Capstone Project 2

Housing Price Prediction

K MAHESH **|** Mentor – Sumit Dutta

Housing Price Prediction

1. Introduction:

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. This project is to predict the housing prices in the given area considering various elements like whether the house contains car garage, swimming pool and how many bedrooms it contains and what is the dimensions of the building etc. The goal of this project is to create a regression model that are able to accurately estimate the price of the house given the features.

This project does not have a definitive client. But the analysis performed could be of use to anyone in the Real Estate Business (House Owners, Buyers, Tenets, etc.).

If we can make use of the data to find some insights from the past data and could possibly predict the future, many House Owners and Tenets could use this recommendation system on their own. Giving good recommendations directly entails one or many of the following:

1. Customers use the platform more frequently due to the quality and relevance of content shown to them.

2. Better User Experience. Customers do not relay on brokers and will search on their own according to their need and deed.

The data used in this project has been obtained from Kaggle and it is available in csv (Coma Separated File) format, the data set consists of 1460 instances of training data and 1460 of test data. Total number of attributes equals 81, of which 36 is quantitative, 43 categorical + Id and Sale Price.

Dataset to downloaded from the below link

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

2. Problem statement:

This project is to predict the housing prices in the given area considering various elements like whether the house contains car garage, swimming pool and how many bedrooms it contains and what is the dimensions of the building etc. The goal of this project is to create a regression model that are able to accurately estimate the price of the house given the features.

3. Data Wrangling:

This section describes the various data cleaning and data wrangling methods applied on the **House Price Prediction** to make it more suitable for further analysis. The following sections are divided based on the procedures followed.

Cleaning:

We have 18 categorical features with missing values. And 11 numerical features with missing values.

So, 18 categorical features and 10 numerical features to clean.

- We start with the numerical features, first thing to do is have a look at them to learn more about their distribution and decide how to clean them:
- Most of the features are going to be filled with mean values, because we assume that they don't exist, for example GarageArea, GarageCars with missing values are simply because the house lacks a garage.
- GarageYrBlt: Year garage was built can't be filled with 0s, so we fill with the median (1980).

And we have 18 Categorical features with missing values:

- Some features have just 1 or 2 missing values, so we will just use the forward fill method because they are obviously values that can't be filled with 'Missing'
- Features with many missing values are mostly basement and garage related (same as in numerical features) so as we did with numerical features (filling them with 0s), we will fill the categorical missing values with "Missing" assuming that the houses lack basements and garages.

Removing Unnecessary Features

This process was done in different ways

Number of missing values:

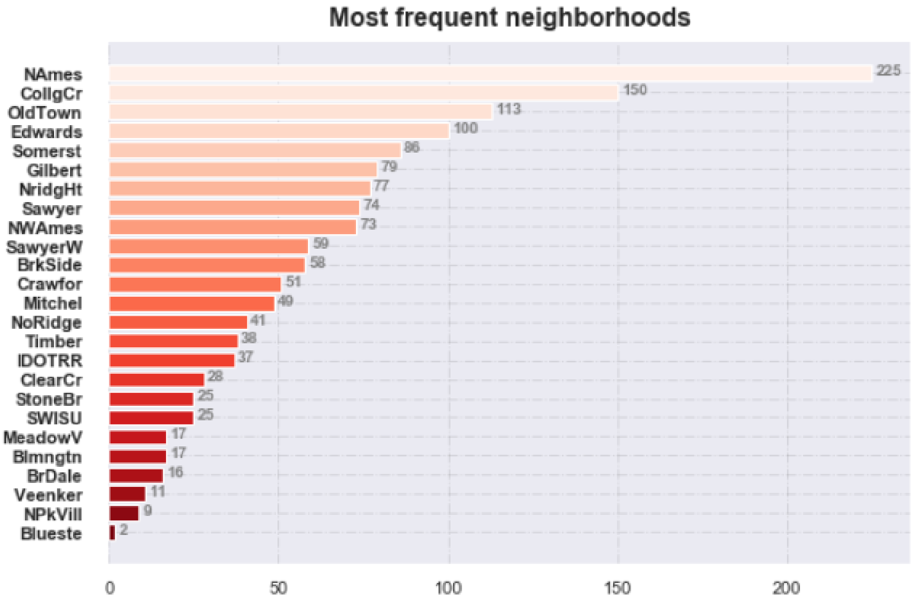
First thing to do is get rid of the features with more than 80% missing values. For example the PoolQC's missing values are probably due to the lack of pools in some buildings, which is very logical. But replacing those (more than 80%) missing values with "no pool" will leave us with a feature with low variance, and low variance features are uninformative for machine learning models. So we drop the features with more than 80% missing values.

Significance:

From co-relation we get to know features that are important for predicting the output and after performing co-relation features which has values greater than 0.7 are carried forward and remaining were dropped.

3. Data Story Telling:

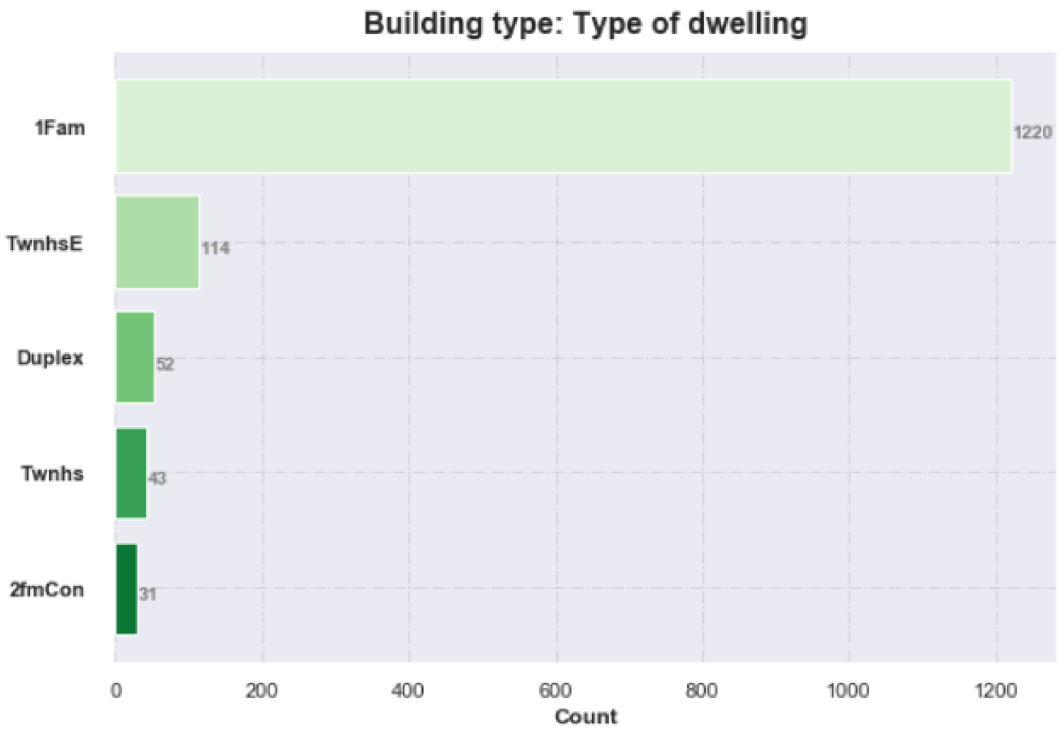
Most Frequent Neighbourhoods:



From the above graph, we noticed that most frequent neighbourhoods are Names with 225, CollgCr with 150, followed by old town, etc. and less frequent neighbourhoods are Blueste.

Type of Dwelling:

In the below bar plot, we noticed that we have 5 different types of dwellings are there. In that Most of the dwelling are of ‘1Fam’ type. And ‘2fmcon’ type of dwellings are very few.



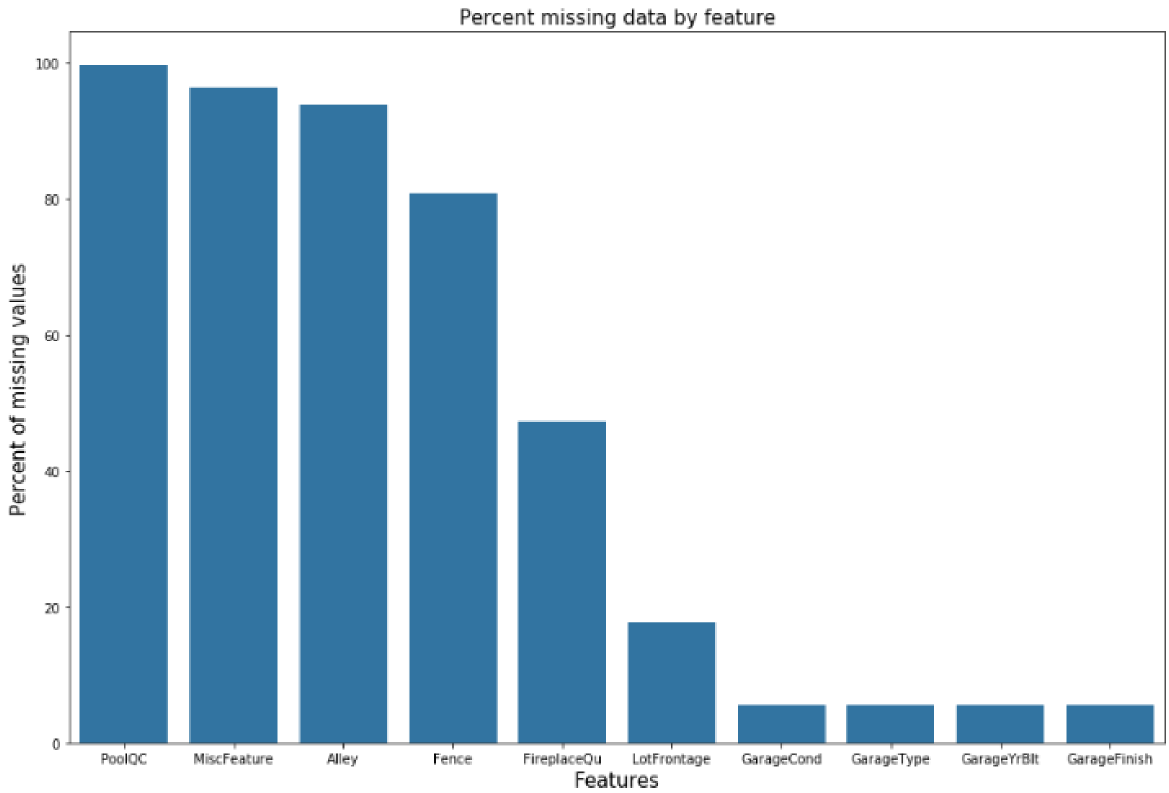
4. What is Exploratory Data Analysis (EDA)?

