FLIP ROBO

A Project on Predicting the Housing Price
Case Report

Submitted by:

Benjamin Emmanuel

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to my SME (Subject Matter Expert) Mohd. Kashif, as well as Flip Robo Technologies, for allowing me to work on this project on Surprise Housing Price Prediction and for assisting me in conducting extensive research that allowed me to learn a lot of new things.

In addition, I used a few outside resources to help me finish the project. I made sure to learn from the samples and adjust things to fit my project's needs. The following are all of the external resources that were utilised to create this project:

1) https://www.youtube.com/
2) https://scikit-learn.org/stable/user_guide.html
3) https://www.kaggle.com/
4) https://medium.com/
5) https://towardsdatascience.com/
6) https://www.analyticsvidhya.com/

INTRODUCTION

- Business Problem Framing

  Houses are one of the most basic requirements of every individual on the planet, and hence the housing and real estate industries are one of the most important contributors to the global economy. It's a huge market with a lot of different companies operating in it.

  Data science is a significant tool for firms to employ to solve challenges in the domain, such as increasing overall revenue, profits, enhancing marketing methods, and focusing on shifting trends in home sales and purchases. Machine learning approaches like as predictive modelling, market mix modelling, and recommendation systems are employed by housing firms to achieve their business objectives. One such housing company is the source of our difficulty.

  With the supplied independent variables, we must simulate the price of dwellings. The management will then utilise this model to figure out how the prices change depending on the variables. They can then adjust the firm's strategy and focus on regions that will generate large profits. In addition, the model will help management understand the pricing dynamics of a new market.

- Conceptual Background of the Domain Problem

Surprise Housing, a housing company located in the United States, has decided to expand int o Australia. The firm employs data analytics to buy houses for less than their true worth and r esell them for more. The firm has also gathered data from Australian property sales for the sa

me purpose. The data is available as a CSV file.The firm is seeking for potential properties to purchase in order to enter the housing market. You must use Machine Learning to create a model that will estimate the true value of potential properties and help you decide whether or not to invest in them. This business wants to know:

1. Which variables are crucial in predicting a variable's price?
2. How do these variables relate to the house's price?

- Review of Literature

Based on the sample data we received from our client database, we believe the company is seeking for potential properties to purchase in order to enter the market.
The data set reveals that it is a regression problem since we need to construct a model using Machine Learning to estimate the actual worth of potential properties and determine whether or not to invest in them.
We also have other independent features that can be used to determine which factors are significant in predicting the price of a variable and how these variables characterise the housing price.

- Motivation for the Problem Undertaken

The main goal of this research is to create a model that can estimate property prices using other supporting features. Machine Learning methods will be used to predict.
We obtained the sample data from our client database. The client desires some in order to improve consumer selection projections that could aid them in increasing their investment and improving their consumer choices
The House Price Index is a popular tool for estimating house price fluctuations. Because housing prices are significantly connected with other characteristics such as location, region, and population, predicting individual property prices requires information other than HPI.
There have been a lot of studies that use typical machine learning algorithms to successfully estimate house prices, but they rarely look

at the performance of various models and ignore the less popular yet sophisticated models.
As a result, this study will use both classic and advanced machine learning methodologies to investigate the differences between numerous advanced models in order to investigate the diverse influences of features on prediction methods.
This study will also present an optimistic result for housing price predicti on by thoroughly validating different strategies in model implementation on regression.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

We're developing a Machine Learning model to forecast the actual worth of potential properties and determine whether or not to invest in them.
As a result, this model will assist us in determining which variables are critical in pred icting the price of variables, as well as how these variables define the property price.
With the provided independent factors, this will aid in determining the price of houses

. They can then adjust the firm's strategy and focus on regions that will generate large profits.

Regression analysis is a set of statistical procedures for determining the associations between a dependent variable (commonly referred to as the 'outcome variable') and one or more independent variables (often referred to as 'predictors,''covariates,' or'features'). Linear regression is the most frequent type of regression analysis, in which one finds the line (or a more sophisticated linear combination) that best fits the data according to a set of mathematical criteria. This permits the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a specified set of values for precise mathematical reasons.

- Regression analysis is a type of predictive modelling technique that examines the connection between a dependent (target) and an independent (control) variable (predictor). Forecasting, time series modelling, and determining the causal effect link between variables are all done with this technique.
- 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

  In Machine Learning, data pre-processing refers to the process of cleaning and organising raw data in order to make it appropriate for creating and training Machine Learning models. In other words, anytime data is acquired from various sources, it is obtained in raw format, which makes analysis impossible. Data pre-processing is an important stage in Machine Learning since the quality of data and the usable information that can be gleaned from it has a direct impact on our model's capacity to learn; consequently, we must pre-

process our data before feeding it into our model. As a result, it is the first and most important stage in developing a machine learning model. The following pre-processing processes were used:

Loading the training dataset as a dataframe

a. Used pandas to set display I ensuring we do not see any truncated information
b. Checked the number of rows and columns present in our training dataset
c. Checked for missing data and the number of rows with null values
d. Verified the percentage of missing data in each column and decided to discard the one's that have more than 50% of null values
e. Dropped all the unwanted columns and duplicate data present in our dataframe
f. Separated categorical column names and numeric column names in separate list variables for ease in visualization
g. Checked the unique values information in each column to get a gist for categorical data
h. Performed imputation to fill missing data using mean on numeric data and mode for categorical data columns
i. Used Pandas Profiling during the visualization phase along with pie plot, count plot, scatter plot and the others
j. With the help of ordinal encoding technique converted all object datatype columns to numeric datatype
k. Thoroughly checked for outliers and skewness information
l. 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
m. Separated feature and label data to ensure feature scaling is performed avoiding any kind of biasness
n. Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details
o. Finally created a regression model function along with evaluation metrics to pass through various model formats

- Data Inputs- Logic- Output Relationships

  We had to go through different data pre-processing processes while loading the training dataset to comprehend what was given to us and what we were expected to forecast for the project. The domain expertise of understanding how real estate works and how we are expected to serve to consumers came in helpful when it came to training the model with the modified input data in the logical section. We had to be very cautious and spent over 80% of our project building time examining each and every part of the data and how they were related to each other as well as our target label, as there is a saying in the Data Science world that "Garbage In Garbage Out."

  We needed to make sure that a model was established that identified the client priorities that were trending in the market and imposed those standards when an appropriate price tag was formed if we wanted to reliably estimate hosting sale prices. I did my best to keep as much data as possible, but I believe that removing columns with a lot of missing data was a good idea. I didn't want to impute data and then have the machine learning model be biassed by values that weren't derived from real people.

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

  For me, the hardest part was depending only on the data provided to me, keeping in mind that the different training and testing datasets were acquired from real people who were polled about their preferences and how reasonable a price for a house with certain attributes inclining to them was.

- Hardware and Software Requirements and Tools Used

  Hardware Used:

  i. RAM: 16 GB
  ii. Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59

  Software Used:

  i. Programming language: Python

ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

Pandas, NumPy, matplotlib, seaborn, scikit-learn and pandas_profiling

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  To solve the problem, I employed both statistical and analytical methodologies, which mostly included data pre-processing and EDA to assess the correlation of independent and dependent features. In addition, before feeding the input data into the machine learning models, I made sure that it was cleaned and scaled.

  We need to anticipate the sale price of houses for this project, which implies our goal column is continuous, making this a regression challenge. I evaluated the prediction using a variety of regression algorithms. After a series of evaluations, I determined that Extra Trees Regressor is the best method for our final model because it has the best r2-score and the smallest difference in r2-score and CV-score of all the algorithms tested. Other regression methods are similarly accurate, however some are over-fitting the results and others are under-fitting the results, which could be due to a lack of data.

  I used K-Fold cross validation to gain good performance and accuracy, as well as to evaluate my model for over-fitting and under-fitting, and then hyper parameter tweaked the final model.

  Once I had my desired final model, I made sure to save it before loading the testing data and beginning to do data pre-processing as

the training dataset and retrieving the anticipated sale price values from 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. Ridge Regularization Regression Model
  C. Lasso Regularization Regression Model
  D. Support Vector Regression Model
  E. Decision Tree Regression Model
  F. Random Forest Regression Model
  G. K Nearest Neighbours Regression Model
  H. Gradient Boosting Regression Model
  I. Ada Boost Regression Model
  J. Extra Trees Regression Model

- Run and Evaluate selected models

  After selecting a random state from a range of 1-1000, I employed a total of 10 Regression Models. Then I built a function for training and evaluating the regression model. The models' code is provided below.

  Random State:

```
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=i)
    lr=LinearRegression()
    lr.fit(X_train, Y_train)
    pred = lr.predict(X_test)
    r2 = r2_score(Y_test, pred)

    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best R2 score is", maxAccu,"on Random State", maxRS)
```

```
Best R2 score is 0.9201149048927508 on Random State 633
```

Regression Model Function:

Machine Learning Model for Regression with Evaluation Metrics

```
def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=663)

    # Training the model
    model.fit(X_train, Y_train)

    # Predicting Y_test
    pred = model.predict(X_test)

    # RMSE - a lower RMSE score is better than a higher one
    rmse = mean_squared_error(Y_test, pred, squared=False)
    print("RMSE Score is:", rmse)

    # R2 score
    r2 = r2_score(Y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

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

    # Result of r2 score minus cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

Linear Regression

```
model=LinearRegression()
reg(model, X, Y)
```

RMSE Score is: 0.15418751457765278
R2 Score is: 86.69919302634408
Cross Validation Score: 85.07586177914146
R2 Score - Cross Validation Score is 1.6233312472026142

Ridge Regulariation

```
model=Ridge(alpha=1e-2, normalize=True)
reg(model, X, Y)
```

RMSE Score is: 0.15419607075216238
R2 Score is: 86.69771680846091
Cross Validation Score: 85.22815539398036
R2 Score - Cross Validation Score is 1.4695614144805518

Lasso Regularization

