House Prices: Advanced Regression Techniques

Springboard Capstone Project #1

Introduction and Objective

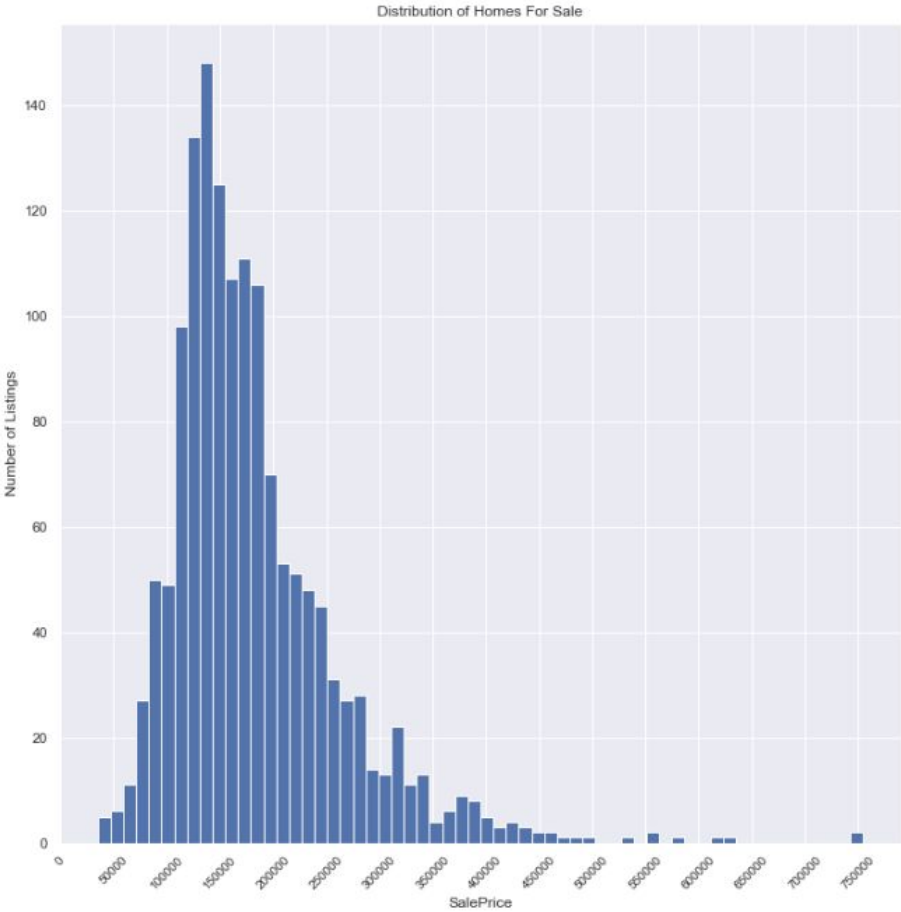
The following dataset, Ames Housing dataset was compiled by Dean De Cock, it is a modernized and expanded version of the Boston Housing Dataset. It can be found from Kaggle, https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview.

The objective of this Capstone project is to predict the sale price of each house in correlation to its features by using different machine learning algorithms to find the lowest root mean squared logarithmic error. There are a total of 79 explanatory describing almost every aspect of residential homes in the Ames area. Buyers in the real estate industry can use this tool to find the features they are looking for in a house and match it with a price. Sellers in the real estate industry can use this to determine the cost of their home and identify which features have a bigger impact on the sale price.

Throughout the project, there will be various tasks performed such as:

- Data analysis - To conceptualize the data, remove unnecessary variables, deal with missing data and handle outliers
- Inferential Statistics - This will help to observe between homes that have 2 bedrooms above grade versus 3 bedrooms above grade.
- Machine Learning Models - Linear regression, random forests, decision tree regressor, extra trees regressor and gradient boosting regressor will be used to determine the training accuracy, root mean squared error (RMSE), root mean squared logarithmic error (RMSLE) and R-squared.
- Model Tuning - Hyperparameter Tuning will yield a lower error.

Data Analysis



The histogram above shows an overview of the corresponding data set. The lowest sale price of a house is $34, 900 and the highest price is $755, 000. The majority of the sale prices are towards the lower end of the spectrum, with an average sale price of $180, 921. There are only a few sales prices that are over $500, 000.

