HOUSING: PRICE PREDICTION

Project Report

Submitted By
Naveen Kumar Ranganathan

Acknowledgement

I would like to express my sincere thanks and gratitude to mymentors from Data Trained academy and Flip Robo Technologies for providing me an opportunity to work on this project. Their suggestions and directions have helped me in the completion of this project successfully. All the required information & the dataset are provided by Flip Robo Technologies

Also, I have utilized a few external resources which helped me to complete this project. All the external resources that were used in creating this project are listed below:

References:

www.geeksforgeeks.org
https://www.kaggle.com/
https://github.com/
https://www.emerald.com/insight/content/doi/10.1108/IJHMA-02-2022-0025/full/html#:~:text=Predictive%20analytics%20is%20an%20effective%20way%20to%20deal,independent%20variables%20contribute%20to%20the%20housing%20price%20prediction.
https://www.researchgate.net/publication/349477129_House_Price_Prediction

Introduction

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.

Business Problem Framing

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:

• 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

After the 2008 global crisis, the housing market fell for several years, especially in the large cities, until the end of 2011. Beginning with 2012, the housing market followed and upward trend with decreasing inventories, increasing demand, and naturally increasing prices. This again made economists and market analysers turn their attention to more precise prediction models to shield the economy from predictable threats that could result in economic downturns (Park & Kwon Bae, 2015). Machine learning has been used in prediction for some time now with increasingly better results that were put in practice and changed the economic landscape.

Practically every economic domain now benefits from machine learning prediction models, and the current models are becoming more accurate given the computational power available for processing immense sets of data.

Review of Literature

Housing and real estate market is one of the markets which is one of the major contributors in the world’s economy. 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.

Likewise we will be creating Machine learning model with the dataset available to predict the price of the houses based on the various conditions present in the dataset

Motivation for the Problem Undertaken

We are 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.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

The study is to predict the sale price of the house and analyzing which features are important and how they contribute in the prediction. There are two datasets. One is train dataset which is supervised and another one is test dataset which is unsupervised.

The target variable is “SalePrice” and it is a regression type problem. I have used train dataset to build machine learning models and then by using this model I made prediction for the test dataset.

I have observed some columns having more than 70% of zero and null values so, I decided to drop those columns. I have performed both univariate and bivariate analysis to analyse the sale price of the house. I have analysed the categorical and numerical features using categorical plots and numerical plots respectively to get better insights from the data.

In this project I have done various mathematical and statistical analysis such as describing the statistical summary of the columns, feature engineering, treating null values, removing outliers, skewness, encoding the data etc. Checked for correlation between the features and visualized it using heat map. Also, I built many regression algorithms while building machine learning models, used hyper tuning method for best model and saved the best model. Finally, I predicted the sale price of the house using the saved trained model.

Data Sources and their formats

We have been given with two datasets train and test

- Train dataset contains 1168 columns and 81 rows, out of 81 columns, 80 are independent variables and remaining 1 is dependent variable (SalePrice)
- Test dataset contains 292 rows and 80 columns

Data contains numerical as well as categorical variable.

Data Pre-processing Done

- Imported the necessary libraries and imported train datasets which was in csv format
- By using info method we can see that both train and test data contains categorical and numerical data
- isnull().sum() method was used to find the null values and the missing ratio was more than 80% present in columns PoolQC','MiscFeature','Alley','Fence'
- EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, BsmtHalfBath columns contains more number of zero values
- We have 4 year variables YearBuilt, YearRemodAdd, GarageYrBlt and YrSold was converted into age for further analysis
- As TotalBsmtSF = BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF, we can remove these columns as we have the total value in TotalBsmtSF
- As GrLivArea = 1stFlrSF + 2ndFlrSF + LowQualFinSF, we can remove these columns as we have the total value in   GrLivArea

- Data has been split into Categorical and Numerical do do further analysis by visualization
- SimpleImputer method used to replace Null values for both Categorical and Numerical
- Used Univariate and Bivariate Analysis on the train dataset to understand the spread of the data
- By LabelEncoder Categorical data was converted into integer for further analysis
- Analysed the data with describe method and Mean is greater than median(50%) in most of the columns which shows data is skewed on right side
- 75% to max value difference is high in most of the columns which means outliers are present
- Skewness in the dataset was transformed using yeo-johnson method
- Below are the columns which are highly positively correlated with the target variables

  FullBath
  TotalBsmtSF
  GarageArea
  GarageCars
  GrLivArea
  OverallQual
- Below are the columns which are highly negatively correlated with the target variables

  BsmtQual
  ExterQual
  KitchenQual
- Z Score Method used to detect and removed Outliers with 9.16% in train dataset and 9.24 % in test dataset
- Most of the columns inflation was higher when checking the VIF but before dropping other columns applied Scaling method to control the data bias ness
- StandardScaler method was used to control the data bias ness

Hardware & Software Requirements & Tools Used Data

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required: • Processor: core i5 or above • RAM: 8 GB or above • ROM/SSD: 250 GB or above

Software required: • Anaconda 3- language used Python 3

Below are the libraries used for the project

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
import warnings
warnings.filterwarnings('ignore')
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import PowerTransformer
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
from scipy.stats import zscore
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neighbors import KneighborsRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import joblib
```

Inputs- Logic- Output Relationships

- Checked the correlation between the target and features using heat map and bar plot
- EDA analysis was done to understand the relation between features and targets
- Both train and test dataset are similar, based on train dataset analysis, test data set was mirrored for Model testing
- Both the data set got 9% data loss when applied Z score method

MODEL/S DEVELOPMENT AND EVALUATION

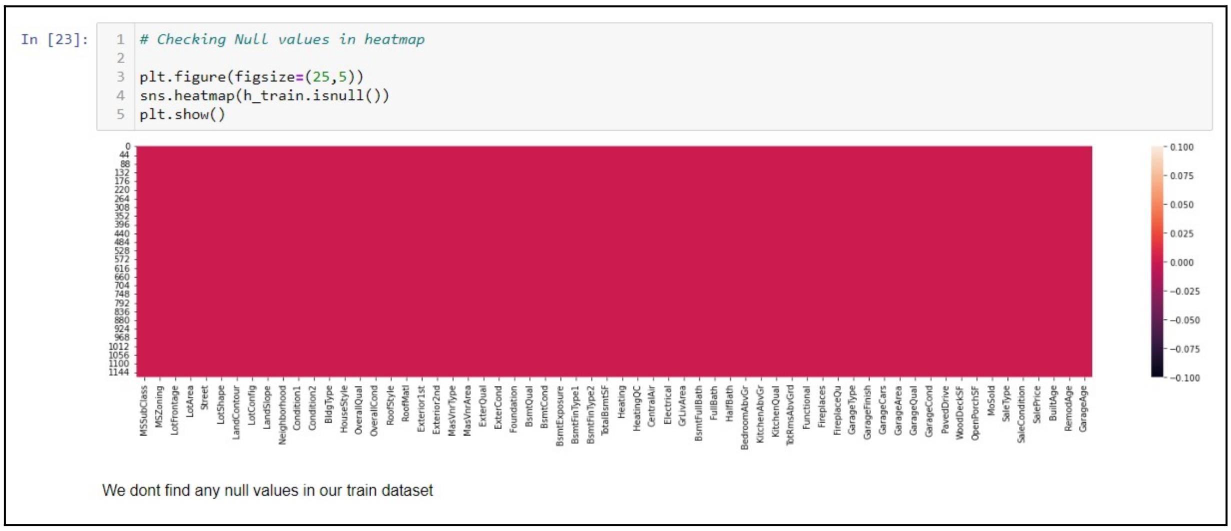
Identification of possible Problem-solving approaches (Methods):

- Data has been split into Categorical and Numerical do do further analysis by visualization
- SimpleImputer method used to replace Null values for both Categorical and Numerical
- Used Univariate and Bivariate Analysis on the train dataset to understand the spread of the data
- By LabelEncoder Categorical data was converted into integer for further analysis
- Analysed the data with describe method and Mean is greater than median(50%) in most of the columns which shows data is skewed on right side
- 75% to max value difference is high in most of the columns which means outliers are present
- Skewness in the dataset was transformed using yeo-johnson method

- Linear Regression ,Random Forest Regressor, Support Vector Regressor, Decision Tree Regressor, Gradient Boost Regressor, K Neighbors Regressor, Bagging Regressor, SGDRegressor, AdaBoostRegressor ,ExtraTrees Regressor are the model evaluated and found Gradient Boost Regressor R2 score was the highest.

Visualizations

- Used heat map to visualize the null values

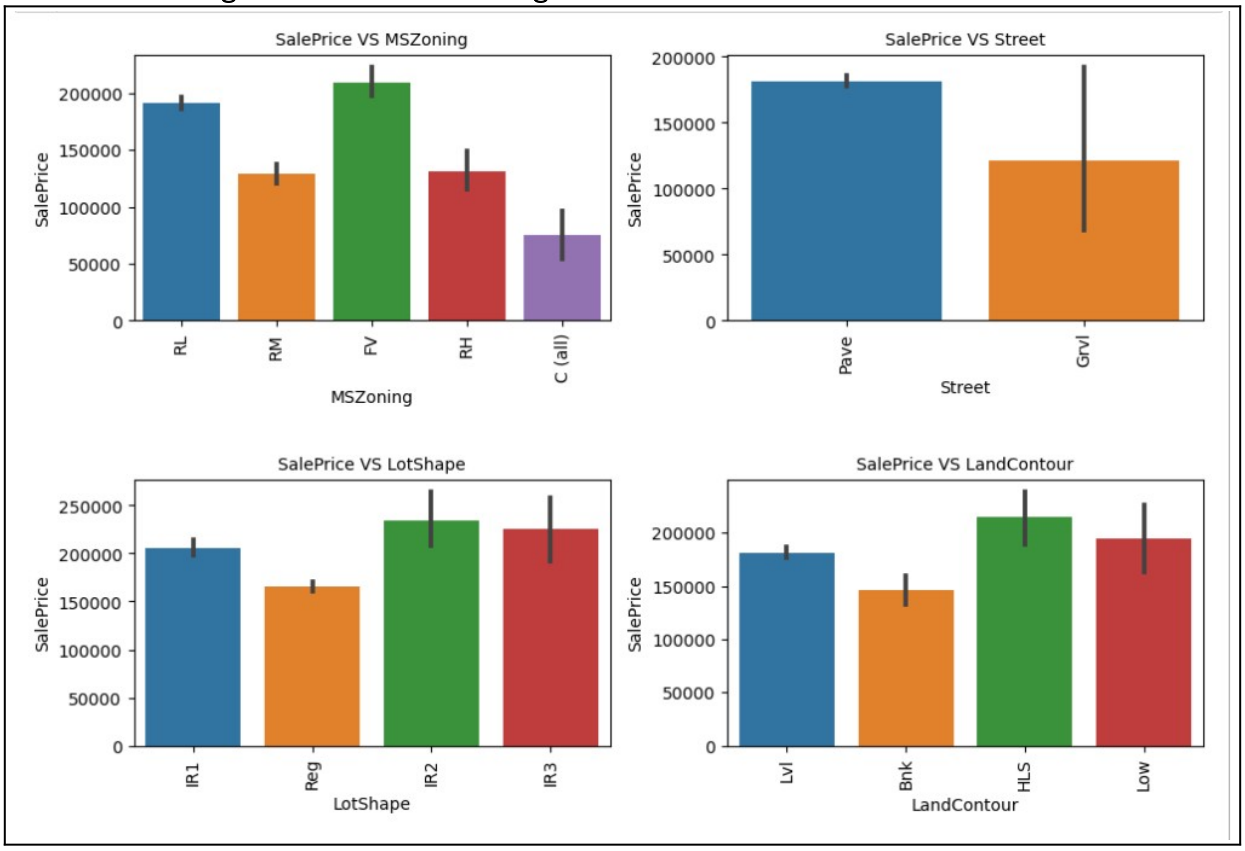


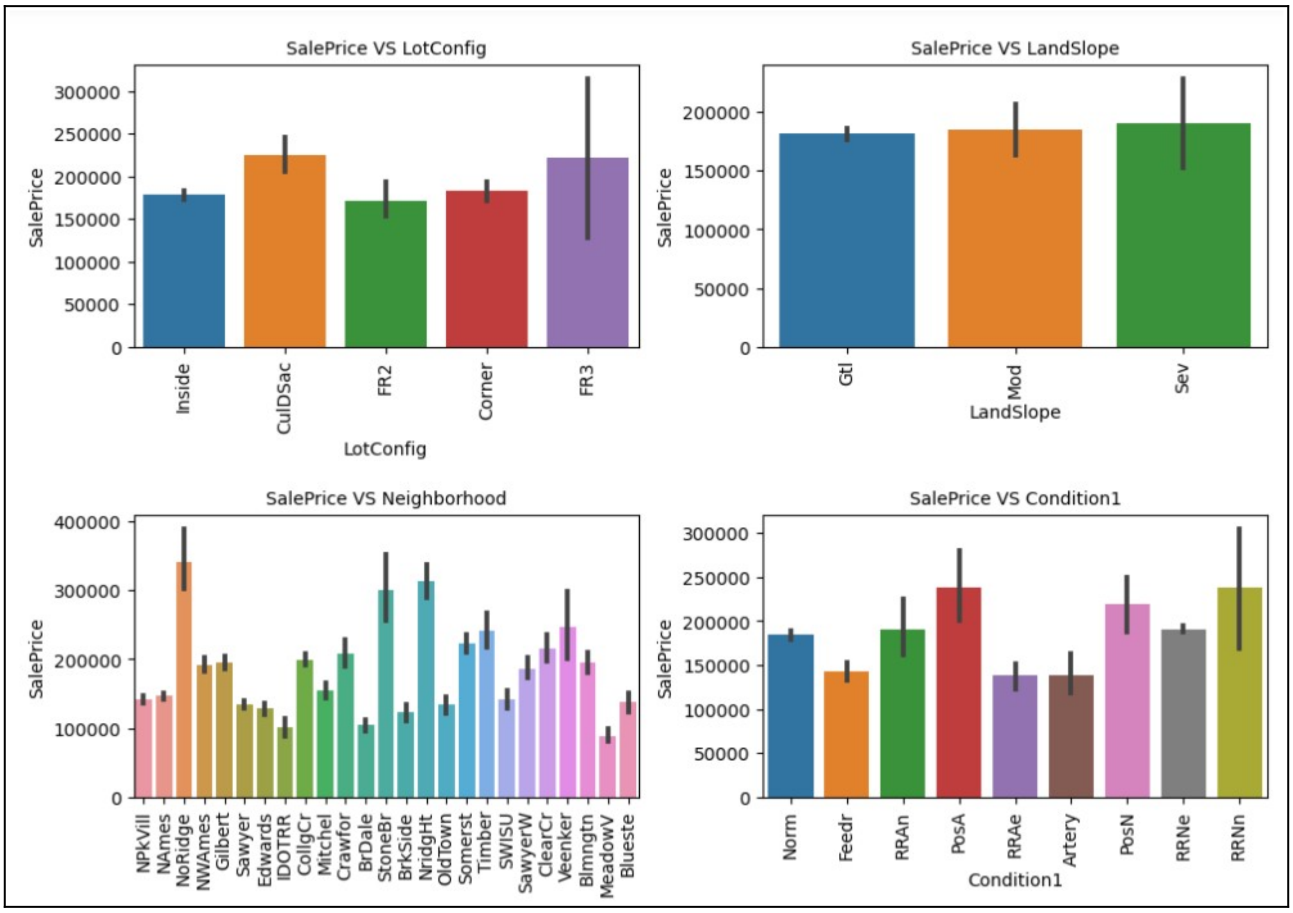
```
In [23]:
# Checking Null values in heatmap

