FLIP ROBO

# Housing: Price Prediction

Submitted by:

Arpan Pattanayak

## ACKNOWLEDGMENT

I would like to express my sincere gratitude to my SME Mr. Shwetank Mishra from Flip Robo Technologies for letting me work on this project and providing me with all necessary information and dataset for the project. I would also like to thank my mentor of Data Trained academy for providing me sufficient knowledge so that I can complete the project.

I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

- https://www.google.com/
- https://www.youtube.com/
- https://scikit-learn.org/stable/user_guide.html
- https://github.com/
- https://www.analyticsvidhya.com/

# INTRODUCTION

## • Business Problem Framing

Houses are one of the necessary needs of each 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. The aim is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. By analysing previous market trends and price ranges, and upcoming developments future prices will be predicted cost of property depending on number of attributes considered. Now as a data scientist our work is to analyse the dataset and apply our skills towards predicting house price

## • Conceptual Background of the Domain Problem

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 to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:
1. Which variables are important to predict the price of a variable?

2. How do these variables describe the price of the house?

- Review of Literature

Based on the sample data provided to us from our client database where we have understood that the company is looking at prospective properties to buy houses to enter the market. The data set explains it is a regression problem as we need to build a model using Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the house.

- Motivation for the Problem Undertaken

We 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

We are building a model in Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. So, this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house. This will help to determine the price of houses with the available independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables (features). The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For specific mathematical reasons this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values. Regression analysis is also a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (feature). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

## • Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of data is 1168 rows and 81 columns. There are 2 data sets that are given. One is training data and one is testing data.
1) Train file will be used for training the model, i.e., the model will learn from this file. It contains all the independent variables and the target variable. Size of training set: 1168 records.
2) Test file contains all the independent variables, but not the target variable. We will apply the model to predict the target variable for the test data. Size of test set: 292 records.

## • Data Preprocessing Done

a. I have imported the necessary libraries and imported both train and test datasets which were in csv format.
b. I have done some statistical analysis like checking shape, nunique, column names, data types of the features, info about the features, value counts etc for both train and test data.

c. I have dropped the columns “Id” and “Utilities” from both the datasets. Since Id is the unique identifier which contains unique value throughout the data also all the entries in Utilities column were unique. They had no significance impact on the prediction.

d. While looking into the value count function I found some of the columns having more than 85% of zero values so, I dropped those columns from both the datasets as they might create skewness which will impact my model.

e. I checked the null values and found them in some of the columns. So, I imputed null values present in categorical and numerical columns using mode and mean methods respectively. I found some columns having more than 80% of null values so, I dropped those columns to overcome with the skewness.

f. Described statistical summary of both train and test datasets.

g. Visualized each feature using seaborn and matplotlib libraries by plotting several categorical and numerical plots.

h. Identified outliers using box plots in both datasets. I tried to remove them using both Zscore and IQR method and got huge data loss of around 24% and 48% respectively, so removed outliers using percentile method by setting data loss to 2%.

i. Checked for skewness and removed skewness in numerical columns using power transformation method (yeo-johnson). Also dropped KitchenAbvGr columns as it contains zero values throughout the data after using power transformation.

j. Encoded both train and test data frames using Ordinal Encoder. Also replaced some categorical columns having ratings by numbers based on specific condition.
k. Used Pearson’s correlation coefficient to check the correlation between label and features.

l. While checking the correlation I came across multicollinearity problem, I checked VIF values and removed GrLivArea to overcome with the multicollinearity issue.

m. Scaled both the datasets using Standard Scalar method and used regression algorithms to build ML models.

## • Data Inputs- Logic- Output Relationships

When we loaded the training dataset, we had to go through various data pre-processing steps to understand what was given to us and what we were expected to predict for the project. When it comes to logical part the domain expertise of understanding how real estate works and how we are supposed to cater to the customers came in handy to train the model with the modified input data. With the objective of predicting hosing sale prices accurately we had to make sure that a model was built that understood the customer priorities trending in the market imposing those norms when a relevant price tag was generated. I tried my best to retain as much data possible that was collected but I feel discarding columns that had lots of missing data was good

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

The assumption part for me was relying strictly on the data provided to me and taking into consideration that the separate training and testing datasets were obtained from real people surveyed for their preferences and how reasonable a price for a house with various features inclining to them were.

- Hardware and Software Requirements and Tools Used

Hardware Used:

1. Processor — core i5 and above
2. RAM — 8 GB or above
3. SSD — 250GB or above

Software Used:

i. Programming language: Python
ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook
Libraries/Packages Used:
Pandas, NumPy, matplotlib, seaborn, scikit-learn and pandas_profiling

## Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

I. I have used imputation methods to treat the null values.

II. Used percentile method to remove outliers.

III. Removed skewness using power transformation (yeo-johnson) method.

IV. Encoded the object type data into numerical using Ordinal Encoder.

V. I have used Pearson's correlation coefficient method to check the
VI. correlation between the dependent and independent variables.
VII. I have scaled the data using Standard Scalar method to overcome with the data biasness.

VIII. Used many Machine Learning models to predict the sale price of the house.

- ## Testing of Identified Approaches (Algorithms)

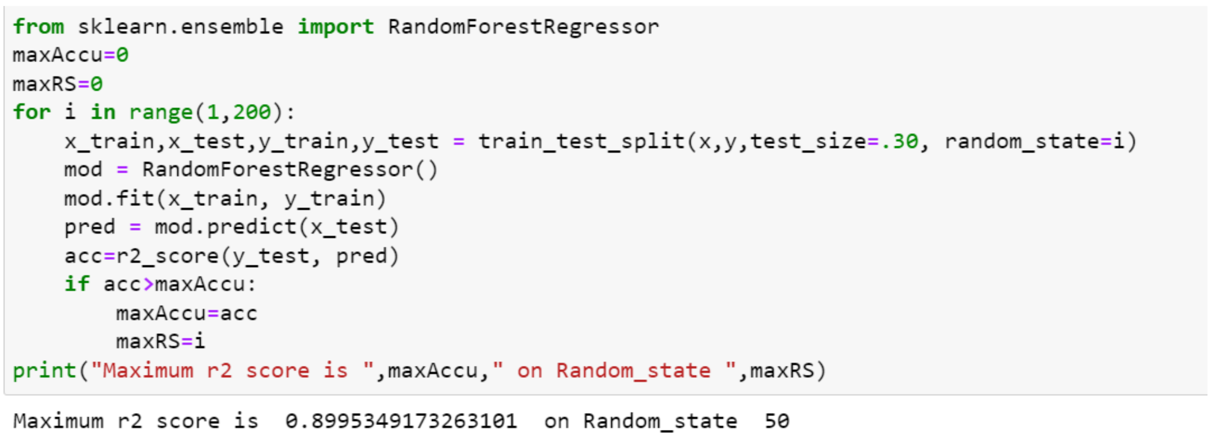
I have used all regression algorithms to build my model. By looking into the difference of r2 score and cross validation score, I found ExtraTreesRegressor as a best model with least difference. Also, to get the best model we have to run through multiple models and to avoid the confusion of overfitting we have go through cross validation. Below is the list of regression algorithms I have used in my project.

1. Linear Regression Model
2. Lasso Regularization Regression Model
3. Ridge Regularization Regression Model
4. Decision Tree Regression Model
5. Random Forest Regression Model
6. Extra Trees Regression Model
7. Gradient Boosting Regression Model
8. Bagging Regression Model
9. Extreme Gradient Boosting Regressor (XGB)

- ## Run and Evaluate selected models

I used a total of 9 Regression Models after choosing the random state amongst 1-200 number. Then I have defined a function for getting the regression model trained and evaluated.
The code for the models is listed below.

