FLIP ROBO

Housing Price Prediction Project Report

Submitted by:

Anubhav Saxena

ACKNOWLEDGMENT

I'd want to thank Shubham Yadav, my SME (Subject Matter Expert), and Flip Robo Technologies for enabling me to work on this project on housing price prediction and assisting me in performing comprehensive research that allowed me to learn a lot of new things.

In addition, I used a few other resources to assist me finish the job. 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.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION

- **Business Problem Framing**

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

Data science has emerged as a critical tool for firms to employ to address challenges in the sector, such as increasing total income and profitability, 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 firm 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 factors. They may then adjust the firm's strategy and focus on regions that will provide large profits. Furthermore, the model would assist management in comprehending the price dynamics of a new market.

- **Conceptual Background of the Domain Problem**

Surprise Housing, a housing firm located in the United States, has chosen to join the Australian market. The firm employs data analytics to buy properties for less than their true value and then resell them for more. The business has also gathered data from property sales in Australia for the same purpose. The information may be found in the CSV file below.

The business is seeking for potential properties to purchase in order to enter the market. You must use Machine Learning to create a model that will estimate the real worth of potential properties and help you decide whether or not to invest in them. For this firm, the following information is required:

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 firm is seeking for potential properties to purchase in order to join 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 additional independent features that may be used to determine which factors are significant in predicting the price of a variable and how these variables characterise the house's 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 forecast.
We obtained the sample data from our customer database. In order to increase customer selection, the client requests certain forecasts that will assist them in making future investments and improving customer selection.

The House Price Index is a popular tool for estimating housing price changes. Because housing prices are significantly connected with other characteristics such as location, region, and population, predicting individual house 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 seldom look at the performance of different models and ignore the less popular yet sophisticated models.

As a result, this study will use both classic and advanced machine learning methodologies to analyse the differences between numerous advanced models in order to evaluate the diverse influences of features on prediction methods. This research will also present an optimistic outcome for housing price prediction by thoroughly validating numerous strategies in model implementation on regression.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  We're using Machine Learning to create a model that will estimate the real worth of potential properties and help us decide whether or not to invest in them. As a result, this model will assist us in determining which factors are critical in predicting the price of variables, as well as how these variables characterise the house's price. With the provided independent factors, this will aid in determining the price of dwellings. They may then adjust the firm's strategy and focus on regions that will provide large profits.

  Regression analysis is a collection of statistical procedures for evaluating 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 approach that examines the connection between a dependent (target) and an independent (control) variable (predictor). Forecasting, time series modelling, and determining the cause effect link between variables are all done with this approach.

- ## Data Sources and their formats

  The data given by Flip Robo was in CSV format (Comma Separated Values). There are 1168 rows and 81 columns in the data. There are two data sets available. There are two types of data: training data and testing data.

1) The train file will be used to train the model, which means that the model will learn from it. All of the independent variables are included, as well as the target variable. The training set has 1168 records.
2) The test file includes all of the independent variables except the target variable. For the test data, we'll use the model to forecast the target variable. The test set has 292 records.

- ## Data Pre-processing 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 received from various sources, it is collected in raw format, which makes analysis impossible. Data pre-processing is a crucial stage in Machine Learning since the quality of data and the relevant 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:

a. Loading the training dataset as a data frame
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 one's 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 datatype 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**

We had to go through different data pre-processing processes while loading the training dataset to comprehend what was supplied to us and what we were supposed to forecast for the project. The domain experience of knowing 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 changed input data in the logical section. We had to be very cautious and spent over 80% of our project building time studying each and every part of the data and how they were connected to each other as well as our goal label, since there is a saying in the Data Science community: "Garbage In, Garbage Out."

We needed to make sure that a model was established that identified the consumer 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 smart idea. I didn't want to impute data and then have the machine learning model be skewed by numbers 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 supplied to me, keeping in mind that the different training and testing datasets were acquired from actual individuals who were polled about their preferences and how acceptable a price for a house with certain attributes inclining to them was.

- **Hardware and Software Requirements and Tools Used**

Hardware Used:

✓ RAM: 8 GB

✓ Processor: Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz

Software Used:

✓ Programming language: Python
✓ Distribution: Anaconda Navigator
✓ 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 tackle the problem, I employed both statistical and analytical methodologies, which mostly included data pre-processing and EDA to examine the connection of independent and dependent characteristics. 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 methods. After a series of assessments, I determined that Extra Trees Regressor is the best method for our final model since 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 findings and others are under-fitting the results, which might be due to a lack of data.

  I used K-Fold cross validation to gain high performance and accuracy, as well as to check for over-fitting and under-fitting in my model, 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 selling 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 may be seen below.

