FLIP ROBO

# Housing Price Prediction Project Use CaseReport

Submitted by:

Anurag Krishnan

# ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Khushboo Garg as well as Flip Robo Technologies 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/

# INTRODUCTION

- Business Problem Framing

  Houses are one of the necessary needs of each and every person around the globe and therefore the housing and real estate market 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.

  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:

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 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 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.

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

  Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre- process our data before feeding it into our model. Therefore, it is the first and crucial step while creating a machine learning model. I have used some following pre-processing steps:

  a. Loading the training dataset as a dataframe
  b. Used pandas to set display I ensuring we do not see any truncated information
  c. Checked the number of rows and columns present in our training dataset
  d. Checked for missing data and the number of rows with null values
  e. Verified the percentage of missing data in each column and decided to discard the ones that have more than 50% of null values
  f. Dropped all the unwanted columns and duplicate data present in our dataframe

g. Separated categorical column names and numeric column names in separate list variables for ease in visualization
h. Checked the unique values information in each column to get a gist for categorical data
i. Performed imputation to fill missing data using mean on numeric data and mode for categorical data columns
j. Used Pandas Profiling during the visualization phase along with pie plot, count plot, scatter plot and the others
k. With the help of ordinal encoding technique converted all object data type columns to numeric datatype
l. Thoroughly checked for outliers and skewness information
m. 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
n. Separated feature and label data to ensure feature scaling is performed avoiding any kind of biasness
o. Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details
p. 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 the 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 the 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 and how they were related to each other as well as our target label.

With the objective of predicting housing 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 biases 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 inclined to them was.

- Hardware and Software Requirements and Tools Used

  Hardware Used:

  i. RAM: 8 GB
  ii. CPU: AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
  iii. GPU: AMD Radeon ™ Vega 8 Graphics and NVIDIA GeForce GTX 1650 Ti

  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.

# Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  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 was cleaned and scaled before it was fed into the machine learning models.

  For this project we need to predict the sale price of houses, meaning 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 Random Forest Regressor as the best suitable algorithm for our final model as it is giving a 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 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 overfitting 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 (Algorithms)

The algorithms used on training and test data are as follows:

A. Linear Regression Model
B. Support Vector Regression Model
C. Decision Tree Regression Model
D. Random Forest Regression Model
E. K Nearest Neighbours Regression Model
F. Gradient Boosting Regression Model
G. AdaBoost Regression Model

- Run and Evaluate selected models

I used a total of 7Regression Models after choosing the random state amongst 1-100 numbers. Then I even defined a function for getting the regression model trained and evaluated. The code for the models is listed below.

Random State:

```
maxAccuracy=0
maxRandomState=0

for i in range(1, 100):
    x_train, x_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0.30, random_state=i)
    lr=LinearRegression()
    lr.fit(x_train, y_train)
    y_lr = lr.predict(x_test)
    lr_r2 = r2_score(y_test, y_lr)

    if lr_r2>maxAccuracy:
        maxAccuracy=lr_r2
        maxRandomState=i

print("Best r2 score is", maxAccuracy ," on Random State", maxRandomState)
```

```
Best r2 score is 0.892936965315257  on Random State 55
```

Linear Regression:

```
x_train, x_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0.30, random_state=55)

#training our model
lr=LinearRegression()
lr.fit(x_train, y_train)

#Predicting y_test
y_lr = lr.predict(x_test)
lr_r2 = r2_score(y_test, y_lr)

#R2 Score
print("R2 score : ", lr_r2*100)

#Cross Validation Score
cross_val_lr = cross_val_score(lr,X_scaled,Y,cv=5)
print(cross_val_lr)
print("Cross Validation Score : ",cross_val_lr.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_lr))
print("Mean Square Error : ",mean_squared_error(y_test,y_lr))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_lr)))
```

```
R2 score :  89.2936965315257
[0.85748973 0.90165847 0.84863998 0.86601646 0.80875789]
Cross Validation Score :  85.6512504721638
Mean Absolute Error : 17075.19933841226
Mean Square Error :  508951851.2489534
Root Mean Square Error : 22559.96124218642
```

## Support Vector Regressor:

```
x_train, x_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0.30, random_state=19)

#training our model
svr=SVR()
svr.fit(x_train, y_train)

#Predicting y_test
y_svr = svr.predict(x_test)
svr_r2 = r2_score(y_test, y_svr)

#R2 Score
print("R2 score : ", svr_r2*100)

#Cross Validation Score
cross_val_svr = cross_val_score(svr,X_scaled,Y,cv=5)
print(cross_val_svr)
print("Cross Validation Score : ",cross_val_svr.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_svr))
print("Mean Square Error : ",mean_squared_error(y_test,y_svr))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_svr)))
```

```
R2 score :  0.10744051281690314
[-0.000283   -0.12197869 -0.01545254 -0.10236721 -0.01877176]
Cross Validation Score :  -5.177063816535852
Mean Absolute Error : 46737.70760985306
Mean Square Error :  3416941074.4322276
Root Mean Square Error : 58454.60695644294
```