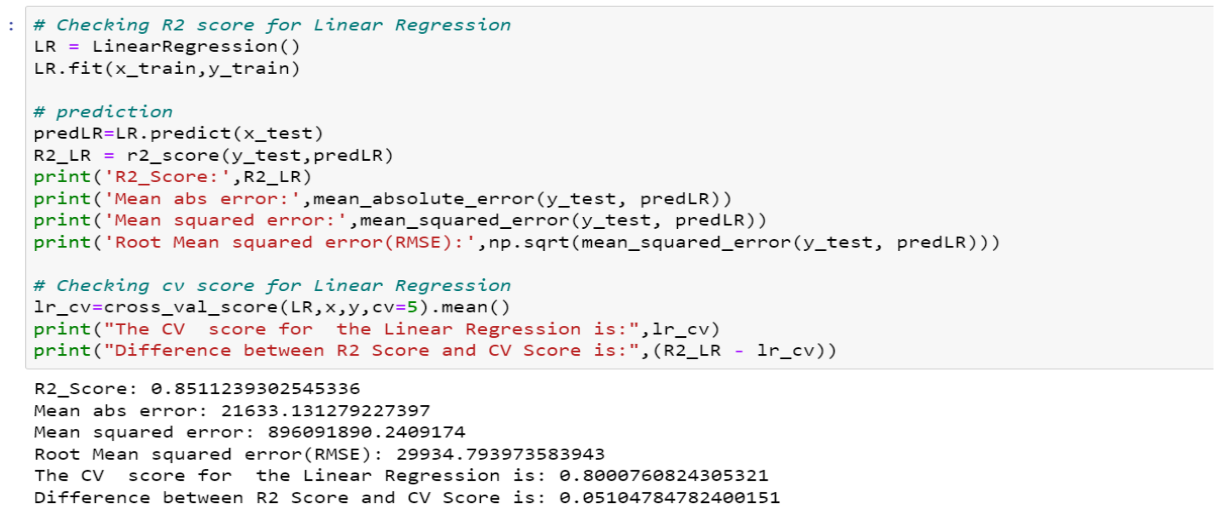
## Random State:



```
from sklearn.ensemble import RandomForestRegressor
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state=i)
    mod = RandomForestRegressor()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Maximum r2 score is ",maxAccu," on Random_state ",maxRS)
```

```
Maximum r2 score is  0.8995349173263101  on Random_state  50
```

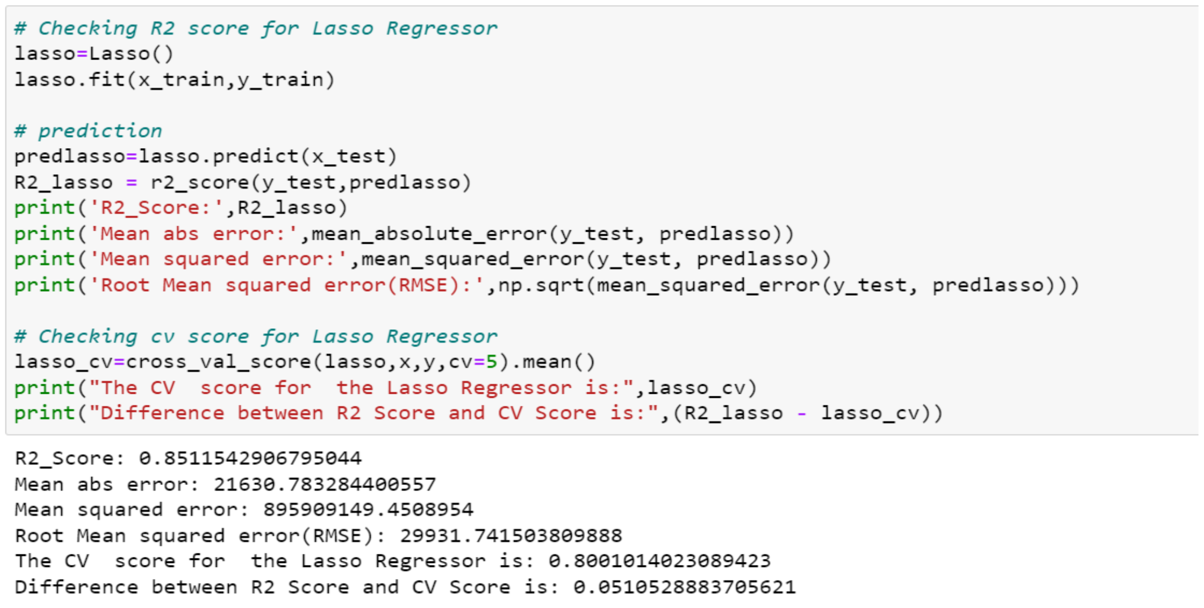
### Linear Regression



```
# Checking R2 score for Linear Regression
LR = LinearRegression()
LR.fit(x_train,y_train)

# prediction
predLR=LR.predict(x_test)
R2_LR = r2_score(y_test,predLR)
print('R2_Score:',R2_LR)
print('Mean abs error:',mean_absolute_error(y_test, predLR))
print('Mean squared error:',mean_squared_error(y_test, predLR))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predLR)))

# Checking cv score for Linear Regression
lr_cv=cross_val_score(LR,x,y,cv=5).mean()
print("The CV  score for  the Linear Regression is:",lr_cv)
print("Difference between R2 Score and CV Score is:",(R2_LR - lr_cv))
```

```
R2_Score: 0.8511239302545336
Mean abs error: 21633.131279227397
Mean squared error: 896091890.2409174
Root Mean squared error(RMSE): 29934.793973583943
The CV  score for  the Linear Regression is: 0.8007608243O5321
Difference between R2 Score and CV Score is: 0.05104784782400151
```

### Lasso Regressor



```
# Checking R2 score for Lasso Regressor
lasso=Lasso()
lasso.fit(x_train,y_train)

# prediction
predlasso=lasso.predict(x_test)
R2_lasso = r2_score(y_test,predlasso)
print('R2_Score:',R2_lasso)
print('Mean abs error:',mean_absolute_error(y_test, predlasso))
print('Mean squared error:',mean_squared_error(y_test, predlasso))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predlasso)))

# Checking cv score for Lasso Regressor
lasso_cv=cross_val_score(lasso,x,y,cv=5).mean()
print("The CV  score for  the Lasso Regressor is:",lasso_cv)
print("Difference between R2 Score and CV Score is:",(R2_lasso - lasso_cv))
```

```
R2_Score: 0.8511542906795044
Mean abs error: 21630.783284400557
Mean squared error: 895909149.4508954
Root Mean squared error(RMSE): 29931.741503809888
The CV  score for  the Lasso Regressor is: 0.8001014023089423
Difference between R2 Score and CV Score is: 0.0510528883705621
```

## Ridge Regressor

```
# Checking R2 score for Ridge Regressor
ridge=Ridge(alpha=100)
ridge.fit(x_train,y_train)
ridge.score(x_train,y_train)

# prediction
predridge=ridge.predict(x_test)
R2_ridge = r2_score(y_test,predridge)
print('R2_Score:',R2_ridge)
print('Mean abs error:',mean_absolute_error(y_test, predridge))
print('Mean squared error:',mean_squared_error(y_test, predridge))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predridge)))

# Checking cv score for Ridge Regressor
ridge_cv=cross_val_score(ridge,x,y,cv=5).mean()
print("The CV  score for  the Ridge Regressor is:",ridge_cv)
print("Difference between R2 Score and CV Score is:",(R2_ridge - ridge_cv))
```

```
R2_Score: 0.8547640849422966
Mean abs error: 21005.556691150156
Mean squared error: 874181632.2625612
Root Mean squared error(RMSE): 29566.56274007111
The CV  score for  the Ridge Regressor is: 0.8045866288464779
Difference between R2 Score and CV Score is: 0.050177456095818695
```

## Decision Tree Regressor

```
# Checking R2 score for Decision Tree Regressor
DTR=DecisionTreeRegressor()
DTR.fit(x_train,y_train)

# prediction
predDTR=DTR.predict(x_test)
R2_DTR = r2_score(y_test,predDTR)
print('R2_Score:',R2_DTR)
print('Mean abs error:',mean_absolute_error(y_test, predDTR))
print('Mean squared error:',mean_squared_error(y_test, predDTR))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predDTR)))

# Checking cv score for Decision Tree Regression
DTR_cv=cross_val_score(DTR,x,y,cv=5).mean()
print('The CV score  for  the Decision Tree Regression is :',DTR_cv)
print("Difference between R2 Score and CV Score is:",(R2_DTR - DTR_cv))
```

```
R2_Score: 0.7452542499373536
Mean abs error: 27661.005698005698
Mean squared error: 1533326350.5327635
Root Mean squared error(RMSE): 39157.711252482106
The CV score  for  the Decision Tree Regression is : 0.6543564856501043
Difference between R2 Score and CV Score is: 0.09089776428724927
```