Random State:

Finding the best random state for building Regression Models

```
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 0.8856355344351948 on Random State 340
```

Regression Model Function:

Machine Learning Model for Regression with Evaluation Metrics

```
# Regression Model Function

def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=340)

    # 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:

```
# Linear Regression Model

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

```
RMSE Score is: 24876.373485691707
R2 Score is: 88.56355344351948
Cross Validation Score: 74.14529018813273
R2 Score - Cross Validation Score is 14.418263255386748
```

Ridge Regularization:

```
# Ridge Regularization

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

```
RMSE Score is: 24815.18998074428
R2 Score is: 88.6197402024921
Cross Validation Score: 74.45483255058483
R2 Score - Cross Validation Score is 14.16490765190727
```

Lasso Regularization:

```python
# Lasso Regularization

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

```
RMSE Score is: 24917.18385422086
R2 Score is: 88.52599905988447
Cross Validation Score: 74.1554161073105
R2 Score - Cross Validation Score is 14.370582952573969
```

Support Vector Regressor:

```python
# Support Vector Regression

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

```
RMSE Score is: 76592.05128076131
R2 Score is: -8.413750687388166
Cross Validation Score: -6.214424099645246
R2 Score - Cross Validation Score is -2.1993265877429202
```

Decision Tree Regressor:

```python
# Decision Tree Regressor

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

```
RMSE Score is: 57727.62379648374
R2 Score is: 38.41366921116711
Cross Validation Score: 41.26696984258857
R2 Score - Cross Validation Score is -2.8533006314214617
```

Random Forest Regressor:

```
# Random Forest Regressor

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

```
RMSE Score is: 40625.4396140173
R2 Score is: 69.49906765983303
Cross Validation Score: 64.61456200338246
R2 Score - Cross Validation Score is 4.884505656450571
```

K Nearest Neighbours Regressor:

```
# K Neighbors Regressor

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

```
RMSE Score is: 40466.730494501026
R2 Score is: 69.73691471173798
Cross Validation Score: 64.42251920085333
R2 Score - Cross Validation Score is 5.314395510884651
```

Gradient Boosting Regressor:

```
# Gradient Boosting Regressor

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

```
RMSE Score is: 34539.463803694656
R2 Score is: 77.95306863017093
Cross Validation Score: 78.2983938466606
R2 Score - Cross Validation Score is -0.34532521648966963
```

Ada Boost Regressor:

```python
# Ada Boost Regressor

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

```
RMSE Score is: 31820.346272586143
R2 Score is: 81.28771728128767
Cross Validation Score: 79.16566313678824
R2 Score - Cross Validation Score is 2.1220541444994296
```

Extra Trees Regressor:

```python
# Extra Trees Regressor

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

```
RMSE Score is: 23816.88408105236
R2 Score is: 89.51696939850329
Cross Validation Score: 84.8703100074016
R2 Score - Cross Validation Score is 4.646659391101693
```

- ## 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 GridSearchCV method.

1. Cross Validation:

Cross-validation aids in determining the model's overfitting and underfitting. The model is constructed to run on several subsets of the dataset in cross validation, resulting in numerous measurements of the model. If we fold the data five times, it will be separated into five parts, each representing 20% of the whole dataset. During the Cross-validation, the first part (20%) of the 5 parts will be left out as a holdout set for validation, while the rest of the data will be utilised

for training. We'll acquire the initial estimate of the dataset's model quality this way.
Further rounds are produced in the same way for the second 20% of the dataset, which is kept as a holdout set while the remaining four portions are utilised for training data during the process. We'll acquire the second estimate of the dataset's model quality this way. During the cross-validation procedure, these stages are repeated to obtain the remaining estimate of model quality.

2. R2 Score:

It is a statistical metric that indicates the regression model's quality of fit. The optimal r-square value is 1. The closer the r-square value is to 1, the better the model fits.

3. Mean Squared Error (MSE):
The average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated — is measured by the MSE of an estimator (of a process for estimating an unobserved variable). MSE is a risk function that represents the squared error loss's anticipated value. The Root Mean Squared Error is abbreviated as RMSE.

4. Mean Absolute Error (MAE):

MAE is a statistic that assesses the average magnitude of mistakes in a set of forecasts without taking into account their direction. It's the average of the absolute differences between forecast and actual observation over the test sample, where all individual deviations are given equal weight.

5. Hyperparameter Tuning:

There is a list of several machine learning models available. They're all distinct in some manner, yet the only thing that distinguishes them is the model's input parameters. Hyperparameters are the name given to these input parameters. These hyperparameters will

establish the model's architecture, and the greatest thing is that you get to choose the ones you want for your model. Because the list of hyperparameters for each model differs, you must choose from a distinct list for each model.
We are unaware of the ideal hyperparameter settings that would produce the best model output. So, we instruct the model to automatically explore and choose the best model architecture. Hyperparameter tuning is the term for the method of selecting hyperparameters. GridSearchCV may be used to tune the system. GridSearchCV is a model selection function in the Scikit-learn (or SK-learn) package. It is vital to remember that the Scikit-learn library must be installed on the PC. This function aids in fitting your estimator (model) to your training set by looping over specified hyperparameters. Finally, we may choose the optimal settings from the hyperparameters presented.

Hyper parameter tuning