## Decision Tree Regressor:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=67)

#training our model
dt=DecisionTreeRegressor()
dt.fit(x_train, y_train)

#Predicting y_test
y_dt = dt.predict(x_test)
dt_r2 = r2_score(y_test, y_dt)

#R2 Score
print("R2 score : ", dt_r2*100)

#Cross Validation Score
cross_val_dt = cross_val_score(dt,X_scaled,Y,cv=5)
print(cross_val_dt)
print("Cross Validation Score : ",cross_val_dt.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_dt))
print("Mean Square Error : ",mean_squared_error(y_test,y_dt))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_dt)))
```

```
R2 score :  77.75040809052895
[0.62942145 0.75687501 0.68868365 0.57167325 0.63775259]
Cross Validation Score :  65.68811916168143
Mean Absolute Error : 21661.551867219918
Mean Square Error :  879780564.7385892
Root Mean Square Error : 29661.095137209435
```

## Random Forest Regressor:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=77)

#training our model
rf=RandomForestRegressor()
rf.fit(x_train, y_train)

#Predicting y_test
y_rf = rf.predict(x_test)
rf_r2 = r2_score(y_test, y_rf)

#R2 Score
print("R2 score : ", rf_r2*100)

#Cross Validation Score
cross_val_rf = cross_val_score(rf,X,Y,cv=5)
print(cross_val_rf)
print("Cross Validation Score : ",cross_val_rf.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_rf))
print("Mean Square Error : ",mean_squared_error(y_test,y_rf))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_rf)))
```

```
R2 score :  88.60869413047931
[0.84801785 0.88272338 0.84806705 0.83238085 0.80816588]
Cross Validation Score :  84.38710017168886
Mean Absolute Error : 16209.932780082987
Mean Square Error :  481051971.228459
Root Mean Square Error : 21932.897009480053
```

## K Nearest Neighbours Regressor:

```
x_train, x_test, y_train, y_test = train_test_split(X_scaled, Y, test_size=0.30, random_state=81)

#training our model
knn=KNeighborsRegressor()
knn.fit(x_train, y_train)

#Predicting y_test
y_knn = knn.predict(x_test)
knn_r2 = r2_score(y_test, y_knn)

#R2 Score
print("R2 score : ", knn_r2*100)

#Cross Validation Score
cross_val_knn = cross_val_score(knn,X_scaled,Y,cv=5)
print(cross_val_knn)
print("Cross Validation Score : ",cross_val_knn.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_knn))
print("Mean Square Error : ",mean_squared_error(y_test,y_knn))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_knn)))
```

```
R2 score :  87.21716877237353
[0.85149839 0.8800854  0.78956883 0.81631759 0.76721691]
Cross Validation Score :  82.09374257460694
Mean Absolute Error : 16694.312033195023
Mean Square Error :  481886299.55352694
Root Mean Square Error : 21951.908790661622
```

## Gradient Boosting Regressor:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=77)

#training our model
gb=GradientBoostingRegressor()
gb.fit(x_train, y_train)

#Predicting y_test
y_gb = gb.predict(x_test)
gb_r2 = r2_score(y_test, y_gb)

#R2 Score
print("R2 score : ", gb_r2)

#Cross Validation Score
cross_score_gb = cross_val_score(gb,X,Y,cv=5)
print(cross_score_gb)
print("Cross Validation Score : ",cross_score_gb.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_gb))
print("Mean Square Error : ",mean_squared_error(y_test,y_gb))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_gb)))
```

```
R2 score :  0.8954359989010091
[0.86041538 0.89485928 0.81771176 0.85471534 0.82430913]
Cross Validation Score :  85.04021758815541
Mean Absolute Error : 16020.299263448127
Mean Square Error :  441571137.00890183
Root Mean Square Error : 21013.594100222403
```

## AdaBoostRegressor:

```python
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=99)

#training our model
ad=AdaBoostRegressor()
ad.fit(x_train, y_train)

#Predicting y_test
y_ad = ad.predict(x_test)
ad_r2 = r2_score(y_test, y_ad)

#R2 Score
print("R2 score : ", ad_r2*100)

#Cross Validation Score
cross_score_ad = cross_val_score(ad,X,Y,cv=5)
print(cross_score_ad)
print("Cross Validation Score : ",cross_score_ad.mean()*100)

#Error Estimation Methods
print("Mean Absolute Error :",mean_absolute_error(y_test,y_ad))
print("Mean Square Error : ",mean_squared_error(y_test,y_ad))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,y_ad)))
```

