HOUSING: PRICE PREDICTION

Submitted by:

ABHIJIT SARKAR

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for giving me the opportunity to work on this project. I am very grateful to the Data Trained team for providing me with the Trainings and knowledge which helped me a lot to complete this project. I took help from Mr. Mohd Kashif where I faced the problem. Moreover, I took the help of internet for finding the meaning of different terminologies in dataset, I used Kaggle, google, medium and used mostly sklearn library to accomplish the task.

NEXT

INTRODUCTION

➢ Business Problem Framing

Houses are one of the necessary needs 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.

➢ Conceptual Background of the Domain Problem

The real estate market is one of the most competitive in terms of pricing and same tends to vary significantly based on numerous factors; forecasting property price is an important module in decision making for both the buyers and investors in supporting budget allocation, finding property finding stratagems and determining suitable policy. 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 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?

Review of Literature

Demand The factors that affect the land price have to be studied and their impact on price has also to be modelled. An analysis of the past data is to be considered. It is inferred that establishing a simple linear mathematical relationship for these time-series data is found not viable for forecasting. Hence it became imperative to establish a non-linear model which can well fit the data characteristic to analyse and forecast future trends. As the real estate is fast developing sector, the analysis and forecast of land prices using mathematical modelling and other scientific techniques is an immediate urgent need for decision making by all those concerned. The increase in population as well as the industrial activity is attributed to various factors, the most prominent being the recent spurt in the knowledge sector viz. Information Technology (IT) and Information technology enabled services. Demand for land started of showing an upward trend and housing and the real estate activity started booming. The need for predicting the trend in land prices was felt by all in the industry viz. The Government, the regulating bodies, lending institutions, the developers and the investors. Therefore, in this project report, we present various important features to use while predicting housing prices with good accuracy. We can use regression models, using various features to have lower Residual Sum of Squares error. While using features in a regression model some feature engineering is required for better prediction. The primary aim of this report is to use these Machine Learning Techniques and curate them into ML models which can then serve the users. The main objective of a Buyer is to search for their dream house which has all the amenities they need. Furthermore, they look for these houses/Real estates with a price in mind and there is no guarantee that they will get the product for a deserving price and not overpriced. Similarly, A seller looks for a certain number that they can put on the estate as a price tag and this cannot be just a wild guess, lots of research needs to be put to conclude a valuation of a house.

➢ Motivation for the Problem Undertaken

I have 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. The relationship between house prices and the economy is an important motivating factor for predicting house prices.

ANALYTICAL PROBLEM FRAMING

➢ Mathematical/ Analytical Modelling of the Problem

I found more than 80% null values and more than 90% zero values so I decided to drop those columns. If I keep those columns as it is, it will create high skewness in the model followed by accuracy prediction score would go beyond the label. If I impute those 80 to 90 percentage data then that may cause to a very odd result. This particular problem has two datasets one is train dataset and the other is test dataset. I have built model using train dataset and predicted SalePrice for test dataset. By looking into the Target column, I came to know that the entries of SalePrice column were continuous Variable and this is a case of Regression problem so I have to use all regression algorithms while building the model. Also, I Found some unnecessary entries in some of the columns like in some columns, while checking the null values in the datasets I found many columns with nan values and I replaced those nan values with suitable entries like mean for numerical columns and mode for categorical columns. To get better insight on the features I have used plotting like distribution plot, bar plot, reg plot and cat plot, strip plot, count plot as well. With these plotting I was able to understand the relation between the features in better manner. Also, I found outliers and skewness in the dataset so I removed outliers using Zscore and I removed skewness using yeo-Johnson method. I even used Standard Scaler to bring the data under one scale. PCA didn't work well so I removed that, I have used all the regression models while building model then turned the best model and saved the best model. At last I have predicted the sale price for test dataset using the saved model of train dataset.

➢ Data Sources and their formats:

