FLIP ROBO

PROJECT REPORT ON SURPRISE HOUSING - HOUSING PRICE PREDICATION & ANALYSIS PROJECT

SUBMITTED BY:
MS. YASHSHREE BAVISKAR

FLIPROBO SME:
MR. MOHD KASHIF

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills. And I want to express my huge gratitude to Mr. Mohd Kashif (SME Flip Robo), he is the person who has helped me to get out of all the difficulties I faced while doing the project.

A huge thanks to “Data trained” who are the reason behind my Internship at Fliprobo. Last but not least my parents who have been my backbone in every step of my life.

References use in this project:

1. SCIKIT Learn Library Documentation
2. Blogs from towards data science, Analytics Vidya, Medium.com
3. Andrew Ng Notes on Machine Learning (GitHub)
4. Data Science Projects with Python Second Edition by Packt
5. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron,
6. Andrew Ng Notes on Machine Learning (GitHub)
7. Sifei Lu, Zengxiang Li, Zheng Qin, Xulei Yang, Rick Siow Mong Goh, ”A Hybrid Regression Technique for House Prices Prediction”, @2017 IEEE, 2017 IEEE International Conference on Industrial Engineering & Engineering Management , Singapore DOI:10.1109/IEEM.2017.8289904
8. CH.Raga Madhuri, Anuradha G, M.Vani Pujitha “House Price Prediction Using Regression Techniques: A Comparative Study”, IEEE 6th International Conference on smart structures and systems ICSSS 2019
9. Zhen Peng, Qiang Huang, Yincheng Han, “Model Research on Forecast of Second-Hand House Price in Chengdu Based on XGboost Algorithm”, 2019 IEEE 11th International Conference on Advanced Infocomm Technology
10. J.-G. Liu, X.-L. Zhang, and W.-P. Wu, “Application of fuzzy neural network for real estate prediction,” Advances in Neural Networks - ISNN 2006, vol. 3973, pp. 1187–1191, 2006.
11. Arietta, Sean M., et al. "City forensics: Using visual elements to predict non-visual city attributes." IEEE transactions on visualization and computer graphics 20.12 (2014): 2624-

Chap. 1 INTRODUCTION

The real estate market is one of the ever-developing markets in the world today. This can be attributed to the fact that housing is an essential need for man. There are several stakeholders interested in the prediction of housing prices in a particular area ranging from landowners, to realtors, potential buyers, as well as government authorities as this, plays a major role in the economy. The sale or purchase of a house is considered one of the crux decisions of life due to a large amount of money involved and the commitment of relocation. The pricing of houses is affected by various factors, from the location of the house, the features in the house, including the demand and supply of houses in that area (Phan, 2019). Studies have also shown that properties also appreciate over time resulting from economic growth and development except in cases of wars, emigration and natural disasters amongst others. Therefore, there is always appreciation in the value of house prices. This has made a real estate investment a lucrative venture. Thus, the prediction of housing sale price can also be considered as an important economic index. The value of a house that increases with time requires the appraisal value to be calculated as this value is required during the sale, purchase or even mortgage of a house (Shinde and Gawande, 2018). There could be bias in determining the appraisal values by the stakeholders especially the professional appraisals. The bias is a shortcoming from professional appraisals which would, in turn, affect actual value on the housing (Shinde and Gawande, 2018). To curb this, determining the actual value of the house sale price would require a system void of the bias. This is hence the need for introducing machine learning in predicting housing sale price. The system, in turn, would guide stakeholders and inexperienced stakeholders can, in turn, be guided by this system in order not to make losses. Machine learning trains a computer to do what comes normally to animals or human beings while learning from experience. A model from machine learning makes use of computational models to "learn" from data without depending on a predetermined equation as a model (MathWorks, 2016). Various trends can influence the prices of hosing. These trends, as well as other parameters such as the construction materials used, number of bedrooms, living area (Bagheri, 2015), location, upcoming projects, proximity (Bourassa, Cantoni and Hoesli, 2011) amongst others, are turned into raw data. The raw data are used as inputs for the model and in turn, produce a just price without any form of bias.

1.1 BUSINESS PROBLEM FRAMING

Real Estate Property is not only the basic need of a man but today it also represents the riches and prestige of a person. Investment in real estate generally seems to be profitable because their property values do not decline rapidly. The market demand for housing is always increasing every year due to increase in population and migrating to other cities for their financial purpose. Changes in the real estate price can affect various household investors, bankers, policy makers and many. Investment in Housing seems to be an attractive choice for the investments.

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. In general, purchasing and investing in any real estate project will involve various transactions between different parties. Thus, it could be a vital decision for both households and enterprises. How to construct a realistic model to precisely predict the price of real estate has been a challenging topic with great potential for further research. There are many factors that have an impact on house prices, such as the number of bedrooms and bathrooms. House price depends upon its location as well. A house with great accessibility to highways, schools, malls, employment opportunities, would have a greater price as compared to a house with no such accessibility. Regression is a supervised learning algorithm in machine learning which is used for prediction by learning and forming a relationship between present statistical data and target value i.e., Sale Price in this case. Different factors are taken into consideration while predicting the worth of the house like location, neighborhood and various amenities like garage scape, no. of bed rooms etc.

Surprise Housing is a US based housing company that just made a decision to enter the Australian market. The company uses data analytics to buy houses at a low price mostly lower than their actual value and in turn sell it at a higher price. The company collected data from sale of houses in Australia and made a dataset of about 1460 observations and 80 variables. They intend to perform analytics on this data to be able to select prospective houses to buy to enter the market and how each variable affect the price of housing in Australia. A clear understanding of the business objective is a major step in creating a good model that will benefit the business. In this project, we would like to predict a good level of accuracy of the prices of houses to be able to marginally make profits. To determine the price of a house various factors can influence it. Three factors affect the price of a house, Rahadi et al., (2013) classifies them into three; physical condition, location, and concept. To solve this business problem, there would be a need to create a model that best combines these features to make good predictions and thus enable the company to make better profit.

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

REVIEW OF LITERATURE

Housing is one out of the 3 basic needs for human survival hence research about housing and all that relates to it can never be over

emphasized. In this section of this research work I will examine different papers or research work that have been published previously as regards housing and prices in the past and the research gap noticed which forms the basis for my own research work.
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.
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.
With its great weather, cosmopolitan cities, diverse natural landscapes and relaxed lifestyle, it's no wonder that Australia remains a top pick for expats.
Living cost in Australia for one person: $2,835 per month. Average living expenses for a couple: $4,118 per month. Average monthly living expenses for a family of 4: $5,378. Australia currently has the 16th highest cost of living in the world, with the USA and UK well behind at 21st and 33rd place respectively. Sydney and Melbourne are popular choices for expats moving to Australia. House pricing in some of the top Australian cities:-
Sydney - median house price A$1,142,212
Adelaide- median house price A$542,947
Hobbart (smaller city)- median house price A$530,570.

Why Housing? Why Housing Prices?
The spread of the 2008 mortgage crisis from the United States to the whole world caused all countries to become more interested in the real estate sector. (The 1997 Asian crisis and the boom in asset prices before the 1929 and 2008 crises in the United States are examples of major financial crises that can be caused by speculative housing price bubbles.) The importance of monitoring the price movements and loan amounts of the housing sector became clear after the 2008 global financial crisis, and since that time, its importance has only increased. Monitoring housing prices, which both reflect developments in the housing market and provide information about the housing market's connection to other macroeconomic variables, has enabled both autonomous and state institutions to follow the housing sector much more closely.

MOTIVATION FOR THE PROBLEM UNDERTAKEN:

To understand real world problems where Machine Learning and Data Analysis can be applied to help organizations in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data. One of such domains is Real Estate.

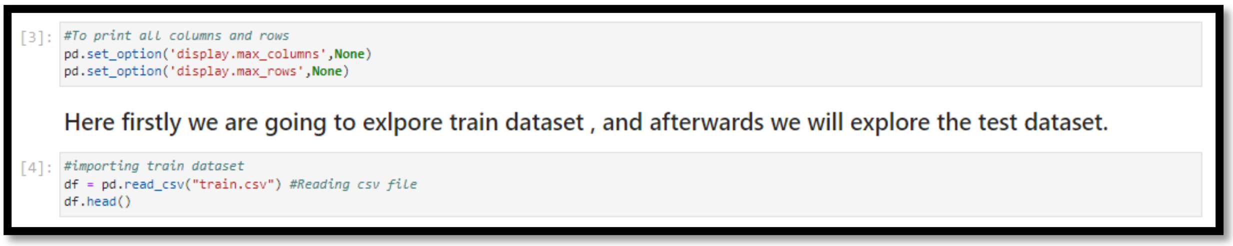
Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science 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. Our problem is related to one such housing company

Chap 2. Analytical Problem Framing

Our objective is to predict House price which can be resolve by use of regression-based algorithm. In this project we are going to use different types of algorithms which uses their own mathematical equation on background. This project comes with two separate data set for training & testing model. Initially data cleaning & pre-processing perform over data. Feature engineering is performed to remove unnecessary feature & for dimensionality reduction. In model building Final model is select based on evaluation benchmark among different models with different algorithms. Further Hyperparameter tuning performed to build more accurate model out of best model.

1. Data Sources and their formats

To better understand the business goal, we are breaking down all the variables of this dataset to see how each of them contributes to or affect the price of the housing price in Australia.



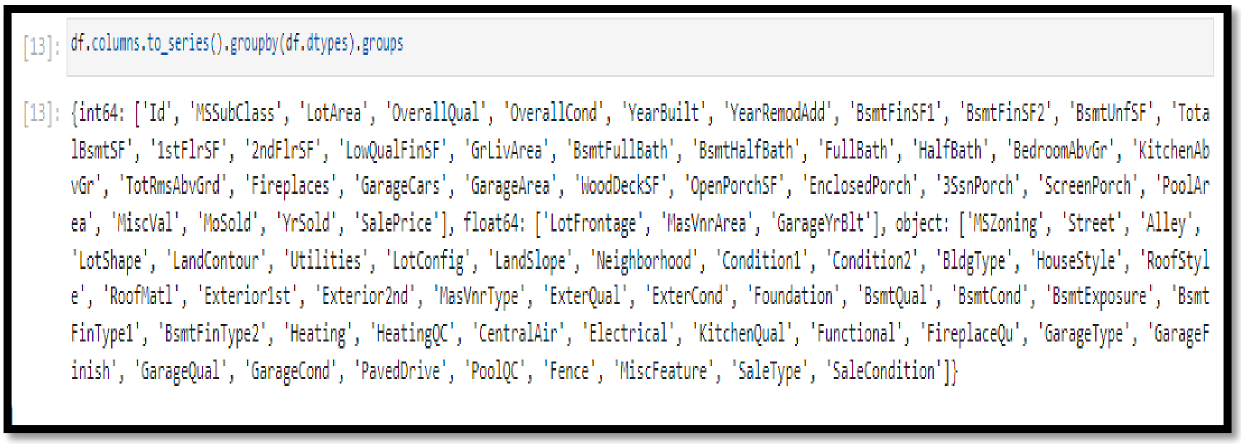
```
[3]: #To print all columns and rows
     pd.set_option('display.max_columns',None)
     pd.set_option('display.max_rows',None)
```

Here firstly we are going to exlpore train dataset , and afterwards we will explore the test dataset.

```
[4]: #importing train dataset
     df = pd.read_csv("train.csv") #Reading csv file
     df.head()
```

Train Dataset contain 81 columns with 1968 rows. Out of which 43 features with object datatypes, 35 features are of integer datatypes and rest are with float datatypes.

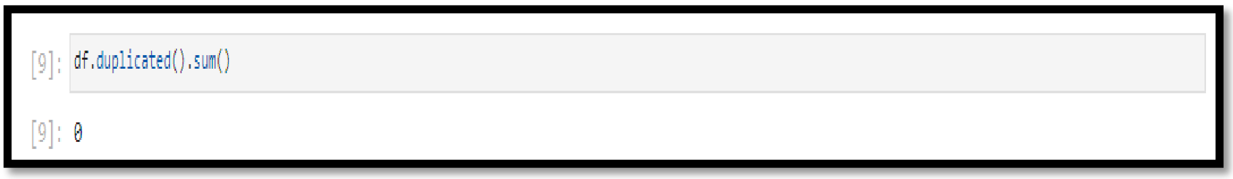
Test file contains all the independent variables, but not the target variable. We will apply the model to predict the target variable for the test data. The dimension of data is 292 rows and 80 columns.



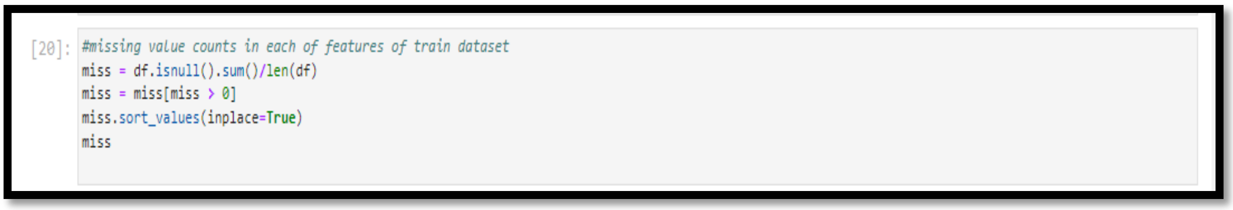
```
[13]: df.columns.to_series().groupby(df.dtypes).groups

[13]: {int64: ['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'Tota
      lBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAb
      vGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolAr
      ea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice'], float64: ['LotFrontage', 'MasVnrArea', 'GarageYrBlt'], object: ['MSZoning', 'Street', 'Alley',
      'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyl
      e', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'Bsmt
      FinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType', 'GarageF
      inish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']}
```

2. Data Pre-processing

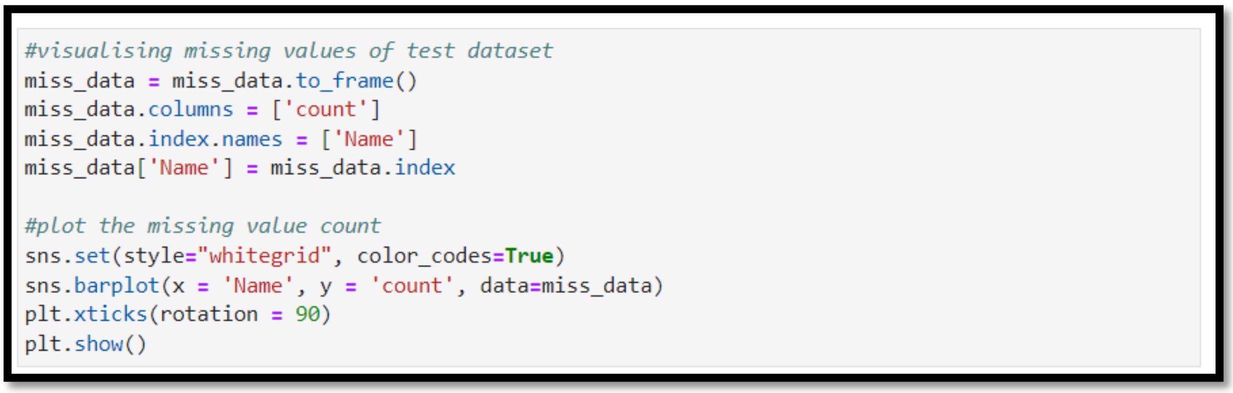
Before pre-processing data, integrity of data is check for missing values, possible duplicates, to check. The dataset is large and it may contain some data error. In order to reach clean, error free data some data cleaning & data pre-processing performed data.