```
R2 score :  83.77191022035797
[0.81733686 0.85893795 0.81287372 0.81045356 0.78564047]
Cross Validation Score :  81.7048512560287
Mean Absolute Error : 19255.67200090899
Mean Square Error :  714623774.1994919
Root Mean Square Error : 26732.4479649637
```

- Key Metrics for success in solving problem under consideration

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

  1. Cross Validation:

  Cross-validation helps to find out the overfitting and underfitting 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 the 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 the 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.
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. This selection procedure for hyperparameters is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.
GridSearchCV is a function that comes in the Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have the 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.

## Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
```

```
#Choosing Random Forest Regressor
params = {
    'criterion' : ['absolute_error','poisson'],
    'max_depth' : [2,3,4],
    'min_samples_split' : [2,3,4],
    'max_features' : ['sqrt','log2']
}

```

```
grid_random = GridSearchCV(RandomForestRegressor(),param_grid=params,cv=5)
grid_random
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'criterion': ['absolute_error', 'poisson'],
                         'max_depth': [2, 3, 4],
                         'max_features': ['sqrt', 'log2'],
                         'min_samples_split': [2, 3, 4]})
```

```
random_hyperparameter = RandomForestRegressor(criterion='poisson',max_depth=4,max_features='sqrt',min_samples_split=2)
print(random_hyperparameter)
print("------------------------------------------------------------------------")
random_hyperparameter.fit(x_train,y_train)
random_hyperparameter_pred = random_hyperparameter.predict(x_test)
print(random_hyperparameter_pred)
print("------------------------------------------------------------------------")
print("R2 Score : ",r2_score(y_test,random_hyperparameter_pred)*100)
print("Cross Validation Score : ",cross_val_score(RandomForestRegressor(),X,Y,cv=5).mean()*100)
print("Mean Absolute Error :",mean_absolute_error(y_test,random_hyperparameter_pred))
print("Mean Square Error : ",mean_squared_error(y_test,random_hyperparameter_pred))
print("Root Mean Square Error :",np.sqrt(mean_squared_error(y_test,random_hyperparameter_pred)))
print("------------------------------------------------------------------------")
```

```
------------------------------------------------------------------------
R2 Score :  83.95774988400765
Cross Validation Score :  84.69598640981856
Mean Absolute Error : 19284.094544363357
Mean Square Error :  677460172.6643658
Root Mean Square Error : 26028.065096437072
------------------------------------------------------------------------
```

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 used pandas profiling to get the over viewed visualization on the pre-processed data. pandas-profiling is an open-source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates interactive reports in web format that can be presented to any person, even if they don't know programming. It also offers report generation for the dataset with lots of features and customizations for the report generated.

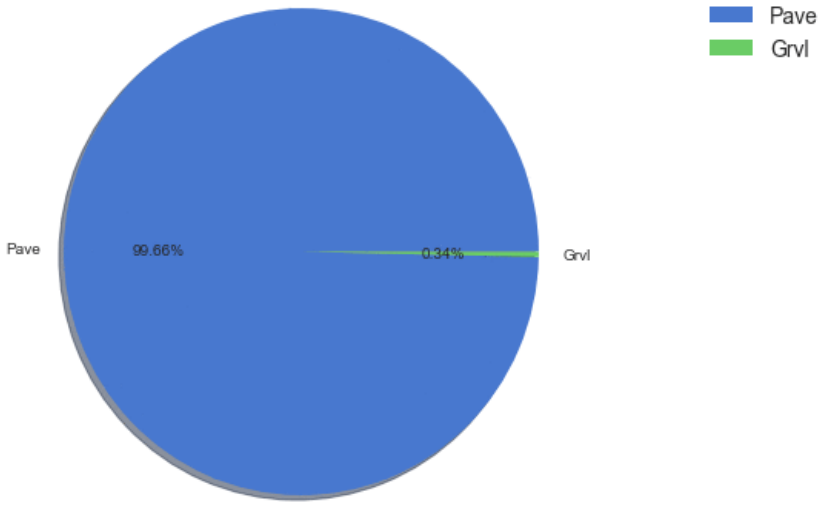
  I then created pie plots, count plots and scatter plots to get further visual insights on our training dataset feature values.

  Code:

```
plt.style.use('seaborn-muted')
def generate_pie(x):
    plt.style.use('seaborn-white')
    plt.figure(figsize=(10,5))
    plt.pie(x.value_counts(), labels=x.value_counts().index, shadow=True, autopct='%1.2f%%')
    plt.legend(prop={'size':14})
    plt.axis('equal')
    plt.tight_layout()
    return plt.show()

for i in train_df[single]:
    print(f"Single digit category column name:", i)
    generate_pie(train_df[i])
```

Output:



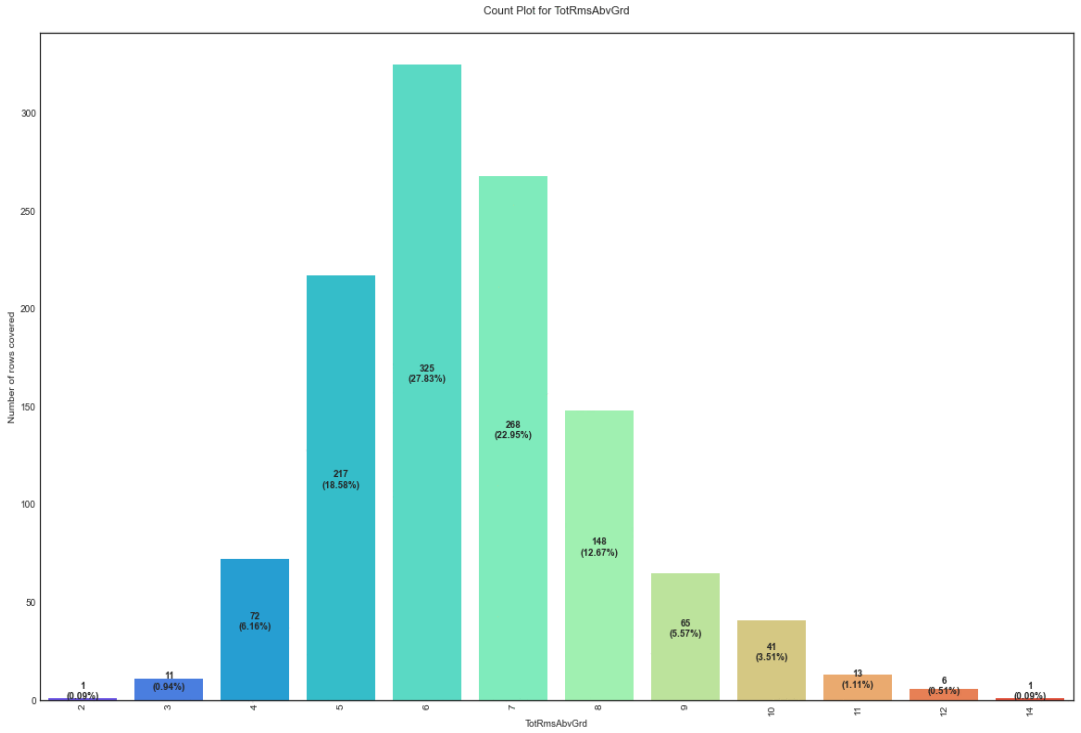

Code:

```
for col in train_df[double]:
    plt.figure(figsize=(20,12))
    col_name = col
    values = train_df[col_name].value_counts()
    index = 0
    ax = sns.countplot(train_df[col_name], palette="rainbow")

    for i in ax.patches:
        h = i.get_height() # getting the count of each value
        t = len(train_df[col_name]) # getting the total number of records using length
        s = f"{h}\n({round(h*100/t,2)}%)" # making the string for displaying in count bar
        plt.text(index, h/2, s, ha="center", fontweight="bold")
        index += 1

    plt.title(f"Count Plot for {col_name}\n")
    plt.xlabel(col_name)
    plt.ylabel(f"Number of rows covered")
    plt.xticks(rotation=90)
    plt.show()
```

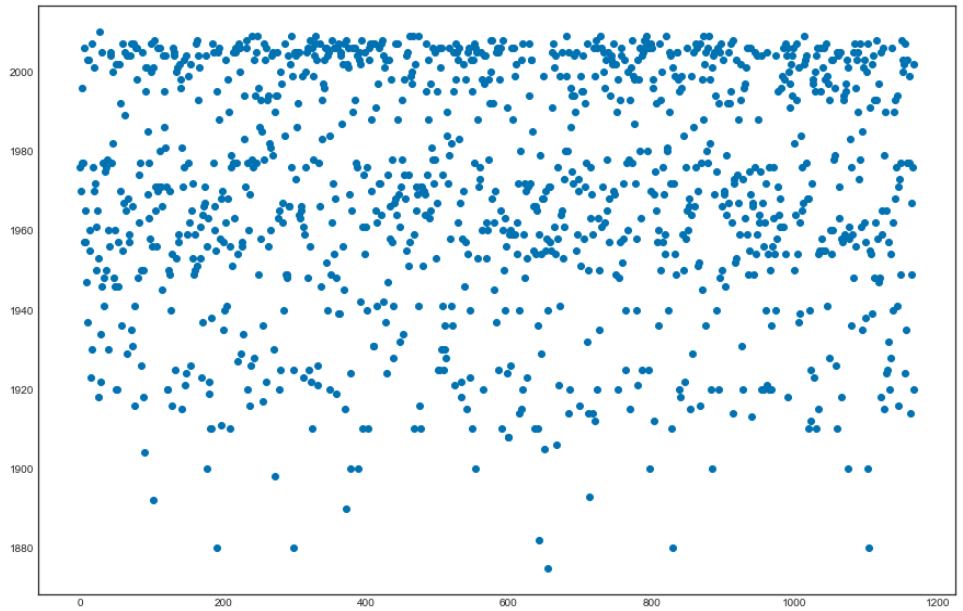
Output:




Code:

```
plt.style.use('seaborn-colorblind')
for j in train_df[triple]:
    plt.figure(figsize=(15,10))
    print(f"Scatter plot for {j} column with respect to the rows covered ->")
    plt.scatter(train_df.index, train_df[j])
    plt.show()
```

Output:



- 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 multicollinearity 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 a 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.

