Multiple models to predict House Sale Price

Submitted by:

Vikash Kumar Singh

ACKNOWLEDGMENT

In this project, I took help from our DataTrained's video of Shankargouda Tegginmani sir and some websites which are- https://www.youtube.com https://www.kaggle.com , https://stackoverflow.com and https://scikit-learn.org in completion of this project.

INTRODUCTION

• Business Problem Framing

- Houses are one of the necessary need 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
- 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 on 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 to enter the market. You are required to build a regression model using regularization in order to predict the actual value of the prospective properties and decide whether to invest in them or not.

• Conceptual Background of the Domain Problem

- In preparation of any Machine Learning Model, we have to have conceptual knowledge of cleansing, scaling, outlier and many more method to perform the EDA analysis.

- The knowledge of Data Visualization is also required to observe the skewness, outlier, correlation and multicollinearity of the given dataset and you are also required to have knowledge to reduce the following observation.

- Use standard techniques (PCA, Scaling, encoding, underfitting/overfitting, AUC-ROC curve, cross-validation, grid search, ensemble techniques) wherever applicable.

- Review of Literature

The literature study gives an overview of this project and the EDA methods, the feature engineering methods that have been used in this study. As well as about evaluation metrics to measure the performance of the algorithms.

- Motivation for the Problem Undertaken

Presently, I am doing an internship with FlipRobo company and they have given us this project to build a Machine Learning Model which can predict the sale price of houses. This project gives me an opportunity to recall the teaching of Datatrained's mentor and to build multiple regression model with better accuracy. This will help me in upgrading my skills and knowledge.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

In the given dataset, target is in continuous form so, the model will be regression model (supervised learning) in which, we will design a predictive model to analyse the relation between features and target variables (sale price of house).

- Data Sources and their formats

FlipRobo have given us the two different datasets of .csv format 1.) train.csv and 2.) test.csv, that contains numerical as well as categorical variable. They also have null values.

- Data Pre-processing Done

In pre-processing, we have done the following steps:

1.) Imported the given dataset in jupyter notebook.
2.) Cleansing the dataset
3.) Concating train and test dataset
4.) Transforming the dataset
5.) Detecting the outlier, skewness, multicollinearity of the dataset

6.) Visualization and analysis the given dataset
7.) Scaling the dataset

- Data Inputs- Logic- Output Relationships

  In the given dataset, we have lots of features that describe the sale price of any house, in this we have given the data like- Road access, Alley access, shape, Types of utilities, No. of bathroom, Electricity, type of roof, building material, Type and Style of dwellings, Types of foundation, Information about air condition, basement, heat condition, parking, water availability of water etc.

  In real estate, we all know how can these data affects the sale price of any house. They are like directly proportional to the sale price of houses.

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

  I don't have any pre-assumption for this, and my output is totally based on the given dataset.

- Hardware and Software Requirements and Tools Used
- Hardware tools:
  1. Windows laptop
  2. i5 processor
  3. 4GB ram
  4. 250 GB SSD card
  - Software tools:
    1. windows 10
    2. Anaconda Navigator
    3. Jupyter Notebook
    4. Python
  - Libraries and packages:
    1. Pandas
    2. NumPy

3. SciPy
4. Seaborn
5. Mat plot
6. Sklearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  From the domain knowledge and my own understanding, I have followed these steps:

  1. Importing the given test and train dataset
  2. Checking null values in the dataset
  3. Imputation of given dataset
  4. Transformation of given dataset
  5. Standardization of given dataset
  6. Splitting the train and test dataset

- Testing of Identified Approaches (Algorithms)
  1. Linear Regression
  2. Random forest regressor
  3. Decision Tree Regressor
  4. Support Vector Regressor
  5. KNeighbors Regressor
  6. XGB Regressor
  7. Lasso Regressor
  8. Ridge Regressor
  9. AdaBoost Regressor
  10. Gradient Boosting Regressor

- Run and evaluate selected models

1. Linear regression

```
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

regressor=LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
regression_results(y_test,y_pred)
model_accuracy(regressor)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('linear_regression', fontsize = 18)
plt.show()
```

```
Explained_variance:  -2.4189249642889793e+21
R2:  -2.4469292540634155e+21
Adjusted_r2:  9.372983244378508e+21
MAE:  374613574149591.06
MSE:  1.2262072237374824e+31
RMSE:  3501724180653699.5
Accuracy: -42270776225921214483687014 4.00 %
Standard Deviation: 502518369103604307254575104.00 %
```

After all the pre-processing, now it's time to prepare the model based on dependent variable (y) and independent variable(X). In this model, its accuracy is very bad and look like pre-processing doesn't fit it. We will try to build different model and let's observe their accuracy.