```
# Choosing Extra Trees Regressor

fmod_param = {'n_estimators' : [100, 200, 300],
              'criterion' : ['squared_error', 'mse', 'absolute_error', 'mae'],
              'n_jobs' : [-2, -1, 1],
              'random_state' : [42, 111, 340]
             }
```

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

I am using the Grid Search CV method for hyper parameter tuning my best model.

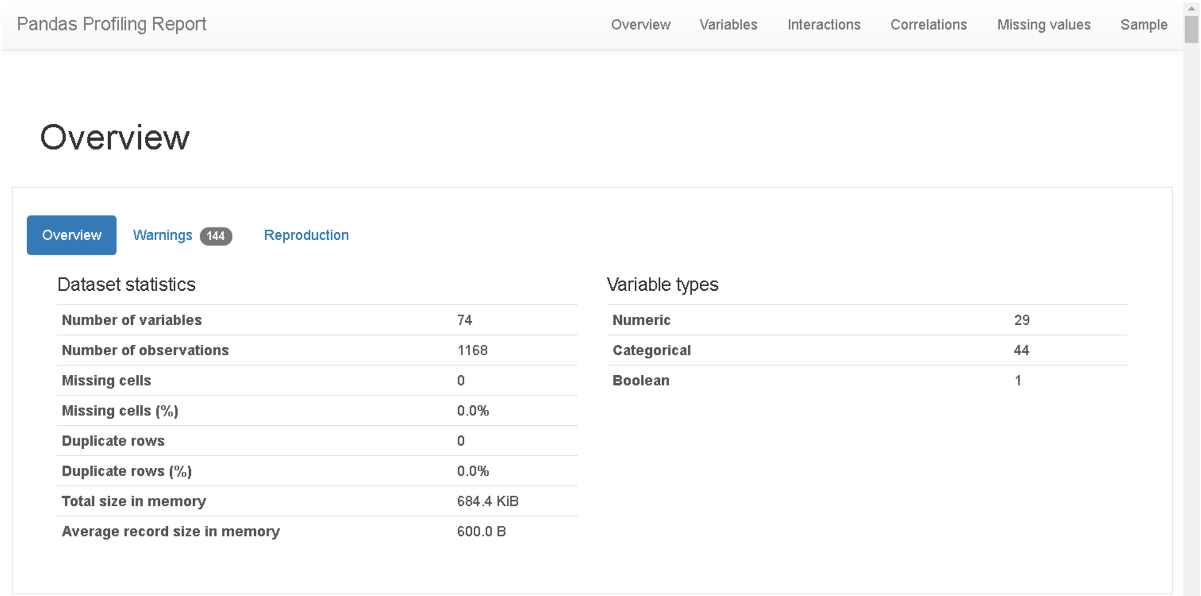
```
Final_Model = ExtraTreesRegressor(criterion='mse', n_estimators=100, n_jobs=-2, random_state=42)
Model_Training = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_r2 = r2_score(Y_test, fmod_pred, multioutput='variance_weighted')*100
print("R2 score for the Best Model is:", fmod_r2)
```

