FLIP ROBO

HOUSING: PRICE PREDICTION

Submitted by:

Vedant Singh Shankar

ACKNOWLEDGMENT

I would like to thank my SME, Muskan Vats, for providing me the opportunity and guidance to work on this project. The sample data is provided to me from Flip Robo Technologies, which was shared to me for this project. Also, the following are the website links as a reference that helped me in the project. Also, the following are the website links as a reference that helped me in the project.

Website Link:

- https://www.mygreatlearning.com/blog/xgboost-algorithm/
- https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b
- https://scikit-learn.org/stable/modules/permutation_importance.html
- https://maxhalford.github.io/blog/target-encoding/
- https://scikit-learn.org/stable/modules/linear_model.html#elastic-net
- https://towardsdatascience.com/which-evaluation-metric-should-you-use-in-machine-learning-regression-problems-20cdaef258e

Contents

1.INTRODUCTION

1.1 Background

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.

1.2 Problem Statement

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. And to find out the variables that are important for price prediction and how these variables describe the price of the house.

1.3 Summary of the project

- In this project we have applied 3 different kinds of supervised Machine Learning Algorithms to build the predictive model for the use case.
- The data set contained 81 features in total to which we filtered out to only important features that may contribute to the target variable using robust feature selection method.
- The data contained missing values to which we handled according to the domain knowledge. We also analysed some potential outliers using appropriate visualizations to detect and discard them.
- We have done various statistical analysis and used techniques to find the data correlation and relationship between output and input variables.
- We have tuned the hyper-parameters of the models to address the overfitting issue and analysed the final test results.

1.4 Motivation for the Problem Undertaken

This project helps to understand what aspect of the house is more significant for people to invest while buying houses of different kinds. Also, from the management perspective, knowing the price beforehand may help to strategize plans more effectively.

2. Analytical Problem Framing

2.1 Modelling of the Problem

In this project, we have used 3 kinds of different Machine Learning Models that is based on supervised learning algorithm. Which works by learning the historical data with associated labels or outcomes, and predicts the possible outcomes, in our case to predict the house selling price given a set of features describing the various aspect of the house and its locality. The high-level overview of each algorithm that is used to model the data is as follows:

Elastic Net Regression

- **Elastic net** is a popular type of regularized linear regression that combines two popular penalties, specifically the **L1** and **L2** penalty functions.

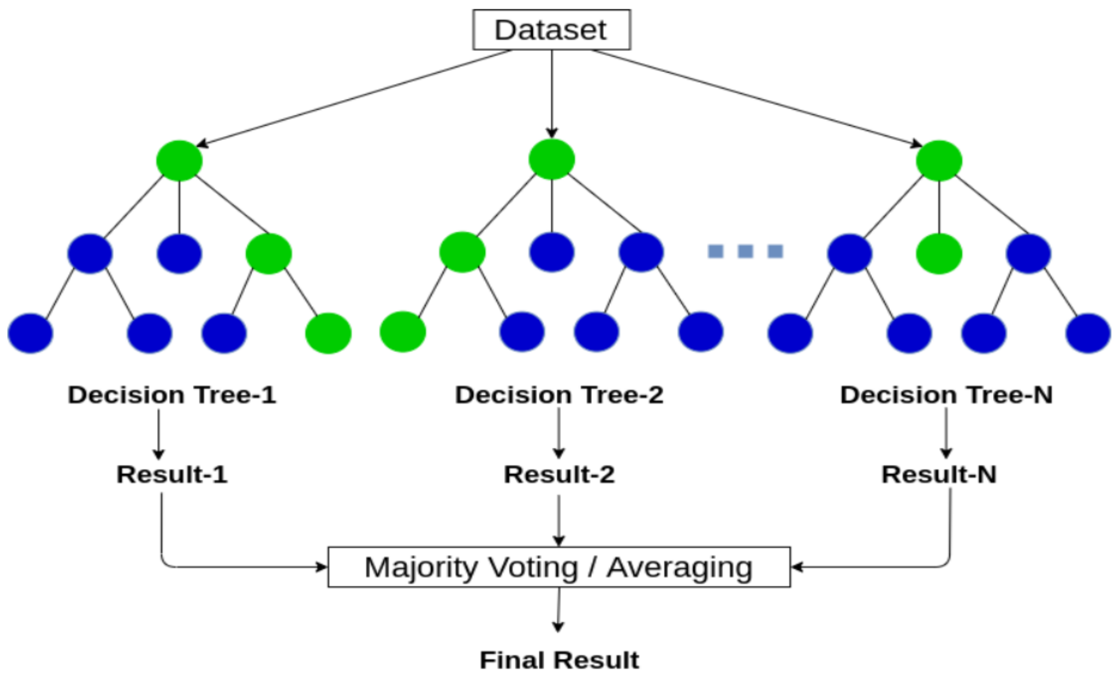
- It is an extension to linear regression that involves adding penalties to the loss function during training that controls and reduces the complexity of the model to reduce ***overfitting*** problem. These extensions are referred to as regularized linear regression or penalized linear regression.

- The benefit is that elastic net allows a balance of both penalties, which can result in better performance than a model with either one or the other penalty on some problems. The hyperparameter is provided called "***lambda***" that controls the weighting of the sum of both penalties to the loss function.

Random Forest Regression

Random forest, like its name implies, consists of a large number of individual **decision trees** which operates in Bootstrap aggregation, also called **bagging,** that is one of the **ensemble** techniques in Machine Learning. It can be used for both classification and regression problems. Here, each individual tree in the random forest gives out a class prediction and the class with the most votes becomes our model's prediction.

The individual decision trees typically exhibit high **variance** and tend to overfit. The injected randomness in forests yield decision trees with somewhat **decoupled** prediction errors. By taking an average of those predictions, some errors can cancel out. Hence, random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.



Overview of Random Forest Algorithm

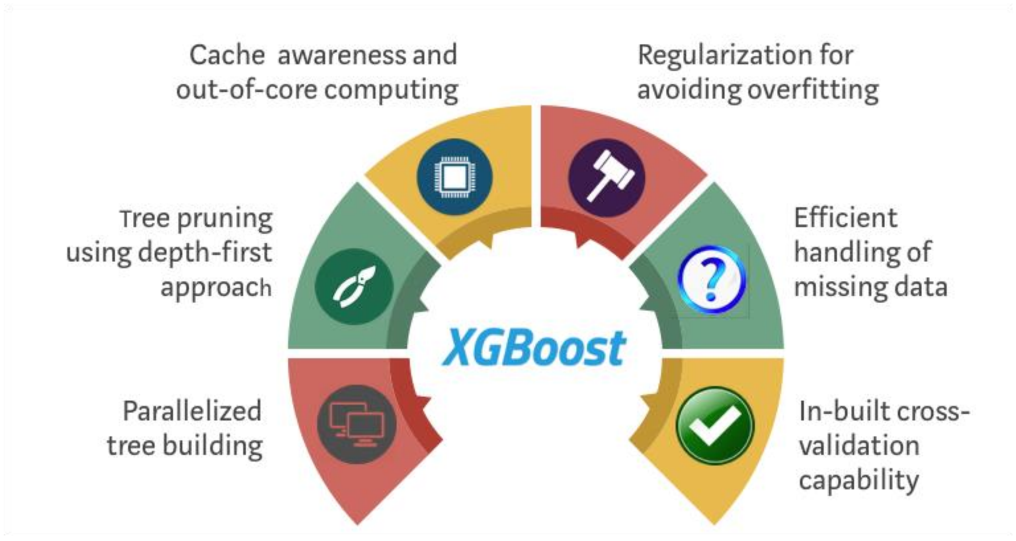
The above figure represents the high-level overview of working of Random Forest Algorithm, which is basically parallel decision trees working individually to yield out a prediction which is further averaged (in case of regression problem).

XGBoost (Extreme Gradient Boosting)

Boosting is an ensemble learning technique to build a strong ***learner*** from several weak ***learners*** in series. Boosting algorithms play a crucial role in dealing with ***bias-variance trade-off***. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias & variance) and is considered to be more effective.

XGBoost is used with a tree as the base learner, in which a decision tree is composed of the series of binary logics and the final predictions happens at the leaf of a node. XGBoost is itself an ensemble method. The trees are constructed iteratively until a stopping criterion is met.

