FLIP ROBO

HOUSING PRICE PREDICTION PROJECT

Use Case Report

Submitted by:

Sourabh Jhod

ACKNOWLEDGMENT

I am highly indebted to my Subject Matter Expert **MR. Shubham Yadav** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my Gratitude to FlipRobo Technologies also who gave me the opportunity to do this project on Housing Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things also, 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:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://www.geeksforgeeks.org/
4) https://towardsdatascience.com/
5) https://www.analyticsvidhya.com/
6) https://www.kdnuggets.com/
7) https://www.kaggle.com/
8) https://medium.com/

INTRODUCTION

- **Business Problem Framing**

Prices of real estate properties are sophisticatedly linked with our economy. Despite this, we do not have accurate measures of housing prices based on the vast amount of data available. Therefore, the goal of this project is to use machine learning to predict the selling prices of houses based on many economic factors.

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.

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

- **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 a variable?
- 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 in order 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**

Our main objective of doing this project is to build a model to predict the house prices with the help of other supporting features. We are going to predict by using Machine Learning algorithms.

The sample data is provided to us from our client database. In order to improve the selection of customers, the client wants some predictions that could help them in further investment and improvement in selection of customers.

House Price Index is commonly used to estimate the changes in housing price. Since housing price is strongly correlated to other factors such as location, area, population, it requires other information apart from HPI to predict individual housing price.

There has been a considerably large number of papers adopting traditional machine learning approaches to predict housing prices accurately, but they rarely concern themselves with the performance of individual models and neglect the less popular yet complex models.

As a result, to explore various impacts of features on prediction methods, this paper will apply both traditional and advanced machine learning approaches to investigate the difference among several advanced models. This paper will also comprehensively validate multiple techniques in model implementation on regression and provide an optimistic result for housing price prediction.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling 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 (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or '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 (predictor). 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.

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.

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.

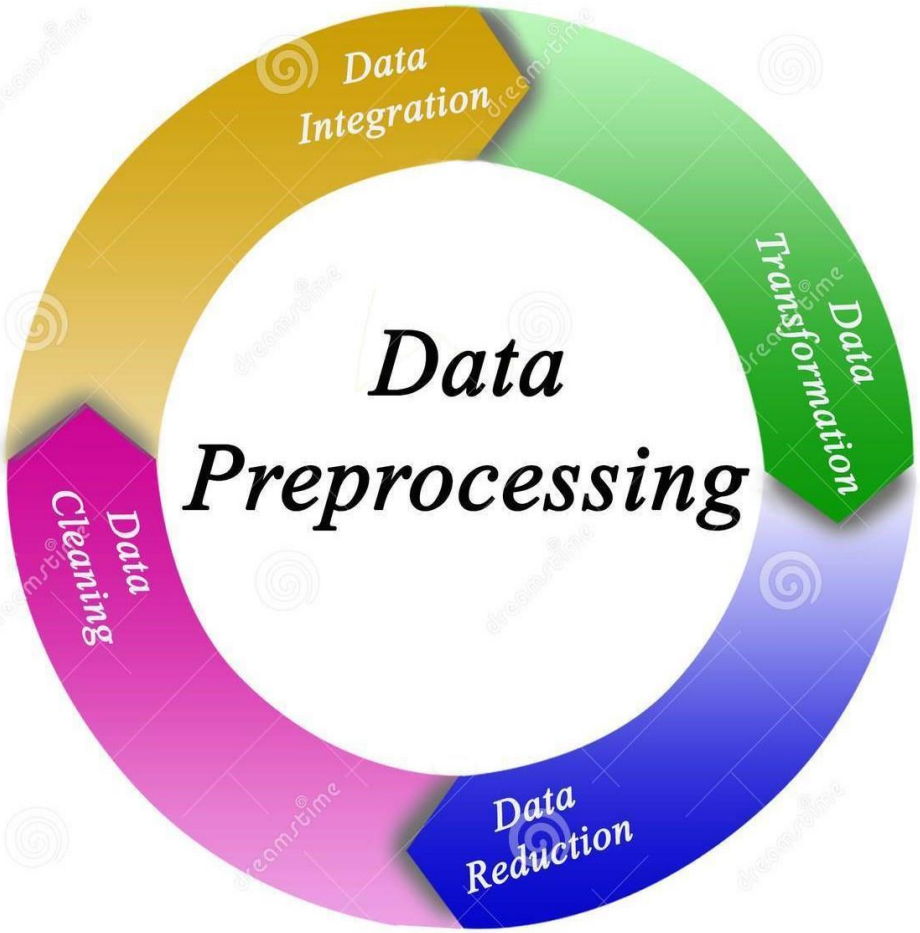
- ## Data Preprocessing Done

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms. Preprocessing of data is mainly to check the data quality. The quality can be checked by the following

- **Accuracy**: To check whether the data entered is correct or not.
- **Completeness**: To check whether the data is available or not recorded.
- **Consistency:** To check whether the same data is kept in all the places that do or do not match.
- **Timeliness**: The data should be updated correctly.
- **Believability**: The data should be trustable.
- **Interpretability**: The understandability of the data.

