

# **HOUSING: PRICE PREDICTION**

**Submitted by:**  
**MONIKA SINGH**

## 1. Introduction

### Business Problem Framing

Real estate study is very important in case of future investment. Real estates are the necessary need of each and every person and it is one of the major market to contributors in the world's economy. It seems to be profitable because the price of real estate do not fall (low/ loss) rapidly. Also, buying real estate can be both satisfying and lucrative. Real estate owners can buy a property by paying a portion of the total cost upfront, then paying off the balance, plus interest, over time. So, the tendency of buying the real estate property is increasing day by day. Also the change in price of real estate can affect various household investors, bankers and many. The real estate market is a very large market and there are various companies working in the domain.

Now a day's data science has a very important role to solve the problems to help the companies to increase the total revenue and profits. So it will be very useful to making a realistic machine learning model in the domain of real estate to predict the selling price of the property and the future scenario of the price that is how it is increased.

There are many factors which can affect the price of the property like, number of bedrooms, number of bathrooms, garage location, total square feet area of the garage etc. Also the price depends upon location of the property. The house with great accessibility with school, college, local market and other daily life necessary is much costlier than others.

Now to predict the selling price, Regression ML model is used. Regression is a supervised learning algorithm in machine learning which is used for prediction of target variable by training with the pre-defined features with the given data. This model is used when the target variable (here Sale price) is continuous data.

### Conceptual Background of the Domain Problem

Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

The company is looking at prospective properties to buy houses to enter the market. We are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

1. Which variables are important to predict the price of variable?
2. How do these variables describe the price of the house?

It is required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

## Review of Literature

Before making any Machine Learning model, we need to study the previous papers about the given problem and on the basis of study we can predict the drawbacks of the problem and start making the new model with the reference of these previous papers. Here we briefly review the related work on house price prediction and the techniques used.

Venelin Valkov, Apr 2, 2019, in his paper *Predicting House Prices with Linear Regression / Machine Learning from Scratch (Part II)* in *Towards Data Science* proposed a model on Housing price prediction. Here, a Linear Regression and a multivariable linear regression is proposed for housing prices prediction. The author use different graphs to determine the relation between different features and target variable, checking missing data, loss function, MSE, OHMSE. In first model, he use simple linear regression and determine the cost. Then for minimizing the cost vale he use multivariable linear regression and the loss at the last iteration is nearly 2 times smaller than previous one.

Sifei Lu, Zengxiang Li, Zheng Qin, Xulei Yang, Rick Siow Mong Goh had proposed an *Advanced house prediction system using linear regression*. This paper proposed a hybrid Regression ML Model for housing selling prices prediction to find the optimal features and their correlation with Sales Prices. Their system showed that working on the Ames Housing showed that the use of Hybrid algorithms, 65% Lasso and 35% Gradient Boost provided results in predicting the house prices rather than using one from lasso, ridge or gradient boost.

In other paper named *House Price Prediction* by Bindu Sivasankar<sup>1</sup>, Arun P. Ashok, Gouri Madhu, Fousiya S., different models are used to determine best score and RMSE. Then among those models, the determine threshold value of RMSE as 0.12 and integrate those algorithms (Ridge regression, Lasso regression, XGBoost regression) with RMSE value less than 0.12. This definitely increases the accuracy.

A project named *Housing Price prediction using Supervised Learning* by Vidya Bhistannavar, Nikita Chauhan, V Matey, A Shitole 3 different ML models Linear Regression, Decision Tree, Random Forest are made. Different vectors are used MSE, MAE, EMSE to calculate best models.

## Motivation for the Problem Undertaken

The project is provided to me by Flip Robo Technologies as a part of the internship programme (Internship Batch No-31). This problem is a real world dataset. The exposure of this data gives me the opportunity to locate my skills in solving a real time problem. It is the primary motivation to solve this problem.

Another aim of this study is to make a suitable ML model to predict the sale price of the housing property. Now a days, real estate market gives an important role in stock market, future investment, property building and the overall GDP hike. So this dataset is vital with respect to economical background also.

This study aims to analyse and predicting house prices when using Regression model like Linear Regression, Random Forest, Decision Tree, KNN, SVR, Gradient boosting, AdaBoost Regression algorithms. Thus, the purpose of this study is to grow the knowledge of **Regression methods** in machine learning fields. Those are the different factors to undertaken the problem for study purpose.

## 2. Analytical Problem Framing

### Mathematical/ Analytical Modelling of the Problem

The goal of this project is to predict the price of house with the help of regression-based algorithm. In this project different types of algorithms are used. The algorithms are used with their own mathematical equation on background.

This project have two separate data set for training & testing. First different steps of data pre-processing is performed over the training and testing data like data cleaning, data visualization, relationship between features with label. After this, unnecessary feature are removed. In model building Final model is select based on Accuracy score, RMSE value, Hyperparameter tuning and Cross Validation score of different algorithms. Then find the most important feature among all features.

### Data Sources and their formats

Both of the given dataset is in csv form.

Training data is used to train the model. It has 1168 rows and 81 columns. The model will train with the help of this dataset. It has 80 independent features and one dependent or target variable (SalePrice)

In other hand test data has 292 rows and 80 columns. After determine the proper model, the model is applied to predict the target variable for the test data.

```
# checking shapes of train and test data

print ("No of rows of train dataset:",train.shape[0])
print ("No of Columns of train dataset:",train.shape[1])

print ("No of rows of test dataset:",test.shape[0])
print ("No of Columns of test dataset:",test.shape[1])

No of rows of train dataset: 1168
No of Columns of train dataset: 81
No of rows of test dataset: 292
No of Columns of test dataset: 80
```

To determine the data format, info() method is used. There are total 43 categorical columns among 81 columns.

## Data Pre-processing Done:

### Checking Null:

Null values are present in the total dataset.

```
null_val= train.isna().sum().sort_values(ascending = False)
null_val_per =(null_val/train.shape[0])*100
l= [null_val, null_val_per]
ll= pd.concat(l, axis =1, keys =['Null Values', 'Null Values percentage']).sort_values('Null Values')

print("missing value details \n\n", ll)
```

```
missing value details
```

|              | Null Values | Null Values percentage |
|--------------|-------------|------------------------|
| MasVnrArea   | 7           | 0.599315               |
| MasVnrType   | 7           | 0.599315               |
| BsmtQual     | 30          | 2.568493               |
| BsmtFinType1 | 30          | 2.568493               |
| BsmtCond     | 30          | 2.568493               |
| BsmtFinType2 | 31          | 2.654110               |
| BsmtExposure | 31          | 2.654110               |
| GarageCond   | 64          | 5.479452               |
| GarageQual   | 64          | 5.479452               |
| GarageType   | 64          | 5.479452               |
| GarageFinish | 64          | 5.479452               |
| GarageYrBlt  | 64          | 5.479452               |
| LotFrontage  | 214         | 18.321918              |
| FireplaceQu  | 551         | 47.174658              |
| Fence        | 931         | 79.708904              |
| Alley        | 1091        | 93.407534              |
| MiscFeature  | 1124        | 96.232877              |
| PoolQC       | 1161        | 99.400685              |

### Checking Duplicate:

No duplicate values are present in the dataset.

```
train.duplicated().sum()
```

```
Out[10]: 0
```

### Null Value Imputation:

To analyse the null, we remove all the features with high null value percentage and drop two unnecessary columns for both train and test dataset.

```
test.drop(['PoolQC', 'MiscFeature', 'Alley', 'Fence', 'FireplaceQu'],axis =1, inplace = True)
```

```
test.drop(['Id', 'Utilities'],axis=1,inplace=True)
```

Other null values are imputed by mean, median or mode of the corresponding features for both train and test dataset.

```
test['LotFrontage'] = test['LotFrontage'].fillna(test['LotFrontage'].median())
test['GarageType'] = test['GarageType'].fillna(test['GarageType'].mode()[0])
test['GarageYrBlt'] = test['GarageYrBlt'].fillna(test['GarageYrBlt'].mode()[0])
test['GarageFinish'] = test['GarageFinish'].fillna(test['GarageFinish'].mode()[0])
test['GarageQual'] = test['GarageQual'].fillna(test['GarageQual'].mode()[0])
test['GarageCond'] = test['GarageCond'].fillna(test['GarageCond'].mode()[0])
test['BsmtFinType2'] = test['BsmtFinType2'].fillna(test['BsmtFinType2'].mode()[0])
test['BsmtExposure'] = test['BsmtExposure'].fillna(test['BsmtExposure'].mode()[0])
test['BsmtFinType1'] = test['BsmtFinType1'].fillna(test['BsmtFinType1'].mode()[0])
test['BsmtCond'] = test['BsmtCond'].fillna(test['BsmtCond'].mode()[0])
test['BsmtQual'] = test['BsmtQual'].fillna(test['BsmtQual'].mode()[0])
test['MasVnrType'] = test['MasVnrType'].fillna(test['MasVnrType'].mode()[0])
test['MasVnrArea'] = test['MasVnrArea'].fillna(test['MasVnrArea'].median())
test['Electrical'] = test['Electrical'].fillna(test['Electrical'].mode()[0])
```

There are some columns with same meaning. Drop those columns also for both train and test dataset.

```
test.drop(['BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF'], axis=1, inplace=True)
test.drop(['1stFlrSF', '2ndFlrSF', 'LowQualFinSF'], axis=1, inplace=True)
```

