HOUSING: PRICE PREDICTION

Submitted by:

Kishan Barochiya

ACKNOWLEDGMENT

Here, I acknowledge that below presented data is as per my best view. Data are got from Flip robo and this data is for study purpose only. For best view purpose, here use different libraries and also taken help from google and other blogs. Here, some considerations are taken as per my best knowledge and I acknowledge that all data and models here I used are proceed and treated as my best view.

INTRODUCTION

• Business Problem Framing

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. 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. You 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:

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

• Conceptual Background of the Domain Problem

This data is collected from field where they got a data of sold houses in Australia in last some years. This data contained different types 81 variables. This all variables are important to decide price of house. For example, how many years old house have? How many square feet it have? In which area, how many bedroom, condition of house etc. Client is getting issue to predict price of that house because they need to give price, which is market compatible and best for them also. Therefore, without any analytics they give price, which is sometimes beneficial, or some time loss making. If we analyze past data and can make prediction model, which can predict price on past data so it will be problem-solving feature for them.

• Review of Literature

We studied data statistically, where check for data frame shape then find any null values are there or not? Checking for data types and then learn data status by using basic method. Visualise data using Seaborn and Matplotlib. Checking for outliers and remove it.checking for skewness and make data perfect for model building. Check about data types, outliers,

skewness, relation before model building. Scaling all data on low scale to get accurate model. As we have big dataset so, we used PCA to dimensionality reduction.

We use feature selection as we have 81 features so we need to work on selected features which are important and we need to work with that features so it becomes time saving. Then we trained this data and input towards model of regression.

We will do hyper parameter tuning for model, which have higher accuracy so we can try for more accuracy then we will see their graph plotting for better understanding and then dump the model and retest it.

- Motivation for the Problem Undertaken

This model will become very useful for buyer and seller of house at it give best price for every houses so buyer gets house at best price and seller got best price for house. There are 10-15 main features on basis clients are ready to pay price. So we will collected that prime features which effect the price of house on primary basis. This model can be changes over time span as there are different situation and demand of house market over the timespan so it changes price and features important over time. For example, before some years there is value of zone x is high so house price was higher in that area but now as zone y get good projects and developing some good commercial plots so now price of house in zone Y is higher compare to zone x. thus we need to stand and give proper weightage to this features.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

First we load the dataset and check some basic statistically as per below.

```
df = pd.read_csv("HOUSE_TRAIN.csv")
pd.set_option('display.max_columns', None)
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condi
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	

What is statistics say for our data?

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
```


Using df.info() and df.describe() we can get data statistics, where we can get data types ,null values available or not, data plotting, highest and lowest data points, avg of data columns etc. This will help use to understand data that how data is plotting and is there any outliers or not, based on it, we can drop column, fill null values and remove outliers etc.

Is there any null values ?

```
df.isnull().sum() #NULL VALUES CHECKING
```

```
Dwelling           0
Zone               0
Lotfrontage      214
SQFT               0
Street             0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 77, dtype: int64
```

As we can see it there is null values in dataset so we can proceed it forward.

From above table, we can conclude that some factors like no of recharge, last recharge amount are the key factor, which can show difference between 0 and 1. Average payback time is not much look effective.

```
categorical_df=[j for j in df if df[j].dtype == 'object']
categorical_df1 =[k for k in df1 if df1[k].dtype == 'object']
```

```
for i in categorical_df:
    columns = df[i].unique()
    print(i,columns)
```

```
Zone ['RL' 'RM' 'FV' 'RH' 'C (all)']
Street ['Pave' 'Grvl']
LotShape ['IR1' 'Reg' 'IR2' 'IR3']
Flatness ['Lvl' 'Bnk' 'HLS' 'Low']
Utilities ['AllPub']
```

Using above formula, we will get unique values in object columns, so it will help us to check unique value in all columns, if there any duplicate values etc.

- # Data Sources and their formats

This data is got from Client Company. They collected this data from field where they collect data of all sold house in their zone and may be error present in this data as this data collected manually. There are 81 variable collected for analysis purpose. This data is in csv files and it have objective columns as well as integer and float type data.

What is the shape of data?
We have 1168 rows and 77 variables in our data set.

```
df.shape
```

```
(1168, 77)
```

- Data Pre-processing:

  We will handle null values by dropping column where null values are higher than 60% and use mode method for object column and mean method for int data column.

```
df['BsmtQual'] = df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df1['BsmtQual'] = df1['BsmtQual'].fillna(df1['BsmtQual'].mode()[0])
```

```
df['BsmtCond'] = df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtFinType1'] = df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0])
df1['BsmtCond'] = df1['BsmtCond'].fillna(df1['BsmtCond'].mode()[0])
df1['BsmtExposure'] = df1['BsmtExposure'].fillna(df1['BsmtExposure'].mode()[0])
df1['BsmtFinType1'] = df1['BsmtFinType1'].fillna(df1['BsmtFinType1'].mode()[0])
df['BsmtFinType2'] = df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0])
df['FireplaceQu'] = df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0])
df['GarageType'] = df['GarageType'].fillna(df['GarageType'].mode()[0])
df1['BsmtFinType2'] = df1['BsmtFinType2'].fillna(df1['BsmtFinType2'].mode()[0])
df1['FireplaceQu'] = df1['FireplaceQu'].fillna(df1['FireplaceQu'].mode()[0])
df1['GarageType'] = df1['GarageType'].fillna(df1['GarageType'].mode()[0])
df['GarageFinish'] = df['GarageFinish'].fillna(df['GarageType'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond'] = df['GarageCond'].fillna(df['GarageCond'].mode()[0])
df['Fence'] = df['Fence'].fillna(df['Fence'].mode()[0])
df1['GarageFinish'] = df1['GarageFinish'].fillna(df1['GarageType'].mode()[0])
df1['GarageQual'] = df1['GarageQual'].fillna(df1['GarageQual'].mode()[0])
df1['GarageCond'] = df1['GarageCond'].fillna(df1['GarageCond'].mode()[0])
df1['Fence'] = df1['Fence'].fillna(df1['Fence'].mode()[0])
```