XGBoost library is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

How XGBoost optimizes standard Gradient Boosting algorithm

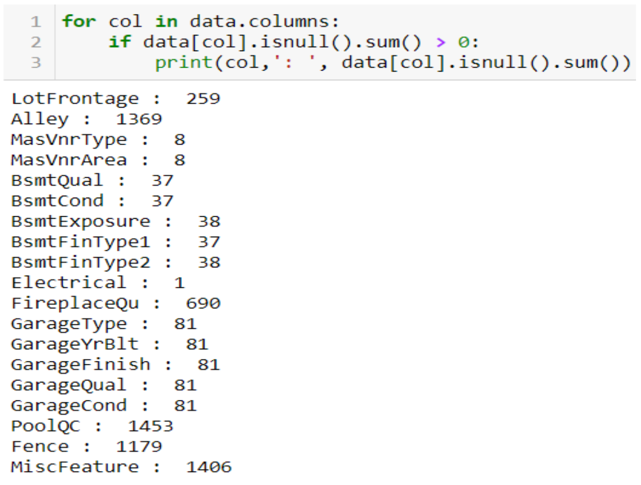
2.2 Data Sources and Description

- The sample data has been provided to us from the Flip Robo Technologies which that has various sales record of houses containing the input features, to describe a house and its locality and output feature, that contains at what price a particular house was sold.

- Data contains 1460 rows and 81 variables or features occupying 452 KB on disk and also contains missing values.

2.3 Exploratory Data analysis and Data Pre-processing

Both numerical as well as categorical data contained missing values, however, according to the data description provided, most of these missing values were actually a separate category, for example, in feature pertaining to ‘Garage’ of a house like garage type, year in which garage was built, garage condition, etc had all same number of missing values that was actually a category of ‘No Garage’. Therefore, it made sense to replace all the missing values of these categorical columns with the separate category. Below is the snapshot of the code that filtered out the missing values with their count for the respective columns.



```
for col in data.columns:
    if data[col].isnull().sum() > 0:
        print(col,': ', data[col].isnull().sum())
```

```
LotFrontage :  259
Alley :  1369
MasVnrType :  8
MasVnrArea :  8
BsmtQual :  37
BsmtCond :  37
BsmtExposure :  38
BsmtFinType1 :  37
BsmtFinType2 :  38
Electrical :  1
FireplaceQu :  690
GarageType :  81
GarageYrBlt :  81
GarageFinish :  81
GarageQual :  81
GarageCond :  81
PoolQC :  1453
Fence :  1179
MiscFeature :  1406
```

From the above code in the snapshot, we are able to filter out the columns containing null values and their count.

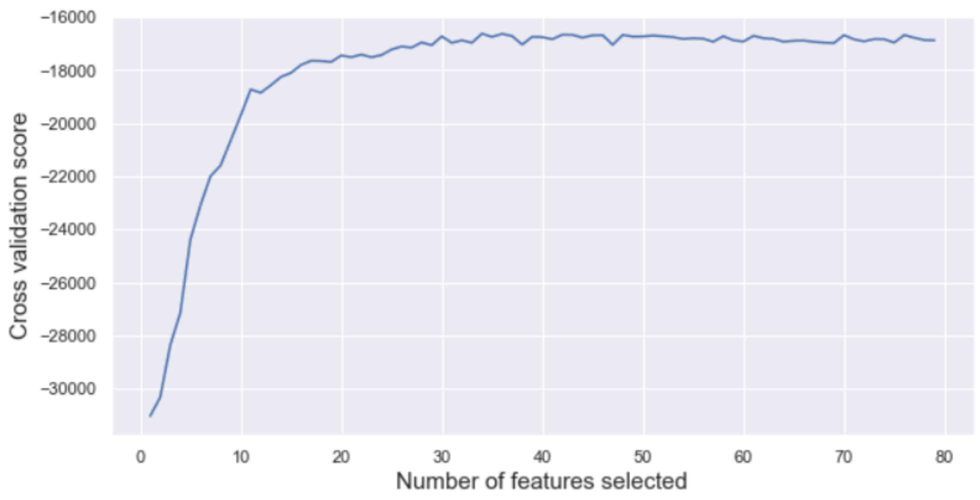
And the numerical variables that had missing features were replaced by **groupby (**with respect to other categorical variable**)** median as their distribution were found out to be skewed. Also, for other missing categorical variables, the null values were replaced by **groupby** mode.

After replacing the null values with the proper assumption and analysis, we then used effective feature selection method known as Recursive Feature Elimination (**RFE**)

Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) as the name suggests recursively removes features to filter out the best combination of the feature for predicting the target variable. It initially builds a model using the remaining attributes and calculates model accuracy by building a model on the entire set of predictors and computing an importance score for each predictor. The least important predictors are then removed, the model is re-built, and importance scores are re-computed again in a recursive manner until the algorithm exhaustively utilises all the feature space.

In our implementation we used **RFE** with 10-fold cross-validation to compute the score of each combination of the feature subset and **ExtraTreesRegressor** as our estimator and scoring function as negative Mean Absolute Error (**MAE**), which is almost similar to random forest regressor with slight variation and comparatively faster in computation speed.

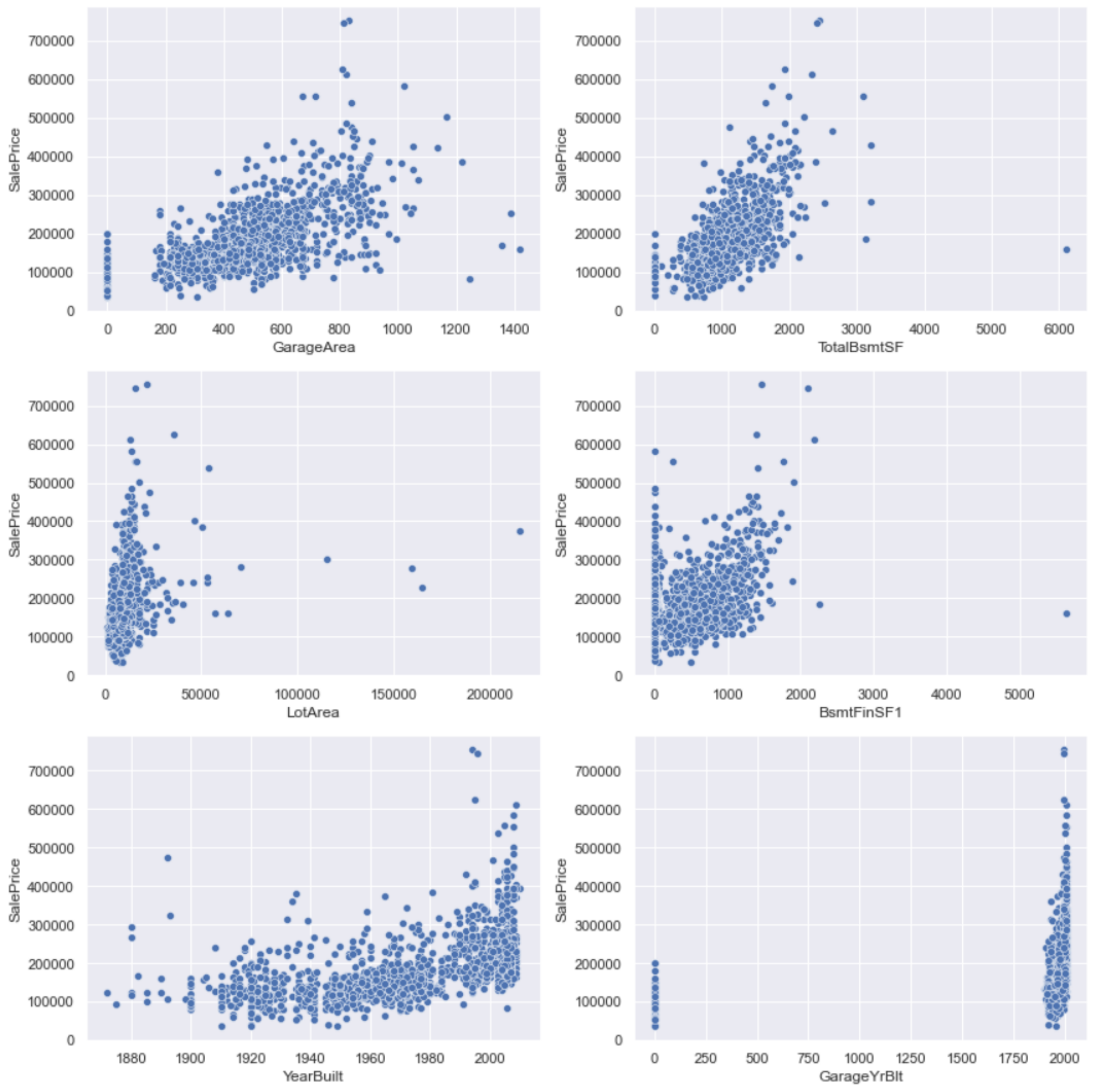


The figure above, depicts the cross-validation score (evaluated on the bases of negative mean absolute error) and the number of combination features selected for each iteration. We can clearly see that the cross-validation score gets plateaued roughly after 30 number of features. And in our case, the algorithm selected 34 features corresponding to the highest score out of 81 feature space. Hence, applying this technique helped us to filter out the feature space to only those relevant features to predict our target variable (i.e., Saleprice) by reducing the dimension of our feature space.