### Converting Age Columns:

Converting years column to age column for both train and test dataset.

```
test['YearBuilt_age'] = test['YearBuilt'].max() - test['YearBuilt']
test['YearRemodAdd_age'] = test['YearRemodAdd'].max() - test['YearRemodAdd']
test['YrSold_age'] = test['YrSold'].max() - test['YrSold']
test['GarageYrBlt_age'] = test['GarageYrBlt'].max() - test['GarageYrBlt']

# dropping old columns
test.drop(['YearBuilt', 'YearRemodAdd', 'YrSold', 'GarageYrBlt'], axis=1, inplace = True)
```

### Data Cleaning:

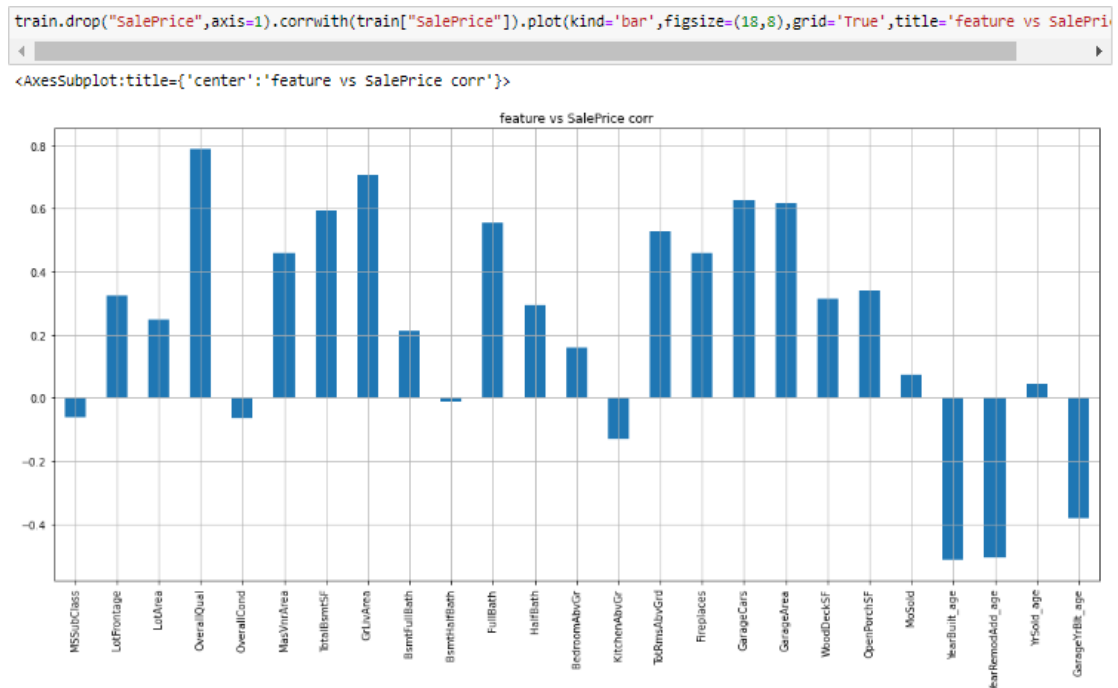
Drop some unnecessary columns also like "Condition1", "Condition2", 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'MiscVal' and 'PoolArea' as they have no such variation of their different unique values for both train and test dataset.

```
test.drop(["Condition1", "Condition2"], axis=1, inplace= True)
test.drop(['EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'MiscVal', 'PoolArea'], axis=1, inplace=True)
```

### Correlation:

All the features are correlated with target. Overall quality and total ground area is highly correlated with target variable sales.





### Outlier Detection of training data:

A lot of outliers contains in upper side that is the dataset have right skewness. Let's keep all the data as the deletion of data may cause information loss.

### Skewness:

Skewness is present in the dataset. Let's say for this case, the skewness range is -1 to +1. Any data have skewness above this level, are skewed. Use PowerTransformer to remove skewness.

```
skew_data = ['MSSubClass', 'GrLivArea', 'WoodDeckSF',
             'TotalBsmtSF', 'OpenPorchSF',
             'LotFrontage', 'MasVnrArea', 'BsmtHalfBath',
             'KitchenAbvGr', 'LotArea']
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method = 'yeo-johnson')

data[skew_data] = scaler.fit_transform(data[skew_data].values)
data.head()
```

### Encoding for data:

The categorical Variable in training & testing dataset are converted into numerical datatype using label encoder.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in cat_features:
    data[i] = le.fit_transform(data[i])
data.head()
```



## Standard Scaling of data:

### Standard Scaling of train data:

```
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()

# Splitting data in target and dependent feature
x = data.drop(['SalePrice'], axis =1)
y = data['SalePrice']
x_scale = scaler.fit_transform(x)
```

### Standard Scaling of test data:

```
x_scale_test = scaler.fit_transform(test)
```

## Data Inputs- Logic- Output Relationships

We can see in the correlation that every features are correlated with each other and also they are highly correlated with target variable Sale Price.

### State the set of assumptions (if any) related to the problem under consideration

No such assumptions are considered in this problem.

## Hardware and Software Requirements and Tools Used

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz

RAM: 4.00 GB

System Type: 64-bit operating system, x64-based processor

Window: Windows 10 Pro

Anaconda – Jupyter Notebook

Libraries Used –

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

Except this, different libraries are used for machine learning model building from sklearn.

### 3. Model/s Development and Evaluation

#### Identification of possible problem-solving approaches (methods)

In this problem regression-based machine learning algorithm like linear regression can be used. For that first data encoding and data scaling using standard scalar is done. For building an appropriate ML model before implementing regression algorithms, data is split in training & test data using `train_test_split`. Then different statistical parameter like R2 Score, MSE, MAE, RMSE score is determined for every algorithm. Hyper parameter tuning performed to get the accuracy score much higher and accurate than earlier.

Then cross-validation is done to check best CV Score. Overfitting problem is also checked by Lasso and Ridge. After observing all the above score, the final model is determined.

#### Testing of Identified Approaches (Algorithms)

Total 7 algorithms used for the training and testing are:

1. Linear Regression.
2. DecisionTreeRegressor
3. KNeighborsRegressor
4. RandomForestRegressor
5. SupportVectorRegressor
6. GradientBoostingRegressor
7. AdaBoostRegressor

#### Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import `mean_absolute_error`, `mean_squared_error` and `r2_score`. From model\_selection also, we use `cross_val_score`. Those are the matrices use to validate the model's quality.

#### Run and Evaluate selected models

First find the best random state of `train_test_split` to get best accuracy. Here the random state is 654. Then after splitting the data into 4 different part, check the shape of the data.

```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (876, 59)
Training target shape: (876,)
Test feature shape: (292, 59)
Test target shape: (292,)
```

## A Linear Regression:

```
x_train,x_test,y_train,y_test = train_test_split(x_scale,y,test_size = 0.25, random_state=654)

lin_reg= LinearRegression()

lin_reg.fit(x_train, y_train)

y_pred = lin_reg.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.8816206614832436
Mean absolute error: 19209.648878261374
Mean square error: 653721476.9612099
Root mean square error: 25567.977568849867
```

For every model, we use Hyperparameter tuning to get the best accuracy. One of the snapshot is attached. This same code is applied for rest of the 6 models.

### Hyperparameter Tuning Using GridSearchCV:

```
from sklearn.model_selection import GridSearchCV

grid = dict(fit_intercept=['True', 'False'], n_jobs=[1,-1])

grid_lin = GridSearchCV(estimator=lin_reg, param_grid= grid, cv=9)

grid_lin.fit(x_train, y_train)
grid_lin.best_params_

{'fit_intercept': 'True', 'n_jobs': 1}
```

```
grid_lin_best = LinearRegression(fit_intercept= 'True', n_jobs= 1)

grid_lin_best.fit(x_train, y_train)
y_pred = grid_lin_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.8816206614832436
Mean absolute error: 19209.648878261374
Mean square error: 653721476.9612099
Root mean square error: 25567.977568849867
```

## ***B*** Decision Tree Regressor:

```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)

y_pred = dt.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:', np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7501874814946436
Mean absolute error: 25915.321917808218
Mean square error: 1379529659.541096
Root mean square error: 37142.02013274313
```

## ***C*** KNeighbors Regressor:

First determine particular K value for minimim RMSE score. Here k= 5 gives the best result.

```
knn =KNeighborsRegressor(n_neighbors= 5)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:', np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.8484815342376684
Mean absolute error: 20412.041780821914
Mean square error: 836724351.2771233
Root mean square error: 28926.187983851647
```

#### **D RandomForestRegressor:**

```
Using RandomForestRegressor():  
In [158]: from sklearn.ensemble import RandomForestRegressor  
          rf= RandomForestRegressor()  
          rf.fit(x_train, y_train)  
          y_pred = rf.predict(x_test)  
          print("R2 Score:", r2_score(y_test, y_pred))  
          print('Mean absolute error:', mean_absolute_error(y_test, y_pred))  
          print('Mean square error:', mean_squared_error(y_test, y_pred))  
          print('Root mean square error:', np.sqrt( mean_squared_error(y_test, y_pred)))  
  
R2 Score: 0.9844728480540242  
Mean absolute error: 16305.737671232879  
Mean square error: 527525762.8778801  
Root mean square error: 22967.929007150916
```

#### **E Support Vector Regressior:**

For different kernel ('rbf', 'poly', 'linear'), the same code is applied and finally take kernel='linear' as it gives the best score.