## Random Forest Regressor

```
# Checking R2 score for Random Forest Regressor
RFR=RandomForestRegressor()
RFR.fit(x_train,y_train)

# prediction
predRFR=RFR.predict(x_test)
R2_RFR = r2_score(y_test,predRFR)
print('R2_Score:',R2_RFR)
print('Mean abs error:',mean_absolute_error(y_test, predRFR))
print('Mean squared error:',mean_squared_error(y_test, predRFR))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predRFR)))

# Checking cv score for Random Forest Regression
RFR_cv=cross_val_score(RFR,x,y,cv=5).mean()
print("The CV  score for  the Random forest regressor is:",RFR_cv)
print("Difference between R2 Score and CV Score is:",(R2_RFR - RFR_cv))
```

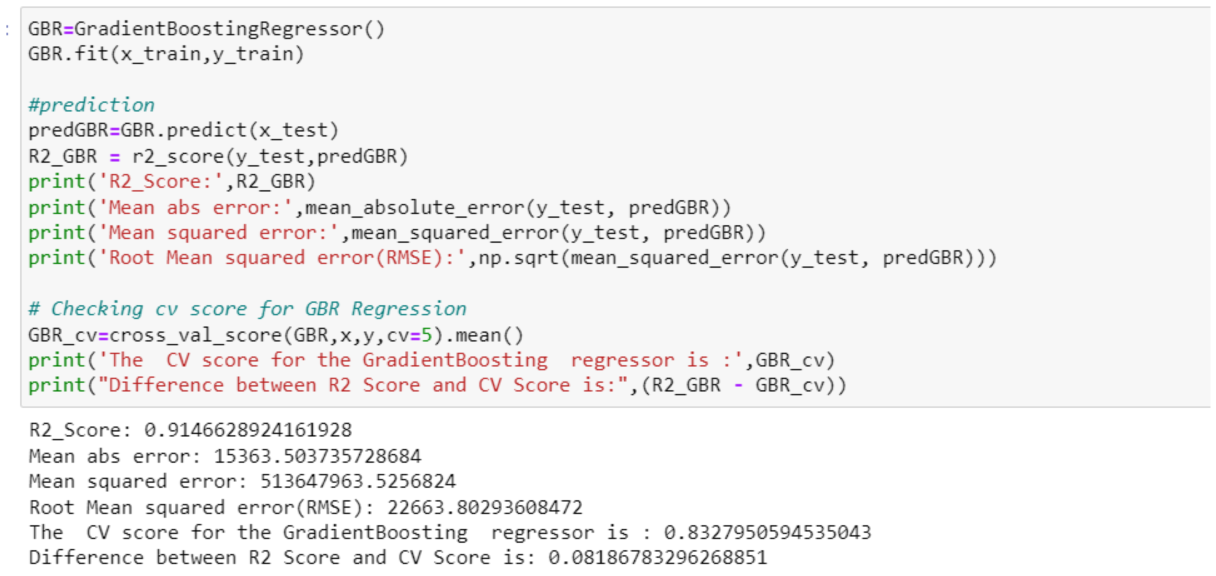
```
R2_Score: 0.9040005010427254
Mean abs error: 16145.478404558407
Mean squared error: 577825386.1072576
Root Mean squared error(RMSE): 24037.99879580781
The CV  score for  the Random forest regressor is: 0.8332620470036881
Difference between R2 Score and CV Score is: 0.07073845403903734
```

## ExtraTrees Regressor

```
XT=ExtraTreesRegressor()
XT.fit(x_train,y_train)

#prediction
predXT=XT.predict(x_test)
R2_XT = r2_score(y_test,predXT)
print('R2_Score:',R2_XT)
print('Mean abs error:',mean_absolute_error(y_test, predXT))
print('Mean squared error:',mean_squared_error(y_test, predXT))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predXT)))

# Checking cv score for Extra Tree Regression
XT_cv=cross_val_score(XT,x,y,cv=5).mean()
print('The  CV score for the Extra Tree  regressor is :',XT_cv)
print("Difference between R2 Score and CV Score is:",(R2_XT - XT_cv))
```

```
R2_Score: 0.8889027824966 71
Mean abs error: 16988.74641025641
Mean squared error: 668699246.314541
Root Mean squared error(RMSE): 25859.219754558355
The  CV score for the Extra Tree  regressor is : 0.8416485754869332
Difference between R2 Score and CV Score is: 0.047254207009737814
```

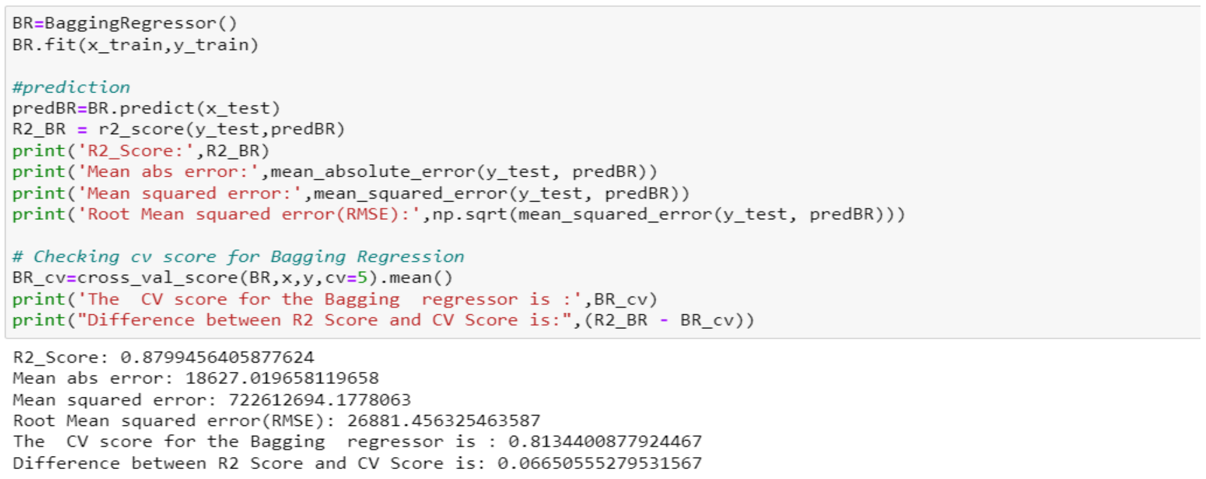
## Gradient Boosting Regressor



```
GBR=GradientBoostingRegressor()
GBR.fit(x_train,y_train)

#prediction
predGBR=GBR.predict(x_test)
R2_GBR = r2_score(y_test,predGBR)
print('R2_Score:',R2_GBR)
print('Mean abs error:',mean_absolute_error(y_test, predGBR))
print('Mean squared error:',mean_squared_error(y_test, predGBR))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predGBR)))

# Checking cv score for GBR Regression
GBR_cv=cross_val_score(GBR,x,y,cv=5).mean()
print('The  CV score for the GradientBoosting  regressor is :',GBR_cv)
print("Difference between R2 Score and CV Score is:",(R2_GBR - GBR_cv))
```

```
R2_Score: 0.9146628924161928
Mean abs error: 15363.503735728684
Mean squared error: 513647963.5256824
Root Mean squared error(RMSE): 22663.80293608472
The  CV score for the GradientBoosting  regressor is : 0.8327950594535043
Difference between R2 Score and CV Score is: 0.08186783296268851
```