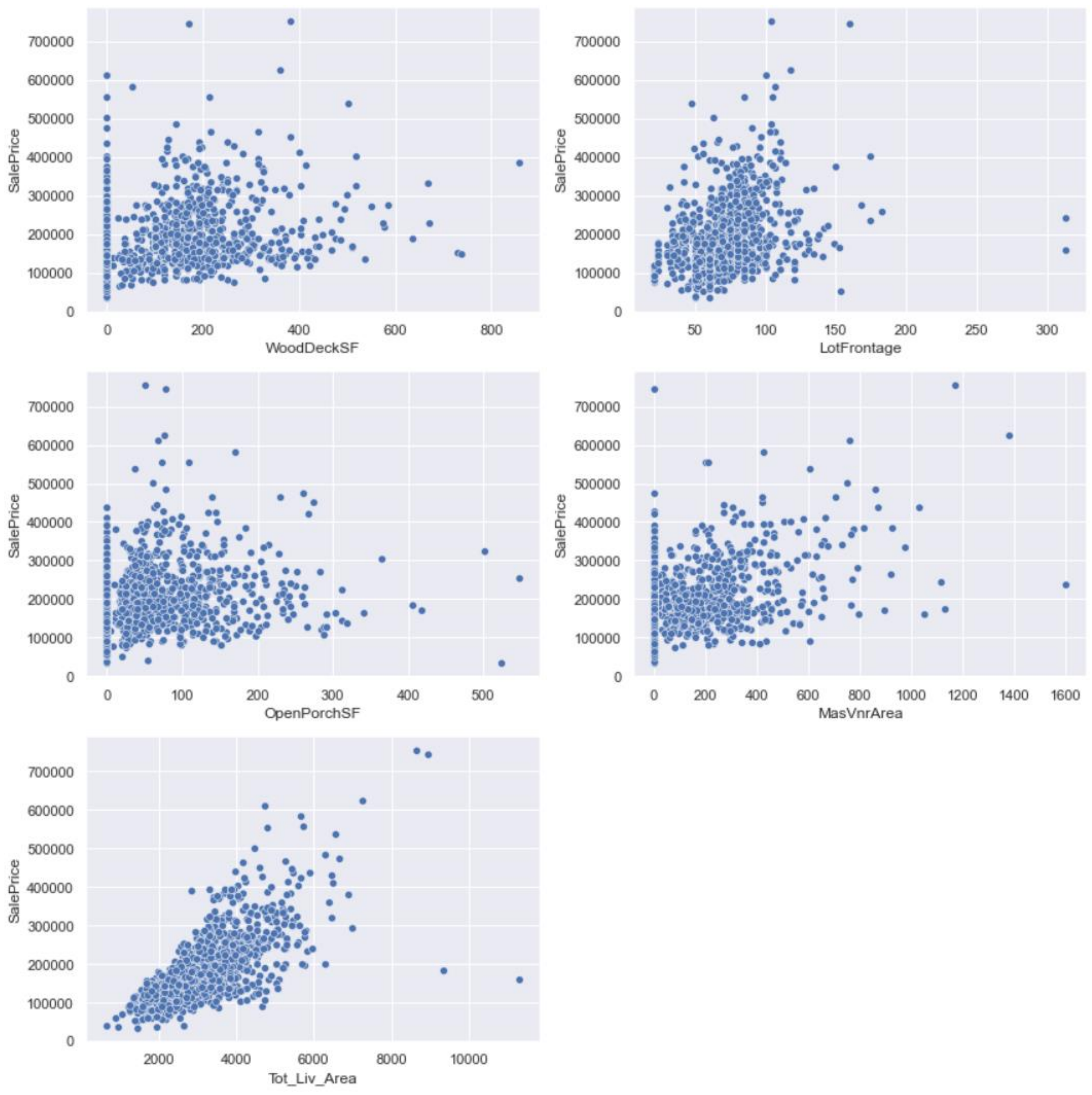
To emphasis on the total living area of a house, which is usually the important factor for house selling price, a new feature called **'Tot_Liv_Area'** is created by adding the existing feature like **'GrLivArea'** (Above ground living area square feet), **'1stFlrSF'** (First Floor square feet) and **'2ndFlrSF'** (Second floor square feet) and these existing features were removed to avoid collinearity with the created feature. Also, feature called **'YearRemodAdd'** (i.e., Remodel date of the house same as construction date if no remodelling or additions are done) was removed and only captured the binary information of whether the house has been remodelled or not in the feature called '**Remodel'**.

Outlier Analysis and Detection

Although tree-based algorithms like Random Forest and XGBoost is robust to outliers but linear algorithm like Elastic Net is highly sensitive to outliers. Here, with the help of scatter plot we have visualized the potential outliers in the 2-D plane with respect to the target variable '**SalePrice'** for all the numeric variables.



Scatterplots of numerical features

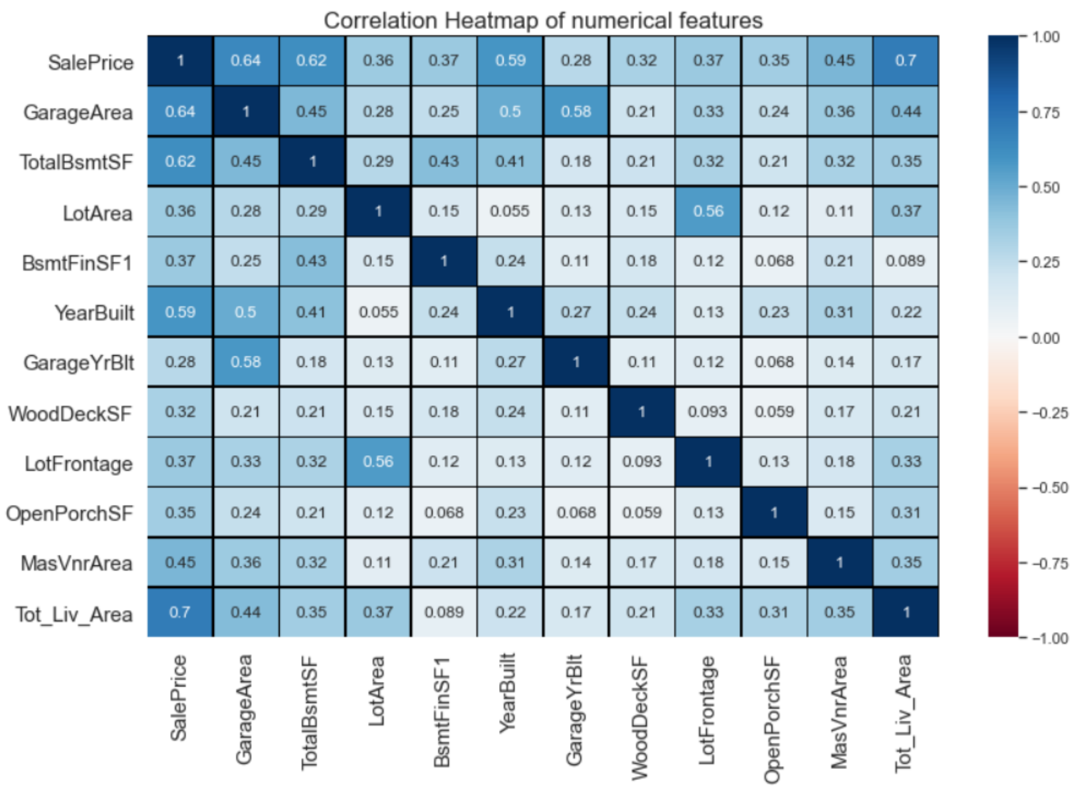


Scatterplots of numerical features

From above scatter plots, we see outliers in the form of individual data points or in clusters. We also see that, for some variables the sale price is non-zero for the zero value of that predictor variable, forming vertical line pattern, which is obvious considering the presence of '**None'** type categories in those variables.

