Project Housing Report

Submitted by:

Vivek Bhadania

Flip Robo Technologies

INTRODUCTION

- Business Problem Framing

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

Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

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.

Flip Robo Technologies

- 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. We are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

1. Which variables are important to predict the price 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.

FlipRobo Technologies

- 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 (HPI) 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.

Flip Robo Technologies

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

We are building a model in Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. So, this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house. This will help to determine the price of houses with the available independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For specific mathematical reasons this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values.

Regression analysis is also a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

Flipkart Internal

- Data Sources and their formats

Data sets provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of Training dataset 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 and 81 columns.

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 and 80 columns.

- Data Pre-processing Done

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a 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 pre-processing 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:

- Loading the training dataset as a dataframe
- Used pandas to set display in ensuring we do not see any truncated information
- Checked the number of rows and columns present in our training dataset

Flip Robo Technologies

- Checked for missing data and the number of rows with null values
- Verified the percentage of missing data in each column and decided to discard the ones that have more than 50% of null values
- Dropped all the unwanted columns and duplicate data present in our dataframe
- Separated categorical column names and numeric column names in separate list variables for ease in visualization
- Checked the unique values information in each column to get an idea of the categorical data
- Performed imputation to fill missing data using mean on numeric data and mode for categorical data columns
- Used Pandas Profiling during the visualization phase along with pie plot, count plot, scatter plot and Bivariate analysis
- With the help of ordinal encoding technique converted all object datatype columns to numeric datatype
- Thoroughly checked for outliers and skewness information
- Removing Skewness using yeo-johnson power transform method only on selected numerical columns.
- 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
- We separated feature and label data and scaled the features data to avoiding any kind of biasness
- Checked VIF Factor for Multicollinearity and removed the column with VIF factor greater than 10
- Checked for the best random state to be used on our Regression Machine Learning model pertaining to
- Checked the feature importance details using Random Forest algorithm
- Finally created a regression model function along with evaluation metrics to pass through various model formats

Flip Robo Technologies

- Data Inputs- Logic- Output Relationships

When we loaded the training dataset, we had to go through various data pre-processing steps to understand what was given to us and what we were expected to predict for the project. When it comes to logical part the domain expertise of understanding how real estate works and how we are supposed to cater to the customers, came in handy to train the model with the modified input data. In Data Science community there is a saying “Garbage In Garbage Out” therefore we had to be very cautious and spent majority of our project building time in understanding each and every aspect of the data how they were related to each other as well as our target label.

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

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

The assumption part for me was relying strictly on the data provided to me and taking into consideration that the separate training and testing datasets were obtained from real people surveyed for their preferences and how reasonable a price for a house with various features inclining to them were.

FlipRobo Technologies

- Hardware and Software Requirements and Tools Used

Hardware Used:

- Processor: Core i5 -10300H CPU @ 2.50GHz
- RAM: 8 GB
- Operating System: 64-bit
- ROM/SSD: 1 TB SSD
- Graphics: NVIDIA GeForce GTX 1650 Ti

Software Used:

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

Libraries/Packages Used:

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

Flip Robo Technologies

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

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

In order to get good performance as well as accuracy and to check my model from over-fitting and under-fitting I have made use of the K-Fold cross validation and then hyper parameter tuned the final model.

Once I was able to get my desired final model, I ensured to save that model before I loaded the testing data and started performing the data pre-processing as the training dataset and obtaining the predicted sale price values out of the Regression Machine Learning Model.

Flip Robo Technologies

- Testing of Identified Approaches (Algorithms)

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

  - Linear Regression Model
  - Ridge Regularization Regression Model
  - Lasso Regularization Regression Model
  - Elastic Net Regularization Regression Model
  - Support Vector Regression Model
  - Decision Tree Regression Model
  - Random Forest Regression Model
  - K Nearest Neighbours Regression Model
  - Stochastic Gradient Descent Regression Model
  - Gradient Boosting Regression Model
  - Ada Boost Regression Model
  - Extra Trees Regression Model

- Run and evaluate selected models

  After choosing the random state amongst 1-1000 number we got the best random state. Then I defined a function for getting the regression model trained and evaluated. The code for the models are listed below.

  Random State:

```
lr=LinearRegression()
```

For Test size 0.25

```
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=.25, random_state =i)
    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 accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.8775578272556023  on Random_state  654
```

EduBoost Technologies

For Test size 0.2

```
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=.20, random_state =i)
    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 accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.8845504029138069  on Random_state  942
```

We tested using 2 test sizes and we see that out of the random state value ranging from 1 to 1000 the best random state number found is 942 with test size 0.20 and we will use this in our Machine Learning models. I used a total of 12 Regression Models as follows below.