```
R2 score for the Best Model is: 83.64443386563624
```

It is conceivable 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

To generate the above-mentioned visualisation on the pre-processed data, I utilised pandas profiling. pandas-profiling is a free Python module that allows us to perform exploratory data analysis with just a few lines of code. It creates interactive online reports that may be delivered to anybody, even if they have no programming experience. It also provides report production for the dataset, with a variety of features and customizations. In other words, pandas-profiling saves us the time and effort of seeing and comprehending each variable's distribution. It creates a report with all of the data in one place.



Number of variables	74
Number of observations	1168
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	684.4 KiB
Average record size in memory	600.0 B

Numeric	29
Categorical	44
Boolean	1

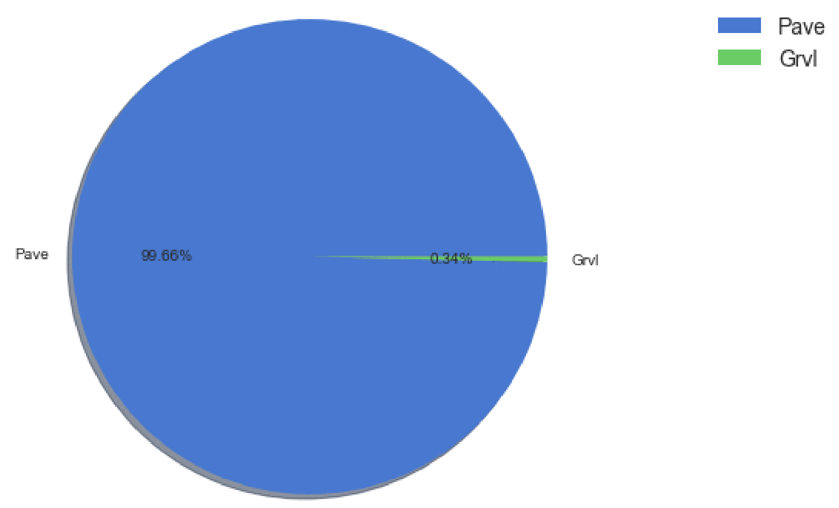
To have a better visual understanding of our training dataset feature values, I constructed pie charts, count plots, and scatter plots.

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:



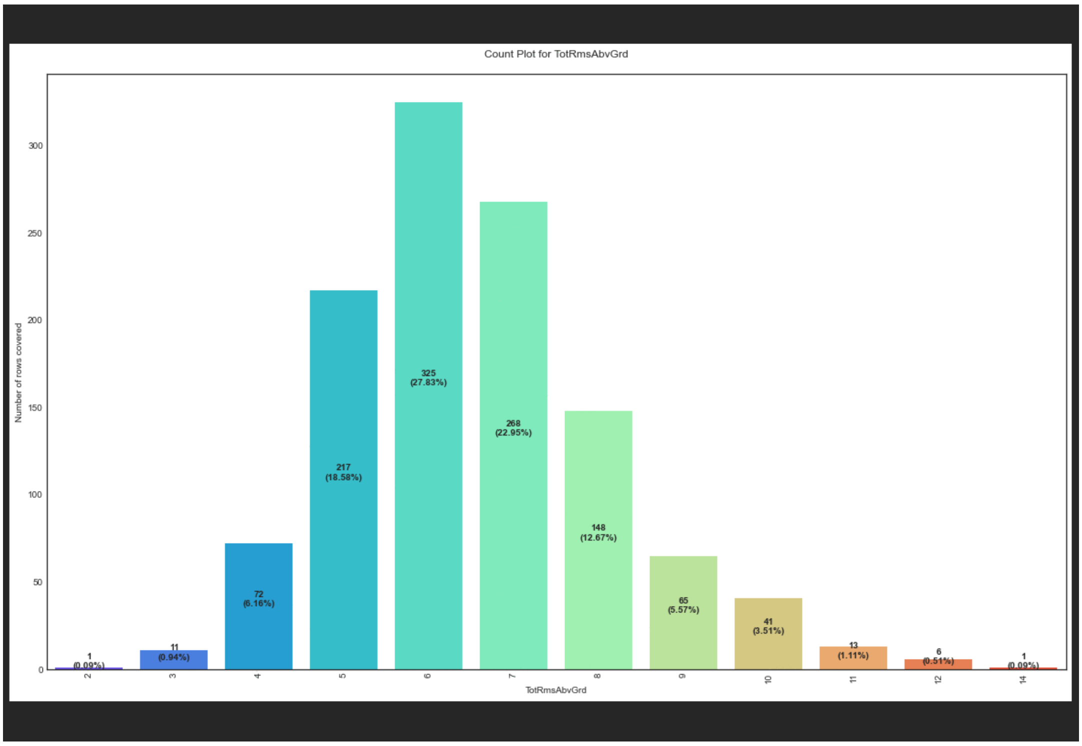
Code:

```
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()
```

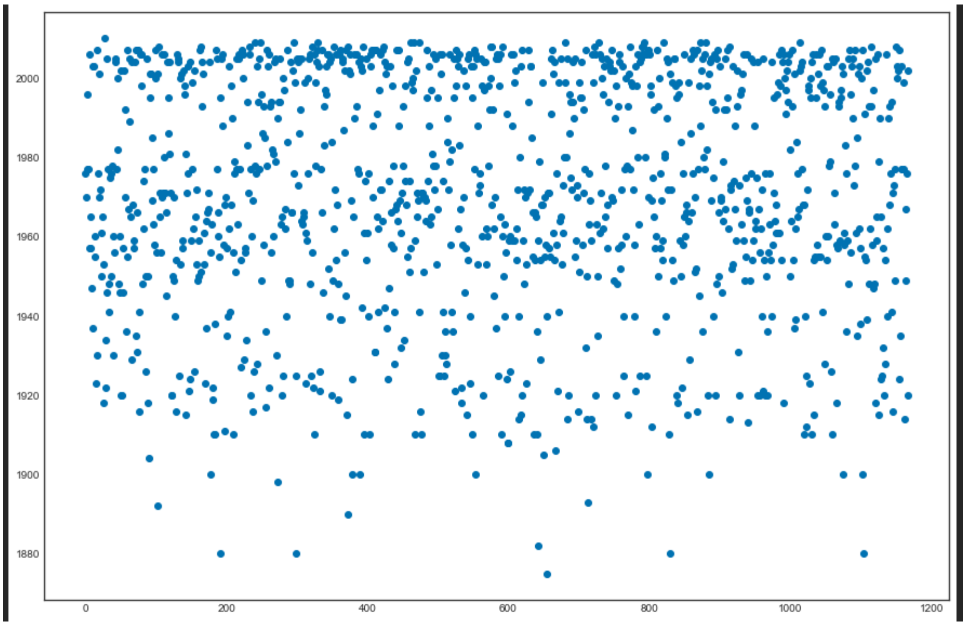
Output:



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 assisted me in comprehending the relationship between independent and dependent characteristics. Also, it assisted me in determining the relevance of features and checking for multi-collinearity concerns. 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 scaled before developing the model by following a few procedures. As I indicated before in the pre-processing phases, the dataset contains all of the necessary characteristics and is ready for model creation.