There are several columns that have missing values. The 79 columns have the following data types; float64, int64 and object. From the columns that have missing data, only one of the columns “LotFrontage” was a float64 thus all the missing data were filled with a mean value. For the rest of the missing data, the columns were objects thus they were all replaced with 'Not Available'. The missing data were replaced with ‘Not Available’ because forward filling and backward filling these values would be wrong for that particular home, this would compile incorrect data when predicting the sale price of a house from its features.

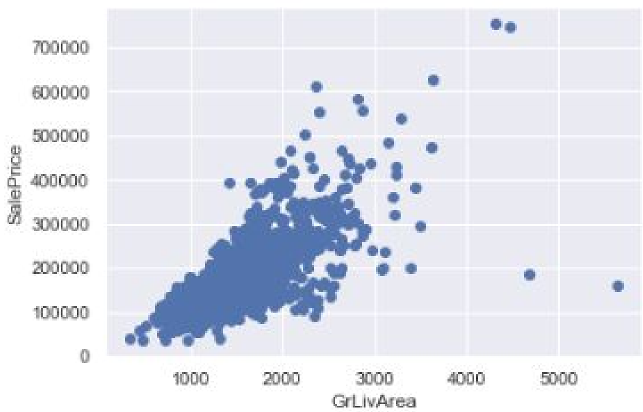
Sale Price Correlation

```
Id               -0.021917
MSSubClass       -0.084284
LotArea           0.263843
OverallQual       0.790982
OverallCond      -0.077856
YearBuilt         0.522897
YearRemodAdd      0.507101
BsmtFinSF1        0.386420
BsmtFinSF2       -0.011378
BsmtUnfSF         0.214479
TotalBsmtSF       0.613581
1stFlrSF          0.605852
2ndFlrSF          0.319334
LowQualFinSF     -0.025606
GrLivArea         0.708624
BsmtFullBath      0.227122
BsmtHalfBath     -0.016844
FullBath          0.560664
HalfBath          0.284108
BedroomAbvGr      0.168213
KitchenAbvGr     -0.135907
TotRmsAbvGrd      0.533723
Fireplaces        0.466929
GarageCars        0.640409
GarageArea        0.623431
WoodDeckSF        0.324413
OpenPorchSF       0.315856
EnclosedPorch    -0.128578
3SsnPorch         0.044584
ScreenPorch       0.111447
PoolArea          0.092404
MiscVal          -0.021190
MoSold            0.046432
YrSold           -0.028923
SalePrice         1.000000
Name: SalePrice, dtype: float64
```

```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
1stFlrSF        0.605852
FullBath        0.560664
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
YearRemodAdd    0.507101
MasVnrArea      0.475241
GarageYrBlt     0.470177
Fireplaces      0.466929
BsmtFinSF1      0.386420
LotFrontage     0.334901
WoodDeckSF      0.324413
2ndFlrSF        0.319334
OpenPorchSF     0.315856
HalfBath        0.284108
LotArea         0.263843
BsmtFullBath    0.227122
BsmtUnfSF       0.214479
BedroomAbvGr    0.168213
ScreenPorch     0.111447
PoolArea        0.092404
MoSold          0.046432
3SsnPorch       0.044584
Name: SalePrice, dtype: float64
```