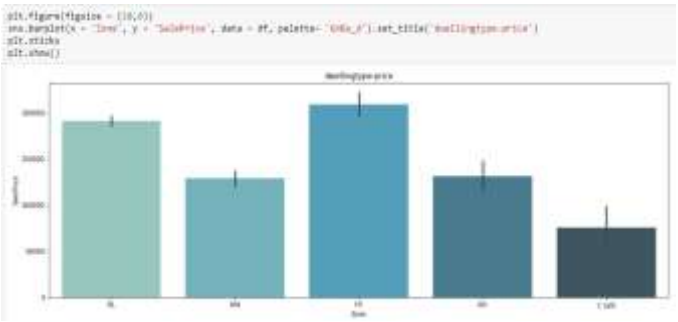
We will change data type of dwelling column as it have stable and desecrate value.so we will change it to object data type

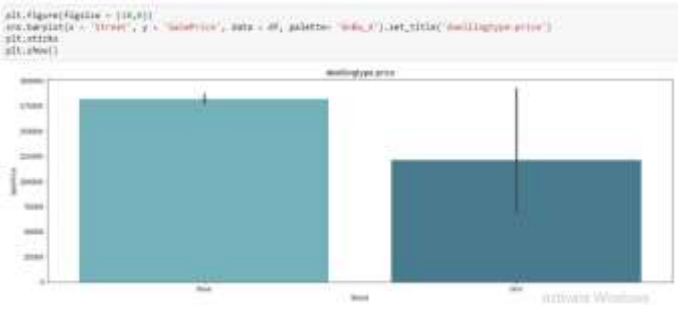
```
df['Dwelling'].apply(str)
df1['Dwelling'].apply(str)

#CONVERT DWELLING COLUMN TO OBJECT BECAUSE GIVEN VALUE IS STABLE AND DESCRETE
```

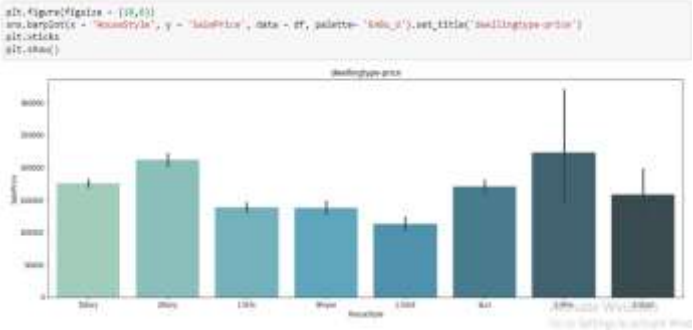
```
0       20
1      120
2       20
3       70
4       60
      ...
287     20
288     20
289     20
290     50
291    160
Name: Dwelling, Length: 292, dtype: object
```

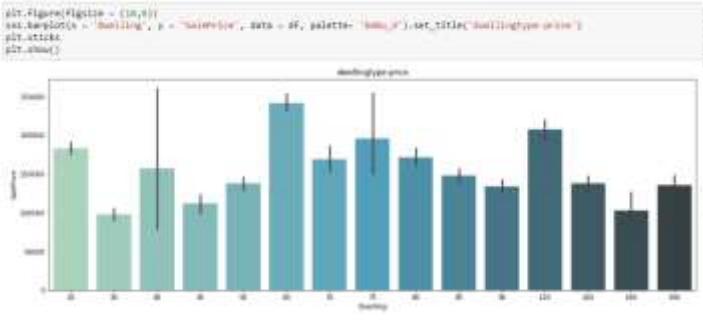
- Data Inputs- Logic- Output Relationships

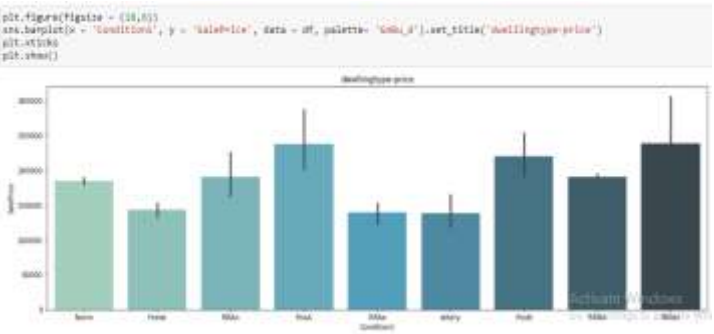
  First, we will check about data distribution of columns