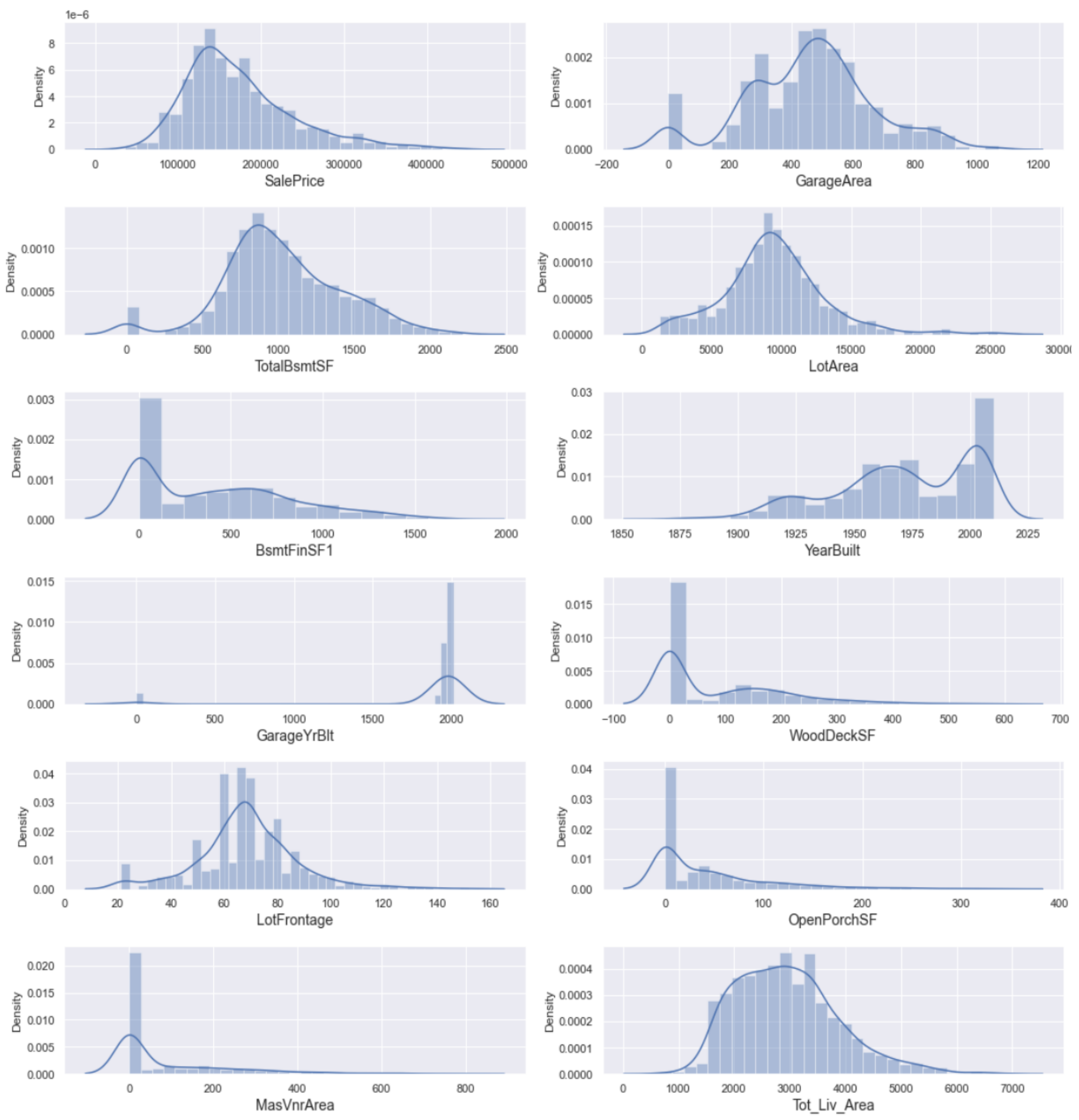
Most significant anomaly occurred at extreme right side of the graph, indicating small sale price of a house corresponding to high value of a particular predictor. Also, most data points corresponding to sale price above 500000 are scattered far away from the main cluster. These remote data points or outliers in a multivariate plane like 2-D plane potentially affects the regression model, and unless the problem statement demands to model these anomalous data points, we can safely discard them before building our regression model. Hence, total of 72 outliers where detected and removed through visual analysis from the 2-D scatterplots.

Then we have analysed the rest numerical variables using a correlation heatmap below.



	SalePrice	GarageArea	TotalBsmtSF	LotArea	BsmtFinSF1	YearBuilt	GarageYrBlt	WoodDeckSF	LotFrontage	OpenPorchSF	MasVnrArea	Tot_Liv_Area
SalePrice	1	0.64	0.62	0.36	0.37	0.59	0.28	0.32	0.37	0.35	0.45	0.7
GarageArea	0.64	1	0.45	0.28	0.25	0.5	0.58	0.21	0.33	0.24	0.36	0.44
TotalBsmtSF	0.62	0.45	1	0.29	0.43	0.41	0.18	0.21	0.32	0.21	0.32	0.35
LotArea	0.36	0.28	0.29	1	0.15	0.055	0.13	0.15	0.56	0.12	0.11	0.37
BsmtFinSF1	0.37	0.25	0.43	0.15	1	0.24	0.11	0.18	0.12	0.068	0.21	0.089
YearBuilt	0.59	0.5	0.41	0.055	0.24	1	0.27	0.24	0.13	0.23	0.31	0.22
GarageYrBlt	0.28	0.58	0.18	0.13	0.11	0.27	1	0.11	0.12	0.068	0.14	0.17
WoodDeckSF	0.32	0.21	0.21	0.15	0.18	0.24	0.11	1	0.093	0.059	0.17	0.21
LotFrontage	0.37	0.33	0.32	0.56	0.12	0.13	0.12	0.093	1	0.13	0.18	0.33
OpenPorchSF	0.35	0.24	0.21	0.12	0.068	0.23	0.068	0.059	0.13	1	0.15	0.31
MasVnrArea	0.45	0.36	0.32	0.11	0.21	0.31	0.14	0.17	0.18	0.15	1	0.35
Tot_Liv_Area	0.7	0.44	0.35	0.37	0.089	0.22	0.17	0.21	0.33	0.31	0.35	1

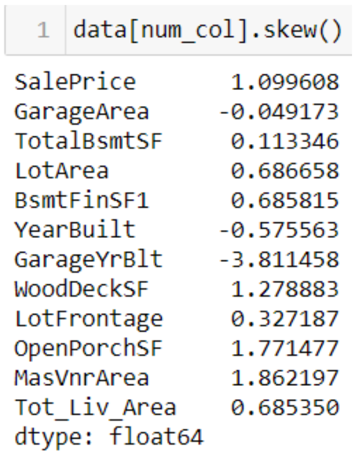
From above heatmap, we see that total living area (**Tot_Liv_Area**) is most correlated to the sale price of the house. In other words, total living area of the house is the most influencer factor in the prediction of the sale price. Also, features like Garage Area, Total basement surface area (**TotalBsmtSF**) and year built also contributed slightly in price prediction. Whereas, other features are not considerably correlated with the target price. Also, features are not significantly correlated to each other to consider dropping them from the dataset.



Distribution plot of numerical features

From the distribution plot, we see that most of the features are skewed. We also see that, most of the houses were bought after year 2000.

Most of this skewness is probably due to ‘**No category’** that exit within the respective categorical variables that concentrates majority points towards zero value. The below snapshot gives the skewness value of these numeric features.



```
data[num_col].skew()
```

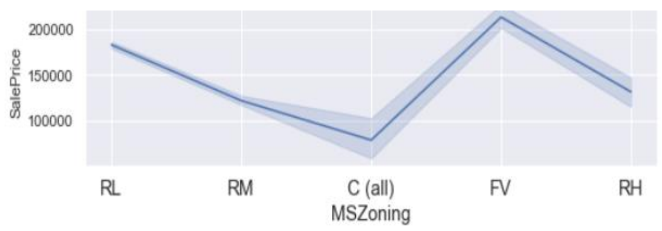
```
SalePrice        1.099608
GarageArea      -0.049173
TotalBsmtSF      0.113346
LotArea          0.686658
BsmtFinSF1       0.685815
YearBuilt       -0.575563
GarageYrBlt     -3.811458
WoodDeckSF       1.278883
LotFrontage      0.327187
OpenPorchSF      1.771477
MasVnrArea       1.862197
Tot_Liv_Area     0.685350
dtype: float64
```

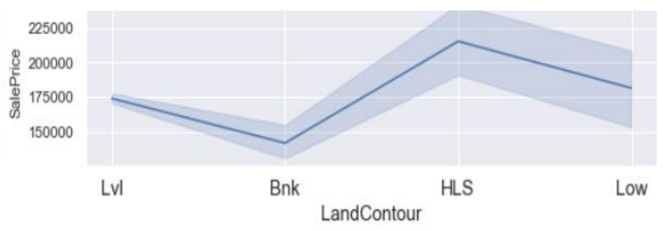
As we can see that **‘GarageYrBlt’** (year in which garage was built) is most skewed with skewness of **-3.8**, moreover, since this feature is least correlated to the sale price, therefore, we discard it from our dataset.