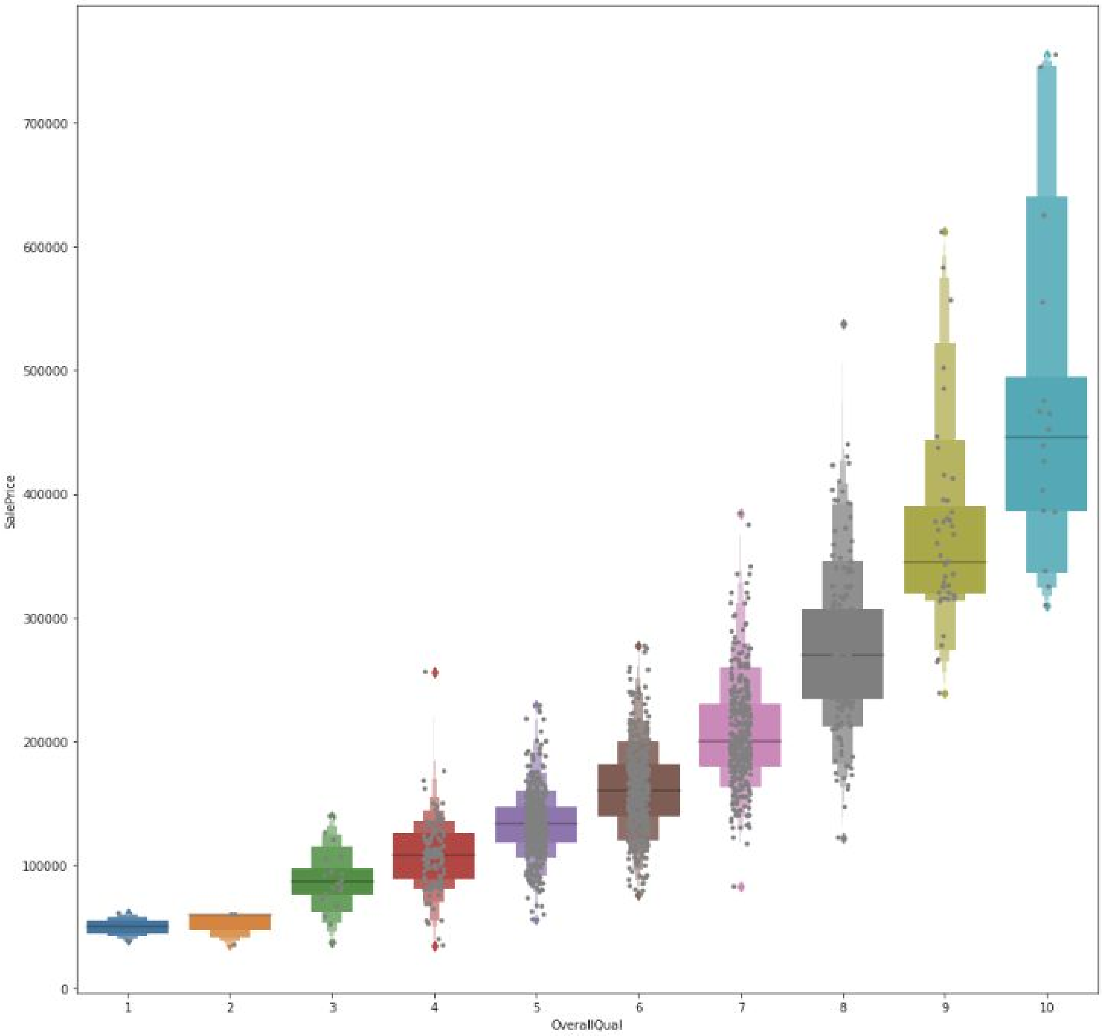
The list above on the left shows the correlation of the sales price towards the following features of a house. All of the columns that had a negative correlation to the sale price were also deleted as they were not relevant in determining the sales price. The list on the right is the updated correlation after the negative values were dropped.