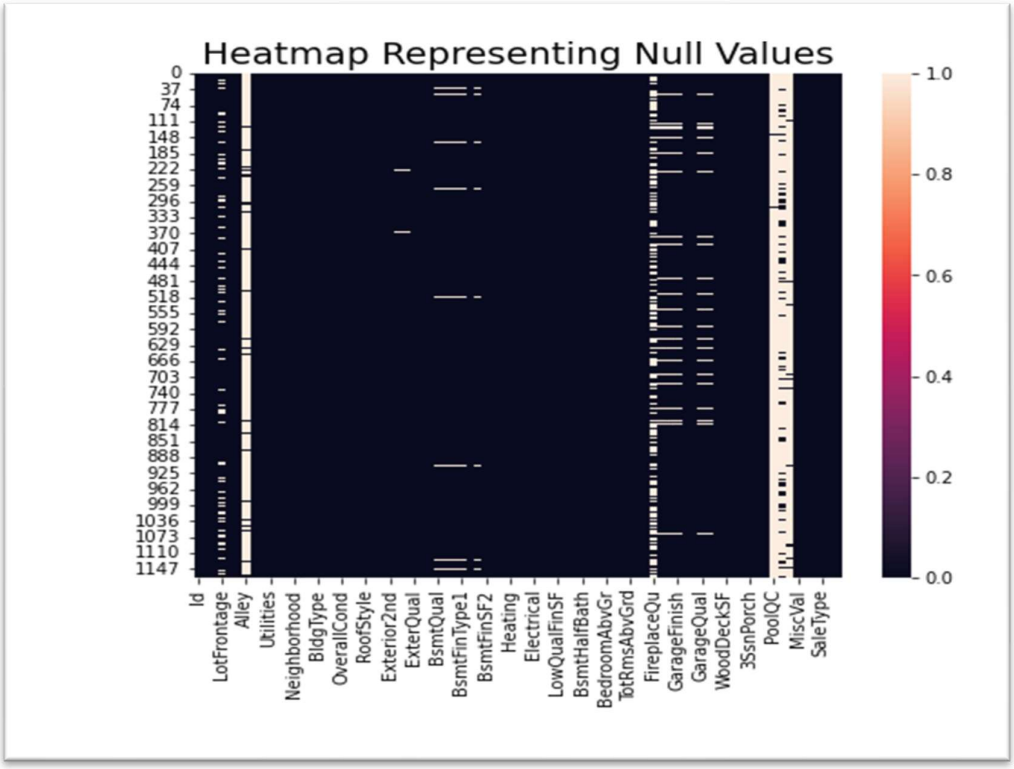
Data contains 1460 entries each having 81 variables. The data was collected for my internship company Flip Robo technologies in csv (comma separated values) format. Also, I was having two datasets one is train and other is test data set. Model using train dataset and predicted SalePrice for test dataset.

There are three Types of data herein: <u>float64 having 3 columns, int64 having 35 columns, object or category having 43 columns.</u>

```
 75  MiscVal        1168 non-null   int64
 76  MoSold         1168 non-null   int64
 77  YrSold         1168 non-null   int64
 78  SaleType       1168 non-null   object
 79  SaleCondition  1168 non-null   object
 80  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 739.2+ KB
```

Maximum data Missing:

- **Alley 1091**
- **PoolQC 1161**
- **MiscFeature 1124**
- **Fence 931**
- **FireplaceQu 551**



WHITE DASH REPRESENT MISSING COLUMNS

➢ Data Pre-processing Done

These Steps:

1. loading Data Set, Locating Null Values
2. Finding the unique values in categorical and numerical columns.
3. Finding Data Missing percentage
4. Finding nan values and replacing nan values.
5. Finding duplicated values in columns.
6. Use encoding If required.
7. As all found to be intact. Feature Extraction with I don't feel important here.

➢ Data Inputs/ Logic-Output Relationships

X variables plays a very import role in machine learning for the Prediction of Target variable. Here 'Sale Price' is the target variable on which the predictions are being made. Here sales price is the target variable which is dependent on rest all X columns variable.

I used the following to determine the relationship between variable:

- I have used Catplot for each pair of categorical features that shows the relation with the sale price. Used (Box plot and Distplot to determine the relations) And also, for continuous numerical variables I have used lmplot, scatterplot to show the relationship between a continuous numerical variable and target variable.
- Used univariate, Bivariate and Multivariate Graph to check for relations.

I found that there is a Linear relationship between continuous numerical variable and Sale Price.

➢ State the set of assumption (if any) related to the problem under consideration.

1. **Age of the Home can play a very good role for prediction.**
2. **Location of the House.**
3. **Infrastructure of the area.**
4. **Area of the House.**
5. **Material used to Build the House.**
6. **Crime Rate.**
7. **Communication from House.**

➢ Hardware and Software Requirements and Tools Used

➢ HARDWARE USED:

➢ SOFTWARE USED:

Anaconda (Jupyter NoteBook)

Model/s Development and Evaluation

➢ IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

- I have used the simple imputation method to replace null values.
- To check outliers, I used boxplot. To remove outliers, I have used Zscore.
- To check skewness, I used distplot. I remove skewness I have used the yeo-johnson method.
- to encode the categorical columns, I have to used label encoder.
- Use of Pearson's correlation coefficient to check the correlation between dependent and independent features.
- Also, I have used standardization.
- Then followed by model building with all regression algorithms.