Regression Model Function:

```
# creating a function to run all the regressors

def regressor(model, x, y):
    x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, random_state=942)

    # Training the model
    model.fit(x_train, y_train)

    # Predicting y_test
    pred = model.predict(x_test)

    # Root Mean Square Error (RMSE)
    rmse = mean_squared_error(y_test, pred, squared=False)
    print("Root Mean Square Error 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 is:", cv_score)

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

I have built a regression function that splits the training and testing features and labels, then trains the model, predicts the label, calculates the RMSE score, generates the R2 score, calculates the

Cross Validation score and finally finds the difference between the R2 score and Cross Validation score.

Linear Regression Model

```
model=LinearRegression()
regressor(model, x, y)
```

```
Root Mean Square Error is: 24669.973270297083
R2 Score is: 88.4550402913807
Cross Validation Score is: 76.05393506616835
R2 Score - Cross Validation Score is 12.401105225212348
```

Ridge Regularization Regression Model

```
model=Ridge(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 24669.96635036763
R2 Score is: 88.45504676810401
Cross Validation Score is: 76.05397209064995
R2 Score - Cross Validation Score is 12.401074677454062
```

Lasso Regularization Regression Model

```
model=Lasso(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 24669.97122546663
R2 Score is: 88.45504220524448
Cross Validation Score is: 76.05394024646566
R2 Score - Cross Validation Score is 12.401101958778824
```

Elastic Net Regularization Regression Model

```
model=ElasticNet(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 24666.772658555114
R2 Score is: 88.4580357171353
Cross Validation Score is: 76.0711123437302
R2 Score - Cross Validation Score is 12.386923373405097
```

Support Vector Regression Models

```
model=SVR(kernel='rbf')
regressor(model, x, y)
```

```
Root Mean Square Error is: 72610.53064237421
R2 Score is: -0.01250035464317456
Cross Validation Score is: -6.181094755060825
R2 Score - Cross Validation Score is 6.168594400417651
```

```
model=SVR(kernel='poly')
regressor(model, x, y)
```

```
Root Mean Square Error is: 72624.85470209415
R2 Score is: -0.051963675457589176
Cross Validation Score is: -6.218264793631496
R2 Score - Cross Validation Score is 6.166301118173907
```

```
model=SVR(kernel='linear')
regressor(model, x, y)
```

```
Root Mean Square Error is: 66567.38617347975
R2 Score is: 15.94218971752256
Cross Validation Score is: 8.427824053143972
R2 Score - Cross Validation Score is 7.514365664378587
```

Decision Tree Regression Model

```
model=DecisionTreeRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 35075.41969094207
R2 Score is: 76.66215150359568
Cross Validation Score is: 72.37759971960335
R2 Score - Cross Validation Score is 4.2845517839923275
```

Random Forest Regression Model

```
model=RandomForestRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 22254.926292626802
R2 Score is: 90.60477069415772
Cross Validation Score is: 84.52528600007395
R2 Score - Cross Validation Score is 6.079484694083774
```

K Nearest Neighbours Regression Model

```
model=KNeighborsRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 32349.277026925753
R2 Score is: 80.14891623834271
Cross Validation Score is: 69.82284962070386
R2 Score - Cross Validation Score is 10.326066617638844
```

Stochastic Gradient Descent Regression Model

```
model=SGDRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 25019.905960627424
R2 Score is: 88.12519710241718
Cross Validation Score is: 75.54496036425553
R2 Score - Cross Validation Score is 12.580236738161645
```

Gradient Boosting Regression Model

```
model=GradientBoostingRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 21648.421257063088
R2 Score is: 91.10988202440862
Cross Validation Score is: 85.27201886965973
R2 Score - Cross Validation Score is 5.837863154748888
```

Ada Boost Regression Model

```
model=AdaBoostRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 31995.566329555568
R2 Score is: 80.58065098841882
Cross Validation Score is: 78.71938620111871
R2 Score - Cross Validation Score is 1.8612647873001151
```

Extra Trees Regression Model

```
model=ExtraTreesRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 23591.92755632515
R2 Score is: 89.44199372858674
Cross Validation Score is: 81.77265050173519
R2 Score - Cross Validation Score is 7.669343226851552
```

- 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 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 and their mean are taken for the final CV score.

2. R2 Score:

   It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for R2 Score is 1. The closer the value of R2 Score to 1, the better is the model fitted.

Flip Robo Technologies

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 the actual value. 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.

Flip Robo Technologies

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

```
# creating parameters list to pass into GridSearchCV

parameters = {'loss' : ['squared_error', 'absolute_error', 'huber', 'quantile'],
              'criterion' : ['friedman_mse', 'squared_error', 'mse', 'mae'],
              'max_features' : ['sqrt', 'log2'],
              'learning_rate' : [0.1, 0.25, 0.5],
              'random_state' : [None, 942],
              'n_estimators': [100, 200, 300]}
```

```
GCV = GridSearchCV(GradientBoostingRegressor(), parameters, cv=5)
```

```
GCV.fit(x_train,y_train)
```

GridSearchCV

```
final_model = GradientBoostingRegressor(criterion = 'mse', learning_rate = 0.1,
```

```
final_fit = final_model.fit(x_train,y_train)   # final fit
```

```
final_pred = final_model.predict(x_test)   # predicting with best parameters
```