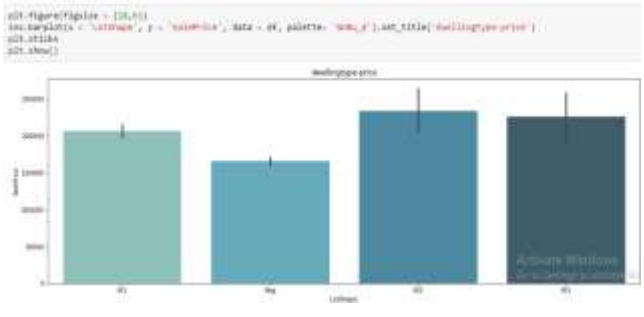


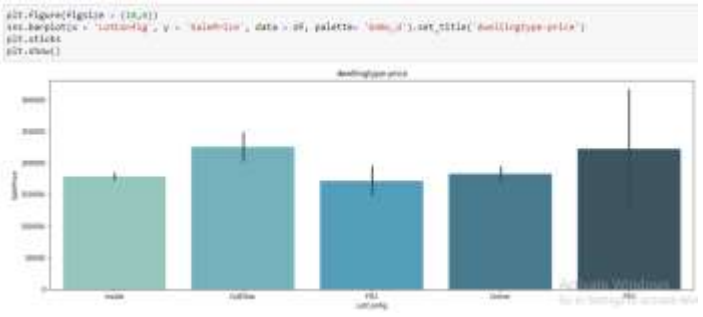


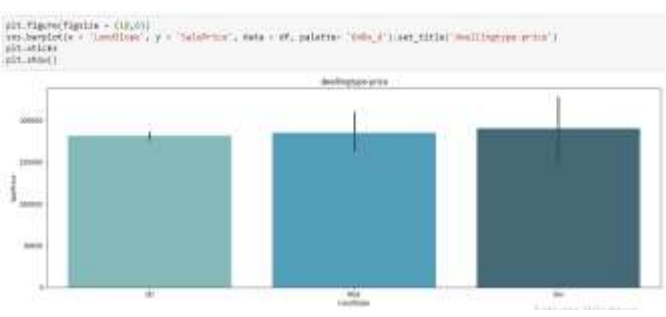


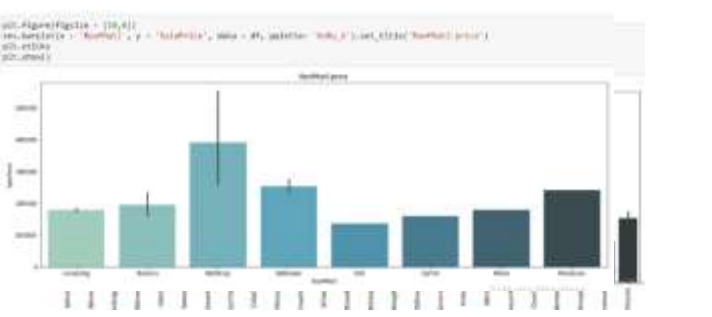


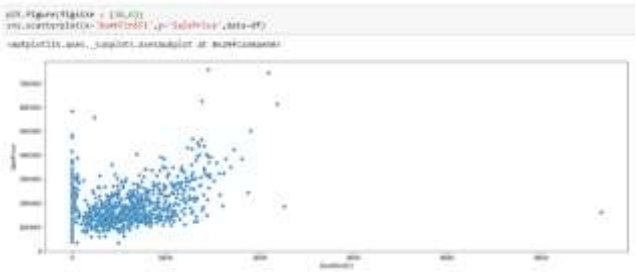


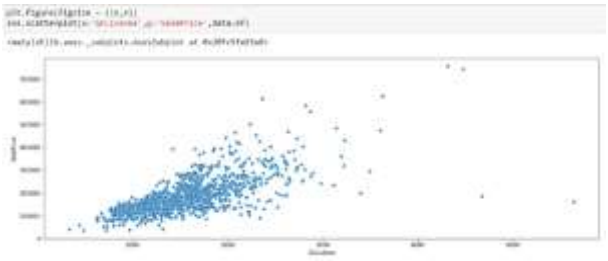


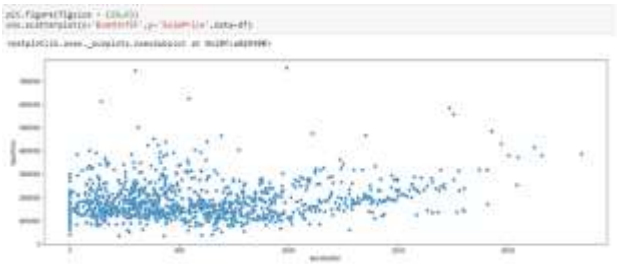


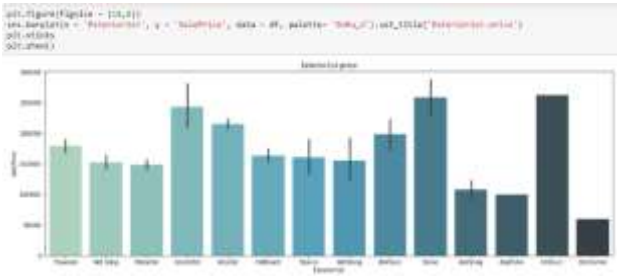


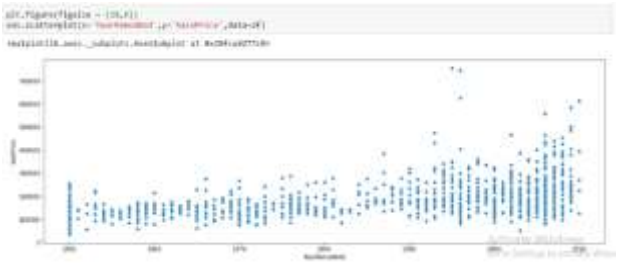


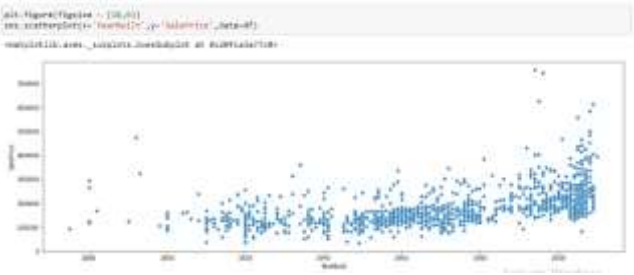


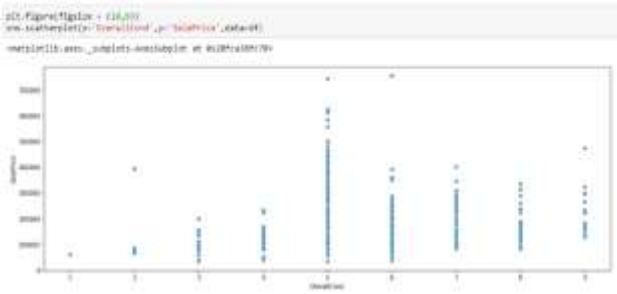