Outliers

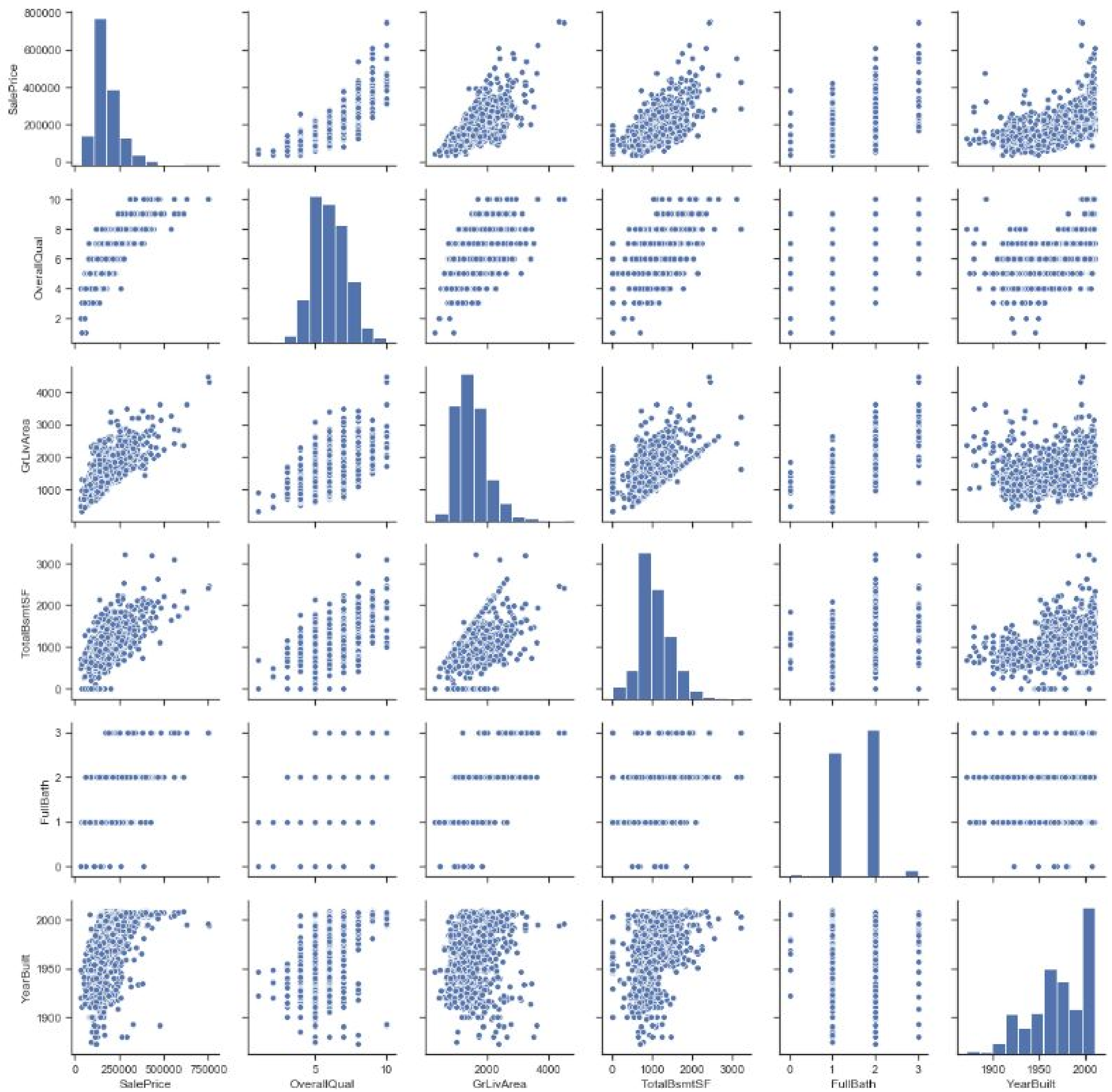


The graph above shows a plot of SalePrice Vs. GrLivArea. There are a total of 9 outliers with a deviation that is greater than 3.5 or less than -3.5. Out of the 9 outliers, there are two that are far greater outliers than the others with a deviation of 6.01 and 7.86. In the graph above the two significant outliers are the two points located on the bottom right, these two points are significantly away from the rest of the data. Thus both points are removed.