- How to ensure you are ready to use machine learning algorithms in a project?
- How to choose the most suitable algorithms for your data set?
- How to define the feature variables that can potentially be used for machine learning?

Exploratory Data Analysis (EDA) helps to answer all these questions, ensuring the best outcomes for the project. It is an approach for summarizing, visualizing, and becoming intimately familiar with the important characteristics of a data set

Missing Values:

In the fig below, I'm basically finding out the percentage of missing values in each and every feature. And also we observed that 'PoolQC', 'MiscFeature', 'Alley' and 'Fence' these 4 features are having above 50% of missing values.



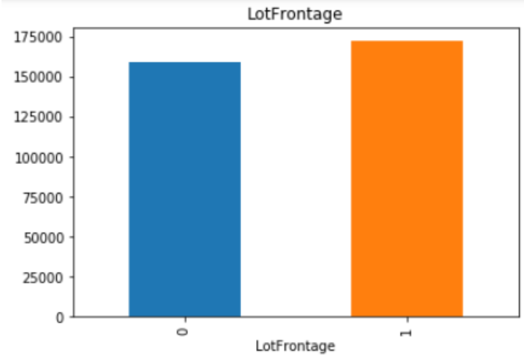
Since they are many missing values, we need to find the relationship between missing values and Sales Price:

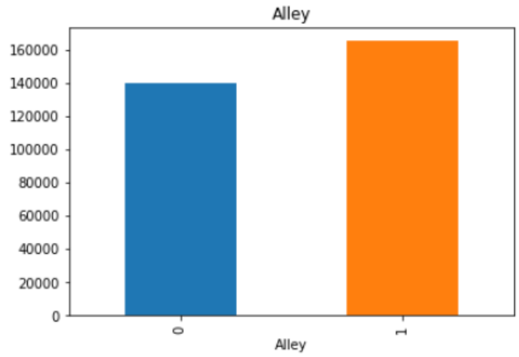
Understanding, whether that missing values has some dependency or is there any relationship with the dependent feature which is the SalesPrice. For that we are plotting. Let's make a variable that indicates 1 if the observation was missing or zero otherwise.

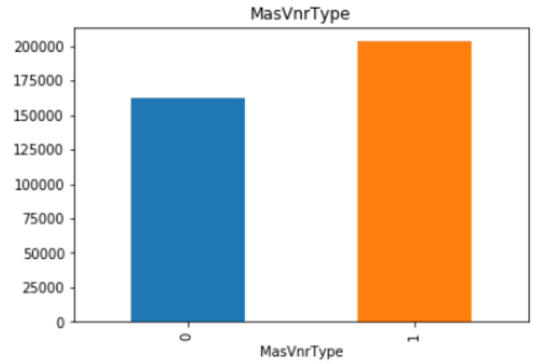
Suppose if a feature had a null value I'm converting it as '1' or else '0'. The reason I'm doing this is to create count plot that will help me to understand that the missing values plays an important role or not.

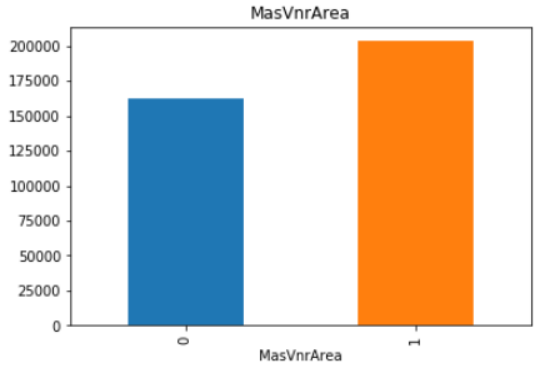
For ex, in the below bar plot LotFrontage feature had lot NaN values. And I convert these Nan values into '1'. Because of this NaN values Salesprice also increases. That means LotFrontage feature plays an important role. Since there are many missing values, we need to find the relationship between missing values and Sales Price.

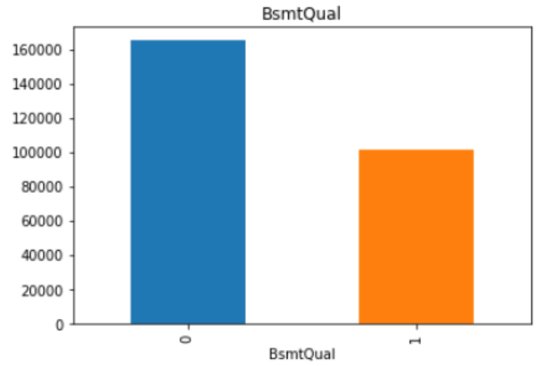
Let's plot some diagram for this relationship