```
from sklearn.svm import SVR  
svr_rbf = SVR(kernel='rbf')  
svr_rbf.fit(x_train, y_train)  
  
y_pred = svr_rbf.predict(x_test)  
  
print('R2 Score:', r2_score(y_test, y_pred))  
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))  
print('Mean square error:', mean_squared_error(y_test, y_pred))
```

```
R2 Score: -0.10138556204325022  
Mean absolute error: 55211.98959111769  
Mean square error: 6082137350.52043
```

#### **F Gradient Boosting Regressor:**

```
from sklearn.ensemble import GradientBoostingRegressor  
gbdt= GradientBoostingRegressor()  
gbdt.fit(x_train, y_train)  
  
y_pred = gbdt.predict(x_test)  
  
print('R2 Score:', r2_score(y_test, y_pred))  
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))  
print('Mean square error:', mean_squared_error(y_test, y_pred))  
print('Root mean square error:', np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9225635206080466  
Mean absolute error: 15039.738727809416  
Mean square error: 427624366.82832634  
Root mean square error: 20679.08041544223
```



## G Ada Boost Regressor:

```
from sklearn.ensemble import AdaBoostRegressor

ada= AdaBoostRegressor()
ada.fit(x_train, y_train)

y_pred = ada.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:', np.sqrt( mean_squared_error(y_test, y_pred)))

R2 Score: 0.8023081985086989
Mean absolute error: 24301.575415917385
Mean square error: 1091705512.7464035
Root mean square error: 33040.967188422364
```

As per 7 different regression model we can see that the model with maximum R2 score and minimum RMSE value is GradientBoostingRegressor().

### Cross validation:

Let's check the cross validation score taking cross fold =5, before final prediction. Each of the model with best accuracy score of that corresponding model is taken. Here also GradientBoostingRegressor() is the best model with max CV score and min standard deviation.

```
from sklearn.model_selection import cross_val_score

all_models = [lin_reg , grid_dt_best , grid_knn_best , rf , grid_svr_best , grid_gbd_t_best, grid_ada_best]

for i in all_models:
    cvscore = cross_val_score(i, x_scale,y, cv = 5)
    print('Cross Validation Score of :',i)
    print("\n Cross Validation Score : " ,cvscore)
    print("\nMean CV Score :",cvscore.mean())
    print("\nStd deviation :",cvscore.std())
    print("\n-----")
    print("-----")
```

## Overfitting Checking:

Now overfitting problem is checked using Lasso and Ridge.

```
from sklearn.linear_model import Ridge, Lasso, RidgeCV, LassoCV
#LASSO Regression
lassocv= LassoCV()
lassocv.fit(x_train, y_train)
alpha= lassocv.alpha_ #best learning rate for LASSO
alpha

1926.8413624973484

lasso_reg=Lasso(alpha)
lasso_reg.fit(x_train,y_train)
print ("Score after applying LASSO regression on the model is :", lasso_reg.score(x_test, y_test))

Score after applying LASSO regression on the model is : 0.8795786186447174

#RIDGE Regression

ridgecv= RidgeCV(alphas=np.arange(0.0001, 1.0, 0.001), normalize= True, cv=10 )
ridgecv.fit(x_train, y_train)
alpha= ridgecv.alpha_ #best learning rate for RIDGE
alpha

0.3751

ridge_reg=Ridge(alpha)
ridge_reg.fit(x_train,y_train)
print ("Score after applying LASSO regression on the model is :", ridge_reg.score(x_test, y_test))

Score after applying LASSO regression on the model is : 0.881648924570051
```

After using LASSO() and Ridge(), there is no large change in score. So this model is not OVERFITTED. So the final model is GradientBoostingRegressor() for this particular dataset.

## Final Model:

```
print('Final R2 Score:', r2_score(y_test, y_pred))
print('\nFinal Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('\nFinal Mean square error:', mean_squared_error(y_test, y_pred))
print('\nFinal Root mean square error:', np.sqrt( mean_squared_error(y_test, y_pred)))