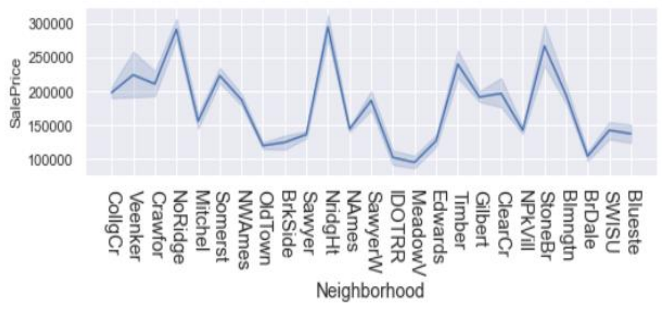
We have then used box cox transformation on these skewed data to make to more normal like distribution.

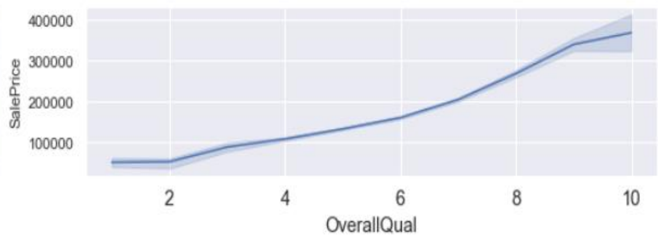
Analysis of Categorical and Discrete Variables

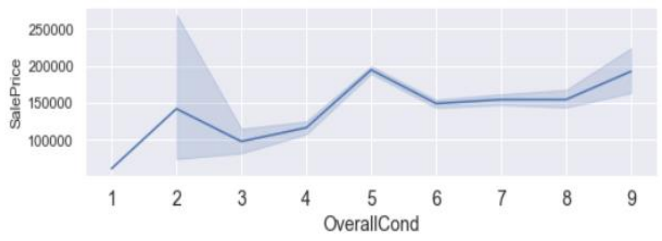
Below is the line plot of each categorical or discrete variables with respect to the target variable, sale price. Here, the line plot gives the mean of a sale price corresponding to particular category or discrete value.

