BUAN 6340 Programming for Data Science (Python)

Project on

**Kaggle: House Prices Prediction**

Presented by

Arpit Chaukiyal

Komal Sasane

Surya Thummanapelly

Vidheesha Kudipudi

**1. Executive Summary**

The intent of this project report is to provide the information on various machine learning techniques used to predict the house prices in Ames. The report documents data collection, data description exploratory data analysis, models and finally the conclusion. The dataset has been taken from Kaggle website. With 79 explanatory variables describing (almost) every aspect of 1460 residential homes in Ames, Iowa, the goal is to predict the final price of other 1459 homes. Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad, however this dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

To perform predictive analysis on the dataset, we first cleanse the data using techniques such as removing null values, converting datatypes of the attributes, checking for duplicate records, log transforming ‘y’ variable etc. Feature Engineering is the process that increases the predictive power of machine learning algorithms by creating features from raw data. By using feature engineering technique, we created new input features from existing one.

We then explore the data to see hidden patterns and trends such as seasonality of the houses sold, % distribution of type of houses sold, most and least influencing factors for house sales, correlation factor between numerical features and sale price etc. We then apply supervised machine learning techniques on train dataset. In our project, we have used linear regression, KNN regression, Lasso and Ridge, Decision tree, Random forest algorithms. Eventually, we compare the results based on various parameters i.e., accuracy score, root mean square error value, r square value etc. By choosing the appropriate algorithm, we apply that model to predict sale price of the houses.

**2. Project Motivation/Problem Statement**

**2-1 Inputs & Outputs:**

**2-1-1 Inputs**

* train.csv - the training set
* test.csv - the test set

**2-1-2 Outputs**

Sale prices for every record in test.csv

**2-2 Target Feature:**

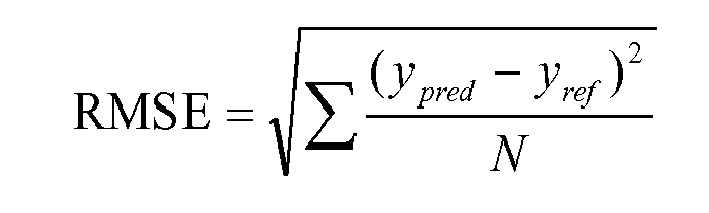
We will use the house prices data set. This dataset contains information about house prices and the target value is: SalePrice

**2-3 Objective:**

Based on training and testing dataset, we predict the **log of sale prices** for every record in test.csv for each Id in the test set.

**2-4 Evaluation Metric:**

Submissions are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. (Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.)



**3. Data Description**

We are using training and testing dataset for this project. Training Data contains 1460 unique labelled observations and 81 features including y variable ‘SalePrice’. Data contains 1459 unique and new unlabeled observations with 80 features. Each observation represents a house. The dataset has 53 numerical (Ratio and Ordinal) features, 26 categorical features and 1 identity feature.

Data fields and their description:

* SalePrice - the property's sale price in dollars. This is the target variable that we are 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

**4. Data Transformation**

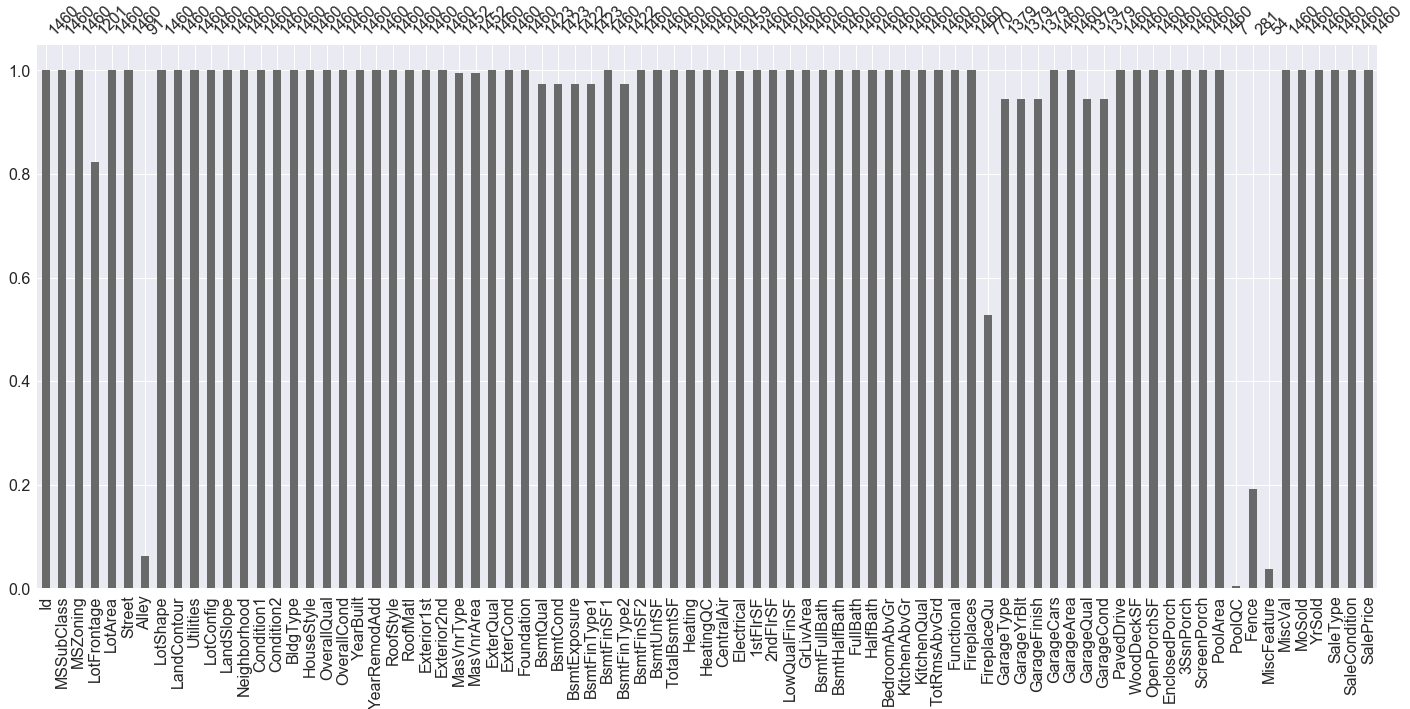
We identified the categorical variables, ordinal variables, and numeric variables based on the given “data\_description.txt” file.

**4-1 Uncommon Features Between Train and Test datasets:**

Initially, we compared the feature names in Train & Test datasets. The only different feature is ‘SalePrice’ (Target Variable), which is anticipated. So, all the other features are present in both datasets.

**4-2 Handling Missing Data:**

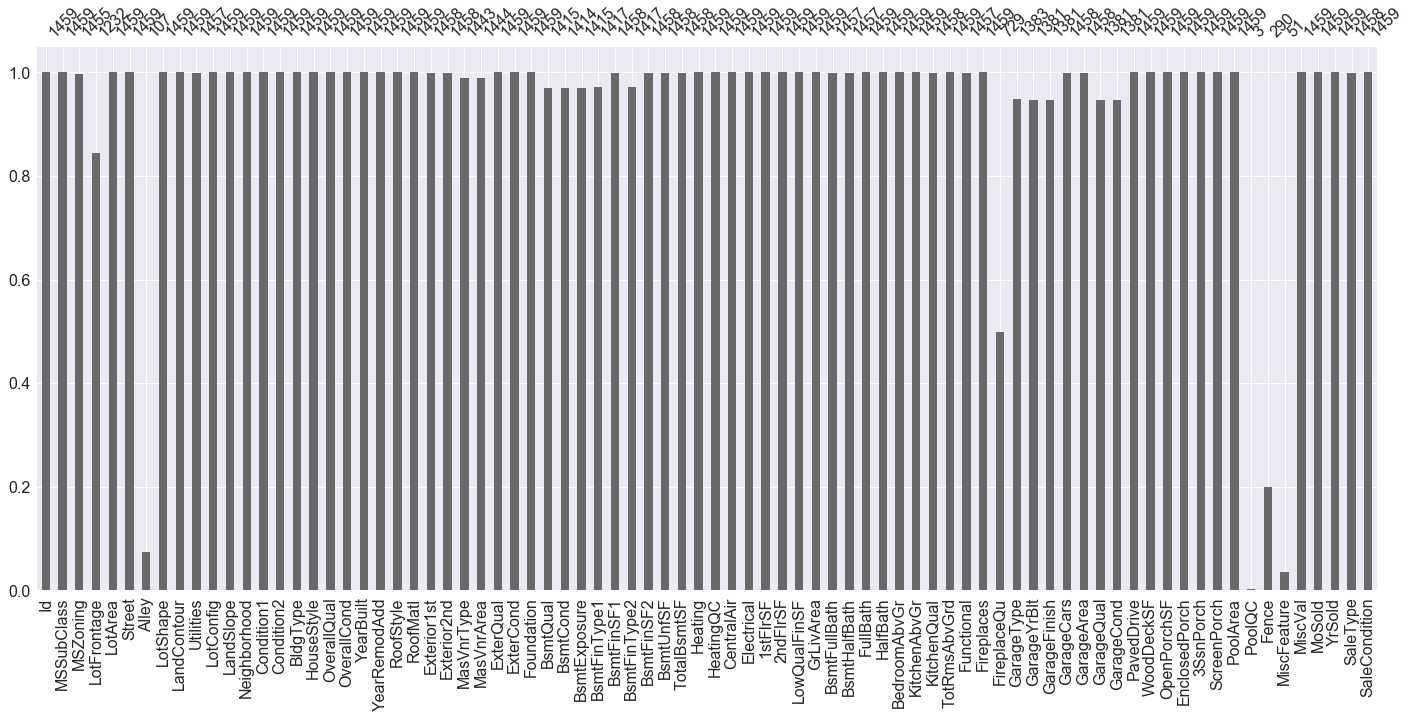
We checked if there are any missing observations in the training & test data sets. Following are the visualizations of the missing observations in the respective datasets.