2. Random Forest Regressor

```
rand_regressor = RandomForestRegressor()
rand_regressor.fit(X_train, y_train)
y_pred_rf = rand_regressor.predict(X_test)
regression_results(y_test,y_pred_rf)
model_accuracy(rand_regressor)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_rf, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('rand_regressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  -0.1197
R2:  -0.1203
Adjusted_r2:  5.2912
MAE:  62815.4352
MSE:  6131810415.9647
RMSE:  78305.8773
Accuracy: 87.20 %
Standard Deviation: 1.60 %
```

In this model, we have better accuracy (87.20) and Standard deviation is very low (1.60), that's good for our model.

3. Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor

decision_tree=DecisionTreeRegressor(criterion='mse',splitter='random',random_state=10)
decision_tree.fit(X_train, y_train)
y_pred_dt = decision_tree.predict(X_test)
regression_results(y_test,y_pred_dt)
model_accuracy(decision_tree)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_dt, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('decision_tree', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.7416
R2:  0.7407
Adjusted_r2:  1.9931
MAE:  25039.4934
MSE:  1299165688.9471
RMSE:  36043.9411
Accuracy: 72.30 %
Standard Deviation: 2.02 %
```

this model has accuracy of 72.30 % and has minimal standard deviation of 2.02 %.

4. Support Vector Regressor

```
from sklearn.svm import SVR
```
Run this cell
```
svr=SVR()
svr.fit(X_train, y_train)
y_pred_svr = svr.predict(X_test)

regression_results(y_test,y_pred_svr)
model_accuracy(svr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_svr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('svr', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.0005
R2:  -0.0069
Adjusted_r2:  4.857
MAE:  53636.4639
MSE:  5045912211.7906
RMSE:  71034.5846
Accuracy: -5.46 %
Standard Deviation: 3.67 %
```

SVR model is fitted for this dataset, we will build some different model.

5. KNeighbors Regressor

```
from sklearn.neighbors import KNeighborsRegressor

knr=KNeighborsRegressor(n_neighbors=2)
knr.fit(X_train, y_train)
y_pred_knr = knr.predict(X_test)

regression_results(y_test,y_pred_knr)
model_accuracy(knr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_knr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('knr', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.6809
R2:  0.6803
Adjusted_r2:  2.2244
MAE:  25275.8348
MSE:  1601835350.2059
RMSE:  40022.9353
Accuracy: 72.23 %
Standard Deviation: 4.46 %
```

This model has an accuracy of 72.23% and standard deviation of 4.46%.

6. XGB Regressor

```
from xgboost import XGBRegressor
xgbr=XGBRegressor(random_state=10)
xgbr.fit(X_train, y_train)
y_pred_xgbr = xgbr.predict(X_test)

regression_results(y_test,y_pred_xgbr)
model_accuracy(xgbr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_xgbr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('xgbr', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.8822
R2:  0.8814
Adjusted_r2:  1.4543
MAE:  16495.305
MSE:  594278980.2954
RMSE:  24377.8379
Accuracy: 86.67 %
Standard Deviation: 3.31 %
```

This also has very good accuracy of 86.67% and standard deviation of only 3.31%.

7. Lasso Regressor

```
ls = Lasso()
ls=Lasso(random_state=10)
ls.fit(X_train, y_train)
y_pred_ls = ls.predict(X_test)

regression_results(y_test,y_pred_ls)
model_accuracy(ls)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_ls, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('ls', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.8311
R2:  0.831
Adjusted_r2:  1.6474
MAE:  19866.2758
MSE:  846942878.5822
RMSE:  29102.283
Accuracy: 84.13 %
Standard Deviation: 3.79 %
```

This model has accuracy of 84.13% and standard deviation of 3.79%.

8. Ridge Regressor