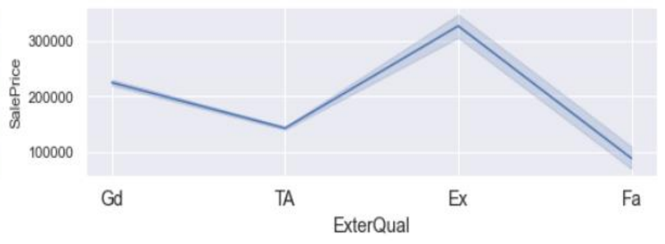


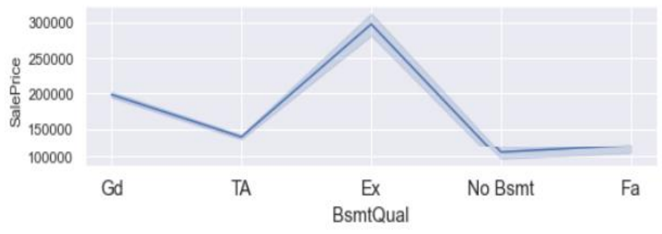


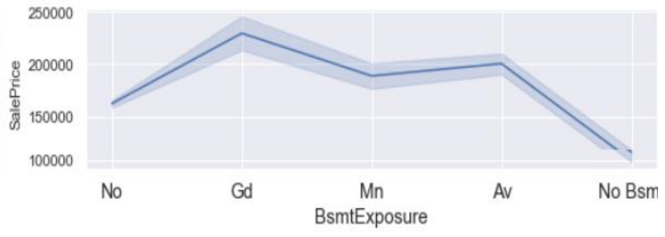


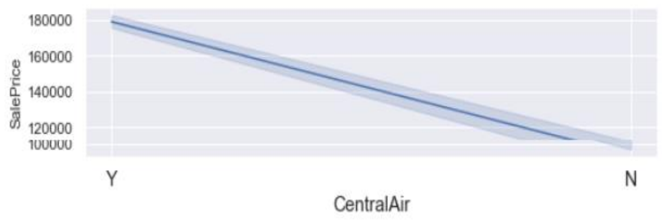


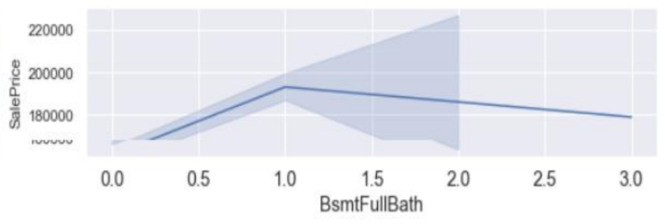


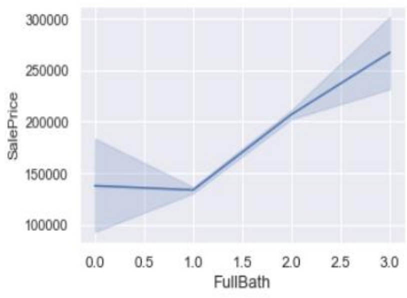


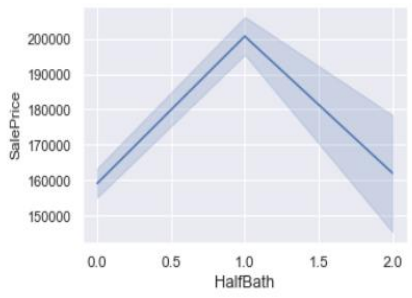


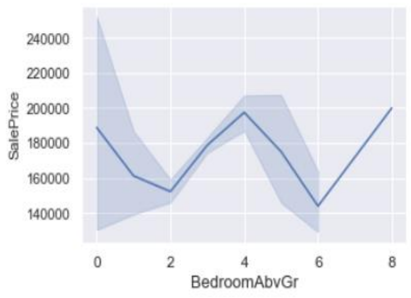


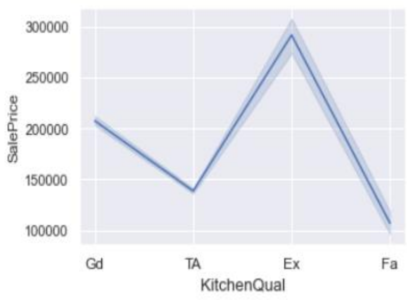


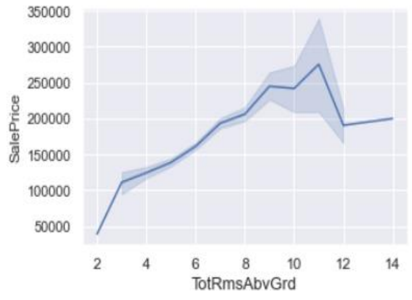


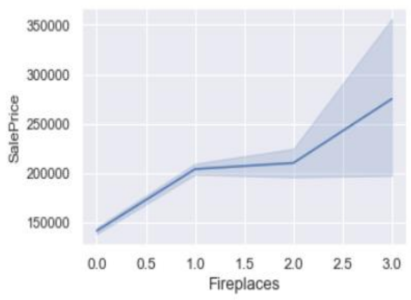


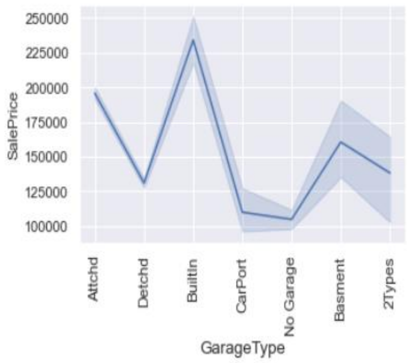


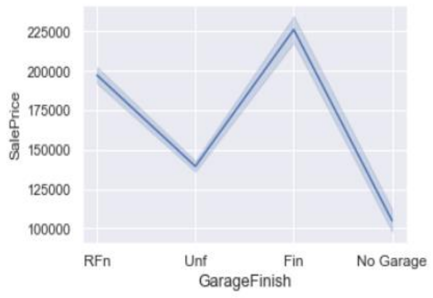


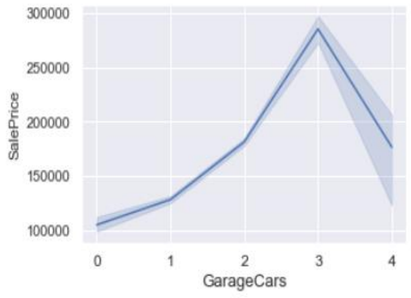


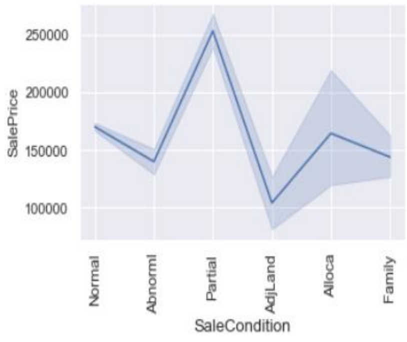


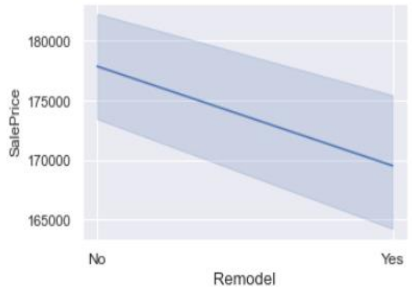












From above plots, we see that column ‘**Neighborhood’** have highest variations in mean sale price of house across different neighborhoods. This quality of a variable is quite useful in determining the effect on overall sale price of a house, moreover, places like Northridge Heights, Stone Brook, Northridge have highest average sale price, whereas, places like Briardale, Meadow Village and Iowa DOT and Rail Road are among least expensive in terms of average sale price of the house. We also see that price of the house is likely to increase monotonically with overall quality of the house. And for rest of the variables the sale price usually increases with increase in quality or number of that variable.

After analysing the above categorical or discrete variables, we decided to separately encode or transform some of these variables containing numbers or having order within them as an ordinal variable before modelling the data and as for the rest of the categorical variables (having nominal characteristics), we have encoded them with the supervised encoding technique called **Target Encoding.**

Target Encoding

It is a supervised encoding technique that uses the target (can be binary or continuous) to encode the categorical variable with the ***'smoothed'*** mean. This technique is particularly useful if we have high cardinality categorical variables that are usually nominal in nature and it also may increase the predictive power of the variable.

Mathematically this smoothed mean is given by:

$$\boldsymbol{\mu} = \frac{\boldsymbol{n * \bar{x} + m * w}}{\boldsymbol{n + m}}$$

Here,

- $\boldsymbol{\mu}$ = is the mean we're trying to compute, also called '***smoothed'*** mean (the one that's going to replace our categorical values)
- $\boldsymbol{n}$ = Number of records of a categorical feature
- $\overline{\boldsymbol{x}}$ = Estimated mean of each category of a feature through group by aggregation
- ***m =*** The weight to assign to overall mean (also a hyper-parameter to tune)
- ***w*** = Overall mean of the target variable

We have implemented the above target encoding in python using user defined function. Also, since the implementation of this supervised encoding method is usually prone to overfitting, therefore, we have tuned the parameter ***m*** using cross-validation on the training set using negative root mean squared error as the scoring metric.

After tuning the ***m*** parameter for target encoding, we got ***m =*** 1.5, we then split our data using 70:30 ratio between train and test set.

We then robust scaled the ordinal/discrete and numerical variables, where we first fitted the robust scaling class on the training set and then used the statistics of the train set to transform both train and test set in order to prevent data leakage.

Now, using the tuned ***m*** parameter as 1.5, we have then target encoded our training and testing set using the statistics of the training set.

2.4 Hardware and Software Tools Used

Hardware used:

Processor: intel i7-9750 H

System type: 64-bit operating system, x64-based processor

Installed RAM: 16 GB

GPU: GeForce GTX 1650 (supports CUDA libraries)

Software and tools used:

Operating Software: Windows 10

Programming Language: Python

IDEs/Framework: Jupyter notebook using Anaconda Framework

Libraries and Packages used:

NumPy: Contains a multi-dimensional array and matrix data structures. It is utilised to perform a number of mathematical operations on arrays.

Pandas: It contains inbuilt Data-structures like Data-Frames and Series, which is used for data manipulation and analysis.

Matplotlib and Seaborn: Used as a data visualization tool.

Scikit-learn: It is used in implementation of various machine learning algorithms and it also contains various classes and functions for data pre-processing metrics for model evaluation.

SciPy: SciPy is an open-source Python library used for scientific computing and technical computing. In our case, we used it for the box cox transformation.

XGBoost: It is an optimized gradient boosting framework that uses tree-based learning algorithms. In our case, we used one of the class called **XGBRegressor** for our regression problem.

3. Model Development and Evaluation

3.1 Problem-solving approaches

- The first step is to identify the irregularities and null values in the dataset according to the data description and handle them accordingly by understanding the distribution of these variables.

- To look out for features that do not add any value to the analysis and modelling, for example, features having only one unique value or all unique values, so that we can discard it before proceeding.

- As the number of features in the dataset was large enough for just 1460 rows as it may cause unnecessary noise and overfitting in our regression analysis, therefore, it is wise to first use some robust feature elimination or selection technique (in our case **RFE**) to filter out to only those features that contribute in the prediction of the target variable.

- Try to find out features that can be used to build a new important feature as a feature engineering task, like in our case using first, second and ground floor area to calculate total living area that can contribute to the overall sale price of the house.

- To separately analyse categorical and numerical features using appropriate visualization techniques for each.

- One of the important aspects of regression is outlier analysis, and to understand why it occurred in the first place, as these outliers heavily influence the least square lines in linear models. To build a robust model it is important to inspect them using rather 2-D scatterplot (with respect to target variable) then a 1-D boxplot to understand the formation of these remote data points and until the problem statement does not demand the anomaly detection use-case, it is better to discard them for a regression problem before proceeding.

- Do multivariate analysis with pair-plots or correlation heatmaps of continuous/numerical variables, as for a regression problem, we can directly visualize the effect of the independent variable on the target variable using these visualization techniques.

- Try to analyse the categorial data using appropriate visualization tools and encode the ordinal variable separately from the nominal variables.

- Using of supervised encoding technique for nominal variable is helpful for high cardinality, however, this technique may lead to overfitting, therefore, it is better to first tune the weight parameter (***m***) using cross-validation on the training set.

- Finally, before robust scaling or target encoding variables for modelling, it is very important to use the statistics of only train set before transforming both train and test set in order to prevent data leakage.

Algorithm used for Data Modelling

As mentioned earlier, we have used 3 supervised learning algorithms that consists of linear algorithm like:

- Elastic Net

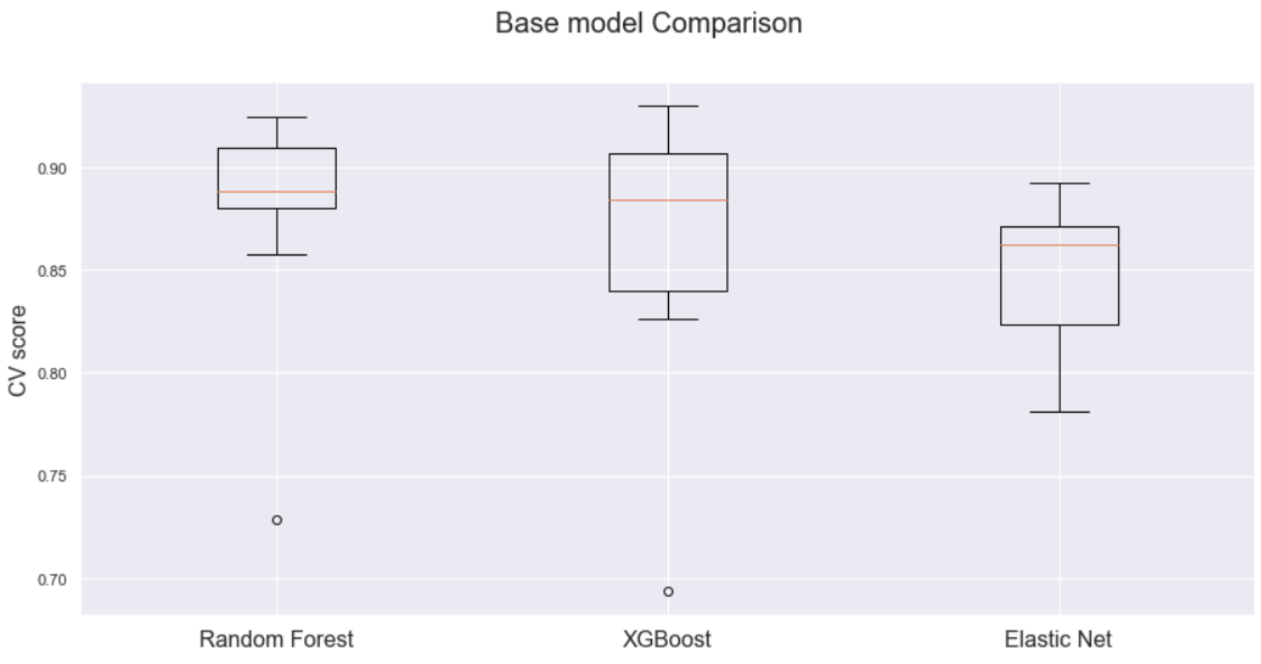
And algorithms featuring tree-based ensembled techniques like:

- Random Forest

- XGBoost (Extreme Gradient Boosting)

3.2 Base Model Evaluation

Base model evaluation was done by cross validation on training set using k-fold = 10, that implies, 90% of data was used in training and 10% on testing, which was iteratively done on each of kth fold, and the final score being the average test score of all these folds. Also, we have used scoring function as negative root mean squared error.



From above boxplot, we see that Random Forest and XGBoost performed better overall, having said that, Random Forest and XGBoost might suffer overfitting which is indicated by high standard deviations in score due to presence of outliers, whereas, elastic net has comparatively more stable score indicated by lesser standard deviation. Nevertheless, this could probably due to lack of regularization, which can we further tuned by hyper-parameter tuning.

3.3 Train and Test set results

Metric	Random Forest	XGBoost	Elastic Net
Root Mean Squared Error (RMSE)	8429.1	929.9	25232.7
Mean Absolute Error (MAE)	5762	635.6	18373.2
R^2 score (%)	98.43	99.98	85.94

Train Results

Metric	Random Forest	XGBoost	Elastic Net
Root Mean Squared Error (RMSE)	19966	19942.2	24297.3
Mean Absolute Error (MAE)	14131.3	14549	18523.8
R^2 score (%)	90.54	90.56	85.99

Test Result

By comparing train and test results, although tree-based complex models like Random Forest and XGBoost performed better overall on the test set but overfitted more when compared to their training results, whereas, the Elastic Net generalized better on the test set compared to its training result.

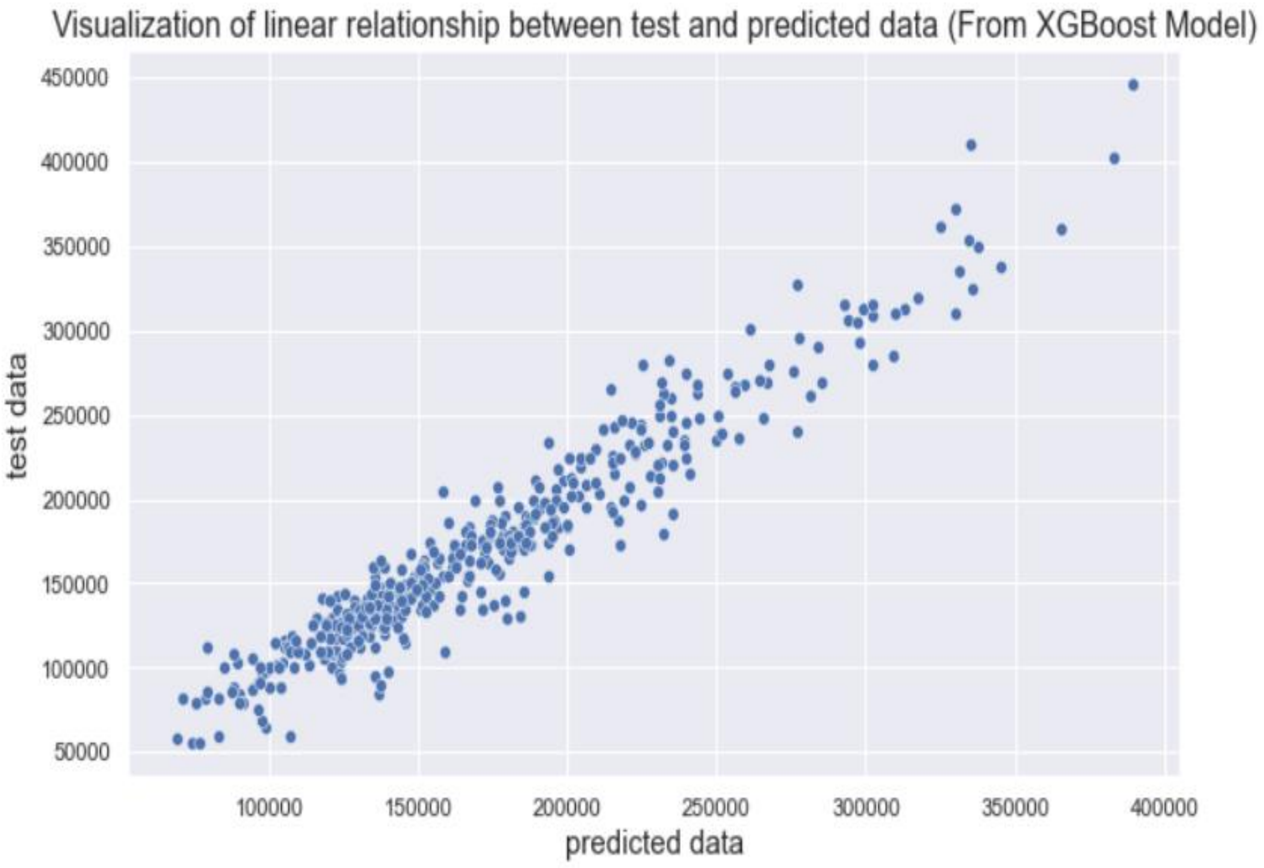
3.4 Test results after Hyper-Parameter Tuning

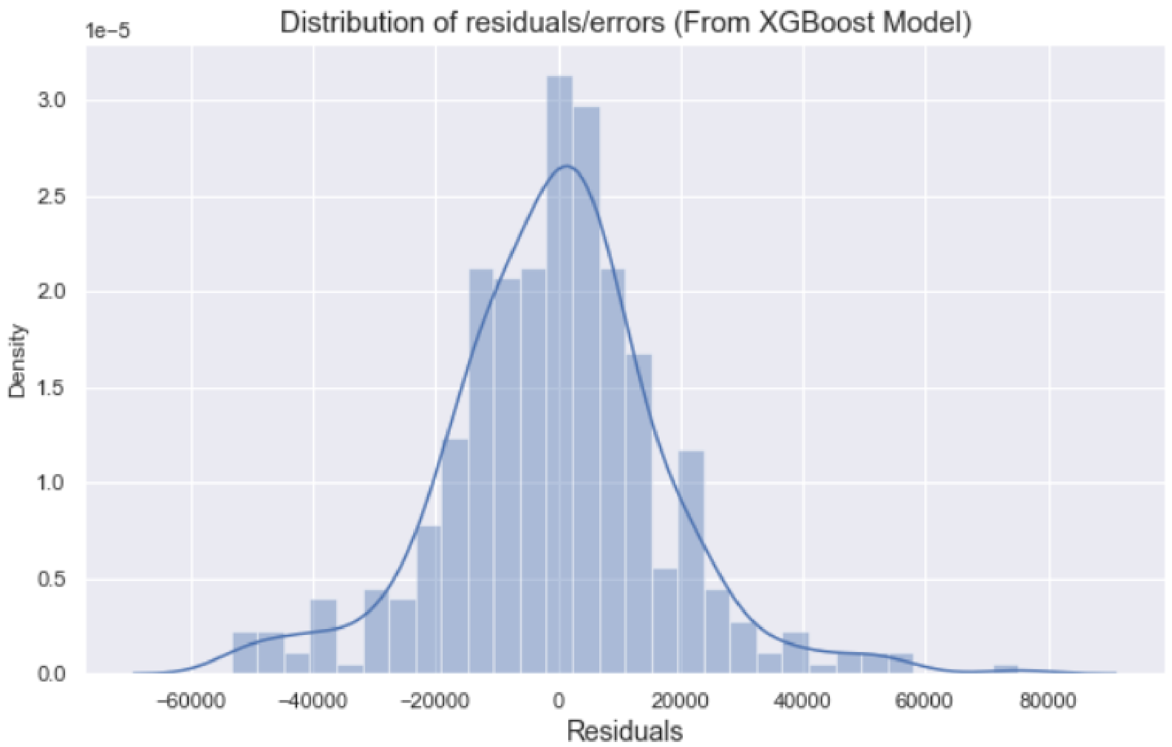
Tuning of hyper-parameters for improving model's performance was done using **RandomizedSearchCV** class of scikit-learn. Although this technique of searching parameter does not guarantee the most optimal combination of hyper-parameters, but **RandomizedSearchCV** randomly searches the combination of hyper-parameters of the model which is very computationally efficient when trying out different set of hyper-parameters to tune.