**Missing Observations in Training data**

From above visualization, it seems like **Alley, FieplaceQuality, PoolQC, Fence, & MiscFeature** columns have more than 50% missing observations. But “data\_description.txt” defines that missing values in most of the features indicate that the feature is in fact not applicable for the record. So, we have handled the missing data case-by-case (feature wise).

We replaced missing values in almost all the features with “None” instead of imputing them with “median” or “mean”. We performed this cleanup process on both training & test data sets in parallel. Then we made sure that no more observations are missing in either datasets.



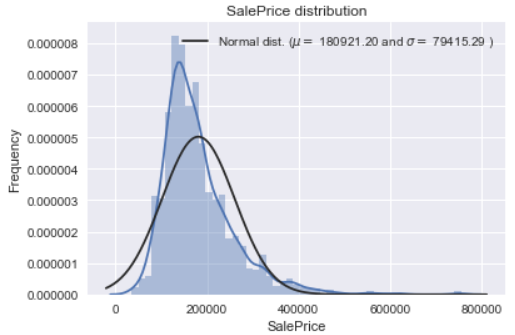
**Missing Observations in Test data**

**4-3 Duplicate Records:**

Then we checked if there are any duplicate records within the Training and Test datasets on “Id” column. There are no duplicate records both within the datasets and across the datasets.

**4-4 Skewness:**

Most of the Machine Learning algorithm perform better when the data is normally distributed. Though, algorithms like SVM does not hold these assumptions, we wanted to make sure the data is Normal. We observed that the target variable, ‘SalePrice’ is highly positively skewed.



**SalePrice Original Distribution**

To make the ‘SalePrice’ distribution Normal, we applied “Log Transformation”. The resulting distribution looked more like Normal, as shown below:



**Log Transformed SalePrice Distribution**

Similarly, we checked skewness of all the features after further “Feature Engineering” and applied “Log Transformation” to all the highly right-skewed features (Skewness > +0.5). There were no left-skewed features.

**4-5 Feature Engineering:**

There are 4 Date fields in the dataset, namely **YrSold, YearBuilt, YearRemodAdd,** and **GarageYrBlt.** These Date fields are by nature categorical, so we created 3 new features in terms of age to make use of these Date fields like numerical features. Following are the new features:

ageHouse = YrSold – YearBuilt

ageRemodel = YrSold – YearRemodAdd

ageGarage = YrSold – GarageYrBlt

We dropped 1 record in training data and 2 records in test data, where **ageRemodel** is greater than the **ageHouse** with an assumption of bad data.

**4-5-1 Label Encoding:**

Some of the numerical features are categorical as per the data description. For ex. **MSSubClass** lies between 20 & 190 but each number just represents the class of the house.

Similarly, we converted some ordinal features to Likert Scale (1 to 5 rating) since the ordering of these values contain information. Then again, we created bins to convert 1-5 scale of these features to 1-3 scale to represent a simplified **Bad, Avergae, and Good** scale.

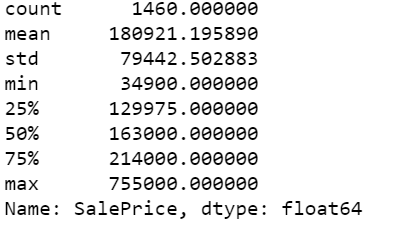
**4-5-2 Dummy Variables:**

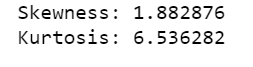
For all the categorical variables, we created dummy binary variables and dropped the first dummy to avoid Multi-Collinearity. While creating these dummy variables, we had to combine training and test datasets, because some of the features don’t have all the classes/levels in one dataset. After the dummy variable creation, we separated the datasets again.

**5. Exploratory Data Analysis**

**5-1 Summary statistics of Sale Price:**

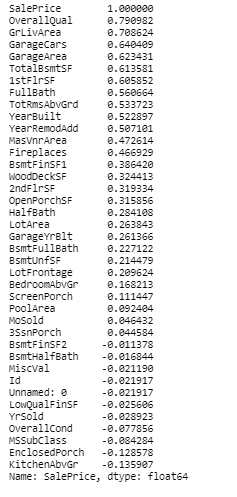
We used the describe() function to get various summary statistics of our target variable SalePrice. We also found the skewness and Kurtosis.





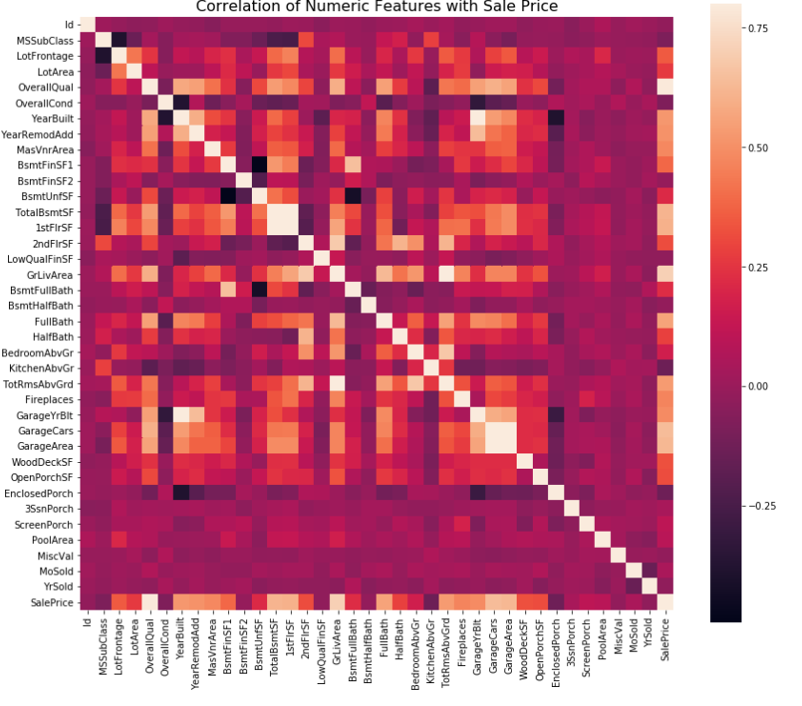
**5-2 Correlation coefficients:**

We then calculated the Correlation coefficients between Sale Price and the numeric features and got the following results -

****

**5-2-1 Correlation Heat Map**

To explore further we started with a correlation heat map visualization to analyze the data better.



**6. Predictive Modeling**

Darker shades depict low positive correlation and lighter shades depict high positive correlation.

At initial glance it is observed that there are two red colored squares that attract attention.

1. The first one is 'TotalBsmtSF' and '1stFlrSF' variables.
2. Second one refers to the 'GarageX' variables. Both cases show how significant the correlation is between these variables. This correlation is so strong that it can indicate a situation of multicollinearity. If we think about these variables, we can conclude that they give almost the same information, so multicollinearity really occurs.

Heatmaps are great to detect this kind of multicollinearity situations and in problems related to feature selection like this project, it comes as an excellent exploratory tool.