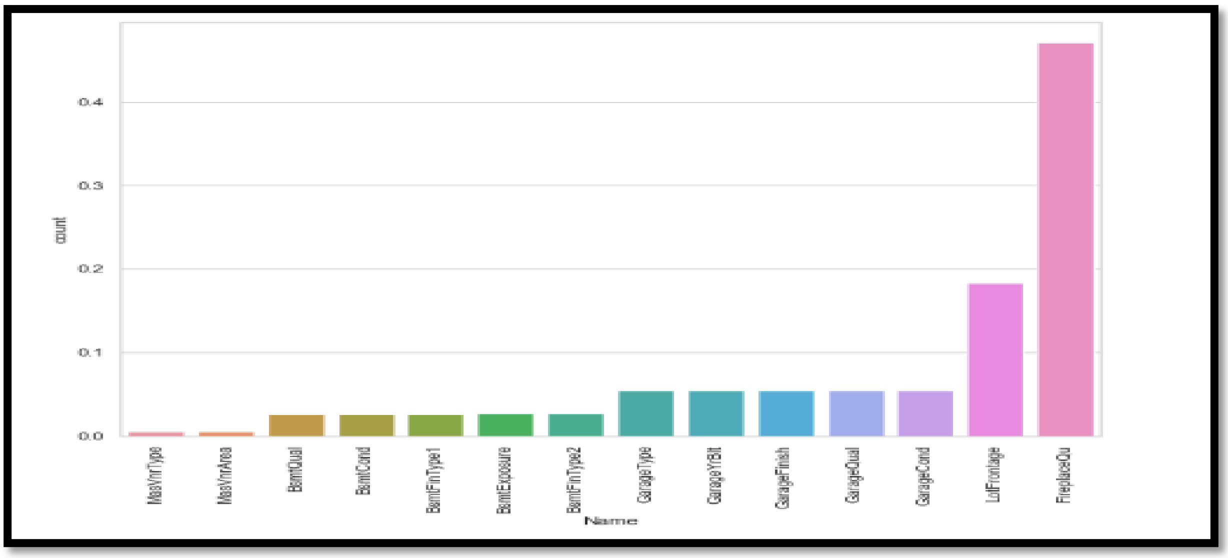
```
[9]: df.duplicated().sum()

[9]: 0
```



```
[20]: #missing value counts in each of features of train dataset
      miss = df.isnull().sum()/len(df)
      miss = miss[miss > 0]
      miss.sort_values(inplace=True)
      miss
```



```
#visualising missing values of test dataset
miss_data = miss_data.to_frame()
miss_data.columns = ['count']
miss_data.index.names = ['Name']
miss_data['Name'] = miss_data.index

#plot the missing value count
sns.set(style="whitegrid", color_codes=True)
sns.barplot(x = 'Name', y = 'count', data=miss_data)
plt.xticks(rotation = 90)
plt.show()
```

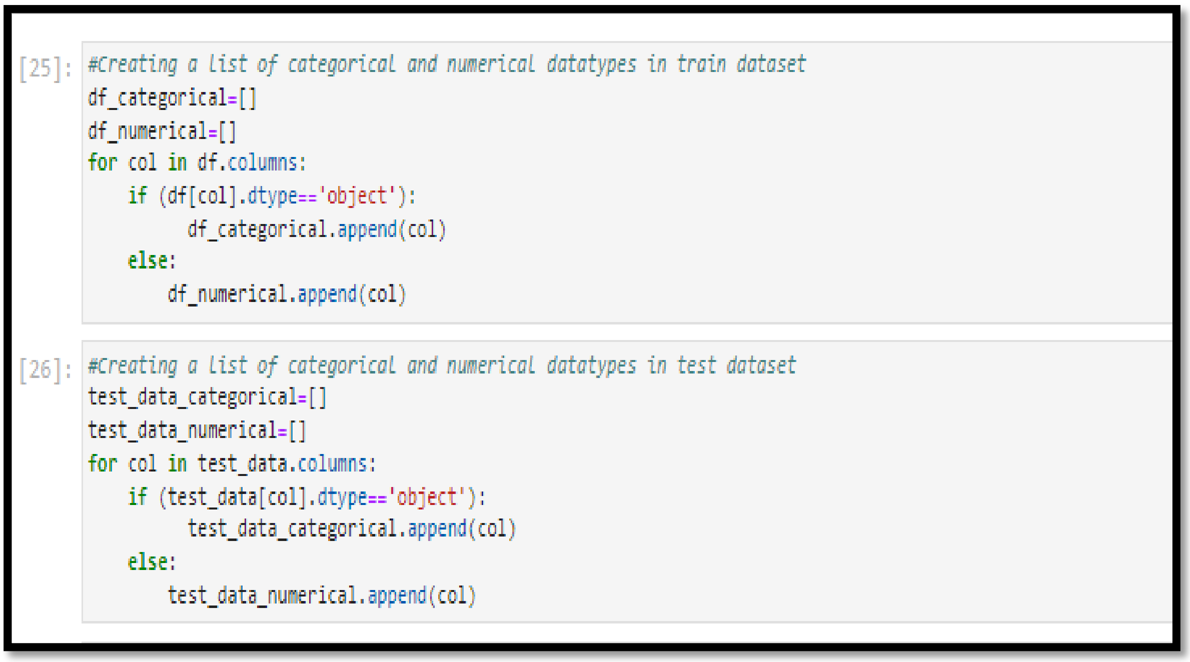


Data Pre-processing perform on data:

- We can infer that the variable PoolQC has 99.5% missing values followed by MiscFeature, Alley, and Fence.
- Here we have remove/ dropped the unwanted features which are not showing much contribution towards our target variable.
- Here we have check the unique value of all features to get more inference about our data set.
- We have explore all the features which are available in the dataset to get clear idea while doing further data investigation.

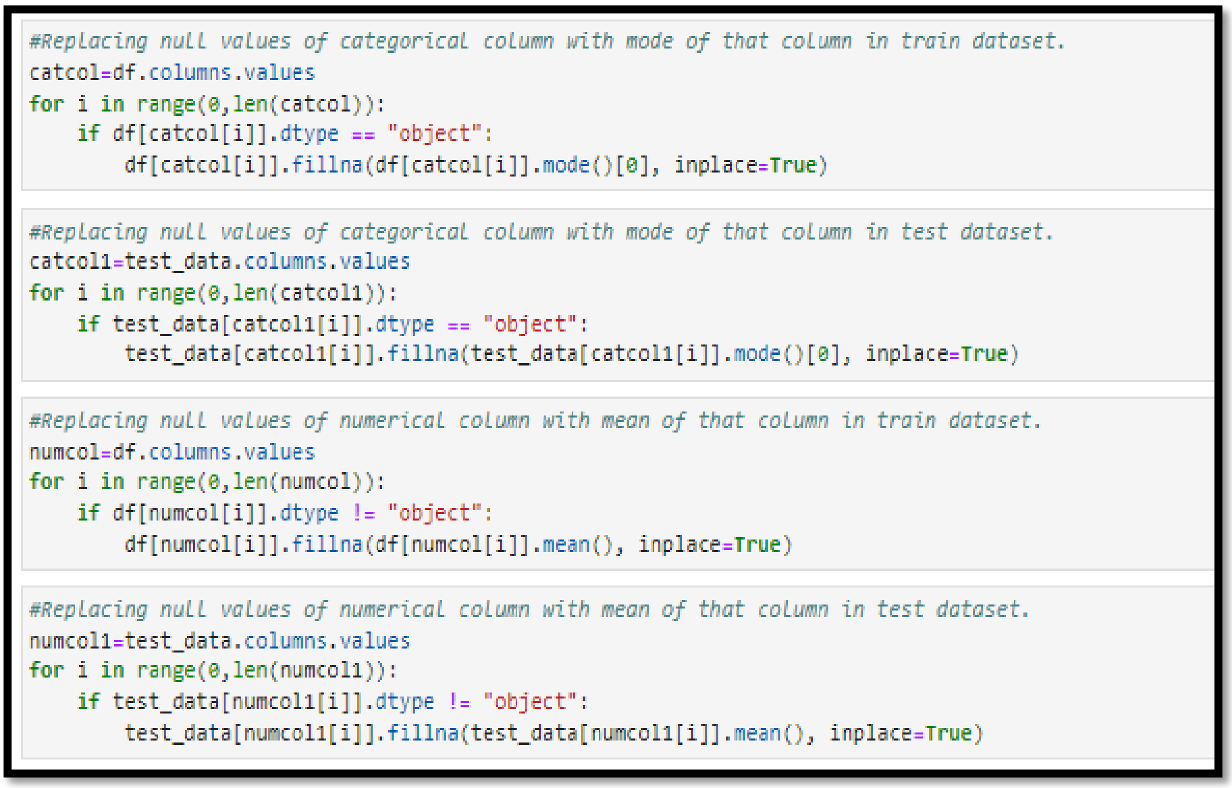
Here we have done data imputation in various features where missing value available using mean for numerical features and mode for categorical features.

Here we have segregated the categorical and numerical columns using loop as per below for both the dataset (i.e., train and test):



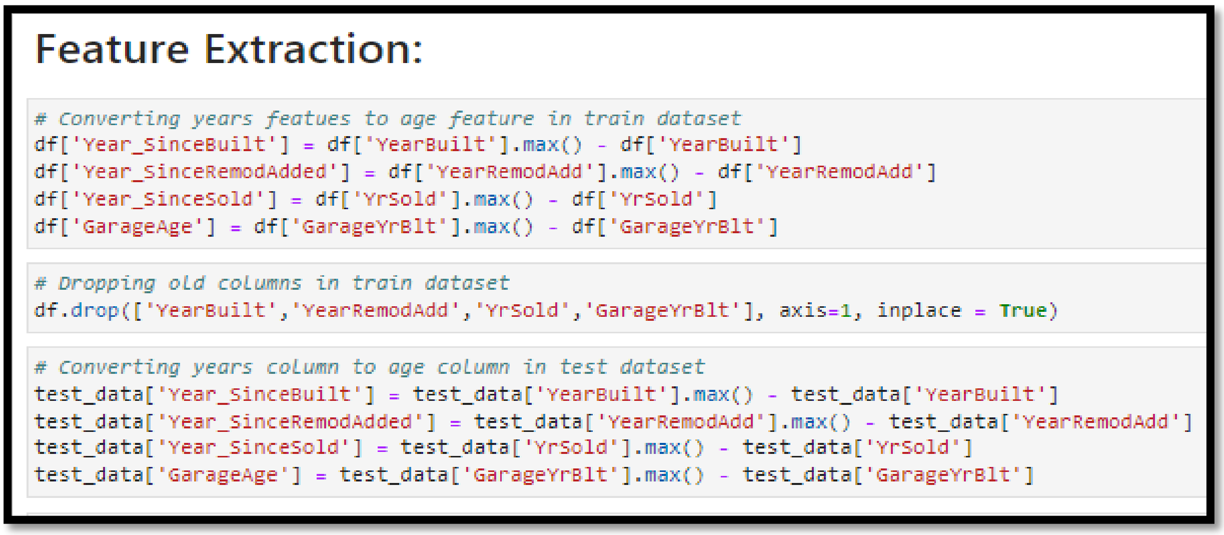
```
[25]: #Creating a list of categorical and numerical datatypes in train dataset
      df_categorical=[]
      df_numerical=[]
      for col in df.columns:
          if (df[col].dtype=='object'):
                df_categorical.append(col)
          else:
              df_numerical.append(col)

[26]: #Creating a list of categorical and numerical datatypes in test dataset
      test_data_categorical=[]
      test_data_numerical=[]
      for col in test_data.columns:
          if (test_data[col].dtype=='object'):
                test_data_categorical.append(col)
          else:
              test_data_numerical.append(col)
```

Here we have done the data imputation for train and test dataset respectively.



```
#Replacing null values of categorical column with mode of that column in train dataset.
catcol=df.columns.values
for i in range(0,len(catcol)):
    if df[catcol[i]].dtype == "object":
        df[catcol[i]].fillna(df[catcol[i]].mode()[0], inplace=True)

#Replacing null values of categorical column with mode of that column in test dataset.
catcol1=test_data.columns.values
for i in range(0,len(catcol1)):
    if test_data[catcol1[i]].dtype == "object":
        test_data[catcol1[i]].fillna(test_data[catcol1[i]].mode()[0], inplace=True)

#Replacing null values of numerical column with mean of that column in train dataset.
numcol=df.columns.values
for i in range(0,len(numcol)):
    if df[numcol[i]].dtype != "object":
        df[numcol[i]].fillna(df[numcol[i]].mean(), inplace=True)

#Replacing null values of numerical column with mean of that column in test dataset.
numcol1=test_data.columns.values
for i in range(0,len(numcol1)):
    if test_data[numcol1[i]].dtype != "object":
        test_data[numcol1[i]].fillna(test_data[numcol1[i]].mean(), inplace=True)
```

Here we have done feature extraction to get the some more information in relation to our target variable from some features on both the data set and drop the old features which are not needed in our further steps.



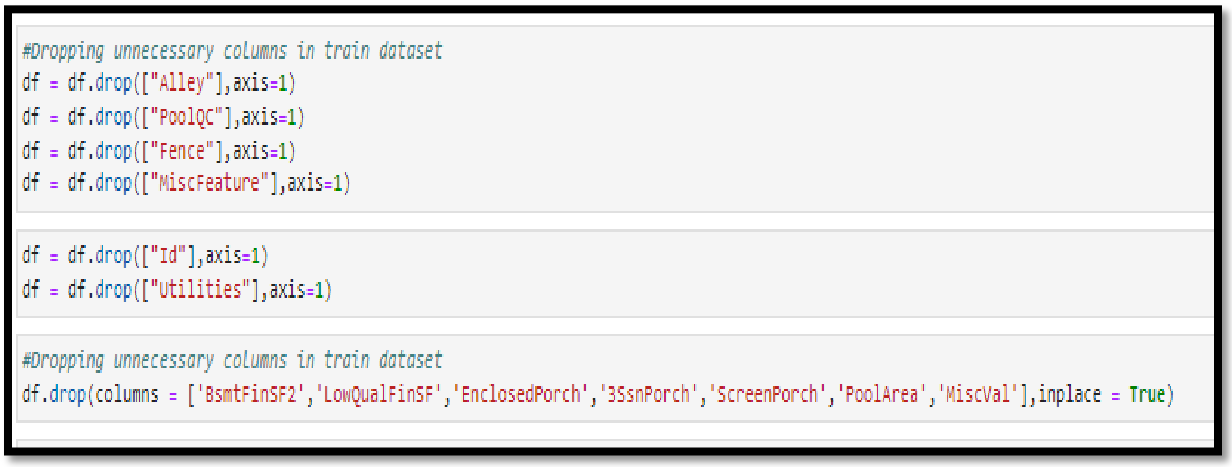
Feature Extraction:

```
# Converting years featues to age feature in train dataset
df['Year_SinceBuilt'] = df['YearBuilt'].max() - df['YearBuilt']
df['Year_SinceRemodAdded'] = df['YearRemodAdd'].max() - df['YearRemodAdd']
df['Year_SinceSold'] = df['YrSold'].max() - df['YrSold']
df['GarageAge'] = df['GarageYrBlt'].max() - df['GarageYrBlt']
```

```
# Dropping old columns in train dataset
df.drop(['YearBuilt','YearRemodAdd','YrSold','GarageYrBlt'], axis=1, inplace = True)
```

```
# Converting years column to age column in test dataset
test_data['Year_SinceBuilt'] = test_data['YearBuilt'].max() - test_data['YearBuilt']
test_data['Year_SinceRemodAdded'] = test_data['YearRemodAdd'].max() - test_data['YearRemodAdd']
test_data['Year_SinceSold'] = test_data['YrSold'].max() - test_data['YrSold']
test_data['GarageAge'] = test_data['GarageYrBlt'].max() - test_data['GarageYrBlt']
```

We have dropped or remove some features which are showing the same information which are other features are inferencing, so here dropped then in further data investigation.