Here, we have used 10-fold cross-validation to evaluate the model for a combination of hyper-parameters, in which it is evaluated based on the scoring metric, negative Root Mean Squared Error (**RMSE**).

Metric	**Random Forest**	**XGBoost**	**Elastic Net**
Root Mean Squared Error (RMSE)	19575.7	17703.2	22305.8
Mean Absolute Error (MAE)	14131	12975	17062.2
R^2 score (%)	90.9	92.6	88.2

Test Results after tuning hyper-parameters


Visualization of linear relationship between test and predicted data (From XGBoost Model)
test data
predicted data
450000
400000
350000
300000
250000
200000
150000
100000
50000
100000
150000
200000
250000
300000
350000
400000


Distribution of residuals/errors (From XGBoost Model)
1e−5
Density
Residuals
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−60000
−40000
−20000
0
20000
40000
60000
80000

3.5 Understanding Key metrics for success and the interpretation of the final test results

Root Mean Squared Error (RMSE): RMSE is a very common evaluation metric. It can range between 0 and infinity. Lower values are better.

Mathematically it is denoted by:

square root of (1/n * (∑ (y -ŷ) ^{2}))

Here,

n = Number of observations

y = Actual value

ŷ = Predicted value

RMSE is proportional to the squared error and is sensitive to outliers and can exaggerate results if there are outliers in the data set. Therefore, it may not be an appropriate metric for evaluation in order to tell how well the model is doing as it may lead to less interpretation of the final result. However, if we care about penalizing large errors, it's not a bad choice, indeed it is a great choice for a loss metric when hyperparameter tuning the model is concerned.

Mean Absolute Error (MAE): Mean Absolute Error (MAE) is simply the average of the absolute value of the errors.

Mathematically it is denoted by:

(1 / n) * (∑ |y - ŷ|)

MAE is the simplest evaluation metric and most easily interpreted. It’s a great metric to use if we don’t want a few far-off predictions to overwhelm a lot of close ones. It’s a less good choice if you want to penalize predictions that were really far off the mark.

R-Squared (R^2): R^2 represents the proportion of variance explained by our model.

R^2 is a relative metric, so we can use it to compare with other models trained on the same data. And we can use it to get a rough a feel for how well a model performs, in general.

Mathematically it is denoted by:

1 - (SSE/SST)

Here,

SSE = Sum of squared errors; the sum of the squared differences between the actual values and predicted values.

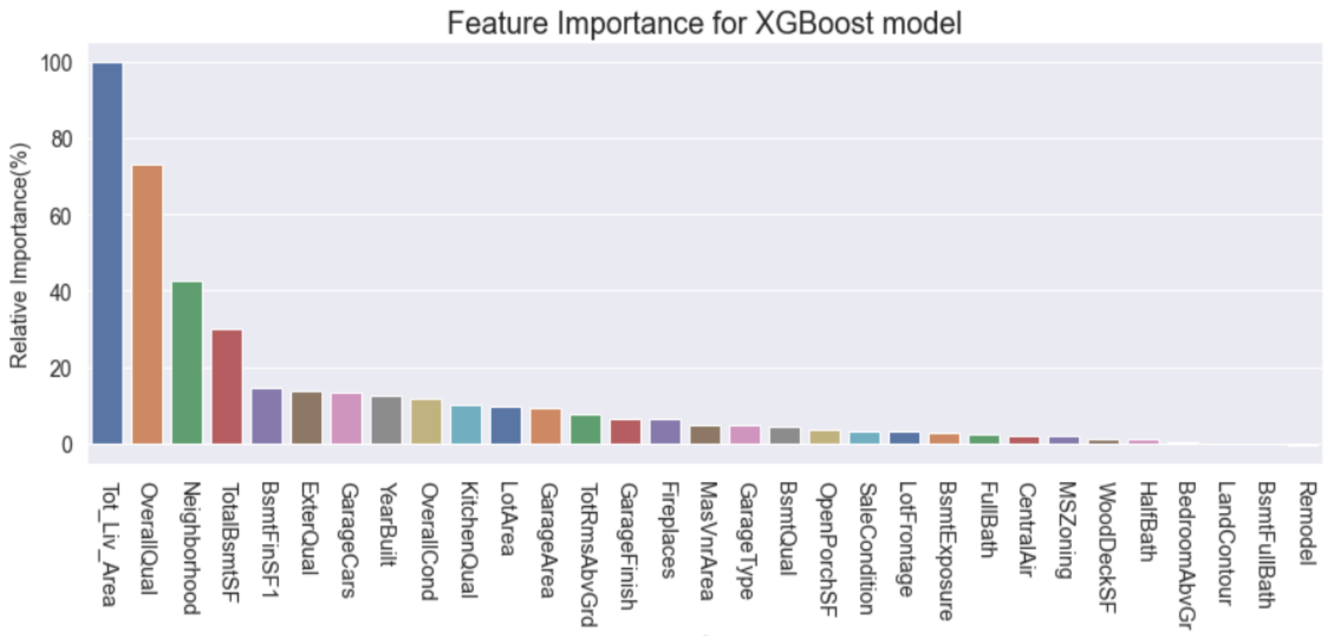
SST = The sum of the squared differences between the actual values and the mean of the actual values.

R² tells you how much variance your model accounts for. It's useful because the R² for any regression problem will immediately provide some understanding of how well the model is performing.

Therefore, from above final test results, it is clear that XGBoost model performed the best out of other two models, and to put in into perspective, in terms of **MAE** it has approximately **8.2 %** less error compared to the Random Forest and **24 %** less error compared to the Elastic Net.

From the scatterplot of the predicted and actual test data from XGBoost model, as these data are correlated to each other, we can visualize their linear relationship, that gives us the indication of a good regression model.

Also, from the distribution plot of residuals, we see that residuals from the XGBoost model are almost normally distributed with mean approximately close to zero, which is a standard for a preferred regression model.



The above figure is the Feature Importance plot of the XGBoost model, which is calculated with the help of **scikit-learn** class **permutation importance.**

Permutation feature importance is a model inspection technique that can be used for any fitted model. This is especially useful for non-linear or opaque models. The **permutation_importance** function calculates the feature importance of the estimators/models for a given dataset. The **n_repeats** parameter of the function sets the number of times a feature is randomly shuffled and returns a sample of mean feature importance. Also, it is important to perform this feature importance technique on test data for better generalized result.

4. CONCLUSION

- It was observed that, through statistical analysis and feature importance calculations, total living area of a house **('Tot_Liv_Area'**) is most important feature in predicting selling price of the house. Apart from this feature, in order of decreasing importance, 2nd being Overall quality of a house **('OverallQual'**), 3rd being the neighborhood at which the house is located and 4th being the total basement surface area of a house (**'TotBsmtSF'**) also plays important role in predicting house price. And after **'TotBsmtSF'**, the relevance of the features for predicting the sale price decreases below 20 % in relative importance.

- The overall material and finish (i.e., the quality of the house) is much significant than the overall condition of the house for determining the sale price of that house.

- The various visualizations tools and software helped to understand the dependencies of variables with respect to each other and the sale price.

- The main criteria for selecting the algorithm used in this project was to use different types of algorithm in terms of their working principles and advantages and disadvantages.

- Just like for any data science project, before modelling and prediction part, the most important and time-consuming part was the data analysis, cleaning and pre-processing with appropriate assumptions and methods, which requires the both, respective domain knowledge and efficiency in data manipulation.

- For any Data Science and Machine Learning project, the quality of the data is the main fuel to uncover any insight to the problem statement. And since the population from where this sample data is collected, is dynamic in nature and may change with time, therefore, it is recommended to use similar kinds of recent datasets from various data sources and combining them to start all over again to find whether similar patterns or results would occur. This could probably reduce the bias in results for a particular sample of a population and can possibly give much better and more concrete scenario for the recent years.