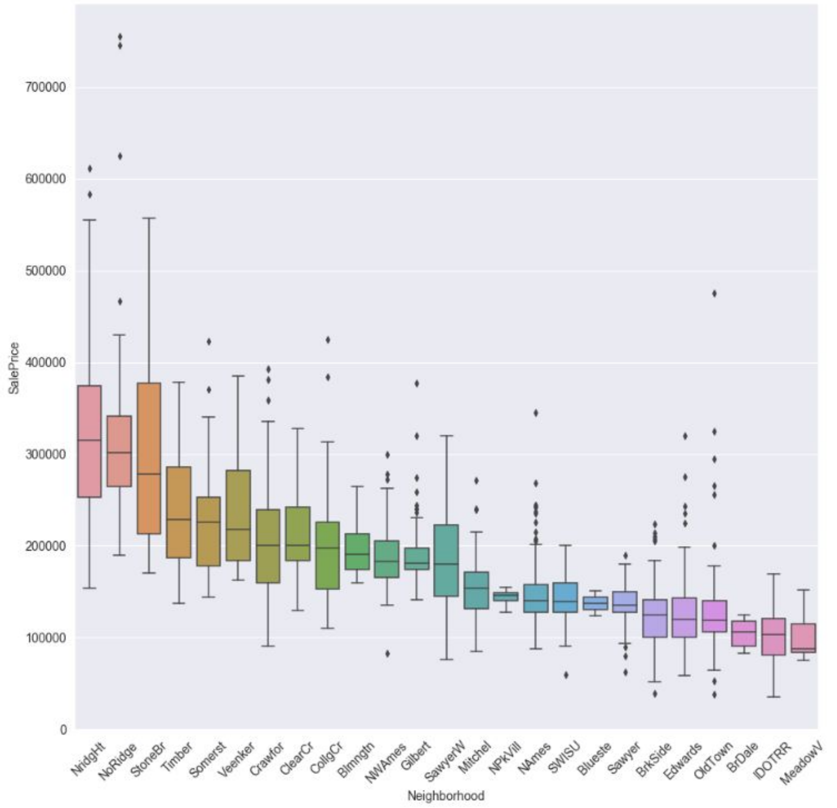
Data Exploration



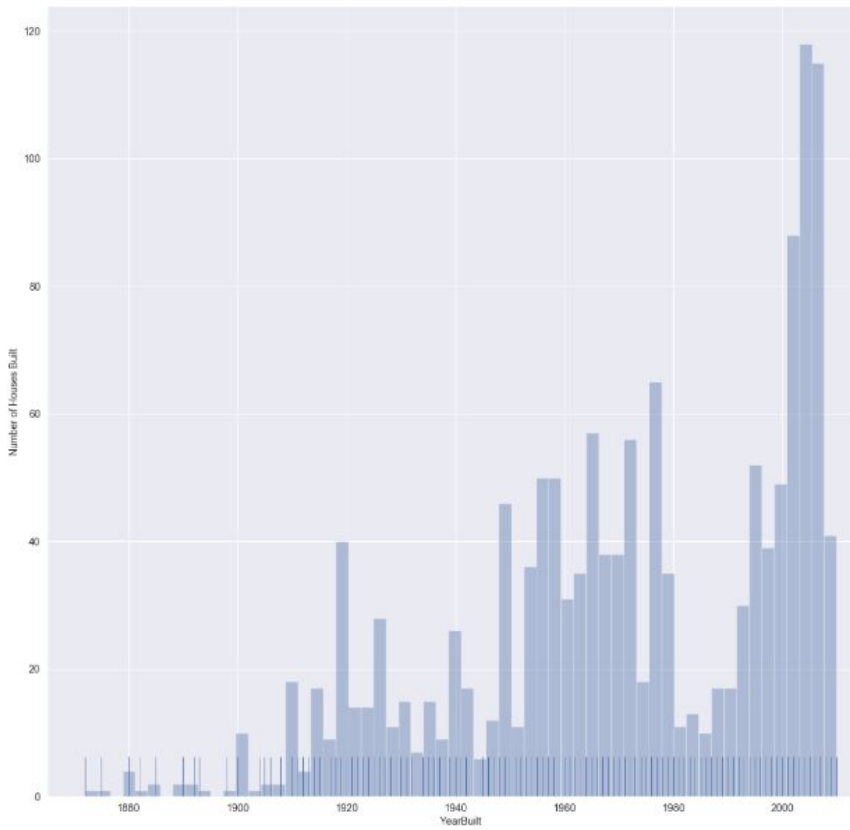
The box plot above shows a plot of the individual sale price of a house versus the overall quality of each house. The gray horizontal line in each box plot represents the average sale price for each overall quality. On average, as the overall quality increases the sale price increases with it.