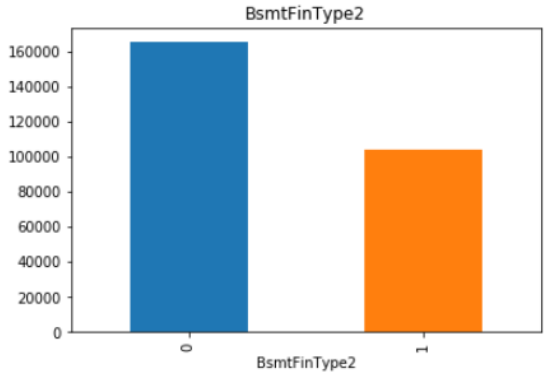

LotFrontage
175000
150000
125000
100000
75000
50000
25000
0
0
1
LotFrontage

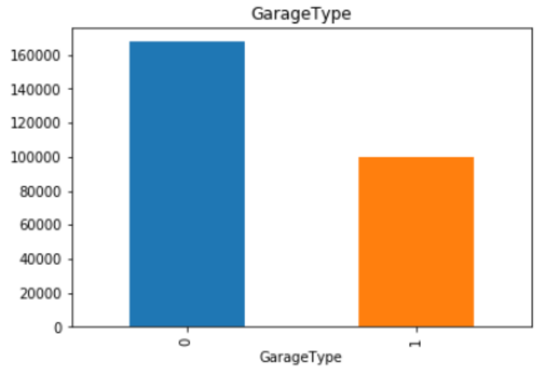

Alley
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
Alley

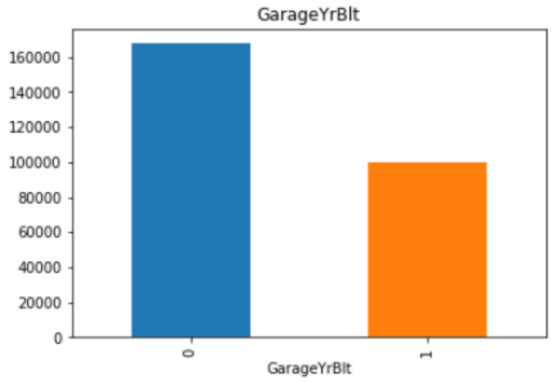

MasVnrType
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrType

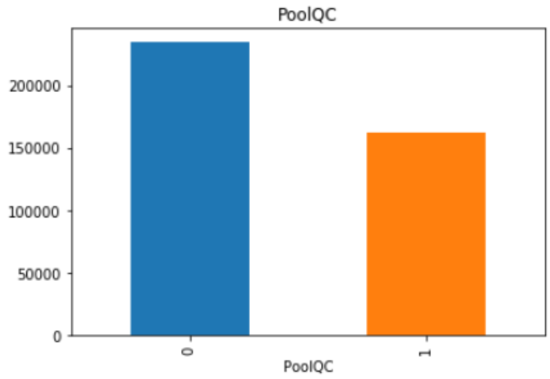

MasVnrArea
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
MasVnrArea

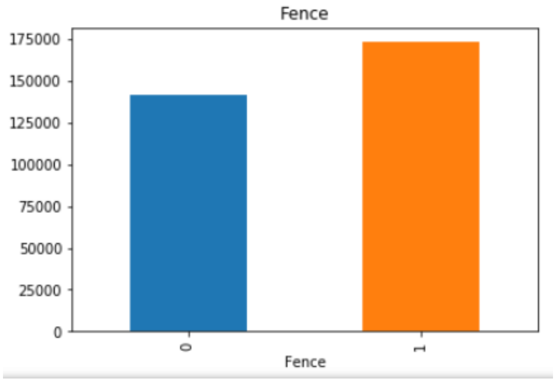

BsmtQual
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtQual


BsmtFinType2
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
BsmtFinType2


GarageType
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageType

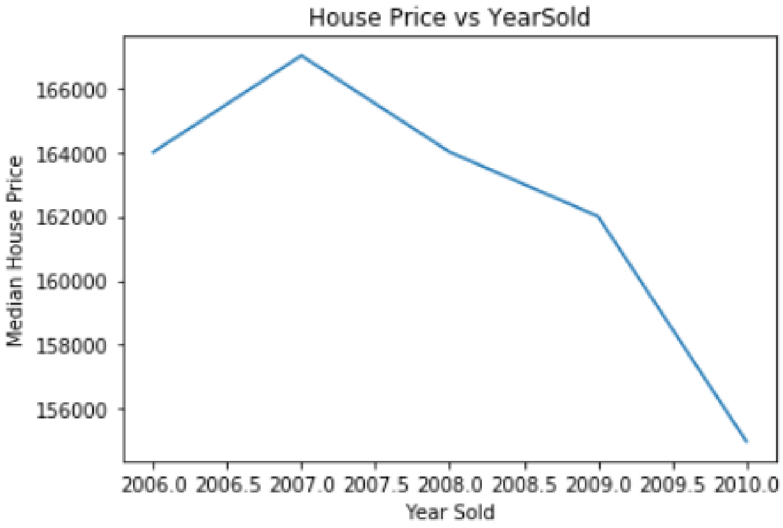

GarageYrBlt
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
GarageYrBlt



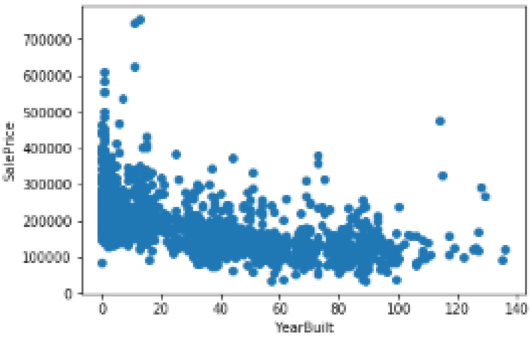


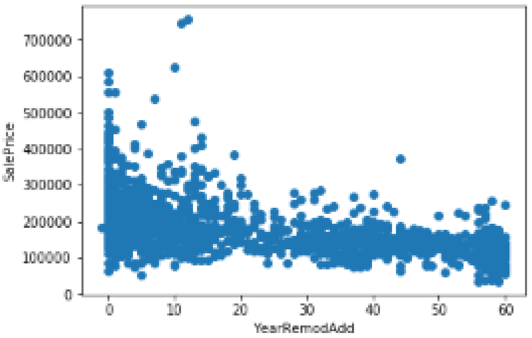
Let’s analyze the Temporal Date-time Variables:

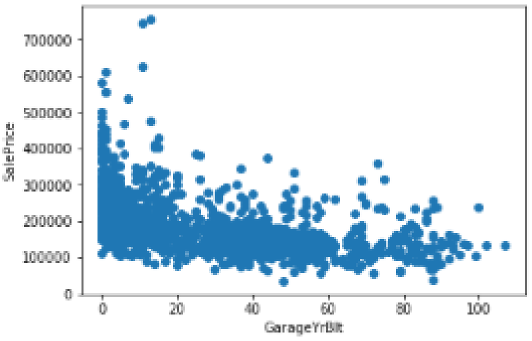
We will check whether there is a relation between year the house is sold and the sales price. As we see in the below fig, as the year sold is going on the price is decreasing this cannot be just true. So we’ll try to find out some more information from this.



Here we will compare the difference between all years feature with SalePrice. We will capture the difference between year variable and year the house was sold for. That difference I’m trying to shown in the below plot. It tell us that suppose if the house was 140 years old then the price of the house was decreasing. If the house was newly built in that case the price of the house was too high. Likewise if you see the GarageYrBlt vs SalePrice, if the GarageYrBlt year is too old then the SalePrice decreases or else the SalePrice increases.







Numerical variables are usually of 2 types:

1. Discrete Variables

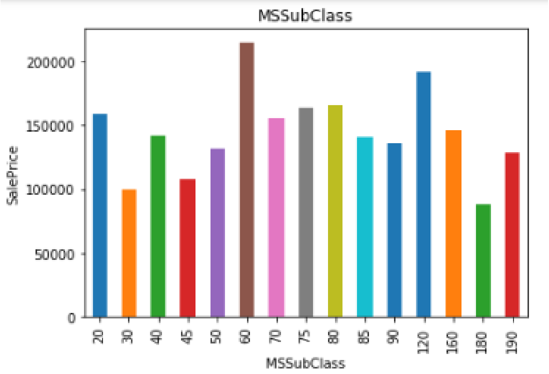
2. Continuous Variables

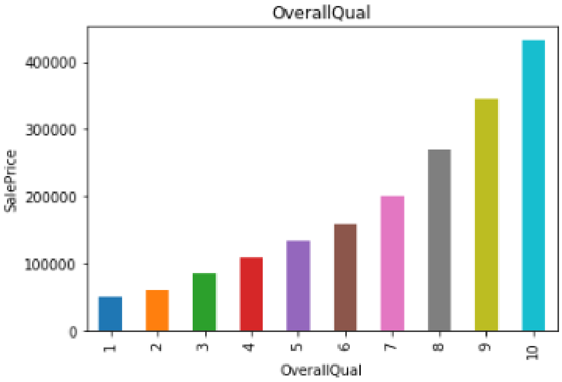
There are 38 Numerical variables, in that 17 features are Discrete Variables, 16 features are Continuous Variables, 4 temporal date-time Variables and Id variable.

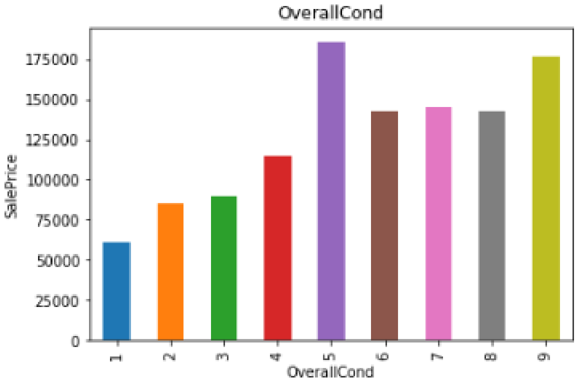
1. Discrete Variables:

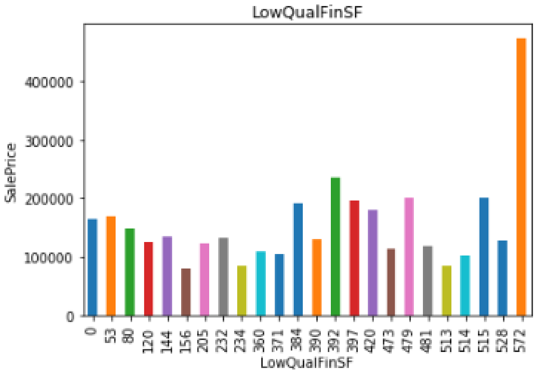
Let’s find the relationship between Discrete Variables and SalePrice.

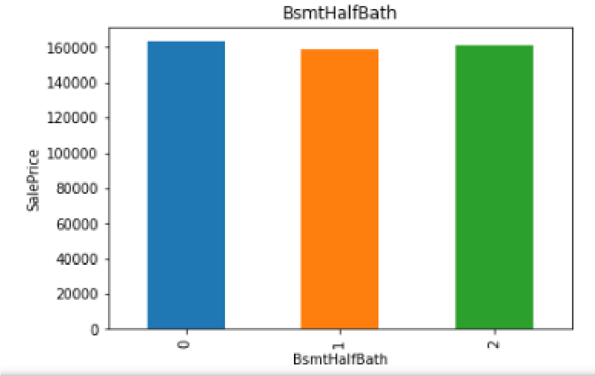
Suppose if we take the OverallQual vs SalePrice, As overall quality increases then saleprice is also increases. It tells us based on the overall quality our saleprice also increases. Likewise we’ll see the other Discrete Variables (vs) SalePrice.

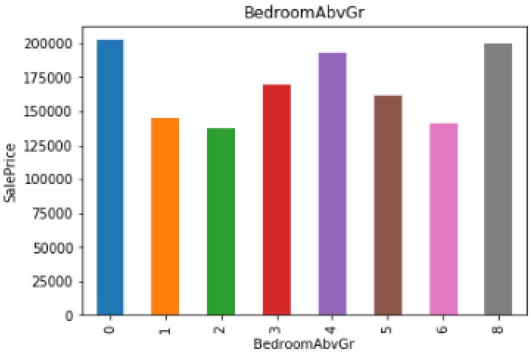

MSSubClass
SalePrice
0
50000
100000
150000
200000
20
30
40
45
50
60
70
75
80
85
90
120
160
180
190
MSSubClass

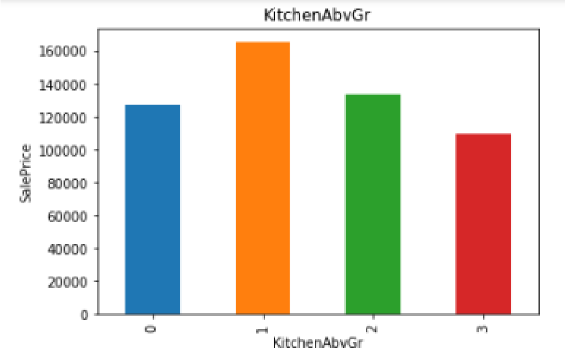

OverallQual
SalePrice
0
100000
200000
300000
400000
1
2
3
4
5
6
7
8
9
10
OverallQual