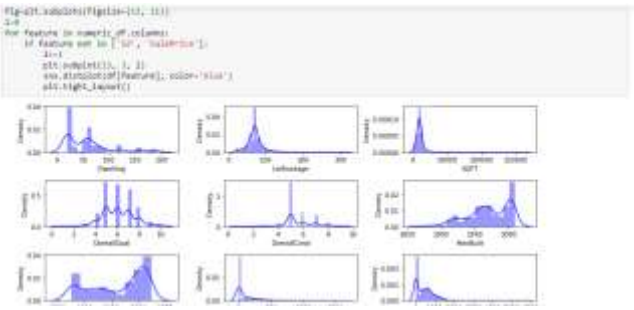


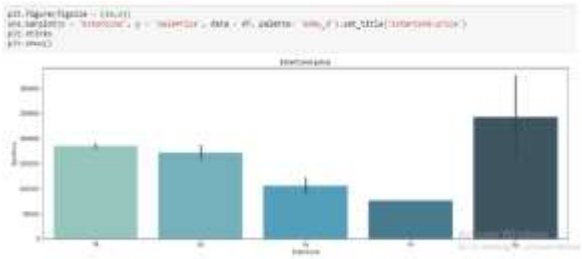




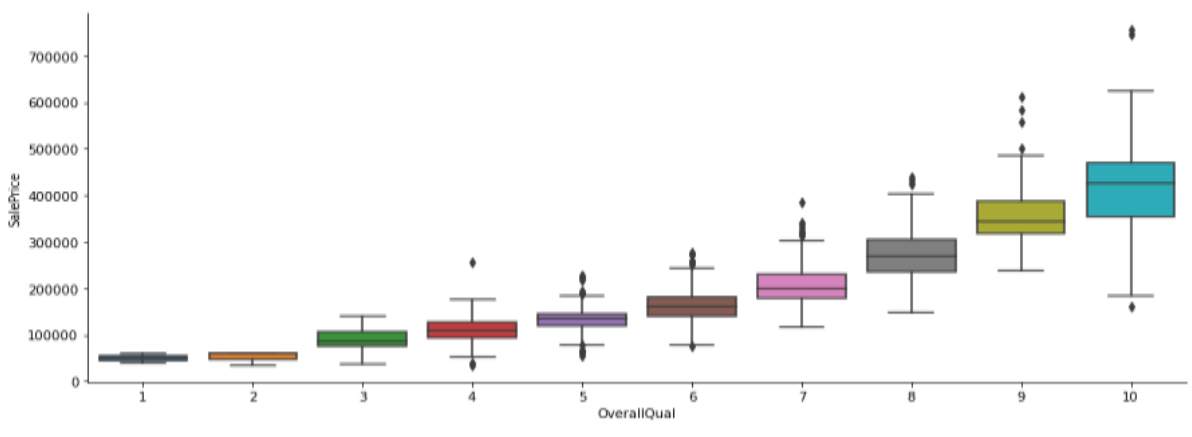




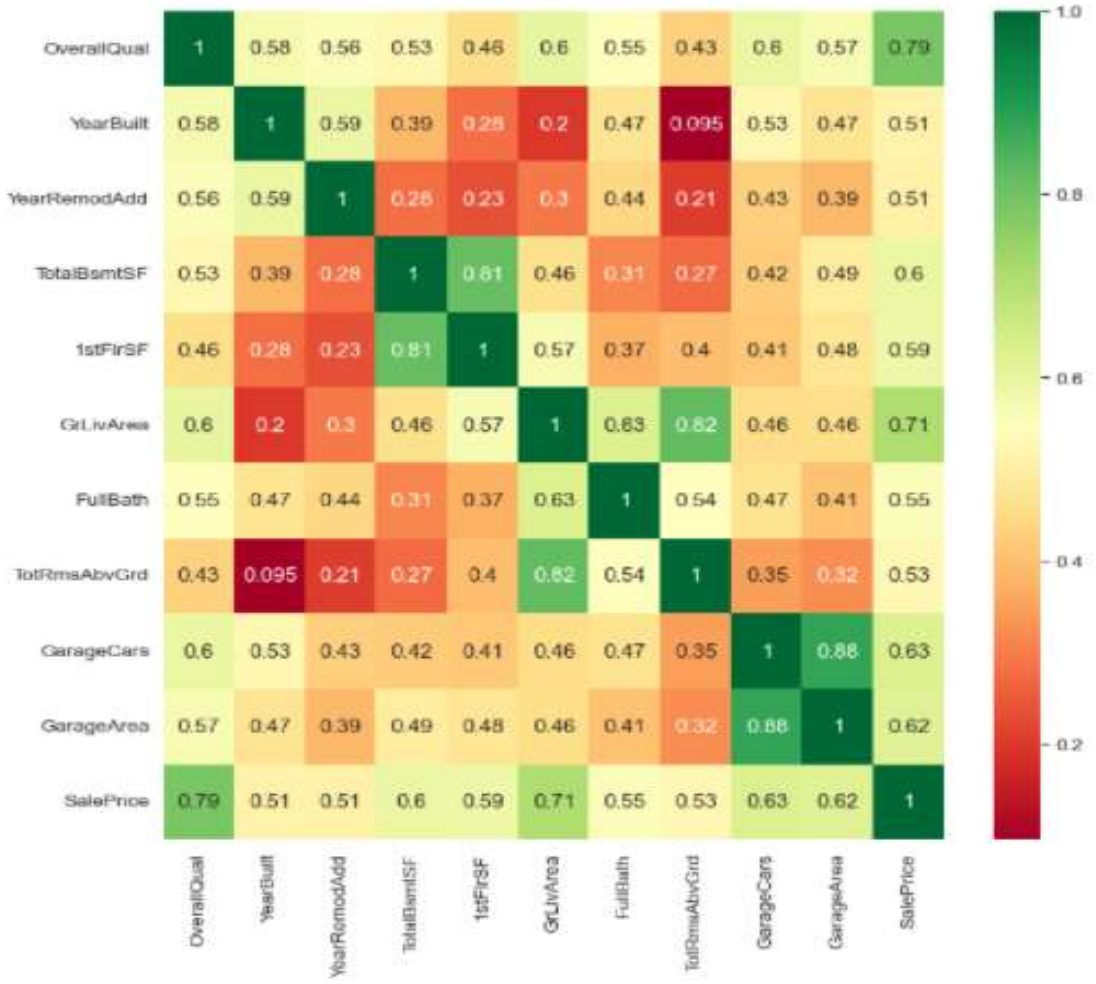




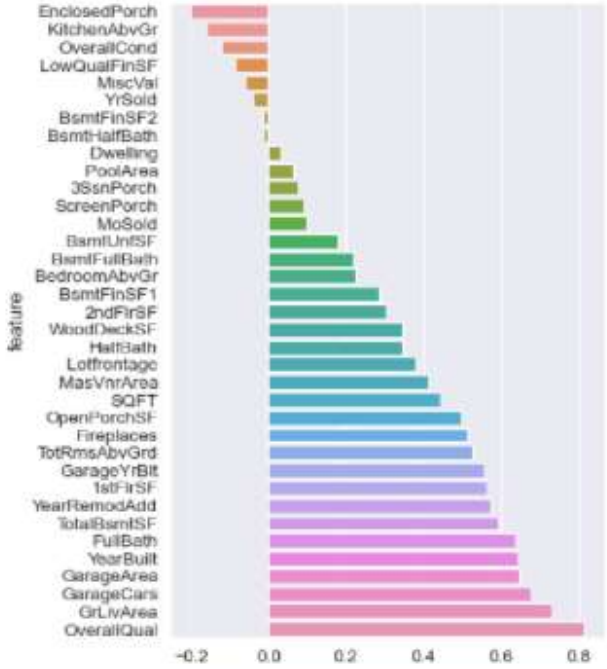
```
g = sns.factorplot(x="OverallQual",y="SalePrice",data=df,kind='box',aspect=2.5)
```



```
plt.figure(figsize=(12,12))
top_corr_features = corrmat.index[abs(corrmat["SalePrice"])>0.5]
g = sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