➢ TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

- Linear Regression
- Decision Tree Regressor.
- KNeighbors Regressor
- Support Vector Regressor
- SGD Regressor
- Extra Trees Regressor
- Random Forest Regressor
- Ada Boost Regressor
- Gradient Boosting Regressor
- MLP Regressor
- Bagging Regressor

➢ RUN AND EVALUATE SELECTED MODELS

Linear Regression ¶

In [170]:
```
#Train Test
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=39,test_size=0.3)

#training
lr.fit(x_train,y_train)
lr_score=lr.score(x_train,y_train)

#predict
pred_train=lr.predict(x_train)
pred_test=lr.predict(x_test)

#result
print("Training Accuracy r2_score ",r2_score(y_train,pred_train)*100,"Testing Accuracy R2_score ",r2_score(y_test,pred_test)*100
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_t
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pre
```

```
Training Accuracy r2_score  80.05947852523427 Testing Accuracy R2_score  80.46031281208037
Training Mean_squared_Error  1312036753.0209816 Testing Mean_squared_error  1069532970.1255641
Training Absolute_Error  22503.409961619203 Testing Absolute Error 21766.23416119535
```

A

Model- Decision Tree Regressor

In [178]:
```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()

#training Dtc model
dtr.fit(x_train,y_train)
dtr_score=dtr.score(x_train,y_train)

#predict dtc
pred_train=dtr.predict(x_train)
pred_test=dtr.predict(x_test)

#result dtc
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  1.0 Testing Accuracy R2_score  0.6928382304577889
Training Mean_squared_Error  0.0 Testing Mean_squared_error  1681294262.9430199
Training Absolute_Error  0.0 Testing Absolute Error 27434.327635327634
```

KNeighbors Regressor

In [182]:
```
from sklearn.neighbors import KNeighborsRegressor
knr=KNeighborsRegressor()

#training knr model
knr.fit(x_train,y_train)
knr_score=knr.score(x_train,y_train)

#predict knr
pred_train=knr.predict(x_train)
pred_test=knr.predict(x_test)

#result knr
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.8179653611569416 Testing Accuracy R2_score  0.8149768922889767
Training Mean_squared_Error  1197742680.8382864 Testing Mean_squared_error  1012750675.2225641
Training Absolute_Error  19200.77747858017 Testing Absolute Error 20306.13333333333
```

Model3: Support Vector Machine(Regressor)

In [186]:
```
from sklearn.svm import SVR
svr=SVR()

#training SVR
svr.fit(x_train,y_train)
svr_score=svr.score(x_train,y_train)

#predict SVR
pred_train = svr.predict(x_train)
pred_test  = svr.predict(x_test)

#result SVR
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  -0.05997081758574785 Testing Accuracy R2_score  -0.11207033931031973
Training Mean_squared_Error  6974344535.38301 Testing Mean_squared_error  6087077451.917714
Training Absolute_Error  56168.75574753584 Testing Absolute Error 53360.34687775143
```

Model 4: SGD Regressor

In [190]:
```
from sklearn.linear_model import SGDRegressor
sgd=SGDRegressor()

#training SGD
sgd.fit(x_train,y_train)
sgd_score=sgd.score(x_train,y_train)

#predict
pred_train=sgd.predict(x_train)
pred_test=sgd.predict(x_test)

#result
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.7912771857354328 Testing Accuracy R2_score  0.8077433638091669
Training Mean_squared_Error  1373344242.0532358 Testing Mean_squared_error  1052344437.0115553
Training Absolute_Error  23842.931440963188 Testing Absolute Error 22965.797356126106
```

Ensemble Techniques (Bagging)

Extra Trees Regressor

In [195]:
```
from sklearn.ensemble import ExtraTreesRegressor
etr=ExtraTreesRegressor()

#training score
etr.fit(x_train,y_train)
etr_score=etr.score(x_train,y_train)

#predict
pred_train=etr.predict(x_train)
pred_test=etr.predict(x_test)

#result
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  1.0 Testing Accuracy R2_score  0.8883907318199324
Training Mean_squared_Error  0.0 Testing Mean_squared_error  610909432.3882954
Training Absolute_Error  0.0 Testing Absolute Error 16106.224586894587
```

Random Forest Regressor

[199]:
```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()

#training Random Forest
rfr.fit(x_train,y_train)
rfr_score= rfr.score(x_train,y_train)

#predict Model
pred_train = rfr.predict(x_train)
pred_test = rfr.predict(x_test)

#result
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.9757506397673887 Testing Accuracy R2_score  0.8844956981408182
Training Mean_squared_Error  159554763.413253 Testing Mean_squared_error  632229461.2070664
Training Absolute_Error  7125.065324357405 Testing Absolute Error 16607.458547008548
```

Ada Boost

[204]:
```
from sklearn.ensemble import AdaBoostRegressor
ada=AdaBoostRegressor()

#training_ada_Boost
ada.fit(x_train,y_train)
ada_score= ada.score(x_train,y_train)

#predict_ada_Boost
pred_train = ada.predict(x_train)
pred_test = ada.predict(x_test)

#result
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.8851108757631092 Testing Accuracy R2_score  0.8178854688058432
Training Mean_squared_Error  755941883.0242264 Testing Mean_squared_error  996830162.0075676
Training Absolute_Error  21510.982936854845 Testing Absolute Error 21850.992580548726
```