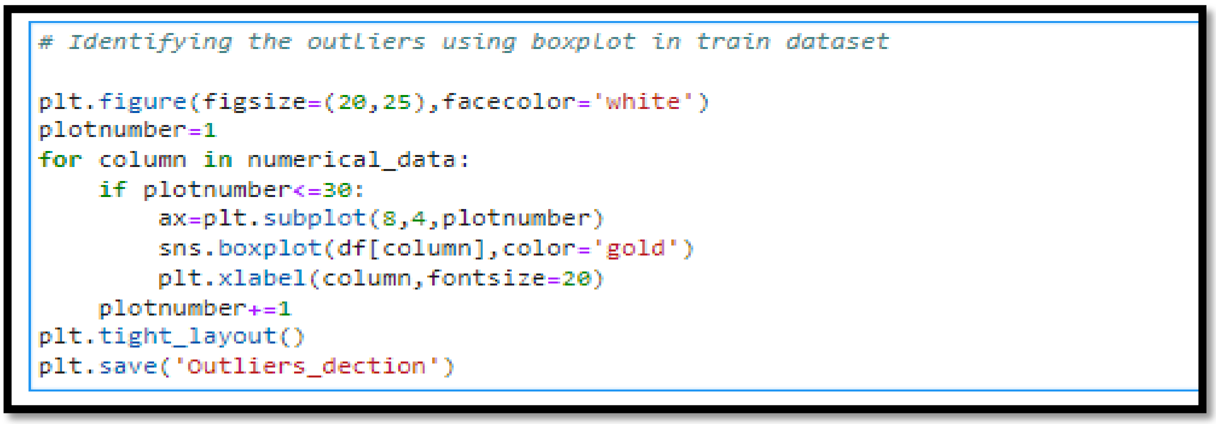


```
#Dropping unnecessary columns in train dataset
df = df.drop(["Alley"],axis=1)
df = df.drop(["PoolQC"],axis=1)
df = df.drop(["Fence"],axis=1)
df = df.drop(["MiscFeature"],axis=1)
```

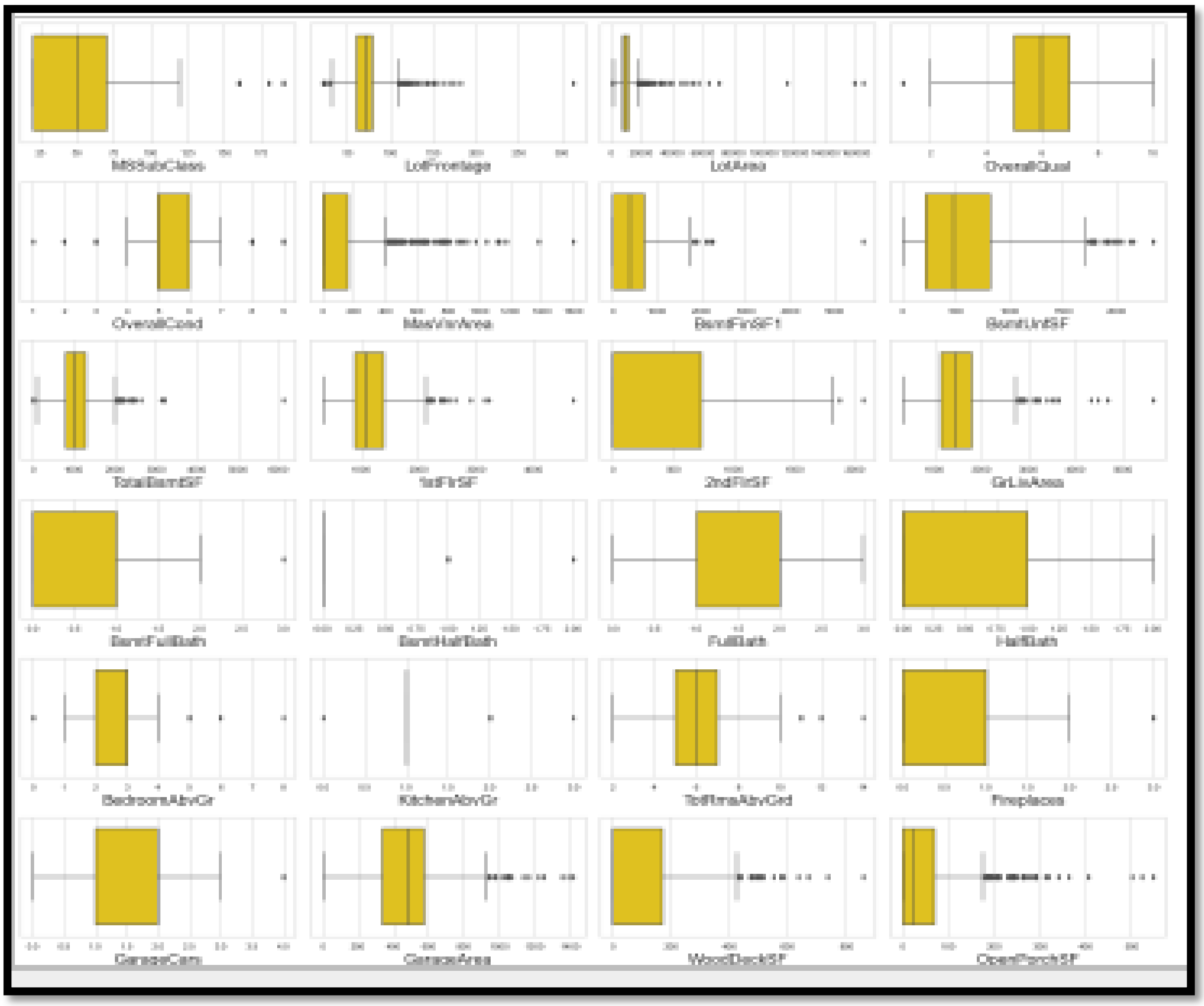
```
df = df.drop(["Id"],axis=1)
df = df.drop(["Utilities"],axis=1)
```

```
#Dropping unnecessary columns in train dataset
df.drop(columns = ['BsmtFinSF2','LowQualFinSF','EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal'],inplace = True)
```

Outlier Detection and removal:



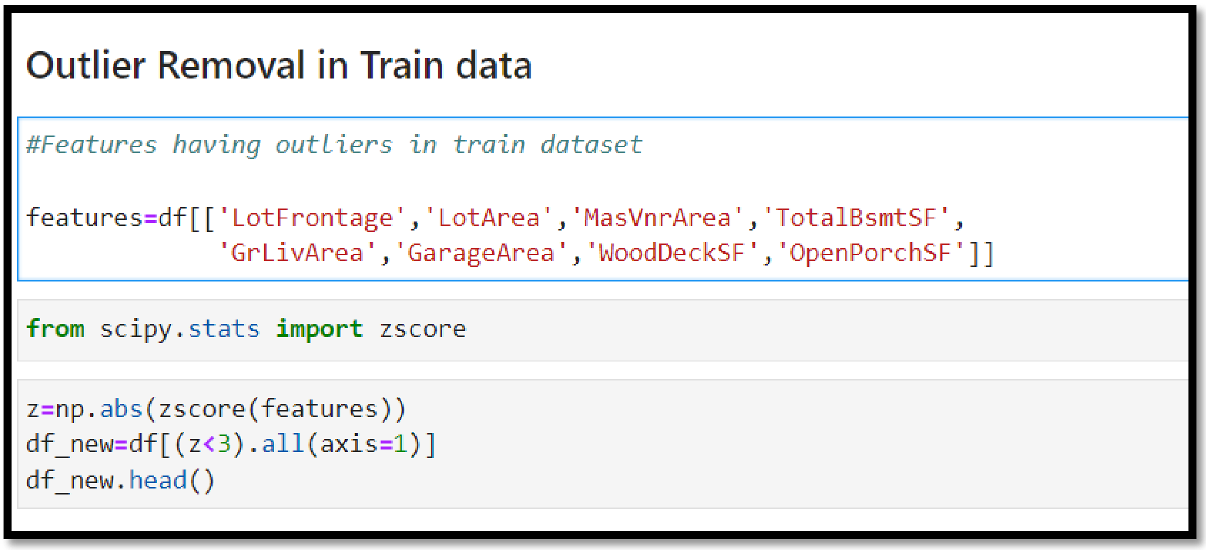
```
# Identifying the outliers using boxplot in train dataset

plt.figure(figsize=(20,25),facecolor='white')
plotnumber=1
for column in numerical_data:
    if plotnumber<=30:
        ax=plt.subplot(8,4,plotnumber)
        sns.boxplot(df[column],color='gold')
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.tight_layout()
plt.save('Outliers_dection')
```



We can see lot of features contain outliers which are in upper bound limit i.e. extreme outliers. We also know that value present in dataset are realsitic in nature, so to gain accuracy of model we cannot afford large data loss.

- BsmtFinSF1: Type 1 finished square feet
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area

TotalBsmtSF is sum of above remaining features. We will drop other three features for modelling.

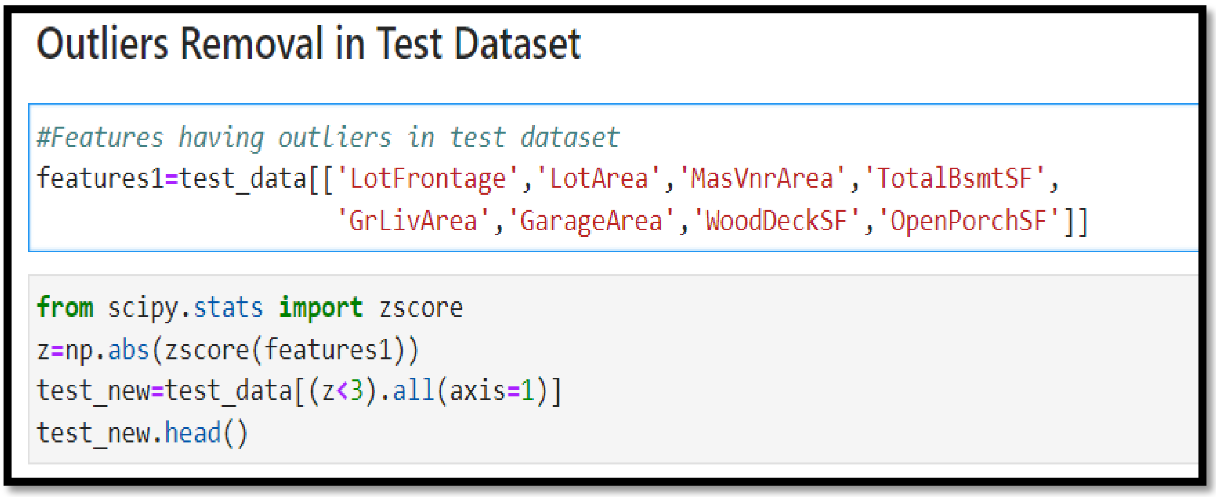


Outlier Removal in Train data

```
#Features having outliers in train dataset

features=df[['LotFrontage','LotArea','MasVnrArea','TotalBsmtSF',
             'GrLivArea','GarageArea','WoodDeckSF','OpenPorchSF']]
```

```
from scipy.stats import zscore
```

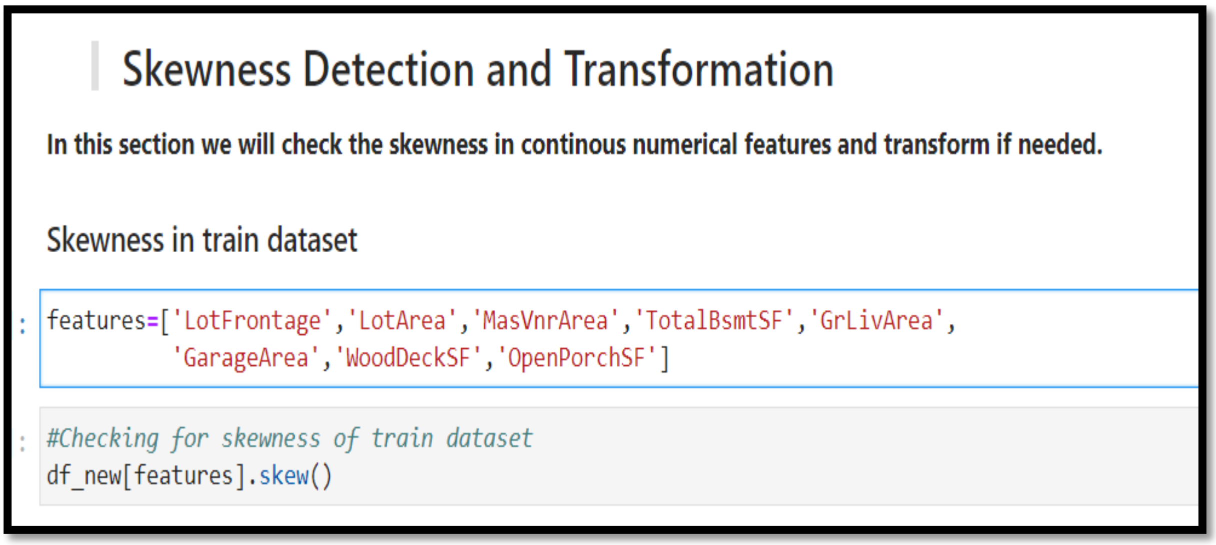
```
z=np.abs(zscore(features))
df_new=df[(z<3).all(axis=1)]
df_new.head()
```



Outliers Removal in Test Dataset

```
#Features having outliers in test dataset
features1=test_data[['LotFrontage','LotArea','MasVnrArea','TotalBsmtSF',
                     'GrLivArea','GarageArea','WoodDeckSF','OpenPorchSF']]
```

```
from scipy.stats import zscore
z=np.abs(zscore(features1))
test_new=test_data[(z<3).all(axis=1)]
test_new.head()
```



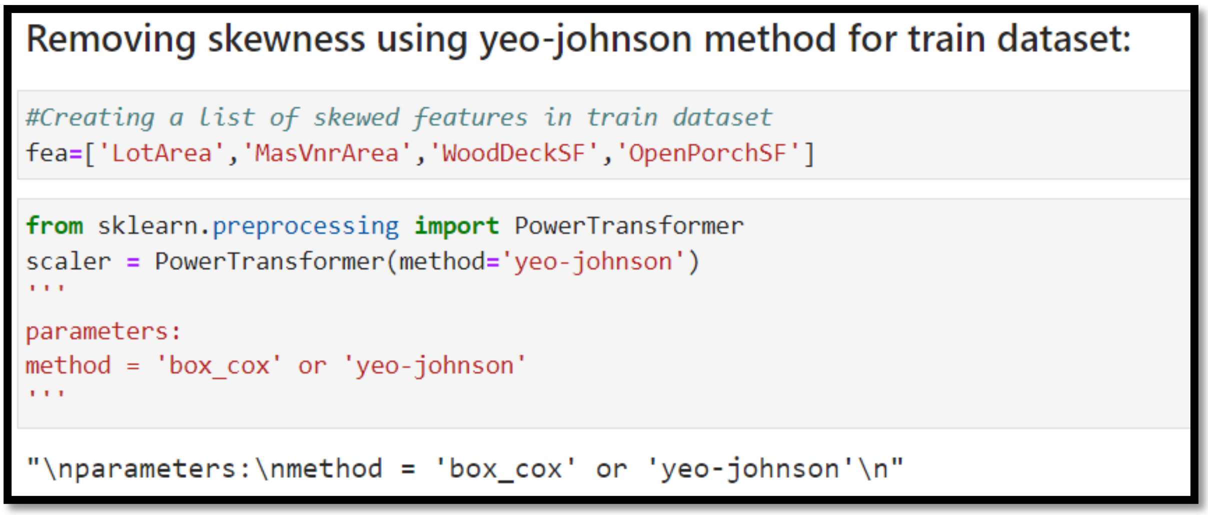
Skewness Detection and Transformation

In this section we will check the skewness in continous numerical features and transform if needed.