Another aspect observed here is the 'SalePrice' correlations. As it is observed that 'GrLivArea', 'TotalBsmtSF', and 'OverallQual' having a high correlation to SalePrice, however we cannot exclude the fact that rest of the features have some level of correlation to the SalePrice. To observe this correlation closer let us see it in Zoomed Heat Map.

**5-2-2 SalePrice Zoomed Heat Map:**

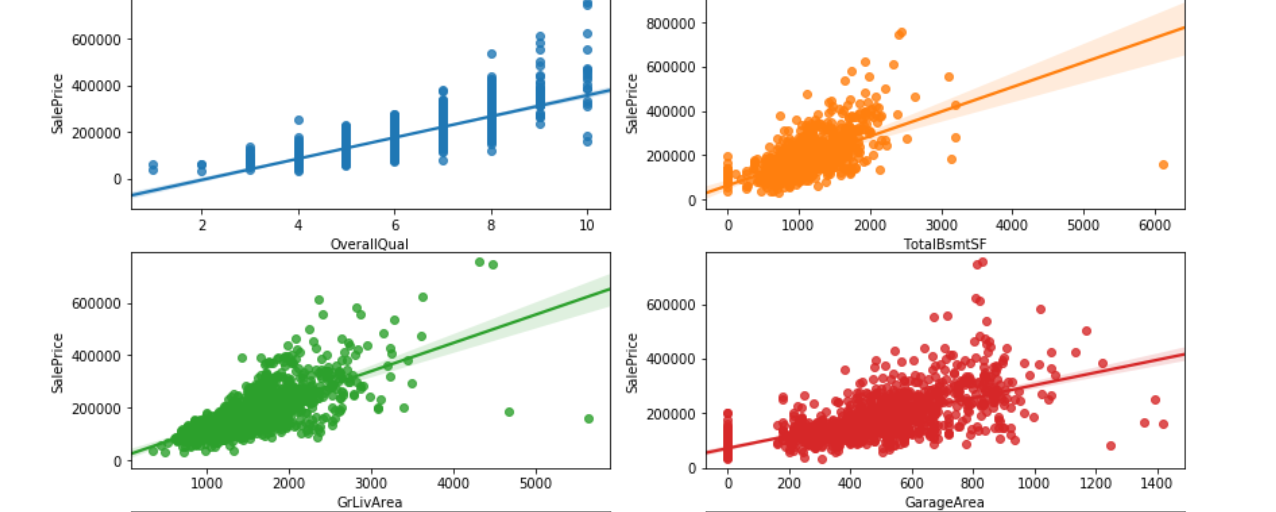


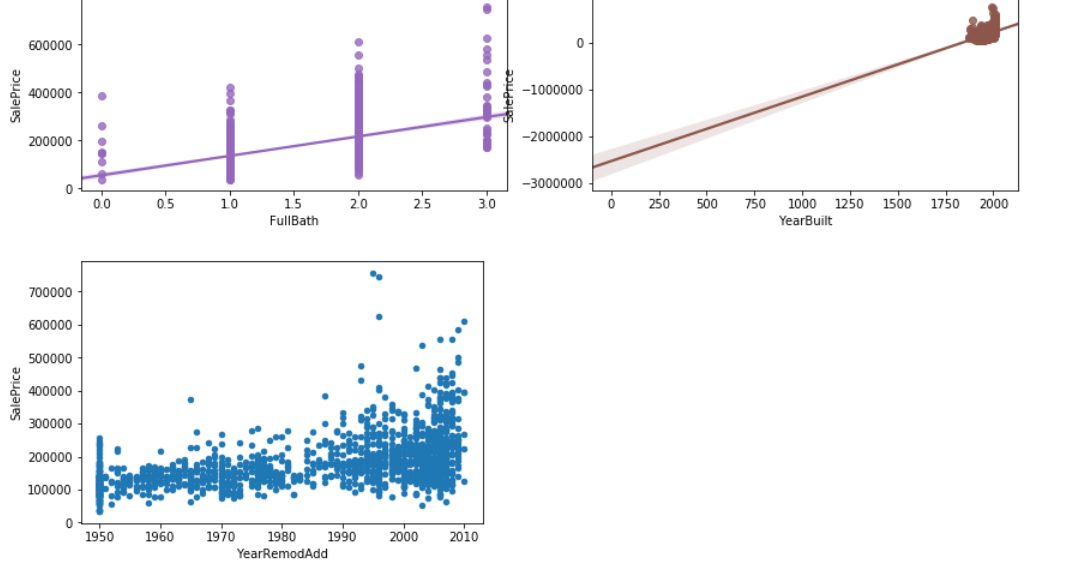
Blue colors depict low positive correlation and green and yellow depict high positive correlation.

From above zoomed heatmap it is observed that GarageCars & GarageArea are closely correlated. Similarly, TotalBsmtSF and 1stFlrSF are also closely correlated.

We created more visualizations to dig deeper into the correlation.

**5-3 Scatter plots between the most correlated variables:**

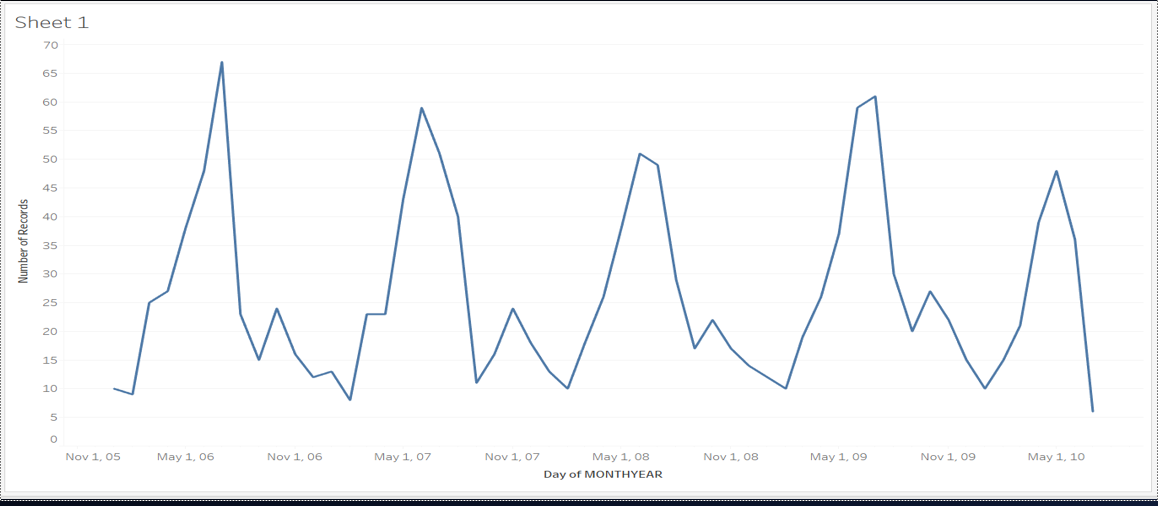




In this graph, the sub plots represent the scatter plots between Sale price and the 7 most correlated variables- **OverallQual, TotalBsmtSF, GrLivArea, GarageArea, FullBath, YearBuilt, YearRemodAdd**.

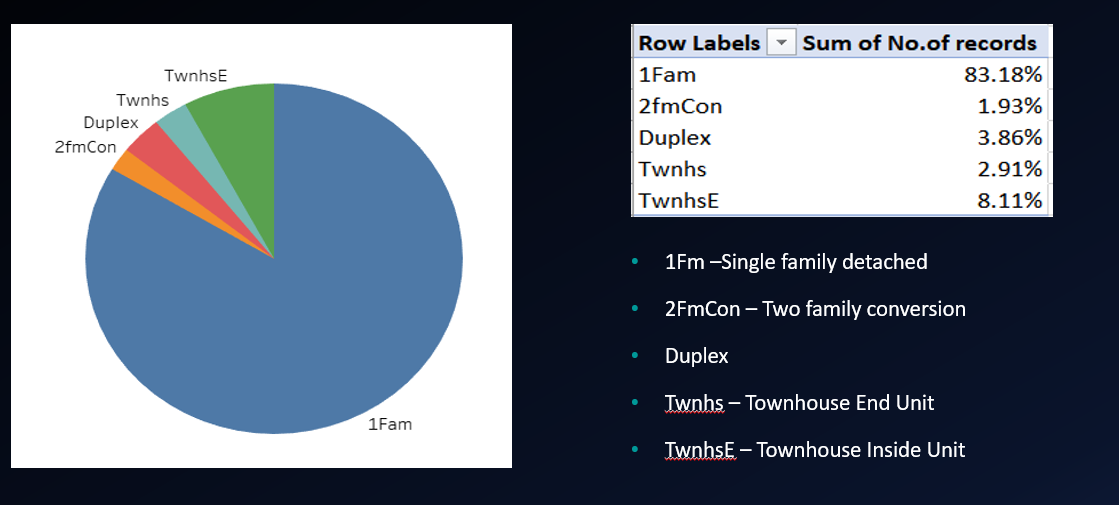
We explore the data further using more graphs to get more insights.