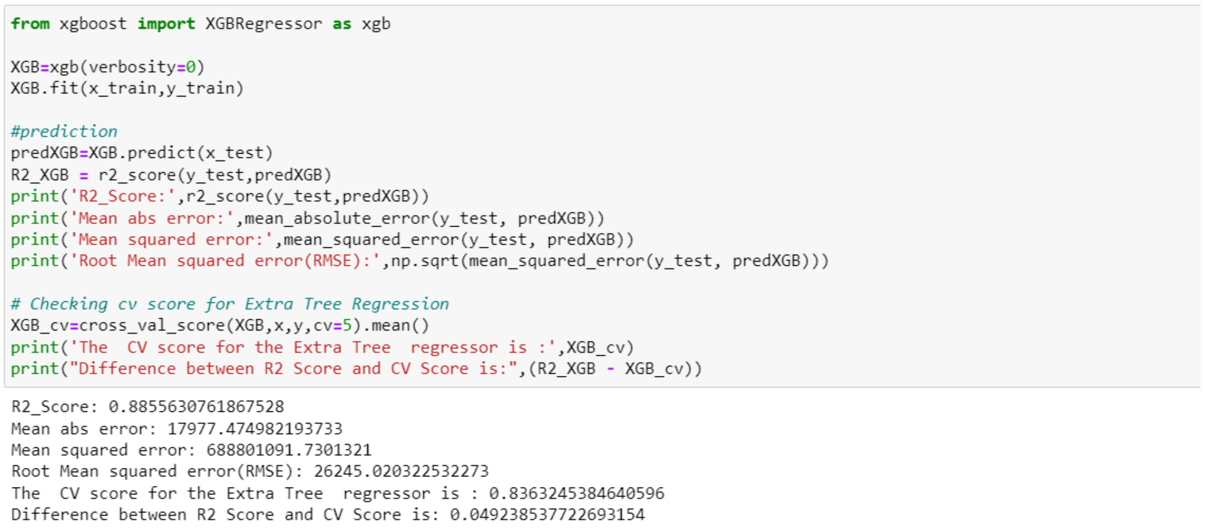
## Bagging Regressor



```
BR=BaggingRegressor()
BR.fit(x_train,y_train)

#prediction
predBR=BR.predict(x_test)
R2_BR = r2_score(y_test,predBR)
print('R2_Score:',R2_BR)
print('Mean abs error:',mean_absolute_error(y_test, predBR))
print('Mean squared error:',mean_squared_error(y_test, predBR))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predBR)))

# Checking cv score for Bagging Regression
BR_cv=cross_val_score(BR,x,y,cv=5).mean()
print('The  CV score for the Bagging  regressor is :',BR_cv)
print("Difference between R2 Score and CV Score is:",(R2_BR - BR_cv))
```

```
R2_Score: 0.8799456405877624
Mean abs error: 18627.019658119658
Mean squared error: 722612694.1778063
Root Mean squared error(RMSE): 26881.456325463587
The  CV score for the Bagging  regressor is : 0.8134400877924467
Difference between R2 Score and CV Score is: 0.06650555279531567
```

## Extreme Gradient Boosting Regressor (XGB)



```
from xgboost import XGBRegressor as xgb

XGB=xgb(verbosity=0)
XGB.fit(x_train,y_train)

#prediction
predXGB=XGB.predict(x_test)
R2_XGB = r2_score(y_test,predXGB)
print('R2_Score:',r2_score(y_test,predXGB))
print('Mean abs error:',mean_absolute_error(y_test, predXGB))
print('Mean squared error:',mean_squared_error(y_test, predXGB))
print('Root Mean squared error(RMSE):',np.sqrt(mean_squared_error(y_test, predXGB)))

# Checking cv score for Extra Tree Regression
XGB_cv=cross_val_score(XGB,x,y,cv=5).mean()
print('The  CV score for the Extra Tree  regressor is :',XGB_cv)
print("Difference between R2 Score and CV Score is:",(R2_XGB - XGB_cv))
```

```
R2_Score: 0.8855630761867528
Mean abs error: 17977.474982193733
Mean squared error: 688801091.7301321
Root Mean squared error(RMSE): 26245.020322532273
The  CV score for the Extra Tree  regressor is : 0.8363245384640596
Difference between R2 Score and CV Score is: 0.049238537722693154
```

## • Key Metrics for success in solving problem under Consideration

The key metrics used here were r2_score, Cross Validation Score, MAE, MSE and RMSE. We tried to find out the best parameters and to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:
Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the

2. R2 Score:
It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE):
MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE):
MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

5. Hyperparameter Tuning:
There is a list of different machine learning models. They all are

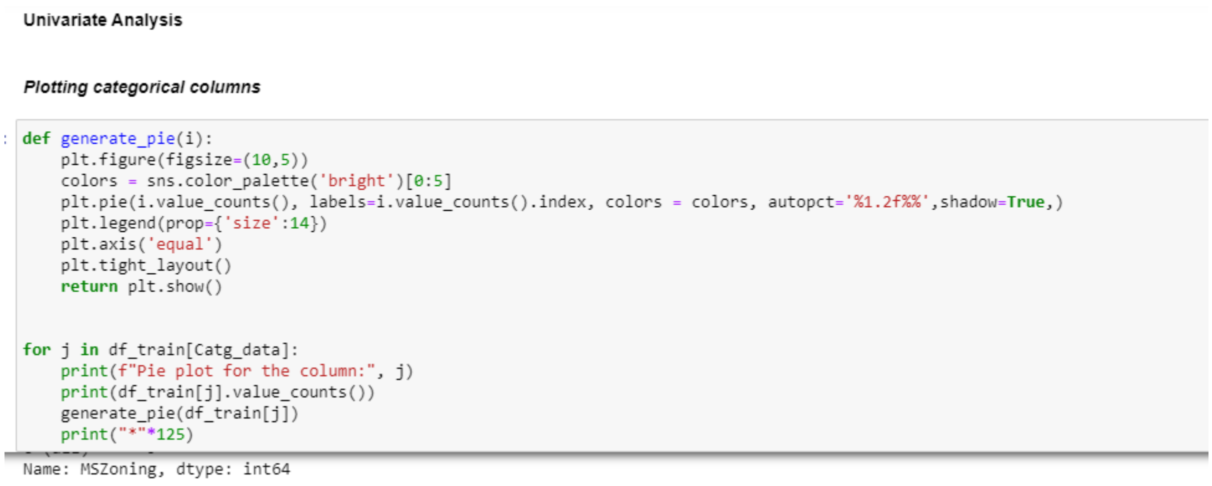
different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.
We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically.

6.GridSearchCV:

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.
It is possible that there are times when the default parameters perform better than the parameters list obtained from the tuning and it only indicates that there are more permutations and combinations that one needs to go through for obtaining better results.

## • Visualizations

I have analysed the data using both univariate and bivariate analysis to visualize the data. In univariate analysis I have used pie plots, count plots and distribution plot and in bivariate analysis I have used bar plots for categorical columns and used reg plots, scatter plots, strip plots, swarm plots, violin plots and line plot to visualize numerical columns. These plots have given good pattern. Here I will be showing few plots from univariate and bivariate analysis to get the better insights of relation between label and the features.

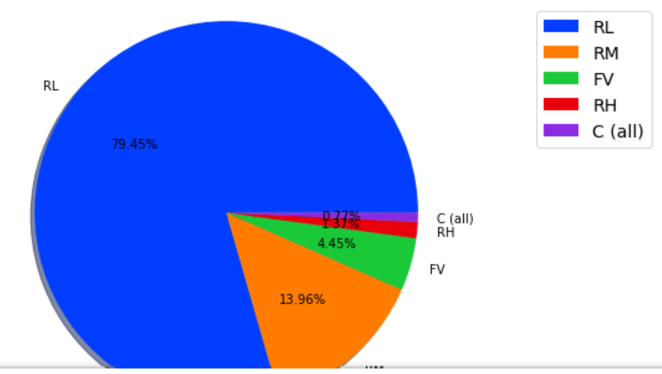


Univariate Analysis

*Plotting categorical columns*

```
def generate_pie(i):
    plt.figure(figsize=(10,5))
    colors = sns.color_palette('bright')[0:5]
    plt.pie(i.value_counts(), labels=i.value_counts().index, colors = colors, autopct='%1.2f%%',shadow=True,)
    plt.legend(prop={'size':14})
    plt.axis('equal')
    plt.tight_layout()
    return plt.show()


for j in df_train[Catg_data]:
    print(f"Pie plot for the column:", j)
    print(df_train[j].value_counts())
    generate_pie(df_train[j])
    print("*"*125)
```

```
Name: MSZoning, dtype: int64
```

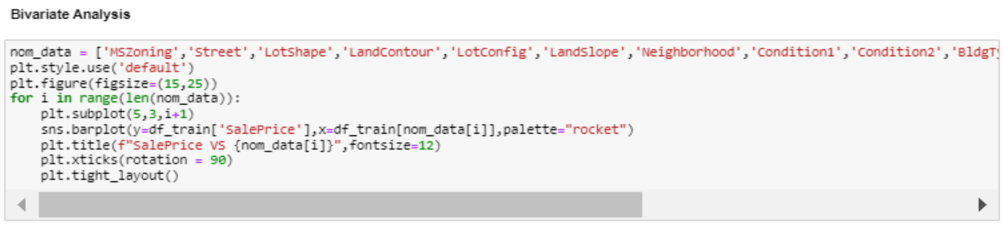


Observations:

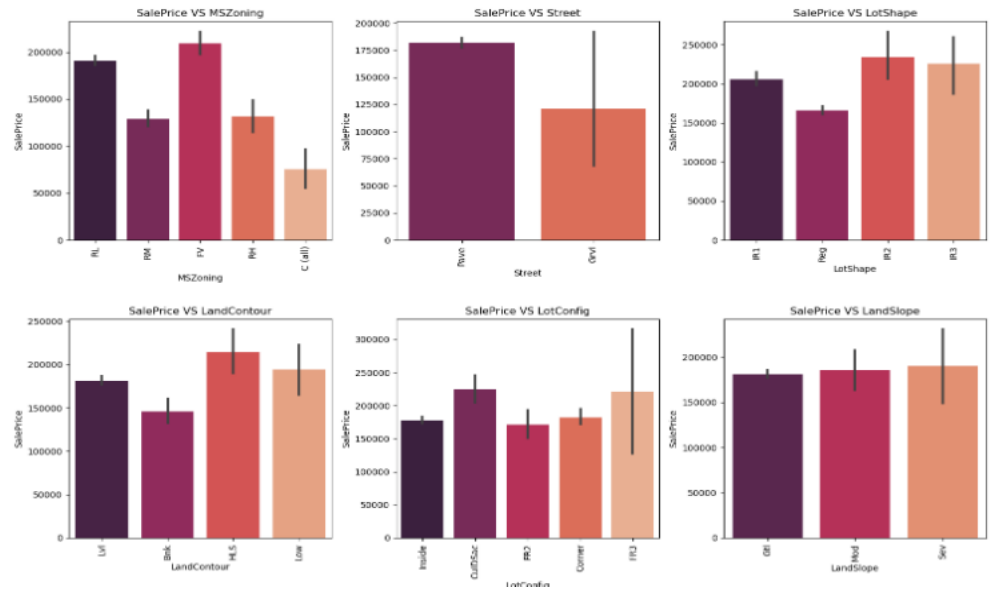
1. Street: The count of road access to the property Paved is 1164 which covers around 99.66% of the property where Graved type has count 4 that is only 0.34%.
2. LotShape: The count is high for the property having the shape regular followed by slightly regular and the count is less for irregular shape of property.
3. LandContour: The total number of flatness of the property for level is high which has 89.55% and the others like banked, hillside and depression have very less flatness of area.
4. LandSlope: The slope of the property Gentle slope has very high count of 1105 i.e, 94.61% and other types of Moderate Slope and Severe Slope have very less count.
5. MasVnrType: Around 60% of the houses does not have Masonry veneer type, 30% of the houses contains Brick Face type of Masonry veneer.
6. ExterQual: Around 61% of the houses evaluates typical/average quality of the material on the exterior, 34% of the houses have

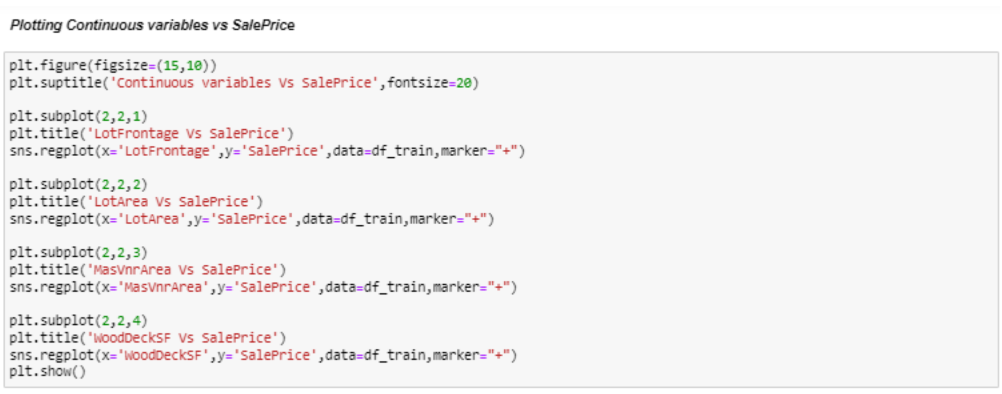
good quality of the material on the exterior. Only few have excellent quality.

7. BsmtQual: Most of the houses evaluates typical/average and good quality of height of the basement, only few of the houses have excellent quality.
8. BsmtCond: Around 91% of the houses have typical/average condition of the basement and only 2 houses have poor condition.
   A. BsmtExposure: Around 67% of the houses does not contain any walkout or garden level walls and 7% of the houses have minimum exposure.
9. CentralAir: 93.32% of the houses have central air conditioning and only 6.68% of houses do not have air conditioning.
10. KitchenQual: 49% of the houses contains typical/average kitchen quality and 40% of the houses have good kitchen quality. The count for excellent kitchen quality is very low and is around 2%.
11. GarageFinish: 47% Of the houses have unfinished garage interior, 29% rough finished and only 23% of the houses' interior garage has finished.
12. PavedDrive: 91.70% of the houses contains the paved driveway.


Bivariate Analysis

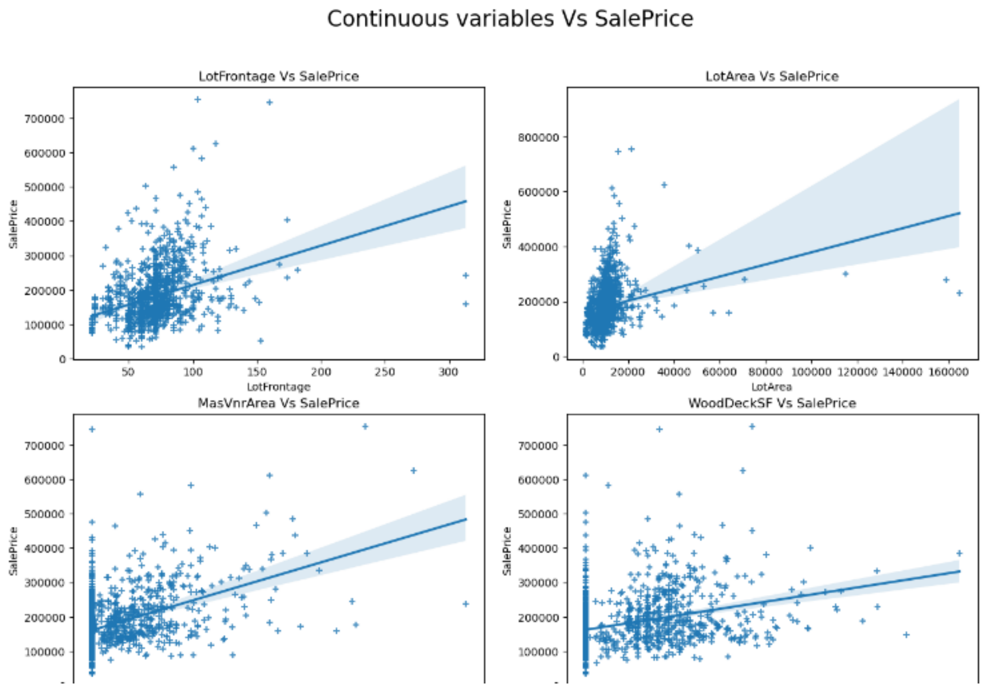
```
nom_data = ['MSZoning','Street','LotShape','LandContour','LotConfig','LandSlope','Neighborhood','Condition1','Condition2','BldgT
plt.style.use('default')
plt.figure(figsize=(15,25))
for i in range(len(nom_data)):
    plt.subplot(5,3,i+1)
    sns.barplot(y=df_train['SalePrice'],x=df_train[nom_data[i]],palette="rocket")
    plt.title(f"SalePrice VS {nom_data[i]}",fontsize=12)
    plt.xticks(rotation = 90)
    plt.tight_layout()
```