Major Tasks in Data Preprocessing:

1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation



I have used some following pre-processing steps:

- Loading the training dataset as a dataframe.
- Used pandas to set display I ensuring we do not see any truncated information.

- Checked the number of rows and columns present in our training dataset.
- Checked for Duplicated values present in the data.
- Checked for missing data and the number of rows with null values.
- Verified the percentage of missing data in each column and decided to discard the one's that have more than 50% of null values.
- Dropped all the unwanted columns and duplicate data present in our dataframe.
- Separated categorical column names and numeric column names in separate list variables for ease in visualization.
- Performed imputation to fill missing data using mode on both numeric data and for categorical data columns
- Replaced Some missing value with a "Missing" category .
- Used the visualization libraries (matplotlib , seaborn) to visualise the data.
- checked for outliers and skewness information.
- With the help of heatmap, correlation bar graph was able to understand the Feature vs Label relativity and insights on multicollinearity amongst the feature columns.
- With the help of Label encoding technique converted all object datatype columns to numeric datatype
- Separated feature and label data to ensure feature scaling is performed avoiding any kind of biasness.
- Performed the Standard scaling to make data into same scale.
- Performed Feature selection using Lasso Regularization to select the important features from all the features.
- Finally created a regression model function along with evaluation metrics to pass through various model formats

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. In Data Science community there is a saying “Garbage In Garbage Out” therefore we had to be very cautious and spent almost 80% of our project building time in understanding each and every aspect of the data how they were related to each other as well as our target label.

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. I did not want to impute data and then cause a biasness in the machine learning model from values that did not come from real people.

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:

- RAM: 4 GB
- CPU : Intel^{R} Core ™ i3-6006U CPU @ 2.00 GHZ
- Gpu: Intel^{R} HD Graphics 520

Software Used:

- Programming language: Python
- Distribution: Anaconda Navigator
- Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

- Pandas
- NumPy
- Matplotlib
- Seaborn
- Scikit-learn

Model/s Development and Evaluation

Identification of possible problem-solving approaches

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data and EDA to check the correlation of independent and dependent features. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models.

For this project we need to predict the sale price of houses, means our target column is continuous so this is a regression problem. I have used various regression algorithms and tested for the prediction. By doing various evaluations I have selected Gradient Boost Regressor as best suitable algorithm for our final model as it is giving good r2-score and least difference in r2-score and CV-score among all the algorithms used.Other regression algorithms are also giving me good accuracy but some are over-fitting and some are with under-fitting the results which may be because of less amount of data.

In order to get good performance as well as accuracy and to check my model from over-fitting and under-fitting I have made use of the K-Fold cross validation and then hyper parameter tuned the final model.
Once I was able to get my desired final model I ensured to save that model before I loaded the testing data and started performing the data pre-processing as the training dataset and obtaining the predicted sale price values out of the Regression Machine Learning Model.

Testing of Identified Approaches

- Linear Regression Model
- Lasso Regularization Model
- Ridge Regularization Model
- ElasticNet Regularization Model
- K Nearest Neighbours Regression Model
- Support Vector Regression Model

- Decision Tree Regression Model
- Random Forest Regression Model
- Extra Trees Regression Model
- Ada Boost Regression Model
- Gradient Boosting Regression Model
- Bagging Regressor Model
- XGBRegressor Model
- VotingRegressor Model

Run and Evaluate selected models

I used a total of 14 Regression Models The code for the models is listed below.

LinearRegression

```
model = LinearRegression()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round( cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round(cross_val_score(model,X,y,cv=
print('MAE               ',round(mean_absolute_error(y_test,y_pred),2))
```

```
Training R2 score  88.1
Testing R2 score   90.7
Cross_val_score    87.4
Difference         3.299999999999997
MAE                0.09
```

Lasso

```
model = Lasso(alpha=0.006)
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),2))
```

```
Training R2 score  86.9
Testing R2 score   90.9
Cross_val_score    88.1
Difference         2.8000000000000114
MAE                0.09
```

Ridge

```
model = Ridge()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),2))
```

```
Training R2 score  88.1
Testing R2 score   90.7
Cross_val_score    87.4
Difference         3.299999999999997
MAE                0.09
```

ElasticNet

```
model = ElasticNet(alpha=0.006)
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),2))
```

```
Training R2 score  87.6
Testing R2 score   90.9
Cross_val_score    88.0
Difference         2.9000000000000057
MAE                0.09
```

KNeighborsRegressor

```
model = KNeighborsRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),2))
```

```
Training R2 score  83.6
Testing R2 score   77.2
Cross_val_score    75.4
Difference         1.7999999999999972
MAE                0.13
```