plt.figure(figsize=(25,5))
sns.heatmap(h_train.isnull())
plt.show()
```

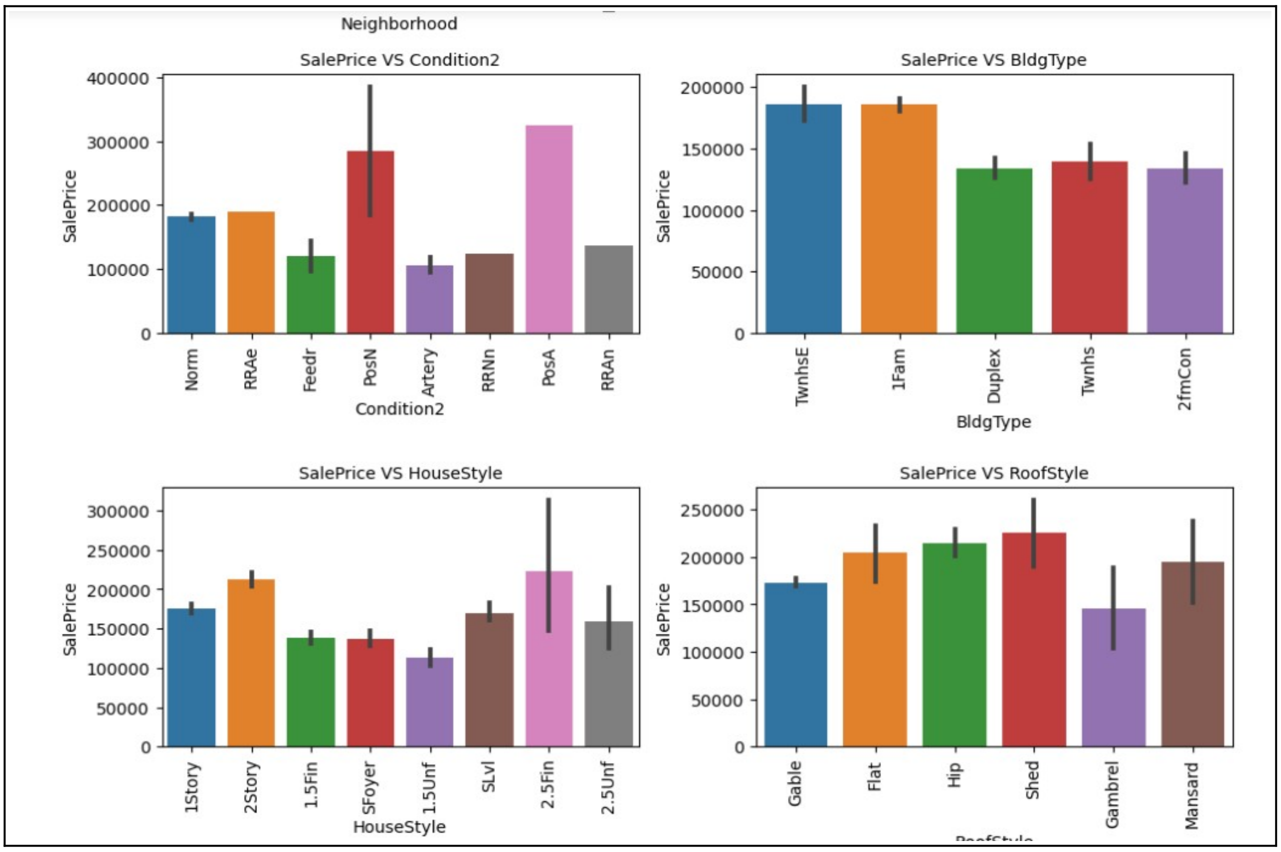
0 44 88 132 176 220 264 308 352 396 440 484 528 572 616 660 704 748 792 836 880 924 968 1012 1056 1100 1144
MSSubClass MSZoning LotFrontage LotArea Street LotShape LandContour LotConfig LandSlope Neighborhood Condition1 Condition2 BldgType HouseStyle OverallQual OverallCond RoofStyle RoofMatl Exterior1st Exterior2nd MasVnrType MasVnrArea ExterQual ExterCond Foundation BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinType2 TotalBsmtSF Heating HeatingQC CentralAir Electrical GrLivArea BsmtFullBath FullBath HalfBath BedroomAbvGr KitchenAbvGr KitchenQual TotRmsAbvGrd Functional Fireplaces FireplaceQu GarageType GarageFinish GarageCars GarageArea GarageQual GarageCond PavedDrive WoodDeckSF OpenPorchSF MoSold SaleType SaleCondition SalePrice BuiltAge RemodAge GarageAge
0.100 0.075 0.050 0.025 0.000 −0.025 −0.050 −0.075 −0.100

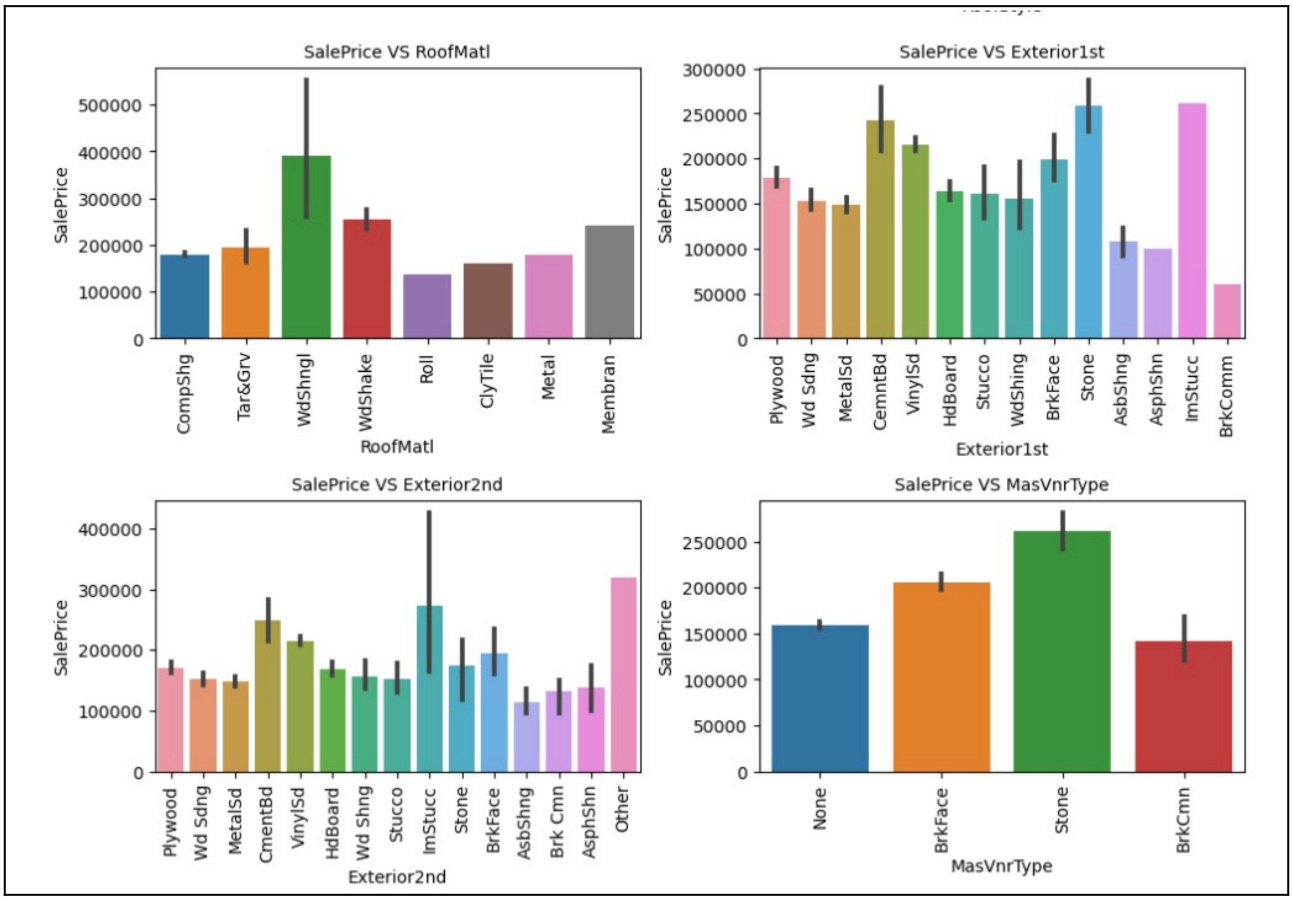
We dont find any null values in our train dataset

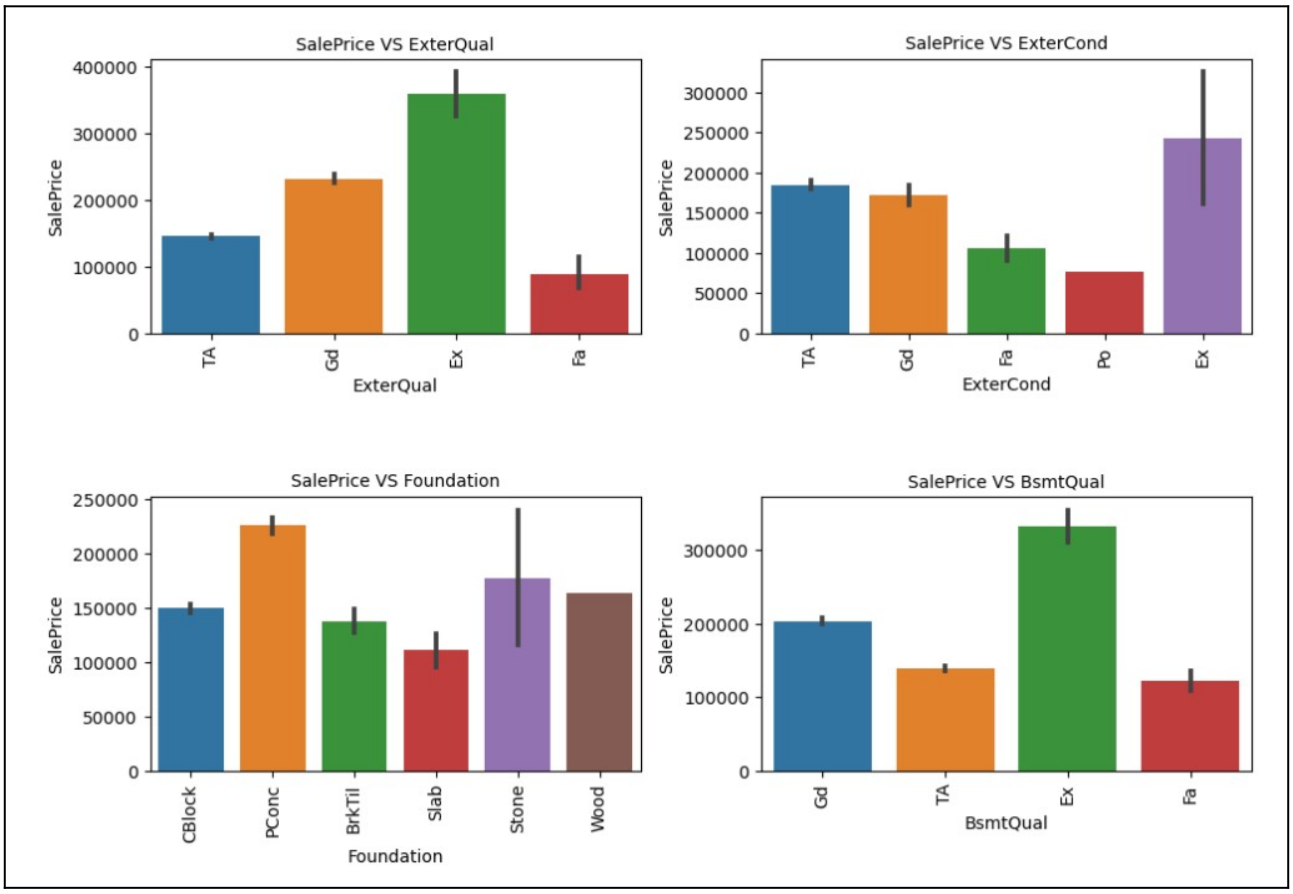
- Visualizing the features with target variable

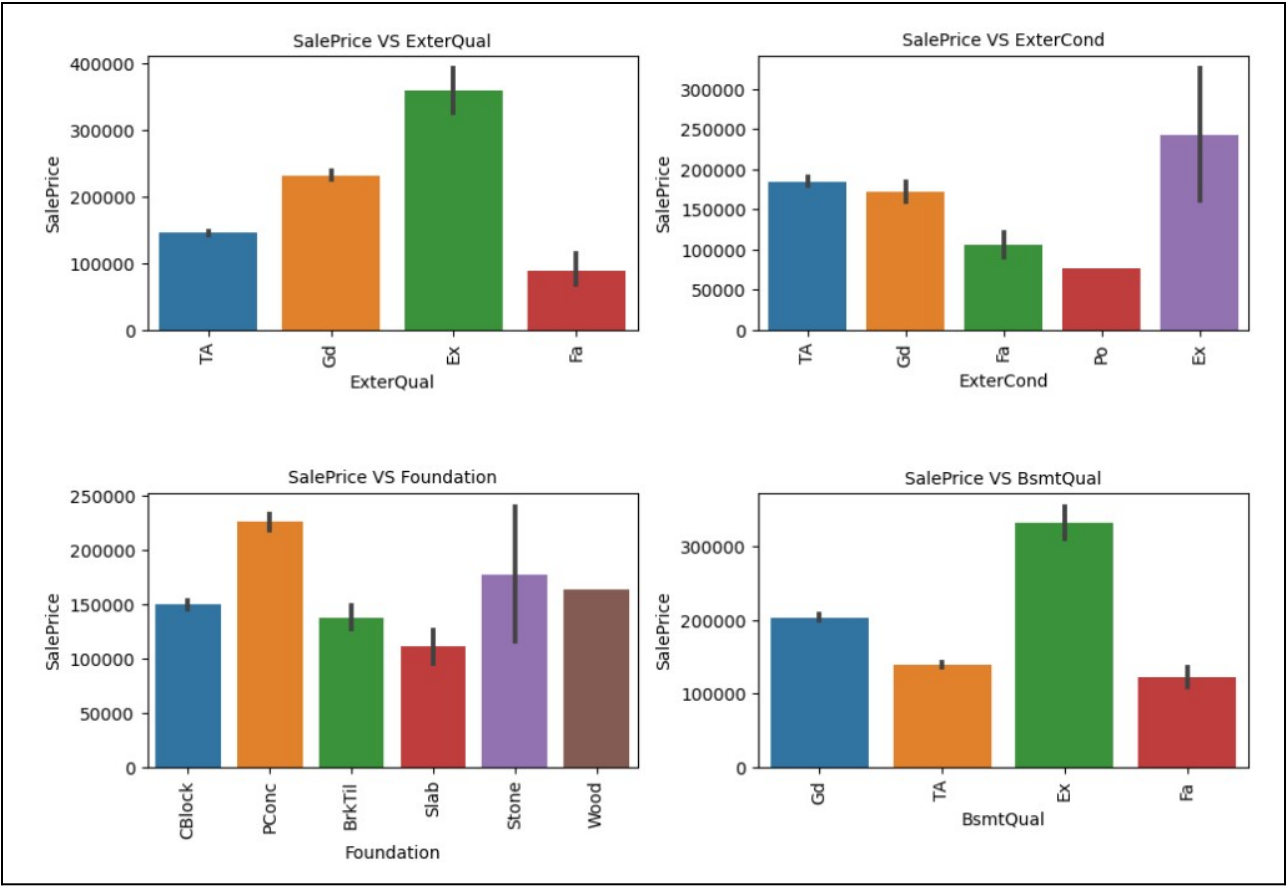


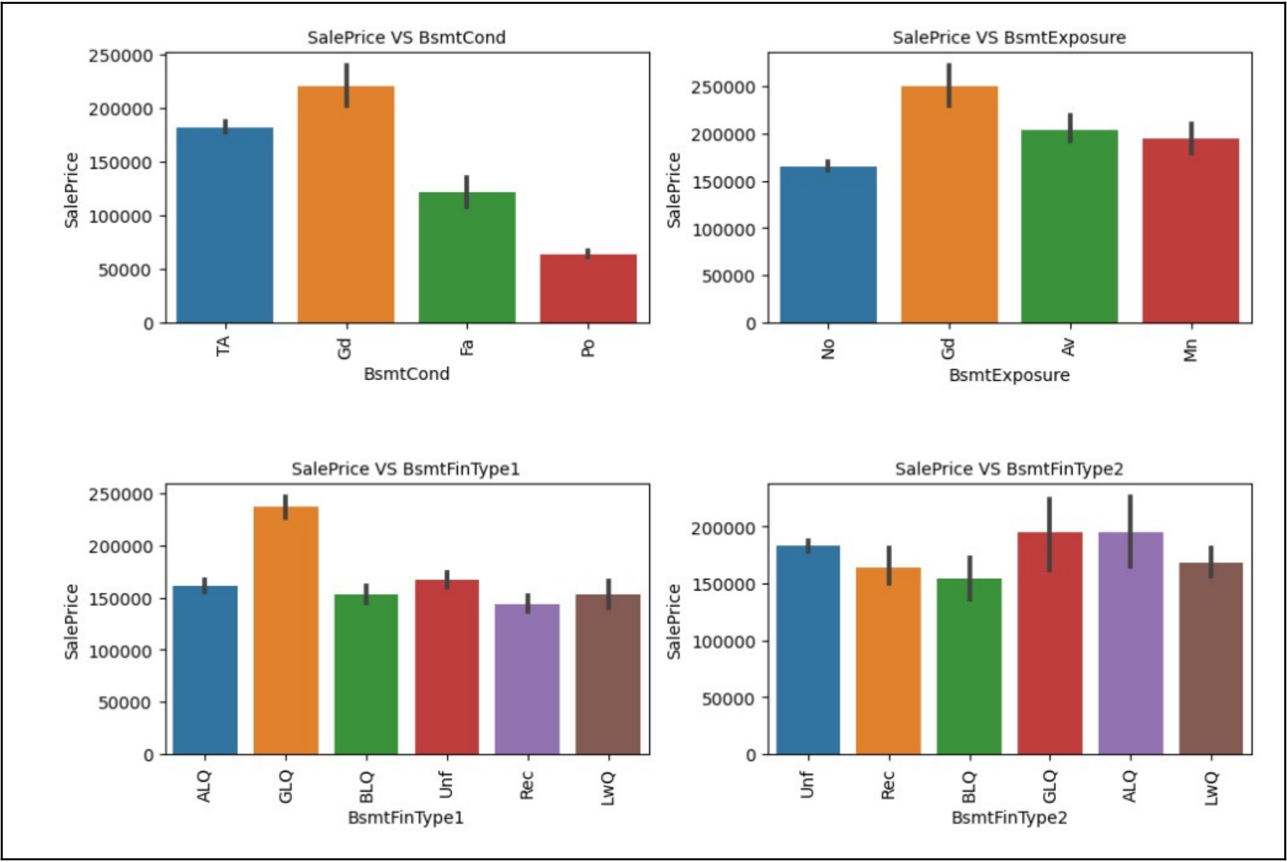

SalePrice VS LotConfig
SalePrice
LotConfig
Inside
CulDSac
FR2
Corner
FR3
SalePrice VS LandSlope
LandSlope
Gtl
Mod
Sev
SalePrice VS Neighborhood
NPkVill
NAmes
NoRidge
NWAmes
Gilbert
Sawyer
Edwards
IDOTRR
CollgCr
Mitchel
Crawfor
BrDale
StoneBr
BrkSide
NridgHt
OldTown
Somerst
Timber
SWISU
SawyerW
ClearCr
Veenker
Blmngtn
MeadowV
Blueste
SalePrice VS Condition1
Condition1
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn

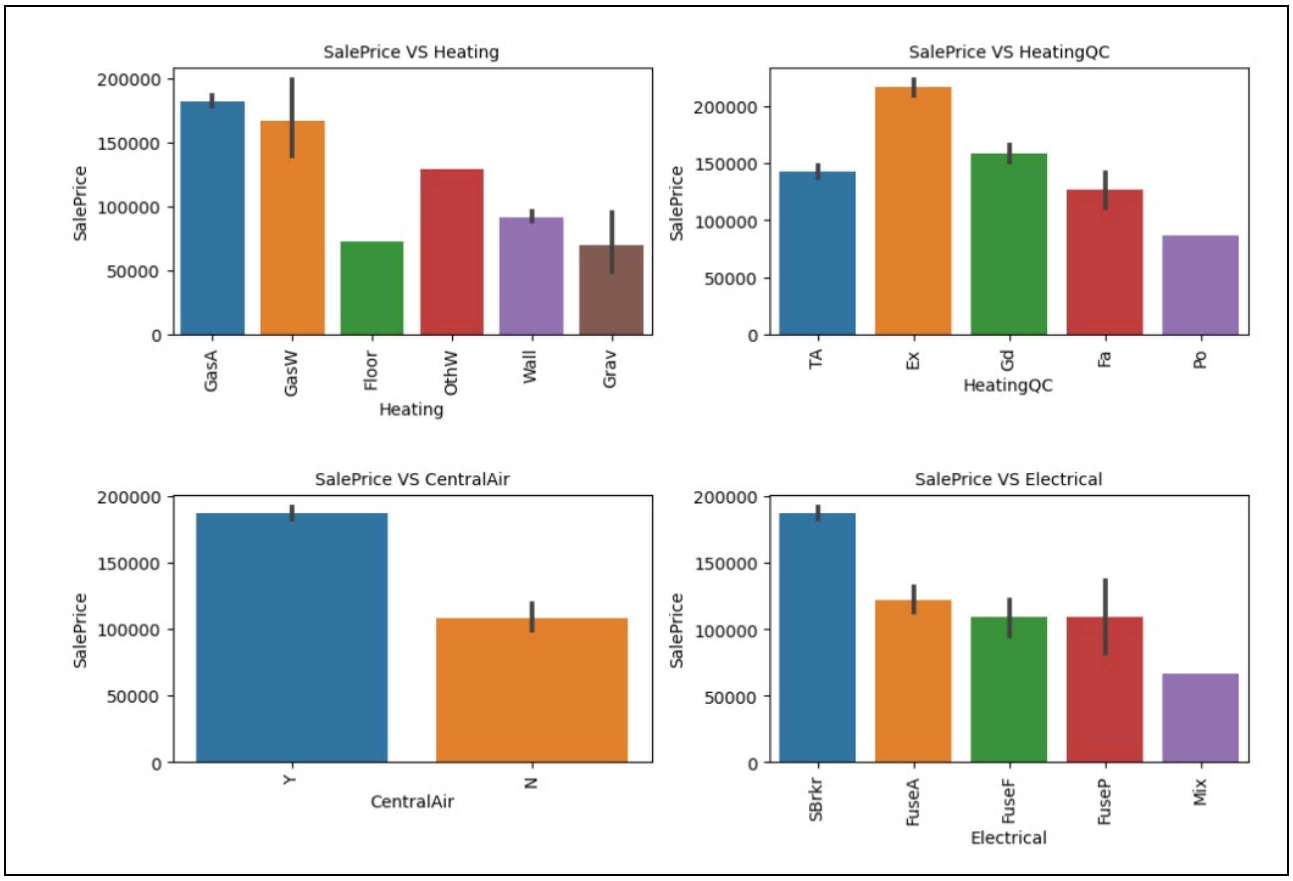

Neighborhood
SalePrice VS Condition2
SalePrice
Condition2
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
SalePrice VS BldgType
BldgType
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
SalePrice VS HouseStyle
HouseStyle
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
SalePrice VS RoofStyle
Gable
Flat
Hip
Shed
Gambrel
Mansard

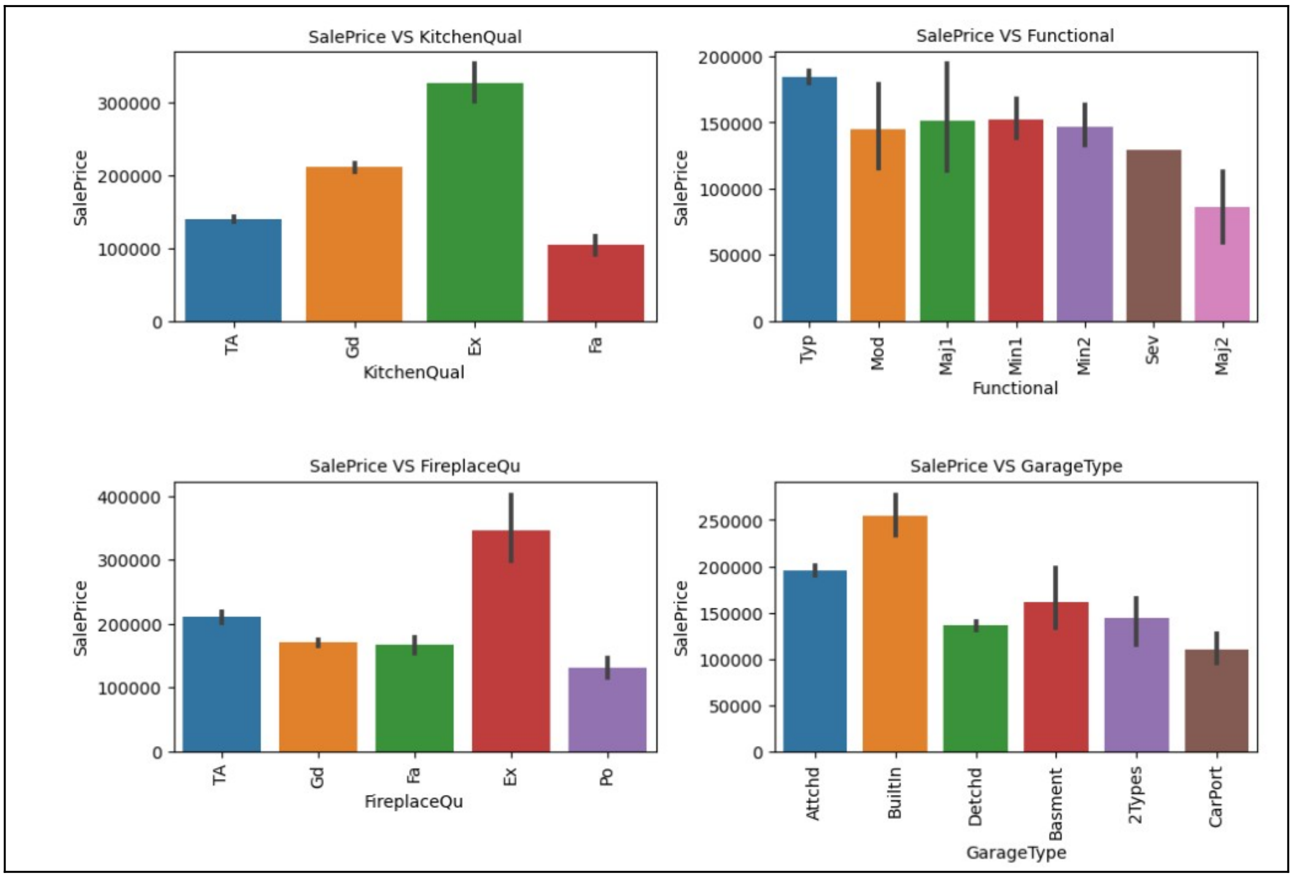

SalePrice VS RoofMatl
SalePrice
RoofMatl
CompShg
Tar&Grv
WdShngl
WdShake
Roll
ClyTile
Metal
Membran
SalePrice VS Exterior1st
SalePrice
Exterior1st
Plywood
Wd Sdng
MetalSd
CemntBd
VinylSd
HdBoard
Stucco
WdShing
BrkFace
Stone
AsbShng
AsphShn
ImStucc
BrkComm
SalePrice VS Exterior2nd
SalePrice
Exterior2nd
Plywood
Wd Sdng
MetalSd
CmentBd
VinylSd
HdBoard
Wd Shng
Stucco
ImStucc
Stone
BrkFace
AsbShng
Brk Cmn
AsphShn
Other
SalePrice VS MasVnrType
SalePrice
MasVnrType
None
BrkFace
Stone
BrkCmn

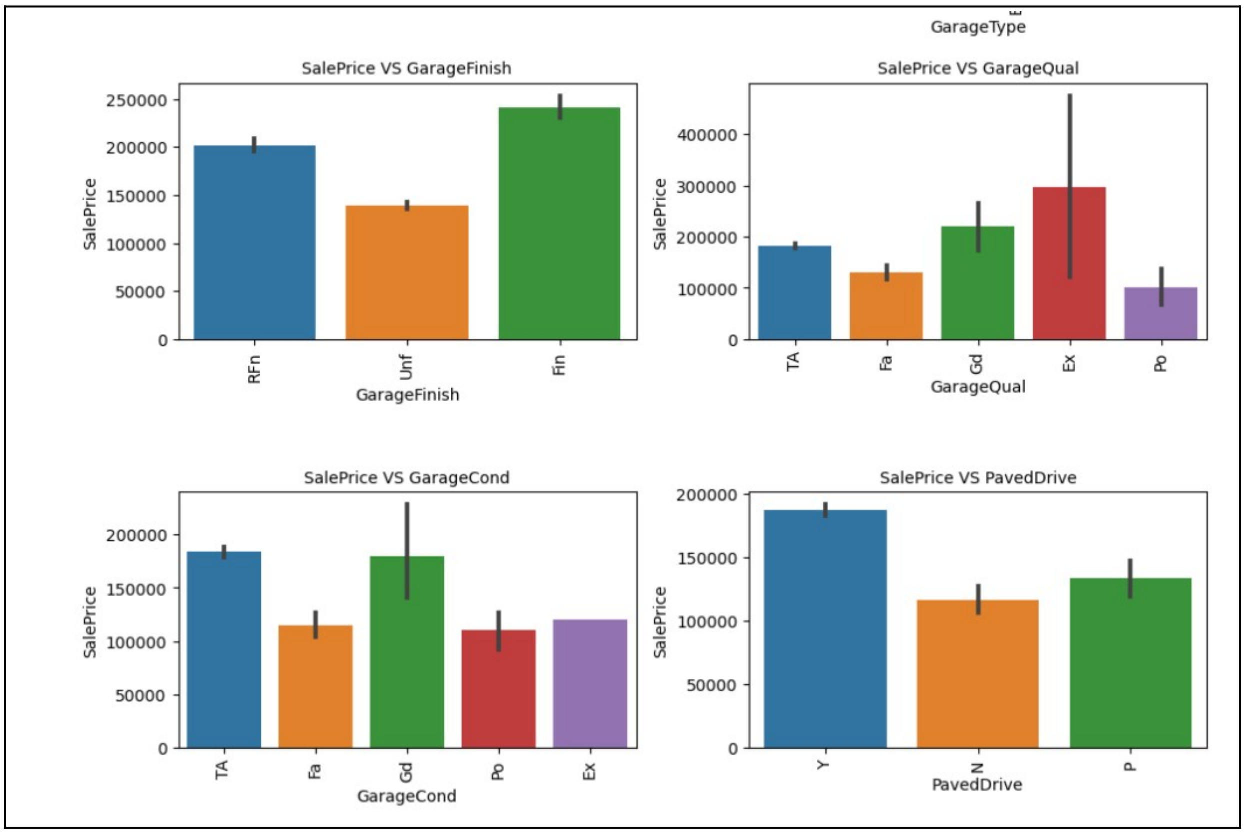

SalePrice VS ExterQual
SalePrice
ExterQual
TA
Gd
Ex
Fa
SalePrice VS ExterCond
SalePrice
ExterCond
TA
Gd
Fa
Po
Ex
SalePrice VS Foundation
SalePrice
Foundation
CBlock
PConc
BrkTil
Slab
Stone
Wood
SalePrice VS BsmtQual
SalePrice
BsmtQual
Gd
TA
Ex
Fa

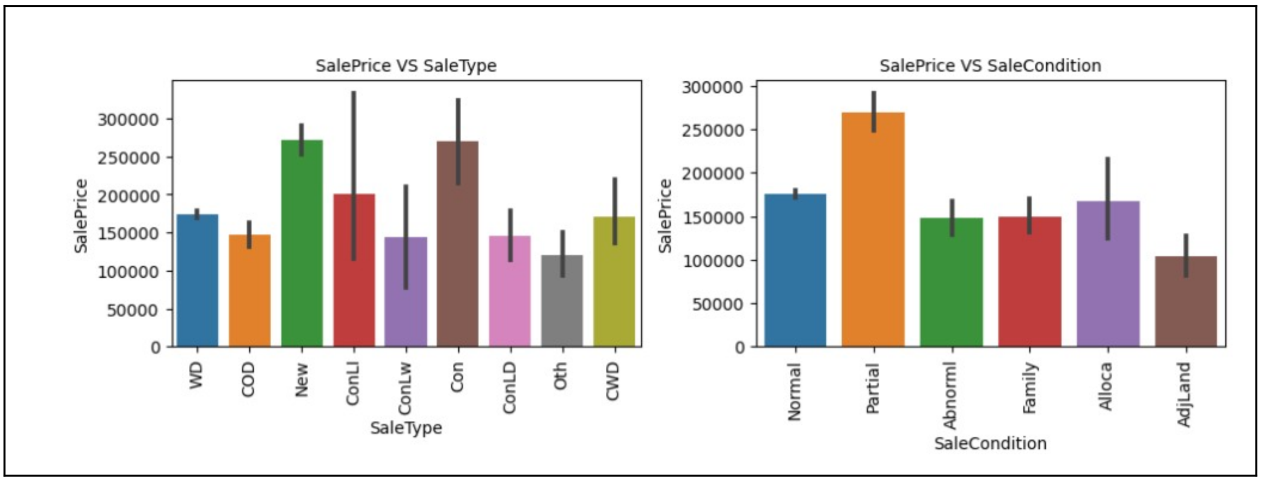

SalePrice VS ExterQual
SalePrice
ExterQual
TA
Gd
Ex
Fa
0
100000
200000
300000
400000
SalePrice VS ExterCond
SalePrice
ExterCond
TA
Gd
Fa
Po
Ex
0
50000
100000
150000
200000
250000
300000
SalePrice VS Foundation
SalePrice
Foundation
CBlock
PConc
BrkTil
Slab
Stone
Wood
0
50000
100000