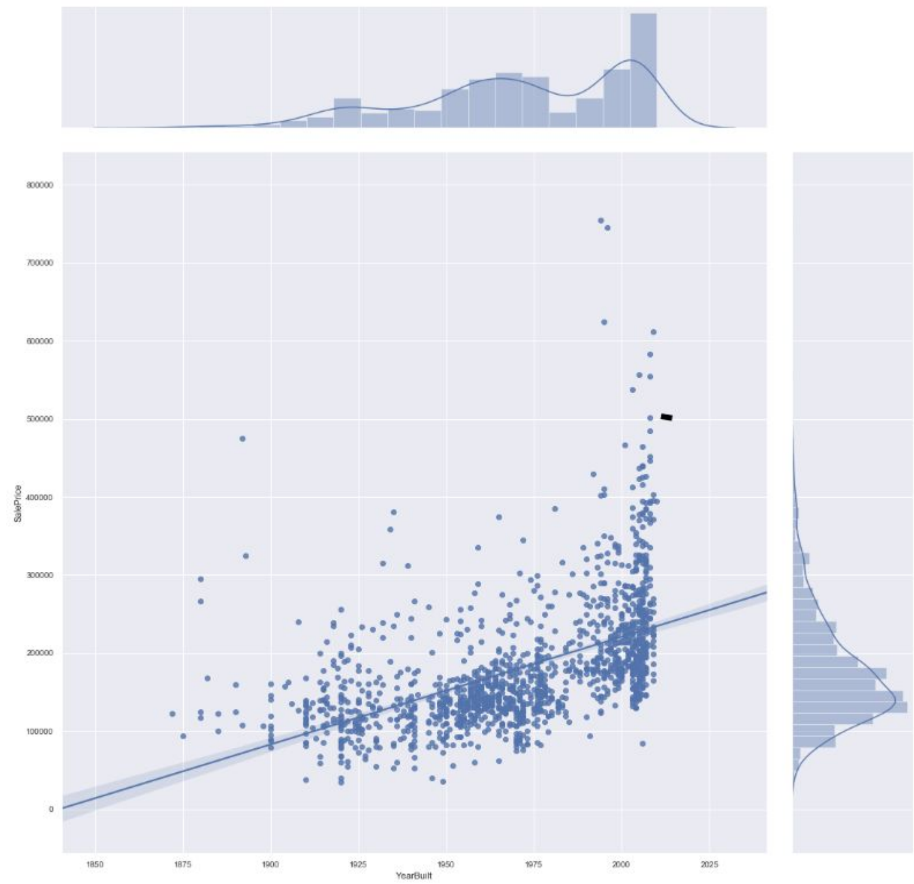


The pair plot above shows the correlation between the sale price of each house, the overall quality (OverallQual), above grade living area in square feet (GrLivArea), total square feet of basement area (TotalBsmtSF), full bathrooms above grade (FullBath) and the original construction date (YearBuilt). In the first row, it shows the correlation of the sales price towards the other features. Visually there is a linear increase between each feature and the sales price. Several graphs in the pair plot above show linear relationships.



The above box plot graph shows the sales prices for each of the neighborhoods in the given data. The neighborhood NridgHt has the highest average sales price. That said, the neighborhood NoRidge has three of the most expensive homes.