```
model=Lasso(alpha=1e-2, normalize=True, max_iter=1e5)
reg(model, X, Y)
```

RMSE Score is: 0.4057402752141291
R2 Score is: 7.896567874037297
Cross Validation Score: 6.295789540089332
R2 Score - Cross Validation Score is 1.6007783339479653

Support Vector Regression

```
model=SVR(C=1.0, epsilon=0.2, kernel='poly', gamma='auto')
reg(model, X, Y)
```

RMSE Score is: 0.2533006843854383
R2 Score is: 64.10348382913983
Cross Validation Score: 69.8866156478451
R2 Score - Cross Validation Score is -5.783131818705272

Decision Tree Regressor

```
model=DecisionTreeRegressor(criterion="poisson", random_state=111)
reg(model, X, Y)
```

RMSE Score is: 0.324660774413689
R2 Score is: 41.028901803862674
Cross Validation Score: 52.2735483549674
R2 Score - Cross Validation Score is -11.24464655110473

Random Fores Regressor

```
model=RandomForestRegressor(max_depth=2, max_features="sqrt")
reg(model, X, Y)
```

RMSE Score is: 0.2715085628918895
R2 Score is: 58.75734292732666
Cross Validation Score: 64.17290594747367
R2 Score - Cross Validation Score is -5.415563020147005

K-Neighbors Rgeressor

```
KNeighborsRegressor(n_neighbors=2, algorithm='kd_tree')
reg(model, X, Y)
```

RMSE Score is: 0.26579832096601697
R2 Score is: 60.47389304588543
Cross Validation Score: 63.39361437910286
R2 Score - Cross Validation Score is -2.9197213332174243

Gradient Boosting Regressor

```
model=GradientBoostingRegressor(loss='quantile', n_estimators=200, max_depth=5)
reg(model, X, Y)
```

RMSE Score is: 0.254754523749564
R2 Score is: 63.69023949631658
Cross Validation Score: 70.15779137747798
R2 Score - Cross Validation Score is -6.467551881161398

Ada Boost Regressor

```
model=AdaBoostRegressor(n_estimators=300, learning_rate=1.05, random_state=42)
reg(model, X, Y)
```

RMSE Score is: 0.17895172976073248
R2 Score is: 82.08357415401066
Cross Validation Score: 81.01442217082004
R2 Score - Cross Validation Score is 1.069151983190622

Extra Trees Regressor

```
model=ExtraTreesRegressor(n_estimators=200, max_features='sqrt', n_jobs=6)
reg(model, X, Y)
```

RMSE Score is: 0.17979948233325582
R2 Score is: 81.9134201910449
Cross Validation Score: 85.70280138806163
R2 Score - Cross Validation Score is -3.7893811970167235

- Key Metrics for success in solving problem under consideration

  R2 score, cross val score, MAE, MSE, and RMSE were the main metrics employed in this study. We used Hyperparameter Tuning to find the optimal parameters and to improve our scores, and we'll be using the GridSearchCV method to do it.

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

where each part being 20% of full dataset. While running the 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.
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 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.

Hyper parameter tuning

```
fmod_param = {'alpha':[0,0.1,1,10,100],
              'random_state' : [42, 111, 340,663]
             }
```

```
GSCV = GridSearchCV(Ridge(), fmod_param, cv=5)
```

```
GSCV.fit(X_train,Y_train)
```

```
Final_Model = Ridge(alpha=100, random_state=663)
Model_Training = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_r2 = r2_score(Y_test, fmod_pred, multioutput='variance_weighted')*100
print("R2 score for the Best Model is:", fmod_r2)
```