150000
200000
250000
SalePrice VS BsmtQual
SalePrice
BsmtQual
Gd
TA
Ex
Fa
0
100000
200000
300000


SalePrice VS BsmtCond
SalePrice
BsmtCond
TA
Gd
Fa
Po
0
50000
100000
150000
200000
250000
SalePrice VS BsmtExposure
SalePrice
BsmtExposure
No
Gd
Av
Mn
0
50000
100000
150000
200000
250000
SalePrice VS BsmtFinType1
SalePrice
BsmtFinType1
ALQ
GLQ
BLQ
Unf
Rec
LwQ
0
50000
100000
150000
200000
250000
SalePrice VS BsmtFinType2
SalePrice
BsmtFinType2
Unf
Rec
BLQ
GLQ
ALQ
LwQ
0
50000
100000
150000
200000


SalePrice VS Heating
SalePrice
Heating
GasA
GasW
Floor
OthW
Wall
Grav
SalePrice VS HeatingQC
HeatingQC
TA
Ex
Gd
Fa
Po
SalePrice VS CentralAir
CentralAir
Y
N
SalePrice VS Electrical
Electrical
SBrkr
FuseA
FuseF
FuseP
Mix


SalePrice VS KitchenQual
SalePrice
KitchenQual
TA
Gd
Ex
Fa
SalePrice VS Functional
Functional
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
SalePrice VS FireplaceQu
FireplaceQu
TA
Gd
Fa
Ex
Po
SalePrice VS GarageType
GarageType
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort



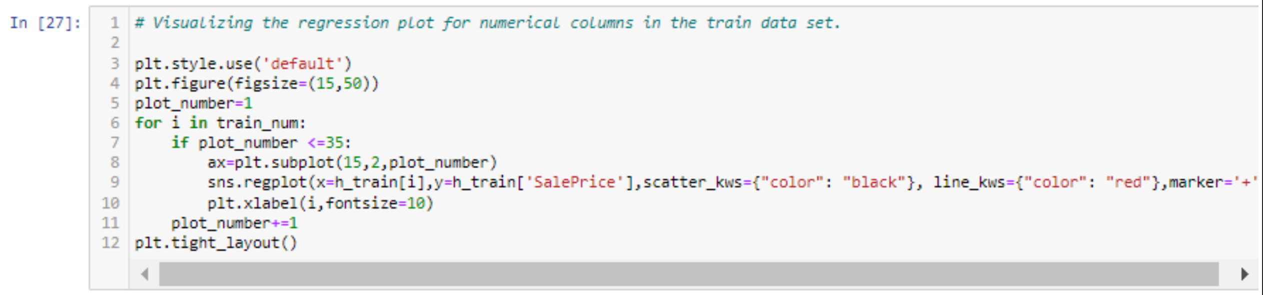


Observations:

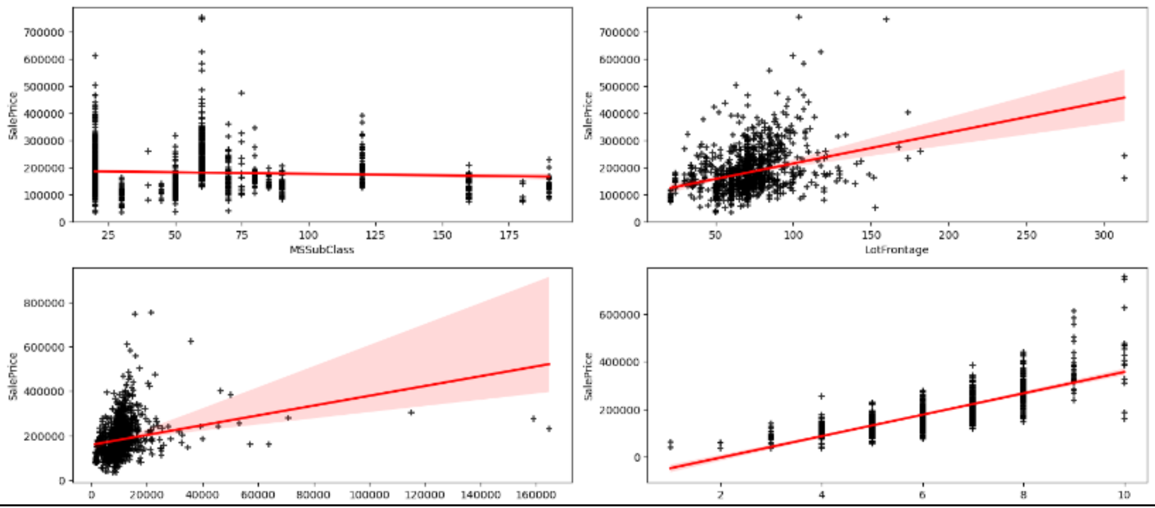
- Residential Low Density count is high as compare to other MSZoning
- In the Street Pave count is high, in increase of sales it also depends on the type of street
- In the Neighborhood: Physical locations within Ames city limits, NAMES has the maximum count as compare to others

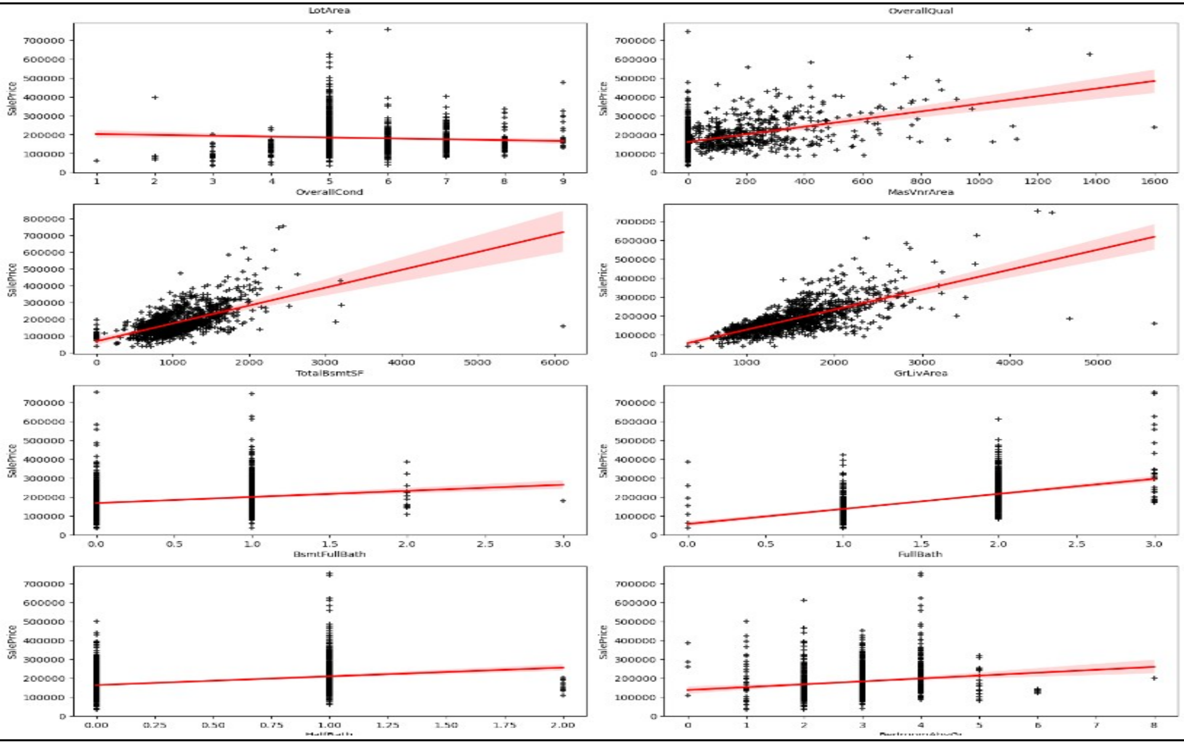
- In Condition1, proximity to various condition the normal condition has the maximum count. Many people to have normal proximity condition. So, its count is high
- In BldgType, the type of dwelling, 1 Farm-Single Family Detached has the maximum count
- Houses which are having style of dwelling 2nd level finished and two story have high sale price compared to other types
- The houses having the roof style Flat, Hip and Shed have high sale price and the houses having gabrel roof style have less sale price
- Houses with Wood Shingles root materials have high sale prices
- Houses having Imitation Stucco, Stone and Cement Board as 1st exterior cover have high sale price
- Houses having Imitation Stucco and other as 2nd cover have high sale price
- Houses having Stone Masonry veneer type have high sale price than other types
- Houses having Poured Concrete as foundation type have high sale price compared to other types
- For Typical/Average(TA) and good Kitchen quality the count is maximum for the feature Kitchen quality(KitchenQual)
- For good Fireplace quality the count is high for the feature Fireplace quality(FireplaceQu)
- If Garage location Attached to home then the count is high, for the feature Garage location(GarageType)
- For Unfinished Interior of the garage the count is maximum, for the feature Interior finish of the garage(GarageFinish)
- For Typical/Average(TA) Garage quality the count is high, for the feature Garage quality(GarageQual)
- For Typical/Average(TA) Garage condition the count is high, for the feature Garage condition(GarageCond)

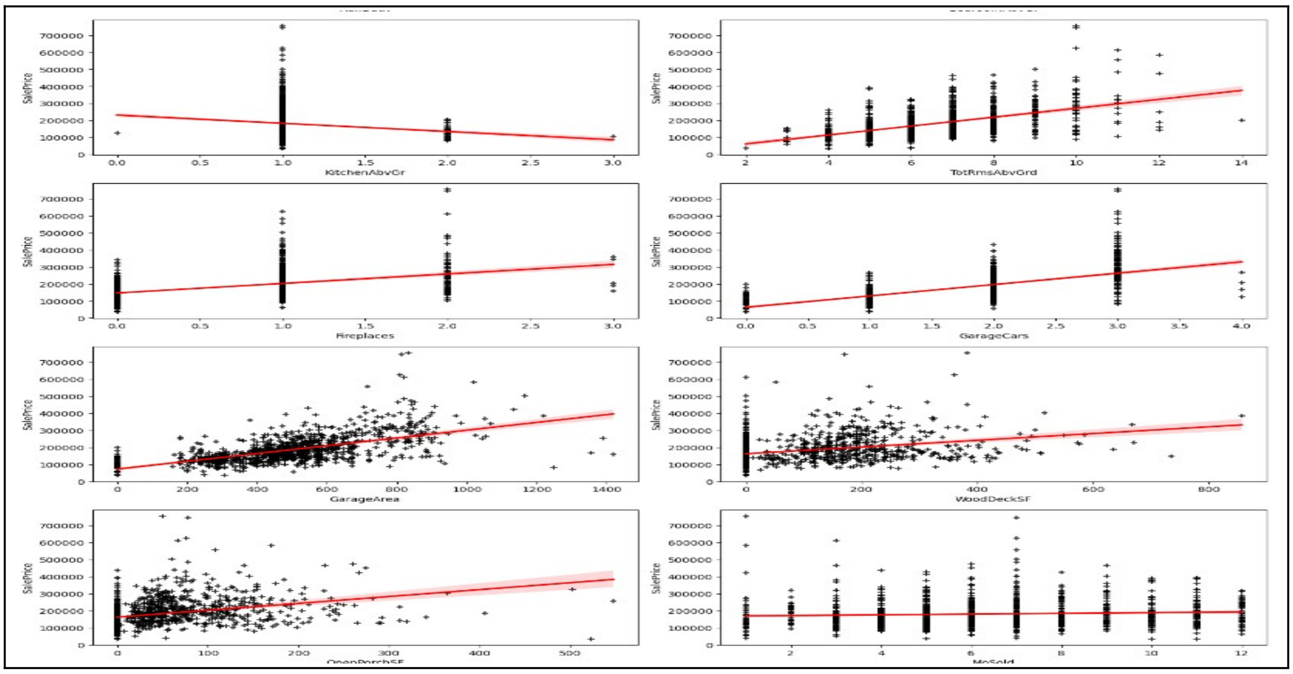
Visualizing the regression plot for numerical columns in the train data set.

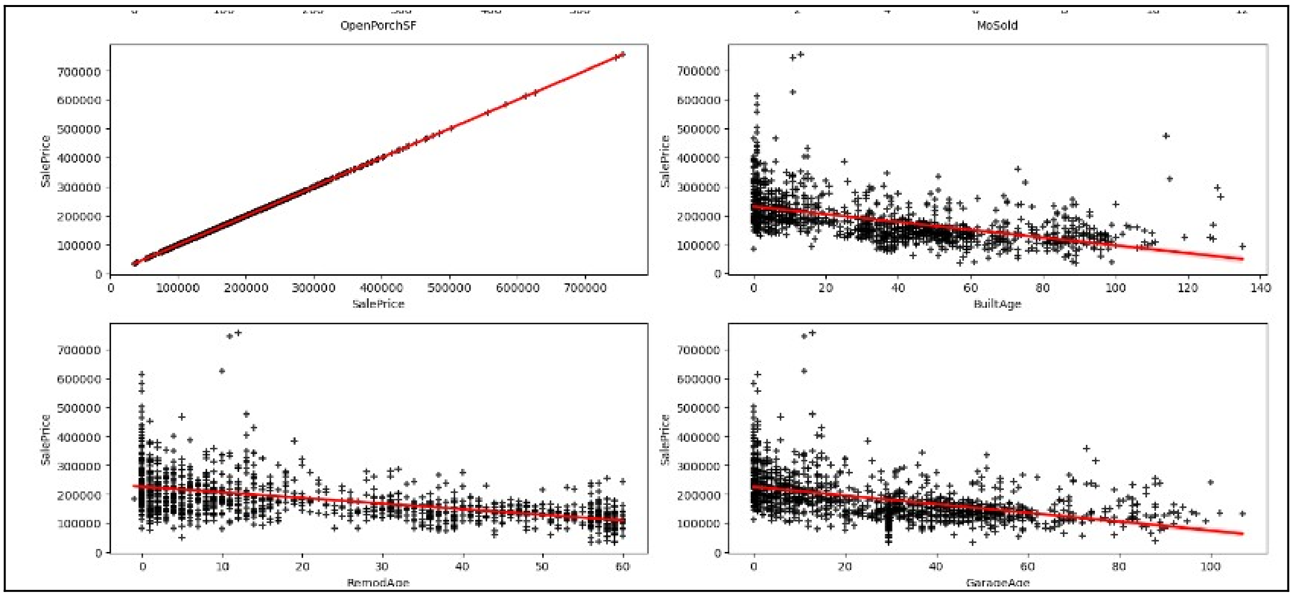