Final R2 Score: 0.9243063124706136

Final Mean absolute error: 15077.59305866895

Final Mean square error: 418000217.1692032

Final Root mean square error: 20445.053611306652
```



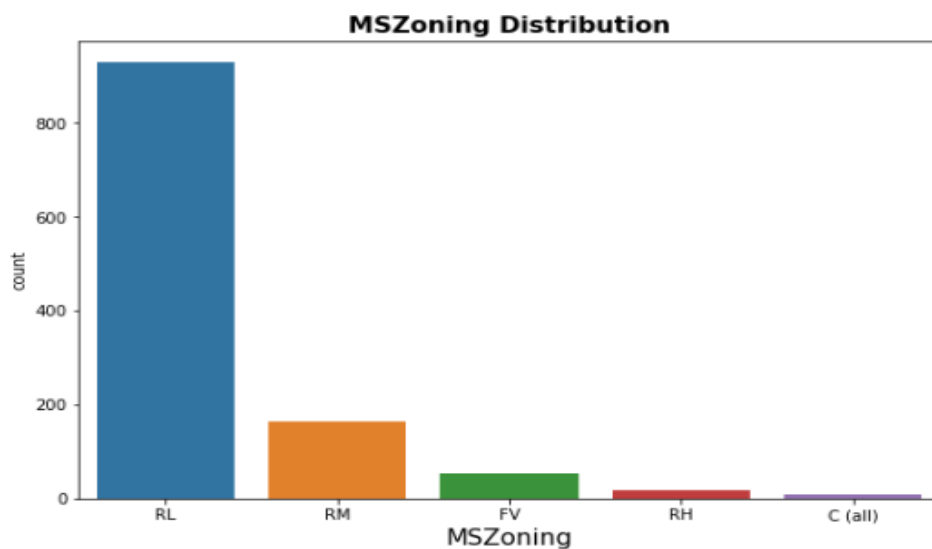
## Visualizations:

Let's start the observation exploration of feature analysis.



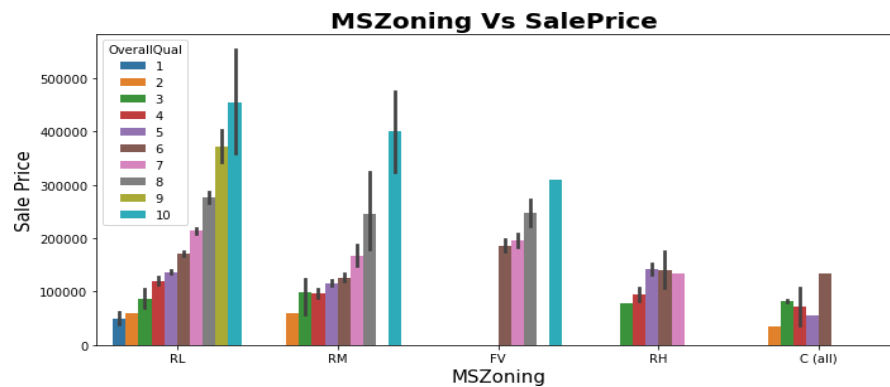
## Observations:

For MSSubClass class 20 and 60, SalePrice is high and maximum MSZoning is Residential Low Density



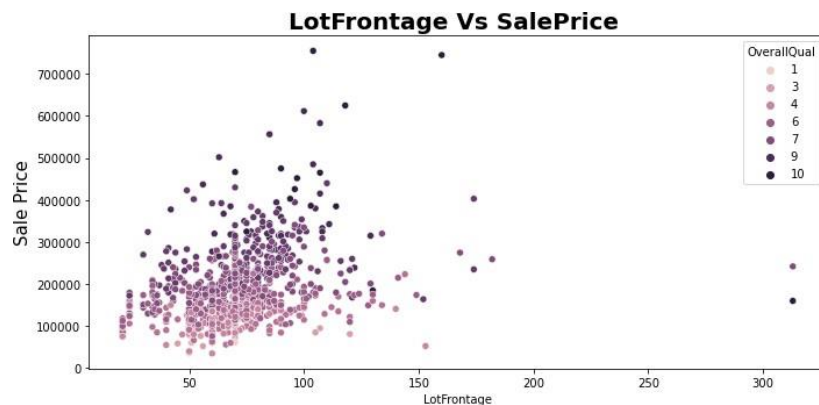
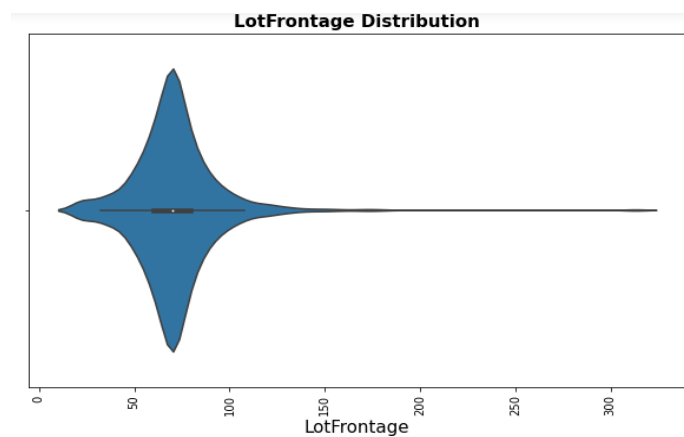
### Observations:

1. 79.5% of house are in Residential Low Density followed by 14 % of house are in Residential Medium Density area.
2. Floating Village Residential is only 4.5%
3. Very few house (0.8%) are in Commercial zone.



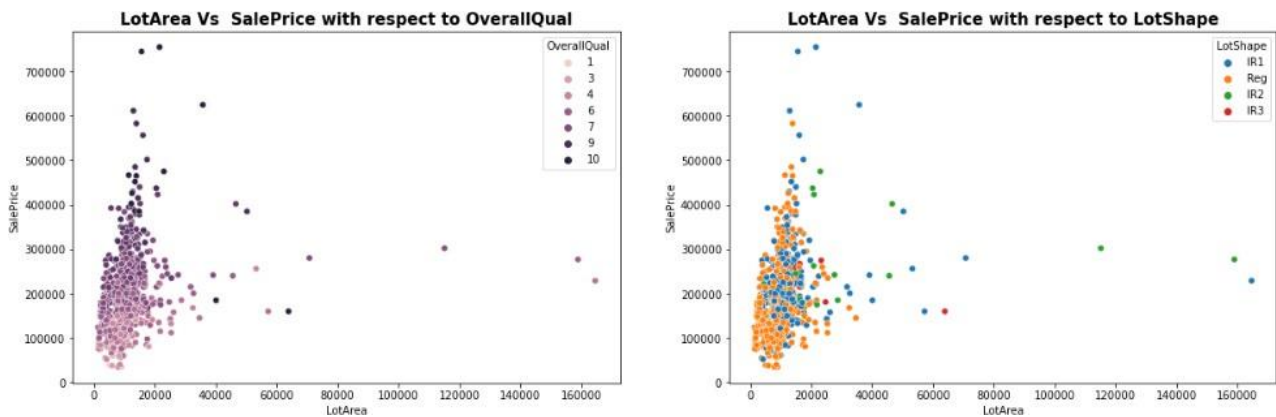
### Observations:

1. Max property have overall condition rating of either 6 to 7.
2. Average selling price for RL zone property is 200000-500000.
3. In Commercial zone the property price is minimum.
4. Sale Price inside RL Zone is much higher than other zone.
5. It is clear that if OverallQual increased, SalePrice also increased.



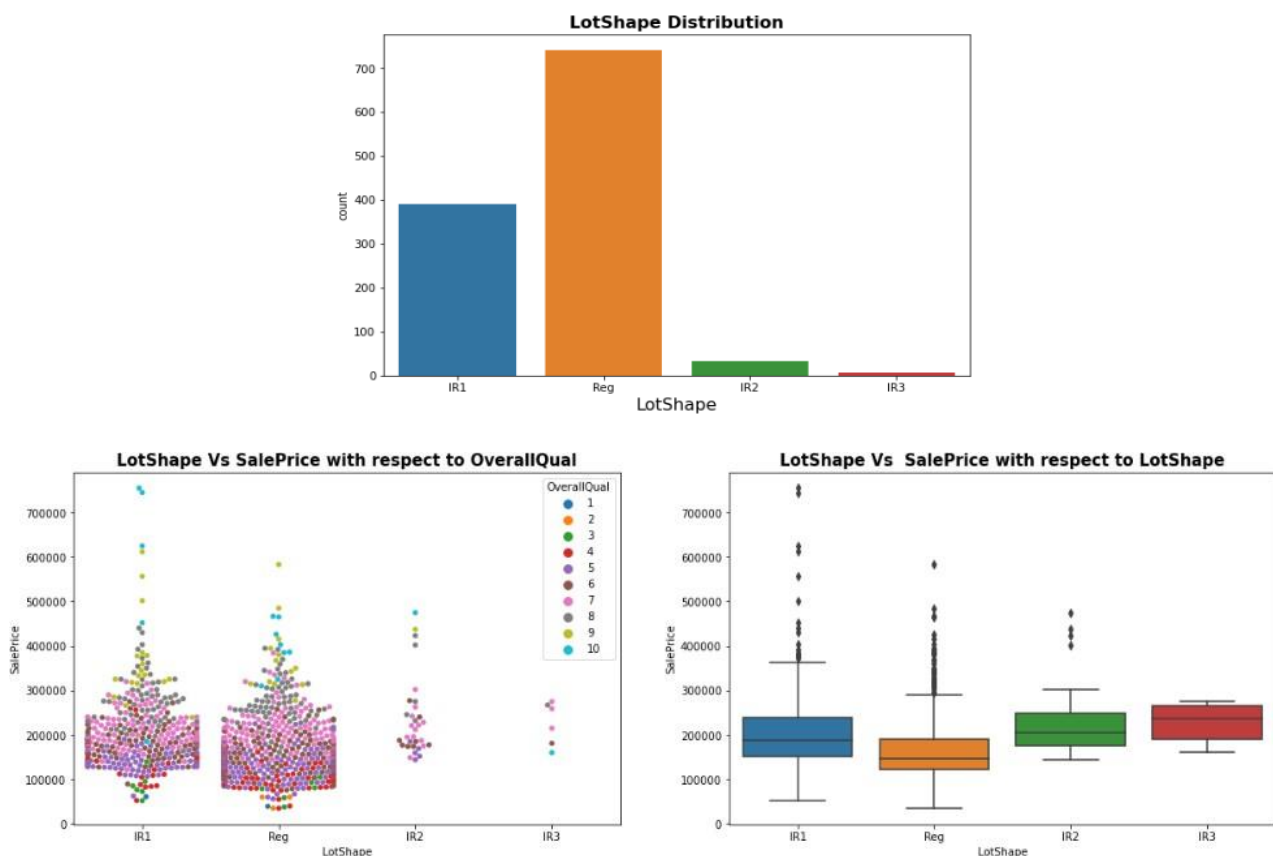
### Observations:

1. Average LotFrontage is around 50-80
2. A lot of outliers are present.
3. There is No Significant relationship found between SalePrice & LotFrontage.



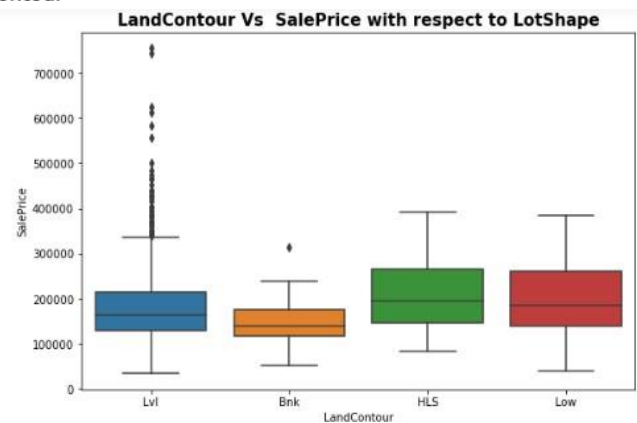
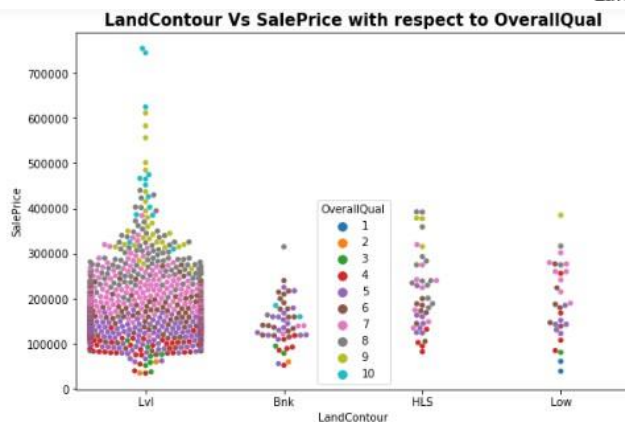
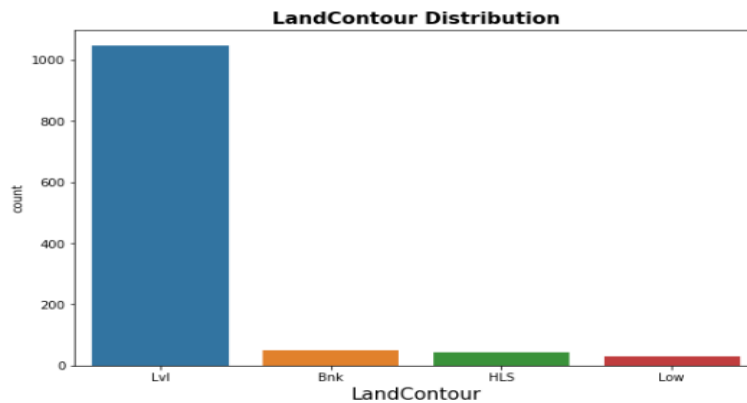
### Observations:

1. Average LotArea is around 0-25000
2. A lot of outliers are present.
3. As OverallQual increased, selling price of the property also increased.
4. No relation is found between LotArea and SalePrice with respect to LotShape.



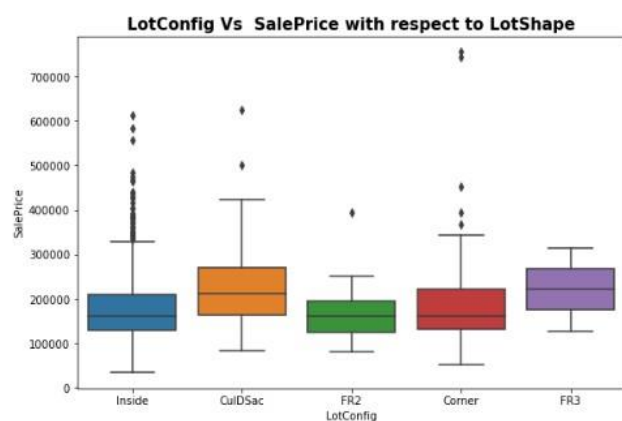
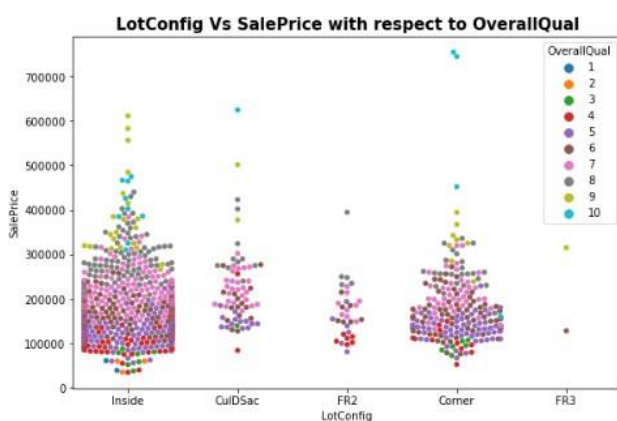
### Observations:

1. Min LotShape is Irregular.
2. 63.4% LotShape is Regular followed by Slightly irregular and there is some outliers also with very high price range
3. There is no relationship between LotShape and selling price.



### Observations:

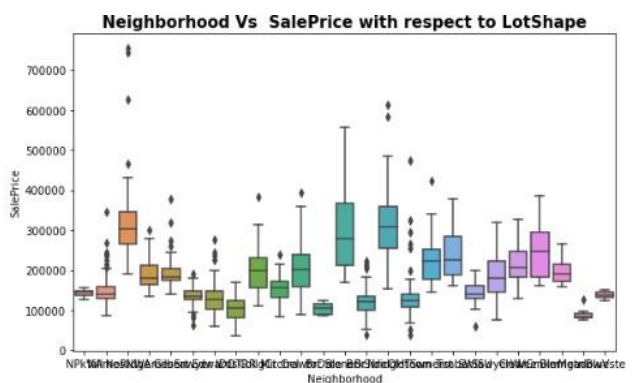
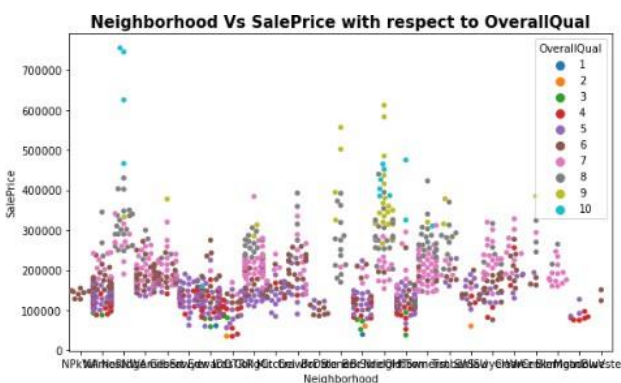