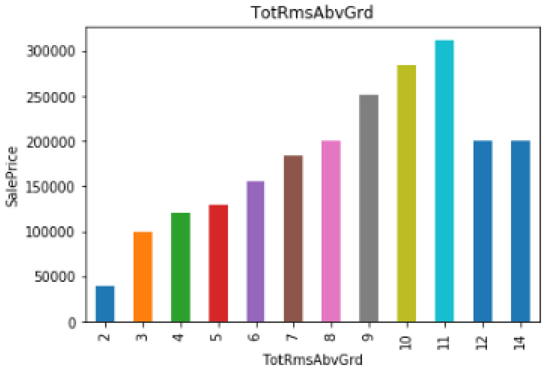

OverallCond
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
1
2
3
4
5
6
7
8
9
OverallCond

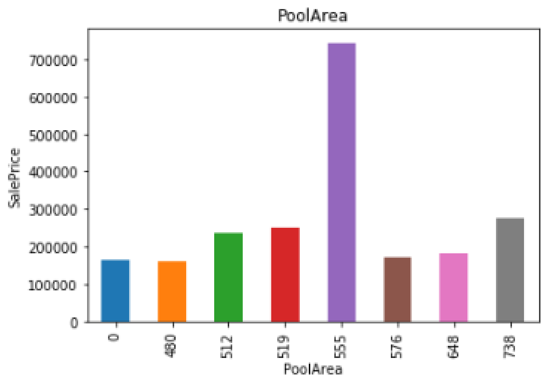

LowQualFinSF
SalePrice
0
100000
200000
300000
400000
0
53
80
120
144
156
205
232
234
360
371
384
390
392
397
420
473
479
481
513
514
515
528
572
LowQualFinSF

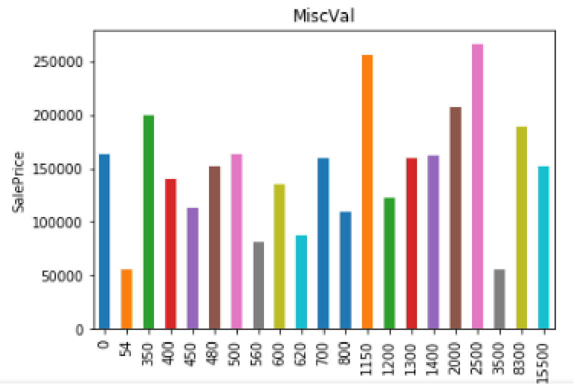

BsmtHalfBath
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
0
1
2
BsmtHalfBath


BedroomAbvGr
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
0
1
2
3
4
5
6
8
BedroomAbvGr


KitchenAbvGr
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
0
1
2
3
KitchenAbvGr


TotRmsAbvGrd
SalePrice
0
50000
100000
150000
200000
250000
300000
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd



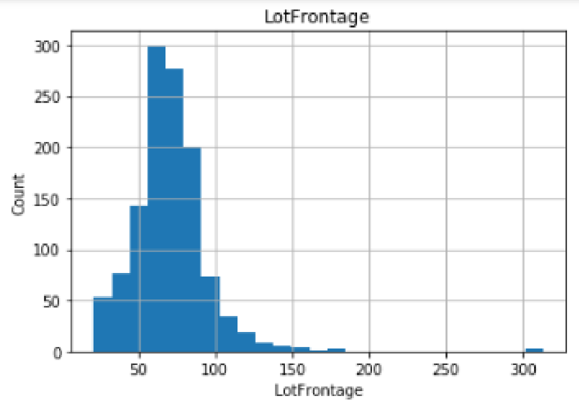


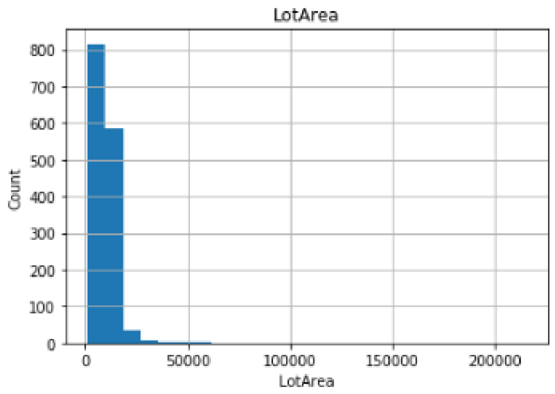
2. Continuous Variable:

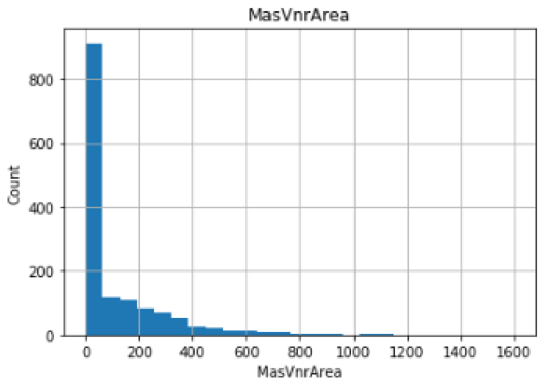
Let’s analyse the continuous values by creating histograms to understand the distribution.

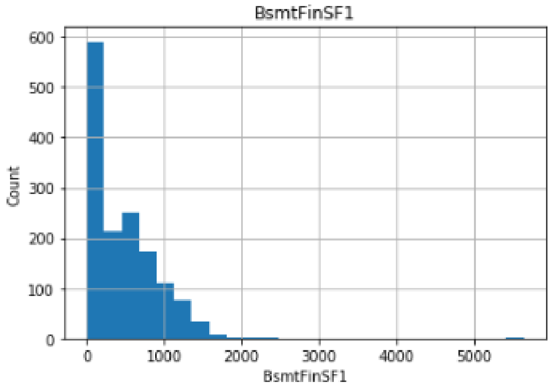
Here in this case we need to find out the distribution of continuous values. For that we are creating the histogram. Here some of the features doesn’t having Gaussians Distribution, these are skewed data. So we need to convert those skewed data into Gaussian’s Distribution using log normalization, because that will be helpful for linear model prediction, that is pretty much important.

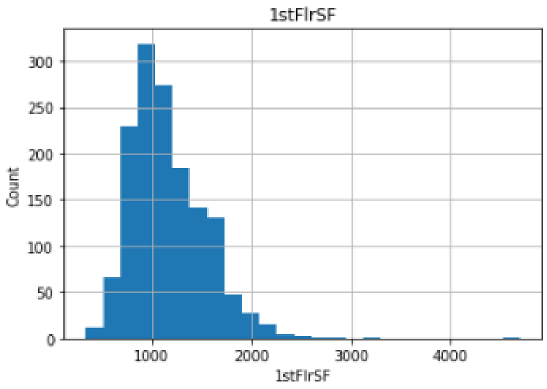
For example in the below, 1stFlrSf feature is not a Gaussian’s Distribution. So we need to convert that feature into Gaussian distribution.

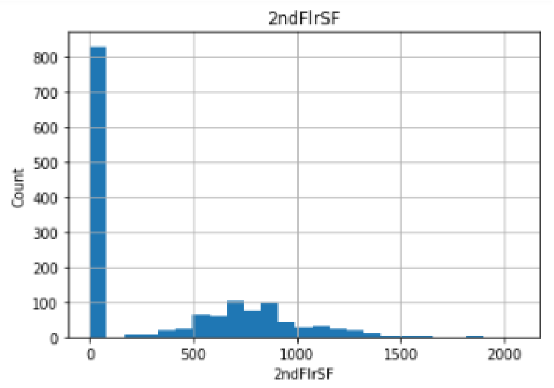


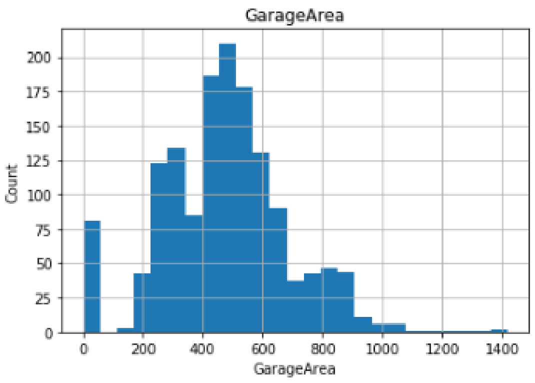


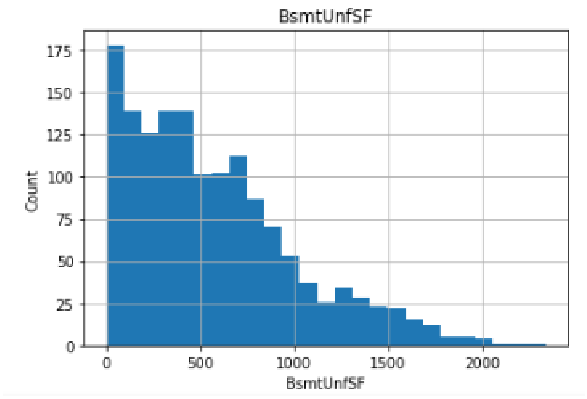






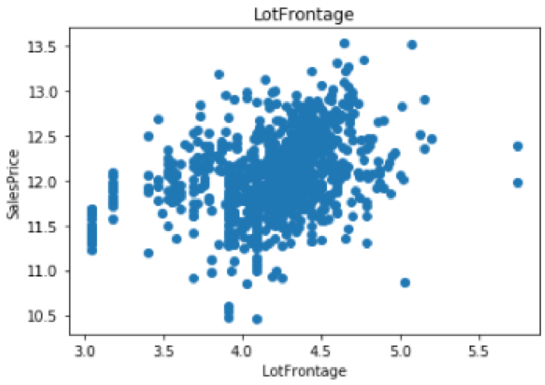


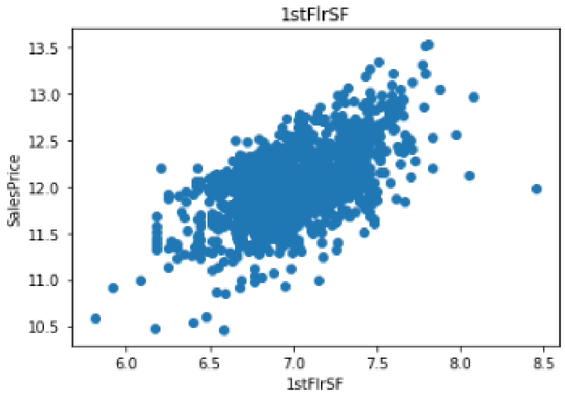


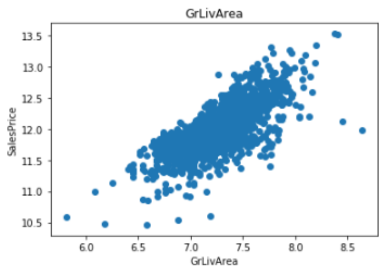


Using logarithmic transformation:

From the above continuous variables we saw that some of the features are not Gaussians Distribution. So it is very important to convert that features into Gaussian's Distribution that's why we are using this logarithmic transformation.







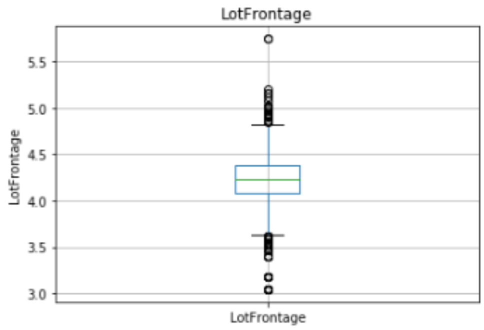
Outliers:

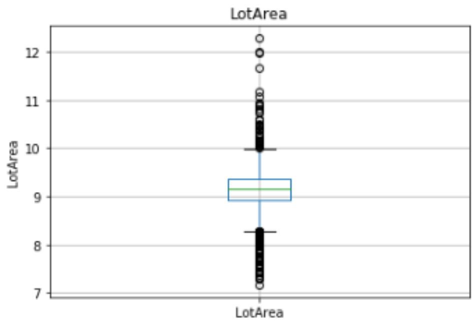
An outlier is a data point in a data set that is distant from all other observations. A data point that lies outside the overall distribution of the dataset.

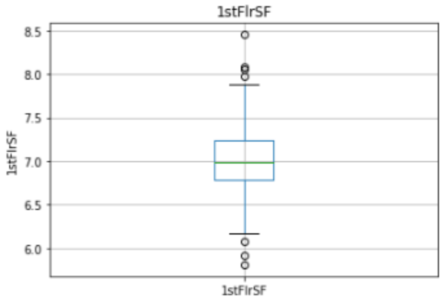
There are a huge number of ways optimised to detect outliers in different situations. These are mostly targeted to identify outliers when those are the observations that we indeed want to focus on, for example for fraudulent credit card activity. Using the following methods we can detect an outlier

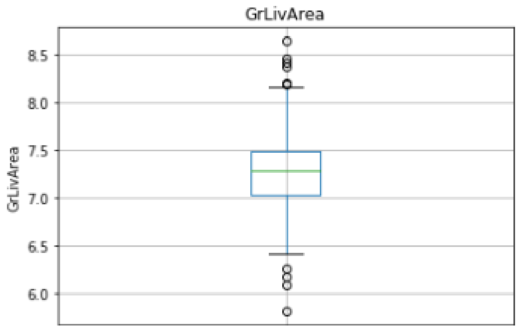
- IQR interquartile range
- z score
- Scatter plots
- Box plot

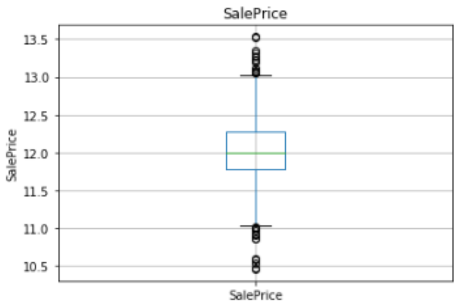
We're using Boxplot in this case, to find each and every continuous variable we can actually find out the outliers.



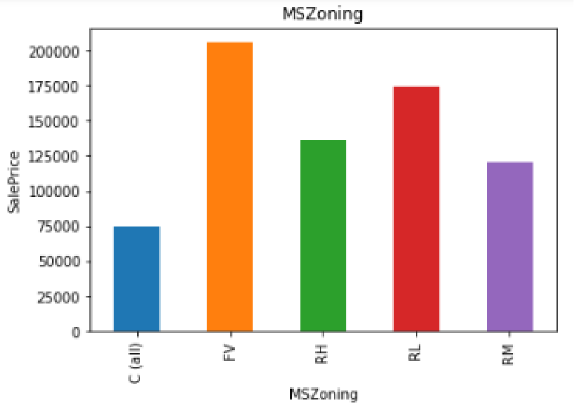


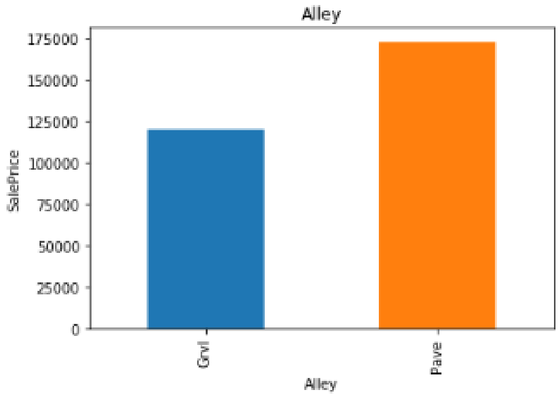


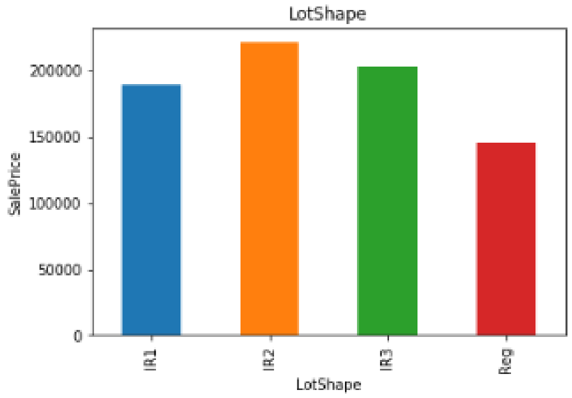


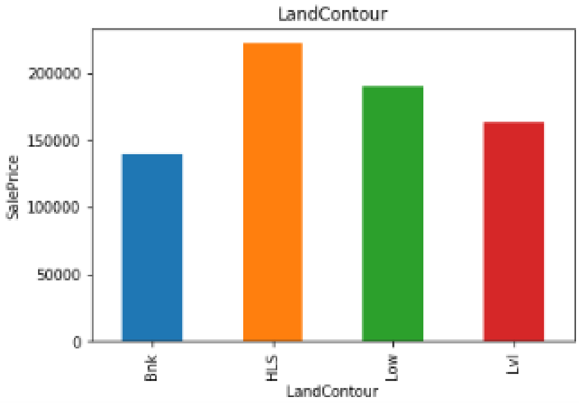


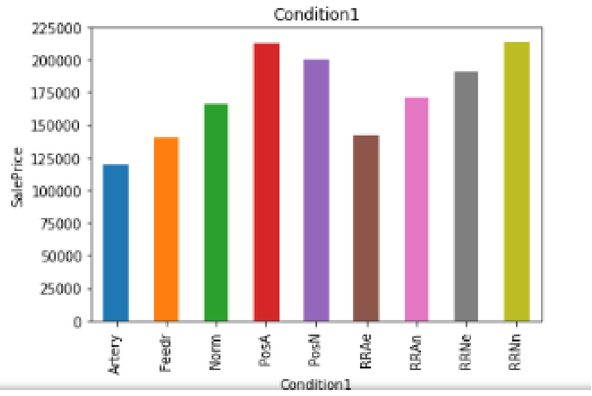
Find out the relationship between categorical variable and dependent feature SalesPrice