SVR

```
model = SVR()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  95.5
Testing R2 score   77.9
Cross_val_score    80.4
Difference         -2.5
MAE                0.112
```

DecisionTreeRegressor

```
model = DecisionTreeRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),2))
```

```
Training R2 score  100.0
Testing R2 score   76.2
Cross_val_score    72.2
Difference         4.0
MAE                0.14
```

RandomForestRegressor

```
model = RandomForestRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  97.6
Testing R2 score   87.9
Cross_val_score    86.1
Difference         1.8000000000000114
MAE                0.099
```

ExtraTreesRegressor

```
model = ExtraTreesRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  100.0
Testing R2 score   87.8
Cross_val_score    86.8
Difference         1.0999999999999943
MAE                0.096
```

AdaBoostRegressor

```
model = AdaBoostRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  82.1
Testing R2 score   81.6
Cross_val_score    79.80000000000001
Difference         1.3999999999999915
MAE                0.133
```

GradientBoostingRegressor

```
model = GradientBoostingRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  95.3
Testing R2 score   87.8
Cross_val_score    86.8
Difference         0.8999999999999915
MAE                0.094
```

BaggingRegressor

```
model = BaggingRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  96.5
Testing R2 score   86.2
Cross_val_score    84.39999999999999
Difference         2.1000000000000085
MAE                0.105
```

XGBRegressor

```
model = XGBRegressor()
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  100.0
Testing R2 score   80.0
Cross_val_score    86.4
Difference         -6.400000000000006
MAE                0.116
```

VotingRegressor

```
rf=RandomForestRegressor()
gb=GradientBoostingRegressor()
etc=ExtraTreesRegressor()

est=[('etc',etc),
     ("gb",gb),
     ("rf",rf)]

model = VotingRegressor(estimators=est)
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

```
Training R2 score  98.6
Testing R2 score   88.8
Cross_val_score    87.7
Difference         1.0999999999999943
MAE                0.091
```

Key Metrics for success in solving problem under consideration

The key metrics used here were r2_score, cross_val_score, & MAE. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV Or randomized search cv method.

Cross Val Score : 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 Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.
In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

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.

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.

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. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using 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.

Randomized Search CV : Random search is a method in which random combinations of hyperparameters are selected and used to train a model. The best random hyperparameter combinations are used.
Random search bears some similarity to grid search.
However, a key distinction is that we do not specify a set of possible values for every hyperparameter. Instead, we sample values from a statistical distribution for each hyperparameter. A sampling distribution is defined for every hyperparameter to do a random search.
This technique allows us to control the number of attempted hyperparameter combinations. Unlike grid search, where every possible combination is attempted, random search allows us to specify the number of models to train. We can base our search iterations on our computational resources or the time taken per iteration. The image below shows a random layout.

Hyperparameter tuning

```
params={
    "learning_rate":np.arange(0.01,2.01,0.01),
    "n_estimators":np.arange(100,500,50),
    "max_depth":np.arange(3,20,1),
    }
```

```
randomscv = RandomizedSearchCV(model,params,scoring='r2',random_state=0,n_jobs=-1)
```

```
randomscv.fit(X_train,y_train)
```

```
randomscv.best_params_
```

```
{'n_estimators': 300, 'max_depth': 4, 'learning_rate': 0.21000000000000002}
```

I used Randomized SearchCV method for hyper parameter tuning my best model.