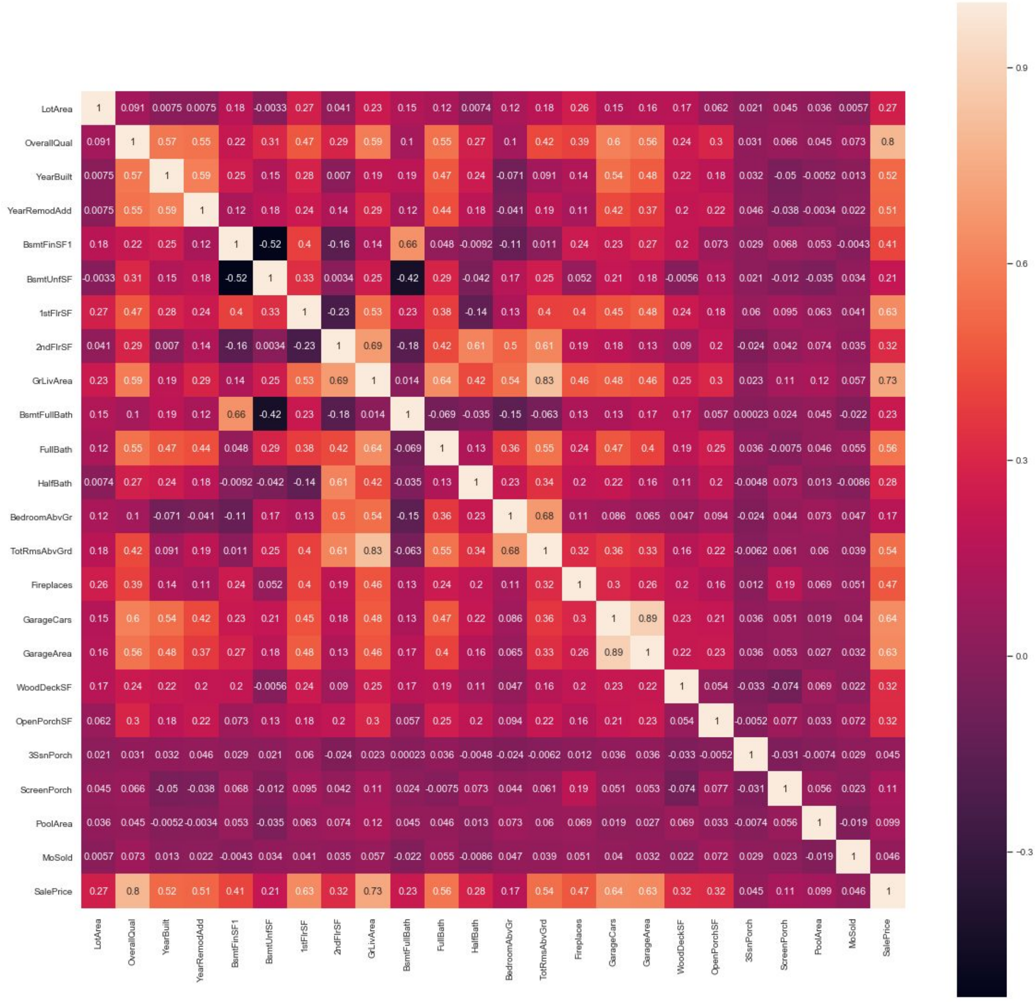


The histograms above show the number of homes built by the year. On average the homes built are increasing every 2 decades. The joint plot of SalePrice Vs. year built shows as increasing linear regression. Prices are higher for new homes.

Statistical Data Analysis

<u>Correlation Matrix</u>

The heat map of a correlation matrix below shows a good representation of the correlation between each feature of the house. The most important is the SalePrice correlation towards the rest of the features with the highest correlation being 0.8 for OverallQual and the lowest at 0.045 for 3SsnPorch. The light red means high correlation and the darker red to black means lower correlation to inverse correlation.



In [44]:
```
train_df[['SalePrice', 'BedroomAbvGr']].describe()
```

Out[44]:

	SalePrice	BedroomAbvGr
count	1458.000000	1458.000000
mean	180932.919067	2.866255
std	79495.055285	0.816323
min	34900.000000	0.000000
25%	129925.000000	2.000000
50%	163000.000000	3.000000
75%	214000.000000	3.000000
max	755000.000000	8.000000

In [35]:
```
two_bedroom = train_df.SalePrice.loc[train_df.BedroomAbvGr == 2]
three_bedroom = train_df.SalePrice.loc[train_df.BedroomAbvGr == 3]
```

In [36]:
```
import scipy
scipy.stats.ttest_ind(two_bedroom, three_bedroom)
```

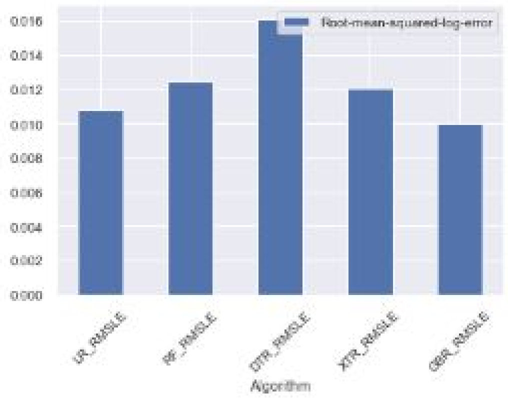
Out[36]:
```
Ttest_indResult(statistic=-5.246983309011471, pvalue=1.8374907162225815e-07)
```

The above shows code for a t-test. The t-test observes homes with 2 bedrooms and 3 bedrooms above grade. Since the p-value is less than 0.05, we reject the null hypothesis. This is because the average sales price for a 2 bedroom house will be lower than that of a 3 bedroom house.

Machine Learning

The purpose of this project is to explore advanced regression techniques, so I will be using linear regression, random forests, decision tree regressor, extra trees regressor and gradient boosting regressor models to evaluate which of the following gives the best RMSLE results. Before using the machine learning models I scaled the features using robust scaler. Robust scaler was the best option because it is suitable for data with outliers. Below is a list of the five machine learning models I have used.