*Plotting Continuous variables vs SalePrice*

```
plt.figure(figsize=(15,10))
plt.suptitle('Continuous variables Vs SalePrice',fontsize=20)

plt.subplot(2,2,1)
plt.title('LotFrontage Vs SalePrice')
sns.regplot(x='LotFrontage',y='SalePrice',data=df_train,marker="+")

plt.subplot(2,2,2)
plt.title('LotArea Vs SalePrice')
sns.regplot(x='LotArea',y='SalePrice',data=df_train,marker="+")

plt.subplot(2,2,3)
plt.title('MasVnrArea Vs SalePrice')
sns.regplot(x='MasVnrArea',y='SalePrice',data=df_train,marker="+")

plt.subplot(2,2,4)
plt.title('WoodDeckSF Vs SalePrice')
sns.regplot(x='WoodDeckSF',y='SalePrice',data=df_train,marker="+")
plt.show()
```



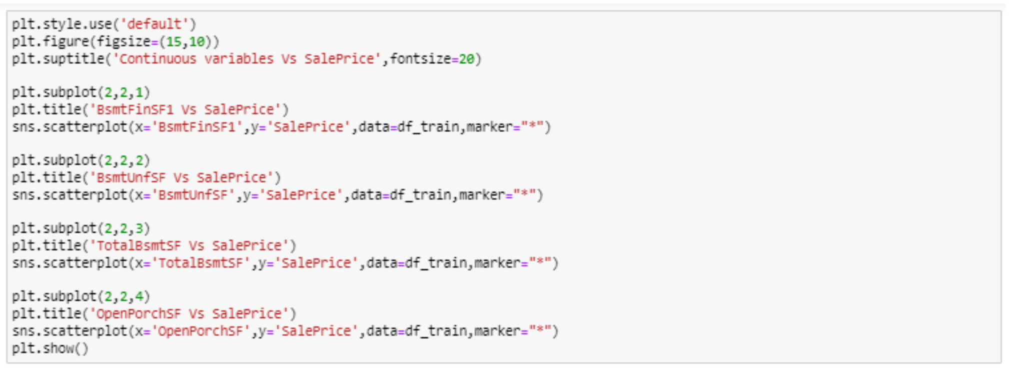
**Observations:**

✓ SalePrice vs LotFrontage: From the plot we can observe there is not much linear relation between the label and feature. If the linear feet of street connected to property is more, the sale price is also high.

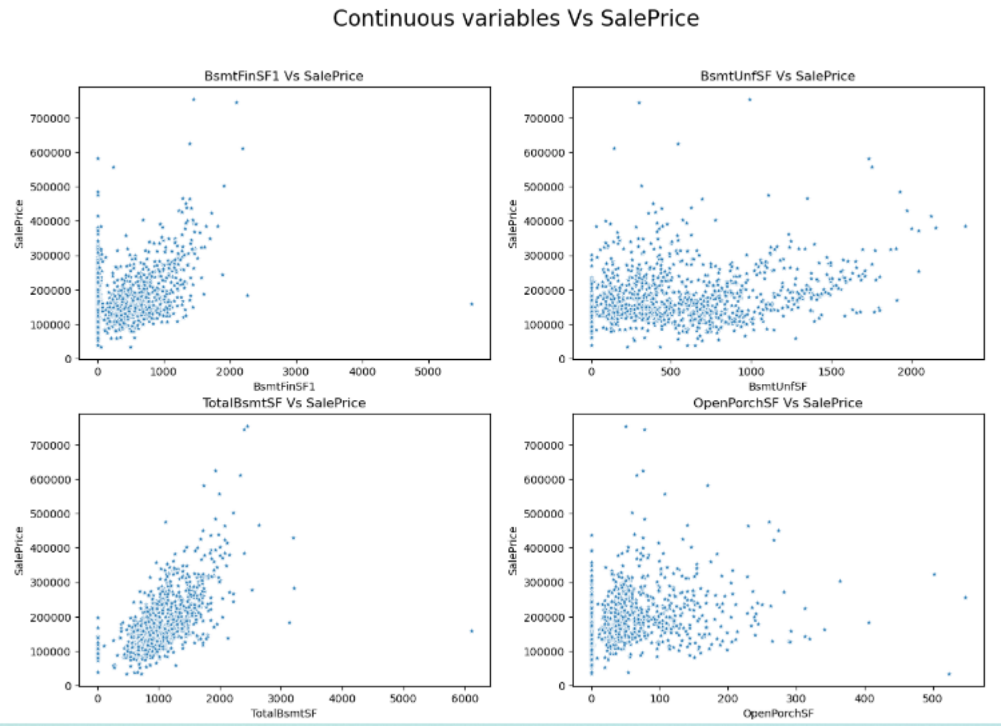
✓ SalePrice vs LotArea: There is weakly positive linear relation between the label and feature. But the sale price is high when lot size has around 20000 square feet area. Also, as the lot size increases the price is also increasing moderately.

✓ SalePrice vs MasVnrArea: There is bit positive linear relation between feature and target. Also, the sale price is high when Masonry veneer area has around 50-400 square feet. So as the Masonry veneer area in square feet increases sale price is also increasing.

✓ SalePrice vs WoodDeckSF: There is weakly positive linear relation between the feature and target. As the Wood deck area increases, sale price is also increases.



```
plt.style.use('default')
plt.figure(figsize=(15,10))
plt.suptitle('Continuous variables Vs SalePrice',fontsize=20)

plt.subplot(2,2,1)
plt.title('BsmtFinSF1 Vs SalePrice')
sns.scatterplot(x='BsmtFinSF1',y='SalePrice',data=df_train,marker="*")

plt.subplot(2,2,2)
plt.title('BsmtUnfSF Vs SalePrice')
sns.scatterplot(x='BsmtUnfSF',y='SalePrice',data=df_train,marker="*")

plt.subplot(2,2,3)
plt.title('TotalBsmtSF Vs SalePrice')
sns.scatterplot(x='TotalBsmtSF',y='SalePrice',data=df_train,marker="*")

plt.subplot(2,2,4)
plt.title('OpenPorchSF Vs SalePrice')
sns.scatterplot(x='OpenPorchSF',y='SalePrice',data=df_train,marker="*")
plt.show()
```



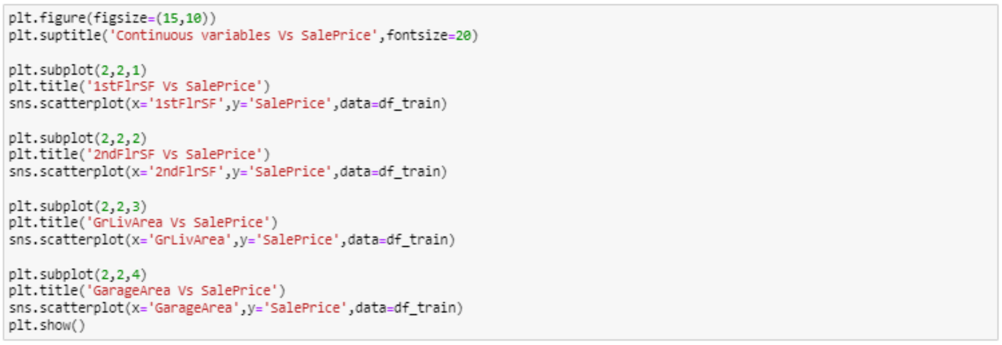
Observations:

✓ SalePrice vs BsmtFinSF1: There is weakly positive linear relation between feature and label. The sale price is high that is 100000-300000 when basement square feet lie upto 1500 square feet. So as the type 1 basement finished square feet increases, sale price is also increases.