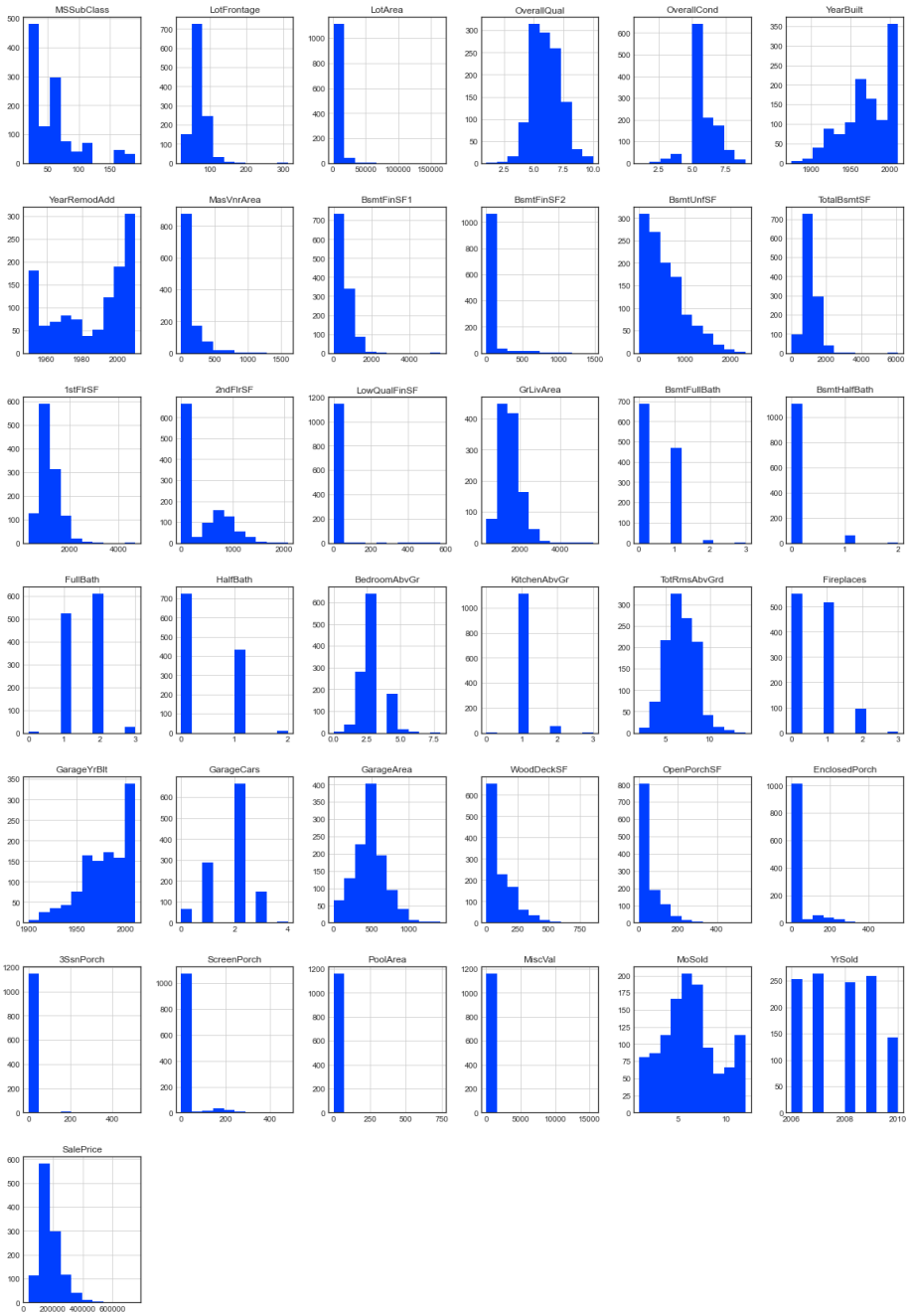
  Model Creation: Now, after performing the train test split, I have x_train, x_test, y_train & y_test, which are required to build Machine learning models. I have built multiple regression models to get the best R2 score, MSE, RMSE & MAE out of all the models.