```
# final accuracy score
best_r2=r2_score(y_test,final_pred,multioutput='variance_weighted')*100   # chec
print("R2 score for the Best Model is:", best_r2)
```

```
R2 score for the Best Model is: 91.18707842238815
```

```
# final Cross Validation score
final_cv_score = (cross_val_score(final_model, x, y, cv=5).mean())*100
print("Cross Validation Score is:", final_cv_score)
```

```
Cross Validation Score is: 84.43091783261536
```

```
# final RMSE
final_rmse = mean_squared_error(y_test, final_pred, squared=False)
print("Root Mean Square Error is:", final_rmse)
```

```
Root Mean Square Error is: 21554.225439656977
```

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

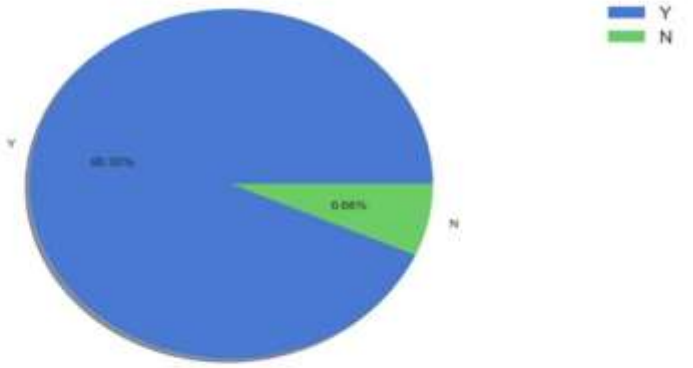
I used pandas profiling to get the over viewed visualization on the pre-processed data. pandas-profiling is an open-source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates interactive reports in web format that can be presented to any 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. pandas-profiling does work of visualizing and understanding the distribution of each variable. It generates a report with all the information easily available.

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

Pie Plots:

```
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])
```

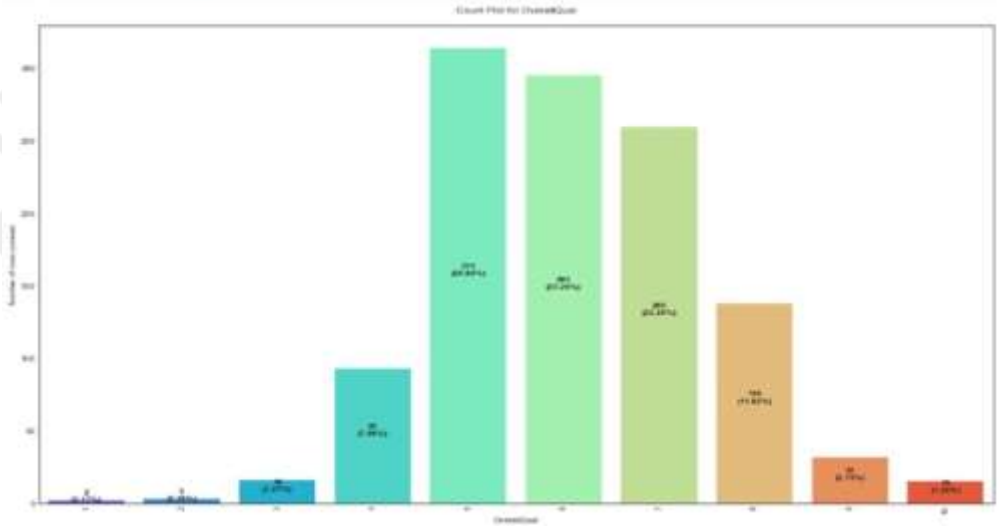
```
Single digit category column name: CentralAir
```




```
Single digit category column name: Street
```

count plots:

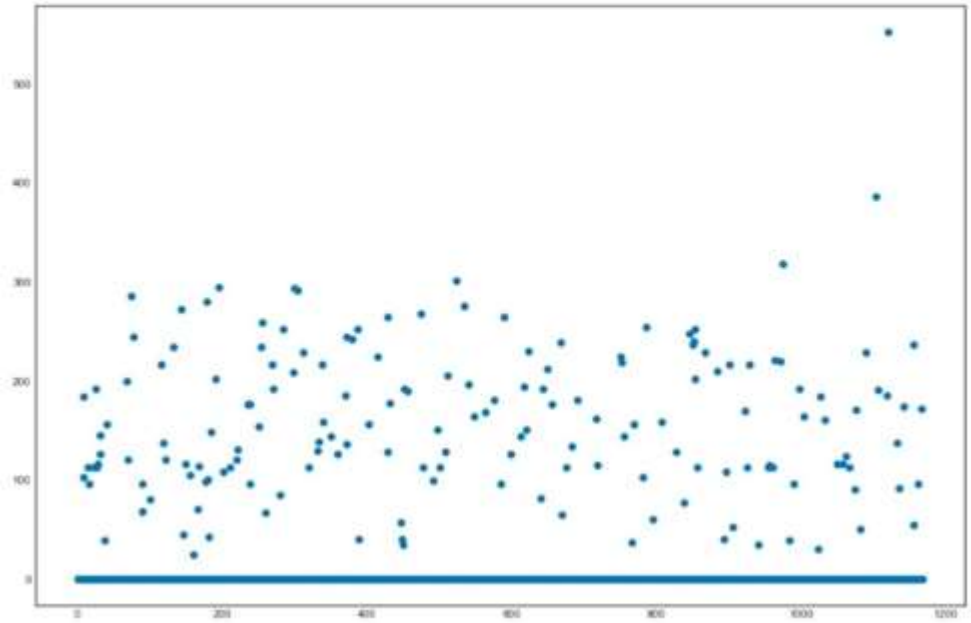
```
for col in train_df[double]:
    plt.figure(figsize=(20,12))
    col_name = col
    values = train_df[col_name].value_counts()
    index = 0
    ax = sns.countplot(train_df[col_name], palette="rainbow")