Skewness in train dataset

```
features=['LotFrontage','LotArea','MasVnrArea','TotalBsmtSF','GrLivArea',
          'GarageArea','WoodDeckSF','OpenPorchSF']
```

```
#Checking for skewness of train dataset
df_new[features].skew()
```

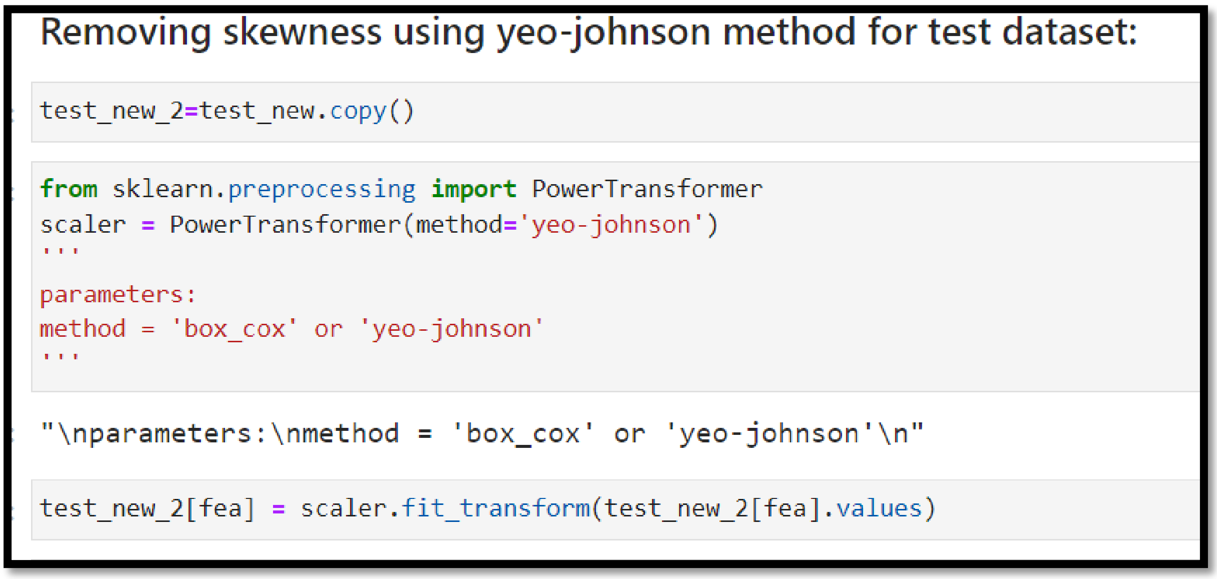


Removing skewness using yeo-johnson method for train dataset:

```
#Creating a list of skewed features in train dataset
fea=['LotArea','MasVnrArea','WoodDeckSF','OpenPorchSF']
```

```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
'''
parameters:
method = 'box_cox' or 'yeo-johnson'
'''
```

```
"\nparameters:\nmethod = 'box_cox' or 'yeo-johnson'\n"
```



Removing skewness using yeo-johnson method for test dataset:

```
test_new_2=test_new.copy()
```

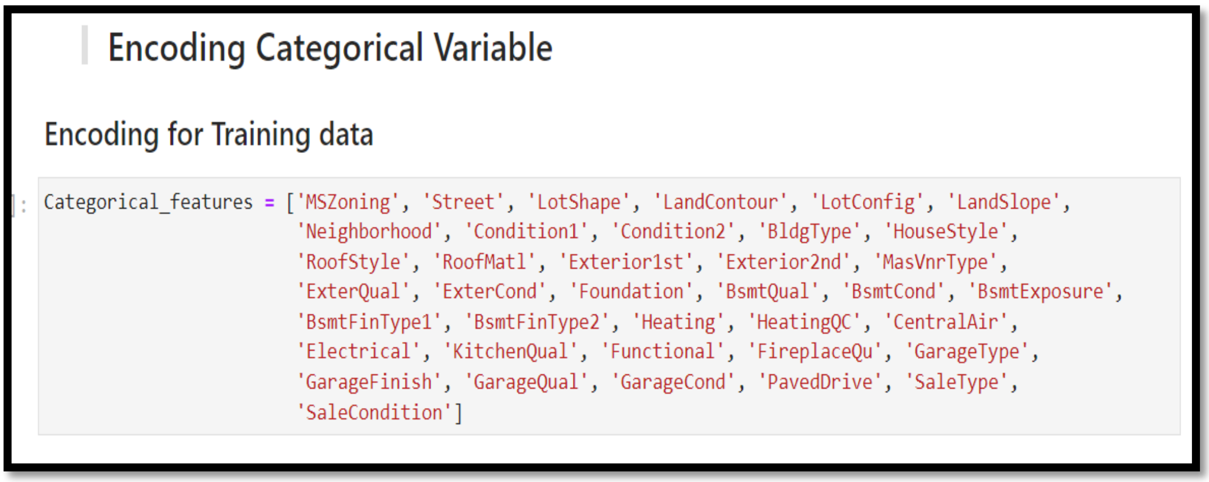
```
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
'''
parameters:
method = 'box_cox' or 'yeo-johnson'
'''
```

```
"\nparameters:\nmethod = 'box_cox' or 'yeo-johnson'\n"
```

```
test_new_2[fea] = scaler.fit_transform(test_new_2[fea].values)
```

Label Encoding of Categorical features:

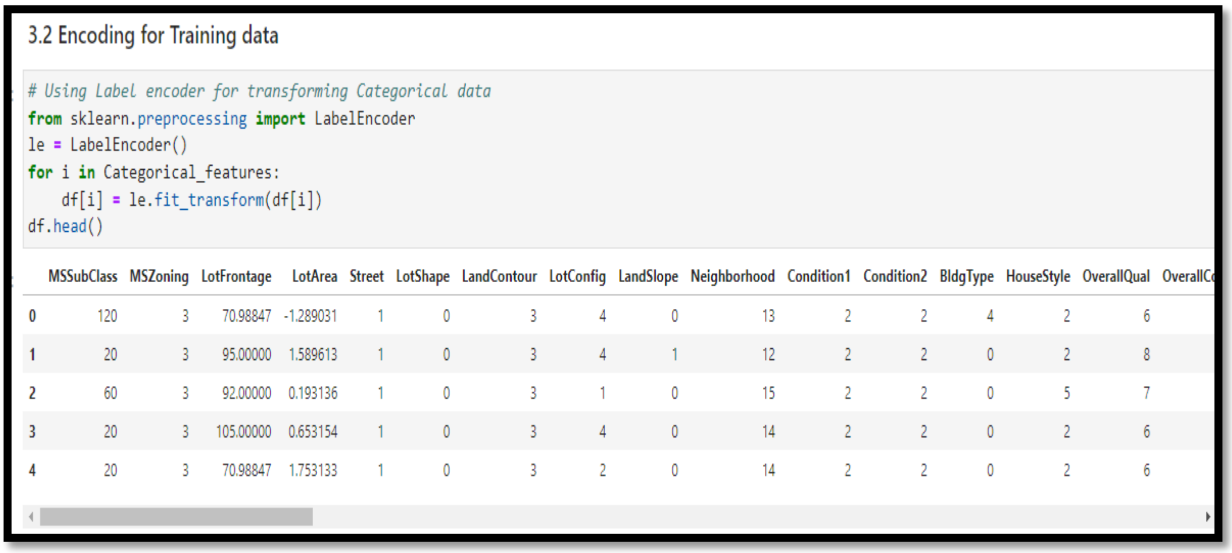
The categorical Variable in training & testing dataset are converted into numerical datatype using label encoder from scikit library.



Encoding Categorical Variable

Encoding for Training data

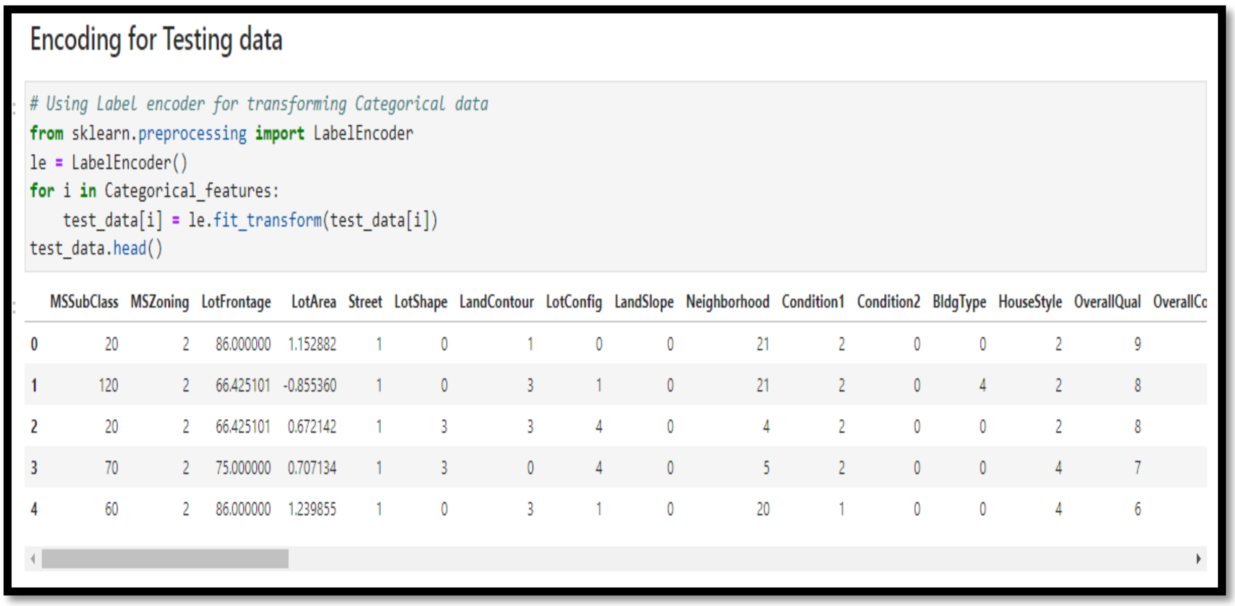
```
Categorical_features = ['MSZoning', 'Street', 'LotShape', 'LandContour', 'LotConfig', 'LandSlope',
                        'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
                        'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
                        'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
                        'BsmtFinType1', 'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir',
                        'Electrical', 'KitchenQual', 'Functional', 'FireplaceQu', 'GarageType',
                        'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType',
                        'SaleCondition']
```



3.2 Encoding for Training data

```
# Using Label encoder for transforming Categorical data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Categorical_features:
    df[i] = le.fit_transform(df[i])
df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle	OverallQual	OverallCo
0	120	3	70.98847	-1.289031	1	0	3	4	0	13	2	2	4	2	6	
1	20	3	95.00000	1.589613	1	0	3	4	1	12	2	2	0	2	8	
2	60	3	92.00000	0.193136	1	0	3	1	0	15	2	2	0	5	7	
3	20	3	105.00000	0.653154	1	0	3	4	0	14	2	2	0	2	6	
4	20	3	70.98847	1.753133	1	0	3	2	0	14	2	2	0	2	6	

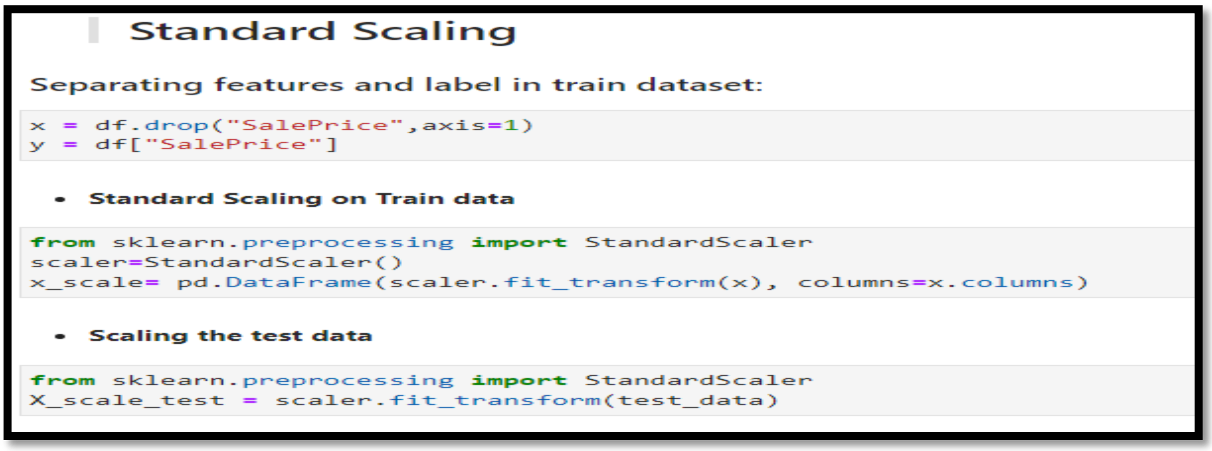


Encoding for Testing data

```
# Using Label encoder for transforming Categorical data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Categorical_features:
    test_data[i] = le.fit_transform(test_data[i])
test_data.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle	OverallQual	OverallCo
0	20	2	86.000000	1.152882	1	0	1	0	0	21	2	0	0	2	9	
1	120	2	66.425101	-0.855360	1	0	3	1	0	21	2	0	4	2	8	
2	20	2	66.425101	0.672142	1	3	3	4	0	4	2	0	0	2	8	
3	70	2	75.000000	0.707134	1	3	0	4	0	5	2	0	0	4	7	
4	60	2	86.000000	1.239855	1	0	3	1	0	20	1	0	0	4	6	

Standard Scaling:



Standard Scaling

Separating features and label in train dataset:

```
x = df.drop("SalePrice",axis=1)
y = df["SalePrice"]
```

- Standard Scaling on Train data

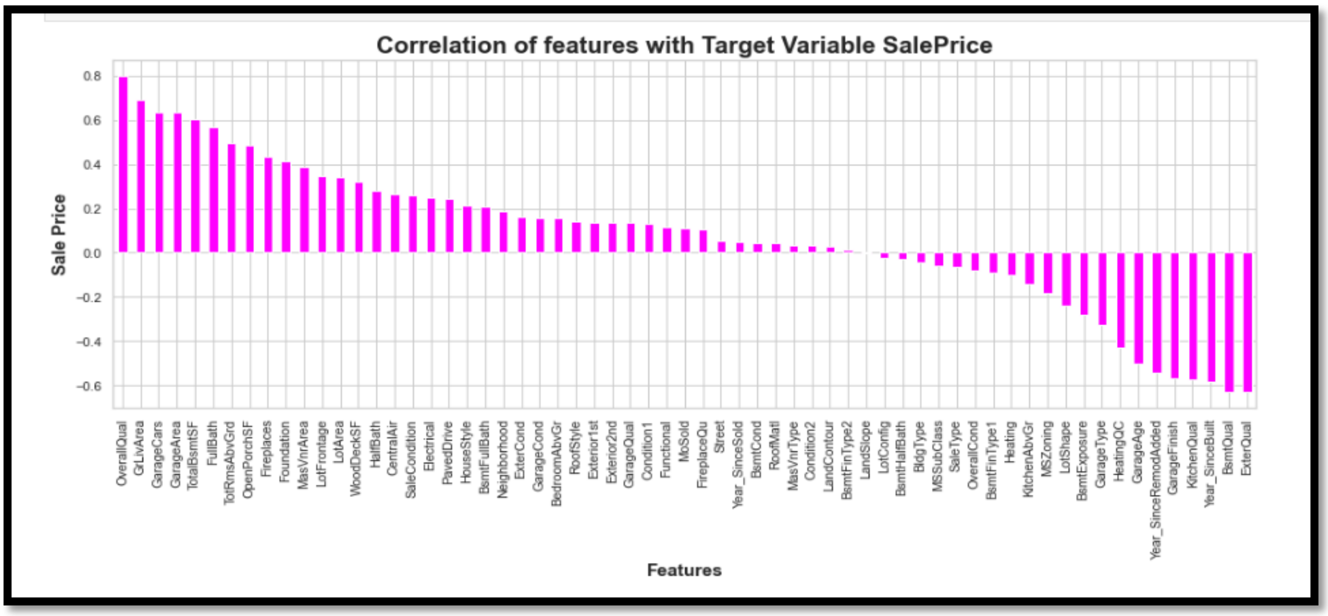
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_scale= pd.DataFrame(scaler.fit_transform(x), columns=x.columns)
```

- Scaling the test data

```
from sklearn.preprocessing import StandardScaler
X_scale_test = scaler.fit_transform(test_data)
```

4.Data Inputs- Logic- Output Relationships

Correlation heatmap is plotted to gain understanding of relationship between target features & independent features. We can see that lot of features are highly correlated with target variable Sale Price. To gain insights about relationship between Input & output different types of visualization are plotted which we will see in EDA section of this report.



5. Hardware & Software Requirements with Tool Used

Hardware Used –

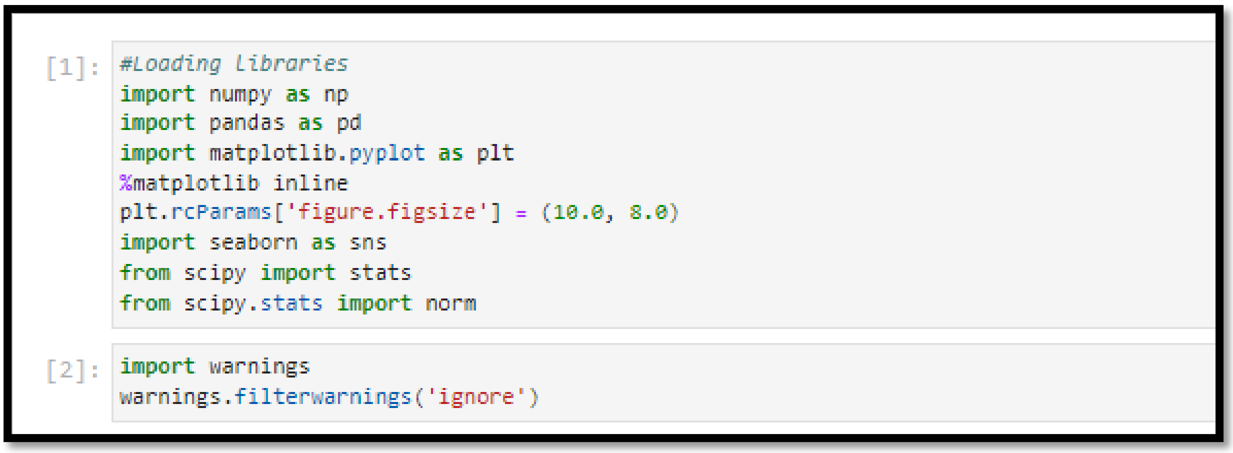
1. Processor — Intel i7 processor with 2.4GHZ
2. RAM — 4 GB
3. GPU — 2GB N Series Graphics card

Software utilized –

1.Anaconda – Jupiter Notebook/Jupiter Lab

Libraries Used –

Different libraries are used while building ML model and Visualization of data.