**5-4 Seasonality of Houses sold:**



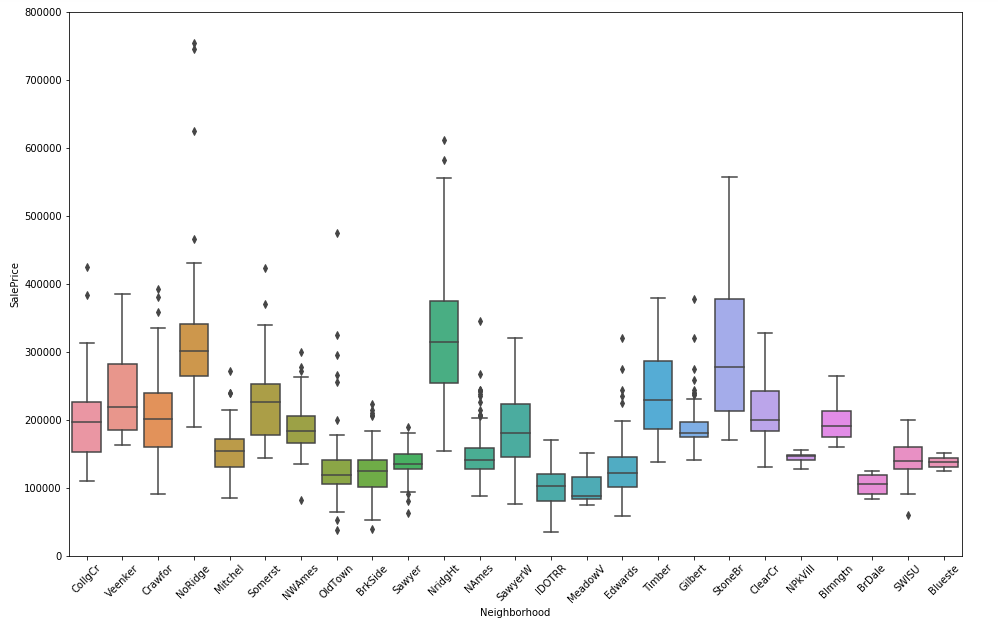
This line graph describes the no. of houses sold in the months of every year. From the graph , we can see that the houses were predominantly sold during the summer i.e. in the months of May , June, July.

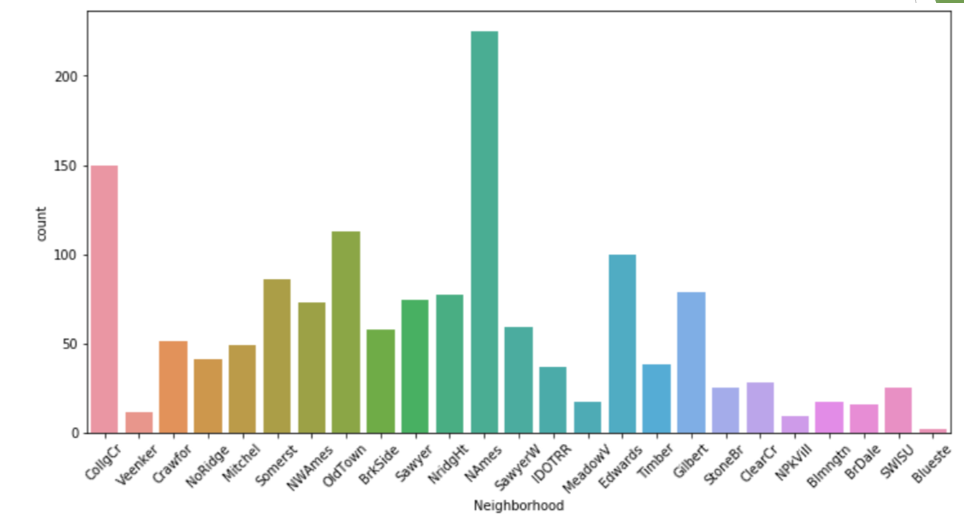
**5-5 Type of houses sold:**



From this pie chart, we can see that the most houses sold were single family detached houses.

**5-6 Boxplot and Histogram of Neighborhood:**

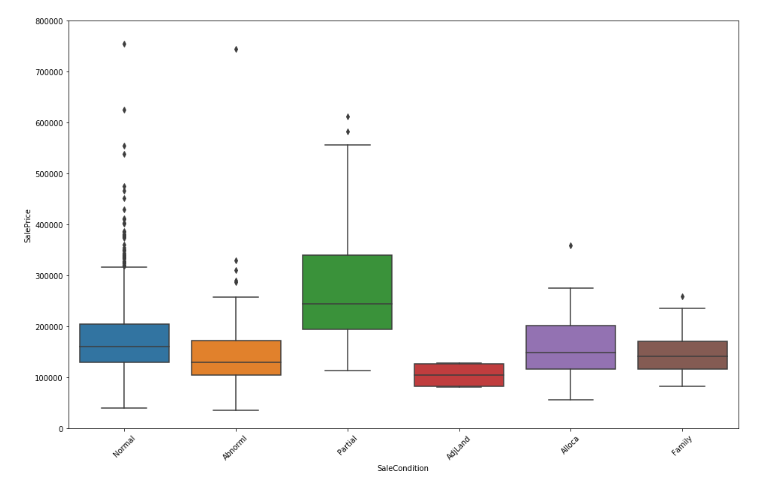


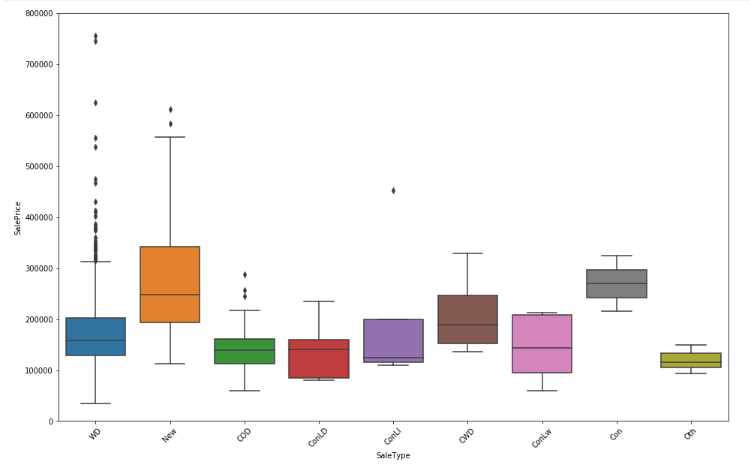


From these graphs, we can see that “Names” was the most popular neighborhood with the most number of houses sold and Blueste was the least popular neighborhood with the least number of houses sold.

**5-7 More Boxplots:**

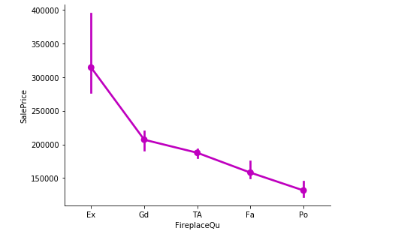
To explore the SaleCondition and SaleType variables further, we created box plots with SalePrice.



****

**5-8 Line graph of FireplaceQu:**

We also created a Line Graph to explore the FireplaceQu variable.



**6. Predictive Modeling**

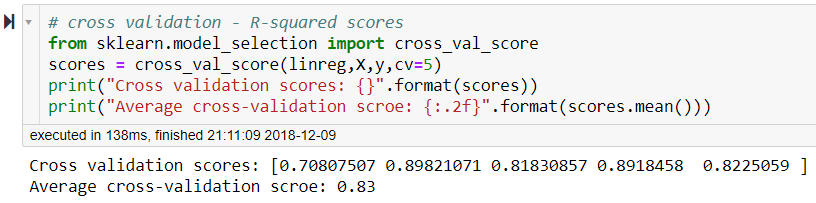
**6-1 Preparing X, Y data:**

We stored ‘SalePrice’ data in data frame **y** and remaining all features in data frame **X.** Then, the datasets are divided into **train** and **test** datasets with 75:25 ratio respectively.