```
In [27]:
# Visualizing the regression plot for numerical columns in the train data set.

plt.style.use('default')
plt.figure(figsize=(15,50))
plot_number=1
for i in train_num:
    if plot_number <=35:
        ax=plt.subplot(15,2,plot_number)
        sns.regplot(x=h_train[i],y=h_train['SalePrice'],scatter_kws={"color": "black"}, line_kws={"color": "red"},marker='+'
        plt.xlabel(i,fontsize=10)
    plot_number+=1
plt.tight_layout()
```







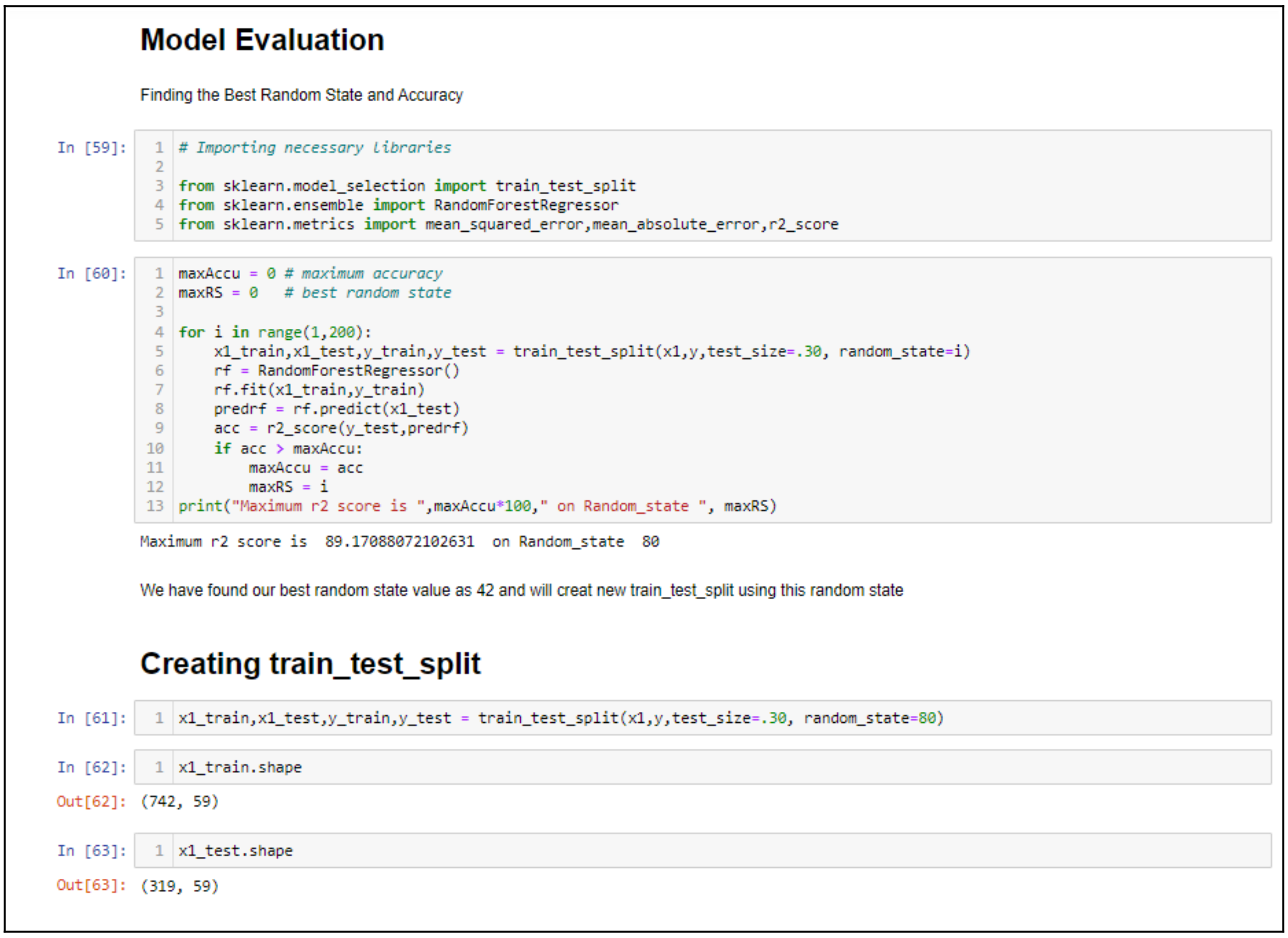


Testing of Identified Approaches (Algorithms)

- Linear Regression
- Random Forest Regressor
- Support Vector Regressor,
- Decision Tree Regressor
- Gradient Boost Regressor
- K Neighbors Regressor
- Bagging Regressor

- SGDRegressor
- AdaBoostRegressor
- ExtraTrees Regressor
  Are the models evaluated and found Gradient Boost Regressor R2 score was the highest with 89.49% as R2 score

Run and evaluate selected models



Model Evaluation

Finding the Best Random State and Accuracy