```
R2 score for the Best Model is: 85.23701730624937
```

It is possible that the default settings work better than the parameters list produced after tweaking, but this just means that there are more permutations and combinations to go through in order to achieve better results.

- Visualizations

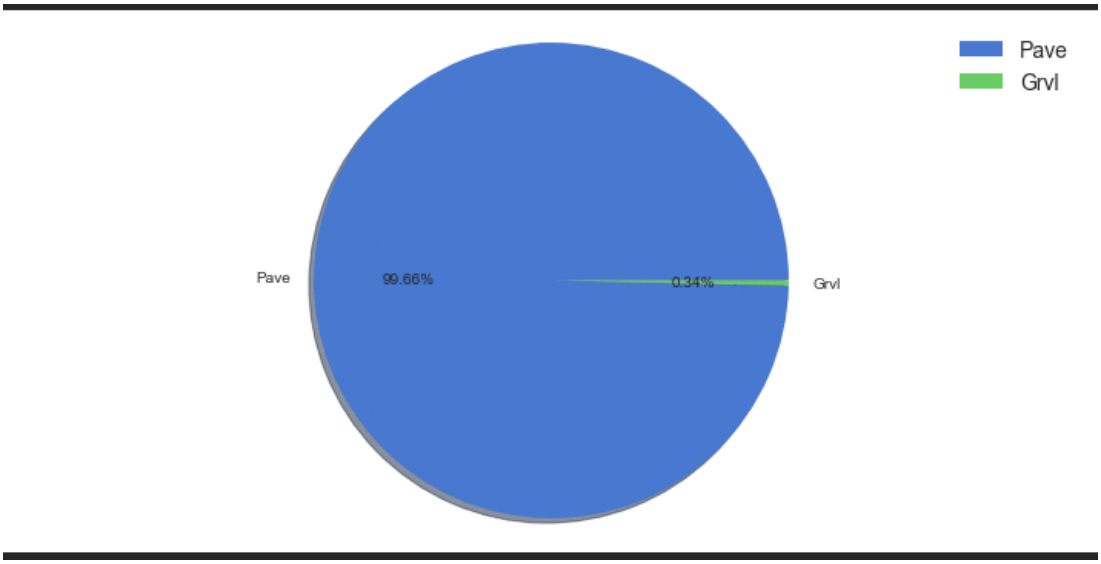
I then created pie plots, count plots and scatter plots to get further visual insights on our training dataset feature values.

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:



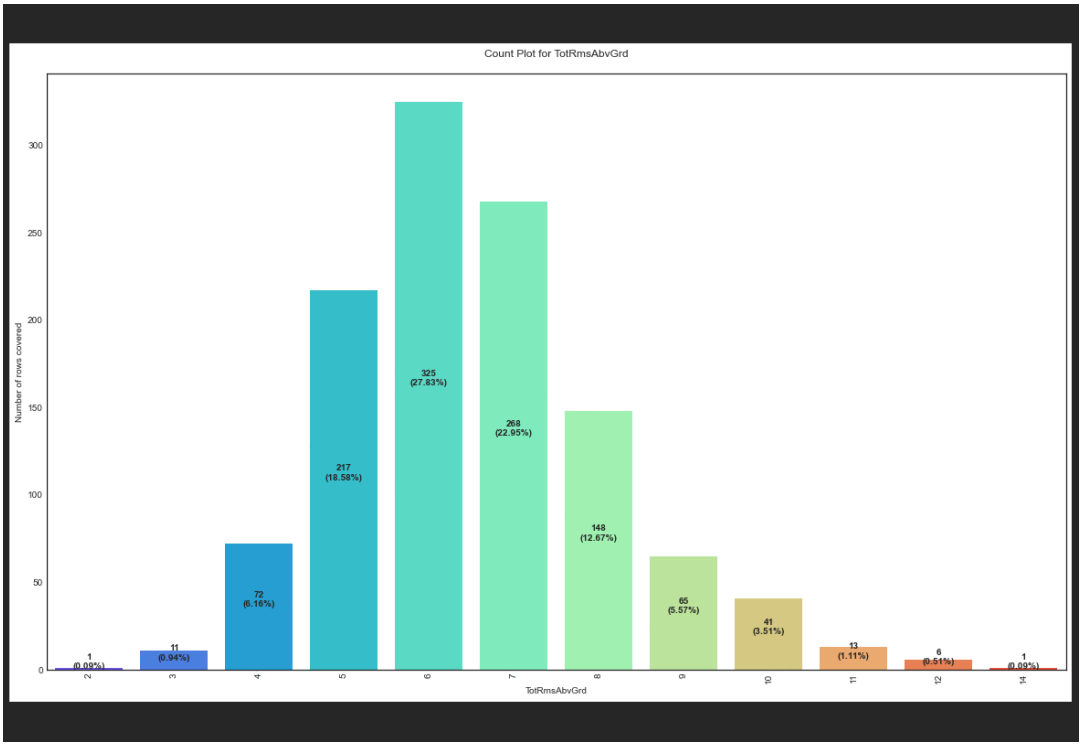
Code:

```
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()
```

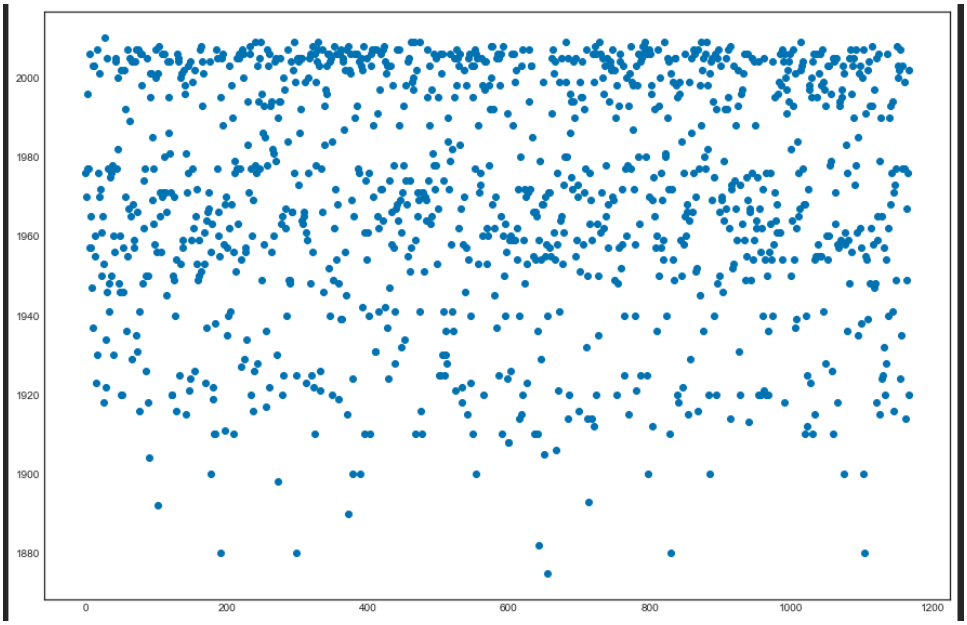
Output:



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

  It helped me comprehend the relationship between independent and dependent properties through visualisations. Also, it assisted me in determining the relevance of features and checking for multi collinearity issues. Boxplot and distribution plot were used to detect outliers/skewness. The count of a given category for each feature was determined using the count plot, and the anticipated target value distribution, as well as the scatter plot, assisted me in selecting the optimum model.