Gradient Boosting

[208]:
```
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor()

#training_Gradient_Boost
gbr.fit(x_train,y_train)
gbr_score= gbr.score(x_train,y_train)

#predict_gbr
pred_train = gbr.predict(x_train)
pred_test = gbr.predict(x_test)

#result_gbr
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.9727803450159803 Testing Accuracy R2_score  0.8995326711612333
Training Mean_squared_Error  179098564.6427487 Testing Mean_squared_error  549922419.8427321
Training Absolute_Error  9973.128999099741 Testing Absolute Error 15991.996240673408
```

MLP Regressor

[213]:
```
from sklearn.neural_network import MLPRegressor
mlp= MLPRegressor()

#training
#training_mlp
mlp.fit(x_train,y_train)
mlp_score= mlp.score(x_train,y_train)

#predict_mlp
pred_train = mlp.predict(x_train)
pred_test = mlp.predict(x_test)

#result_mlp
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  -4.819801200970982 Testing Accuracy R2_score  -6.122675214533063
Training Mean_squared_Error  38292845453.47771 Testing Mean_squared_error  38986990447.570015
Training Absolute_Error  178481.52621037027 Testing Absolute Error 183420.02493562092
```

Bagging Regressor

[217]:
```
from sklearn.ensemble import BaggingRegressor
br= BaggingRegressor()

#training
#training_mlp
br.fit(x_train,y_train)
br_score= br.score(x_train,y_train)

#predict_mlp
pred_train = br.predict(x_train)
pred_test = br.predict(x_test)

#result_mlp
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_test))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.9626264246805812 Testing Accuracy R2_score  0.8725400897226521
Training Mean_squared_Error  245908836.80212978 Testing Mean_squared_error  697670208.841168
Training Absolute_Error  8578.072949816402 Testing Absolute Error 17791.575783475782
```

BY Using Hyper Parameter.

Model- 1: Random Forest Regressor With Grid Search CV

[231]:
```
from sklearn.model_selection import GridSearchCV

parameters={'n_estimators':[100,104,],
            'criterion':['mse', 'mae'],
            'min_samples_leaf':[1,2,3],
             'random_state':[39],
            'max_features':['auto','sqrt','log2']}

gsv=GridSearchCV(rfr,parameters,cv=3)
gsv.fit(x_train,y_train)

print("Best Parameter ",gsv.best_params_,"Best Score ",gsv.best_score_)
```

```
Best Parameter  {'criterion': 'mae', 'max_features': 'log2', 'min_samples_leaf': 1, 'n_estimators': 104} Best Score  0.80999546
64075724
```

[288]:
```
rfr=RandomForestRegressor(n_estimators=104,criterion='mae',min_samples_leaf=1,max_features='log2',min_weight_fraction_leaf=0.0001
                          min_impurity_decrease=0.00000001,n_jobs=1,ccp_alpha=0.00000001)

#training Random Forest
rfr.fit(x_train,y_train)
rfr_score= rfr.score(x_train,y_train)

#predict Model
pred_train = rfr.predict(x_train)
pred_result1 = rfr.predict(x_test)

#result
print("Training Accuracy r2_score ",r2_score(y_train,pred_train),"Testing Accuracy R2_score ",r2_score(y_test,pred_result1))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train),"Testing Mean_squared_error ",mean_squared_error(y_te
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train),"Testing Absolute Error",mean_absolute_error(y_test,pred
```

```
Training Accuracy r2_score  0.9781933947266256 Testing Accuracy R2_score  0.8943292796773769
Training Mean_squared_Error  143482042.9019122 Testing Mean_squared_error  578403933.8758519
Training Absolute_Error  6944.253295358253 Testing Absolute Error 15174.957538899846
```

Model 2: Gradient Boosting Regressor with Grid Search CV

[293]:
```
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor()
parameters={ 'loss':['squared_error', 'absolute_error', 'huber', 'quantile'],
             'learning_rate':[0.1,0.2,0.3],
             'n_estimators':[100,102,104],
             'ccp_alpha':[0.00000001],
            'criterion':['friedman_mse', 'squared_error', 'mse'],
            'max_features':['auto', 'sqrt', 'log2']}

gsv=GridSearchCV(gbr,parameters,cv=3)
gsv.fit(x_train,y_train)

print("Best Parameter ",gsv.best_params_,"Best Score ",gsv.best_score_)
```

```
Best Parameter  {'ccp_alpha': 1e-08, 'criterion': 'squared_error', 'learning_rate': 0.1, 'loss': 'squared_error', 'max_feature
s': 'log2', 'n_estimators': 102} Best Score  0.8302688263536596
```