```
rd = Ridge()
rd=Ridge(random_state=10)
rd.fit(X_train, y_train)
y_pred_rd = rd.predict(X_test)

regression_results(y_test,y_pred_rd)
model_accuracy(rd)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_rd, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('Ridge', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.8392
R2:  0.839
Adjusted_r2:  1.6168
MAE:  19776.8922
MSE:  806878578.6377
RMSE:  28405.6082
Accuracy: 84.92 %
Standard Deviation: 3.38 %
```

Accuracy is 84.79% and Standard deviation is 3.38%.

9. AdaBoost Regressor

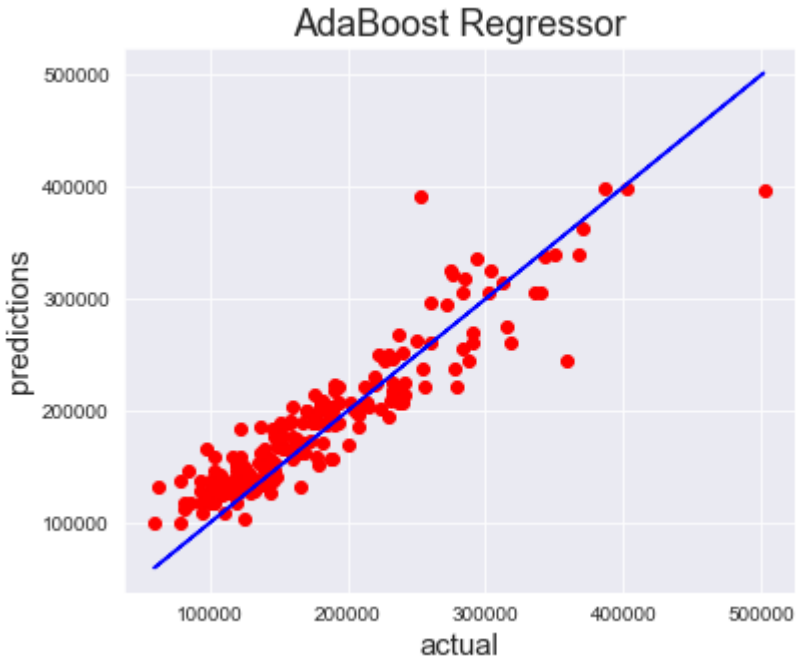
```
abr = AdaBoostRegressor()
abr=AdaBoostRegressor(random_state=10)
abr.fit(X_train, y_train)
y_pred_abr = abr.predict(X_test)

regression_results(y_test,y_pred_abr)
model_accuracy(abr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_abr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('AdaBoost Regressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.8604
R2:  0.842
Adjusted_r2:  1.6052
MAE:  21273.1036
MSE:  791738137.7745
RMSE:  28137.8417
Accuracy: 83.51 %
Standard Deviation: 1.68 %
```

Its accuracy is 83.51% and standard deviation is 1.68.



10. Gradient Boosting Regressor

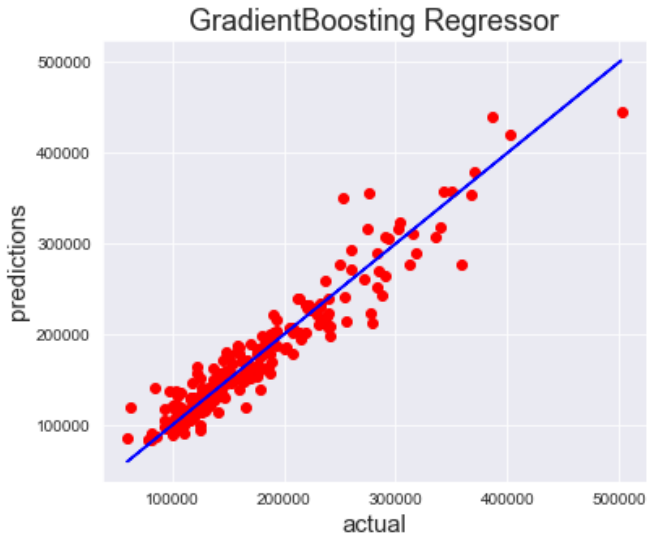
```
gbr = GradientBoostingRegressor()
gbr=GradientBoostingRegressor(random_state=10)
gbr.fit(X_train, y_train)
y_pred_gbr = gbr.predict(X_test)

regression_results(y_test,y_pred_gbr)
model_accuracy(gbr)

plt.figure(figsize = (6,5))
plt.scatter(x = y_test, y=y_pred_gbr, color = 'r')
plt.plot(y_test,y_test, color = 'b')
plt.xlabel('actual', fontsize = 15)
plt.ylabel('predictions', fontsize = 15)
plt.title('GradientBoosting Regressor', fontsize = 18)
plt.show()
```