    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()
```



scatter plots:

```
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()
```

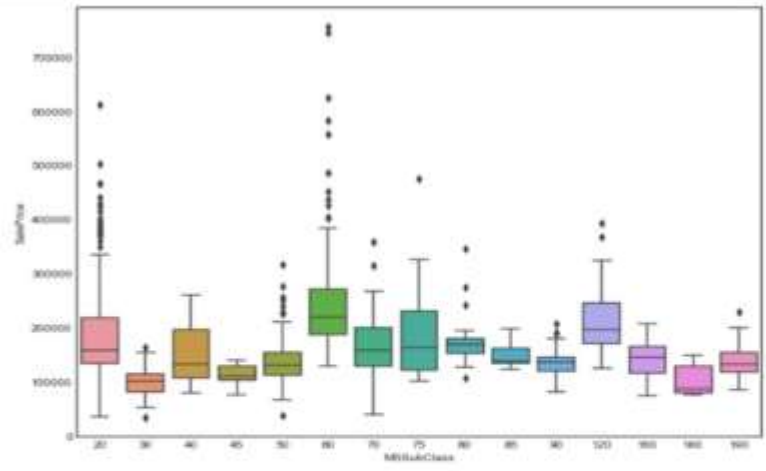
```
Scatter plot for EnclosedPorch column with respect to the rows covered ->
```



```
Scatter plot for LotFrontage column with respect to the rows covered ->
```

Bivariate Analysis (Independent variables vs Target Variable):

```
plt.figure(figsize=(10,8))
sns.boxplot(x='MSSubClass',y='SalePrice',data=train_df.sort_values('SalePrice',ascending=False))
plt.show()
```



MSSubClass vs SalePrice