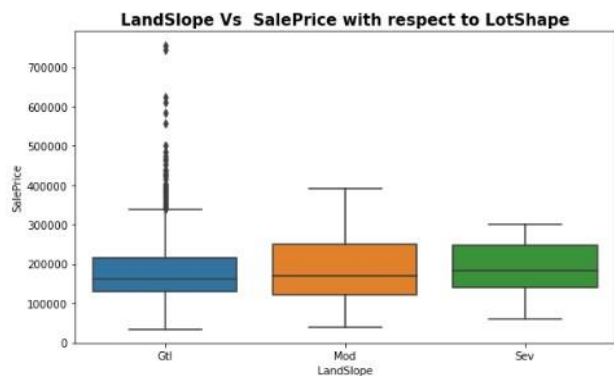
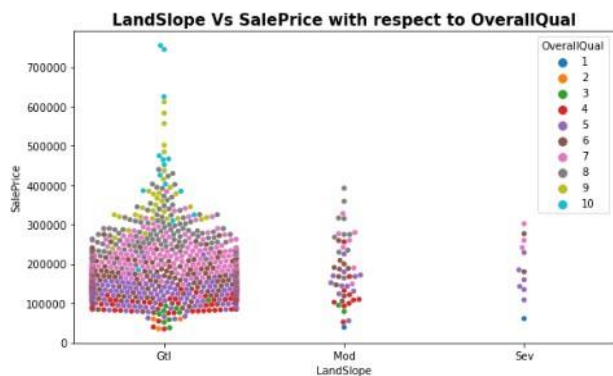
1. Around 90% property is in Level land contour type. It is quite obvious also!!
2. Banked area is the less costly area compare to others.



### Observations:

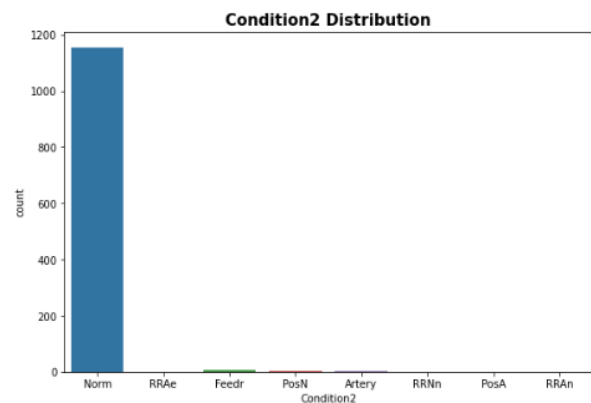
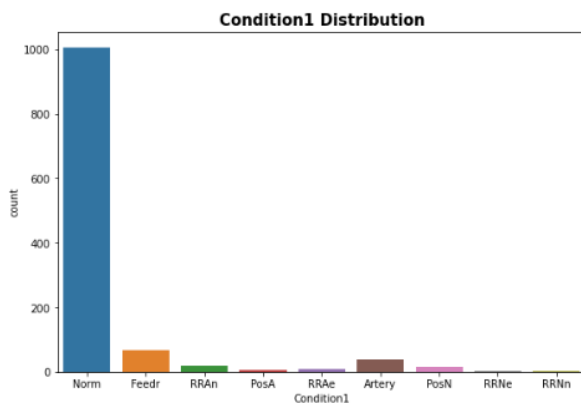
1. 72.1% of house comes with inside Lot configuration.

2. Cheapest property are in Inside lot configuration.
3. Only 2 are in Frontage on 3 sides of property.



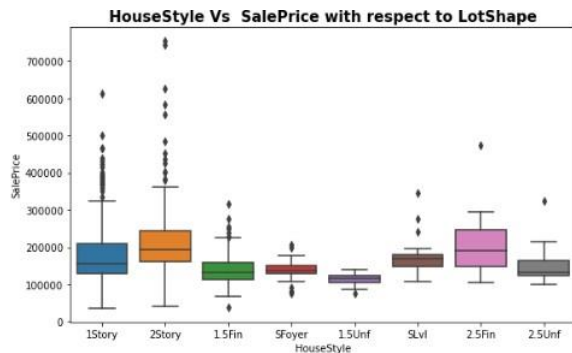
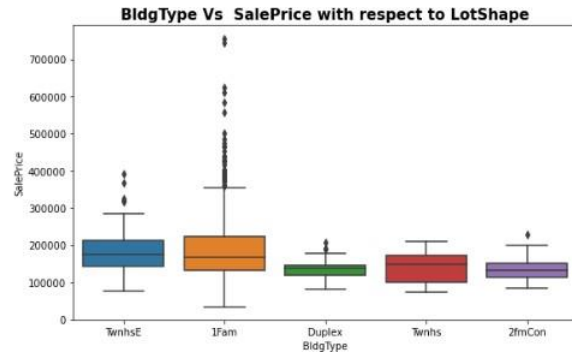
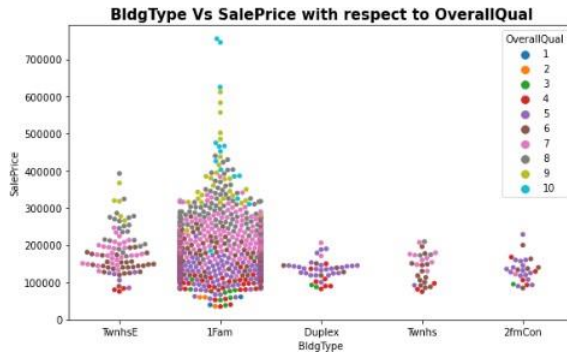
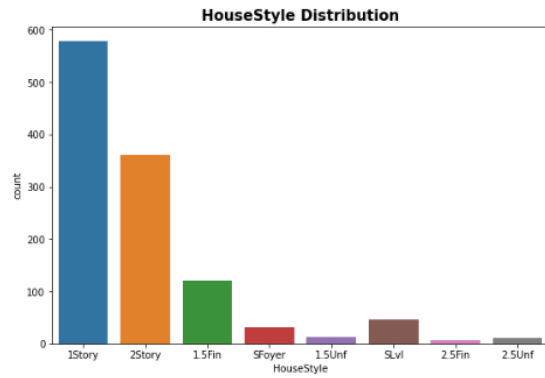
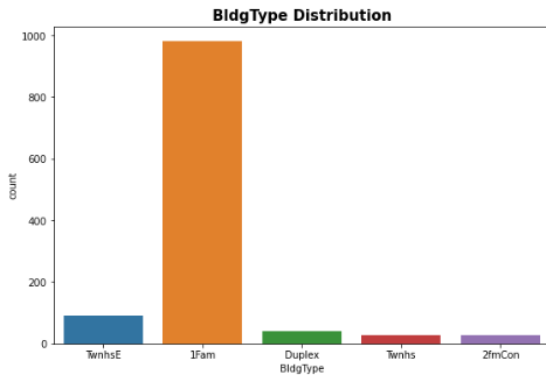
### Observations:

1. 95% LandSlope is Gentle slope.
2. 1% properties come with severe slope and the average price is high compare to Gentle slope.
3. Max Neighborhood is Names with 182 values.



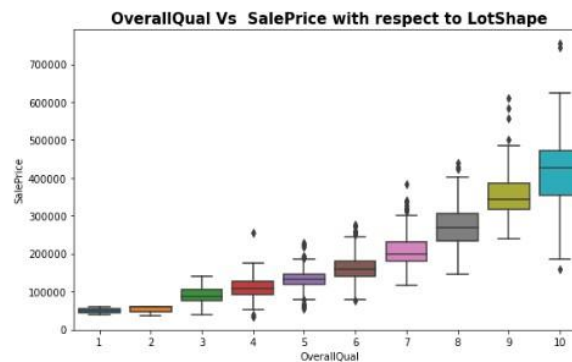
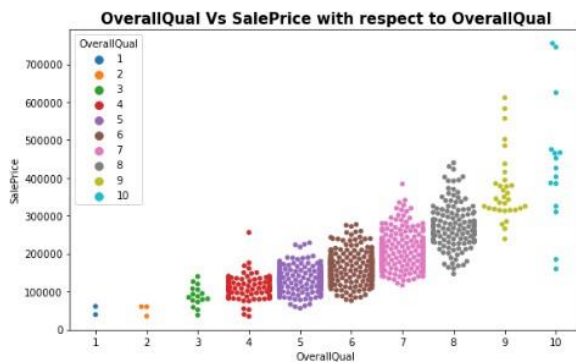
### Observations:

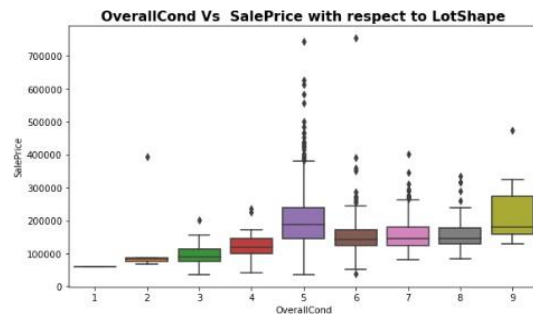
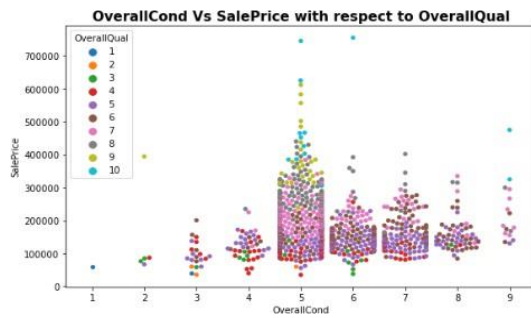
For maximum case the condition is Normal. And there is no such relationship is present in the above graph. So drop these two columns.



## Observations:

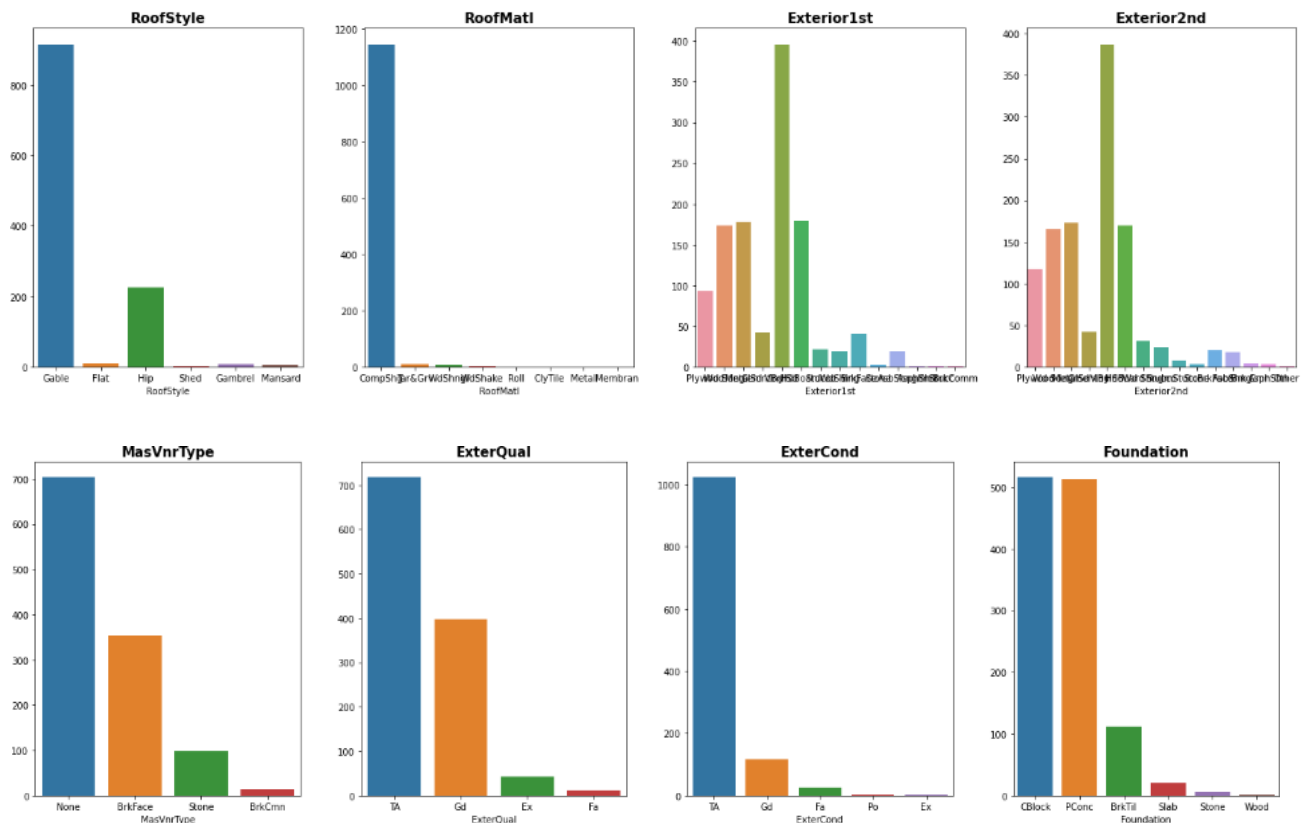
1. Around 1000 house properties are with building type Single-family Detached.
2. More than 550 House Properties are with building type one story.
3. Two story building is costlier than another.





### Observations:

1. Max OverallQual is with rating 5-7.
2. For OverallCond the max value is 5.
3. As usual if Overall Quality is increased, price is also increased.
4. Price is max for OverallCond 5

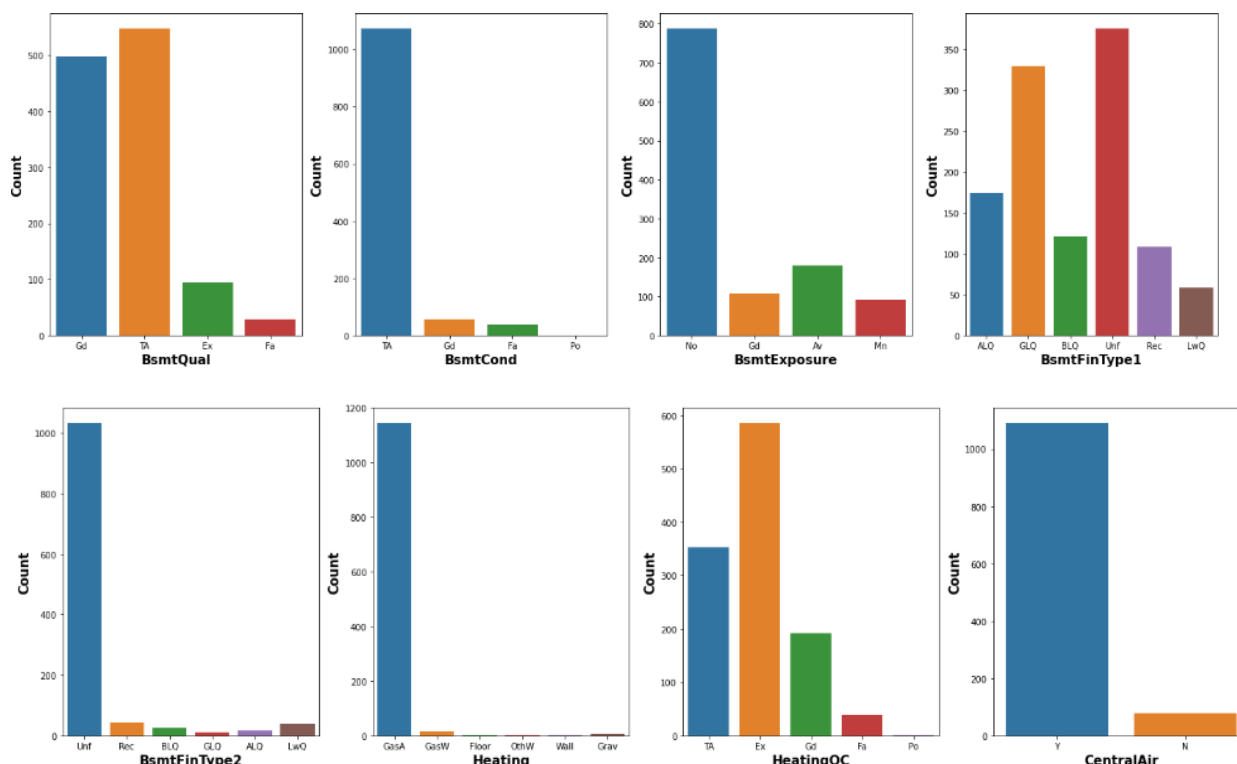


### Observations:

1. 78.3% house is with Gable RoofStyle followed by 19.3 % house is with Hip Style.
2. 98% material is Standard (Composite) Shingle and other 7 types are only 2% togetherly.
3. Around 33% material is Vinyl Siding as Exterior covering on house for both Exterior1st and Exterior2nd.
4. Around 60% of house comes with None as Masonry veneer type. Maybe there is some mistake during entry the data.