```
Explained_variance:  0.9069
R2:  0.9068
Adjusted_r2:  1.3571
MAE:  15529.6409
MSE:  467188781.3364
RMSE:  21614.5502
Accuracy: 88.75 %
Standard Deviation: 1.24 %
```

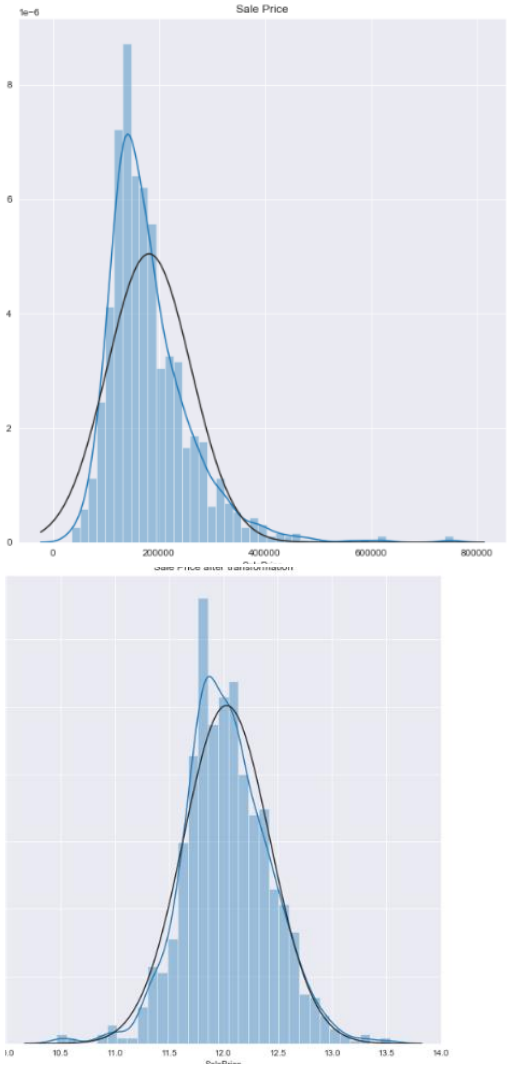
Its has the highest accuracy of 88.75% and standard deviation of 1.24% , which is lowest among all model. We will tune this model using GridSearchCV and observe the accuracy differences between the before and after tuning the model.



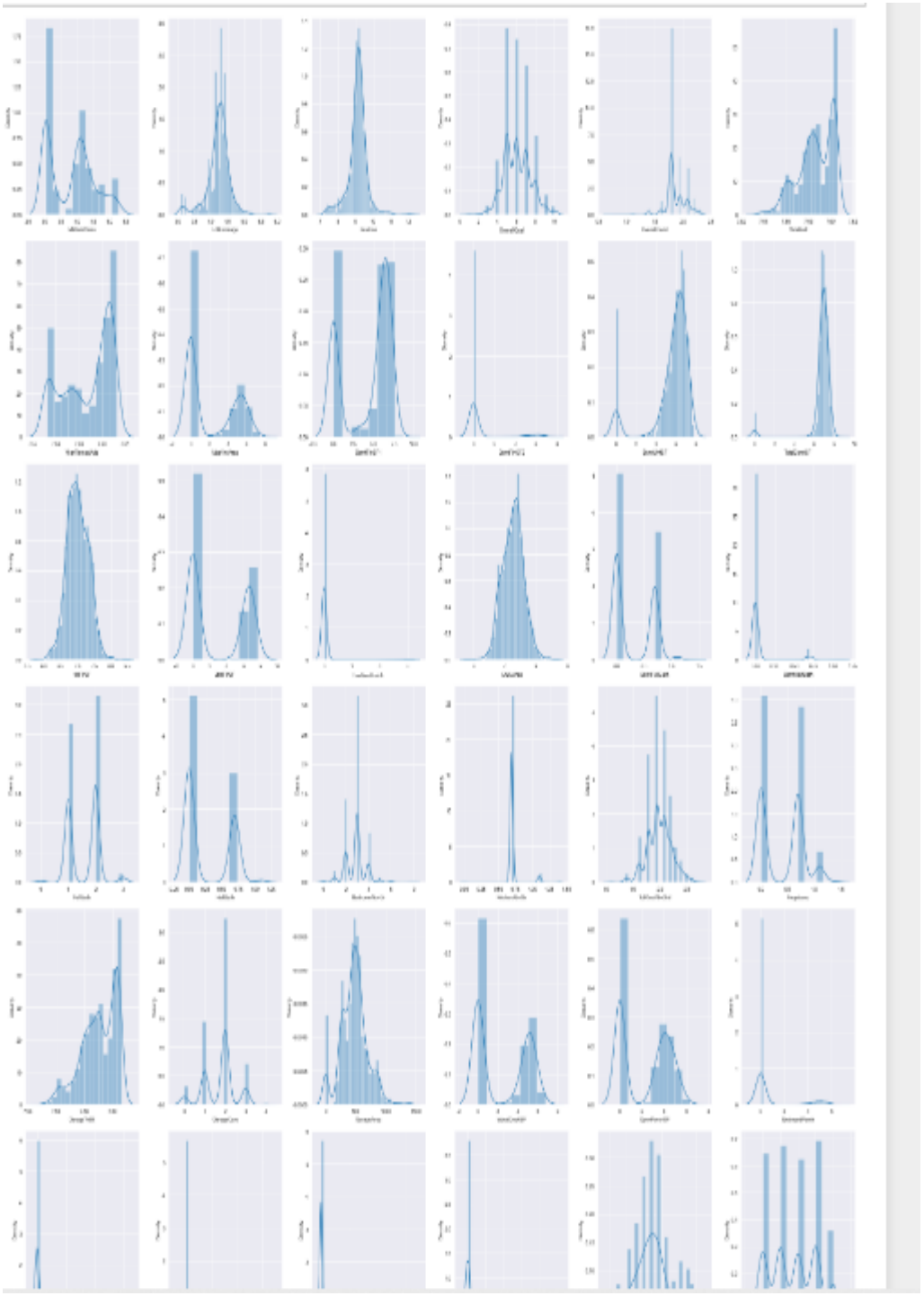
In this graph, we can Clearly see that the actual and predicted sale price is very much linear to each other. It shows our model is performing well for this dataset.

- Key Metrics for success in solving problem under consideration
    a. Standard Deviation
    b. Explained Variance
    c. Mean absolute error
    d. Mean squared error
    e. R2 score
    f. Adjusted r2 score
    g. Accuracy

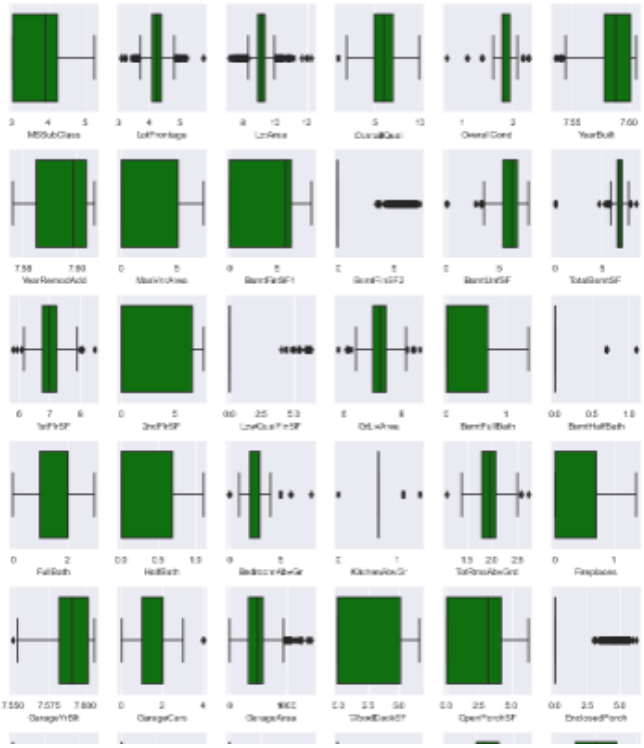
- Visualizations
    A. Distplot of Target variable



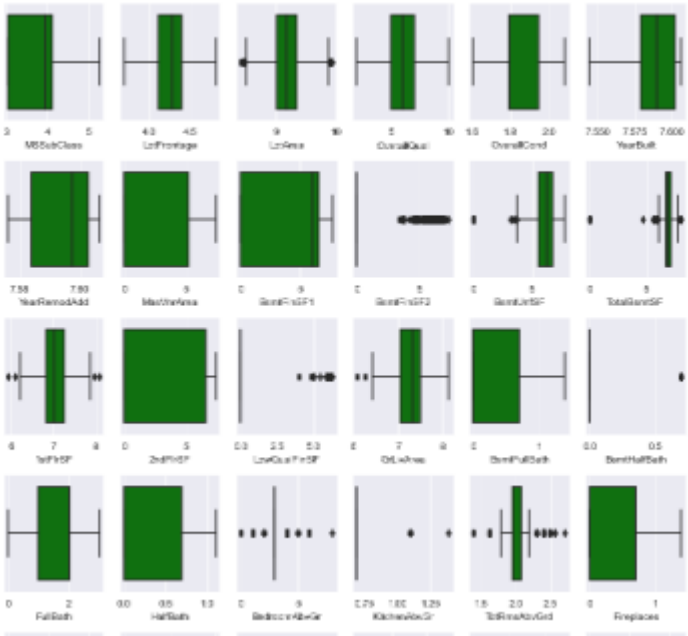
B. Distplot of all numerical data



C. Boxplot of numerical data to analyse outlier in the dataset

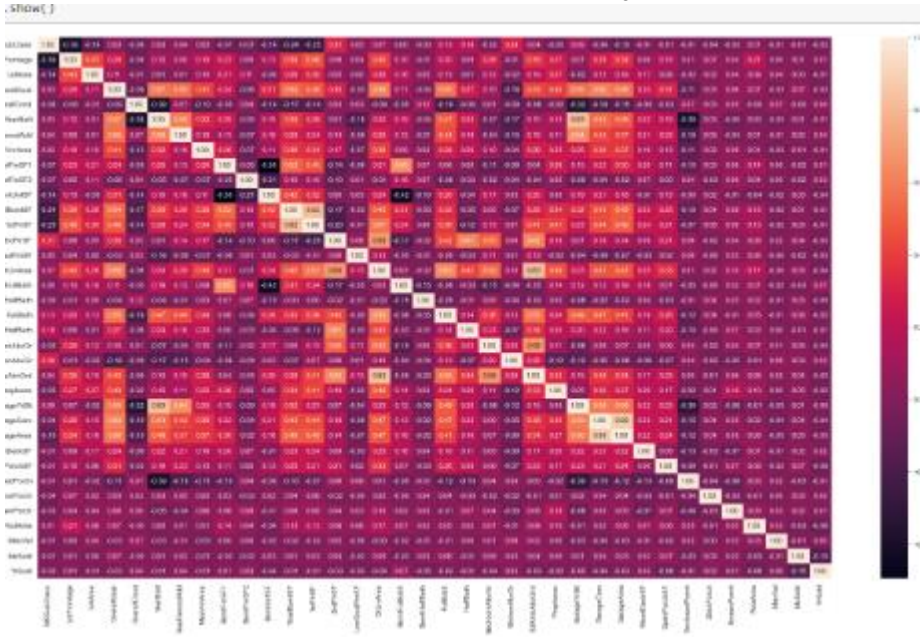


After the removal of outlier



We can clearly see in the 1st rows that the all outlier are removed

D. Heatmap to check the multicollinearity