```
X_train,X_test,y_train,y_test = train_test_split(X[selected_feature],y,test_size=0.2,random_state=51)
model = GradientBoostingRegressor(n_estimators=300,max_depth=4,learning_rate=0.21)
model.fit(X_train,y_train)
y_train_pred = model.predict(X_train)
y_pred = model.predict(X_test)
print('Training R2 score ',round(r2_score(y_train_pred,y_train),3)*100)
print('Testing R2 score  ',round(r2_score(y_test,y_pred),3)*100)
print("Cross_val_score   ",round(cross_val_score(model,X,y,cv=10,scoring='r2').mean(),3)*100)
print("Difference        ",((round(r2_score(y_test,y_pred),3)*100)-(round( cross_val_score(model,X,y,cv
print('MAE               ',round(mean_absolute_error(y_test,y_pred),3))
```

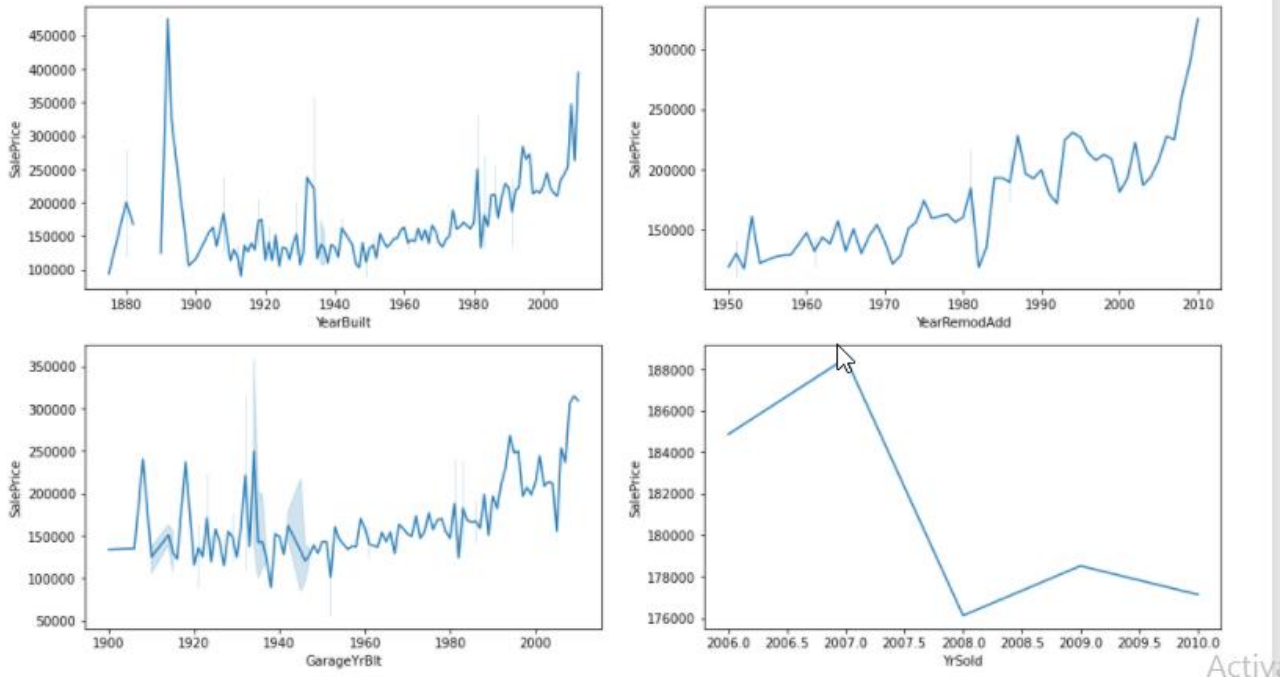
```
Training R2 score  100.0
Testing R2 score   90.7
Cross_val_score    87.8
Difference         3.0
MAE                0.085
```

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.

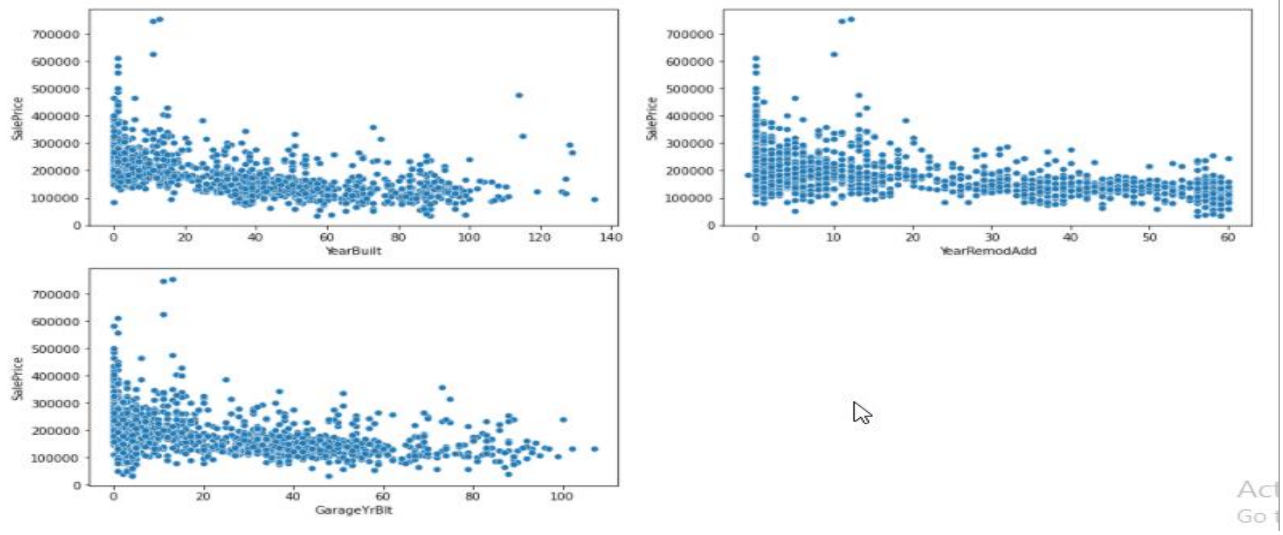
Visualizations:

I created Bar plots, count plots and scatter plots to get further visual insights on our training dataset feature values.

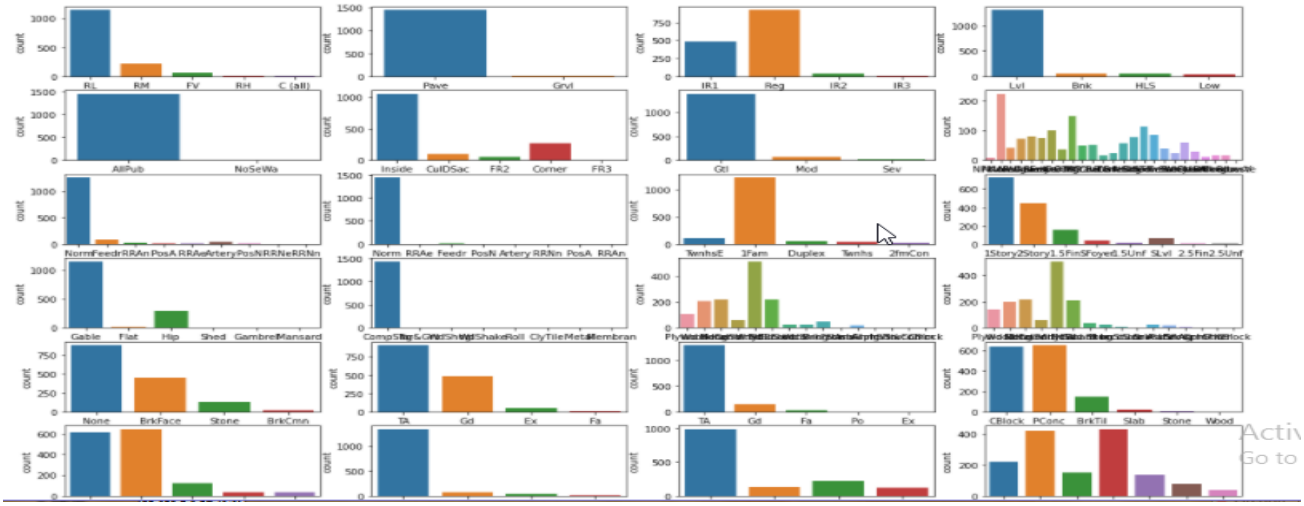
```
plt.figure(figsize=(16,9))
for i,j in enumerate(year_colm):
    plt.subplot(2,2,i+1)
    sns.lineplot(df[j],df["SalePrice"])
plt.show()
```



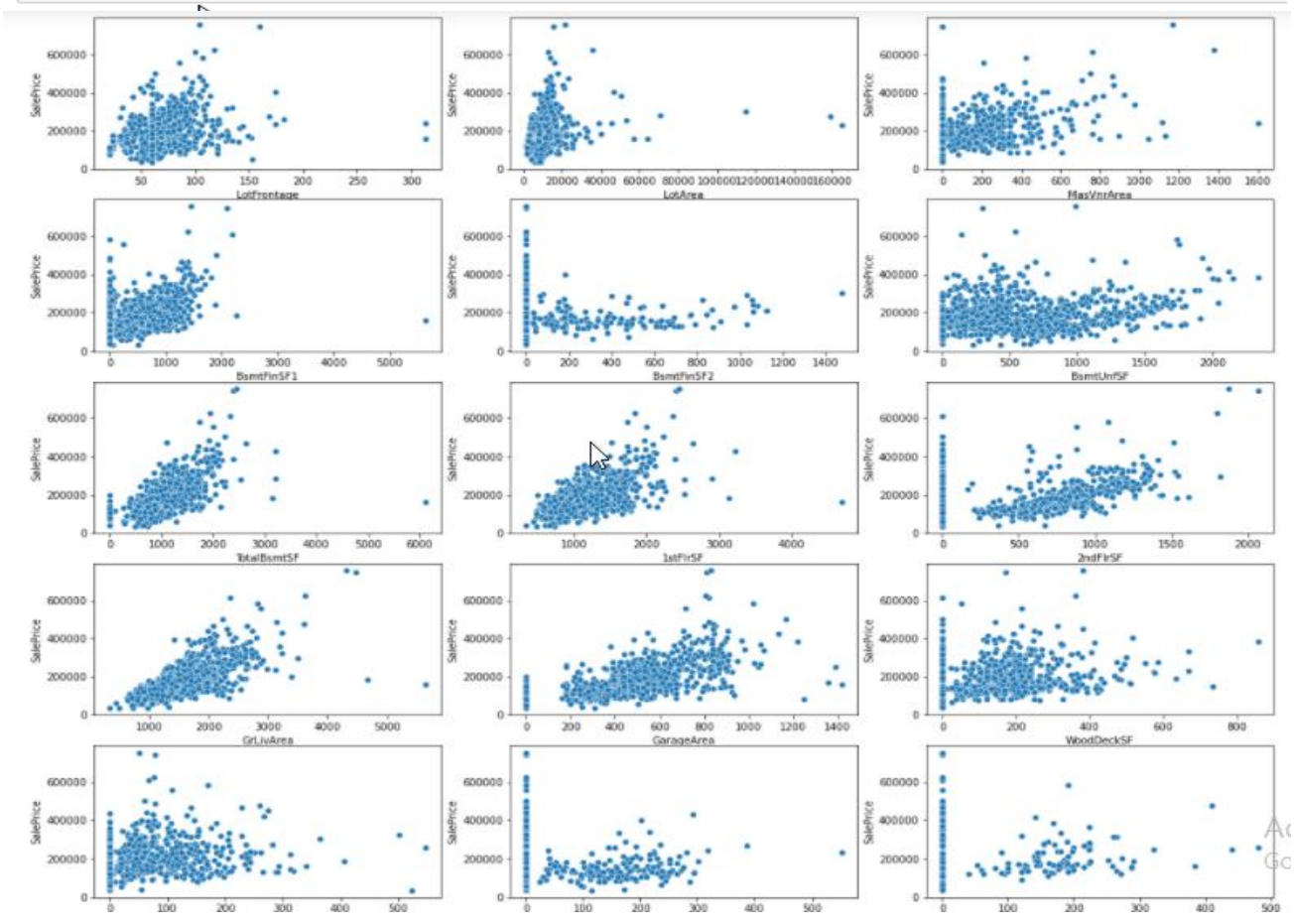