5. Around 60% of house comes with Average quality of the material on the exterior.



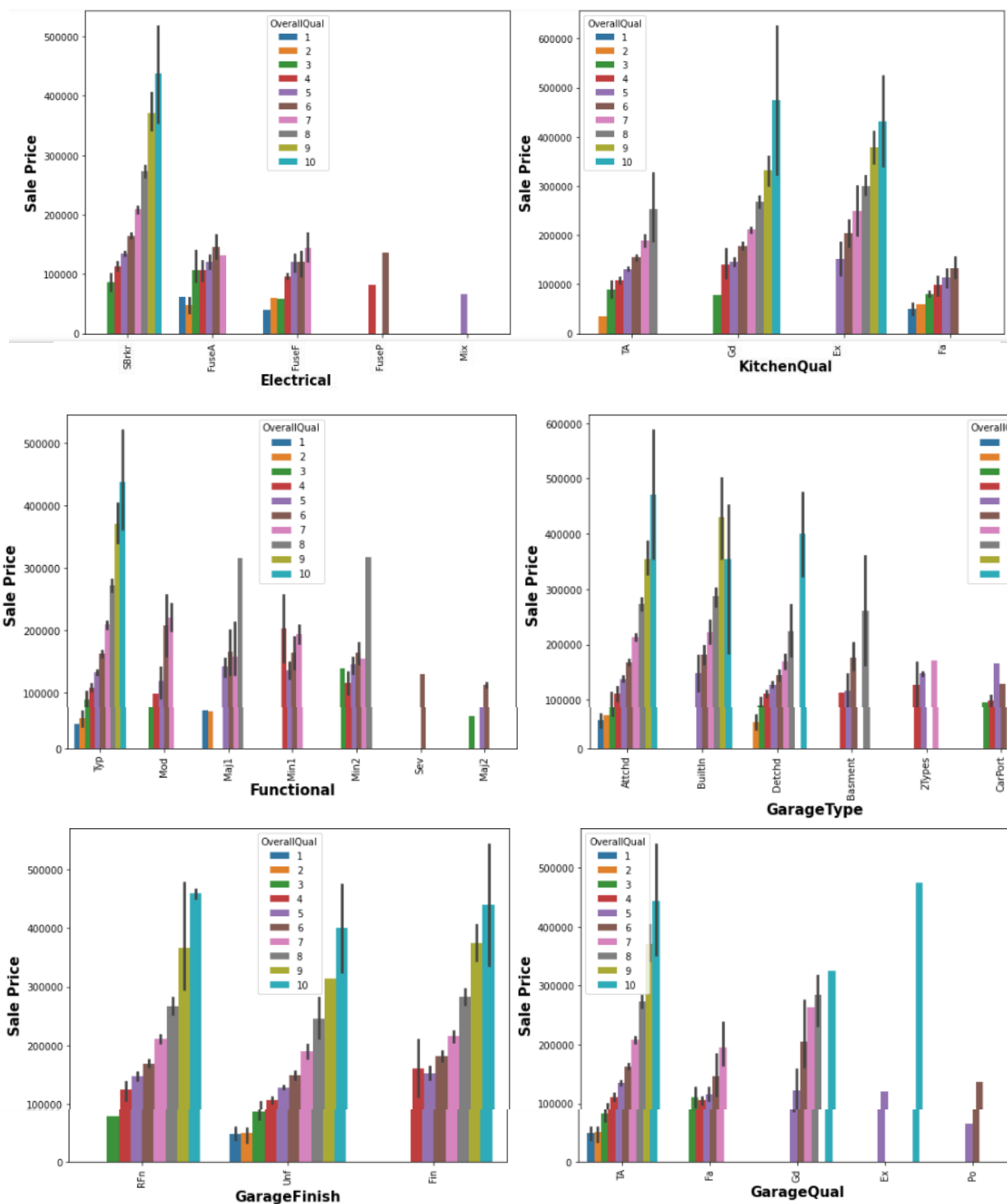
6. 87.5% of house comes with Average/Typical as the present condition of the material on the exterior.
7. Around 44% of house comes with both Cinder Block and Poured Contrete both Type of foundation.
8. Hip style Roof are much costly than rest of the roof style.
9. Wood Shingles is much costly than rest of the roof material.
10. Cement Board is much costly than rest of the Exterior covering on house.
11. The house made with stone as Masonry veneer is most costlier type.
12. Very obviously costlier house come with Excellent and Good exterior quality.
13. Pconc foundation are mostly use in costly housing properties.



Visualising different type of count plot and the relationship between these features ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir') and Sale Price (target variable), we get the following observations.

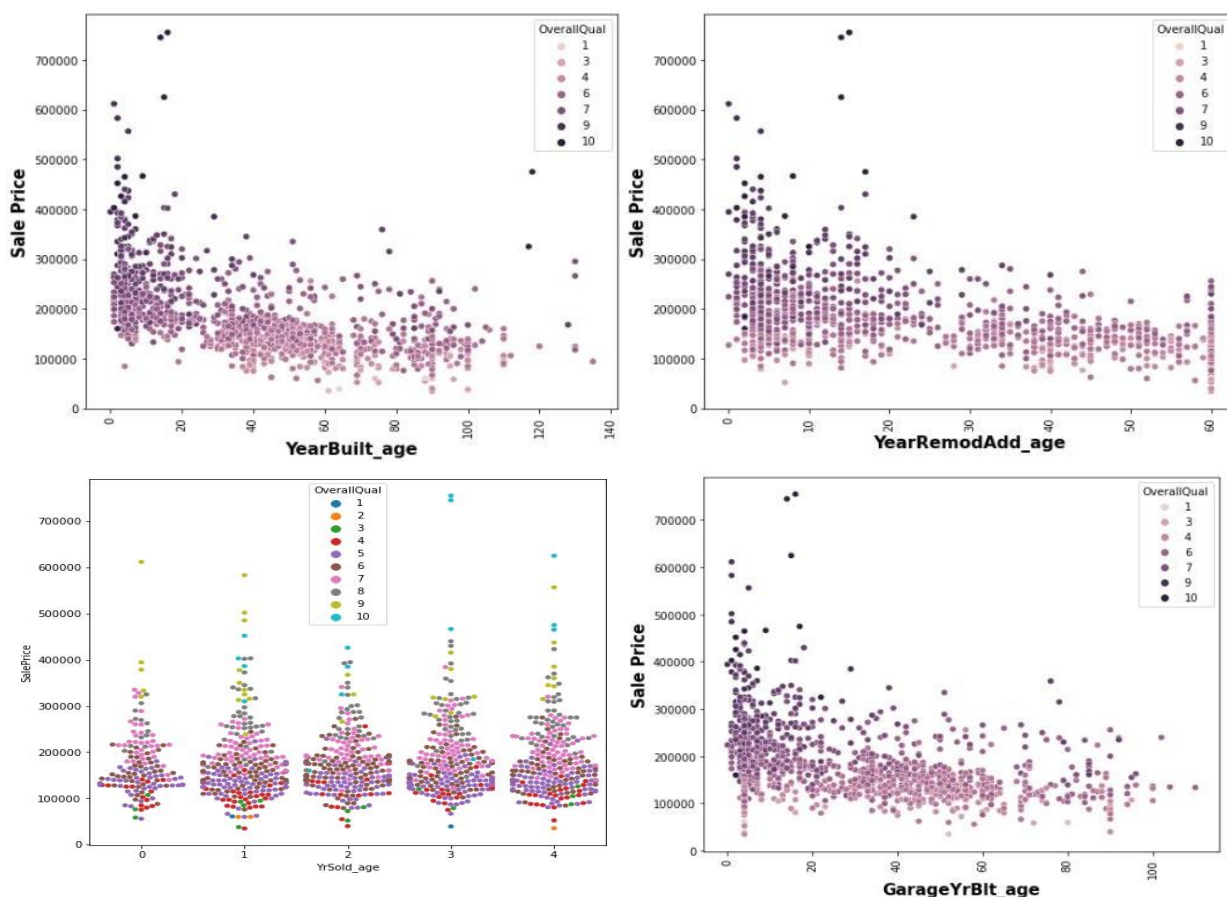
### Observations:

1. Good and Typical are the maximum used basement height.
2. Max number of house is No Exposure with respect to Refers to walkout or garden level walls.
3. Most of the housing type are Good Living Quarters or Unfinished.
4. For maximum case, the Type of heating is Gas forced warm air furnace
5. Max House is Central air conditioning.
6. It is very obvious that Execellent type is much costlier than other types of height of the basement, Heating quality and condition.
7. Central air conditioned housing proper is much costlier than non AC.
8. For maximum case, the Type of heating is Gas forced warm air furnace and it is the costlier type among rest of them.



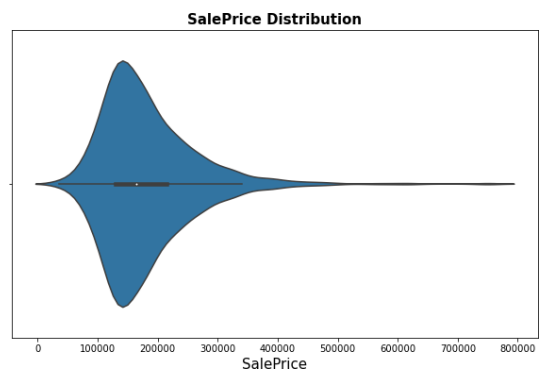
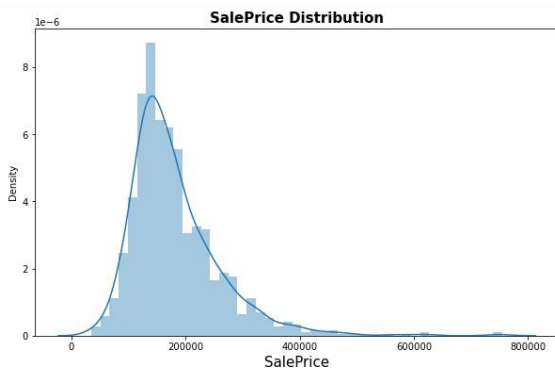
### Observations:

1. Max type of Electrical used in house is Standard Circuit Breakers & Romex and also it is most costly.
2. Max type of KitchenQual used in house is Average and Good. Also it is avg in term of price.
3. Around 95% Home functionality (Assume typical unless deductions are warranted) of the house is Typical Functionality.
4. Above 700 house has attached Garage and it is more convenient to use.