```
# Checking for the sale price on the basis of road access to the property

plt.figure(figsize=[5,5])
sns.barplot(x='Street', y='SalePrice', data = train_df.sort_values('SalePrice', ascending=False))
plt.show()
```

- 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 multi collinearity issues. Detected outliers/skewness with the help of boxplot and distribution plot. I got to know the count of a particular category for each feature by using count plot and most importantly with predicted target value distribution as well as scatter plot helped me to select the best model. Also checked the feature importance with respect to the target variable.

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

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

Flip Robo Technologies

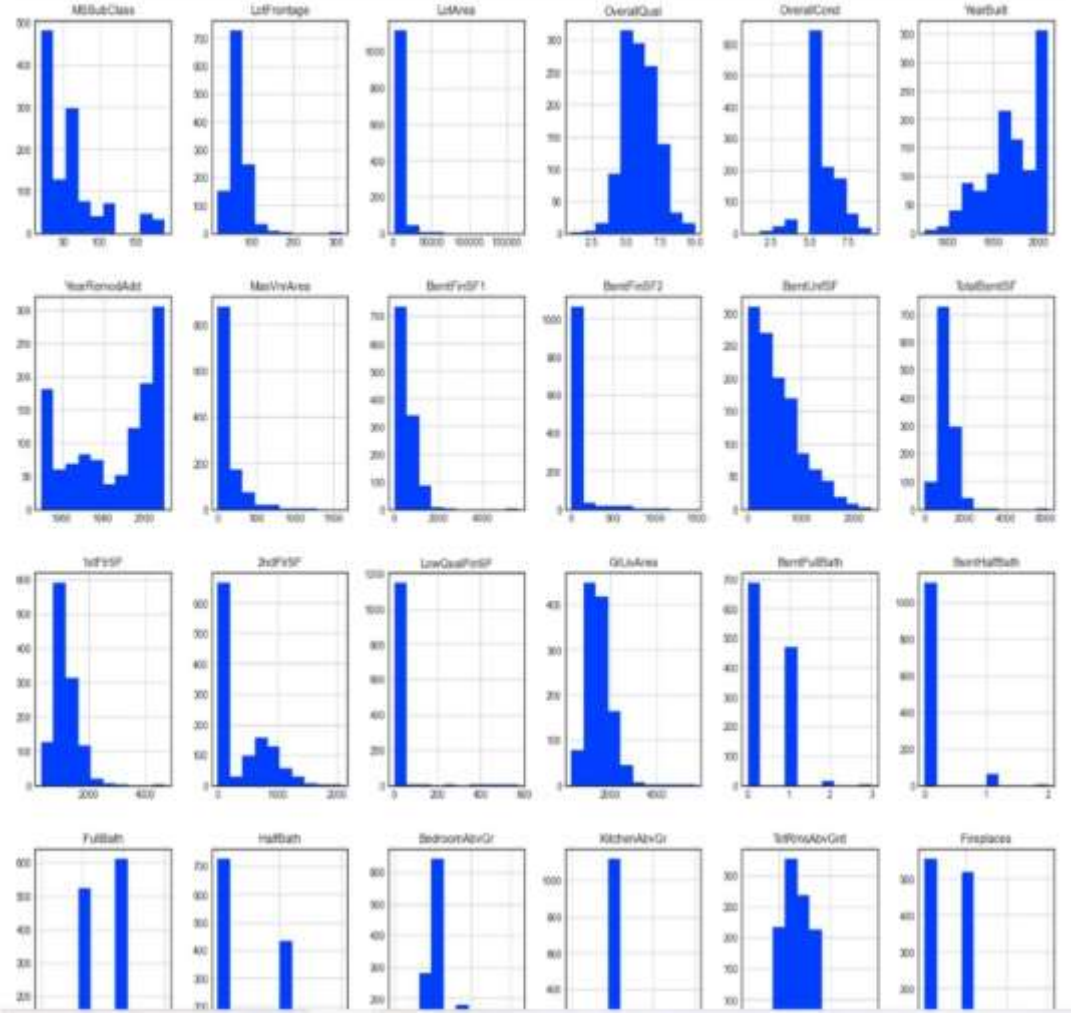
CONCLUSION

- Key Findings and Conclusions of the Study

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

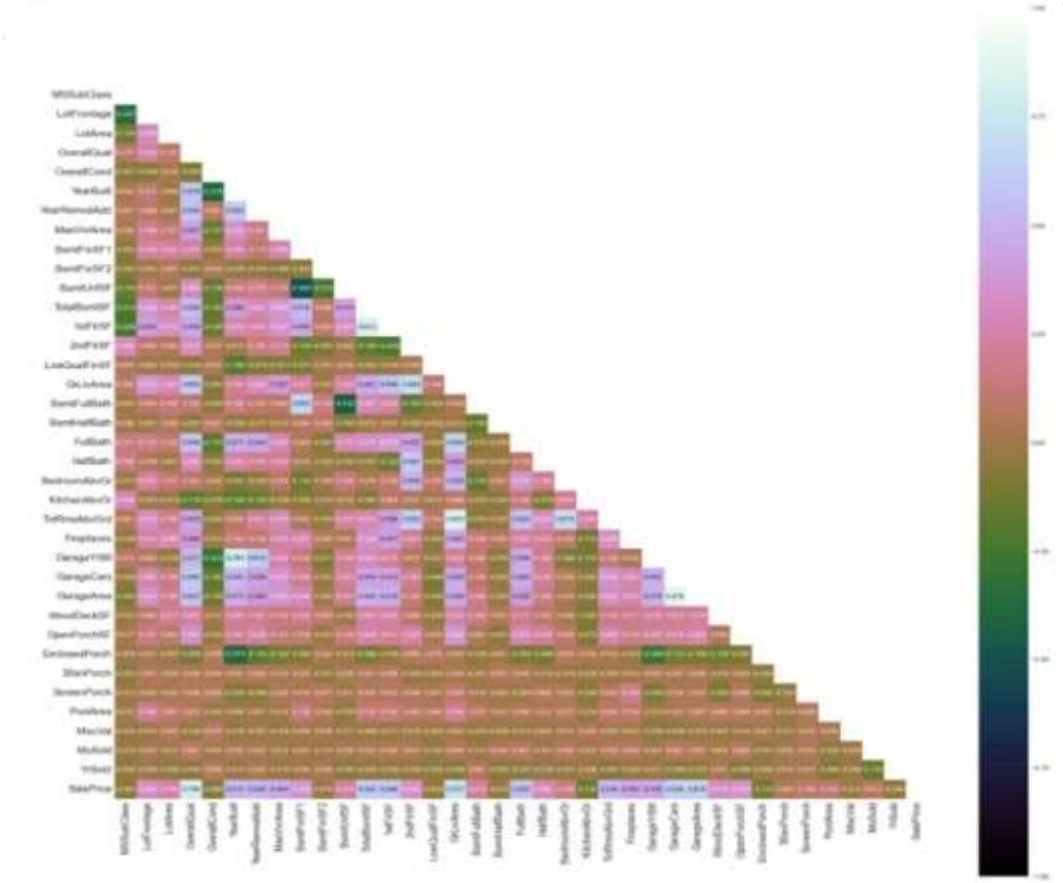
  Histogram:

```
plt.style.use('seaborn-bright')

train_df.hist(figsize=(20,30))
plt.show()
```



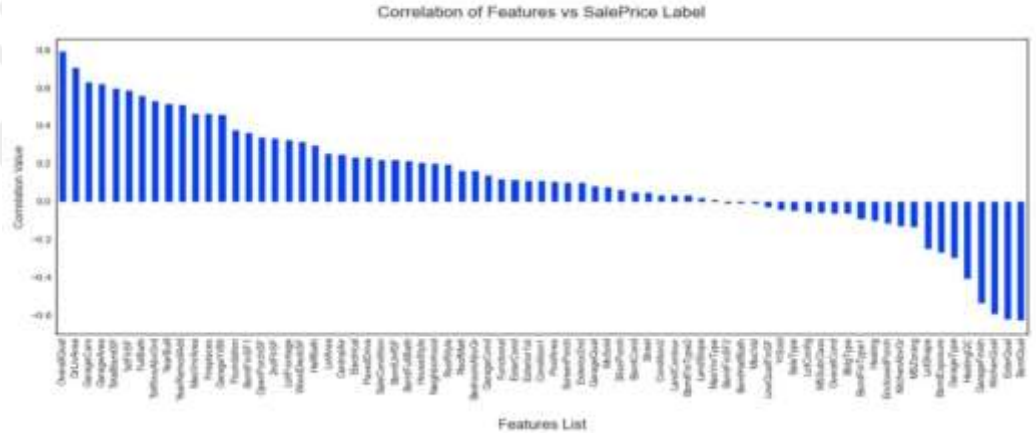
Correlation Heatmap:

```
upper_triangle = np.triu(train_df.corr())
plt.figure(figsize=(25,25))
sns.heatmap(train_df.corr(), vmin=-1, vmax=1, annot=True, square=True, fmt='0.1f',
            annot_kws={'size':10}, cmap="cubehelix", mask=upper_triangle)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```



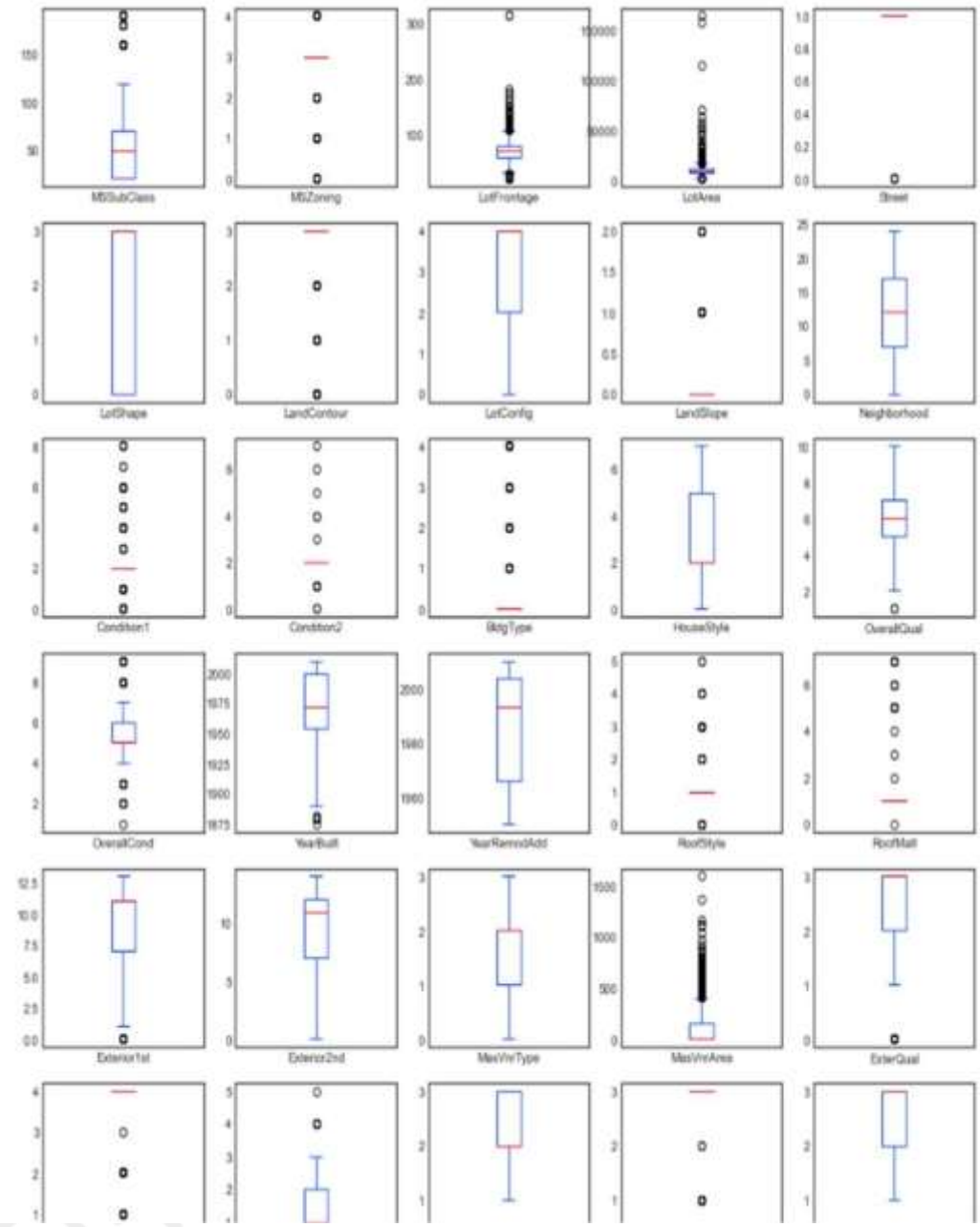
Correlation Bar plot:

```
df_corr = df.corr()
plt.figure(figsize=(16,6))
df_corr['SalePrice'].sort_values(ascending=False).drop('SalePrice').plot.bar()
plt.title("Correlation of Features vs SalePrice Label\n", fontsize=16)
plt.xlabel("\nFeatures List", fontsize=14)
plt.ylabel("Correlation Value", fontsize=12)
plt.show()
```