```
[1]: #Loading Libraries
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     %matplotlib inline
     plt.rcParams['figure.figsize'] = (10.0, 8.0)
     import seaborn as sns
     from scipy import stats
     from scipy.stats import norm

[2]: import warnings
     warnings.filterwarnings('ignore')
```

Chap. 3 Models Development & Evaluation

1. Identification of Possible Problem

Solving Approaches (Methods) Our objective is to predict house price and analyze feature impacting Sale price. This problem can be solve using regression-based machine learning algorithm like linear regression. For that purpose, first task is to convert categorical variable into numerical features. Once data encoding is done then data is scaled using standard scalar. Final model is built over this scaled data. For building ML model before implementing regression algorithm, data is split in training & test data using train_test_split from model selection module of sklearn library. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. After that model is train with various regression algorithm and 5-fold cross validation is performed. Further Hyperparameter tuning performed to build more accurate model out of best model.

2. Testing of Identified Approaches(Algorithms)

The different regression algorithm used in this project to build ML model are as below:

- ❖ Linear Regression
- ❖ Random Forest Regressor
- ❖ Decision Tree Regressor
- ❖ XG Boost Regression
- ❖ Gradient Boosting Regressor
- ❖ Extra Tree Regressor

3. KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

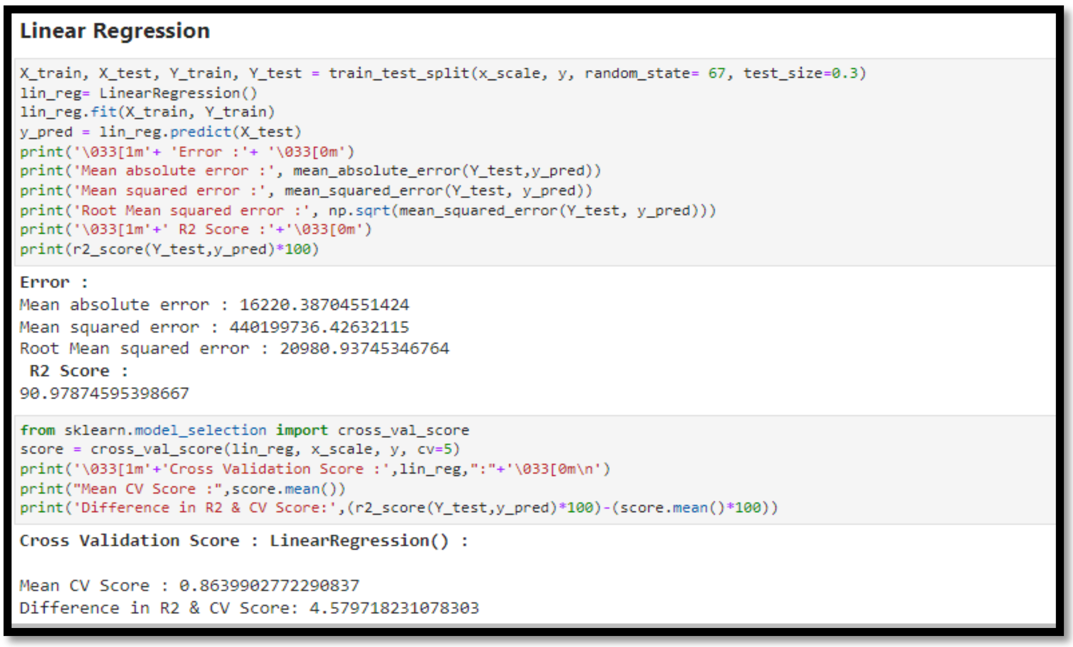
Following metrics used for evaluation:

1. Mean absolute error which gives magnitude of difference between the prediction of an observation and the true value of that observation.

2. Root mean square error is one of the most commonly used measures for evaluating the quality of predictions.
3. R2 score which tells us how accurate our model predict result, is going to important evaluation criteria along with Cross validation score. 4. Cross Validation Score.

4. RUN AND EVALUATE SELECTED MODELS

1. Linear Regression Model:



Linear Regression

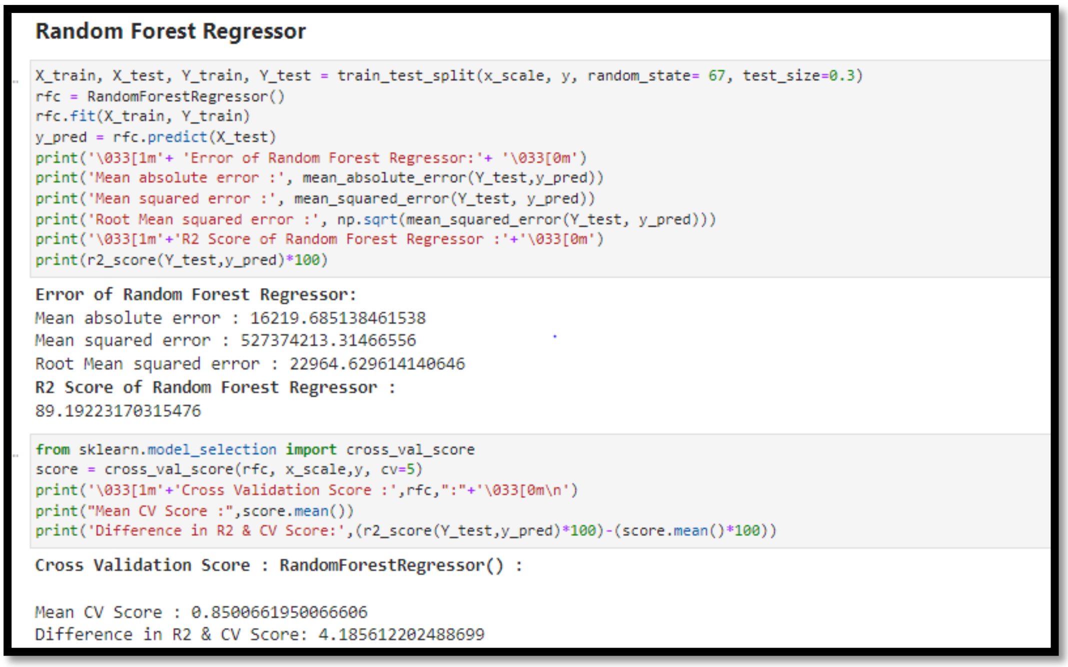
```
X_train, X_test, Y_train, Y_test = train_test_split(x_scale, y, random_state= 67, test_size=0.3)
lin_reg= LinearRegression()
lin_reg.fit(X_train, Y_train)
y_pred = lin_reg.predict(X_test)
print('\033[1m'+ 'Error :'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+' R2 Score :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error :
Mean absolute error : 16220.38704551424
Mean squared error : 440199736.42632115
Root Mean squared error : 20980.93745346764
 R2 Score :
90.97874595398667
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(lin_reg, x_scale, y, cv=5)
print('\033[1m'+'Cross Validation Score :',lin_reg,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : LinearRegression() :

Mean CV Score : 0.8639902772290837
Difference in R2 & CV Score: 4.579718231078303
```

2. Random Forest Regressor:



Random Forest Regressor

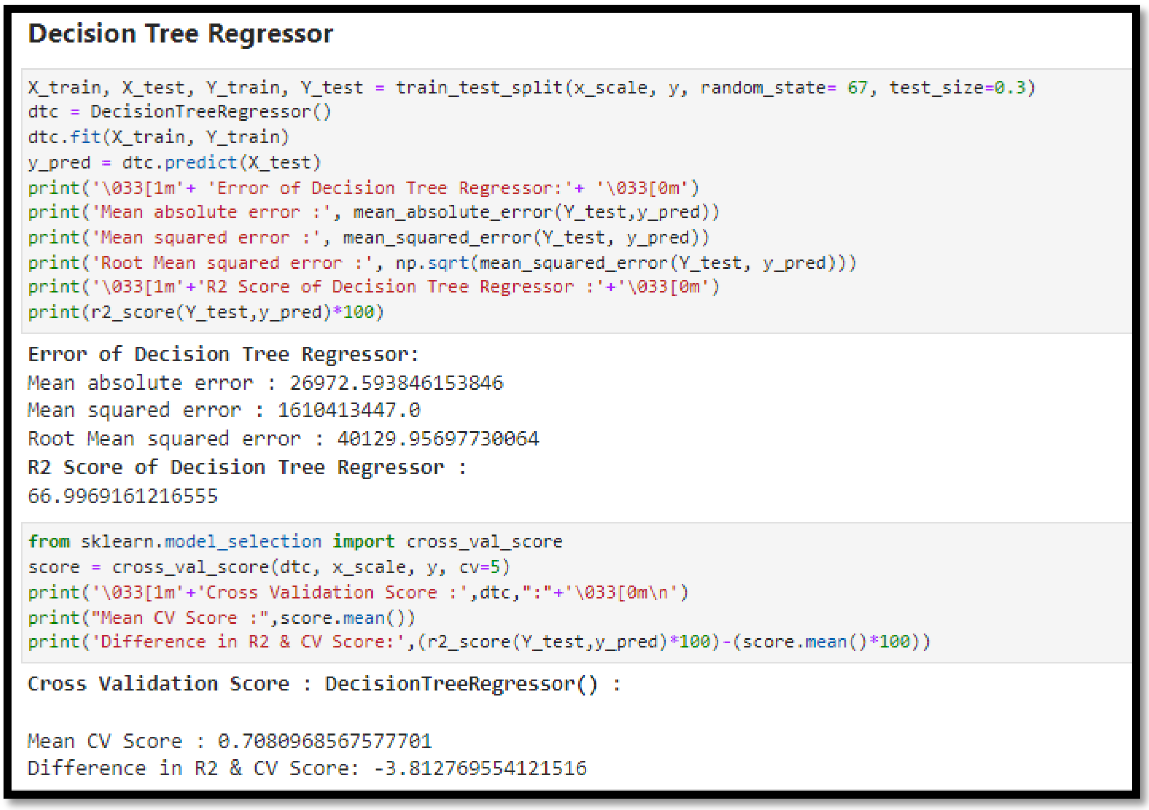
```
X_train, X_test, Y_train, Y_test = train_test_split(x_scale, y, random_state= 67, test_size=0.3)
rfc = RandomForestRegressor()
rfc.fit(X_train, Y_train)
y_pred = rfc.predict(X_test)
print('\033[1m'+ 'Error of Random Forest Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Random Forest Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Random Forest Regressor:
Mean absolute error : 16219.685138461538
Mean squared error : 527374213.31466556
Root Mean squared error : 22964.629614140646
R2 Score of Random Forest Regressor :
89.19223170315476
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(rfc, x_scale,y, cv=5)
print('\033[1m'+'Cross Validation Score :',rfc,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : RandomForestRegressor() :

Mean CV Score : 0.8500661950066606
Difference in R2 & CV Score: 4.185612202488699
```

3. Decision Tree Regressor:



Decision Tree Regressor

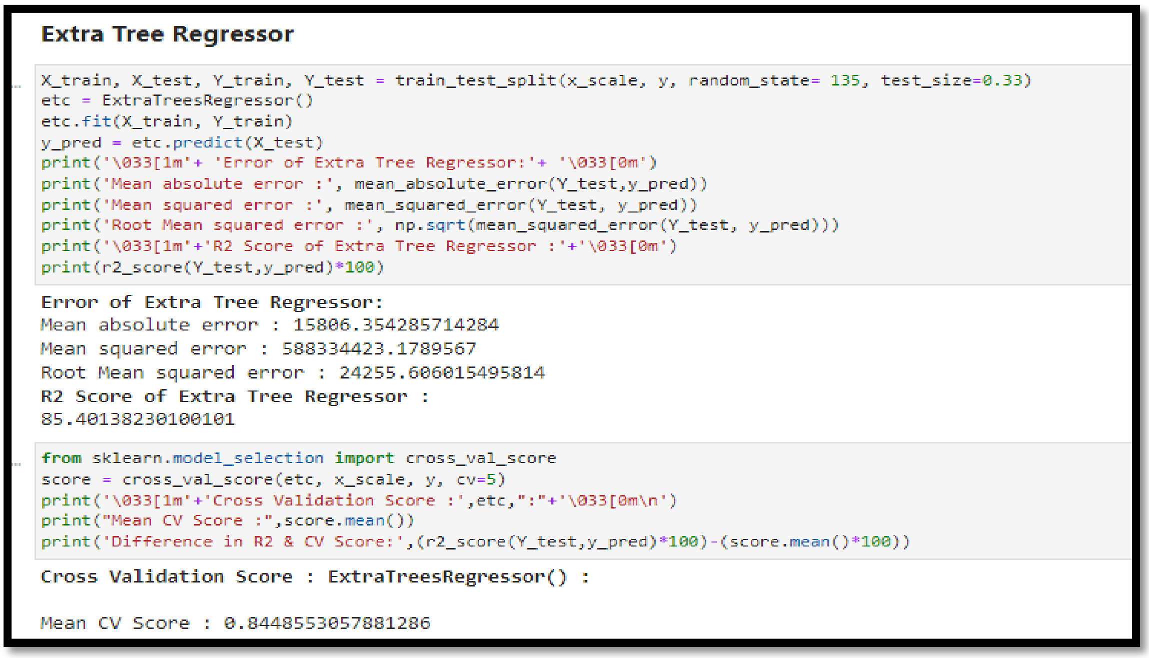
```
X_train, X_test, Y_train, Y_test = train_test_split(x_scale, y, random_state= 67, test_size=0.3)
dtc = DecisionTreeRegressor()
dtc.fit(X_train, Y_train)
y_pred = dtc.predict(X_test)
print('\033[1m'+ 'Error of Decision Tree Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Decision Tree Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Decision Tree Regressor:
Mean absolute error : 26972.593846153846
Mean squared error : 1610413447.0
Root Mean squared error : 40129.95697730064
R2 Score of Decision Tree Regressor :
66.9969161216555
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(dtc, x_scale, y, cv=5)
print('\033[1m'+'Cross Validation Score :',dtc,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : DecisionTreeRegressor() :

