-Value	Expected Value
MSZoning	118.999292	8.737456e-25	4.0
LotConfig	22.748768	1.421318e-04	4.0
Neighborhood	578.482502	7.474895e-107	24.0
Condition1	47.780758	1.088021e-07	8.0
BldgType	31.396279	2.541281e-06	4.0
HouseStyle	118.886455	1.306626e-22	7.0
RoofStyle	15.962580	6.951727e-03	5.0
Exterior1st	241.605635	3.534012e-44	13.0
Exterior2nd	240.270008	1.246502e-42	15.0
Foundation	318.608577	1.047060e-67	4.0
BsmtFinType1	233.173117	2.233639e-48	5.0
BsmtFinType2	21.433226	6.707429e-04	5.0
HeatingQC	241.698074	3.997166e-51	4.0
Electrical	47.413837	1.250359e-09	4.0
Functional	27.978623	9.483774e-05	6.0
GarageType	239.438477	1.013060e-49	5.0
GarageQual	21.259711	2.812462e-04	4.0
GarageCond	24.441246	6.514825e-05	4.0
SaleType	101.780858	1.845728e-18	8.0
SaleCondition	99.578903	6.483400e-20	5.0

In conclusion, we have identified the variables that have a statistically significant relationship with Sale Price. From our analysis, the variables such as ‘GrLivArea’ and ‘Neighborhood’ share a strong relationship with sale price. We will use the variables we identified earlier in the report to develop a model that can best explain the variation is sale price.

Preprocessing and Model Construction

Now that the statistically significant predictor variables have been identified, The data needs to be prepped for model construction. We start by separating our predictors and target variables. Next, I identified skewed numerical features within the dataset and scaled them to improve the model prediction power. Using the code below, I was able to identify the skewed features in the data frame and use function called 'RobustScaler' to scale feature variables based on their median to minimize influence of outliers. Then, to avoid any multicollinearity issues in the final model, I added a function that goes through the feature variables and eliminates correlated features based on the calculated VIF. With numerical features scaled, I generate dummy variables for the categorical features in the dataset. Then split the dataset into a training and test sets.

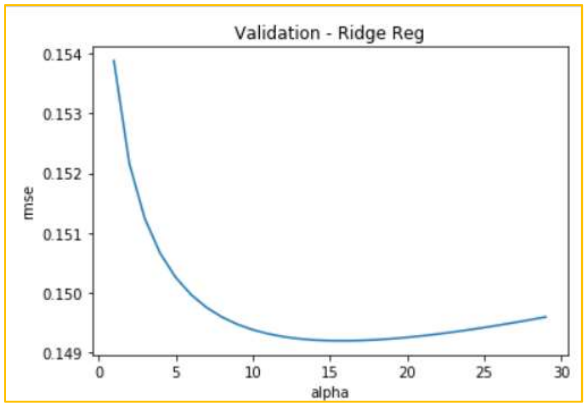


```
In [50]: # Check the skew of all numerical features
         skewed_feats = X[numcol].apply(lambda x: skew(x.dropna())).sort_values(ascending=False)
         print("\nSkew in numerical features: \n")
         skewness = pd.DataFrame({'Skew' :skewed_feats})
         skewness.head(10)
```

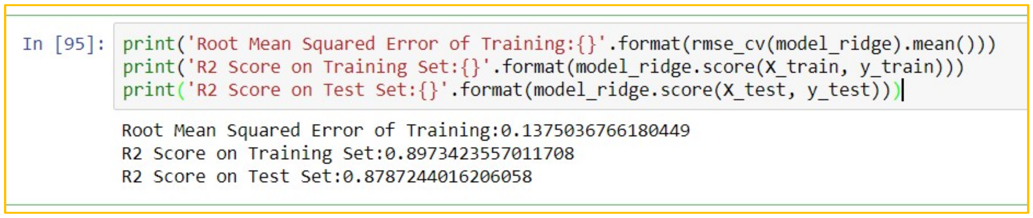
1st Model

In my first attempt, I tried to use Ridge Regression algorithm. This algorithm tries to retain all the features but shrinks coefficients. Since I have identified the variables to be statistically significant, I'm comfortable using a model of this complexity (many features). Before we train the model, I wrote a function that will evaluate the model using root mean squared error. This will provide an additional metric to compare different models. I began model construction by optimizing the hyper-parameter, The graph below shows the change in mean squared error across different alphas.