[360]:
```
gbr=GradientBoostingRegressor(ccp_alpha=1e-08, criterion= 'squared_error', learning_rate= 0.1, loss= 'squared_error', max_feature
                              n_estimators= 102,random_state=39)

#training_Gradient_Boost
gbr.fit(x_train,y_train)
gbr_score= gbr.score(x_train,y_train)

#predict_gbr
pred_train2 = gbr.predict(x_train)
pred_result2 = gbr.predict(x_test)

#result_gbr
print("Training Accuracy r2_score ",r2_score(y_train,pred_train2),"Testing Accuracy R2_score ",r2_score(y_test,pred_result2))
print("Training Mean_squared_Error ",mean_squared_error(y_train,pred_train2),"Testing Mean_squared_error ",mean_squared_error(y_t
print("Training Absolute_Error ",mean_absolute_error(y_train,pred_train2),"Testing Absolute Error",mean_absolute_error(y_test,pre
```

```
Training Accuracy r2_score  0.9645647221083257 Testing Accuracy R2_score  0.8961445983776037
Training Mean_squared_Error  233155321.4705225 Testing Mean_squared_error  568467525.055661
Training Absolute_Error  11298.597099804498 Testing Absolute Error 15883.094238070278
```

Saving Model:

Using Pickle to Save the Model

Saving Model

Selected this model as its:

- R2 Score is better for the model.
- CV Score better for these model they are nearer to training score.
- "MSE" and "MAE" are less on compare to other models.(They are close to the mean)

In [372]:
```
import pickle
filename="US_Housepred.pkl"
pickle.dump(gbr, open(filename, 'wb'))
```

Loading Back Model

In [373]:
```
import pickle
model = pickle.load(open('US_Housepred.pkl', 'rb'))
result=model.score(x_test,y_test)
print(result*100)
```

```
89.61445983776038
```

In [374]:
```
new_df1=pd.DataFrame([model.predict(x_test)[:],pred_result2[:]],index=["Predicted","Original"])
new_df1
```

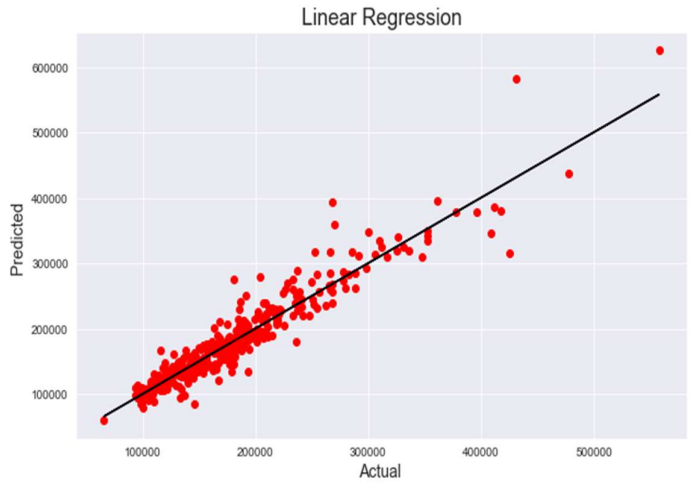
Out[374]:

	0	1	2	3	4	5	6	7	8
Predicted	178734.614991	117861.121183	136739.776622	134517.493599	267682.469578	194669.675165	98886.876706	186346.266607	165968.455522	282311.369(
Original	178734.614991	117861.121183	136739.776622	134517.493599	267682.469578	194669.675165	98886.876706	186346.266607	165968.455522	282311.369(

RESULT:

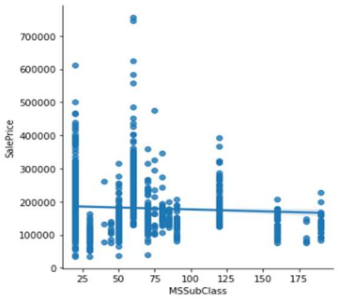
```
Cv_score of Model is  0.8778677003638116  The Training Score is 0.9645647221083257 Testing Score Stands  0.8961445983776037
```

```
[363]: plt.figure(figsize=(10,6))
       plt.scatter(x=pred_result2,y=y_test,color='red')
       plt.plot(pred_result2,pred_result2,color='black')
       plt.xlabel('Actual ',fontsize=14)
       plt.ylabel('Predicted ',fontsize=14)
       plt.title("Linear Regression",fontsize=18)
       plt.show()
```



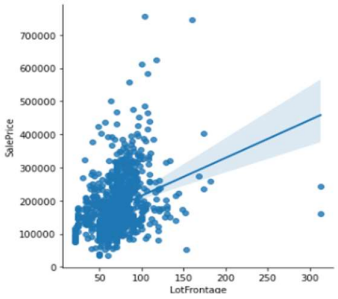
- **Key Metrics for success in solving problem under consideration.**
  1. R2_Score:
  2. Absolute_Error
  3. Mean_Squared_Error