```
plt.figure(figsize=(16,9))
for i,j in enumerate(year_colm[0:-1]):
    plt.subplot(2,2,i+1)
    sns.scatterplot(df['YrSold'] - df[j] , df['SalePrice'])
    plt.xlabel(j)
plt.show()
```



```
plt.figure(figsize=(20,20))
for i,j in enumerate(categorical):
    plt.subplot(10,4,i+1)
    sns.countplot(df[j])
    plt.xlabel(j)
plt.show()
```

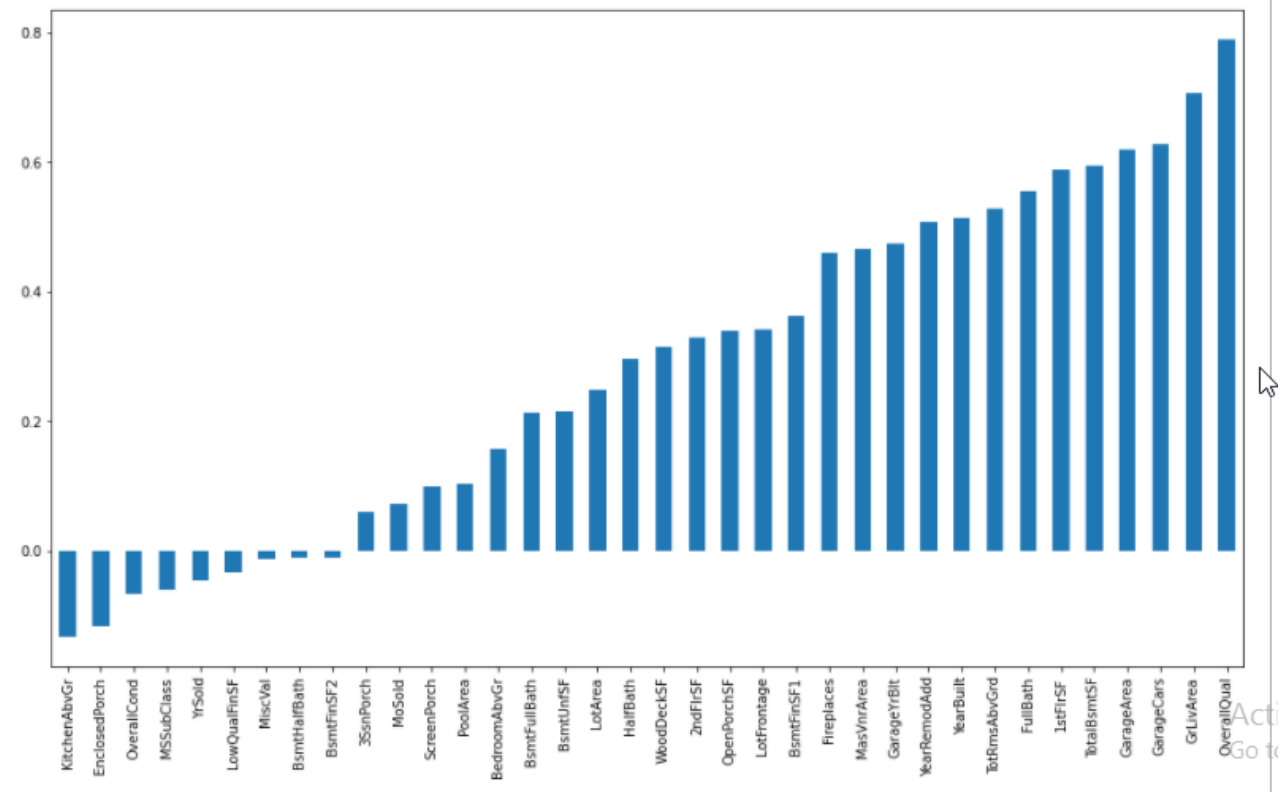