Linear Regression Model
Training accuracy: 0.947952228094347 Root-mean-squared error: 0.1380591560566311 Root-mean-squared-log-error: 0.010839457597885448 Mean-squared-error: 0.019060330571069223 R-squared: 0.8869350967152927
Random Forests Model
Training accuracy: 0.947952228094347 Root-mean-squared error: 0.15740947759552357 Root-mean-squared-log-error: 0.01240498717784635 Mean-squared-error: 0.02477774363689564 R-squared: 0.8530196956724805
Decision Tree Regressor Model
Training accuracy: 0.947952228094347 Root-mean-squared error: 0.2073968217128347 Root-mean-squared-log-error 0.016137377328512947 Mean-squared-error: 0.043013441656585334 R-squared: 0.7448464703846212
Extra Trees Regressor Model
Training accuracy: 0.947952228094347 Root-mean-squared error: 0.15150073877132939 Root-mean-squared-log-error 0.011961564354778064 Mean-squared-error: 0.02295247384825859 R-squared: 0.863847102434984
Gradient Boosting Regressor Model
Training accuracy: 0.947952228094347 Root-mean-squared error: 0.12735835790425343 Root-mean-squared-log-error 0.010032782763355334 Mean-squared-error: 0.016220151328067912 R-squared: 0.903782894303263



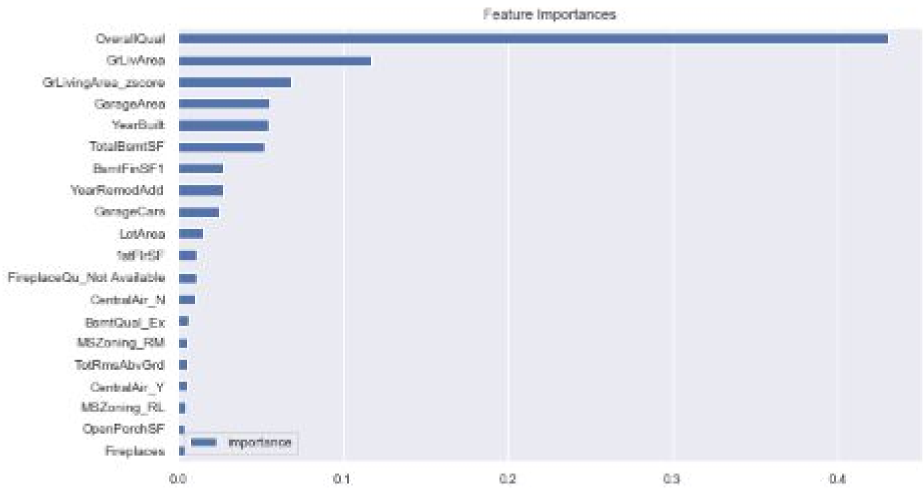
The histogram above shows the difference in RMSLE for the following machine learning models. Gradient Boosting Regressor gave the best results for RMSE and RMSLE. This is the model that will be tuned using hyperparameter tuning to get better results.

Hyperparameter Tuning

Before Hyperparameter Tuning Gradient Boosting Regressor	After Hyperparameter Tuning Gradient Boosting Regressor
Training accuracy: 0.947952228094347 Root-mean-squared error: 0.12735835790425343 Root-mean-squared-log-error 0.010032782763355334 Mean-squared-error: 0.016220151328067912 R-squared: 0.903782894303263	Training accuracy: 0.947952228094347 Root-mean-squared error: 0.11924170052390301 Root-mean-squared-log-error: 0.009355913564395696 Mean-squared-error: 0.014218583143832172 R-squared: 0.9156560941055718

After hyperparameter tuning the gradient boosting regressor, the root mean squared logarithmic error and root mean squared error was significantly reduced.

Feature Importances for Gradient Boosting Regressor



The figure above shows the top 20 most important features for the model. The overall quality feature had significant importance on the model compared to the rest of the features.

Conclusion

From the following machine learning models used in this project, the gradient boosting regressor gave the best results with an RMSLE value of 0.00936. This value cannot be compared to the results on the Kaggle scoreboard because for my project I only used the training data. That said, the value I have obtained is far lower than the results others have got on the scoreboard.