Mean CV Score : 0.7080968567577701
Difference in R2 & CV Score: -3.812769554121516
```

4. Extra Trees Regressor:



Extra Tree Regressor

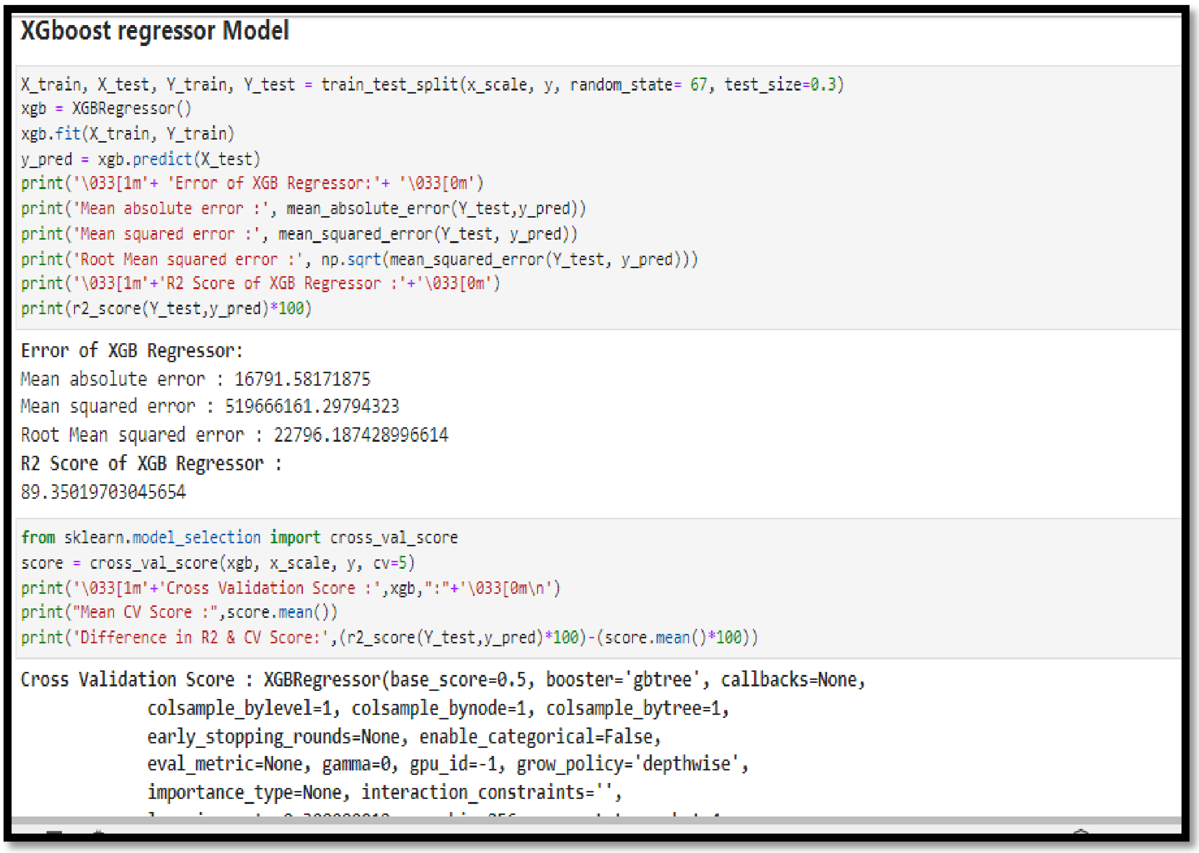
```
X_train, X_test, Y_train, Y_test = train_test_split(x_scale, y, random_state= 135, test_size=0.33)
etc = ExtraTreesRegressor()
etc.fit(X_train, Y_train)
y_pred = etc.predict(X_test)
print('\033[1m'+ 'Error of Extra Tree Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Extra Tree Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Extra Tree Regressor:
Mean absolute error : 15806.354285714284
Mean squared error : 588334423.1789567
Root Mean squared error : 24255.606015495814
R2 Score of Extra Tree Regressor :
85.40138230100101
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(etc, x_scale, y, cv=5)
print('\033[1m'+'Cross Validation Score :',etc,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : ExtraTreesRegressor() :

Mean CV Score : 0.8448553057881286
```

5. XGB Regressor:



XGboost regressor Model

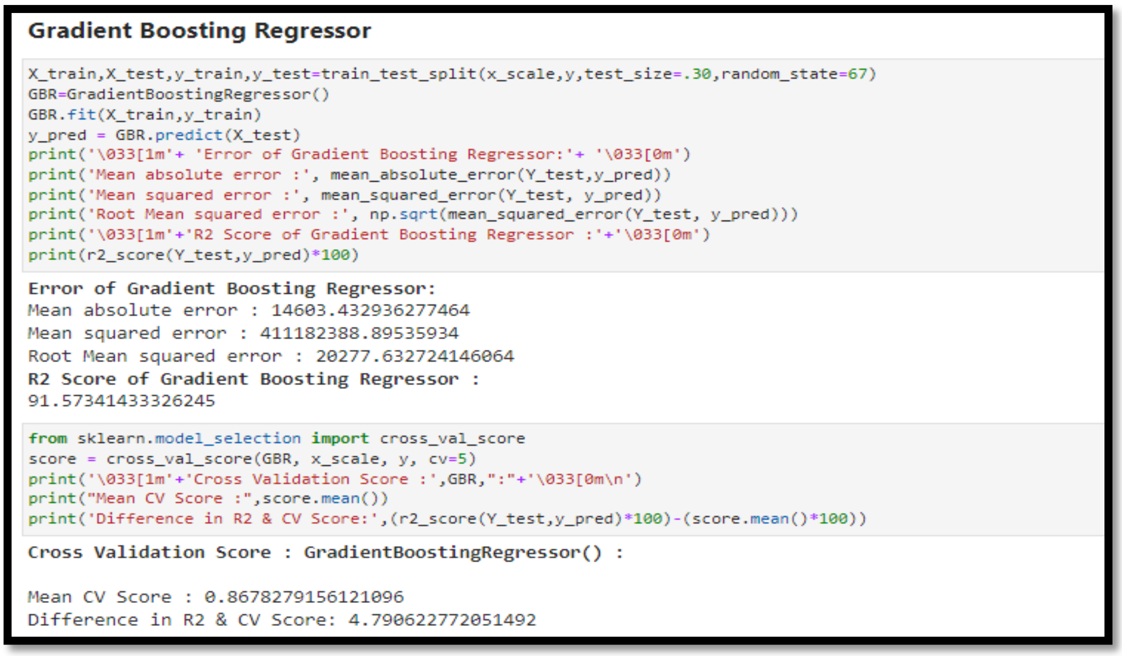
```
X_train, X_test, Y_train, Y_test = train_test_split(x_scale, y, random_state= 67, test_size=0.3)
xgb = XGBRegressor()
xgb.fit(X_train, Y_train)
y_pred = xgb.predict(X_test)
print('\033[1m'+ 'Error of XGB Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of XGB Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of XGB Regressor:
Mean absolute error : 16791.58171875
Mean squared error : 519666161.29794323
Root Mean squared error : 22796.187428996614
R2 Score of XGB Regressor :
89.35019703045654
```

```
from sklearn.model_selection import cross_val_score
score = cross_val_score(xgb, x_scale, y, cv=5)
print('\033[1m'+'Cross Validation Score :',xgb,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
```

6. Gradient Boosting Regressor:



Gradient Boosting Regressor

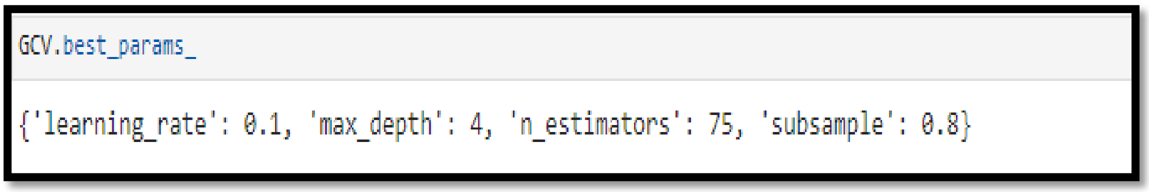
```
X_train,X_test,y_train,y_test=train_test_split(x_scale,y,test_size=.30,random_state=67)
GBR=GradientBoostingRegressor()
GBR.fit(X_train,y_train)
y_pred = GBR.predict(X_test)
print('\033[1m'+ 'Error of Gradient Boosting Regressor:'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+'R2 Score of Gradient Boosting Regressor :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error of Gradient Boosting Regressor:
Mean absolute error : 14603.432936277464
Mean squared error : 411182388.89535934
Root Mean squared error : 20277.632724146064
R2 Score of Gradient Boosting Regressor :
91.57341433326245
```

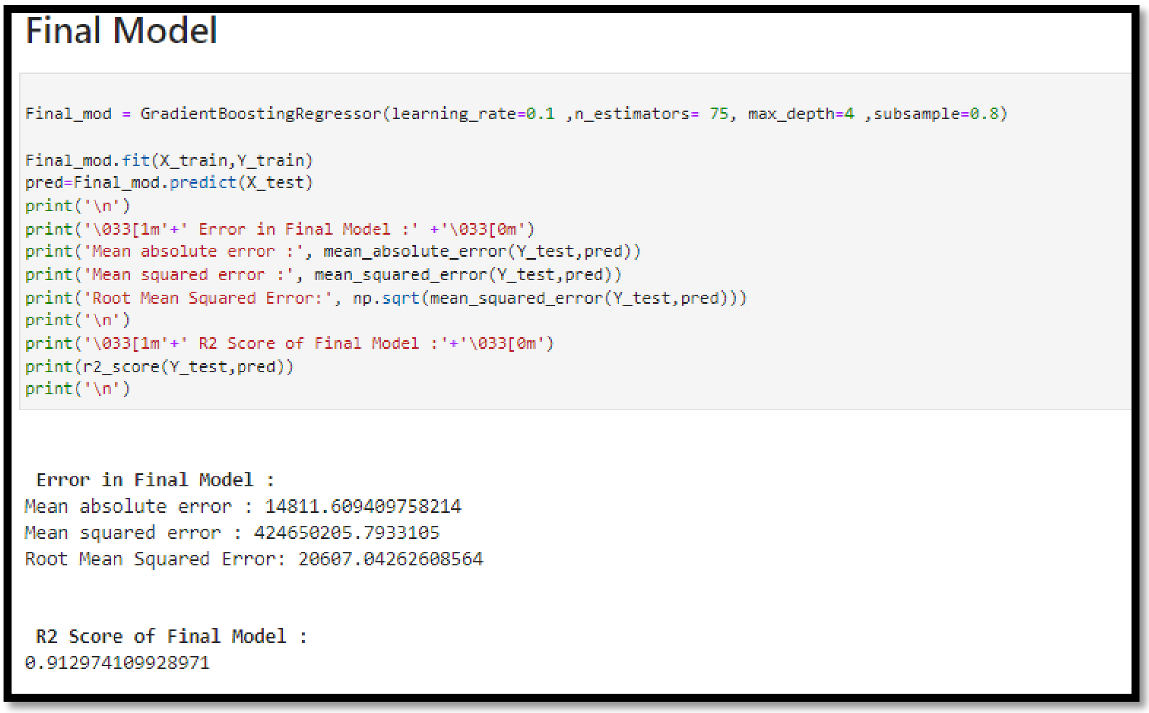
```
from sklearn.model_selection import cross_val_score
score = cross_val_score(GBR, x_scale, y, cv=5)
print('\033[1m'+'Cross Validation Score :',GBR,":"+'\033[0m\n')
print("Mean CV Score :",score.mean())
print('Difference in R2 & CV Score:',(r2_score(Y_test,y_pred)*100)-(score.mean()*100))
```

```
Cross Validation Score : GradientBoostingRegressor() :

Mean CV Score : 0.8678279156121096
Difference in R2 & CV Score: 4.790622772051492
```

Final model is built with best params got in hyper parameter tuning.



```
GCV.best_params_

{'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 75, 'subsample': 0.8}
```



Final Model

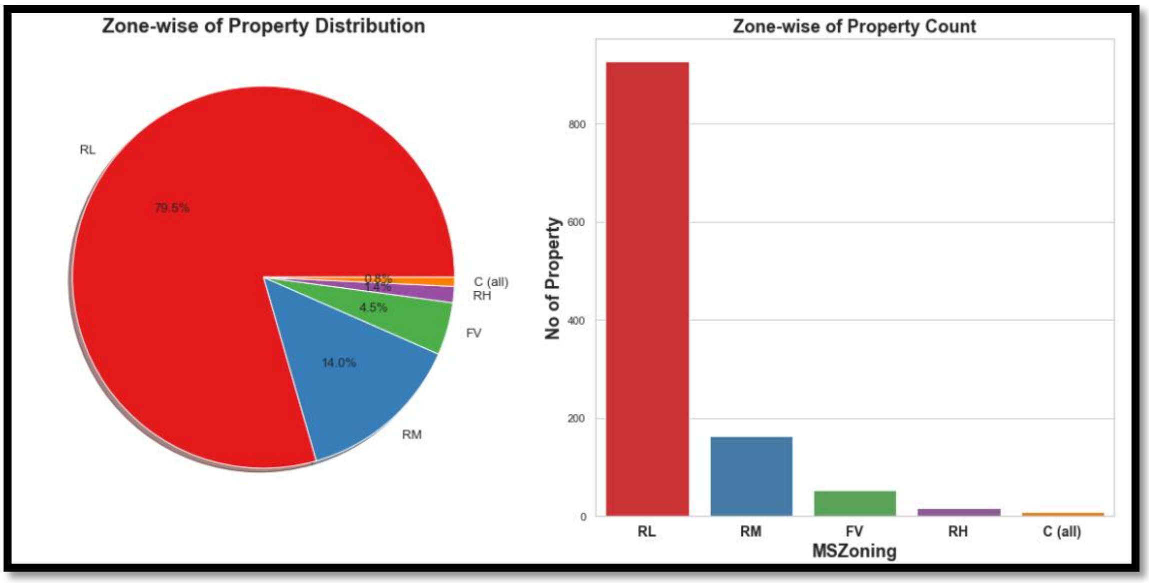
```
Final_mod = GradientBoostingRegressor(learning_rate=0.1 ,n_estimators= 75, max_depth=4 ,subsample=0.8)

Final_mod.fit(X_train,Y_train)
pred=Final_mod.predict(X_test)
print('\n')
print('\033[1m'+' Error in Final Model :' +'\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,pred))
print('Mean squared error :', mean_squared_error(Y_test,pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(Y_test,pred)))
print('\n')
print('\033[1m'+' R2 Score of Final Model :'+'\033[0m')
print(r2_score(Y_test,pred))
print('\n')
```

```


 Error in Final Model :
Mean absolute error : 14811.609409758214
Mean squared error : 424650205.7933105
Root Mean Squared Error: 20607.04262608564