```
In [59]:
# Importing necessary libraries

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

```
In [60]:
maxAccu = 0 # maximum accuracy
maxRS = 0   # best random state

for i in range(1,200):
    x1_train,x1_test,y_train,y_test = train_test_split(x1,y,test_size=.30, random_state=i)
    rf = RandomForestRegressor()
    rf.fit(x1_train,y_train)
    predrf = rf.predict(x1_test)
    acc = r2_score(y_test,predrf)
    if acc > maxAccu:
        maxAccu = acc
        maxRS = i
print("Maximum r2 score is ",maxAccu*100," on Random_state ", maxRS)
```

```
Maximum r2 score is  89.17088072102631  on Random_state  80
```

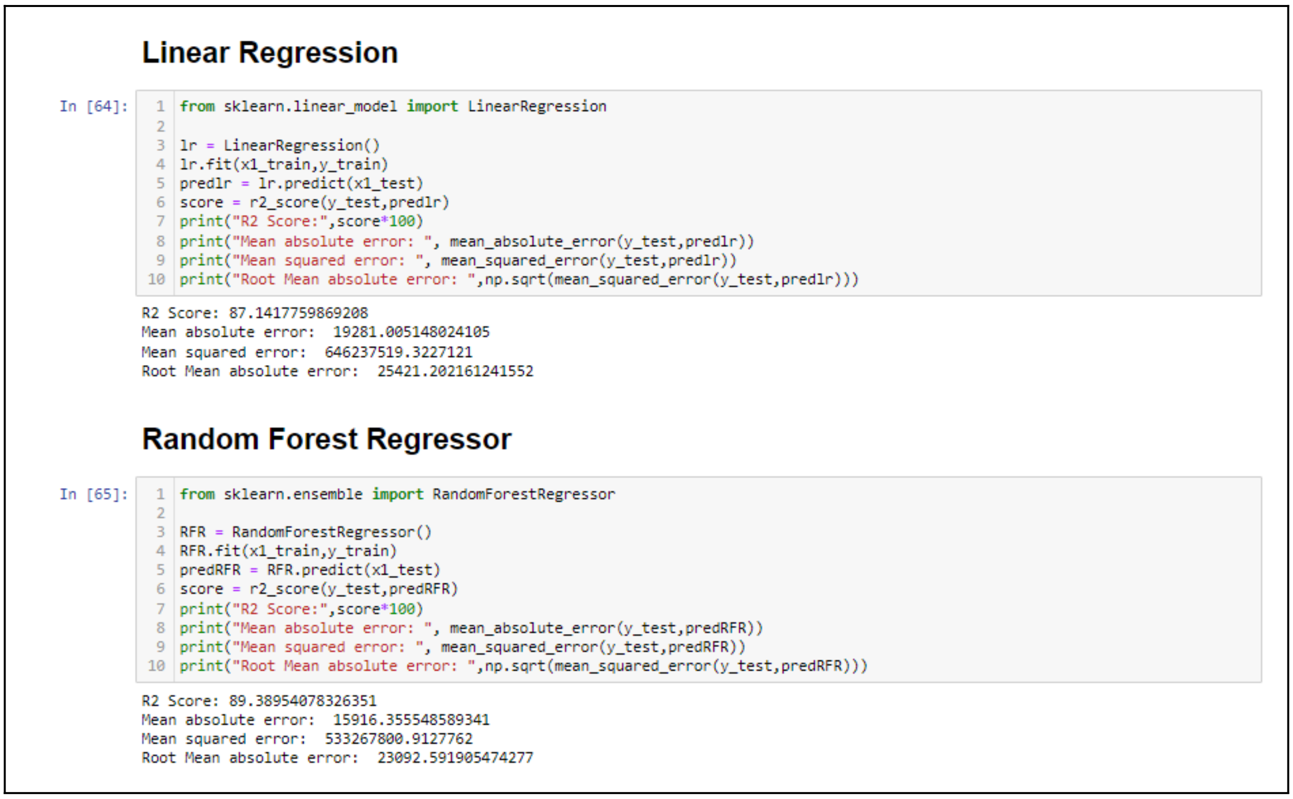
We have found our best random state value as 42 and will creat new train_test_split using this random state

Creating train_test_split

```
In [61]:
x1_train,x1_test,y_train,y_test = train_test_split(x1,y,test_size=.30, random_state=80)
```

```
In [62]:
x1_train.shape
Out[62]: (742, 59)
```

```
In [63]:
x1_test.shape
Out[63]: (319, 59)
```



Linear Regression

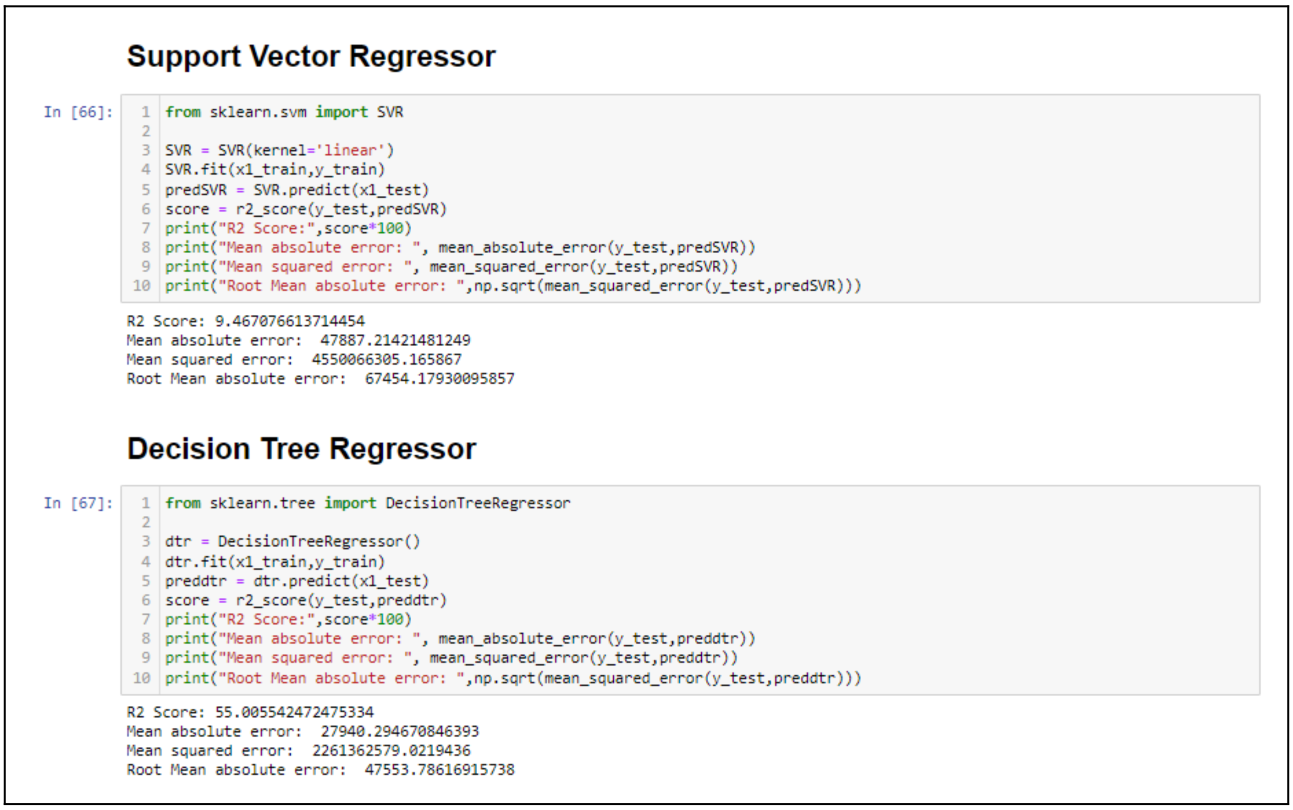
In [64]:
```python
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x1_train,y_train)
predlr = lr.predict(x1_test)
score = r2_score(y_test,predlr)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predlr))
print("Mean squared error: ", mean_squared_error(y_test,predlr))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predlr)))
```

```
R2 Score: 87.1417759869208
Mean absolute error:  19281.005148024105
Mean squared error:  646237519.3227121
Root Mean absolute error:  25421.202161241552
```