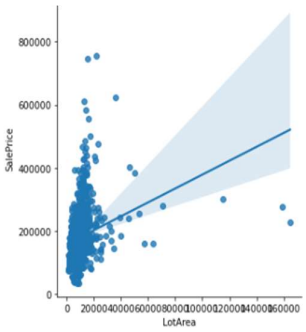
➢ Visualizations



MSSubClass shows little negative relation with SalePrice

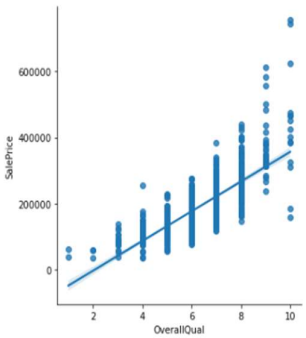
```
1]: sns.lmplot(x='LotFrontage',y='SalePrice',data=df)
1]: <seaborn.axisgrid.FacetGrid at 0x22c8f432940>
```

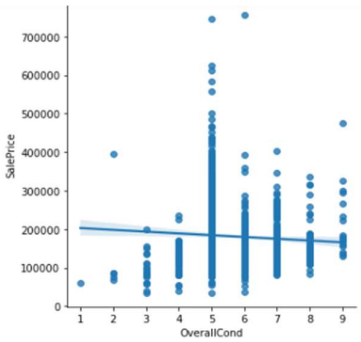




LotArea shows very positive relation with Target Variable

```
n [33]: sns.lmplot(x='OverallQual',y='SalePrice',data=df)
ut[33]: <seaborn.axisgrid.FacetGrid at 0x22c9048b580>
```

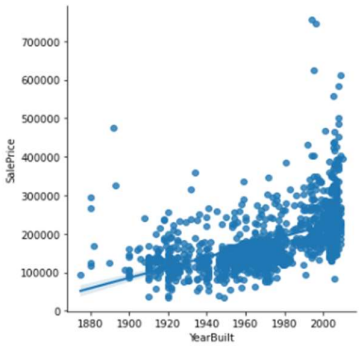


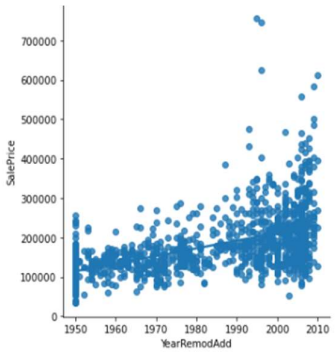


OverallCond is very negative towards SalePrice

In [35]:
```
sns.lmplot(x='YearBuilt',y='SalePrice',data=df)
```

Out[35]:
```
<seaborn.axisgrid.FacetGrid at 0x22c8c3cd280>
```

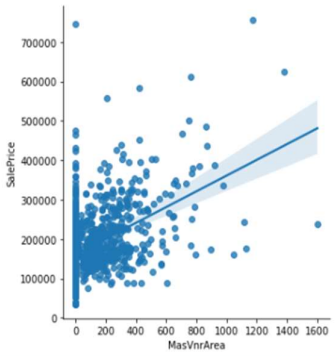


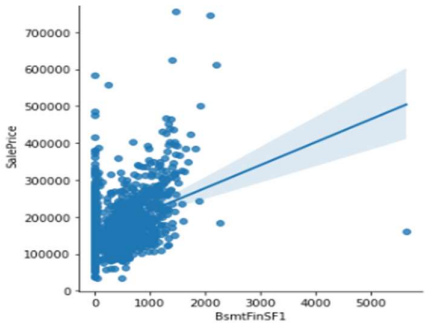


YearRemodAdd is positive towards SalesPrice

In [37]:
```
sns.lmplot(x='MasVnrArea',y='SalePrice',data=df)
```

Out[37]:
```
<seaborn.axisgrid.FacetGrid at 0x22c8c75faf0>
```

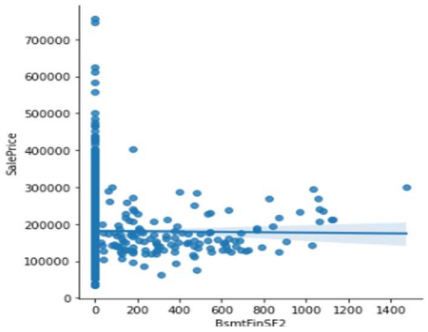


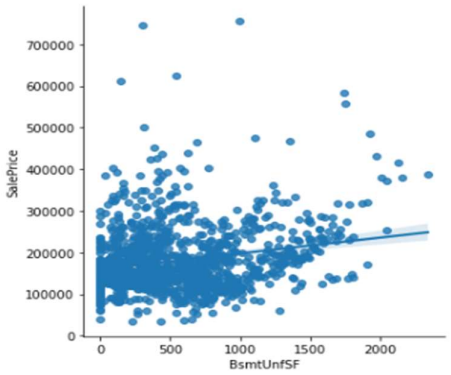


BsmtFinS1 is very positive towards salesPrice

```
In [39]: sns.lmplot(x='BsmtFinSF2',y='SalePrice',data=df)
```