```
def spearman(frame, features):
    spr = pd.DataFrame()
    spr['feature'] = features
    spr['spearman'] = [frame[f].corr(frame['SalePrice'], 'spearman') for f in features]
    spr = spr.sort_values('spearman')
    plt.figure(figsize=(6, 0.25*len(features)))
    sns.barplot(data=spr, y='feature', x='spearman', orient='h')

features = corrmat[['SalePrice']].sort_values(['SalePrice'], ascending=False)
features = [f for f in features.index if f != 'SalePrice']

spearman(numeric_df, features)
```



EDA CONCLUSION:

- 60, 75 and 120 have high sales price while 30 and 180 have low sales price.
- Slightly slope to sqft to sales, where we can say that as SQFT goes high then sales price also increasing
- RL and FV zone have high house selling price, as demand is high in that zone while in C zone selling value is low.
- Utilities have only single variable so we can remove it because it does not matter with result.
- Landslope have also 3 variable but all 3 have same effect with number of sales.
- Overall quality increases leads to increase in sales price.
- Overall quality play an important role in sales price of house.
- Grlivarea have good co relation with selling price as it parallel increase to sales price.
- Garagecars and garage area have similar co-relation with Sales price. Both are parallel increase to sales price.
- Garage year built, 1st floor sf,year remod add, total basement sqft,full bath, year built, garage area, garage cars, grlivearea,overall quality are TOP 10 features which effect to sales price.
- Enclosed porch, kitchenabvgr, overall condition, miscval, yrsold are not much co-related featurs to predict price.

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

We consider that based on EDA, 'LandSlope','BsmtFinType2','Dwelling','BsmtHalfBath' are not effective towards target variable so we drop it and for modelling purpose we convert all object column to int column using label-encoder.

- Hardware and Software Requirements and Tools Used

We will use below libraries and software used:

FOR BASIC DATA STUDY:

- import numpy as np
- import pandas as pd
- pd.get_option("display.max_columns")

FOR DATA VISULISATION

- import matplotlib.pyplot as plt
- import seaborn as sns

DATA CLEANING AND PRE_PROCESSING

- from scipy.stats import boxcox
- from scipy.stats import zscore
- from sklearn.preprocessing import StandardScaler
- from statsmodels.stats.outliers_influence import variance_inflation_factor
- from sklearn.model_selection import train_test_split
- from sklearn.decomposition import PCA
- from imblearn.under_sampling import NearMiss
- from collections import Counter

MODEL BUILDING

- from sklearn.metrics import accuracy_score
- from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,f1_score
- import warnings
- warnings.filterwarnings('ignore')
- from sklearn.model_selection import GridSearchCV
- from sklearn.linear_model import LogisticRegression
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.naive_bayes import GaussianNB

- from sklearn.linear_model import SGDClassifier
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import GradientBoostingClassifier,AdaBoostClassifier

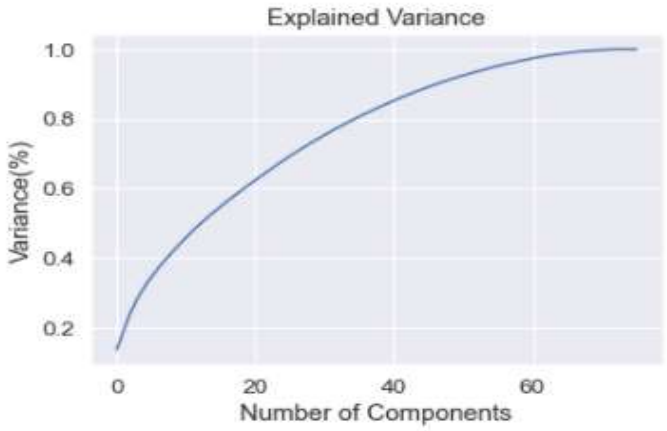
Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

Now our data is ready for modelling purpose but here we will give our best effort to get maximum accuracy. For that we will scaledown data so we can get low error ratio. Remove skewness of data.

```
from scipy.stats import boxcox

for col in df:
    if df[col].skew()>=2:
        df[col]=np.cbrt(df[col])
    if df[col].skew()<= -2:
        df[col]=np.power(df[col],2)

for col in df1:
    if df1[col].skew()>=2:
        df1[col]=np.cbrt(df1[col])
    if df1[col].skew()<= -2:
        df1[col]=np.power(df1[col],2)
```

```
from sklearn.decomposition import PCA
pca = PCA()
PrincipalComponents = pca.fit_transform(x_scaled)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance(%)')
plt.title('Explained Variance')
plt.show()
```



```
pca = PCA(n_components = 70)
new_data = pca.fit_transform(x_scaled)
pri_x = pd.DataFrame(new_data)
pri_x
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	-0.649603	-1.713697	-1.611688	0.762060	0.514987	-0.972133	0.643831	0.305039	0.233960	0.752348	-1.305242	0.019888	0.621714	-1.531984	1.3
1	3.013638	0.814244	5.281425	-0.841889	-0.570696	2.758825	-0.498111	3.606887	0.853249	3.390304	-1.168390	0.381866	0.417654	-0.574871	2.2
2	2.983473	0.157126	0.527313	2.121369	-0.612265	0.716738	0.936181	-0.931876	0.720048	-1.144732	0.147365	0.551871	0.699798	-1.175105	-0.1
3	2.309205	-0.046313	1.959024	-1.590217	0.187259	-3.227530	0.736213	-0.468975	-0.793484	0.310647	-0.069323	0.371532	-0.562921	-0.038337	0.4
4	1.779941	-1.014332	2.752616	0.255256	0.496101	-0.775040	0.503587	-0.604828	0.527377	1.622653	1.669897	-0.200477	0.665800	-0.167705	-0.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	-3.341992	-1.309904	0.584235	-0.159648	-0.829265	-1.694558	0.146617	-0.790957	0.562913	-0.502518	0.800975	-0.511711	0.056167	0.762435	0.0
1164	-2.070390	0.001863	-2.289459	-1.737601	-1.359463	1.236808	0.095949	0.302526	-0.609045	-0.893091	0.567801	-0.693093	-0.405451	-0.043989	-0.4
1165	-1.243911	-0.878468	-2.131409	4.435260	1.346516	-0.882698	-0.506780	-0.706284	0.456458	0.783848	-0.707215	-0.838851	0.241561	-1.178853	0.5
1166	-6.173320	3.422745	-0.914472	0.083695	2.883123	0.192557	-0.252899	-0.429667	1.754867	-1.771467	2.814372	2.649199	-0.239823	1.519111	1.1
1167	2.089513	-0.249172	-1.646707	1.078540	-1.569423	1.542069	0.853123	-0.346568	-0.674695	-0.845057	0.461274	0.265360	-0.108594	-0.627677	-0.1