```
plt.figure(figsize=(20,20))
for i,j in enumerate(continuous_values):
    plt.subplot(6,3,i+1)
    sns.scatterplot(df[j],df["SalePrice"])
    plt.xlabel(j)
plt.show()
```



```
num_var.corr()["SalePrice"].drop(["SalePrice"]).sort_values().plot(kind="bar",figsize=(16,9))
```

```
<AxesSubplot:>
```



Interpretation of the Results

Visualizations: It helped me to understand the correlation between independent and dependent features. Also, helped me with feature importanceand 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 asscatter plot helped me to select the best model.

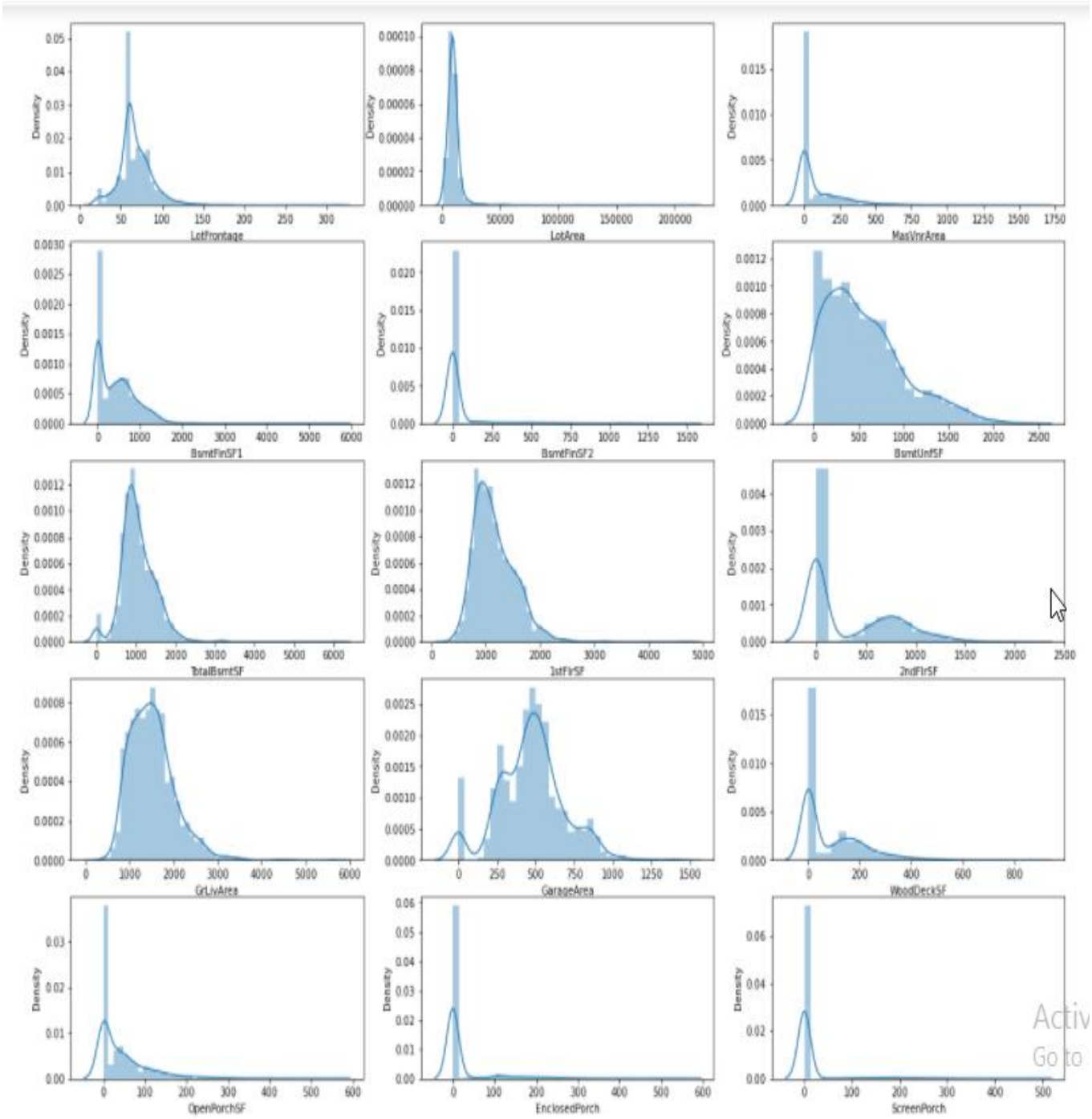
Pre-processing: Basically, before building the model the dataset should be cleaned and scaled by performing few steps. As I mentioned above in the pre-processing steps where all the important features are present in the dataset and ready for model building.

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.