Random Forest Regressor

In [65]:
```python
from sklearn.ensemble import RandomForestRegressor

RFR = RandomForestRegressor()
RFR.fit(x1_train,y_train)
predRFR = RFR.predict(x1_test)
score = r2_score(y_test,predRFR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predRFR))
print("Mean squared error: ", mean_squared_error(y_test,predRFR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predRFR)))
```

```
R2 Score: 89.38954078326351
Mean absolute error:  15916.355548589341
Mean squared error:  533267800.9127762
Root Mean absolute error:  23092.591905474277
```



Support Vector Regressor

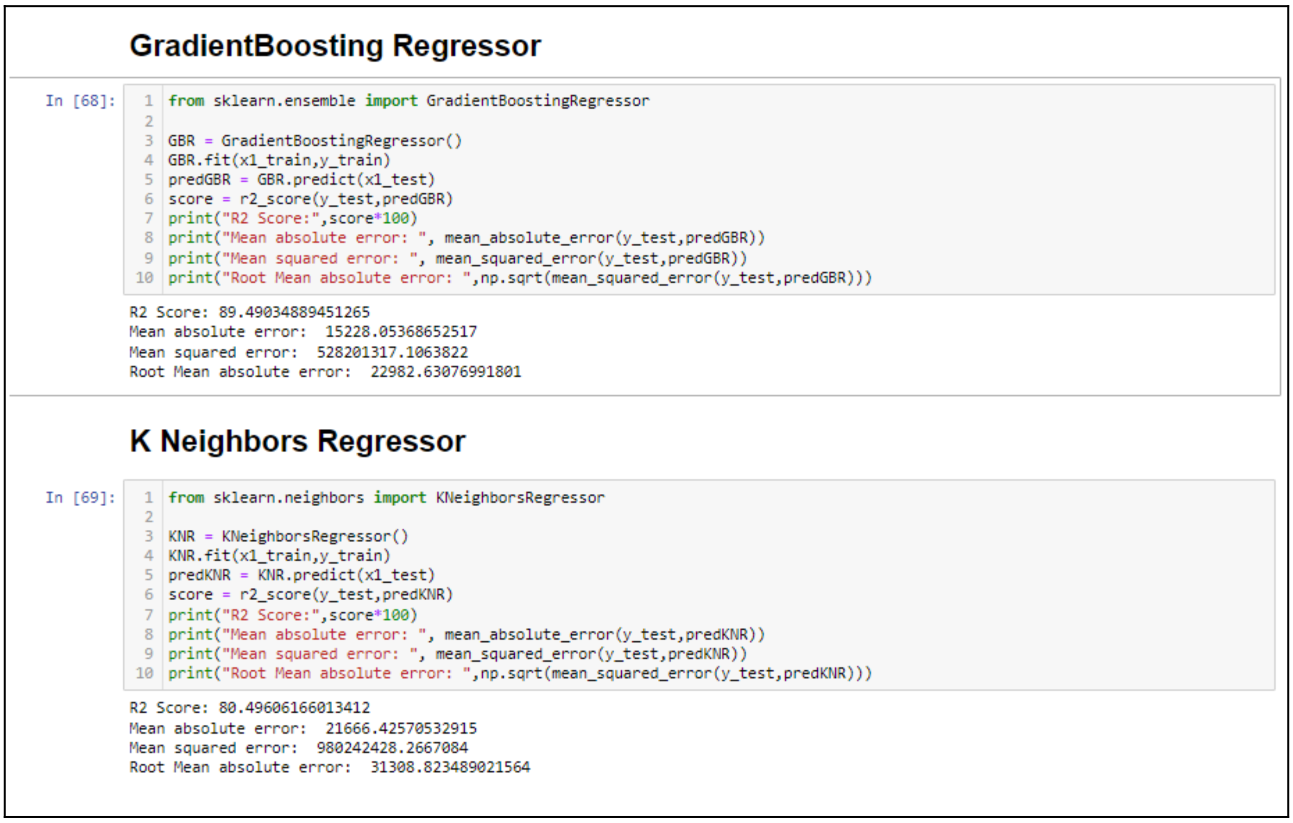
In [66]:
```python
from sklearn.svm import SVR

SVR = SVR(kernel='linear')
SVR.fit(x1_train,y_train)
predSVR = SVR.predict(x1_test)
score = r2_score(y_test,predSVR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predSVR))
print("Mean squared error: ", mean_squared_error(y_test,predSVR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predSVR)))
```

```
R2 Score: 9.467076613714454
Mean absolute error:  47887.21421481249
Mean squared error:  4550066305.165867
Root Mean absolute error:  67454.17930095857
```

Decision Tree Regressor

In [67]:
```python
from sklearn.tree import DecisionTreeRegressor

dtr = DecisionTreeRegressor()
dtr.fit(x1_train,y_train)
preddtr = dtr.predict(x1_test)
score = r2_score(y_test,preddtr)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,preddtr))
print("Mean squared error: ", mean_squared_error(y_test,preddtr))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,preddtr)))
```

```
R2 Score: 55.005542472475334
Mean absolute error:  27940.294670846393
Mean squared error:  2261362579.0219436
Root Mean absolute error:  47553.78616915738
```



GradientBoosting Regressor

In [68]:

```python
from sklearn.ensemble import GradientBoostingRegressor

GBR = GradientBoostingRegressor()
GBR.fit(x1_train,y_train)
predGBR = GBR.predict(x1_test)
score = r2_score(y_test,predGBR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predGBR))
print("Mean squared error: ", mean_squared_error(y_test,predGBR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predGBR)))
```

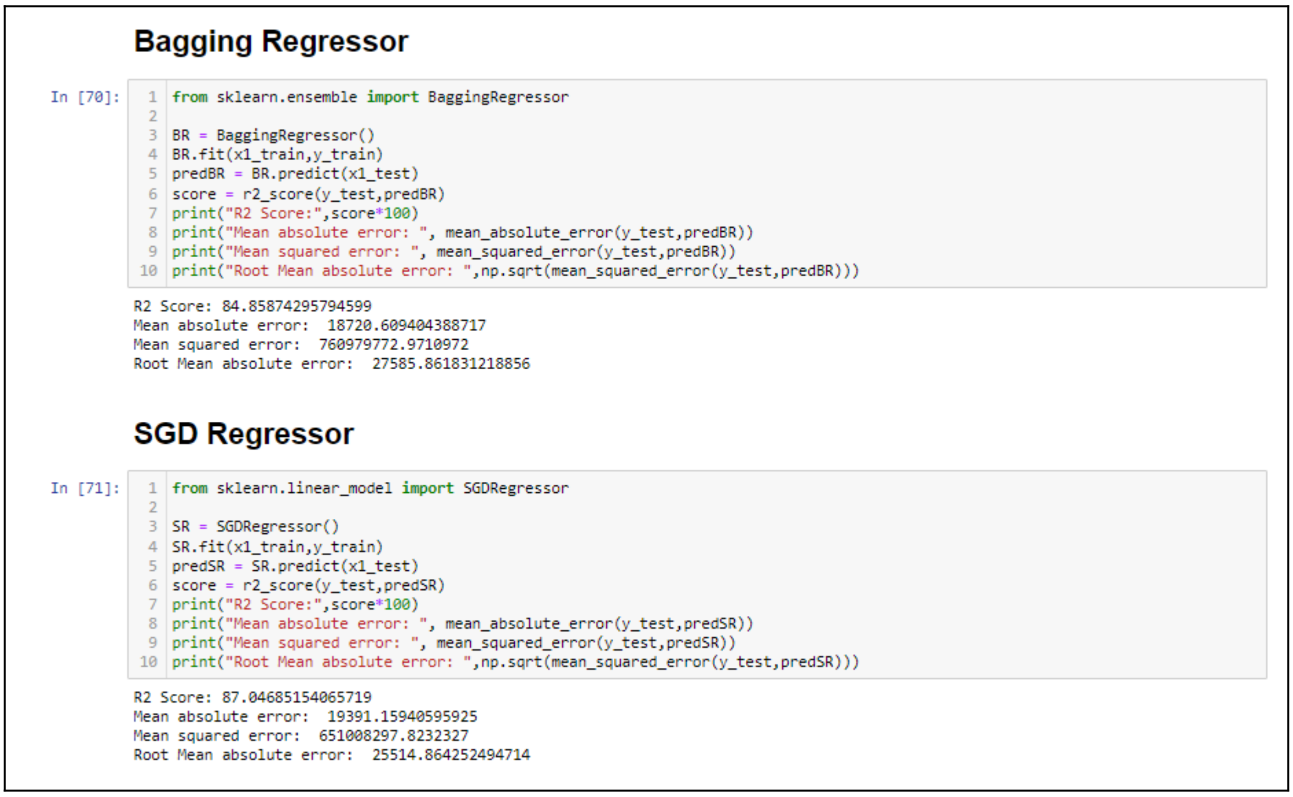
```
R2 Score: 89.49034889451265
Mean absolute error:  15228.05368652517
Mean squared error:  528201317.1063822
Root Mean absolute error:  22982.63076991801
```

K Neighbors Regressor

In [69]:

```python
from sklearn.neighbors import KNeighborsRegressor

KNR = KNeighborsRegressor()
KNR.fit(x1_train,y_train)
predKNR = KNR.predict(x1_test)
score = r2_score(y_test,predKNR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predKNR))
print("Mean squared error: ", mean_squared_error(y_test,predKNR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predKNR)))
```

```
R2 Score: 80.49606166013412
Mean absolute error:  21666.42570532915
Mean squared error:  980242428.2667084
Root Mean absolute error:  31308.823489021564
```



Bagging Regressor

In [70]:

```python
from sklearn.ensemble import BaggingRegressor

BR = BaggingRegressor()
BR.fit(x1_train,y_train)
predBR = BR.predict(x1_test)
score = r2_score(y_test,predBR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predBR))
print("Mean squared error: ", mean_squared_error(y_test,predBR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predBR)))
```

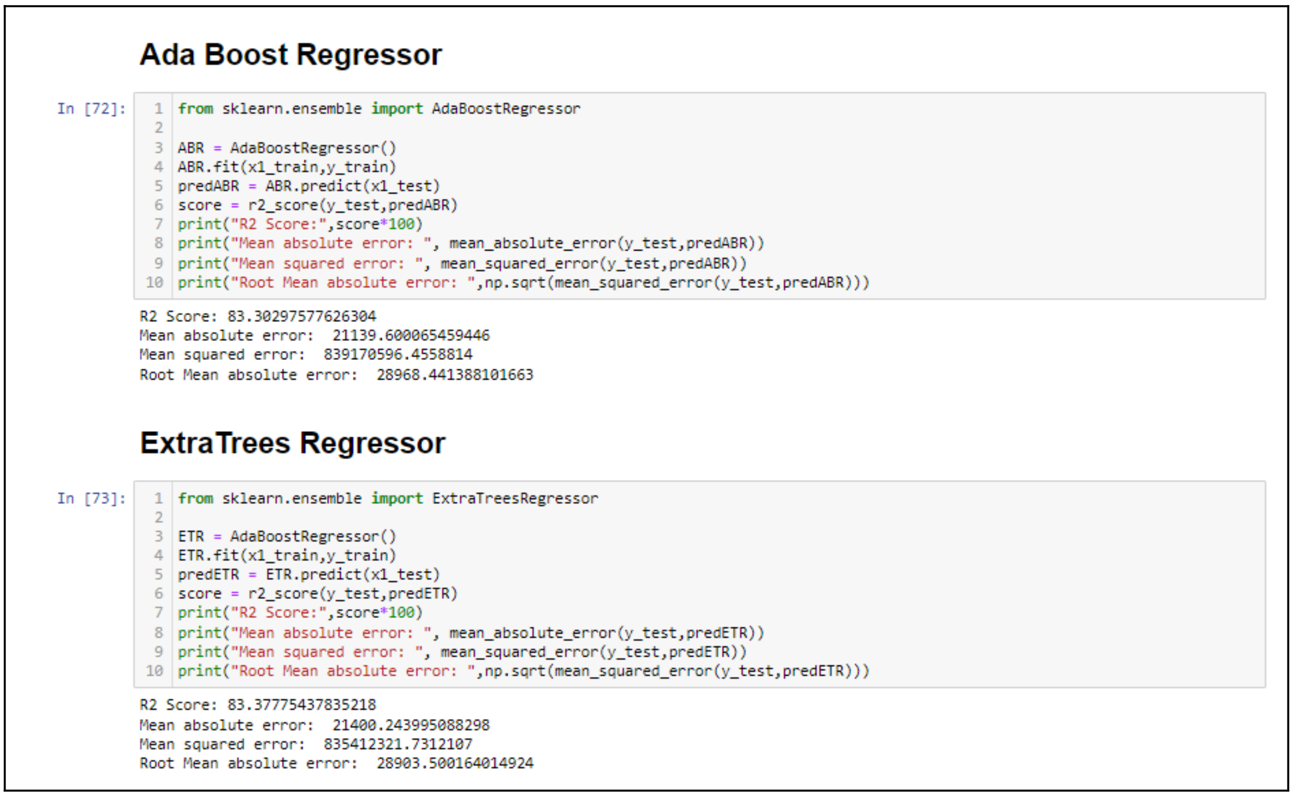
```
R2 Score: 84.85874295794599
Mean absolute error:  18720.609404388717
Mean squared error:  760979772.9710972
Root Mean absolute error:  27585.861831218856
```

SGD Regressor

In [71]:

```python
from sklearn.linear_model import SGDRegressor

SR = SGDRegressor()
SR.fit(x1_train,y_train)
predSR = SR.predict(x1_test)
score = r2_score(y_test,predSR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predSR))
print("Mean squared error: ", mean_squared_error(y_test,predSR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predSR)))
```

```
R2 Score: 87.04685154065719
Mean absolute error:  19391.15940595925
Mean squared error:  651008297.8232327
Root Mean absolute error:  25514.864252494714
```