1168 rows × 70 columns


```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((817, 70), (351, 70), (817,), (351,))
```

Finally we have data shape as per above where in training data we have 817 rows and 351 rows for testing.

- # Testing of Identified Approaches (Algorithms)

We will use below algorithm for training and testing of dataset.

- Decision Tree Regressor
- Support vector regression
- KNeighbors Regressor
- Linear regression
- RandomForest Regressor
- AdaBoost Regressor
- GradientBoosting Regressor

Logistic regression

```
ln=LinearRegression()
ln.fit(x_train,y_train)
pred=ln.predict(x_test)
print('Train Score',ln.score(x_train,y_train)*100)
print('Testing:-',ln.score(x_test,y_test)*100)
print('Error')
print('MAE',mean_absolute_error(y_test,pred))
print('MSE',mean_squared_error(y_test,pred))
print('RMSE',np.sqrt(mean_squared_error(y_test,pred)))
print('R2 Score',r2_score(y_test,pred)*100)
```

```
Train Score 84.63293682938206
Testing:- 76.33119890713564
Error
MAE 20930.877587621555
MSE 1536157902.1921918
RMSE 39193.85031088668
R2 Score 76.33119890713564
```

Conclusion for logistic regression:

- On training data, accuracy score is 84%
- On testing data, accuracy score is 76%
- R2 score is 76%.

Decision Tree Regressor

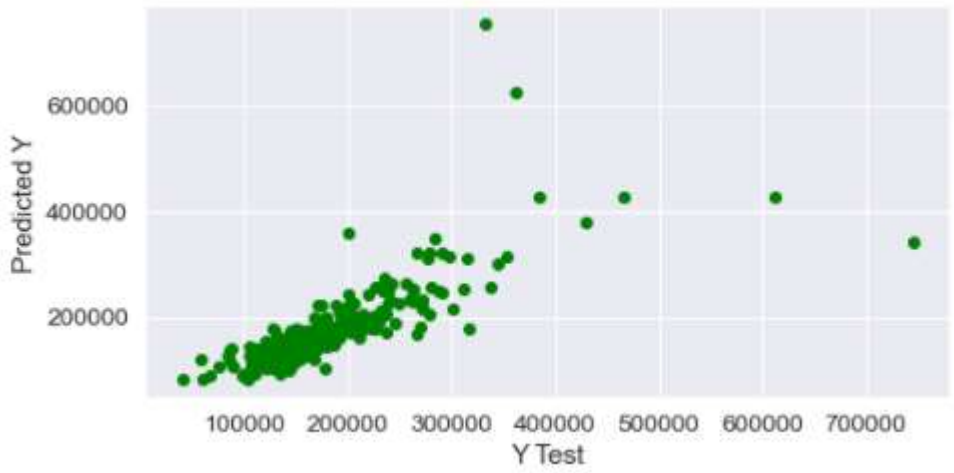
```
dtr.fit(x_train,y_train)
preddtr=dtr.predict(x_test)
print("DecisionTreeRegressor")
print('Training Score:-',dtr.score(x_train,y_train)*100)
print('Testing:-',dtr.score(x_test,y_test)*100)
print('MAE',mean_absolute_error(y_test,preddtr))
print('MSE',mean_squared_error(y_test,preddtr))
print('RMSE',np.sqrt(mean_squared_error(y_test,preddtr)))
print('R2 Score',r2_score(y_test,preddtr)*100)
```

```
DecisionTreeRegressor
Training Score:- 100.0
Testing:- 55.53368923192308
MAE 28097.58547008547
MSE 2885962596.9102564
RMSE 53721.15595284838
R2 Score 55.53368923192308
```

Conclusion for Decision Tree Classifier:

- On training set accuracy score is 100% but on testing, set it 55%, which is not good for dataset because model is not much success to predict testing data accurately.

- In Decision Tree Classifier, R2 score is 55%.



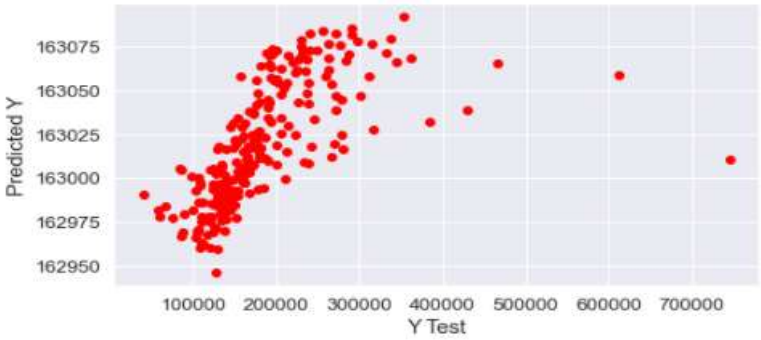
SVR:

```
fun(svr)
```

```
TRAINING:- -5.189372497862221
Testing:- -6.199006148228725
MAE 53094.33162081054
MSE 6892552007.999322
RMSE 83021.39488107461
R2 Score -6.199006148228725
```

Conclusion on SVR:

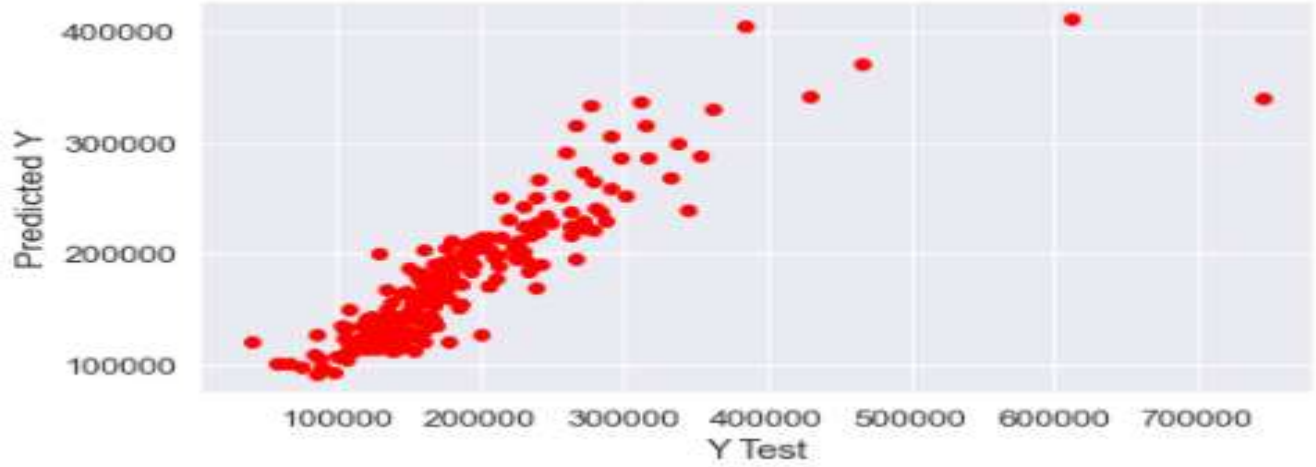
Here we can see that accuracy is in minus so this model is not perfect for this data and we will neglect it.



KNN:

```
fun(kn)
```

```
TRAINING:- 83.99008513604738
Testing:- 75.3340239054887
MAE 22112.8641025641
MSE 1600876780.543419
RMSE 40010.95825575062
R2 Score 75.3340239054887
```



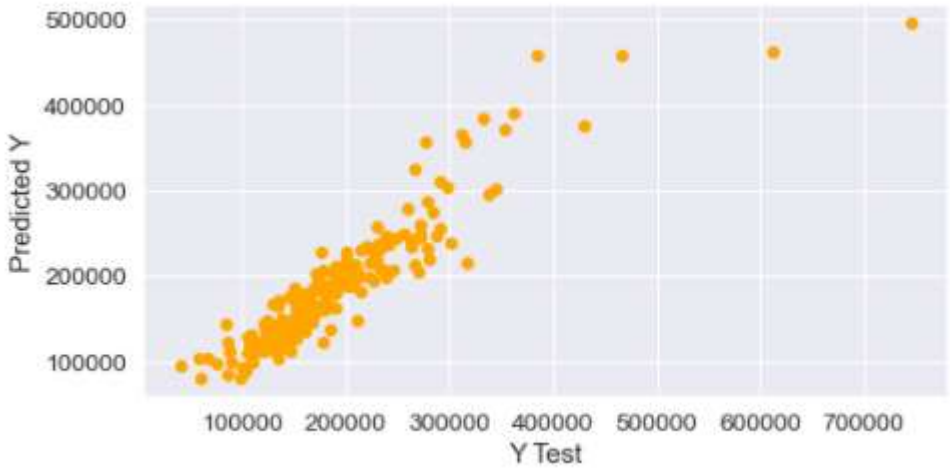
Conclusion on KNN:

Training score is 84% while testing 75% and R2 score is 75%. Its looking better for prediction compare to other and as per visualise we can see that data are predicted on one line.

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
rd=RandomForestRegressor()
fun(rd)
```

```
TRAINING:- 97.56989541193235
Testing:- 85.88893734653358
MAE 18546.54311965812
MSE 915839392.0504354
RMSE 30262.838466515917
R2 Score 85.88893734653358
```



Conclusion on Random Forest Regressor:

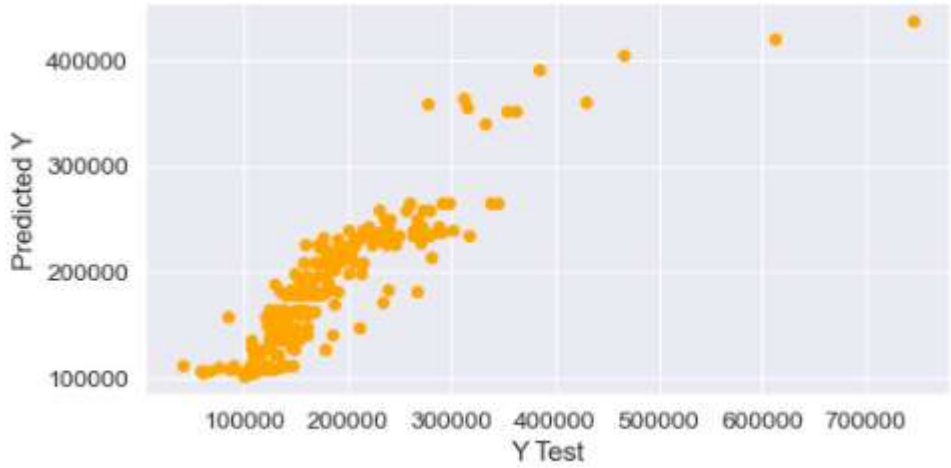
We can mark that 86% accuracy in testing dataset and 97% training accuracy. R2 score is 86%.

BOOSTING TECHNIQUE

Adaboost Classifier:

```
from sklearn.ensemble import AdaBoostRegressor
ad=AdaBoostRegressor()
fun(ad)
```

```
TRAINING:- 86.16629250645865
Testing:- 78.26306043561631
MAE 24343.514802896912
MSE 1410775786.6691818
RMSE 37560.29534853502
R2 Score 78.26306043561631
```



Conclusion on Adaboost Regressor:

We got accuracy of 97% in raining and in testing 86% accuracy.

R2 score is 86% and here we can mark that random forest and adaboost have good accuracy.

Gradient Boosting Classifier

```
]: fun(gd1)
   pred=gd1.predict(prix_test)
```

```
training Score 0.8924169672294382
Accuracy Score
 0.8914914122465838
Confusion Matrix
 [[ 1890  5525]
 [  780 49911]]
Classification Report              precision    recall  f1-score   support

           0       0.71      0.25      0.37      7415
           1       0.90      0.98      0.94     50691

    accuracy                           0.89     58106
   macro avg       0.80      0.62      0.66     58106
weighted avg       0.88      0.89      0.87     58106

f1_score 0.9405900477729513
```

Conclusion on Gradient boosting classifier

Both training and testing accuracy is 89% and 94% f1 score. Recall and precision values is high compares to others.

- Key Metrics for success in solving problem under consideration

Cross validation:

CROSS VALIDATION TECHNIQUE:

When we tested our model on testing data set, it take observation and give result but what happened if this model applied on different data? For this, we will use cross validation technique, in this technique, we will take average of accuracy tried for 10 times on random dataset from testing data.