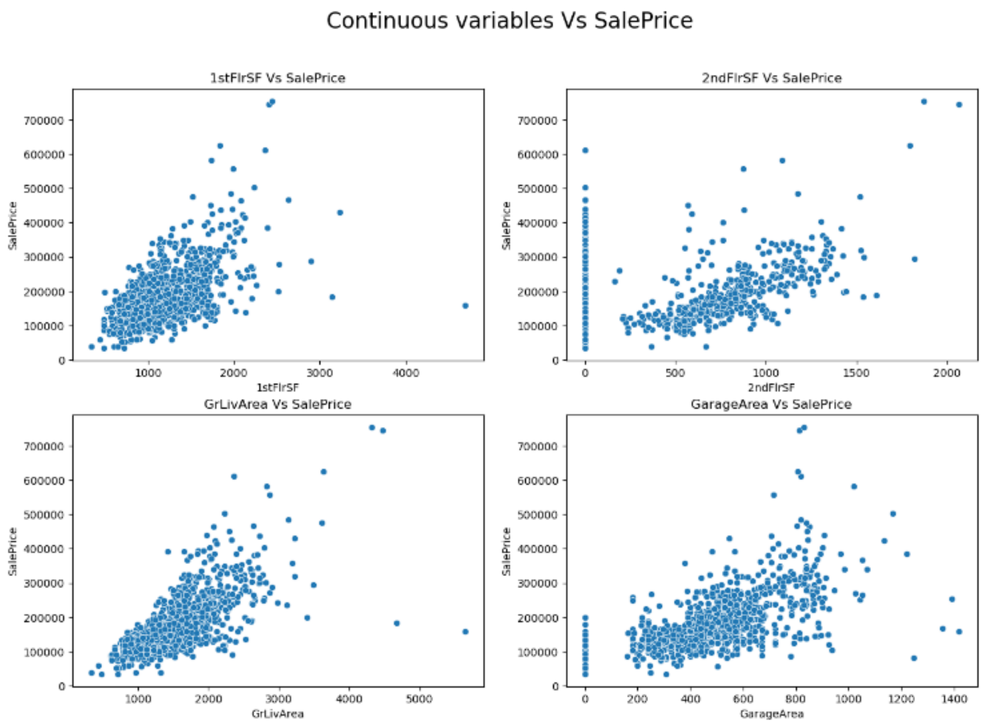
✓ SalePrice vs BsmtUnfSF: There is positive linear relation between the target and BsmtUnfSF. When the unfinished basement area is below 1000 square feet, the sale price is high.

✓ SalePrice vs TotalBsmtSF: There is positive linear relation between sale price nad TotalBsmtSF. As total basement area increases, sale price also increases.

✓ SalePrice vs OpenPorchSF: There is a linear relation between the label and feature. The sale price is high when Open porch area is below 200sf. Here also as the Open porch area increases, sale price is also increases.



```
plt.figure(figsize=(15,10))
plt.suptitle('Continuous variables Vs SalePrice',fontsize=20)

plt.subplot(2,2,1)
plt.title('1stFlrSF Vs SalePrice')
sns.scatterplot(x='1stFlrSF',y='SalePrice',data=df_train)

plt.subplot(2,2,2)
plt.title('2ndFlrSF Vs SalePrice')
sns.scatterplot(x='2ndFlrSF',y='SalePrice',data=df_train)

plt.subplot(2,2,3)
plt.title('GrLivArea Vs SalePrice')
sns.scatterplot(x='GrLivArea',y='SalePrice',data=df_train)

plt.subplot(2,2,4)
plt.title('GarageArea Vs SalePrice')
sns.scatterplot(x='GarageArea',y='SalePrice',data=df_train)
plt.show()
```



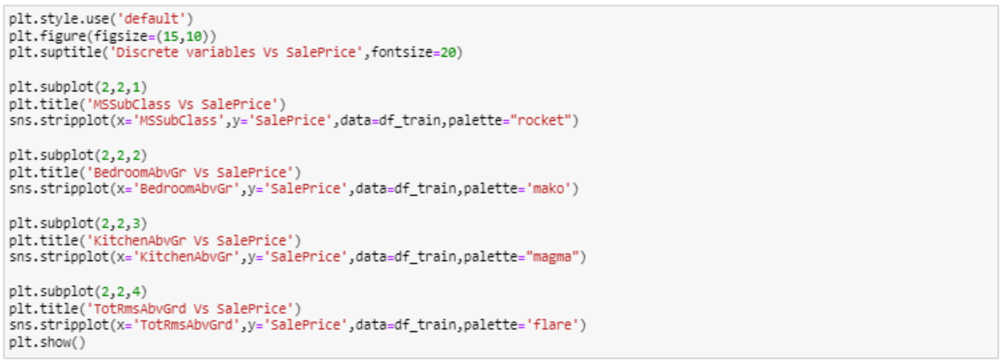
Observations:

✓ SalePrice vs 1stFlrSF: There is a linear relation between the label and feature. As we can observe in the plot, the sale price is high when the first- floor area lies between 500-2000 square feet. So as the 1st floor area increases, sales price also increases moderately.

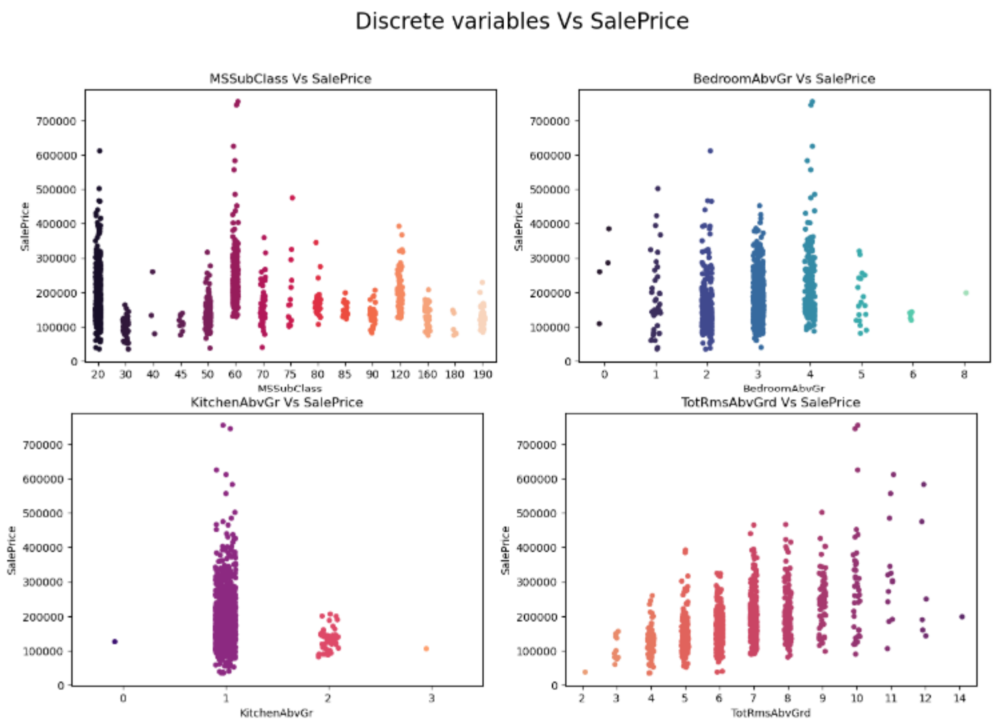
✓ SalePrice vs 2ndFlrSF: There is a positive correlation between SalePrice and 2ndFlrSF. So, it is obvious that the sale price increases based on the floors.

✓ SalePrice vs GrLivArea: Most of the houses have above grade living area. There is a positive correlation between the label and feature. Here as the above grade living area increases, sale price also increases.

✓ SalePrice vs GarageArea: Similar to 2nd floor sf, here also positive linear relation between the label and feature. As size of garage area increases, sale price also increases. The sale price is high when size of garage area is beween 200-800 square feet.



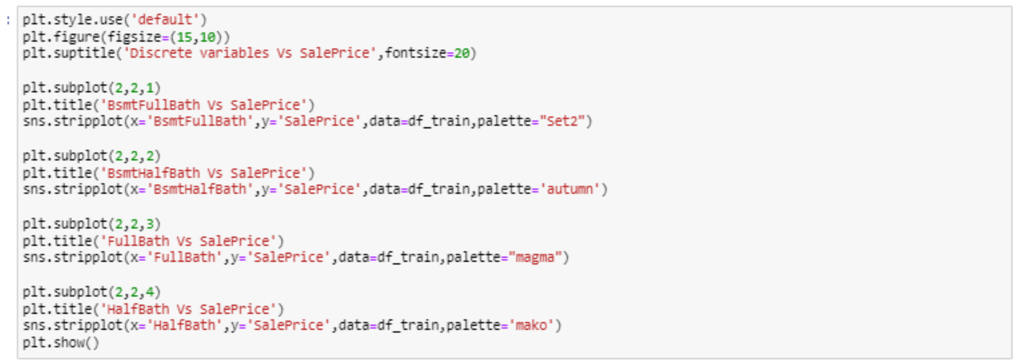
```
plt.style.use('default')
plt.figure(figsize=(15,10))
plt.suptitle('Discrete variables Vs SalePrice',fontsize=20)

plt.subplot(2,2,1)
plt.title('MSSubClass Vs SalePrice')
sns.stripplot(x='MSSubClass',y='SalePrice',data=df_train,palette="rocket")

plt.subplot(2,2,2)
plt.title('BedroomAbvGr Vs SalePrice')
sns.stripplot(x='BedroomAbvGr',y='SalePrice',data=df_train,palette='mako')

plt.subplot(2,2,3)
plt.title('KitchenAbvGr Vs SalePrice')
sns.stripplot(x='KitchenAbvGr',y='SalePrice',data=df_train,palette="magma")

plt.subplot(2,2,4)
plt.title('TotRmsAbvGrd Vs SalePrice')
sns.stripplot(x='TotRmsAbvGrd',y='SalePrice',data=df_train,palette='flare')
plt.show()
```