5. Most of the garage is unfinished.
6. Most of the case the quality of garage and Garage condition is Typical/Average.
7. Maximum case, property sale type is Warranty Deed - Conventional and sale condition is Normal Sale. That is the property just constructed and sold category are exceptionally much costlier than anyone else.
8. All loan based sale are below 300000.
9. The Partial category- Home was not completed when last assessed (associated with New Homes) is much costlier than rest.



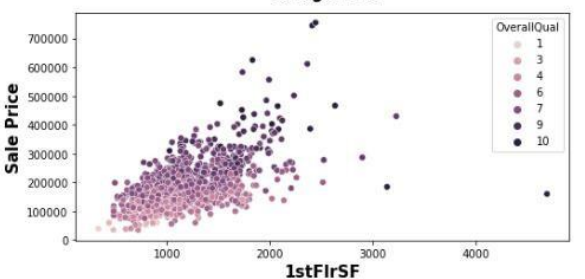
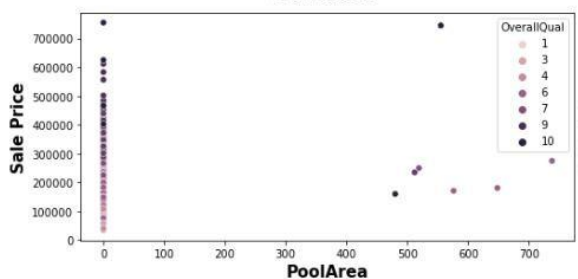
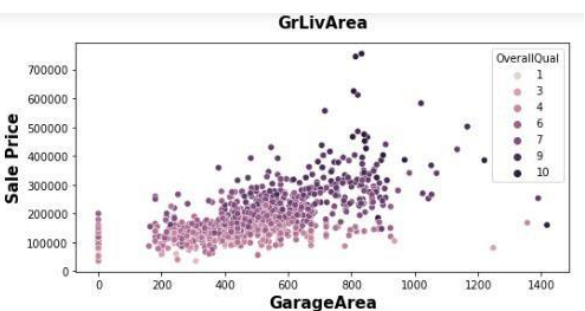
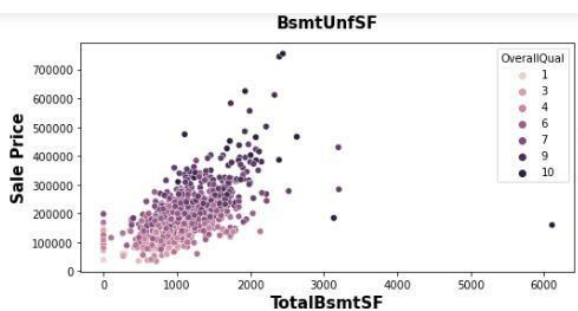
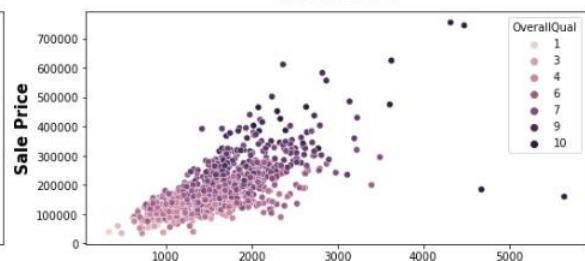
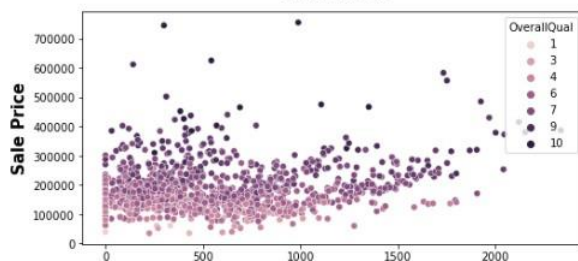
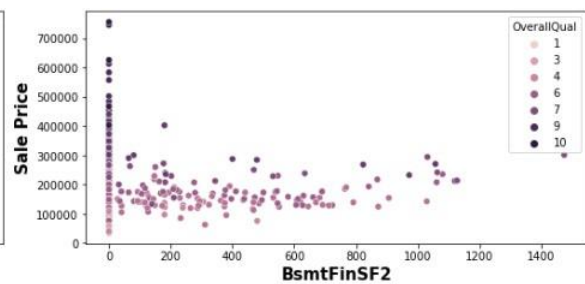
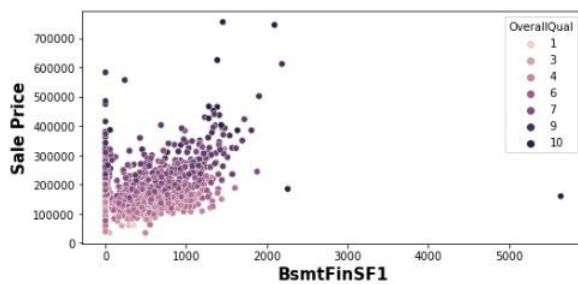
### Observations:

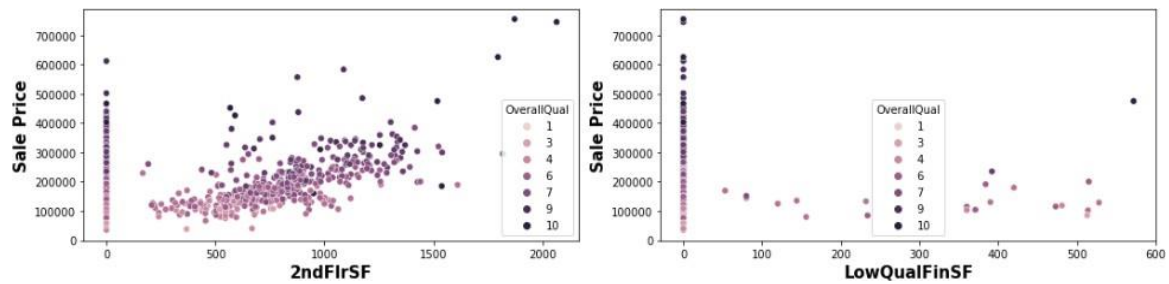
1. The selling price of new property is higher than old one.
2. 10 years after Remodelling the properties, the price start decreasing.
3. The selling price of new garage is also higher than old one.



### Observations:

1. Average SalePrice is 100000 to 300000
2. As OverallQual increased, SalePrice is also increased





### Observations:

1. As total floor area increases the sale price also get increases corresponding the overall quality of House.
2. As Basement Quality increase in relation to it sale Price increases.
3. As total garage area increases the sale price also get increases corresponding the overall quality of House.
4. Here interestingly TotalBsmtSF is sum of BsmtFinSF1, BsmtFinSF2 and BsmtUnfSF. Let's drop those three.
5. Also GrLivArea is sum of '1stFlrSF', '2ndFlrSF', 'LowQualFinSF'. Let's drop those three also.

### Interpretation of the Results

After all the pre-processing steps, the training dataset is ready to train machine learning models. All unnecessary features are deleted as they might give overfitting problem as well as it also could increase the time complexity. Now apply this training dataset on different ML Regression Model (as discussed on part 3.4 - 'Run and Evaluate selected models') and check the best model for this particular dataset.