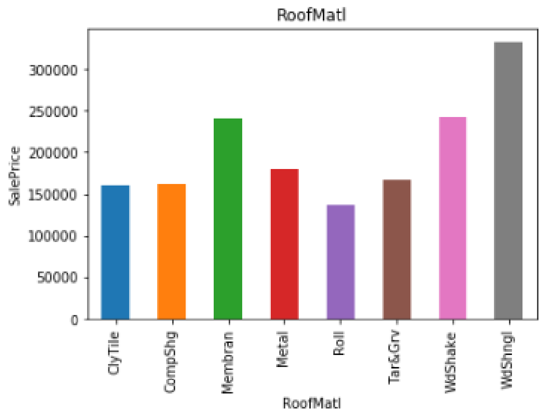




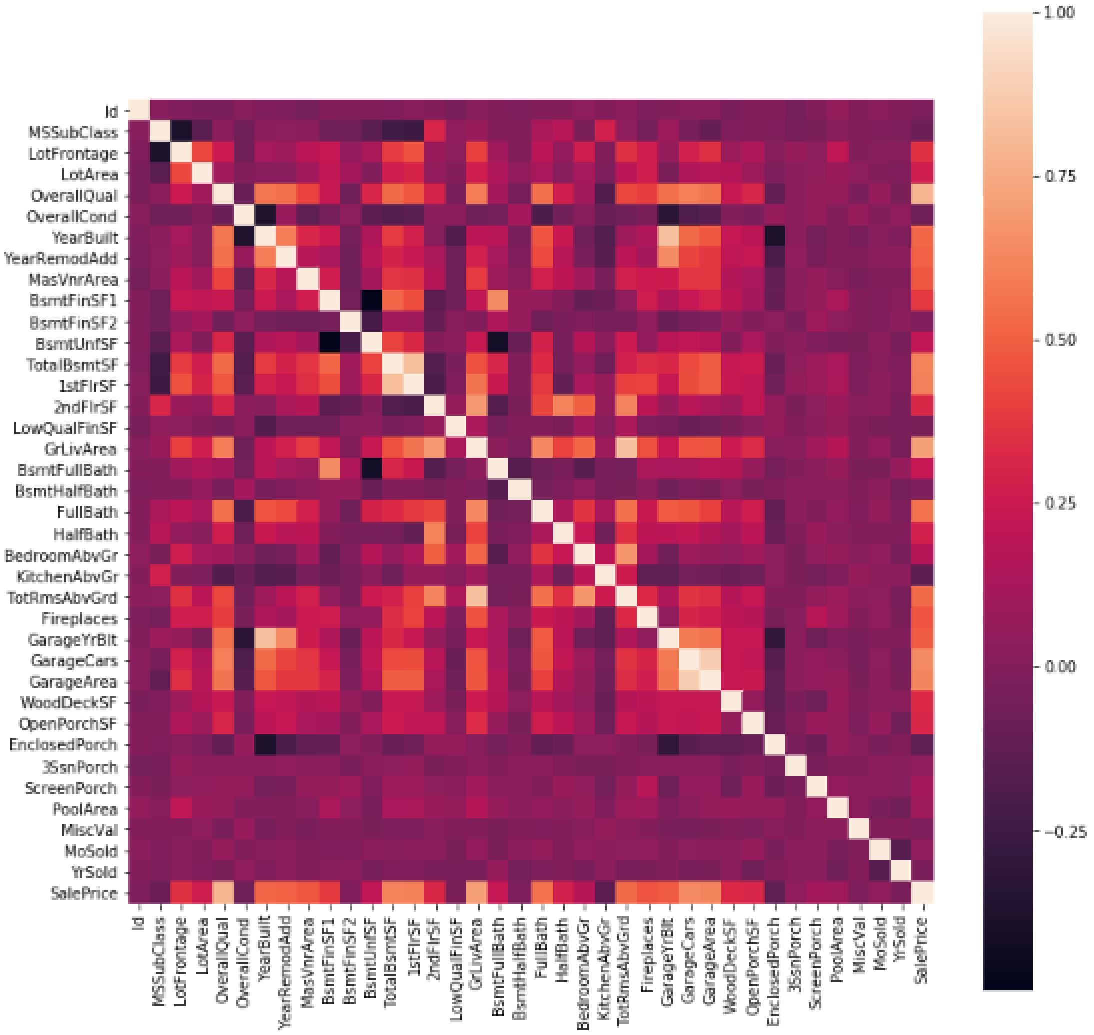








Finding Correlation between different features:



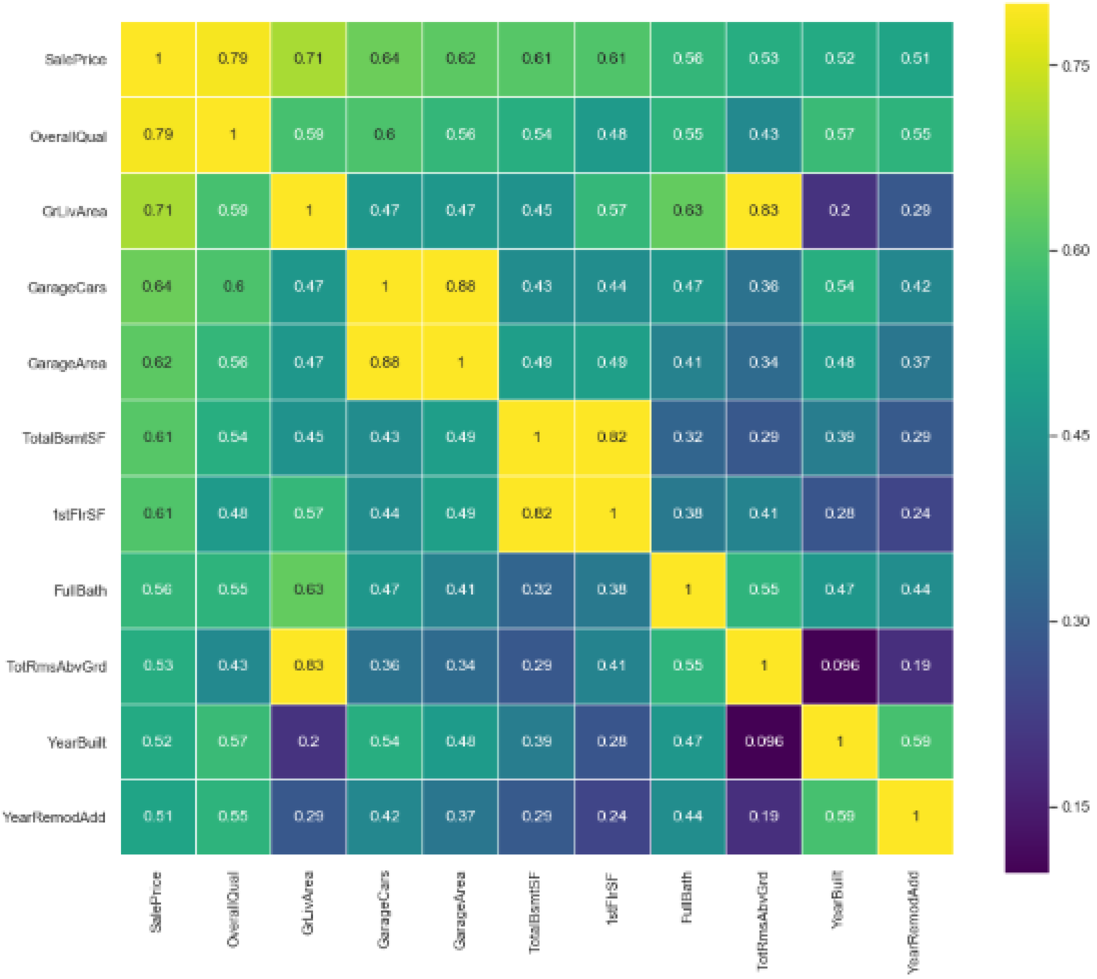
The heat map is the best way to get a quick overview of correlated features thanks to seaborn!

At initial glance it is observed that there are two red coloured squares that get my attention.

The first one refers to the 'TotalBsmtSF' and '1stFlrSF' variables. Second one refers to the 'GarageX' variables. Both cases show how significant the correlation is between these variables. Actually, 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. Heat maps 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 I observed here is the 'SalePrice' correlations.As it is observed that 'GrLivArea', 'TotalBsmtSF', and 'OverallQual' saying a big 'Hello !' 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

SalePrice Correlation matrix:



The above heat map shows that

1 OverallQual', 'GrLivArea' and 'TotalBsmtSF' are strongly correlated with 'SalePrice'.

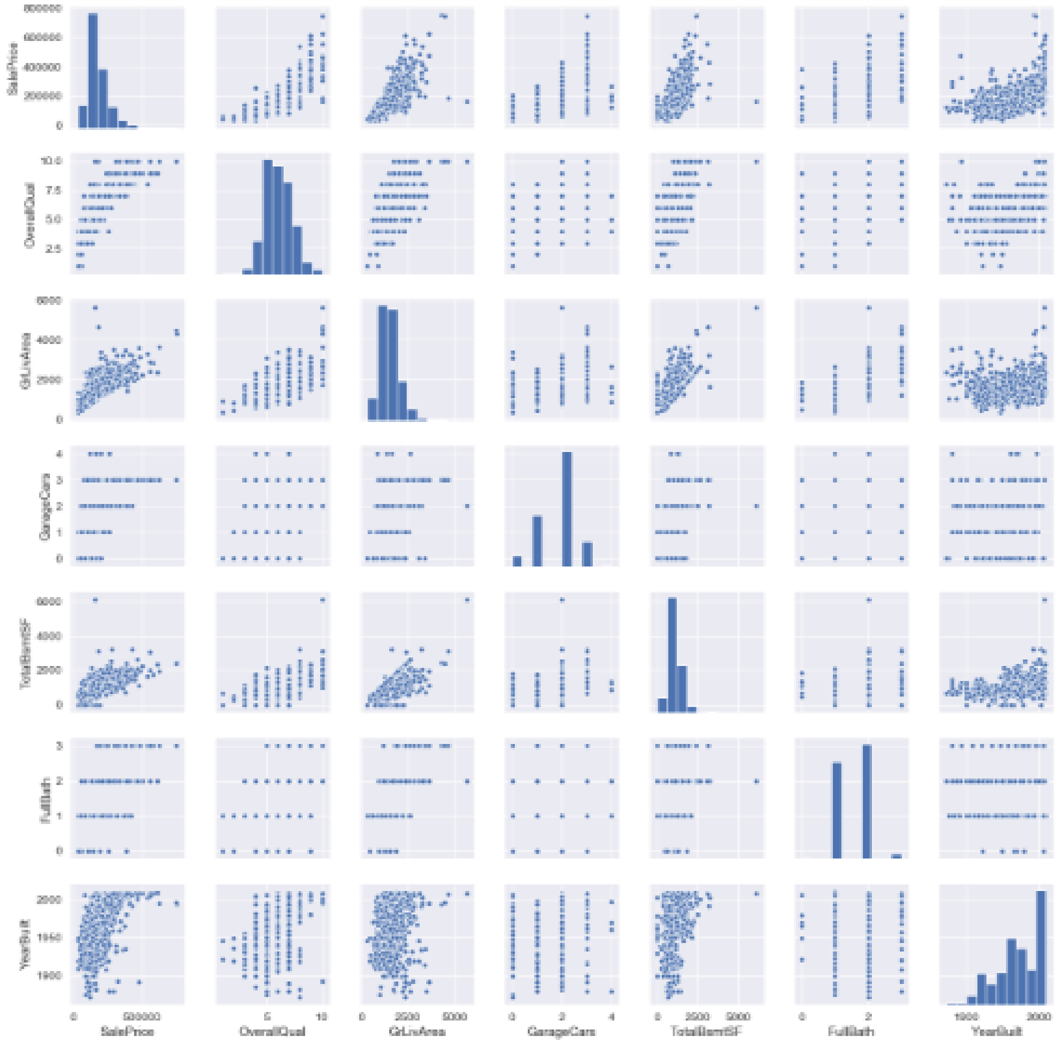
2 'GarageCars' and 'GarageArea' are strongly correlated variables.