Ada Boost Regressor

In [72]:

```python
from sklearn.ensemble import AdaBoostRegressor

ABR = AdaBoostRegressor()
ABR.fit(x1_train,y_train)
predABR = ABR.predict(x1_test)
score = r2_score(y_test,predABR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predABR))
print("Mean squared error: ", mean_squared_error(y_test,predABR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predABR)))
```

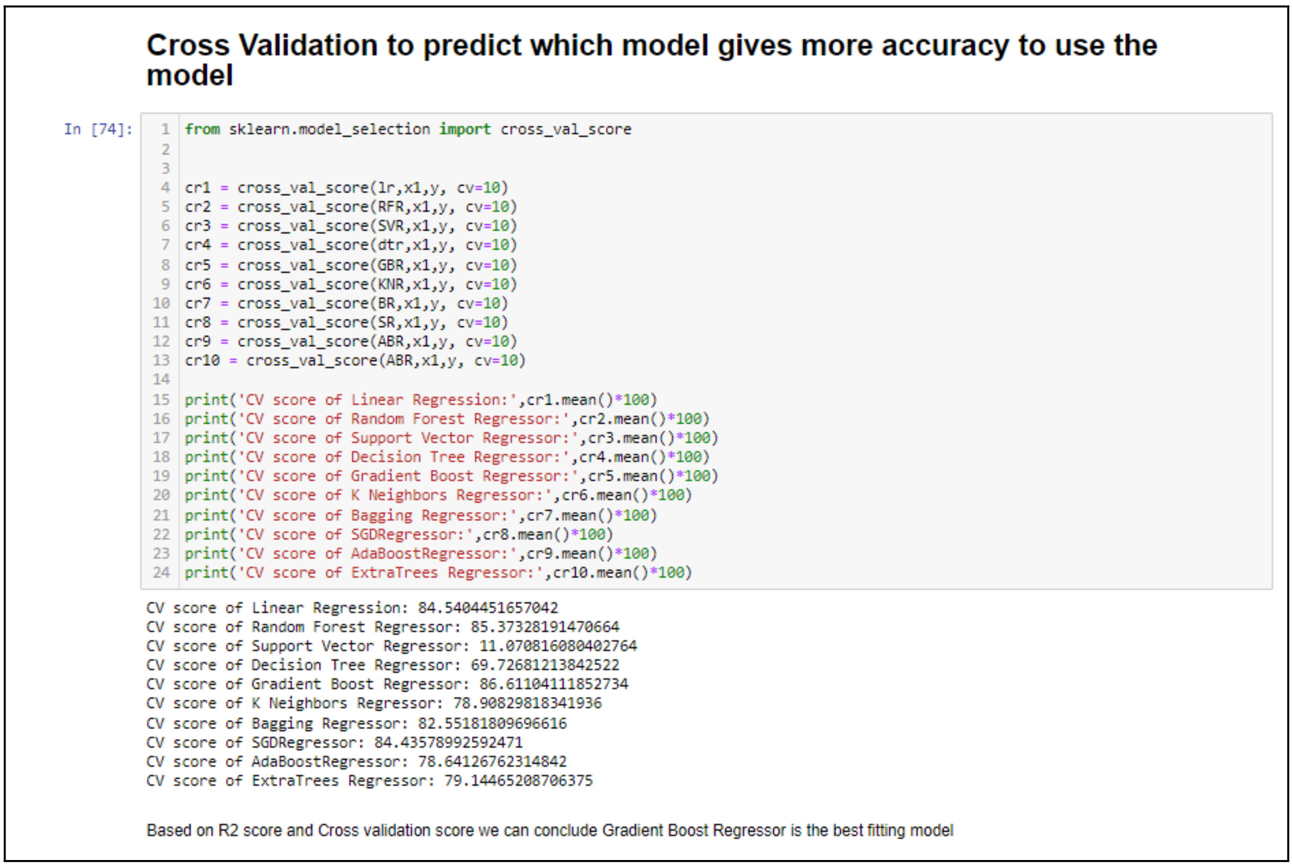
```
R2 Score: 83.30297577626304
Mean absolute error:  21139.600065459446
Mean squared error:  839170596.4558814
Root Mean absolute error:  28968.441388101663
```

ExtraTrees Regressor

In [73]:

```python
from sklearn.ensemble import ExtraTreesRegressor

ETR = AdaBoostRegressor()
ETR.fit(x1_train,y_train)
predETR = ETR.predict(x1_test)
score = r2_score(y_test,predETR)
print("R2 Score:",score*100)
print("Mean absolute error: ", mean_absolute_error(y_test,predETR))
print("Mean squared error: ", mean_squared_error(y_test,predETR))
print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,predETR)))
```

```
R2 Score: 83.37775437835218
Mean absolute error:  21400.243995088298
Mean squared error:  835412321.7312107
Root Mean absolute error:  28903.500164014924
```



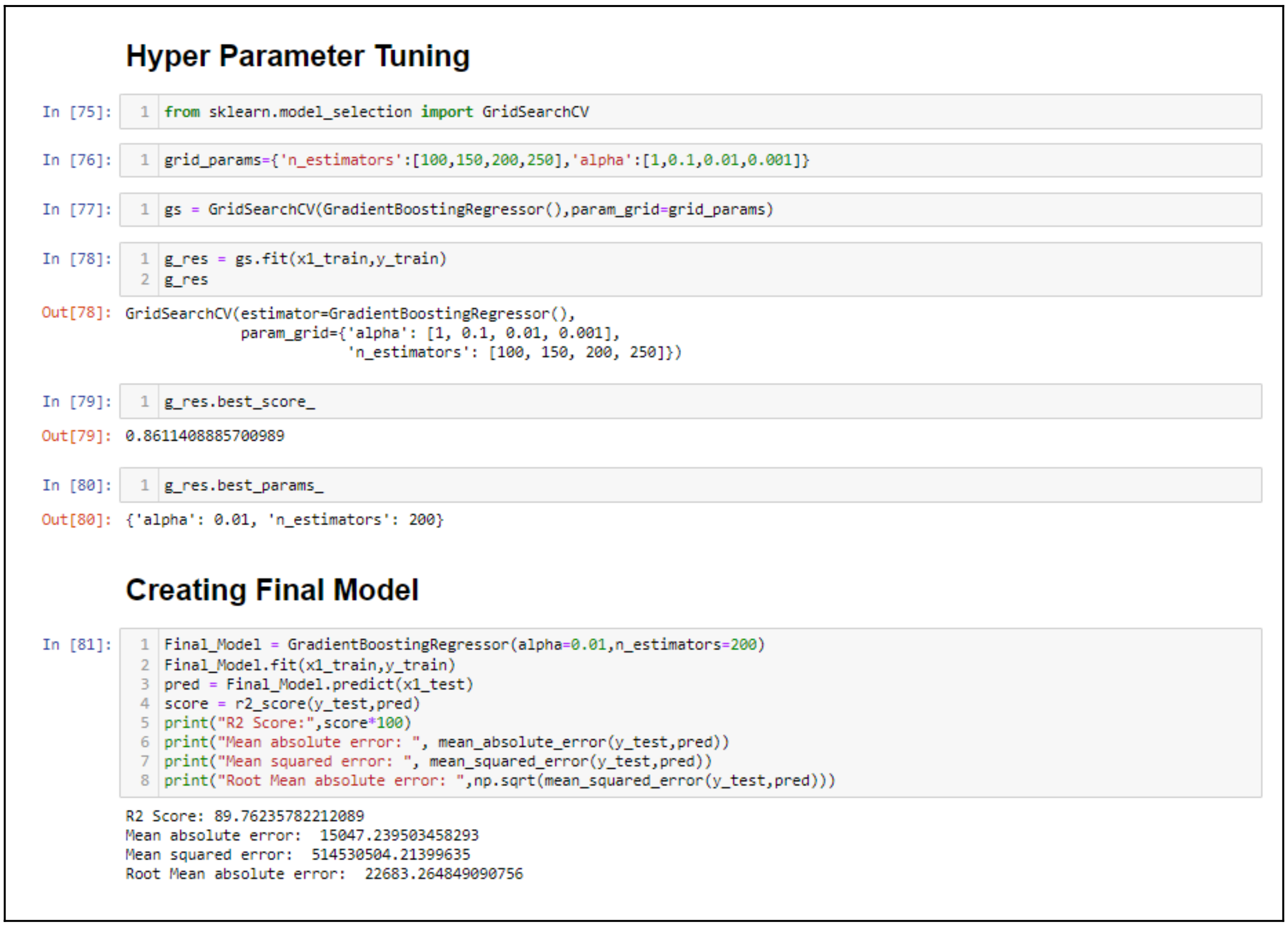
Cross Validation to predict which model gives more accuracy to use the model

In [74]:

```python
from sklearn.model_selection import cross_val_score


cr1 = cross_val_score(lr,x1,y, cv=10)
cr2 = cross_val_score(RFR,x1,y, cv=10)
cr3 = cross_val_score(SVR,x1,y, cv=10)
cr4 = cross_val_score(dtr,x1,y, cv=10)
cr5 = cross_val_score(GBR,x1,y, cv=10)
cr6 = cross_val_score(KNR,x1,y, cv=10)
cr7 = cross_val_score(BR,x1,y, cv=10)
cr8 = cross_val_score(SR,x1,y, cv=10)
cr9 = cross_val_score(ABR,x1,y, cv=10)
cr10 = cross_val_score(ABR,x1,y, cv=10)

print('CV score of Linear Regression:',cr1.mean()*100)
print('CV score of Random Forest Regressor:',cr2.mean()*100)
print('CV score of Support Vector Regressor:',cr3.mean()*100)
print('CV score of Decision Tree Regressor:',cr4.mean()*100)
print('CV score of Gradient Boost Regressor:',cr5.mean()*100)
print('CV score of K Neighbors Regressor:',cr6.mean()*100)
print('CV score of Bagging Regressor:',cr7.mean()*100)
print('CV score of SGDRegressor:',cr8.mean()*100)
print('CV score of AdaBoostRegressor:',cr9.mean()*100)
print('CV score of ExtraTrees Regressor:',cr10.mean()*100)
```

```
CV score of Linear Regression: 84.5404451657042
CV score of Random Forest Regressor: 85.37328191470664
CV score of Support Vector Regressor: 11.070816080402764
CV score of Decision Tree Regressor: 69.72681213842522
CV score of Gradient Boost Regressor: 86.61104111852734
CV score of K Neighbors Regressor: 78.90829818341936
CV score of Bagging Regressor: 82.55181809696616
CV score of SGDRegressor: 84.43578992592471
CV score of AdaBoostRegressor: 78.64126762314842
CV score of ExtraTrees Regressor: 79.14465208706375
```

Based on R2 score and Cross validation score we can conclude Gradient Boost Regressor is the best fitting model



Hyper Parameter Tuning

```
In [75]:  from sklearn.model_selection import GridSearchCV
```

```
In [76]:  grid_params={'n_estimators':[100,150,200,250],'alpha':[1,0.1,0.01,0.001]}
```

```
In [77]:  gs = GridSearchCV(GradientBoostingRegressor(),param_grid=grid_params)
```

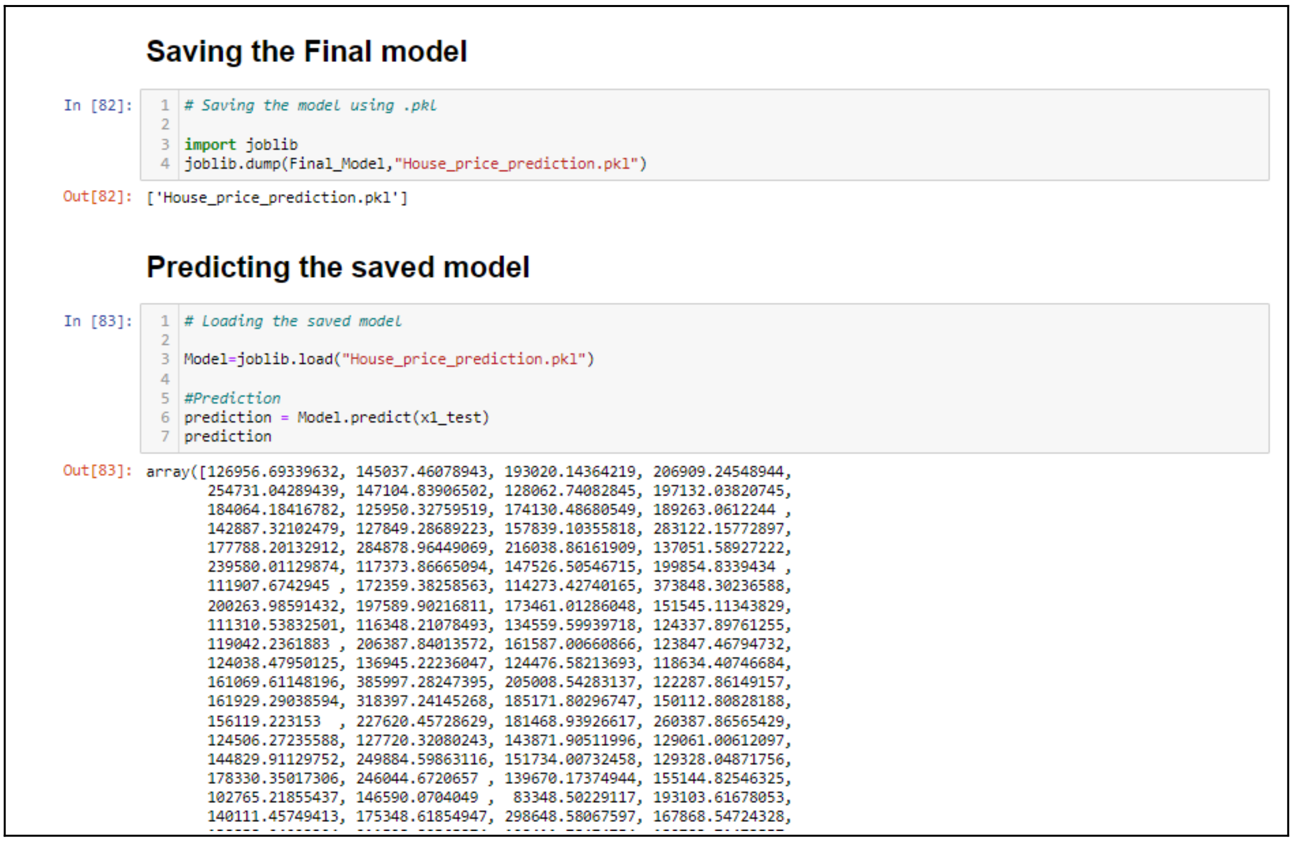
```
In [78]:  g_res = gs.fit(x1_train,y_train)
          g_res

Out[78]: GridSearchCV(estimator=GradientBoostingRegressor(),
                      param_grid={'alpha': [1, 0.1, 0.01, 0.001],
                                  'n_estimators': [100, 150, 200, 250]})
```

```
In [79]:  g_res.best_score_

Out[79]: 0.8611408885700989
```

```
In [80]:  g_res.best_params_

Out[80]: {'alpha': 0.01, 'n_estimators': 200}
```