## 4. CONCLUSION

### Key Findings and Conclusions of the Study

*Tables of Findings using different algorithms after Hyper Parameter Tuning*

| Algorithm                   | R2 Score | RMSE Value | CV Score | Standard Deviation |
|-----------------------------|----------|------------|----------|--------------------|
| Linear Regression           | 0.882    | 25567      | 0.789    | 0.052              |
| Decision Tree Regressor     | 0.788    | 34209      | 0.738    | 0.052              |
| KNeighbors Regressor        | 0.862    | 27563      | 0.790    | 0.030              |
| Random Forest Regressor     | 0.904    | 22967      | 0.829    | 0.035              |
| SVR                         | 0.632    | 45080      | 0.595    | 0.038              |
| Gradient Boosting Regressor | 0.924    | 20445      | 0.853    | 0.035              |
| Ada Boost Regressor         | 0.824    | 31123      | 0.766    | 0.049              |

Here Gradient Boosting Regressor giving maximum R2 Score, minimum RMSE Value, Maximum CV Score and minimum Standard Deviation. So Gradient Boosting Regressor is selected as best model. The final R2 Score is 0.924 that is 92.4%.

### Learning Outcomes of the Study in respect of Data Science

- Null removal is an important part of any problem.
- Scaling and standardization of data is mandatory.
- Feature selection played an important role in any ML problem. Unnecessary features and the features correlated with another needs to be removed.
- Most of the case accuracy score is improved after applying hyper parameter tuning.
- Overfitting check is an important part, it may effect on actual R2 Score.
- Data needs to be much precise and detailed for much better score.

### **Limitations of this work and Scope for Future Work**

- Some additional features can be added to the dataset. For this addition we can able to performsome more advance model of ML.
- Artificial Neural Network can be used create more accurate model.
- The model will be more accurate if the dataset (set of observations) is bigger.
- The model will be more accurate if there is no missing data.

THANK YOU