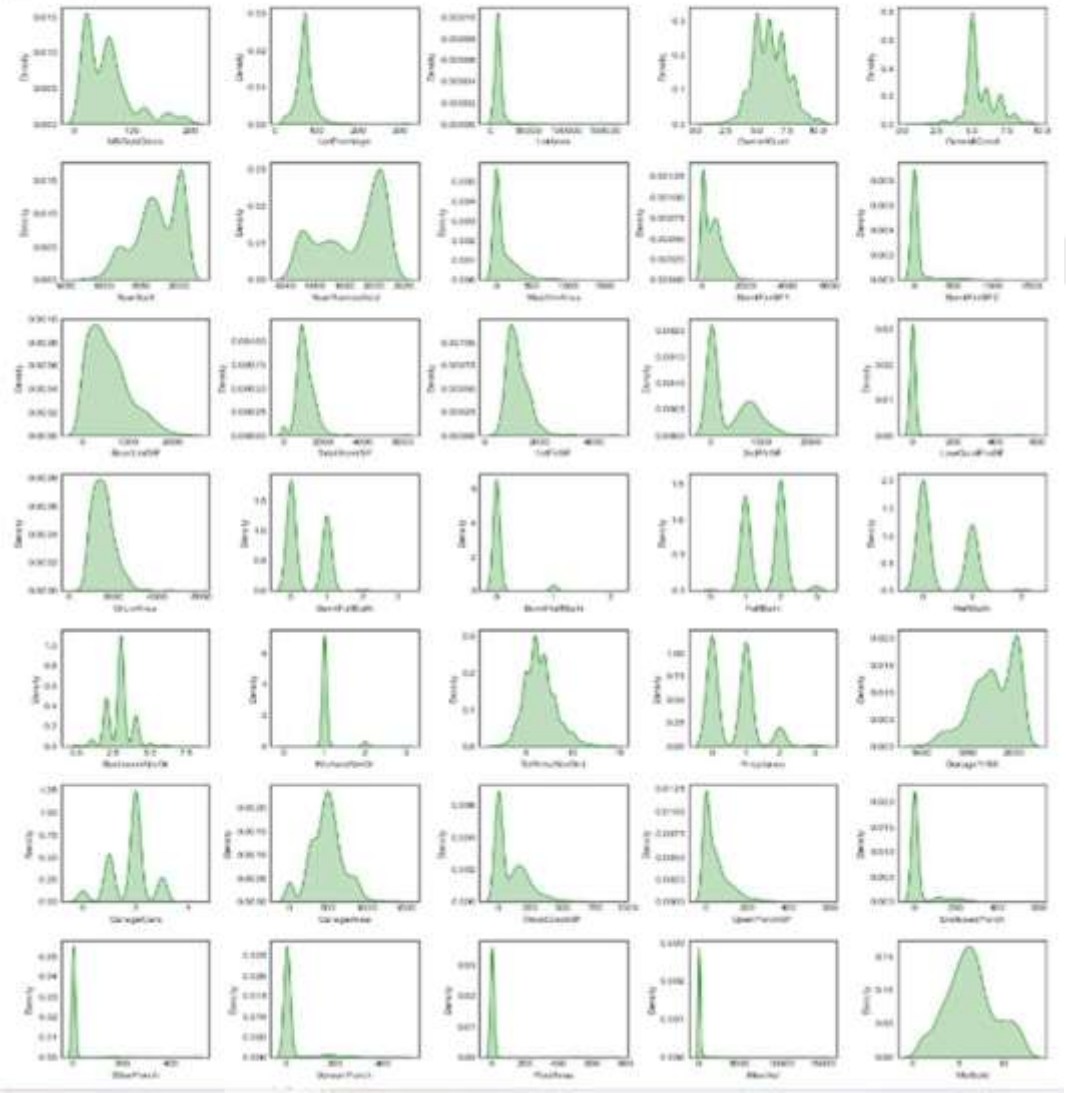
Box Plot (Outlier Checking):

```
# Checking for the outliers
train_df.plot(kind='box', subplots=True, layout=(16,5), sharex=False, legend=True, figsize=(15,45))
plt.show()
```