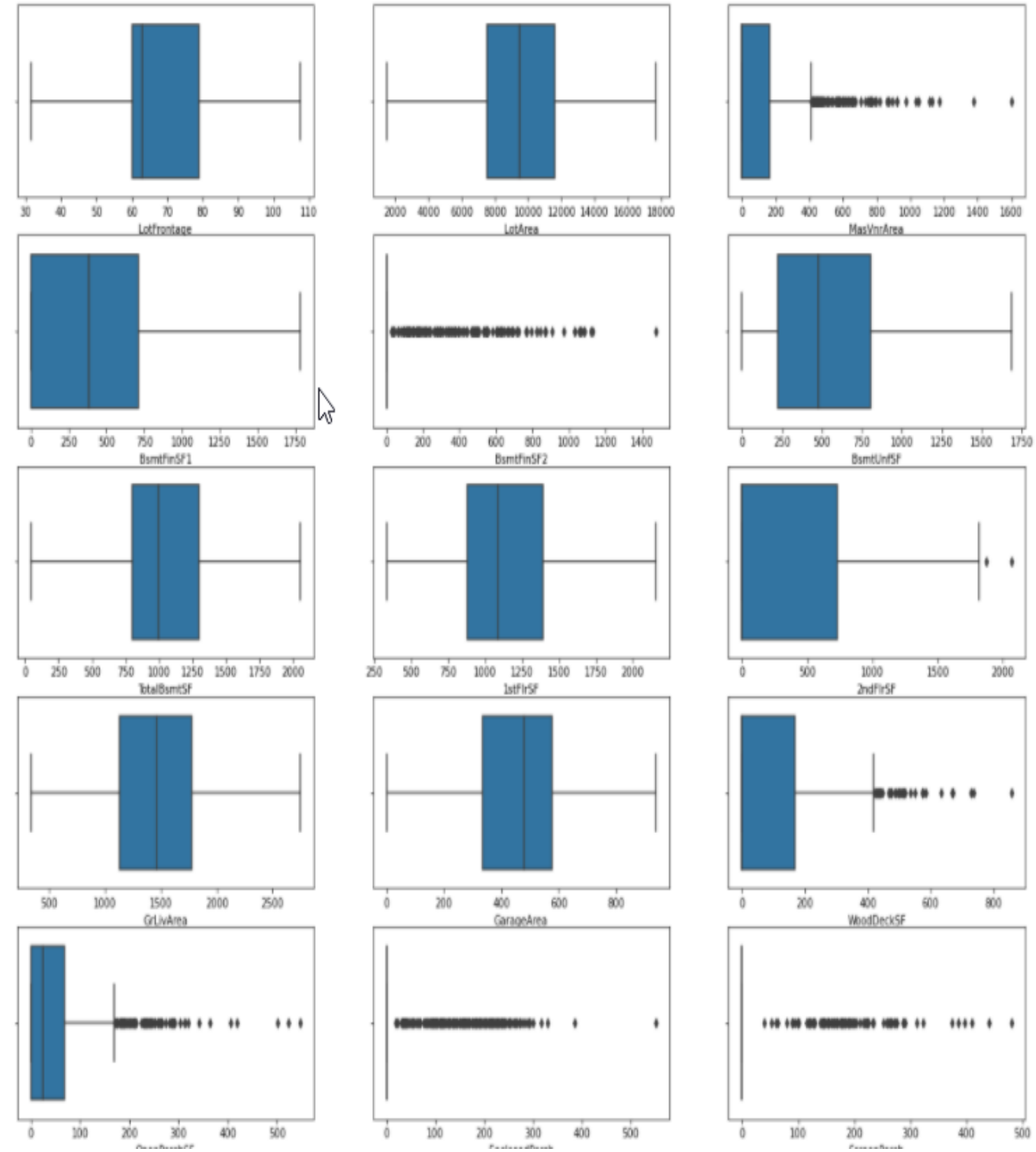
CONCLUSION

Key Findings and Conclusions of the Study

I observed all the encoded dataset information by plotting various graphs and visualized further insights.



Box Plot


LotFrontage
LotArea
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
GrLivArea
GarageArea
WoodDeckSF

Learning Outcomes of the Study in respect of Data Science

The above study helps one to understand the business of real estate. How the price is changing across the properties. With the Study we can tell how multiple real estate amenities like LotArea,LotShape,OverallQual etc decides the cost.With the help of the above analysis, one can sketch the needs of a property buyer and according to need we can project the price of the property.

Limitations of this work and Scope for Future Work

During this project I have faced a problem of low amount 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.