 R2 Score of Final Model :
0.912974109928971
```

We can see that hyper parameter tuning leading to increase in R2 Score slightly from default model.

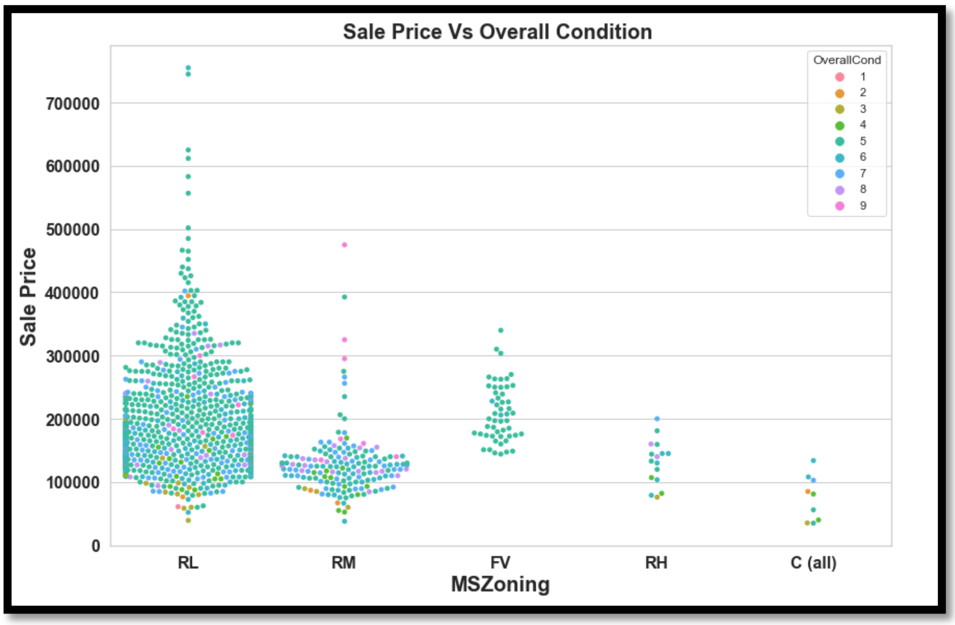
Chap. 4 Visualization

Let's see key features of Explanatory Data analysis, we will it with zone wise distribution of property.



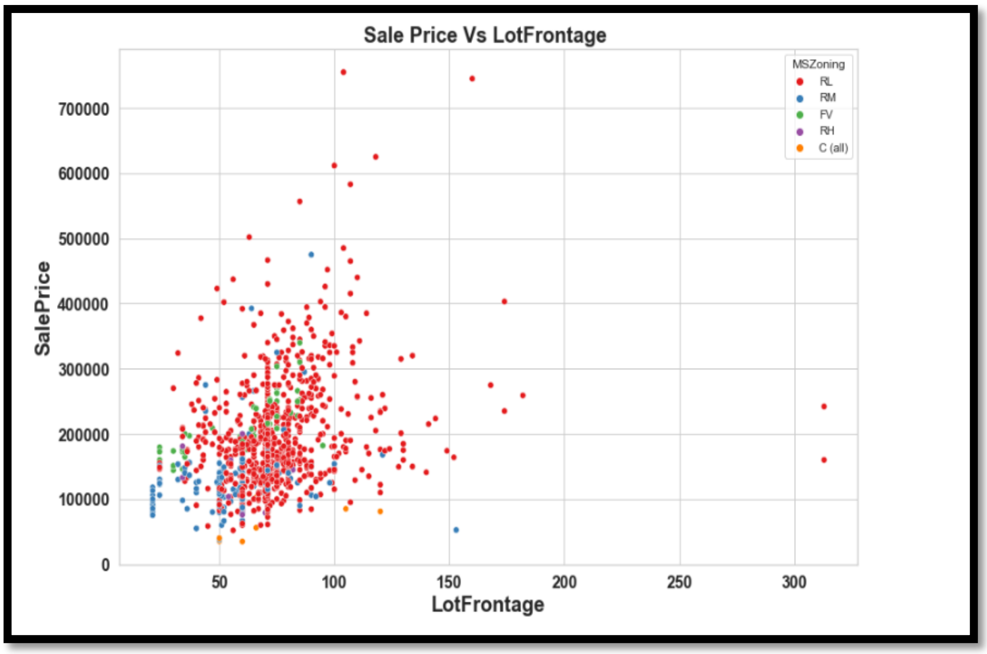
Observation:

- **79.5% of House properties belongs to Low Density Residential Area followed by 14 % of properties belong to Medium Density Residential Area.**
- **Very Few properties (0.8%) belong to Commercial zone.**

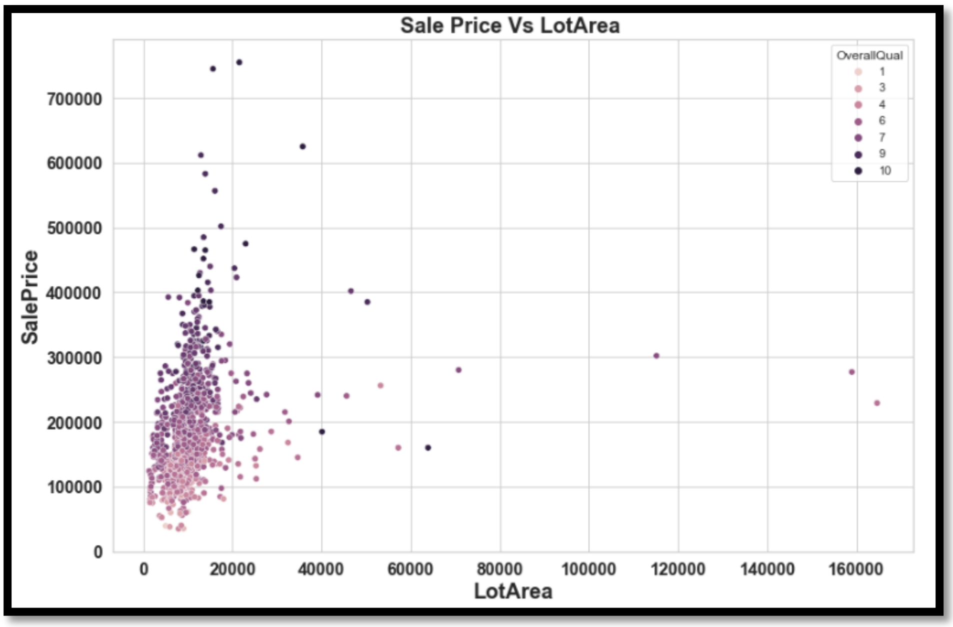


Observation:

- Most of property for sale has overall condition rating of either 5 or 6.
- We already know of 80% of housing data belongs to Low density Residential Area and now we can see in Swarm Plot that Sale Price inside RL Zone is much higher than another remaining zone.
- Cheapest properties are available in Commercial zone.
- Another interesting observation we get here is for some house properties having Overall condition Rating of 8 & 9 have low price compare to others. This indicates that Overall Condition Rating is not significant factor in determination of Sale price. Overall Condition Rating may helpful to buyer in taking decision of Buying property but not in determination of House Price.



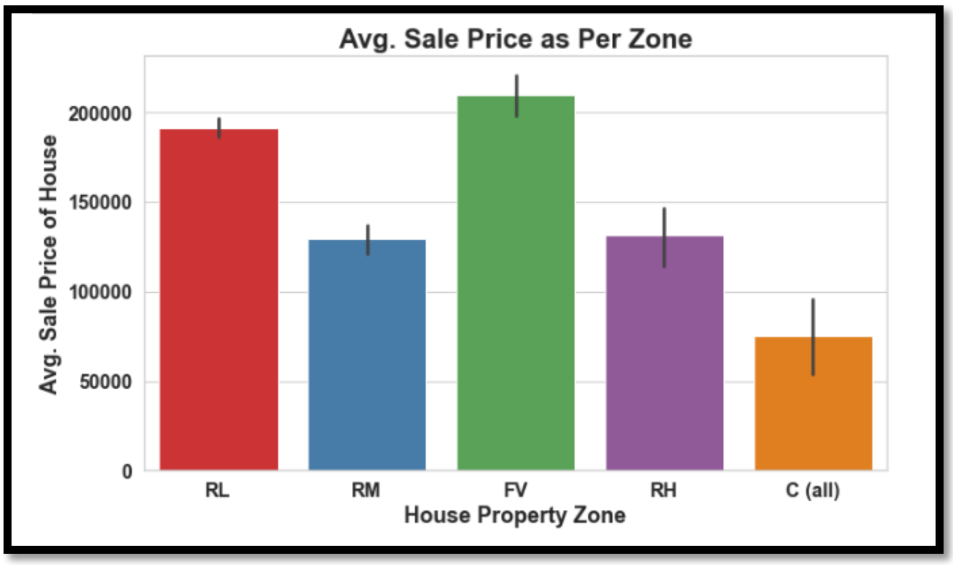
With Exception of Commercial zone, As Lot Frontage area increase (which indicates Size of street connected to property) the Sale Price increases.



Observation:

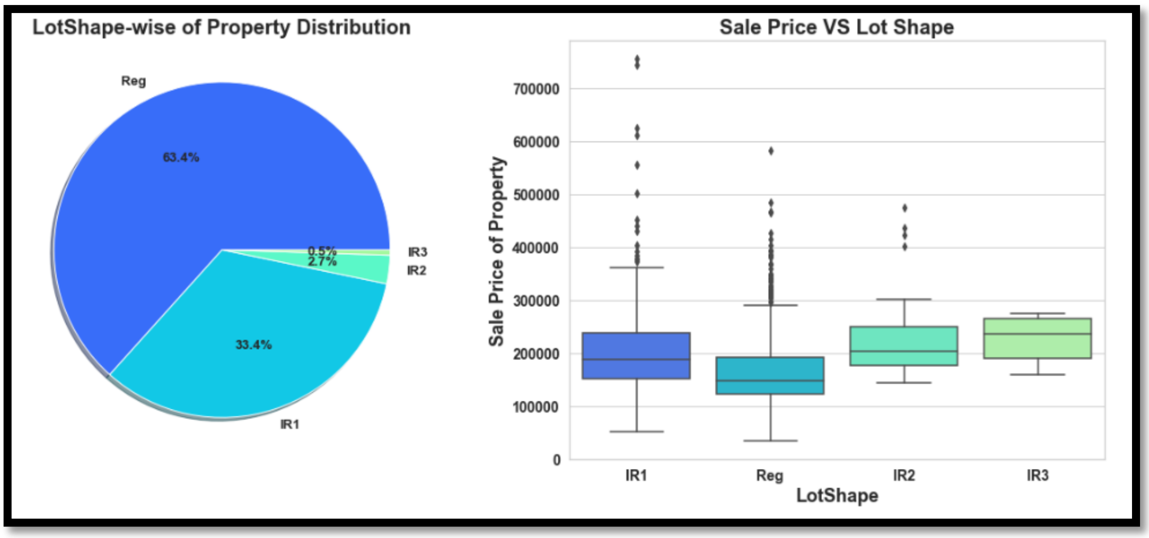
There is No Significant relationship found between Sale price & Lot area.

Here we get Important Observation that - As Overall Quality of House Increase the Sale Price of House also Increases



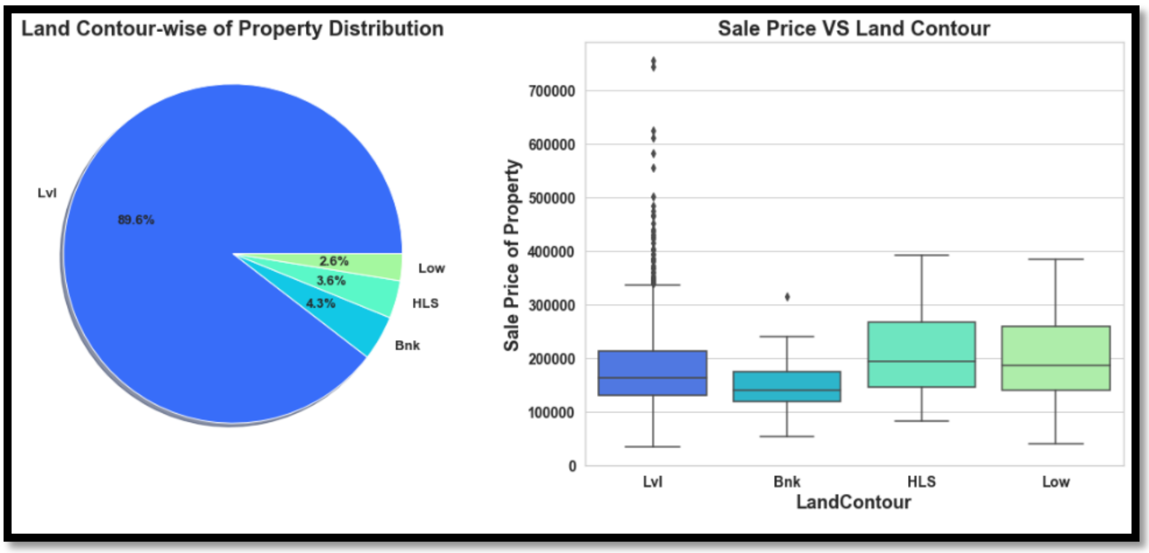
Observation:

In terms of Average Sale price house properties belonging to Floating Village Residential Zone are costlier than rest.



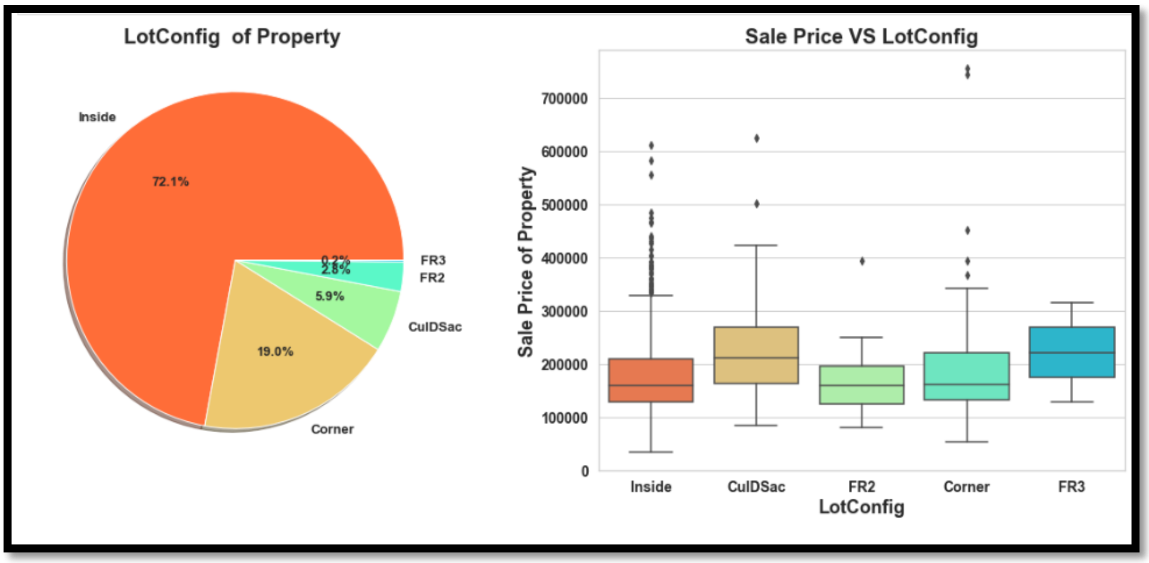
Observation:

- 63.4% house properties are regular in shape.
- Sale Price of property with slight irregular shape is higher than regular shape.



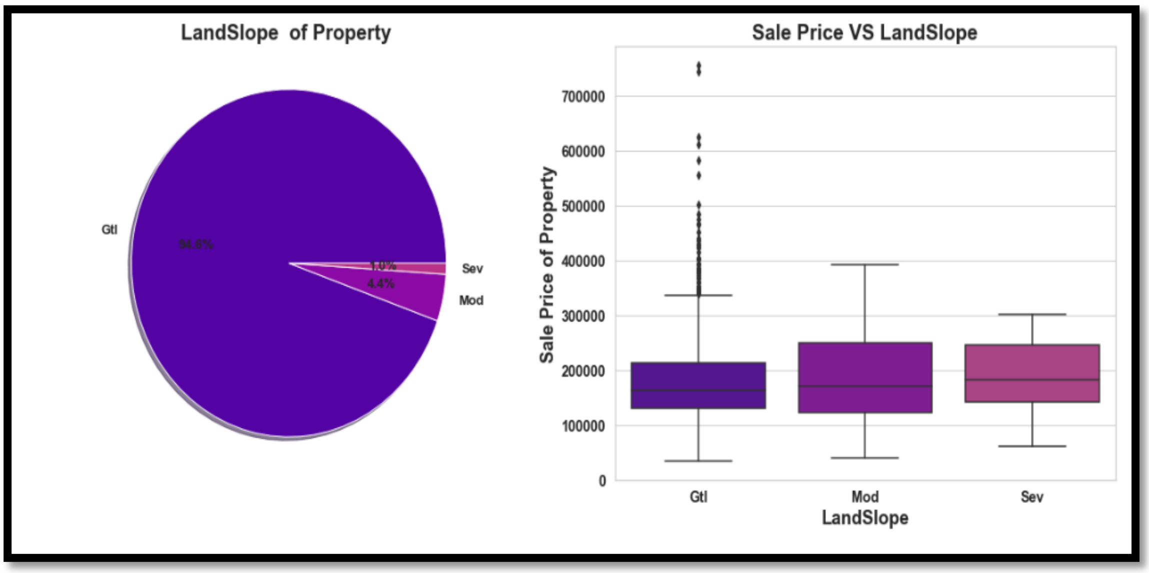
Observation:

- 89.6% of House properties are near flat level surface.
- Also, price for flat level surface house is much higher than other land contour.



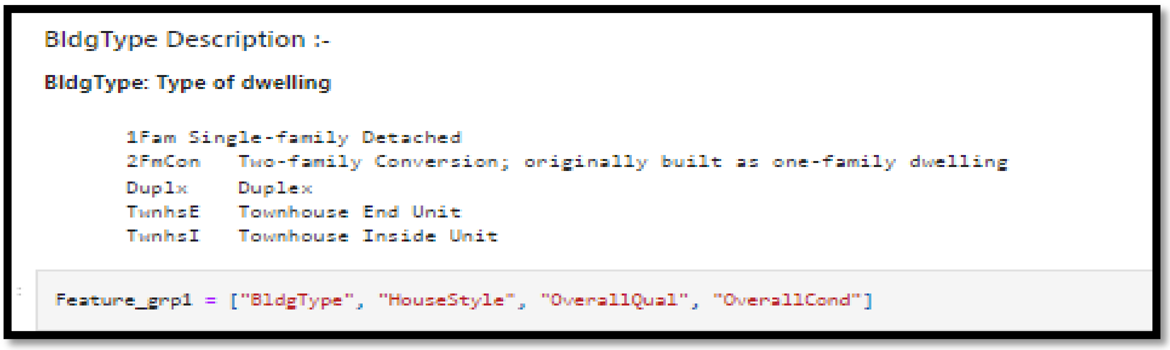
Observation :-

- Around 72 % of house comes with inside Lot configuration.
- Cul-de-sac has maximum Mean Sale Price among all lot configuration.
- Cheapest Houses belong to Inside lot configuration while Costlier houses belongs to Corner Lot Configuration



Observation :

- Clearly we can see in box plot that as Land slope increases the Sale price of house decreases.
- 1% properties come with severe slope and they come with low price compare to Gentle Slope properties.

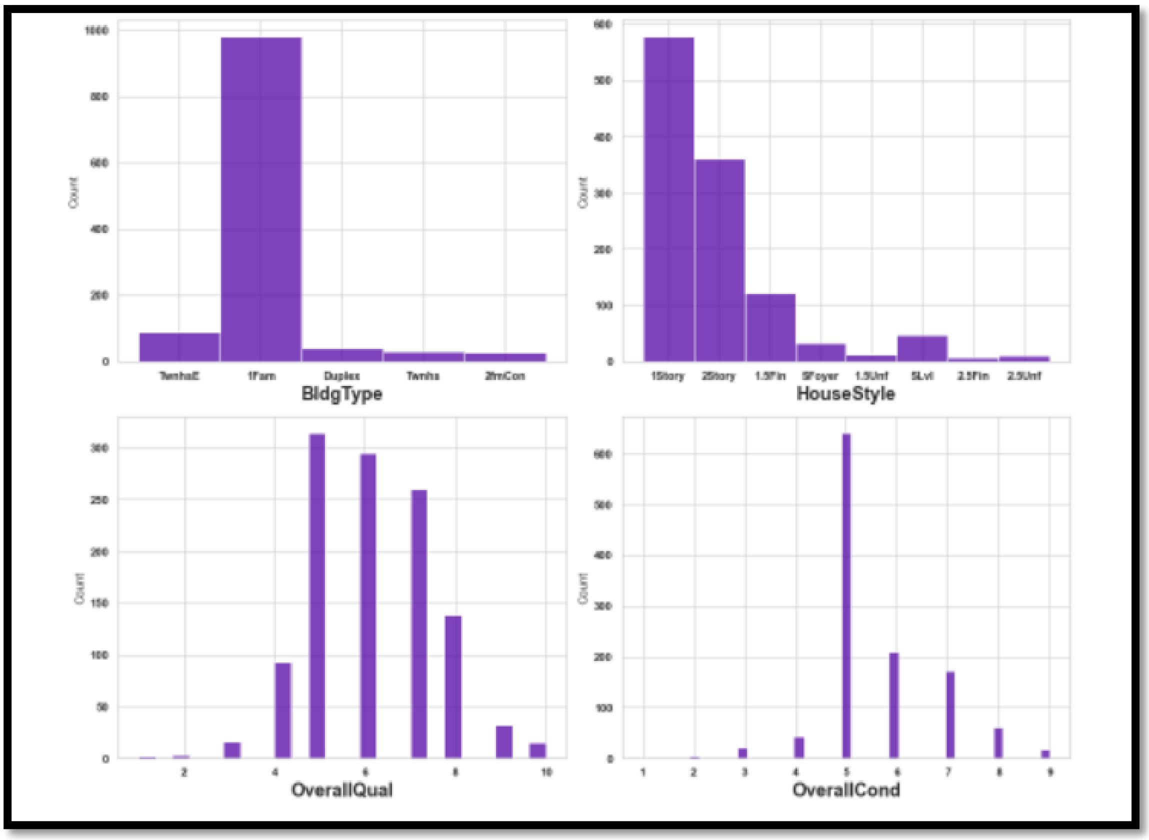


BldgType Description :-

BldgType: Type of dwelling

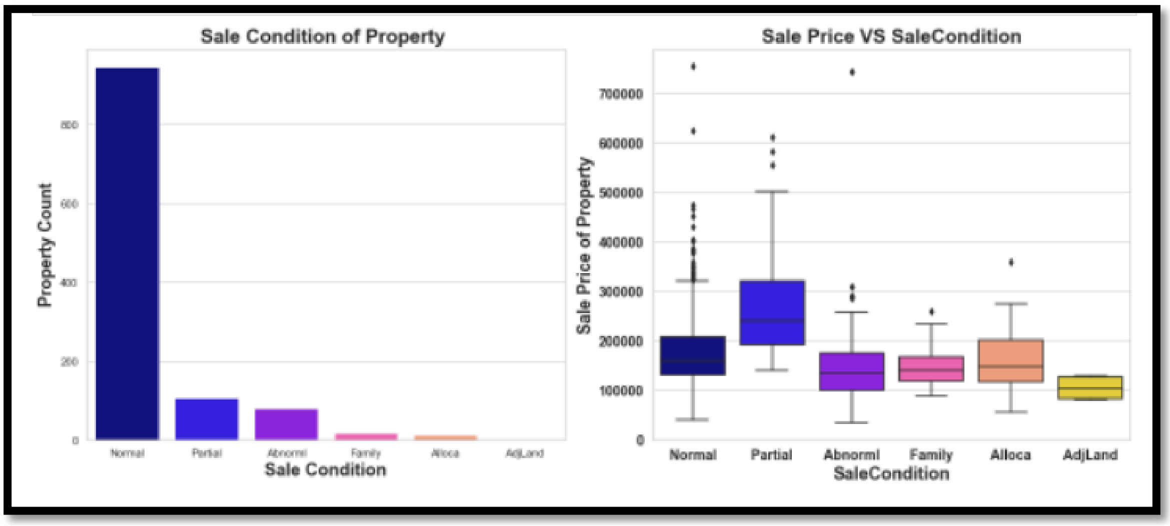
```
1Fam Single-family Detached
2FmCon   Two-family Conversion; originally built as one-family dwelling
Duplx    Duplex
TwnhsE   Townhouse End Unit
TwnhsI   Townhouse Inside Unit
```