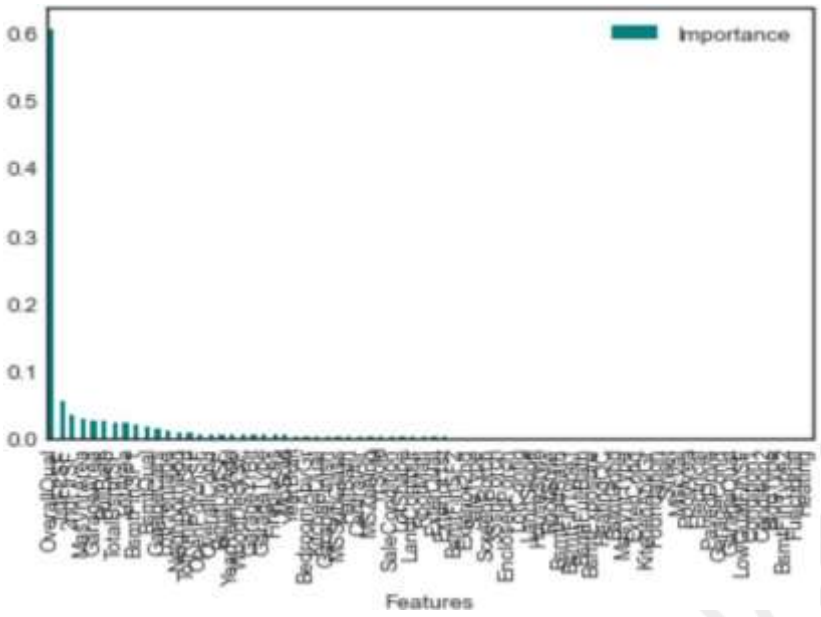


Distribution Plot:

```
fig, ax = plt.subplots(ncols=5, nrows=8, figsize=(15,20))
index = 0
ax = ax.flatten()
for col, value in df[numeric_datatype].items():
    sns.distplot(value, ax=ax[index], hist=False, color="g", kde_kws={"shade": True})
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.4, h_pad=1.0)
plt.show()
```

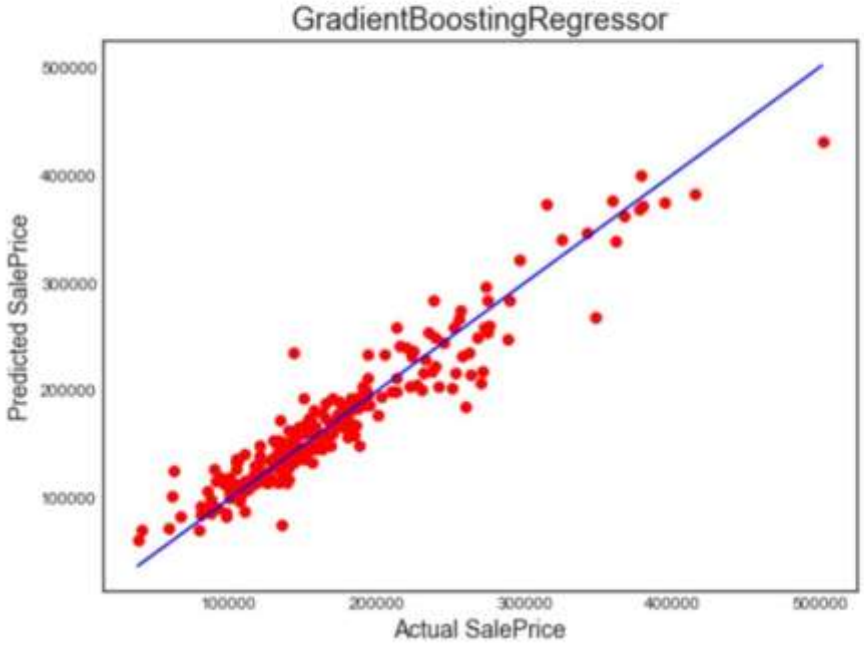


Feature Importance with respect to Target variable:



Plotting Best Fit line:

```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=final_pred, color='r')
plt1 = max(max(final_pred), max(y_test))
plt2 = min(min(final_pred), min(y_test))
plt.plot([plt1, plt2], [plt1, plt2], 'b-')
plt.xlabel('Actual SalePrice', fontsize=14)
plt.ylabel('Predicted SalePrice', fontsize=14)
plt.title('GradientBoostingRegressor',fontsize=18)
plt.show()
```



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 the need we can project the price of the property.

- ## Limitations of this work and Scope for Future Work

During this project I have faced a problem of low amount of data. Many columns are with same entries in a lot of rows which lead to reduction in our model performance. One more issue is there are large number of missing values presents in this data set, so we have to fill those missing values in correct manner. We can still improve our model accuracy with some feature engineering and by doing some extensive hyperparameter tuning on it.

FliRobo Technologies

Thank You

Flip Robo Technologies