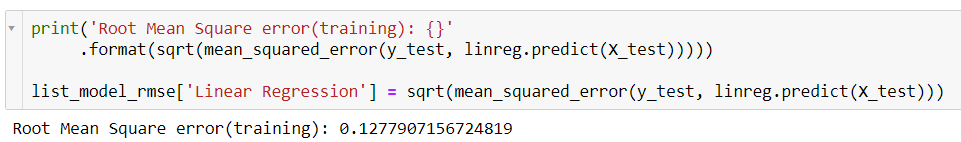
**6-2 Linear Regression:**

We performed simple Linear Regression initially on the **train** dataset. Then we used **Cross-Validation** with 5 folds on the entire **X, y** datasets. Following are the metrics from the Linear Regression model with CV.

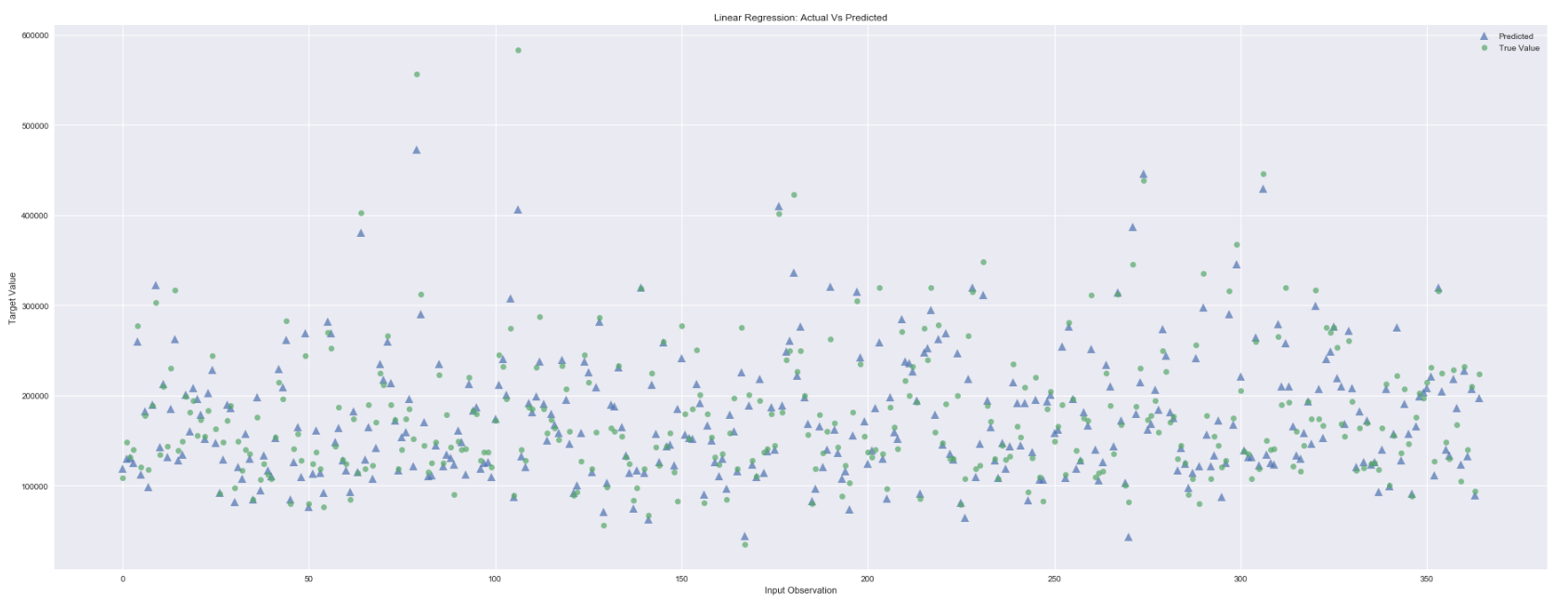
* Cross-Validation Score (R-Squared): 0.83
* RMSE: 0.127



**Cross-Validation Score of Linear Regression**



**RMSE Output of Linear Regression – test split**



**Residual Plot of Linear Regression**

**6-3 Standardizing Data:**

To use the data in next algorithms, we applied **StandardScaler** on the **train** and **test** datasets separately, to get all the numerical and ordinal features on same scale/range.

The standard score of a sample x is calculated as:

**z = (x - u) / s**

where u is the mean of the training and s is the standard deviation of the training samples.

**6-4 KNN Regression:**

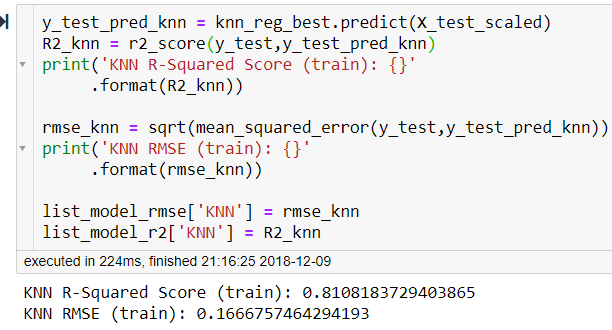
KNN Regression is applied on the Standardized data (**X\_scaled**) using **GridSearch** and **Cross-Validation**. GridSearch is used to identify the best hyperparameters for the model. Following are the hyperparameters tuned:

* + 'n\_neighbors’ : [1,2,3,4,5,7,10,11,12,15]

GridSearch returned best hyper-parameter as n\_neighbors = 11

So, we trained another KNN Regressor model using the hyper-parameter on **X\_train\_scaled** and evaluated on **X\_test\_scaled** data. Following are the metrics:

* R-Squared Value – Test Split: 0.81
* RMSE: 0.166



**R-Squared Score & RMSE of KNN**

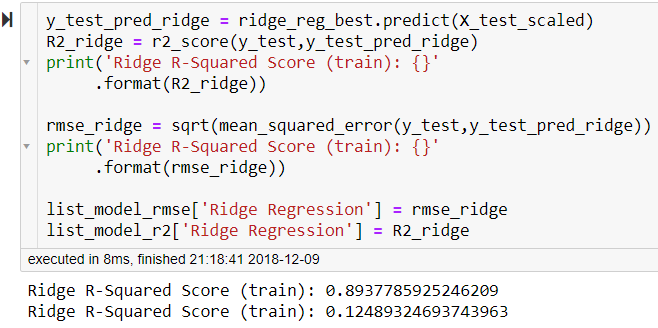
**6-5 Ridge Regression:**

We performed Ridge Regression on the scaled data using **Cross-Validation** and **GridSearch** with 5 folds with following hyper-parameters (learning rate):

* + 'alpha': [0.001, 0.01, 0.1, 1, 10, 100]