```
Feature_grp1 = ["BldgType", "HouseStyle", "OverallQual", "OverallCond"]
```



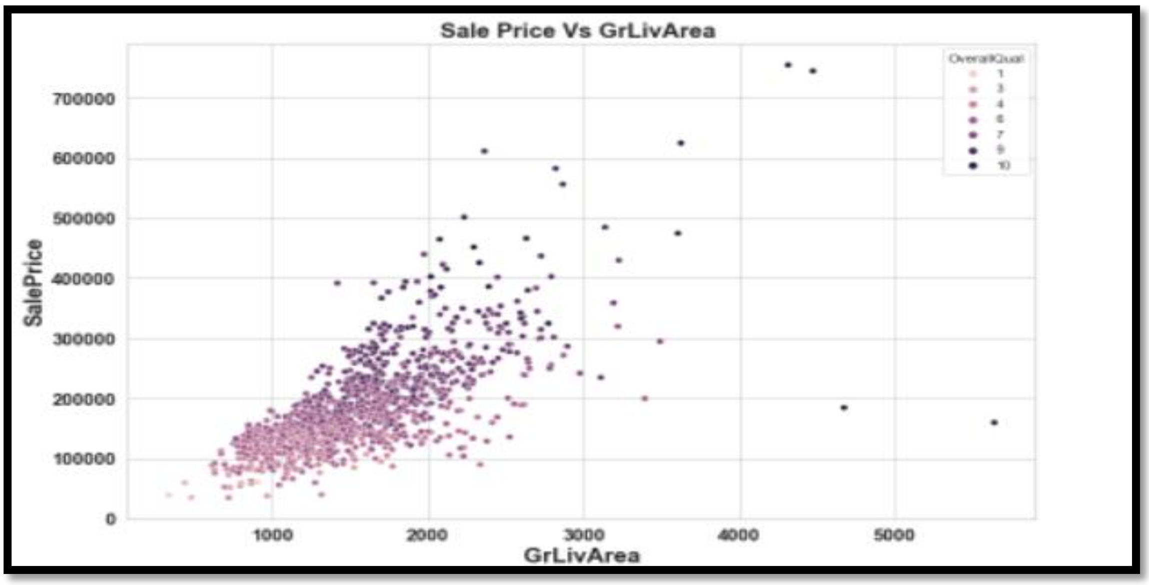
Observation :

- More than 950 house properties are with building type Single-family Detached
- More than 50% of house properties comes with Overall Condition Rating of 5.
- More than 75% of house properties come with overall Quality Rating varies between 5 to 6.
- More than 500 House Properties comes with one story dwelling.
-



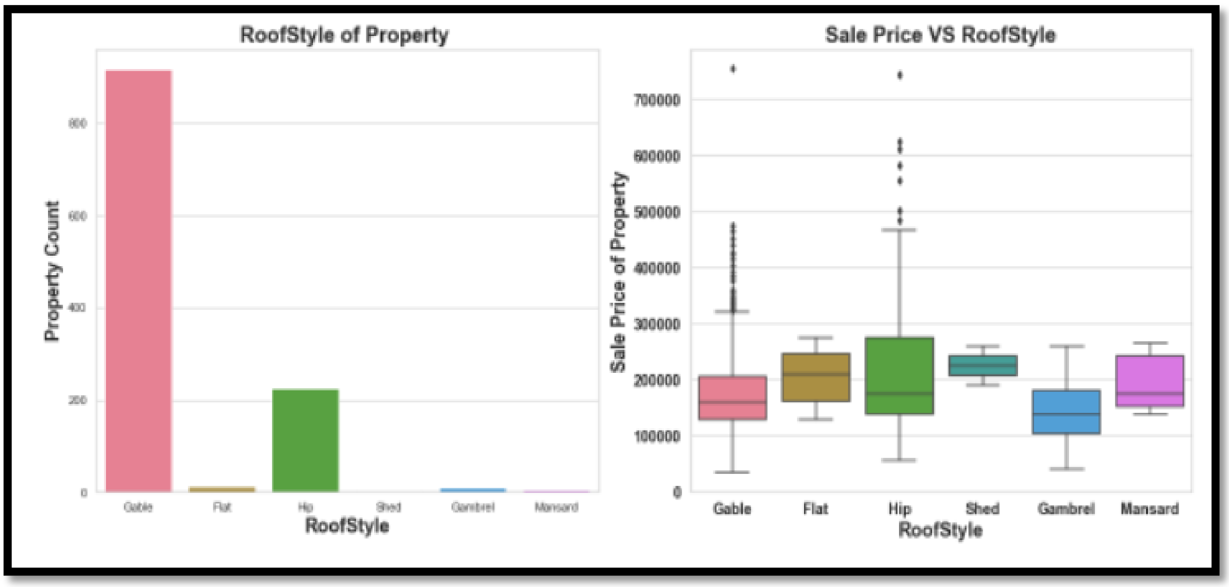
Observation :-

- We can see that Sale with condition like Abnormal, Family, Alloca, AdjLand are below the price of 300000.
- Maximum Base Price for House comes from Partial category- Home was not completed when last assessed (associated with New Homes) is higher than rest.
- Minimum base price comes from Normal condition sale and also highest sale price comes from this category.



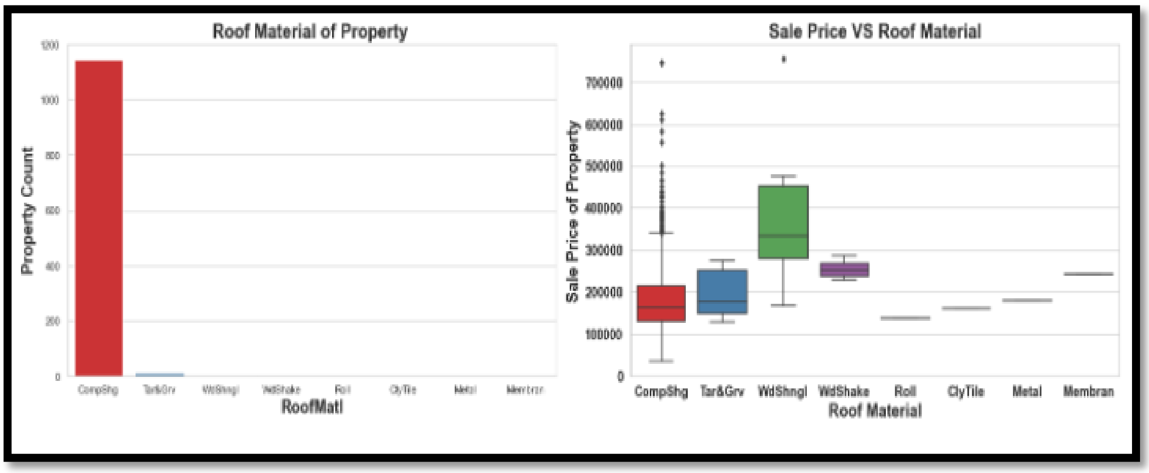
Observation :-

- In above plot we can clearly see relation between all three features very clearly.
- As total floor area increases the sale price also get increases corresponding the overall quality of House.



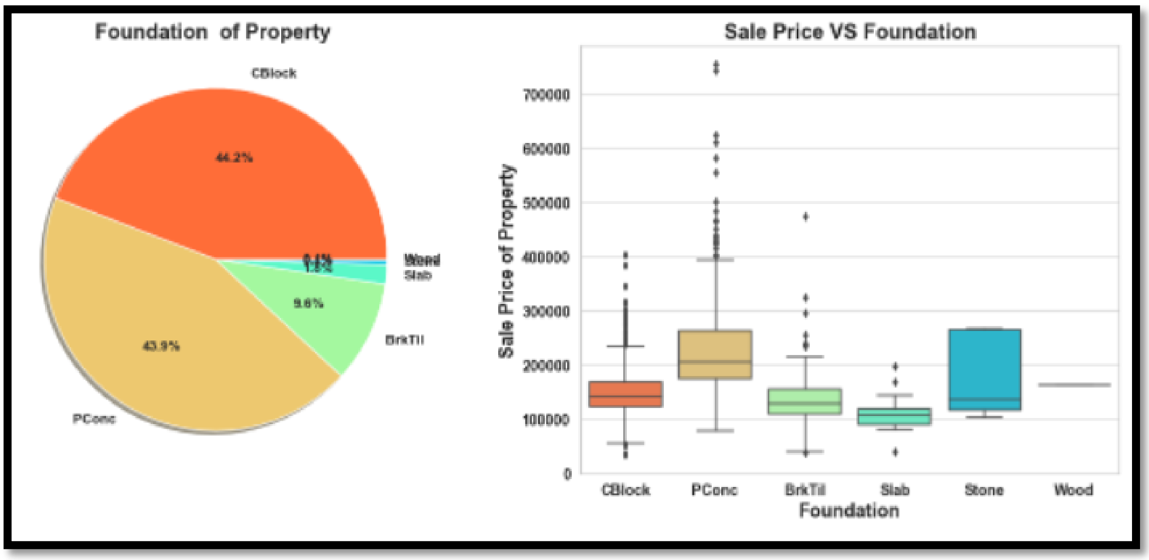
Observation :-

- More than 75% House properties come with Gable Roof Style followed by around 15 % house properties with Hip Style.
- From Box plot we can see that Hip style Roof are much costlier than remaining roof style.



Observation :-

- More than 90% Properties in Data set made with roof material of Standard (Composite) Shingle.
- Wood Shingles is Costlier Material compare to rest.



Foundation Description:-
Foundation: Type of foundation

- BrkTil : Brick & Tile
- CBlock : Cinder Block
- PConc : Poured Contrete
- Slab : Slab
- Stone : Stone
- Wood : Wood

Observation :-

- 44.2% Properties with CBlock Foundation & 43.9% housing property come with PConc Foundation.
- Pconc Foundation are mostly use in costily housing properties.

Chap. 5 Conclusion

1. Key Findings and Conclusions of the Study

Algorithm	R2score	CV score
Linear Regression	**90.97**	**86.40**
Random Forest Regressor	**89.19**	**85.06**
XGB Regressor	**89.35**	**85.34**
Decision Tree Regressor	**66.99**	**70.80**
Gradient Boosting Regressor	**91.57**	**86.78**
Extra Tree Regressor	**85.40**	**84.49**
Gradient Boosting Regressor Hyper Parameter Tuned Final Model	**91.30**	**86.57**

- Gradient Boosting Regressor giving us maximum R2 Score, so Gradient Boosting Regressor is selected as best model.
- After hyper parameter tuning Final Model is giving us R2 Score of 91.57% which is slightly improved compare to earlier R2 score of 91.30%

2. Limitations of this work and Scope for Future Work

- ANN can be used create more accurate model.
- Some additional feature can be added to data which enable us to perform Time series analysis.