```
score=cross_val_score(svr,x_scaled,y,cv=5)
print("score of cross validation score for svr", score.mean()*100)
score=cross_val_score(dtr,x_scaled,y,cv=5)
print("score of cross validation score for dtr",score.mean()*100)
score=cross_val_score(rd,x_scaled,y,cv=5)
print("score of cross validation score for rd",score.mean()*100)
score=cross_val_score(gd,x_scaled,y,cv=5)
print("score of cross validation score for Gd",score.mean()*100)
score=cross_val_score(ad,x_scaled,y,cv=4)
print("score of cross validation score for ad",score.mean()*100)
score=cross_val_score(randmf,x_scaled,y,cv=4)
print('score of cross validation score for rd hyper',score.mean()*100)
score=cross_val_score(GD1,x_scaled,y,cv=4)
print('score of cross validation score for gd hyper',score.mean()*100)
```

```
score of cross validation score for svr -6.185555083224847
score of cross validation score for dtr 71.59007860038618
score of cross validation score for rd 84.92000785265793
score of cross validation score for Gd 87.30208994371431
score of cross validation score for ad 78.90539685501982
score of cross validation score for rd hyper 83.50336490420435
score of cross validation score for gd hyper 82.86104647106875
```

We are getting higher accuracy in Gradient boost regression technique so we will proceed with this model.

Hyper parameter tuning:

Randomforest:

```
from sklearn.model_selection import RandomizedSearchCV
rf_random = RandomizedSearchCV(estimator = rd,param_distributions = random_grid,
               n_iter = 100, cv = 5, verbose=2, random_state=35, n_jobs = -1)
```

```
rf_random.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed:  3.0min finished
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [10, 20, 30, 40, 50, 60,
                                                      70, 80, 90, 100, 110,
                                                      120],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 3, 4],
                                        'min_samples_split': [2, 6, 10],
                                        'n_estimators': [5, 20, 50, 100]},
                   random_state=35, verbose=2)
```

```
fun(randmf)
```

```
TRAINING:- 97.52378688354099
Testing:- 86.44677703588098
MAE 17995.755641025644
MSE 879634353.8828788
RMSE 29658.630344014182
R2 Score 86.44677703588098
```

In hyper parameter tuning we are getting 86.44% accuracy in testing dataset.

Gradient boost regressor:

```
GBR.best_params_
```

```
{'random_state': 1,
 'n_estimators': 600,
 'min_samples_split': 50,
 'min_samples_leaf': 1,
 'max_depth': 2,
 'learning_rate': 0.1}
```

```
GD1 = GradientBoostingRegressor(n_estimators = 600, min_samples_split = 50,
                                min_samples_leaf= 1,max_depth = 2, learning_rate = 0.1)
GD1.fit(x_train, y_train)
```

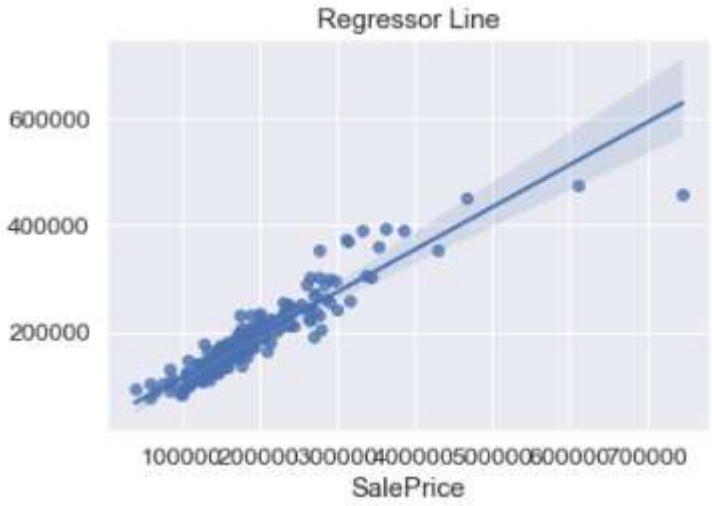
```
GradientBoostingRegressor(max_depth=2, min_samples_split=50, n_estimators=600)
```

```
fun(GD1)
```

```
TRAINING:- 99.07430952740506
Testing:- 86.26304980790927
MAE 18013.51948612169
MSE 891558660.1453412
RMSE 29858.979556330138
```

In gradient boost hyper parameter tuning, we are getting 86.26% accuracy in testing dataset.

PREDICATED RESULT:



```
      Actual        Predict
400   163000  168337.385273
231    89500   99444.191648
530   316600  258994.270124
214   171000  172862.109257
730   239799  235741.661157
..       ...            ...
295   150000  174328.317511
177   102776   97598.932143
606   165500  163666.058701
695   190000  173359.106494
867   160000  135416.428474

[234 rows x 2 columns]

Text(0.5, 1.0, 'Regressor Line')
```

- Interpretation of the Results

First we cleaned that data, normalised it then we remove outliers, convert all objective columns to numerical columns and then scale down data for better accuracy. We done some process like PCA, balancing data, hyper parameter tuning for increase accuracy for model. By using some records and observation finally, we take random forest classifier as best-fit model. We applied model on testing dataset and check actual result.

CONCLUSION

We are getting 87% accuracy in generated model, which is the best model. It will help much to clients to predict best price of house with concluding all parameters. Top important parameters are **OVERALL QUALITY, LIVING AREA, GARAGE AREA,YEAR BUILT.** We need to re-modify this model over the time with data to cover the trend of market with time.