3 'TotalBsmtSF' and '1stFloor' seem to be correlated with each other.

4 'TotRmsAbvGrd' and 'GrLivArea' also seem to correlated with each other.

Pair Plot:

Although we already know some of the main figures, this pair plot gives us a reasonable overview insight about the correlated features.



1. One interesting observation is between 'TotalBsmtSF' and 'GrLiveArea'. In this figure we can see the dots drawing a linear line, which almost acts like a border. It totally makes sense that the majority of the dots stay below that line. Basement areas can be equal to the above ground living area, but it is not expected a basement area bigger than the above ground living area.
2. One more interesting observation is between 'SalePrice' and 'YearBuilt'. In the bottom of the 'dots cloud', we see what almost appears to be an exponential function. We can also see this same tendency in the upper limit of the 'dots cloud'
3. Last observation is that prices are increasing faster now with respect to previous years.

5. Feature Engineering:

Feature Engineering is a process of extracting useful features from existing raw data using maths, statistics and domain knowledge.

Feature Engineering is one of the most important steps to complete before starting a Machine Learning analysis. Most of the basic Feature Engineering techniques consist of finding inconsistencies in the data and of creating new features by combining/diving existing ones. Creating the best possible Machine Learning/Deep Learning model can certainly help to achieve good results, but choosing the right features in the right format to feed in a model can by far boost performances leading to the following benefits:

- Enable us to achieve good model performances using simpler Machine Learning models.
- Using simpler Machine Learning models, increases the transparency of our model, therefore making easier for us to understand how is making its predictions.
- Reduced need to use Ensemble Learning techniques.
- Reduced need to perform Hyperparameters Optimization.

Real-world data often has missing values. Firstly we cannot simply ignore missing values in a dataset. We must handle them in some way for the very practical reason that most algorithms do not accept missing values.

In this project, we have two different features categorical and numerical features. We filled numerical features with median values, because there are lot of outliers in these features. And for categorical features I replaced NaN’s with “Missing” value using fillna() method.

.6. Feature Scaling:

Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one.
The most common techniques of feature scaling are Normalization and Standardization.

Normalisation:
One method utilised to bring all the variables to a more homogeneous scale is normalisation. Normalisation is synonym of centering the distribution. This means subtracting the mean of

the variable to each observation. This procedure will "center" the new distribution at zero (the new mean of the variable will now be zero).

Standardisation:

Standardisation is also used to bring all the variables to a similar scale. Standardisation means centering the variable at zero, and standardising the variance at 1. The procedure involves subtracting the mean of each observation and then dividing by the standard deviation:

z = (x - x_mean) / std

Standardisation:

StandardScaler from scikit-learn removes the mean and scales the data to unit variance.

$$x_{new} = \frac{x - \mu}{\sigma}$$

The Standard Scaler assumes data is normally distributed within each feature and scales them such that the distribution centered around 0, with a standard deviation of 1. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. If data is not normally distributed, this is not the best Scaler to use.

Min-Max Scaling

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Transform features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g., between zero and one. This Scaler shrinks the data within the range of -1 to 1 if there are negative values. We can set the range like [0,1] or [0,5] or [-1,1]. This Scaler responds well if the standard deviation is small and when a distribution is not Gaussian. This Scaler is sensitive to outliers.

In this problem we are using standardisation technique using Min-Max Scaling method is used for solving the problem. Apply Feature selection model to remove the features which are close to zero, firstly I specify the Lasso Regression model, and I select a suitable alpha (equivalent of penalty). The bigger the alpha the less features that will be selected. Then I use the selectFromModel object from sklearn, which will select the features which coefficients are non-zero.

7. In-depth analysis using machine learning:

Here to solve this problem I have used different regression models.

In this problem, 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.)

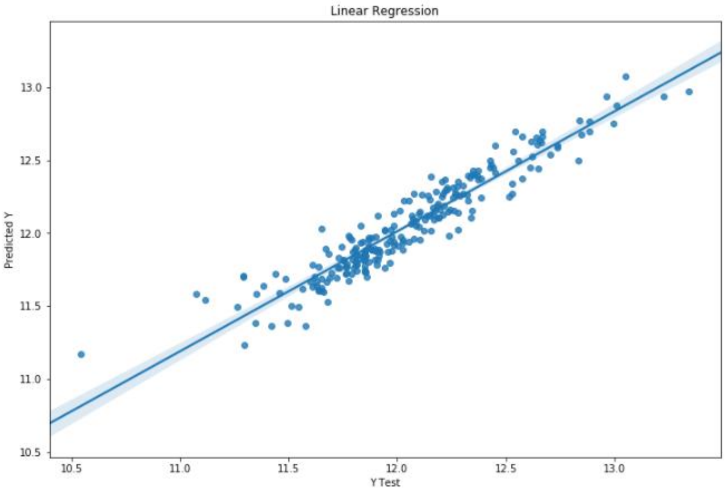
$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(P_i - O_i\right)^2}{n}}$$

Modelling, we'll start by building standalone models, validating their performance and picking the right ones. Later we will stack all our models into an ensemble for better accuracy. I just played with a number of models and ended up picking the following models which gave me best results personally. I tuned the hyper parameters by manually experimenting a lot based on previous experiences, saving you a bunch of time hopefully.

1. Linear Regression:

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an independent variable, and the other is considered to be a dependent variable.

A linear regression line has an equation of the form ***Y = a + bX***, where ***X*** is the explanatory variable and ***Y*** is the dependent variable. The slope of the line is ***b***, and ***a*** is the intercept

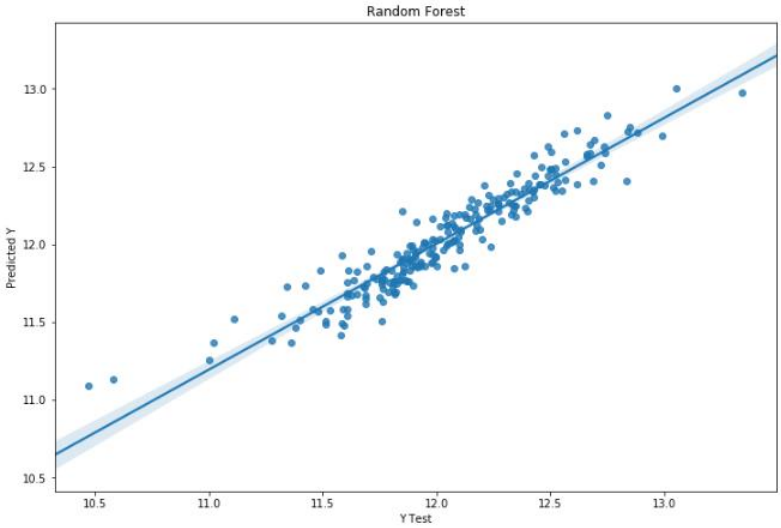


By using Linear Regression, we got accuracy of 90.04 and the RMSE value of 0.01065125

2. Random Forest:

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.

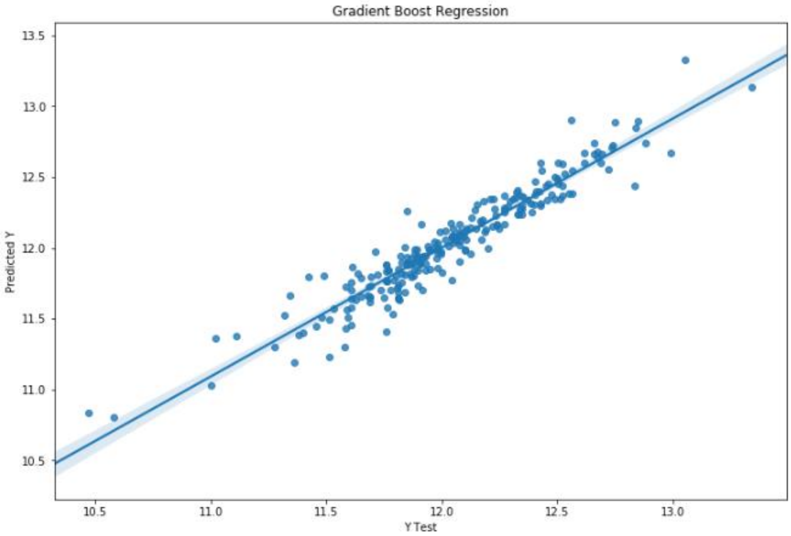
Random forest is collection of trees (forest) and it builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression problems. Example: Suppose we have a bowl of 100 unique numbers from 0 to 99. We want to select a random sample of numbers from the bowl. If we put the number back in the bowl, it may be selected more than once.