Ridge regression validation curve



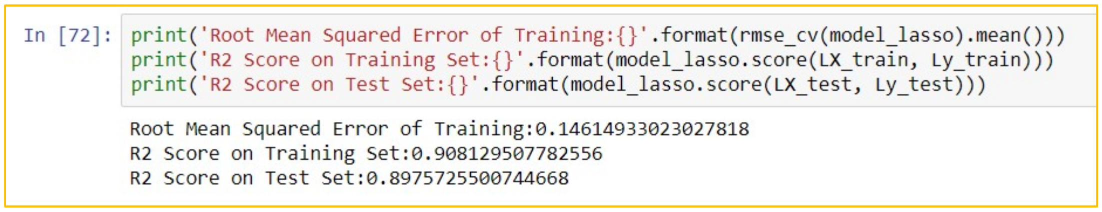
After I have identified the best parameter, I fit the model and used it to predict on the training and test set. The measure metrics listed below, shows that our model suffers from some overfitting. We will need to try and use a different algorithm that can provide more feature flexibility.



```
In [95]: print('Root Mean Squared Error of Training:{}'.format(rmse_cv(model_ridge).mean()))
         print('R2 Score on Training Set:{}'.format(model_ridge.score(X_train, y_train)))
         print('R2 Score on Test Set:{}'.format(model_ridge.score(X_test, y_test)))

         Root Mean Squared Error of Training:0.1375036766180449
         R2 Score on Training Set:0.8973423557011708
         R2 Score on Test Set:0.8787244016206058
```

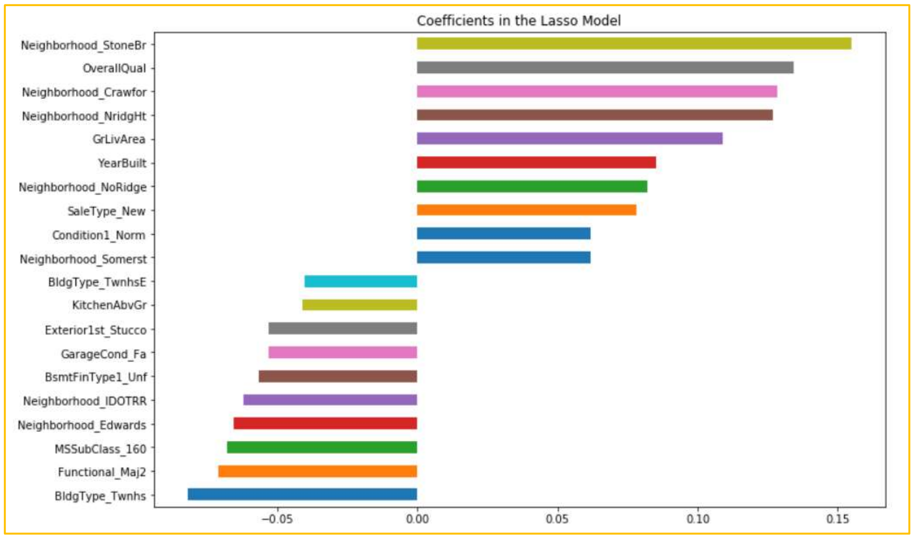
2nd Model

Moving away from Ridge Regression, I wanted to try a lasso regression algorithm to predict sale price. While fitting the model, Lasso regression algorithm automatically does coefficient reduction and feature selection. This algorithm should offer better predictions since it will use a subset of Key features to make prediction. As done previously, I optimized the hyper-parameters for the lasso algorithm and then fit the model to the training data. This model offered a slightly worse score on the training set but better prediction on the test set. Also, the root mean squared error decreased as well. The chart below shows the top feature selected for predictions.



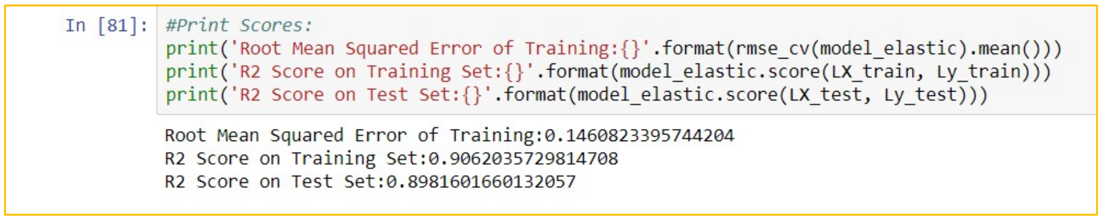
```
In [72]: print('Root Mean Squared Error of Training:{}'.format(rmse_cv(model_lasso).mean()))
         print('R2 Score on Training Set:{}'.format(model_lasso.score(LX_train, Ly_train)))
         print('R2 Score on Test Set:{}'.format(model_lasso.score(LX_test, Ly_test)))

         Root Mean Squared Error of Training:0.14614933023027818
         R2 Score on Training Set:0.908129507782556
         R2 Score on Test Set:0.8975725500744668
```