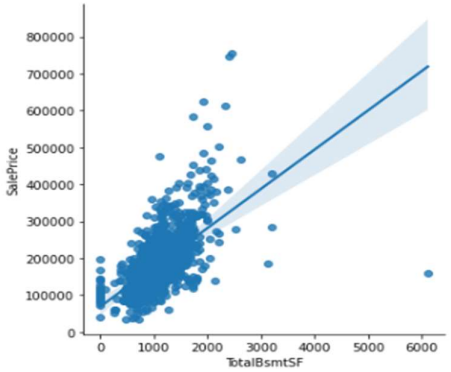
```
Out[39]: <seaborn.axisgrid.FacetGrid at 0x22c8a29dd60>
```

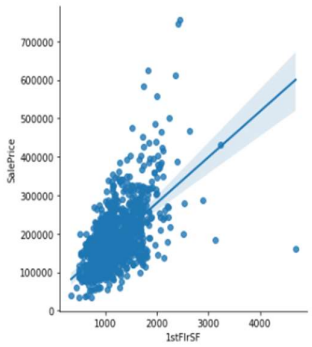





```
In [41]: sns.lmplot(x='TotalBsmtSF',y='SalePrice',data=df)
```

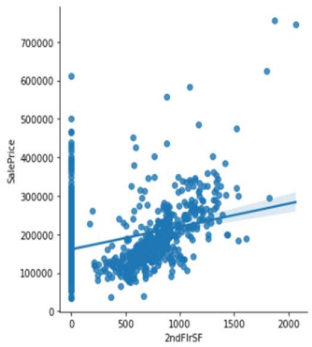
```
Out[41]: <seaborn.axisgrid.FacetGrid at 0x22c8e89ffa0>
```

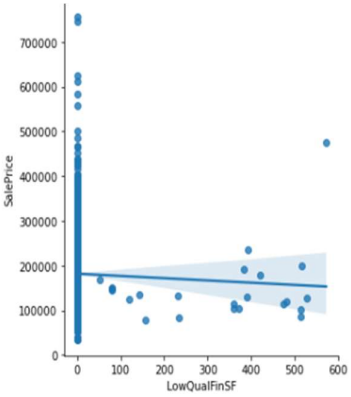




In [43]:
```
sns.lmplot(x='2ndFlrSF',y='SalePrice',data=df)
```

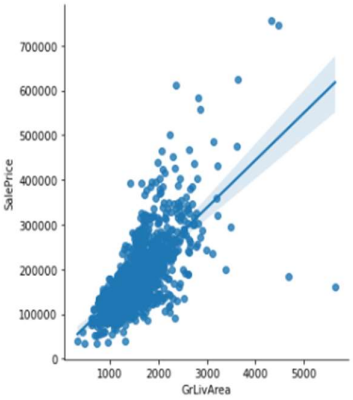
Out[43]: <seaborn.axisgrid.FacetGrid at 0x20ec5a47310>

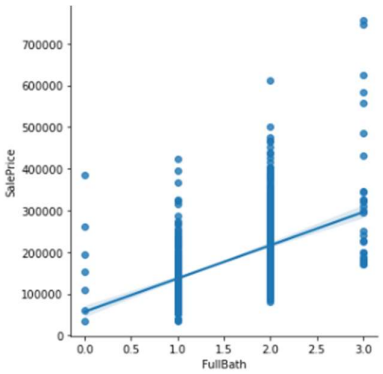




In [45]:
```
sns.lmplot(x='GrLivArea',y='SalePrice',data=df)
```

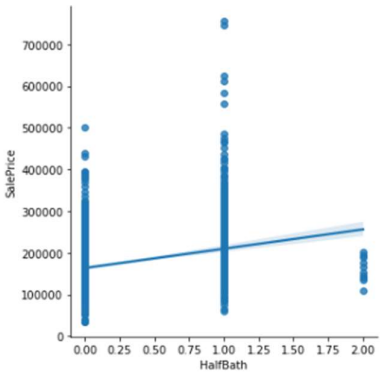
Out[45]: <seaborn.axisgrid.FacetGrid at 0x20ec50a9f70>

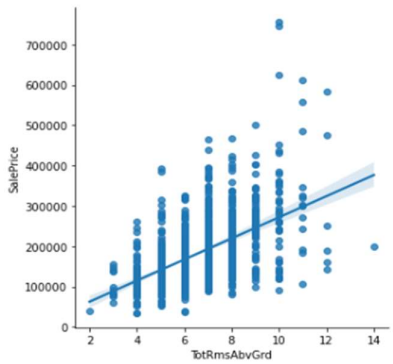




In [49]:
```
sns.lmplot(x='HalfBath',y='SalePrice',data=df)
```

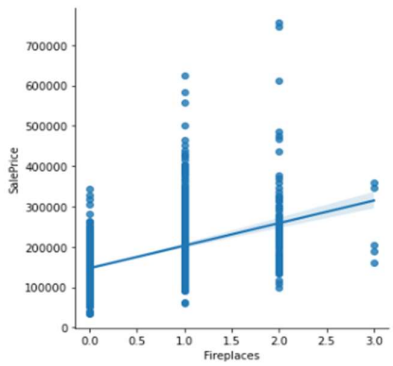
Out[49]:
```
<seaborn.axisgrid.FacetGrid at 0x20ec245a790>
```