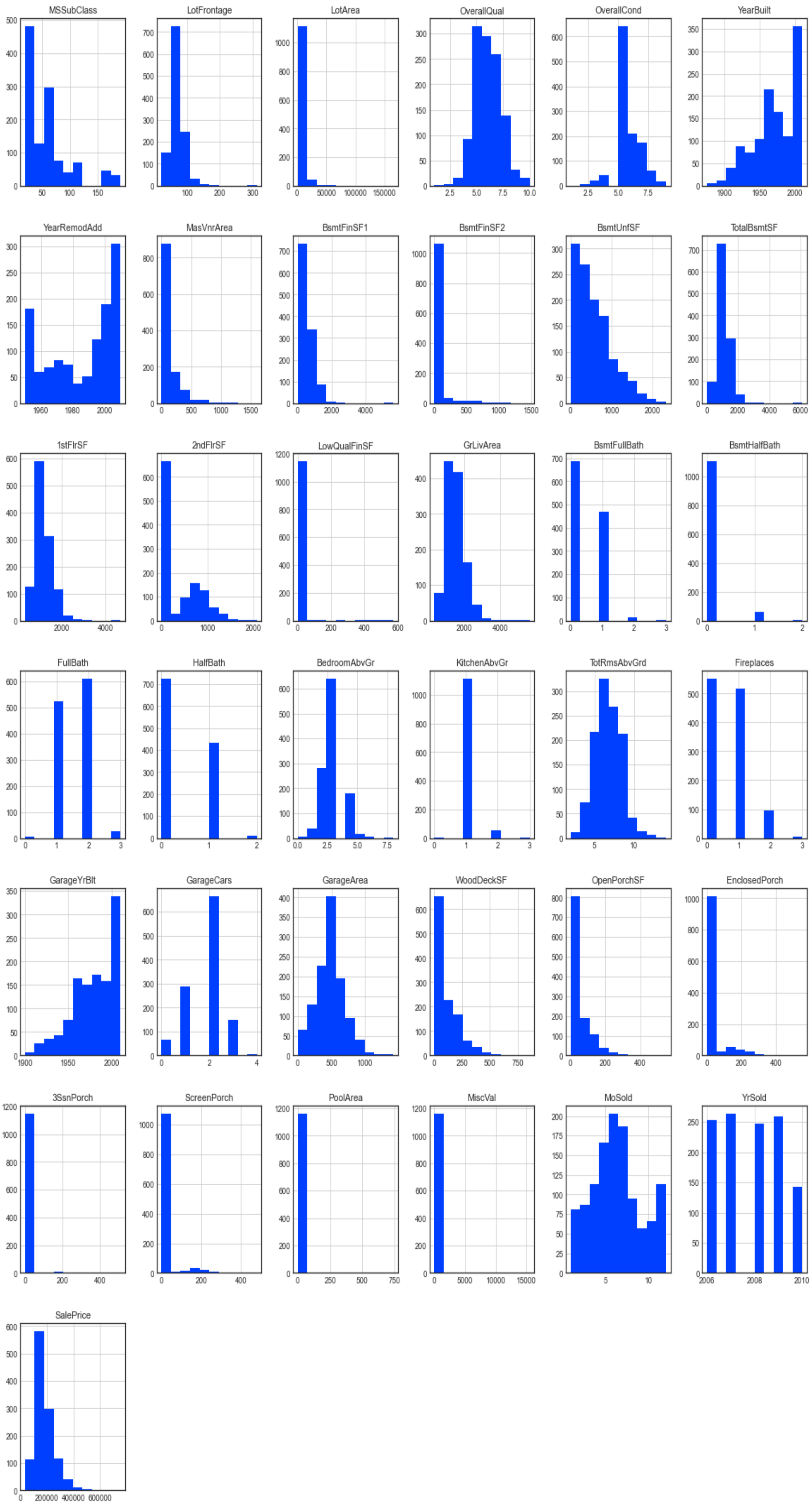
Model Creation: Now that I've divided the train and test data, I have x train, x test, y train, and y test, all of which are needed to create Machine Learning models. I generated numerous regression models to see which one had the greatest R2 score, MSE, RMSE, and MAE.

CONCLUSION

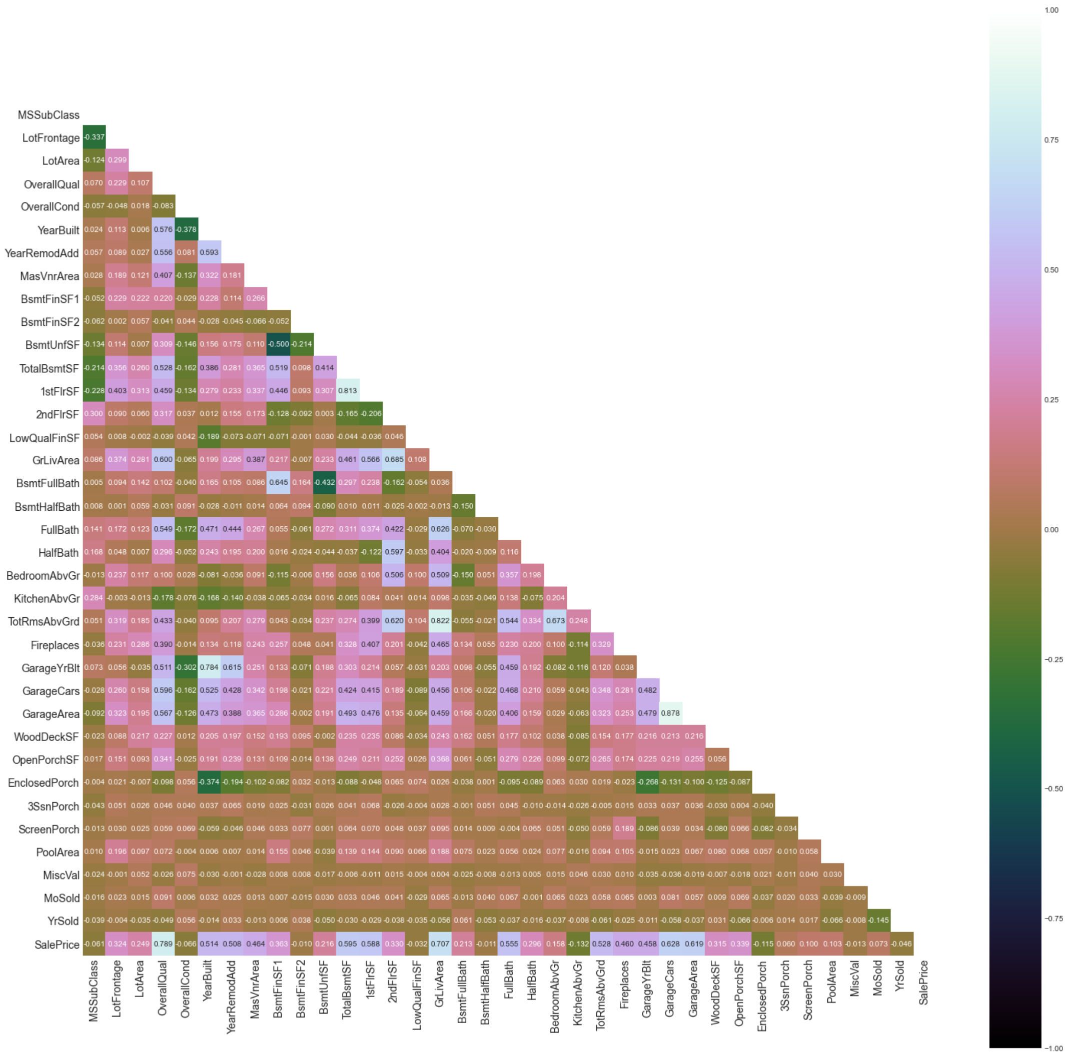
- **Key Findings and Conclusions of the Study**

By producing numerous graphs and visualising more insights, I was able to view all of the encoded dataset information.

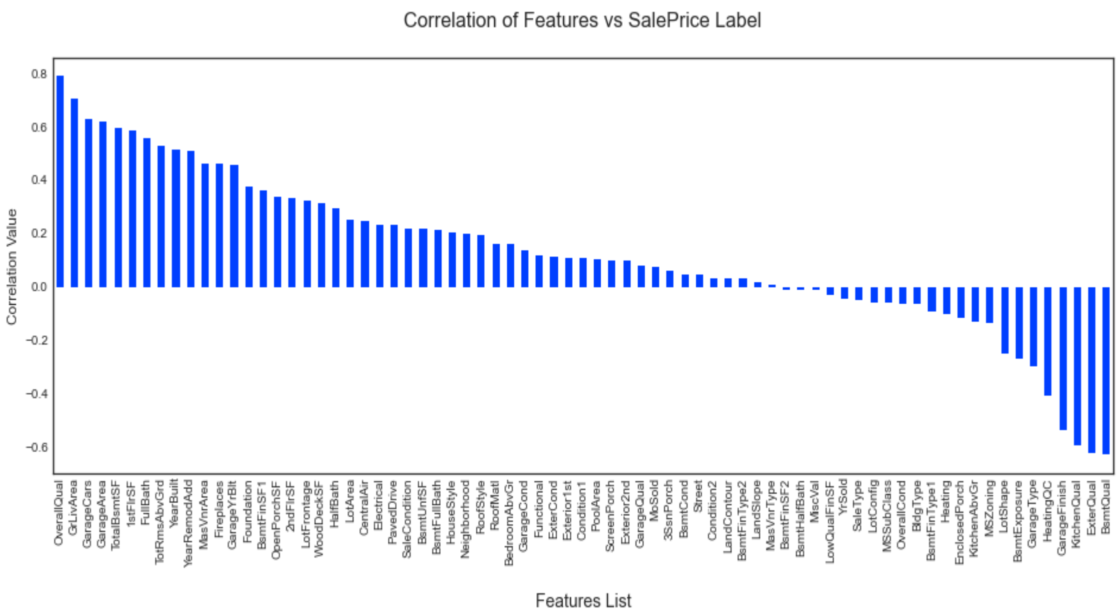
Histogram:


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

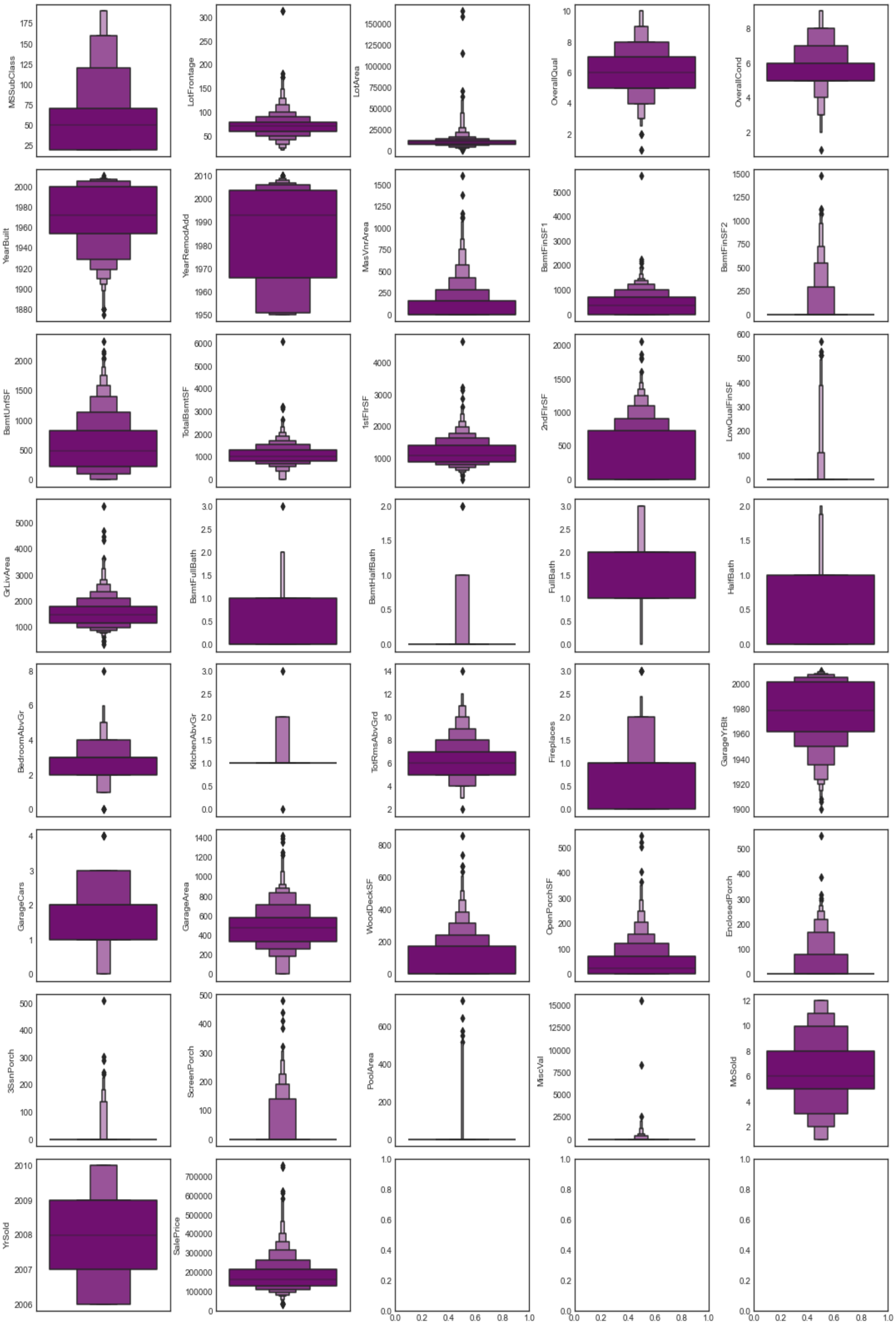
Heatmap:



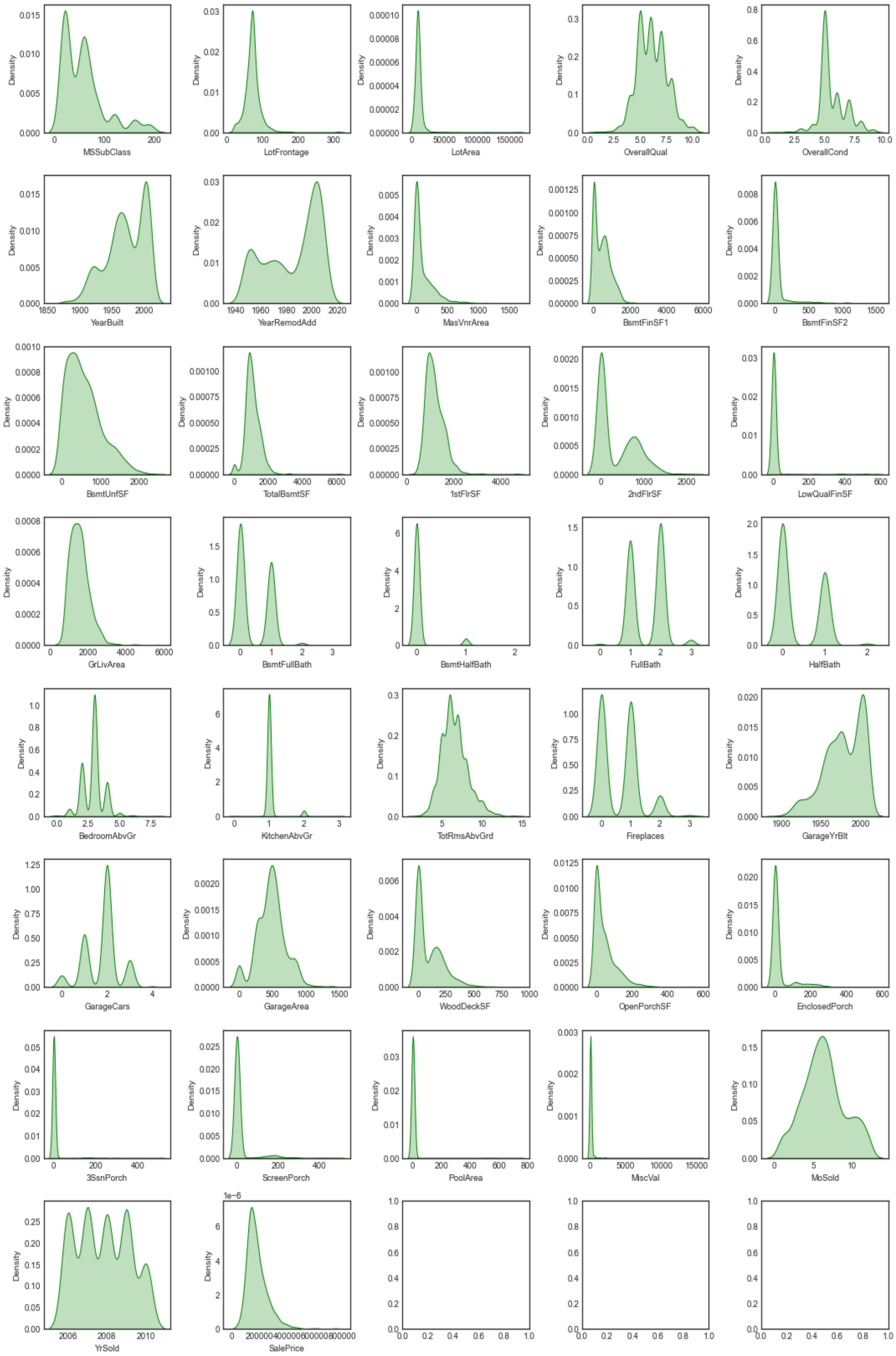
Correlation:



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:



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 pre-processing processes to the training dataset. I transformed the values into a data frame and combined it with the original testing data frame, which only included 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 research presented above aids in the understanding of the real estate industry. How the price of the homes is changing. With the study, we can see 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 kind of building. With the foregoing study, we can 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

During this assignment, I ran into a difficulty with a lack of data. Many columns have the identical data in more than 80% of the rows, causing our model's performance to suffer. Another difficulty is that this data collection has a big number of missing values, thus we must fill those missing values correctly. With some feature engineering and rigorous hyperparameter adjustment, we can still enhance the accuracy of our model.

Thank You

For Your Attention