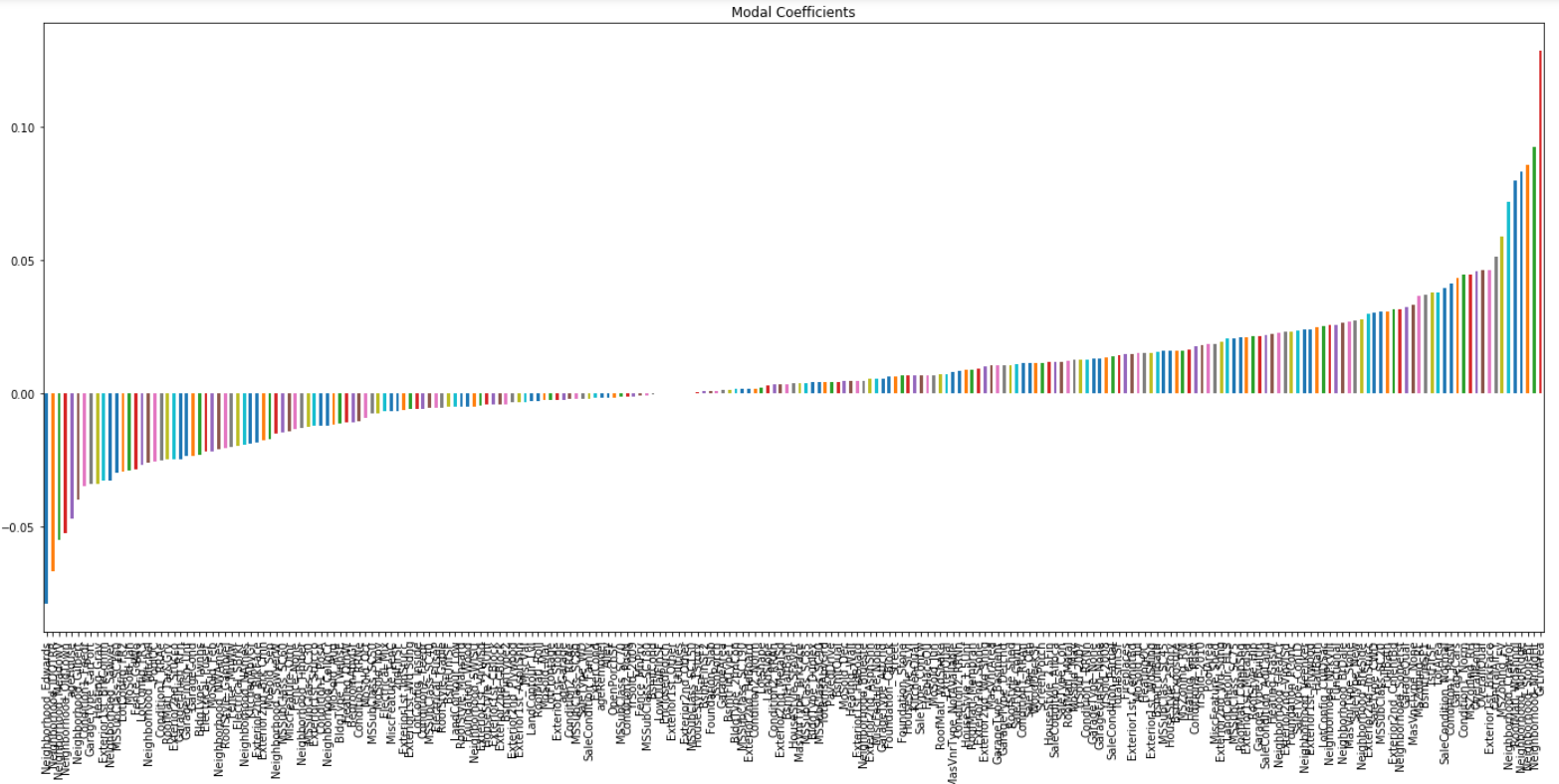
GridSearch returned best hyper-parameter as alpha = 10. Following are the metrics:

* R-Squared Value – Test Split: 0.893
* RMSE: 0.124



**R-Squared Score & RMSE of Ridge Regression**

Ridge Regression uses **L2 Regularization**, through which it penalizes the unimportant features. As we can see in the below plot, that Ridge regression tried to reduce the coefficients of unimportant variables compared to important variables. Some features in the center have 0 coefficients because of the definite decimal precision



**Coefficients of Features – Ridge – L2 Regularization**

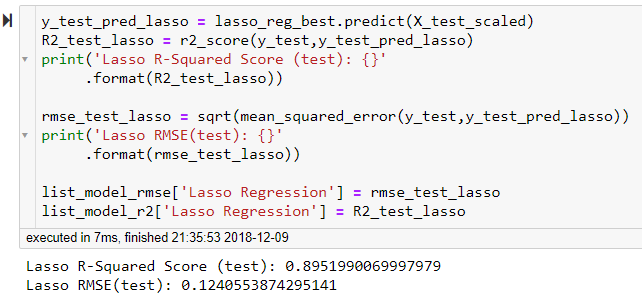
**6-6 Lasso Regression:**

We performed Lasso Regression on the scaled data using **Cross-Validation** and **GridSearch** with 5 folds and following hyper-parameters:

* + 'alpha':[0.0001, 0.0003, 0.0006, 0.001, 0.003, 0.006, 0.01, 0.03, 0.06, 0.1, 0.3, 0.6, 1],
  + 'max\_iter':[10000, 50000]

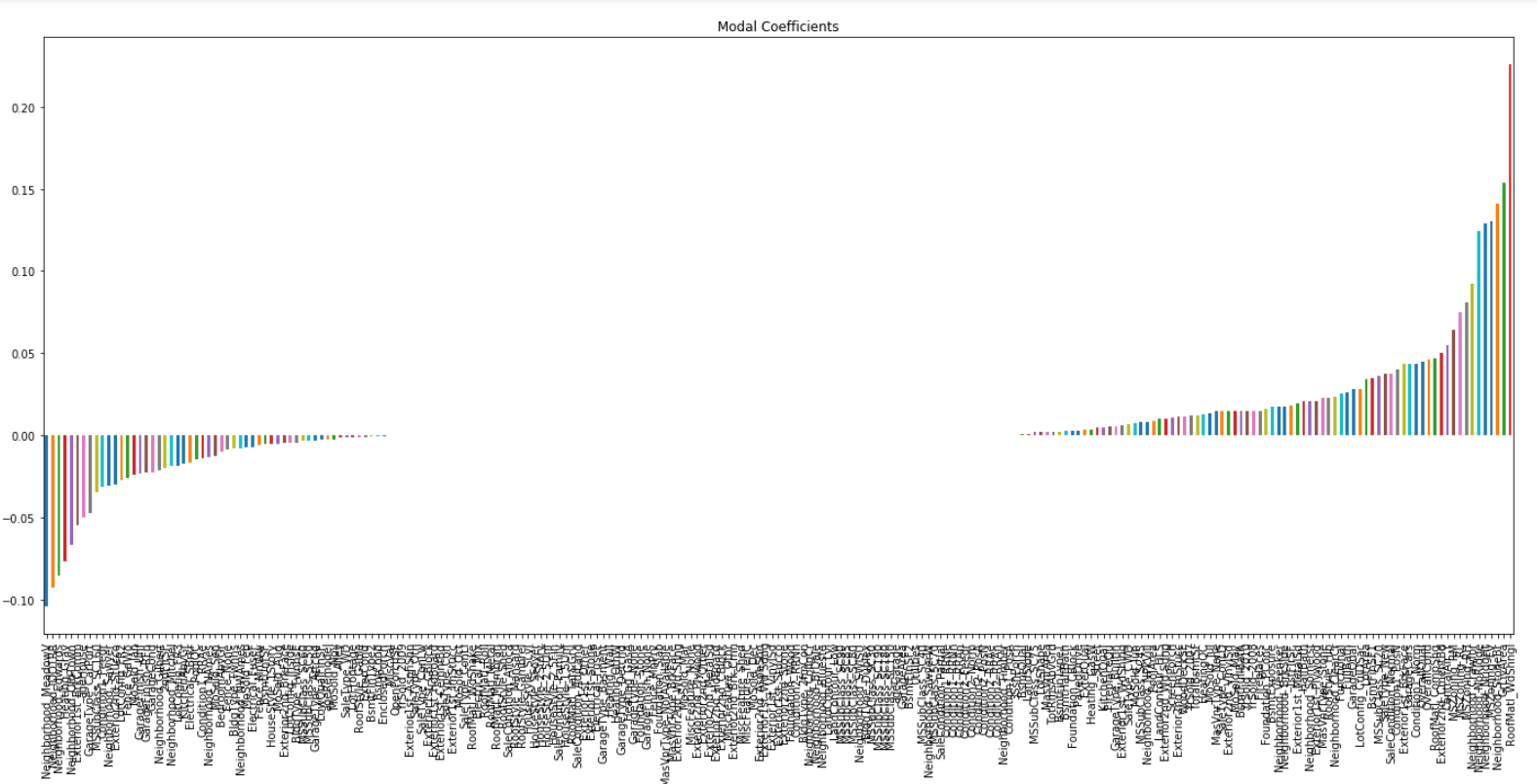
GridSearch returned best hyper-parameters as alpha = 0.0003 & max\_iter = 10K