# CONCLUSION

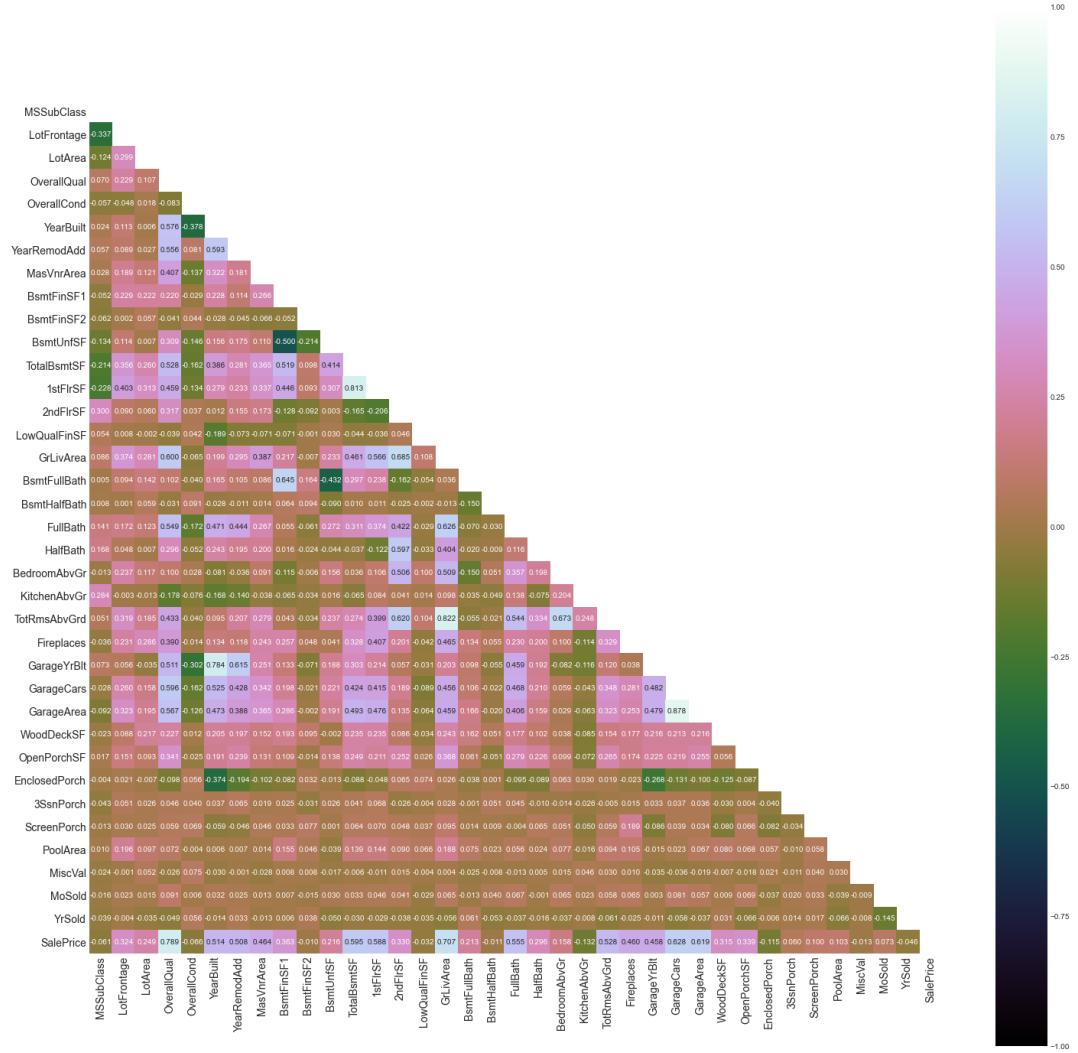
- Key Findings and Conclusions of the Study

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

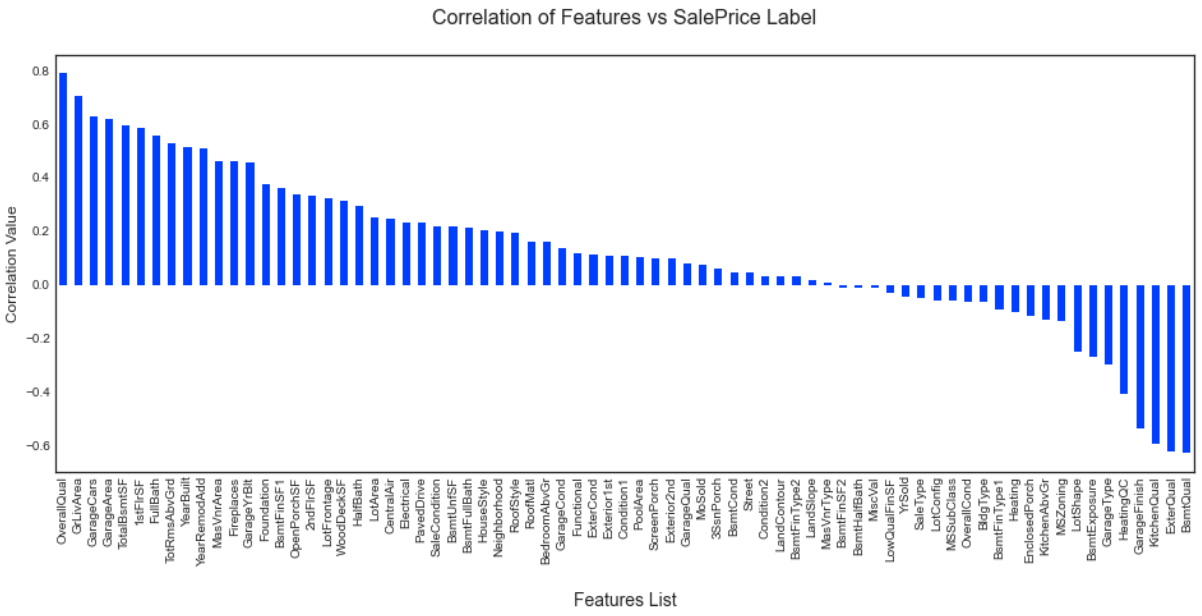
  Histogram:



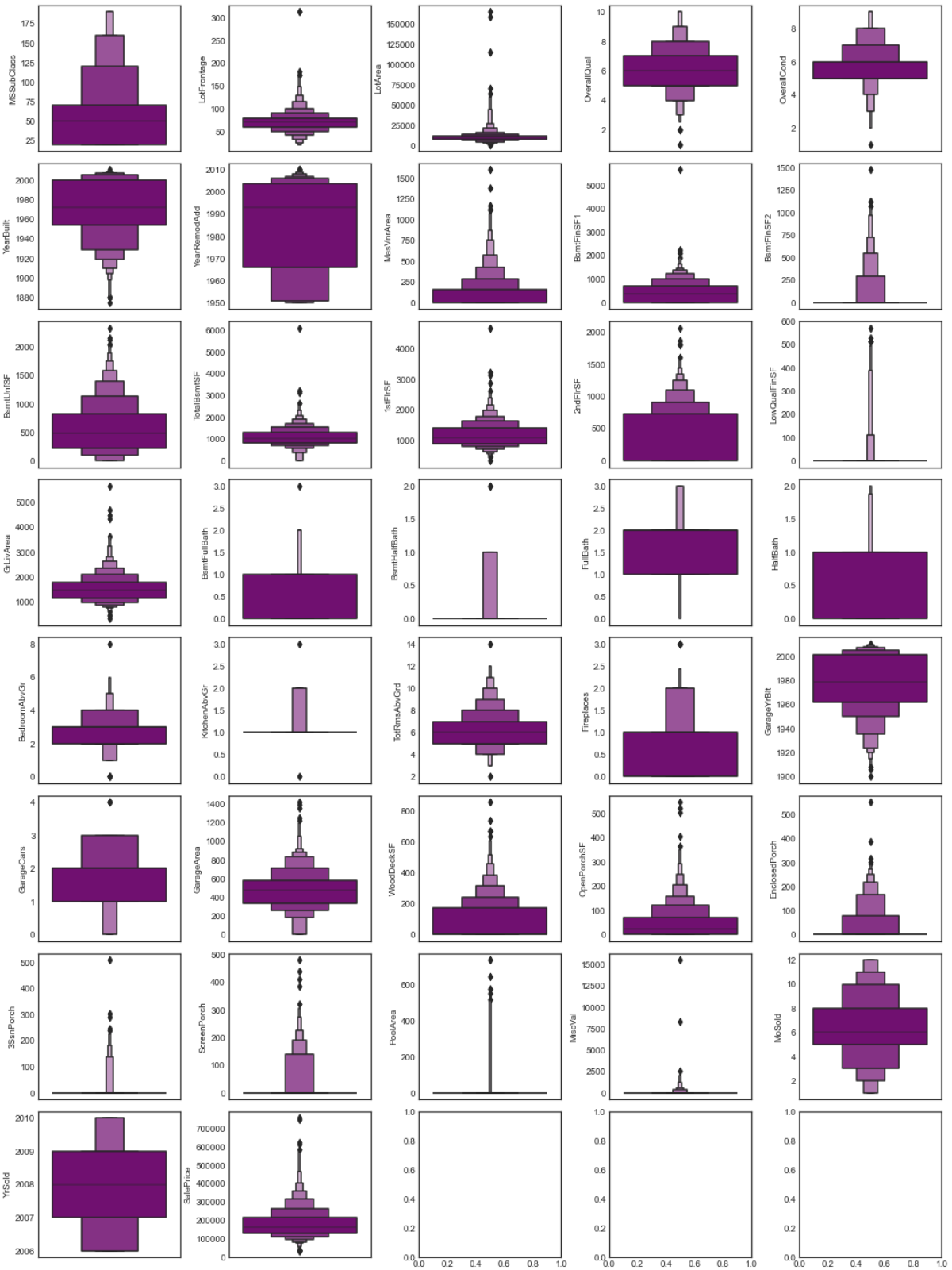

  Heatmap:



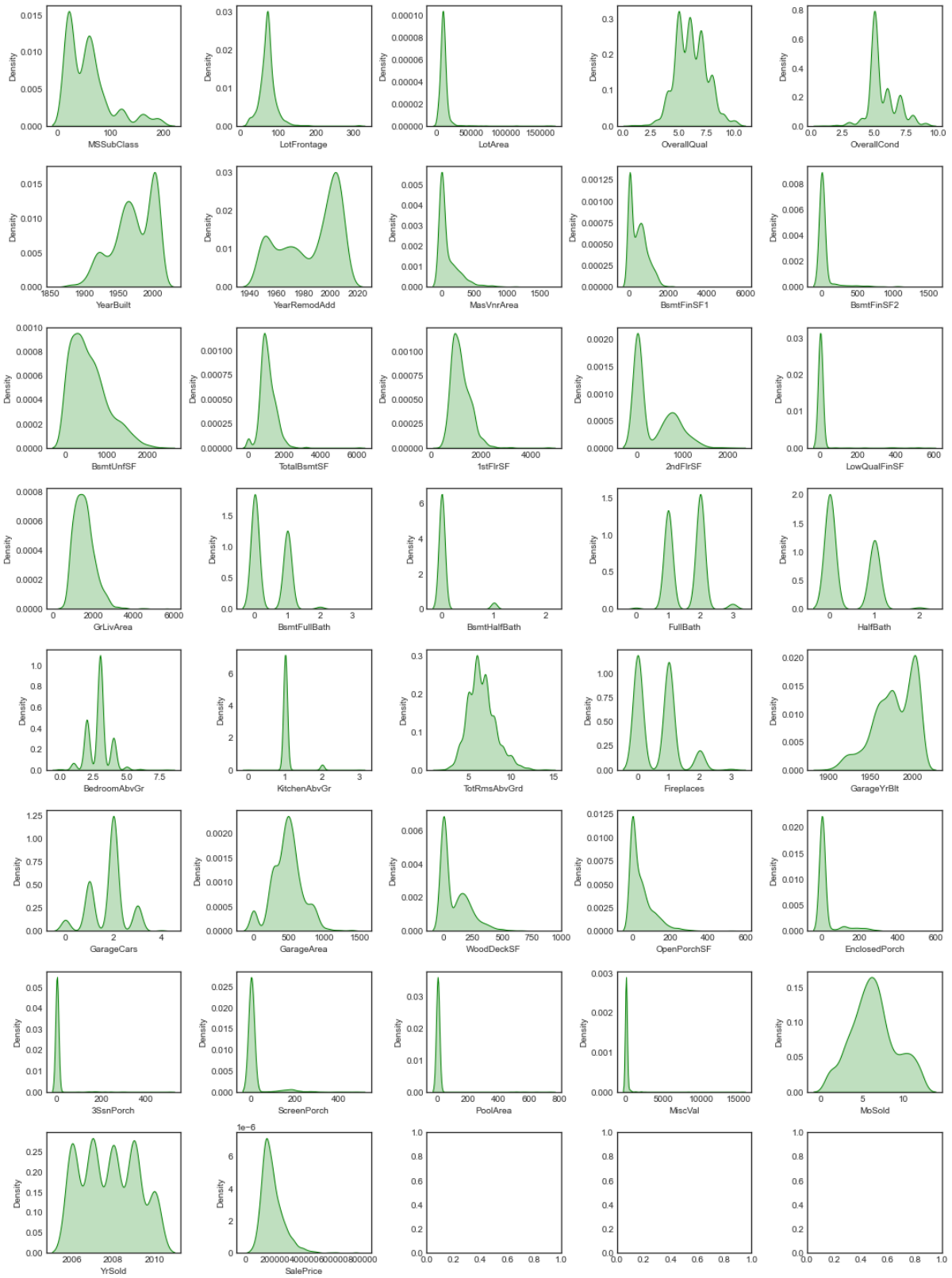

# Correlation:



Boxen Plot:


MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice

Distribution Plot:




Post model building and choosing the appropriate model I went ahead and loaded the testing dataset. After applying all the data pre

processing steps as the training dataset I was then able to get the predicted sale price results. Since the values were in array format, I converted them into a dataframe and merged it with the original testing dataframe that consisted only our feature columns. Once the testing dataset with feature columns and predicted label was formed, I exported the values in a comma separated values file to be accessed as needed.

- 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 swimming pool, garage, pavement and lawn size of Lot Area, and type of Building raise 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 the same entries in more than 80% of rows which lead to reduction in our model performance. One more issue is there are a large number of missing values present in this data set, so we have to fill those missing values in the correct manner. We can still improve our model accuracy with some feature engineering and by doing some extensive hyperparameter tuning on it.


THANK YOU