By using Random Forest model, I have got accuracy value of 88.84. And the RMSE value of 0.01129651.

3. Gradient Boost Regression:

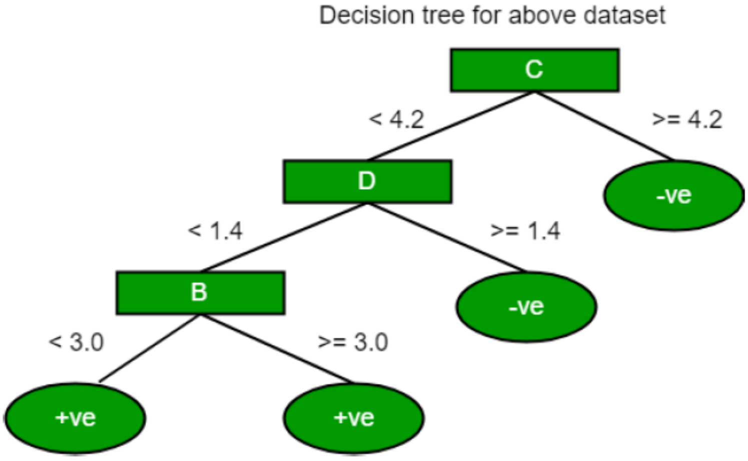
Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between Ada-Boost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (e.g. decision trees). While the Ada-Boost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function (y=ax+b+e, ‘e’ needs a special mention as it is the error term). The loss function is a measure indicating how good model’s coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimise. We are trying to predict the sales prices by using a regression, then the loss function would be based off the error between true and predicted house prices.

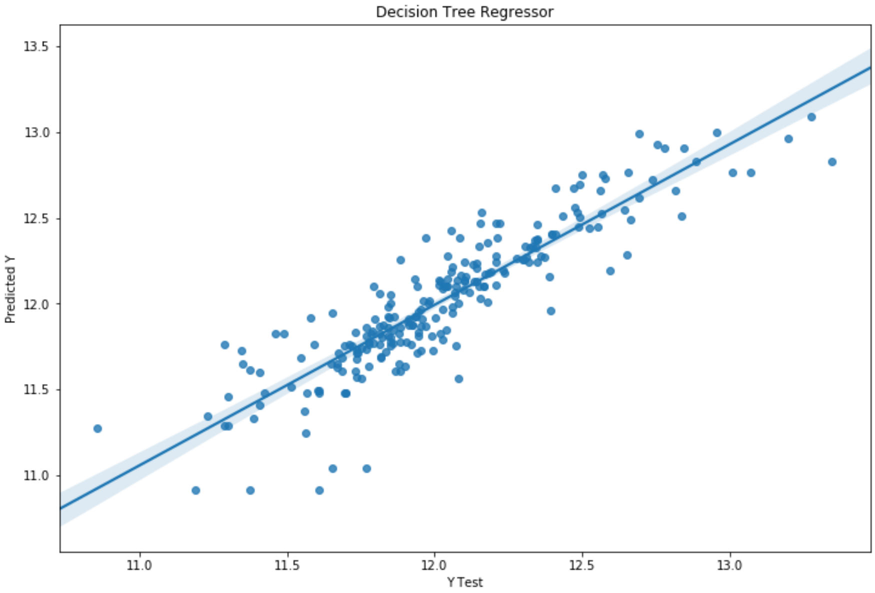


By using Gradient Boost model, I have got accuracy value of 91.1. And the RMSE value of 0.00980553.

4. Decision Tree Regression:

The decision tree is a simple machine learning model for getting started with regression tasks. Background a decision tree is a flow-chart-like structure, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf (or terminal) node holds a class label. The topmost node in a tree is the root node.





By using Decision Tree Regression model, I've got accuracy value of 77.89. And the RMSE value of 0.01557656.

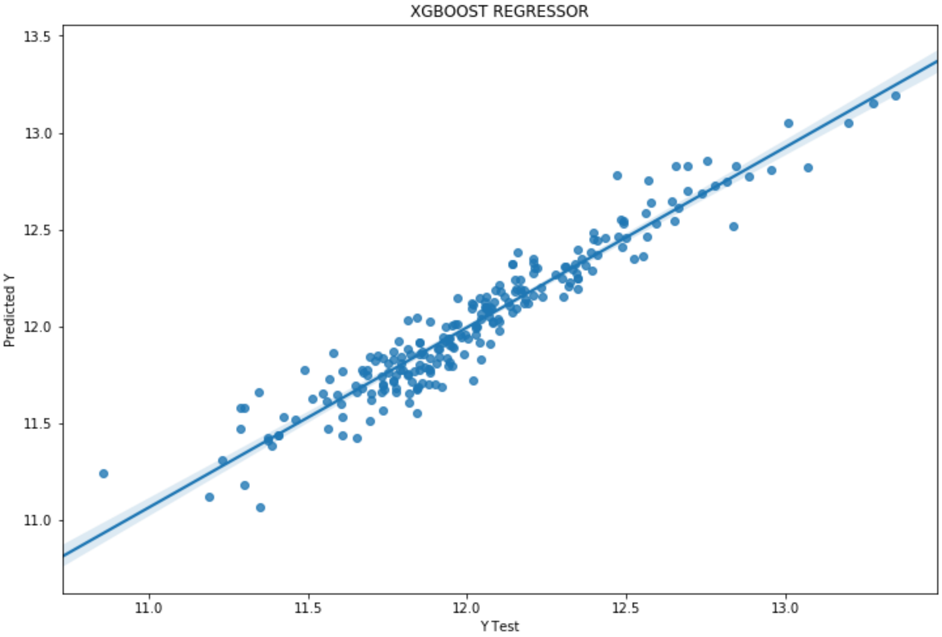
5. XG Boost Regression:

XGBoost is one of the most popular machine learning algorithm these days. Regardless of the type of prediction task at hand; regression or classification. XGBoost is well known to provide better solutions than other machine learning algorithms. In fact, since its inception, it has become the "state-of-the-art" machine learning algorithm to deal with structured data.

The basic idea behind boosting algorithms is building a weak model, making conclusions about the various feature importance and parameters, and then using those conclusions to build a new, stronger model and capitalize on the misclassification error of the previous model and try to reduce it. Now, let's come to XGBoost. To begin with, you should know about the default base learners of XGBoost: **tree ensembles**. The tree ensemble model is a set of classification and regression trees (CART).

k-fold Cross Validation using XGBoost:

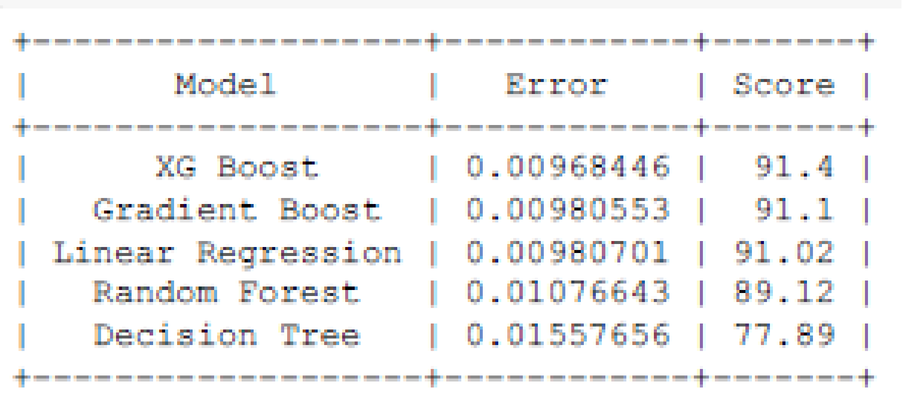
In order to build more robust models, it is common to do a k-fold cross validation where all the entries in the original training dataset are used for both training as well as validation. Also, each entry is used for validation just once. XGBoost supports k-fold cross validation via the cv() method. All you have to do is specify the nfolds parameter, which is the number of cross validation sets you want to build. Also, it supports many other parameters.



By using XGBoost Regression model, I’ve got accuracy value of 91.4. And the RMSE value of 0.00968446.

CONCLUSION:

This report highlighted the processes of data wrangling, inferential statistics, data visualization, feature engineering and predictive modelling performed on the Housing Dataset. All the results and insights gained as part of these processes were also highlighted. With these insights, Linear Regression, Decision tree, Gradient boost regression, XG Boost Regression and Random Forest models were built to predict House prices. Below table will shows the predicted scores along with the RMSE values. And we noticed that XG Boost gives the best value when compared to all other models.



```
+-------------------+------------+-------+
|       Model       |   Error    | Score |
+-------------------+------------+-------+
|     XG Boost      | 0.00968446 |  91.4 |
|  Gradient Boost   | 0.00980553 |  91.1 |
| Linear Regression | 0.00980701 | 91.02 |
|   Random Forest   | 0.01076643 | 89.12 |
|   Decision Tree   | 0.01557656 | 77.89 |
+-------------------+------------+-------+
```