* R-Squared Value – Test split: 0.895
* RMSE: 0.124

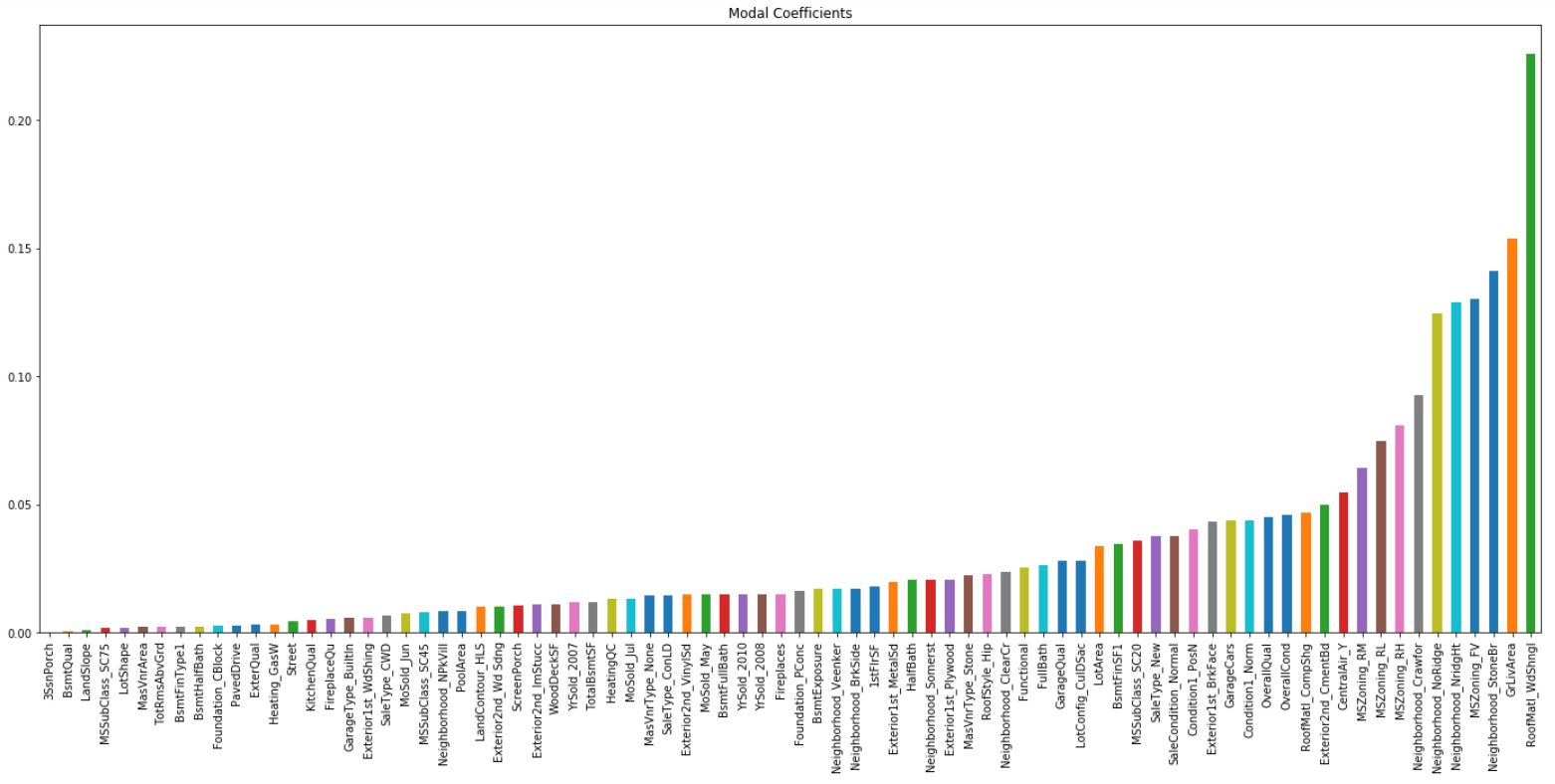


**R-Squared Score & RMSE of Lasso Regression**

Lasso Regression uses **L1 Regularization**, through which it penalizes the unimportant features. This penalty is more compared to L2 regularization, as a result Lasso can eliminate unimportant features by assigning 0 coefficient. As we can see in the below plot, that Lasso regression tried to assign the coefficients of unimportant variables to 0.



**Coefficients of Features – Lasso – L1 Regularization**



**Selected Features – Lasso – L1 Regularization**

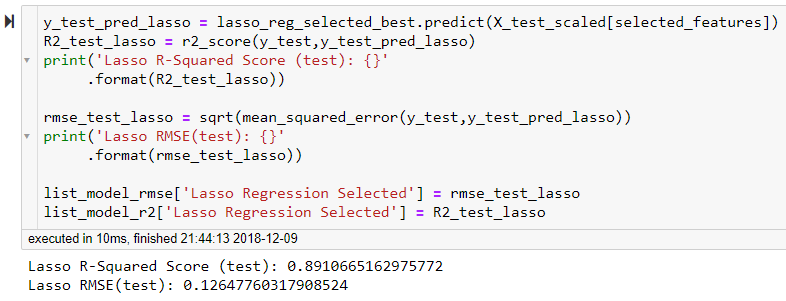
So, Lasso selected just 80 features out of 235 total features. Then we ran Lasso regression using only these selected features.

**6-7 Lasso Regression on Selected Features:**

We ran the Lasso regression by sub-setting the selected 80 features and following are the metrics:

* R-Squared Value – Test split: 0.891
* RMSE: 0.126

Even though, these scores are not better than the Lasso with all features model, the difference is not significant considering this model used only 80 features out of 235.



**R-Squared Score & RMSE of Lasso Regression – Selected Features**

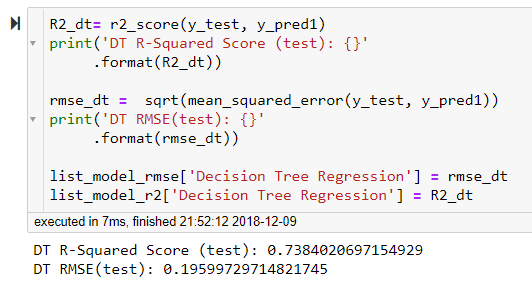
**6-8 Decision Trees Regression:**

We ran Decision Tree Regression on the scaled data using **Cross-Validation and GridSearch** with 5 folds with following hyper-parameters:

* + 'max\_depth': [1,2,3,4,5,6,7,8,9,10]
  + 'min\_samples\_leaf': [1,2,3,4,5,6,7,8,9,10]

GridSearch returned best hyper-parameters as max\_depth = 6 & min\_samples\_leaf = 8

* R-Squared Value – Test Data: 0.738
* RMSE: 0.195



**R-Squared Score & RMSE of Decision Tree Regressor**

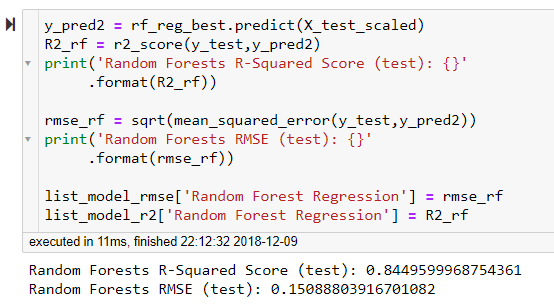
**6-9 Random Forests Regression:**

We applied Random Forest Regression on the scaled data using **Cross-Validation and GridSearch** with 5 folds with following hyper-parameters:

* + 'n\_estimators': [46,47,48,49,50,51,52],
  + 'min\_samples\_split': [11,12,13],
  + 'min\_samples\_leaf' : [4,5,6],
  + 'max\_depth' : [5,6,7,8,9,10]

GridSearch returned best hyper-parameters as max\_depth = 10, min\_samples\_leaf = 4, min\_samples\_split = 11, n\_estimators = 51

* R-Squared Value – Test Data: 0.844
* RMSE: 0.15



**R-Squared Score & RMSE of Random Forests Regressor**

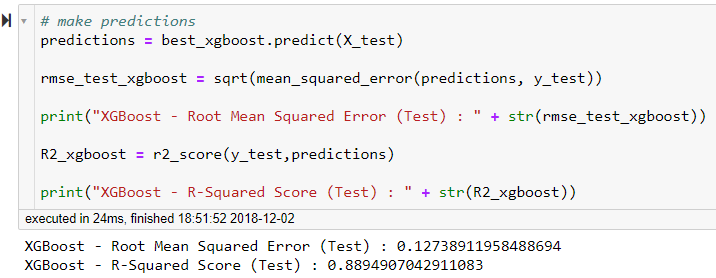
**6-10 XGBoost Model:**

We applied XGBoost model on the unscaled data using **Cross-Validation and GridSearch** with 5 folds with following hyper-parameters:

* + 'n\_estimators': [100,500,1000],
  + 'learning\_rate':[0.02, 0.05, 0.1]

GridSearch returned best hyper-parameters as learning\_rate = 0.02, n\_estimators = 1000