Creating Final Model

```
In [81]:  Final_Model = GradientBoostingRegressor(alpha=0.01,n_estimators=200)
          Final_Model.fit(x1_train,y_train)
          pred = Final_Model.predict(x1_test)
          score = r2_score(y_test,pred)
          print("R2 Score:",score*100)
          print("Mean absolute error: ", mean_absolute_error(y_test,pred))
          print("Mean squared error: ", mean_squared_error(y_test,pred))
          print("Root Mean absolute error: ",np.sqrt(mean_squared_error(y_test,pred)))

         R2 Score: 89.76235782212089
         Mean absolute error:  15047.239503458293
         Mean squared error:  514530504.21399635
         Root Mean absolute error:  22683.264849090756
```

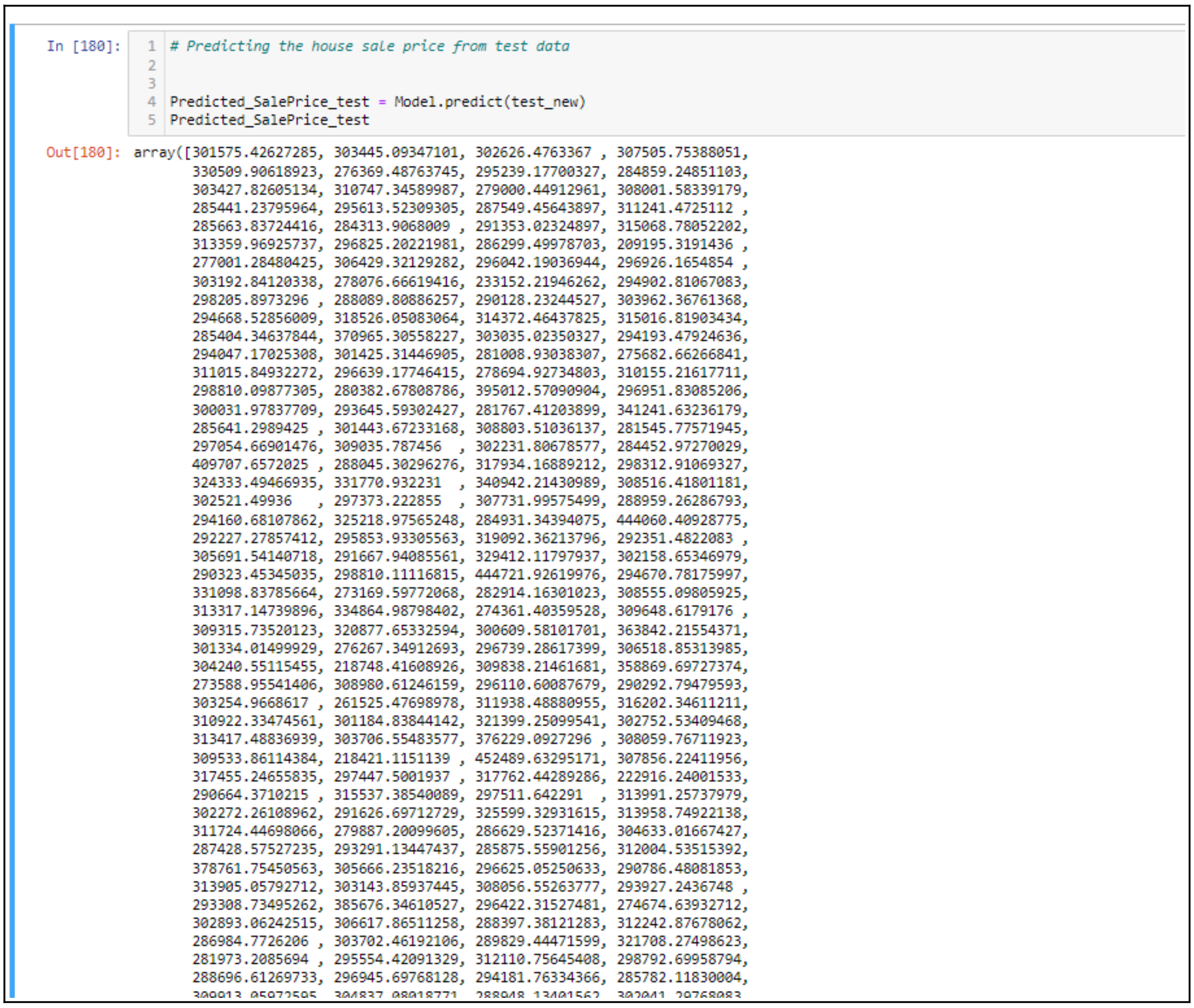


Saving the Final model

```
In [82]:  # Saving the model using .pkl

          import joblib
          joblib.dump(Final_Model,"House_price_prediction.pkl")

Out[82]: ['House_price_prediction.pkl']
```

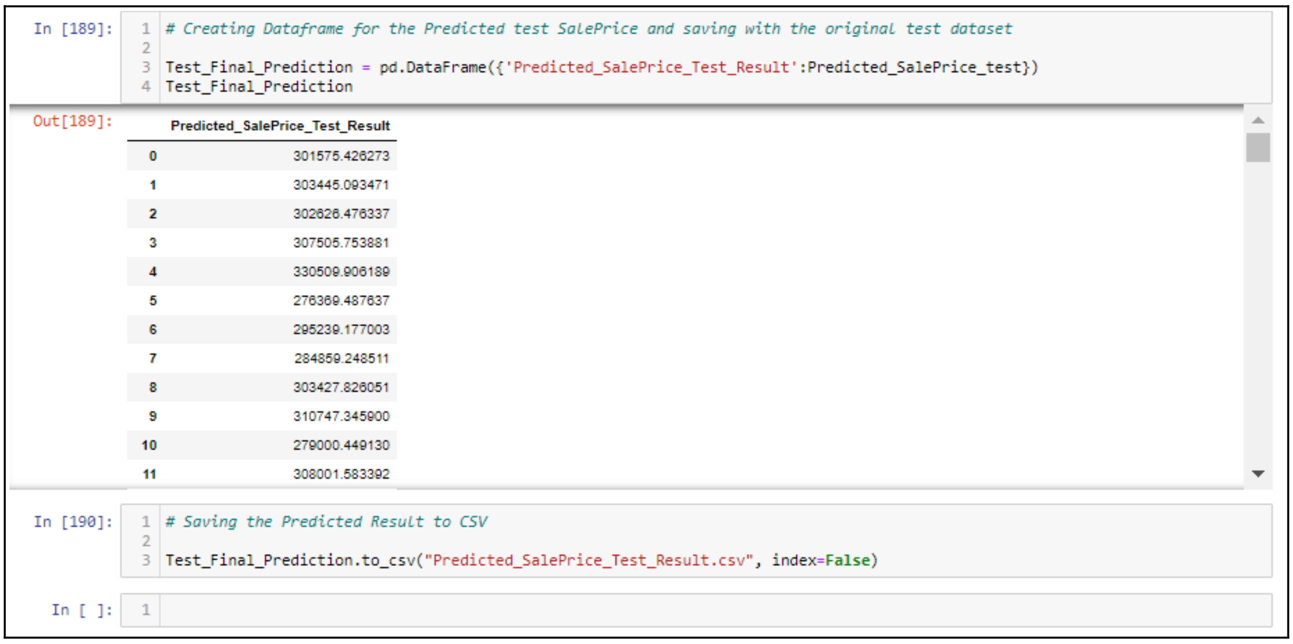
Predicting the saved model

```
In [83]:  # Loading the saved model

          Model=joblib.load("House_price_prediction.pkl")

          #Prediction
          prediction = Model.predict(x1_test)
          prediction

Out[83]: array([126956.69339632, 145037.46078943, 193020.14364219, 206909.24548944,
                254731.04289439, 147104.83906502, 128062.74082845, 197132.03820745,
                184064.18416782, 125950.32759519, 174130.48680549, 189263.0612244 ,
                142887.32102479, 127849.28689223, 157839.10355818, 283122.15772897,
                177788.20132912, 284878.96449069, 216038.86161909, 137051.58927222,
                239580.01129874, 117373.86665094, 147526.50546715, 199854.8339434 ,
                111907.6742945 , 172359.38258563, 114273.42740165, 373848.30236588,
                200263.98591432, 197589.90216811, 173461.01286048, 151545.11343829,
                111310.53832501, 116348.21078493, 134559.59939718, 124337.89761255,
                119042.2361883 , 206387.84013572, 161587.00660866, 123847.46794732,
                124038.47950125, 136945.22236047, 124476.58213693, 118634.40746684,
                161069.61148196, 385997.28247395, 205008.54283137, 122287.86149157,
                161929.29038594, 318397.24145268, 185171.80296747, 150112.80828188,
                156119.223153  , 227620.45728629, 181468.93926617, 260387.86565429,
                124506.27235588, 127720.32080243, 143871.90511996, 129061.00612097,
                144829.91129752, 249884.59863116, 151734.00732458, 129328.04871756,
                178330.35017306, 246044.6720657 , 139670.17374944, 155144.82546325,
                102765.21855437, 146590.0704049 ,  83348.50229117, 193103.61678053,
                140111.45749413, 175348.61854947, 298648.58067597, 167868.54724328,
```



```
In [180]:
# Predicting the house sale price from test data


Predicted_SalePrice_test = Model.predict(test_new)
Predicted_SalePrice_test
```

```
Out[180]: array([301575.42627285, 303445.09347101, 302626.4763367 , 307505.75388051,
                 330509.90618923, 276369.48763745, 295239.17700327, 284859.24851103,
                 303427.82605134, 310747.34589987, 279000.44912961, 308001.58339179,
                 285441.23795964, 295613.52309305, 287549.45643897, 311241.4725112 ,
                 285663.83724416, 284313.9068009 , 291353.02324897, 315068.78052202,
                 313359.96925737, 296825.20221981, 286299.49978703, 209195.3191436 ,
                 277001.28480425, 306429.32129282, 296042.19036944, 296926.1654854 ,
                 303192.84120338, 278076.66619416, 233152.21946262, 294902.81067083,
                 298205.8973296 , 288089.80886257, 290128.23244527, 303962.36761368,
                 294668.52856009, 318526.05083064, 314372.46437825, 315016.81903434,
                 285404.34637844, 370965.30558227, 303035.02350327, 294193.47924636,
                 294047.17025308, 301425.31446905, 281008.93038307, 275682.66266841,
                 311015.84932272, 296639.17746415, 278694.92734803, 310155.21617711,
                 298810.09877305, 280382.67808786, 395012.57090904, 296951.83085206,
                 300031.97837709, 293645.59302427, 281767.41203899, 341241.63236179,
                 285641.2989425 , 301443.67233168, 308803.51036137, 281545.77571945,
                 297054.66901476, 309035.787456  , 302231.80678577, 284452.97270029,
                 409707.6572025 , 288045.30296276, 317934.16889212, 298312.91069327,
                 324333.49466935, 331770.932231  , 340942.21430989, 308516.41801181,
                 302521.49936   , 297373.222855  , 307731.99575499, 288959.26286793,
                 294160.68107862, 325218.97565248, 284931.34394075, 444060.40928775,
                 292227.27857412, 295853.93305563, 319092.36213796, 292351.4822083 ,
                 305691.54140718, 291667.94085561, 329412.11797937, 302158.65346979,
                 290323.45345035, 298810.11116815, 444721.92619976, 294670.78175997,
                 331098.83785664, 273169.59772068, 282914.16301023, 308555.09805925,
                 313317.14739896, 334864.98798402, 274361.40359528, 309648.6179176 ,
                 309315.73520123, 320877.65332594, 300609.58101701, 363842.21554371,
                 301334.01499929, 276267.34912693, 296739.28617399, 306518.85313985,
                 304240.55115455, 218748.41608926, 309838.21461681, 358869.69727374,
                 273588.95541406, 308980.61246159, 296110.60087679, 290292.79479593,
                 303254.9668617 , 261525.47698978, 311938.48880955, 316202.34611211,
                 310922.33474561, 301184.83844142, 321399.25099541, 302752.53409468,
                 313417.48836939, 303706.55483577, 376229.0927296 , 308059.76711923,
                 309533.86114384, 218421.1151139 , 452489.63295171, 307856.22411956,
                 317455.24655835, 297447.5001937 , 317762.44289286, 222916.24001533,
                 290664.3710215 , 315537.38540089, 297511.642291  , 313991.25737979,
                 302272.26108962, 291626.69712729, 325599.32931615, 313958.74922138,
                 311724.44698066, 279887.20099605, 286629.52371416, 304633.01667427,
                 287428.57527235, 293291.13447437, 285875.55901256, 312004.53515392,
                 378761.75450563, 305666.23518216, 296625.05250633, 290786.48081853,
                 313905.05792712, 303143.85937445, 308056.55263777, 293927.2436748 ,
                 293308.73495262, 385676.34610527, 296422.31527481, 274674.63932712,
                 302893.06242515, 306617.86511258, 288397.38121283, 312242.87678062,
                 286984.7726206 , 303702.46192106, 289829.44471599, 321708.27498623,
                 281973.2085694 , 295554.42091329, 312110.75645408, 298792.69958794,
                 288696.61269733, 296945.69768128, 294181.76334366, 285782.11830004,
                 309913.05972595, 304837.08018771, 288948.13401562, 302041.29768083
```



```
In [189]:
# Creating Dataframe for the Predicted test SalePrice and saving with the original test dataset

Test_Final_Prediction = pd.DataFrame({'Predicted_SalePrice_Test_Result':Predicted_SalePrice_test})
Test_Final_Prediction
```

Out[189]:

	Predicted_SalePrice_Test_Result
0	301575.426273
1	303445.093471
2	302626.476337
3	307505.753881
4	330509.906189
5	276369.487637
6	295239.177003
7	284859.248511
8	303427.826051
9	310747.345900
10	279000.449130
11	308001.583392

```
In [190]:
# Saving the Predicted Result to CSV

Test_Final_Prediction.to_csv("Predicted_SalePrice_Test_Result.csv", index=False)
```

```
In [ ]:
```

CONCLUSION

Key Findings and Conclusions of the Study

In this project we have used the various model and analyse the process steps by steps. We have done the cleaning of data and feature engineering to make better prediction. We have the two data set for training and testing. We have analysed both the data set separately. We have built various models and chose the best model which gave the high score . We have done the cross validation of each model. Then we have finally predicted the Sale Price for test data and saved the file in csv format.

Overall Quality is the most contributing and highest positive impacting feature for prediction. The houses which have very excellent overall quality like material and finish of the house have high sale price. Also, we have observed from the plot that asthe overall quality of the house increases, the sale price also increases.

Learning Outcomes of the Study in respect of Data Science

Working on the housing project is very interesting, where I got the opportunity to explore the real estate domain. The graphical representation helped me to understand which features are important and how these features describe the sale price. Data cleaning was one of the important and crucial things in this project where I replaced all the null values with imputation methods and dealt with features having zero values and time variables

The developed model was facilitated the prediction of future housing prices. Particularly, the sellers and buyers of properties can benefit from this study and make better-informed decisions regarding the property evaluation.

The study can be enlarged in a subsequent research by increasing the dataset size so potentially uncovered details and features of the dataset.