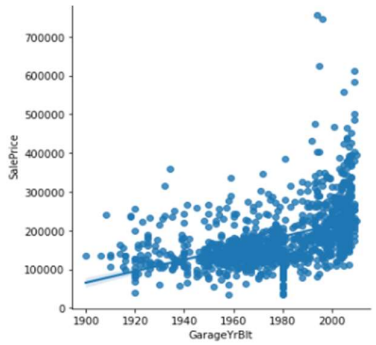




In [53]:
```
sns.lmplot(x='Fireplaces',y='SalePrice',data=df)
```

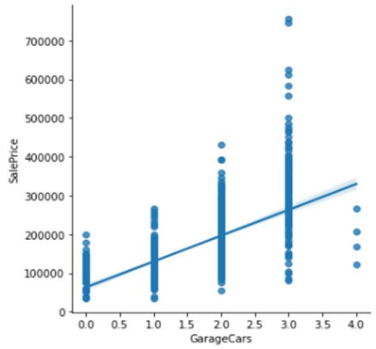
Out[53]:
```
<seaborn.axisgrid.FacetGrid at 0x20ec2ea3f10>
```

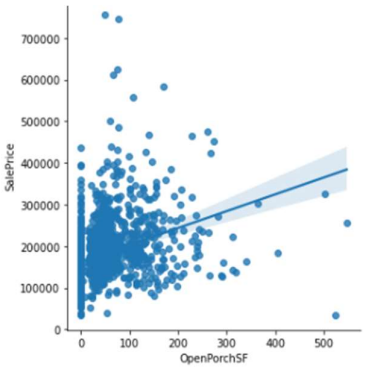




In [55]:
```
sns.lmplot(x='GarageCars',y='SalePrice',data=df)
```

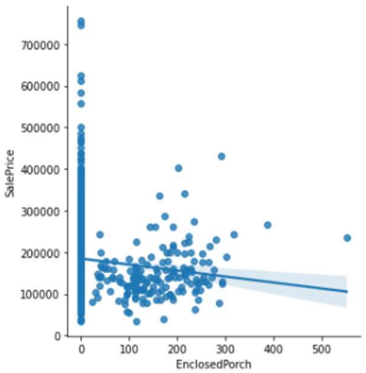
Out[55]:
```
<seaborn.axisgrid.FacetGrid at 0x20ec2a5f850>
```





In [59]:
```
sns.lmplot(x='EnclosedPorch',y='SalePrice',data=df)
```

Out[59]:
```
<seaborn.axisgrid.FacetGrid at 0x20ec5b4ea60>
```



➢ Interpretation of the Results

1. This dataset was very special as it had a separate train and test datasets. We have to work with both datasets simultaneously.
2. Firstly, the datasets were having null values and zero entries in maximum columns so we have to be careful while going through the statistical analysis of the datasets. And proper plotting for proper type of features will help us to get better insight on the data.
3. I found maximum numerical continuous columns were in linear relationship with target column.
4. I notice a huge number of outliers and skewness in the data so we have chosen proper methods to deal with the outliers and skewness. If we ignore this outlier and skewness, we may end up with a bad model which has less accuracy.
5. Then scaling both train and test dataset has a good impact like it will help the model not to get biased.
6. We have to use multiple models while building model using train dataset as to get the best model out of it.

CONCLUSION

➢ Key Findings and Conclusions of the Study

In this project report, we have used machine learning algorithms to predict the house prices. We have mentioned the step-by-step procedure to analyse the dataset and finding the correlation Between the features. Thus, we can select the features which are not correlated to each other and are independent in nature. These feature set were then given as an input to five algorithms and a Csv file was generated consisting of predicted house prices.

Hence, we calculated the performance of each model using different performance metrics and compared them based on these metrics.

And saved the model as filename="US_Housepred.pkl"

➢ Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle as it contains all types of data in it. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove missing value and to replace null value and zero values with there respective mean, median or mode. This study is an exploratory attempt to use machine learning algorithms in estimating housing prices, and then compare their results. To conclude, the application of machine learning in property research is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to property appraisal, and presenting an alternative approach to the valuation of housing prices.

Future direction of research may consider incorporating additional property transaction data from a larger geographical location with more features, or analysing other property types beyond housing development.

➢ Limitations of this work and Scope for Future Work

This model doesn't predict future prices of the houses mentioned by the customer. Due to this, the risk in investment in an apartment or an area increases considerably. To minimize this error, customers tend to hire an agent which again increases the cost of the process.