* R-Squared Value – Test split: 0.889
* RMSE: 0.127



**R-Squared Score & RMSE of XGBoost**

**7. Findings**

**7-1 Findings:**

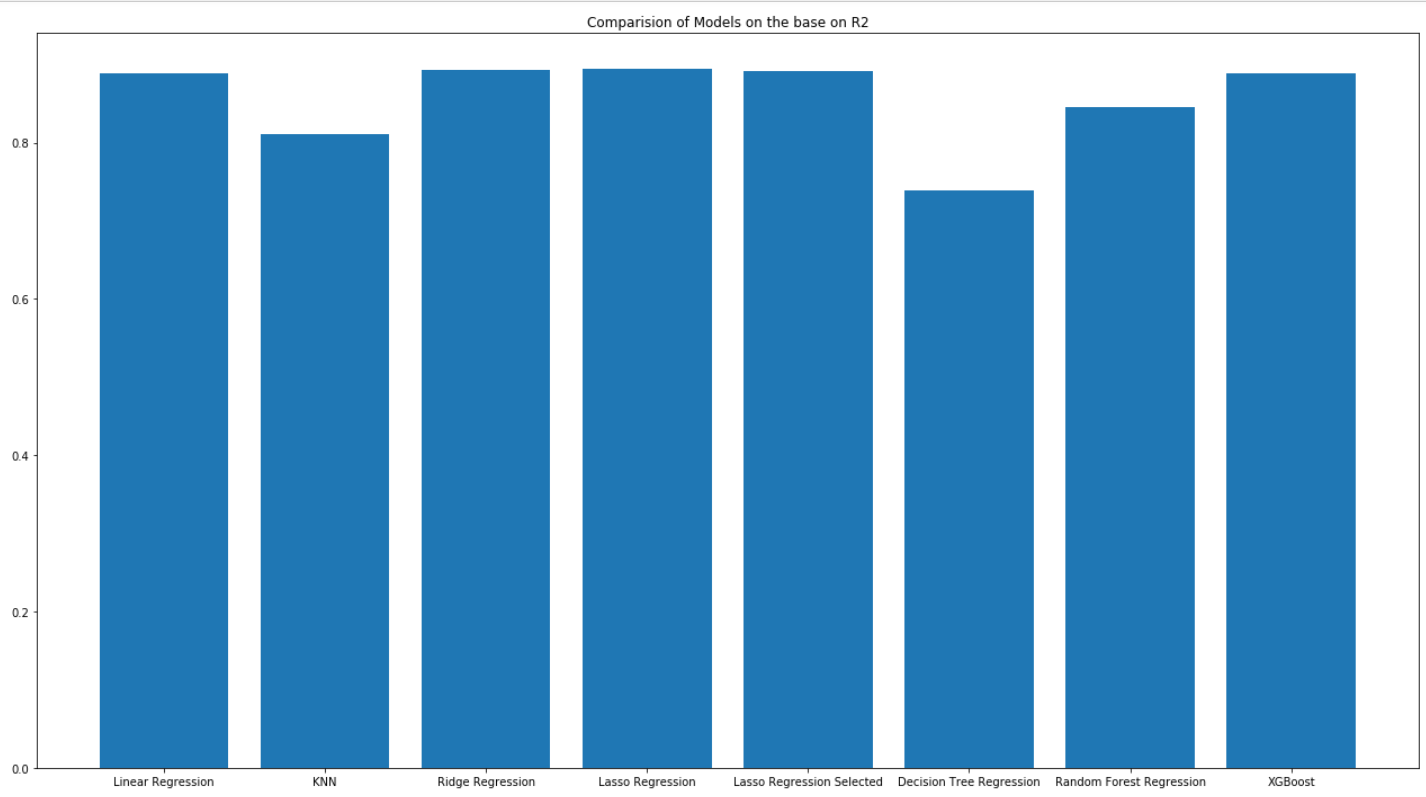
Since our main objective of this project is to build a model with highest predictive capability within our capacity, we did not focus on any kind of inferences or insights.

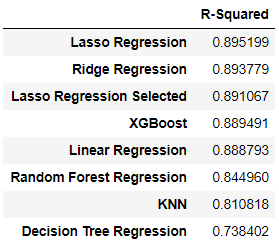
**7-2 Model Evaluation:**

We compared all the models on basis of RMSE & R-Squared scores. We can visually see the difference in following bar plots.

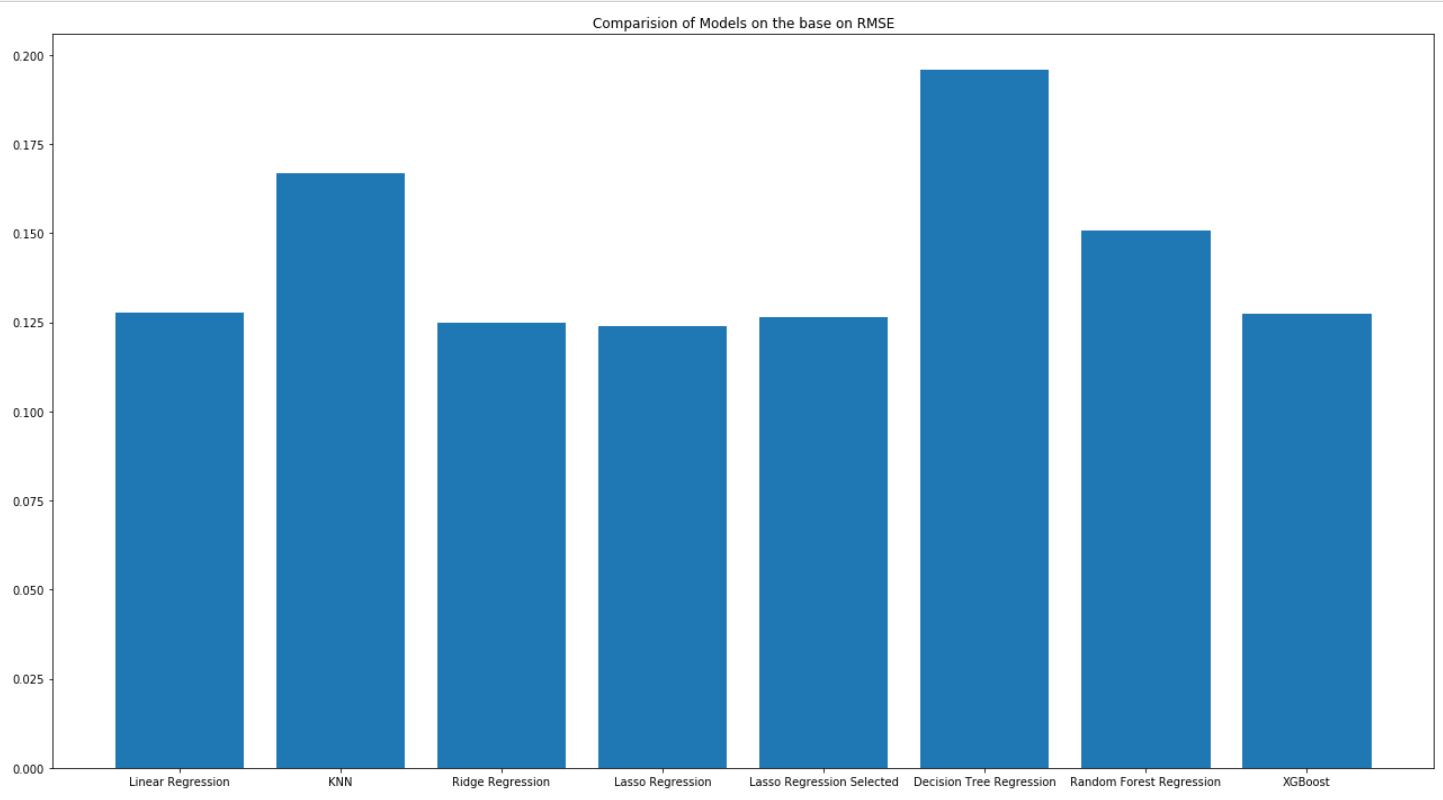
So, based on the evaluation metric of “Root Mean Squared Error”, the best model seems “Lasso Regression”. Which has the lowest RMSE on test split data – **0.124**

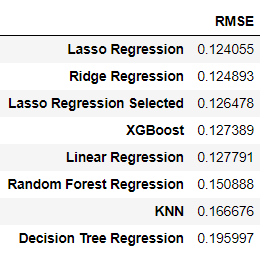
Thus, we predicted the house prices of actual **“test”** data using the **Lasso Regression**.





**R-Squared Scores across the models**





**RMSE across the models**