- Interpretation of the Results
  After pre-processing, we analysed that, a lot of data has missing values, null values and data having zero values. These values have been imputed from different method; mean, median, mode and some of by encoder technique.

  After Visualization, we have observed the skewness, outlier and multicollinearity of the data, and have tried to reduce it by log transformation technique, interquartile range and vif respectively.
  After doing 10 different modelling, we analysed that the gradient boosting regressor model is best among all.

CONCLUSION

- Key Findings and Conclusions of the Study

The main aim of this project was to determine the prediction of sale price. In this project, we have discovered many algorithms and application of machine learning techniques with the objective to predict the sale price of houses. Price can be predicted through many factors like the Electricity, water availability and many related factors with the house. We have first cleaning and exploring of the input data. We have performed Linear regression, Random Forest regressor, Decision Tree regressor, SVR, KNeighbours regressor, Lasso, Ridge, Adaboost regressor, XGB regressor and Gradient boosting regressor as we understand that parameterization of the can drive the significant result in the performance.

- Learning Outcomes of the Study in respect of Data Science

i. From different visualization, we have learned about the skewness, outlier and multicollinearity of the dataset and it helped in making better model.

ii. Data cleansing- we learned, how to fill or drop the null values data for the dataset by performing multiple statistical operation.

iii. Machine learning algorithm- we have performed 10 multiple regression model to predict the sale price of house, out of all Gradient boosting regressor have performed best among the others. We have also the Hyper Para meter tuning to the improve of model accuracy.

iv. Challenges- We got two different dataset train and test, so we have to first concat both the dataset and then have performed cleansing, imputing, encoding and scaling. Both datasets have also contained numerical and categorical data with null values. To overcome these challenges, we had performed the above techniques once again.

- Limitations of this work and Scope for Future Work

  Limitations of this project were that we haven't cover all regression algorithms in our Data Science course, instead of it, they have focused on the basic algorithm. In Hyper Para Meter tuning, parameter for different model is very difficult to choose.