  Pre-processing: Basically, the dataset should be cleaned and resized before developing the model by following a few procedures. As I indicated in the pre-processing procedures, the dataset has all of the relevant features and is ready for model creation.

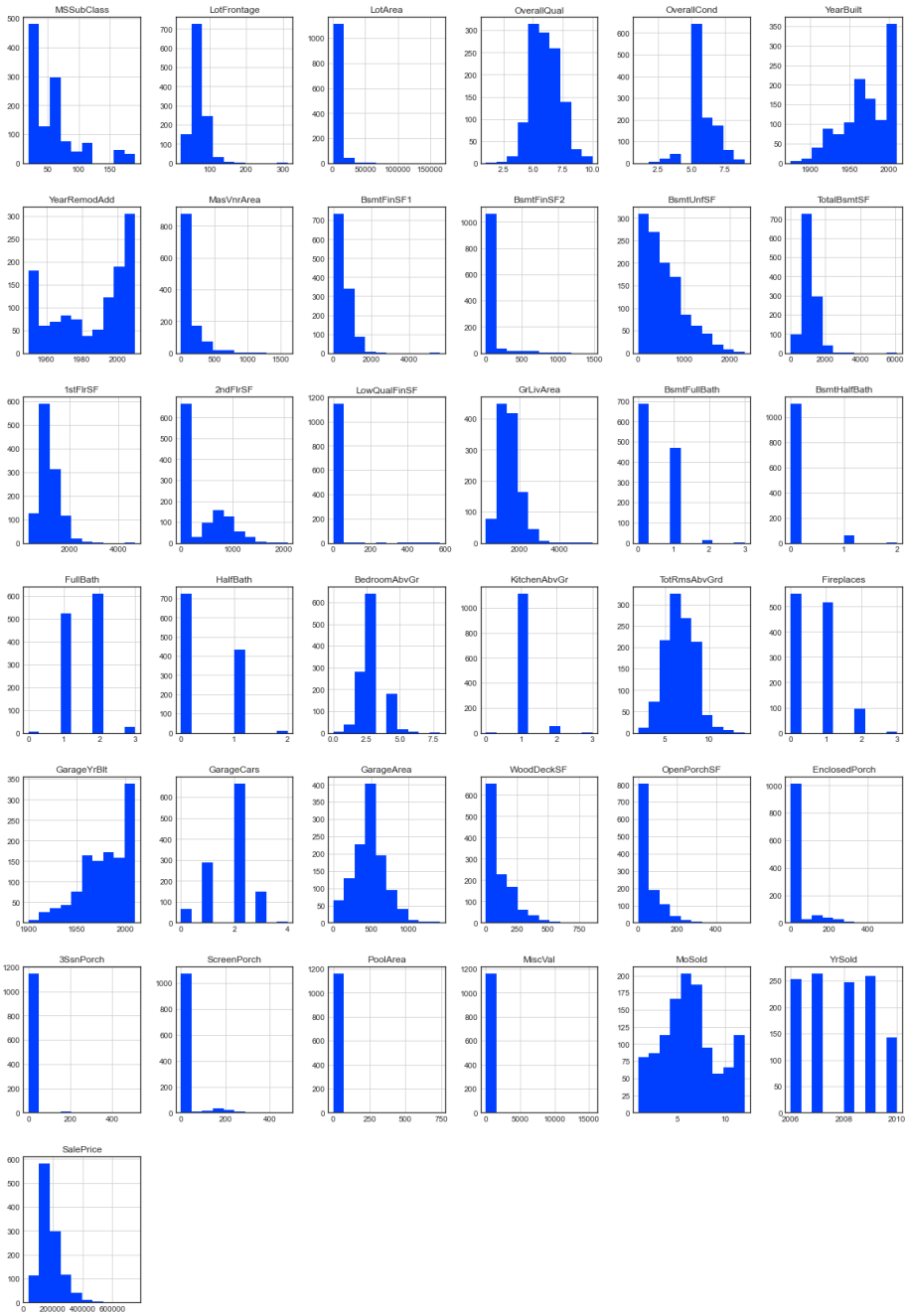
  Model Creation: Now, after performing the train test split, I have x_train, x_test, y_train & y_test, which are required to build Machine learning models. I have built multiple regression models to get the best R2 score, MSE, RMSE & MAE out of all the models.

CONCLUSION

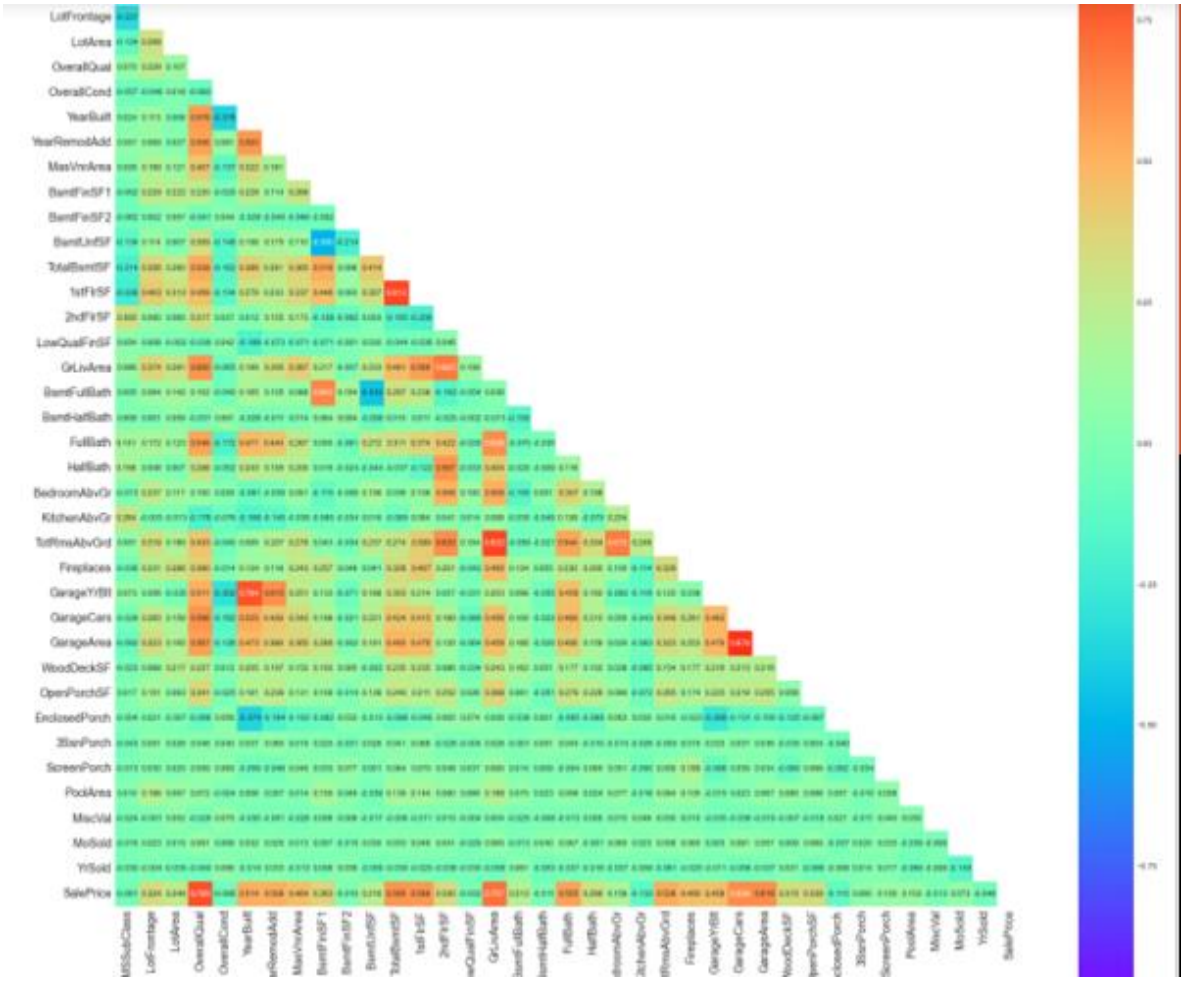
- # Key Findings and Conclusions of the Study

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

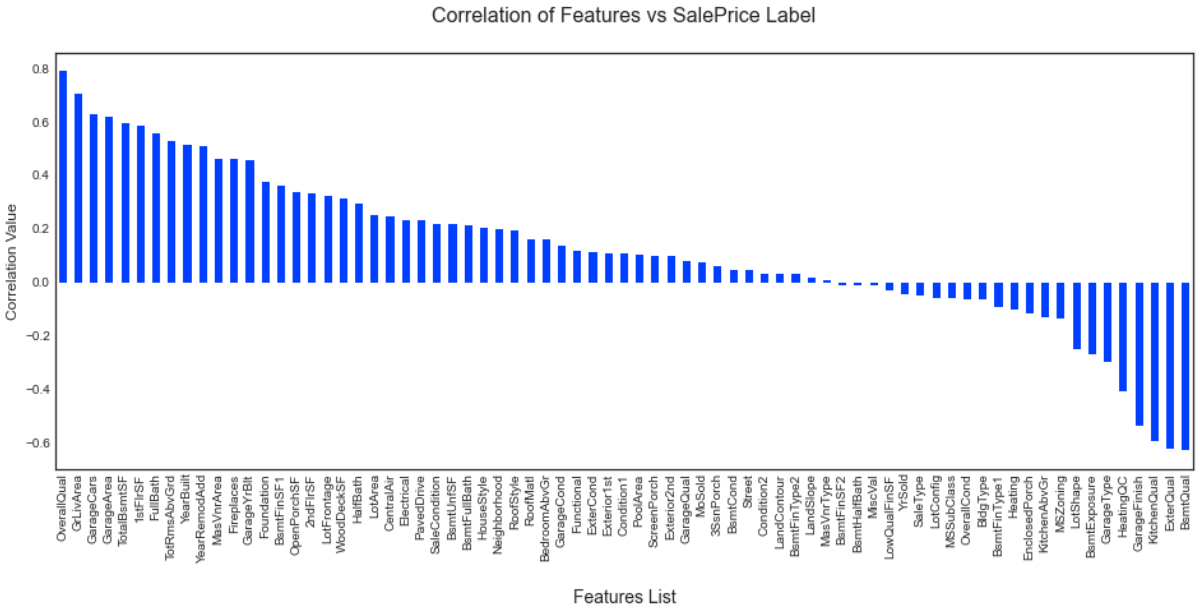
  Histogram:



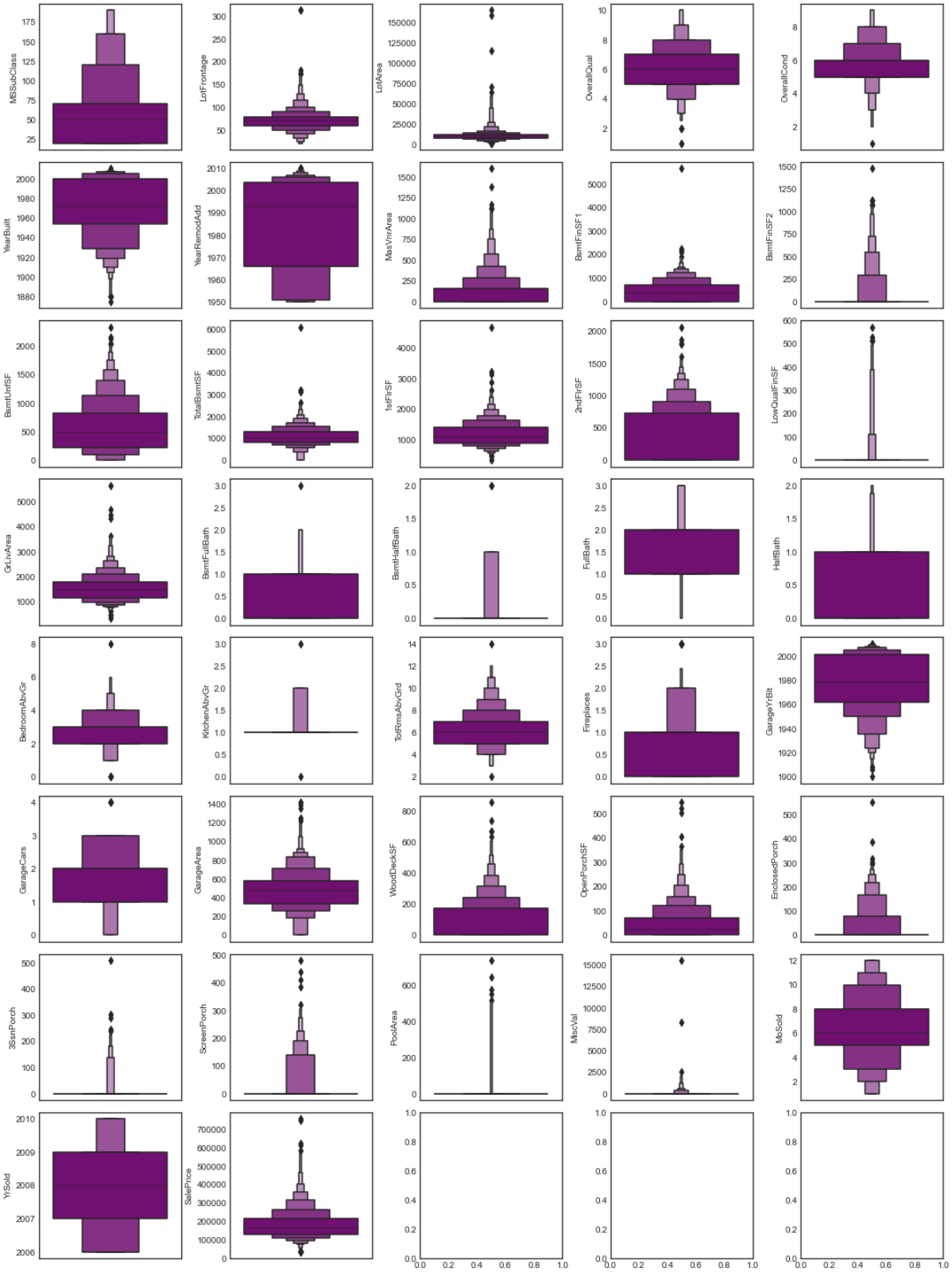
  Heatmap:



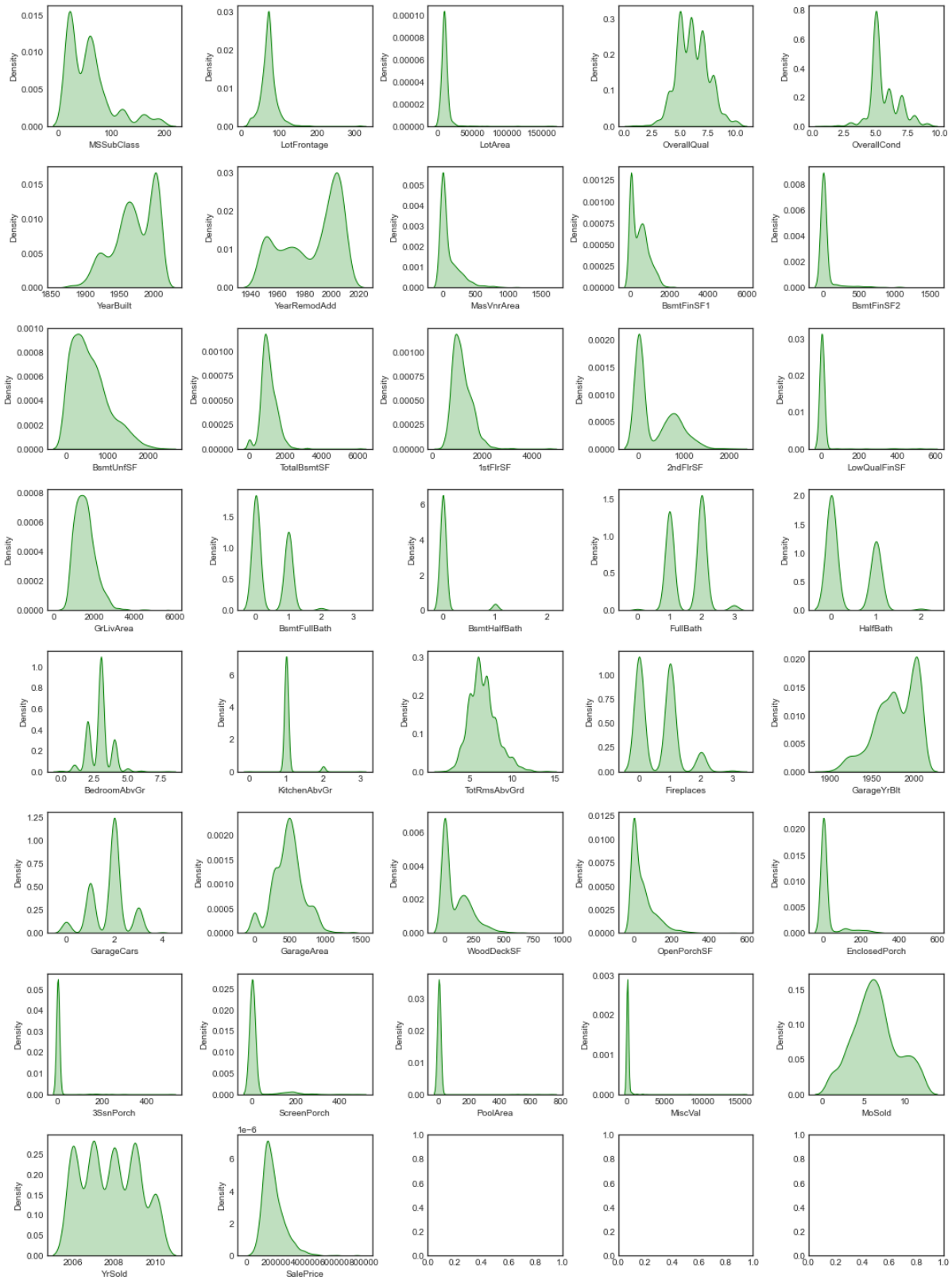
Correlation:



Boxen Plot:



Distribution Plot:



I loaded the testing dataset after constructing the model and selecting the suitable model. I was able to achieve the anticipated selling price findings after applying all of the data preprocessing stages to the training dataset. I transformed the values into a dataframe and

combined it with the original testing dataframe, which only contained our feature columns, because they were in array format. I exported the values in a comma separated values file to be accessed as needed after the testing dataset with feature columns and projected label was created..

- Learning Outcomes of the Study in respect of Data Science

  The aforementioned research aids in the understanding of the real estate industry. How the price of the homes is changing. We may tell how the cost is determined by a variety of real estate amenities such as a swimming pool, garage, pavement, and lawn, as well as the size of the lot area and the type of building. With the foregoing study, we may sketch the wants of a property buyer and predict the price of the property based on those needs.

- Limitations of this work and Scope for Future Work

I had a data shortage during this project. Many columns have the identical entries in more than 80% of the rows, reducing the performance of our model. Another issue is that this data collection contains a big number of missing values, so we must fill those missing values correctly. With some feature engineering and rigorous hyperparameter adjustment, we can still enhance the accuracy of our model.