Feature Coefficient of Lasso Model



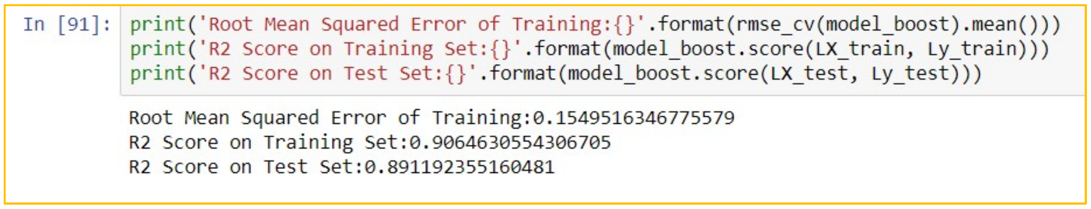
3rd Model and Boosting

The lasso model performed better than the ridge model on the test set, but I believe that we could increase the explained variation by using a Hybrid of both model. Elastic Net Algorithm can provide that hybrid model. I start by optimizing the parameters then fit the model to the training data. This time the prediction on the test set slightly decreased and our error decreased as well.



```
In [81]: #Print Scores:
         print('Root Mean Squared Error of Training:{}'.format(rmse_cv(model_elastic).mean()))
         print('R2 Score on Training Set:{}'.format(model_elastic.score(LX_train, Ly_train)))
         print('R2 Score on Test Set:{}'.format(model_elastic.score(LX_test, Ly_test)))

         Root Mean Squared Error of Training:0.1460823395744204
         R2 Score on Training Set:0.9062035729814708
         R2 Score on Test Set:0.8981601660132057
```

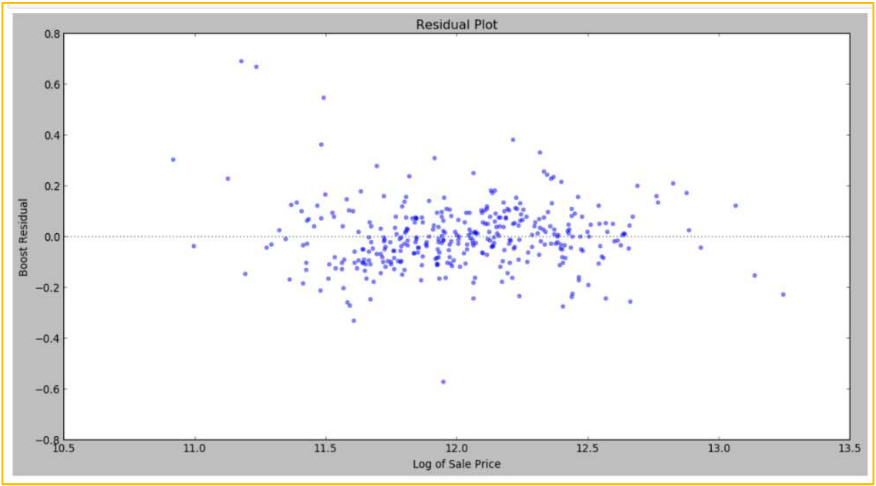
At this point, our models power to predict on the test set hasn't significantly increased. In one last attempt to improve upon the last model, I attempted to use the Ada boost algorithm to increase the prediction power of the model. The Ada boost algorithm fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner until a predefined limit is reached in the number of models or accuracy. I will use Elastic net as my base estimator with slightly different parameters. I optimize the parameters for Ada boosting algorithm and fit using the training data. My last attempt failed to improve on the elastic model.



```
In [91]: print('Root Mean Squared Error of Training:{}'.format(rmse_cv(model_boost).mean()))
         print('R2 Score on Training Set:{}'.format(model_boost.score(LX_train, Ly_train)))
         print('R2 Score on Test Set:{}'.format(model_boost.score(LX_test, Ly_test)))

         Root Mean Squared Error of Training:0.1549516346775579
         R2 Score on Training Set:0.9064630554306705
         R2 Score on Test Set:0.891192355160481
```

The residual plot for the elastic model below gives a visual representation of how well the elastic model has performed on un seen date. Notice the residuals a scattered somewhat evenly around the horizontal zero line, but there is an extreme outlier. Despite this outlier, I was still able to construct an effective model.

<u>Residuals</u>



Key Insights

1. **<u>Know the Squared Footage</u>**
    a. Variables associated with the living space of the home turned out to be one of the strongest predictor of Sale price.. Naturally the more space you have in the home the higher the sale price will be.
2. **<u>Building Type and Year built matter</u>**
    a. 2 story family homes tend to sell for higher prices, while 1 story homes built prior to 1945 are priced lower. Year Built has a positive correlation with sale price and should be considered when determining sale price.
3. **<u>Location, Location, Location</u>**
    a. Some of the strongest predictors in the tied to the location of the home. Be mindful of the neighborhood and zone your home is located in. Depending on what neighborhood or zone your home is located, Sale price is greatly influenced. Within the data set homes that lied within the neighborhood of North Ridge and North Ridge Height had some of the highest sale prices on average. Also homes that reside within a Residential low-density zone had high sale prices as well.
4. **<u>We have a good model</u>**
    a. The model constructed during my analysis is able to account for almost all variation between predicted and actual values. This model can be used with confidence to predict home sale