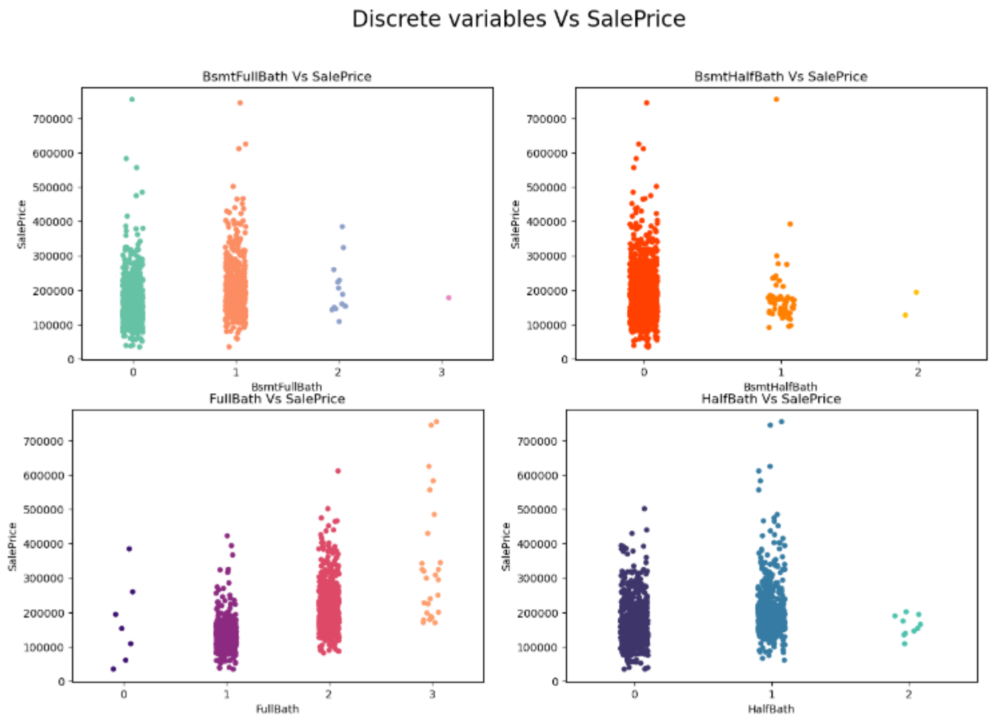
**Observations:**

✓ SalePrice vs MSSubClass: The sale price is high for the MSSubClass 60,120 and 20.

✓ SalePrice vs BedroomAbvGr: Many houses are having 0 and 4 bedrooms have high sales price also houses having 8 bedrooms also have high sales price. Other bedroom grades have average sale price.
✓ SalePrice vs KitchenAbvGr: Most of the houses have single kitchen and few houses have 2 kitchens. The sale price is also high in case of the houses having single kitchen.
✓ SalePrice vs TotRmsAbvGrd: We can observe some linear relation between Total rooms above grade and Sale Prices as the number of rooms increases the sales price also increases.



```
plt.style.use('default')
plt.figure(figsize=(15,10))
plt.suptitle('Discrete variables Vs SalePrice',fontsize=20)

plt.subplot(2,2,1)
plt.title('BsmtFullBath Vs SalePrice')
sns.stripplot(x='BsmtFullBath',y='SalePrice',data=df_train,palette="Set2")

plt.subplot(2,2,2)
plt.title('BsmtHalfBath Vs SalePrice')
sns.stripplot(x='BsmtHalfBath',y='SalePrice',data=df_train,palette='autumn')

plt.subplot(2,2,3)
plt.title('FullBath Vs SalePrice')
sns.stripplot(x='FullBath',y='SalePrice',data=df_train,palette="magma")

plt.subplot(2,2,4)
plt.title('HalfBath Vs SalePrice')
sns.stripplot(x='HalfBath',y='SalePrice',data=df_train,palette='mako')
plt.show()
```



Observations:

✓ SalesPrice vs BsmtFullBath: Most of the houses have basement full bathrooms as 0 and 1 which means some of the houses have single basement bathrooms and some of the houses have no basement bathrooms. And sales price is also high in these cases.
✓ SalesPrice vs BsmtHalfBath: The houses do not have any single basement bathrooms and those houses have average sales price.
✓ SalesPrice vs FullBath: There is positive linear relation between the sale price and full bathrooms above grade. Large number of houses have 1-2 full bathrooms. As the full bathrooms grades increases, sale price is also increasing slightly.
✓ SalesPrice vs HalfBath: Some of the houses have no half bathrooms and some of the houses have single half bathroom and very few houses have 2 half bathrooms. The houses with 0-1 half bathrooms have average sale price.

## • Interpretation of the Results

Visualizations: It helped me to understand the correlation between independent and dependent features. Also, helped me with feature importance and to check for multi collinearity issues. Detected outliers/skewness with the help of boxplot and distribution plot. I got to know the count of a particular category for each feature by using count plot and most importantly with predicted target value distribution as well as scatter plot helped me to select the best model.
Pre-processing: Basically, before building the model the dataset should be cleaned and scaled by performing few steps. As I mentioned above in the pre-processing steps where all the important features are present in the dataset and ready for model building.
Model Creation: Now, after performing the train test split, I have x_train, x_test, y_train & y_test, which are required to build Machine learning models. I have built multiple regression models to

get the best R2 score, MSE, RMSE & MAE out of all the models.

## CONCLUSION

- Key Findings and Conclusions of the Study

In this study, we have used multiple machine learning models to predict the house sale price. We have gone through the data analysis by performing feature engineering, finding the relation between features and label through visualizations. And got the important feature and we used these features to predict the price by building ML models. We have got good prediction results and after hyper parameter tuning, R2 score was nearly 89% also the errors decreased which means no over-fitting issue.

### Findings:

### Which variables are important to predict the price of variable?

✓ Overall Quality is the most contributing and highest positive impacting feature for prediction. Also, the features like GarageArea, LotArea, 1stFlrSF, TotalBsmtSF etc have somewhat linear relation with the price variable.

### How do these variables describe the price of the house?

✓ 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 as the overall quality of the house increases, the sale price also increases. That is there is good linear relation between SalePrice and OverallQual. So, if the seller builds the house according to these types of qualities that will increase the sale price of the house.

✓ There is a linear relation between the SalePrice and 1stFlrSF. As we have seen as the 1st floor area increases, sales price also increases moderately. So, people like to live in the houses which have only 1-2 floors and the cost of the house also increases in this case.

✓ Also, we have seen the positive linear relation between the SalePrice and GarageArea. As size of garage area increases, sale price also increases.

✓ There is positive linear relation between sale price and TotalBsmtSF. As total basement area increases, sale price also increases.

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

While working on this project I learned more things about the housing market and how the machine learning models have helped to predict the price of house which indeed helps the sellers and buyers to understand the future price of the house. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This 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.

Finally, our aim is achieved by predicting the house price for the test data, I hope this will be further helps for sellers and buyers to understand the house marketing. The machine learning models, and data analytic techniques will have an important role to play in this type of problems. It helps the customers to know the future price of the houses.

- Limitations of this work and Scope for Future Work

During this project I have faced a problem with quality of data. Many columns are with same entries in more than 80% of rows which lead to reduction in our model performance. One more issue is there are large number of missing values presents in this data set, so, we have to fill those missing values in correct manner. We can still improve our model accuracy with some feature engineering and by doing some extensive hyperparameter tuning on it.

**Thank You**