House Sales Price Prediction

Submitted by:

Deepro Sengupta

ACKNOWLEDGMENT

This project would not have come to fruition without the kind guidance of our senior/mentor Mr. Sajid Choudhary at FlipRobo Technologies. His help, inputs and kind consideration have been invaluable to this project. Secondly, I would like to extend my gratitude to all mentors and teachers at DataTrained for training me to be a good data scientist and giving me ample opportunity practise my data skills real-life data problems. Lastly, I would like to thank my parents for their support.

INTRODUCTION

The most valuable possession that an average person has is probably a house that he/she owns. Not only are they a secure investment but also an appreciating asset as in general price of house usually goes up year on year.

Before buying a house, it is important for both the buyer as well as the seller to buy (or sell) at the right and fair price. This ensures that the buyer gets a good value for his money while ensuring the seller receives a good return on his investment. Determining the price for a house, however, is a tricky business as there are a lot of aspects to take into consideration. Traditionally, the price of a house was either determined by the seller or by an appraiser after examination of the property. This method, however, is destined to be flawed as human involvement will inadvertently bring in bias.

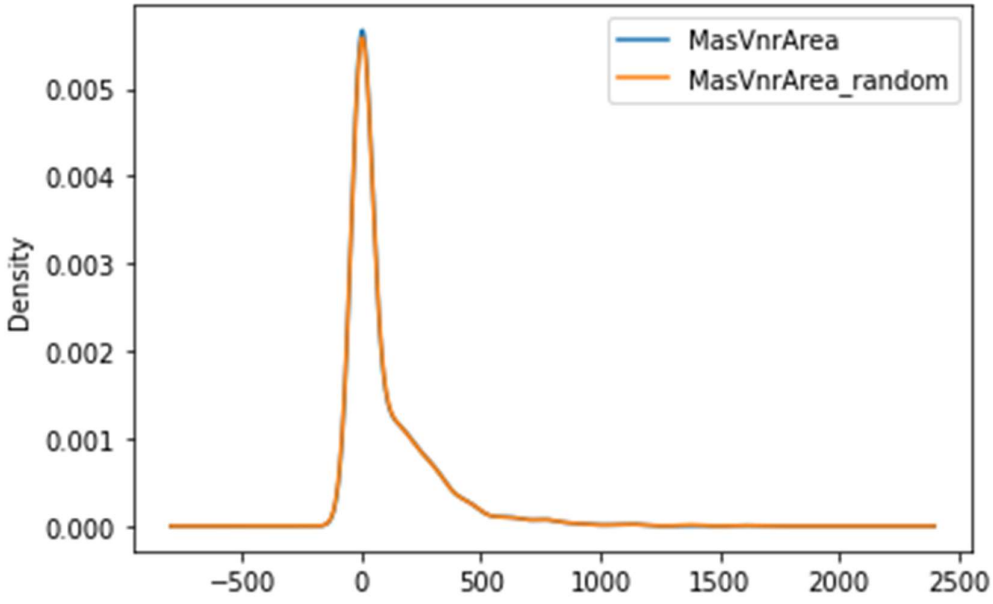
The aim of this project is to use data from previous housing sales to build a model which can predict the sale price given certain inputs.

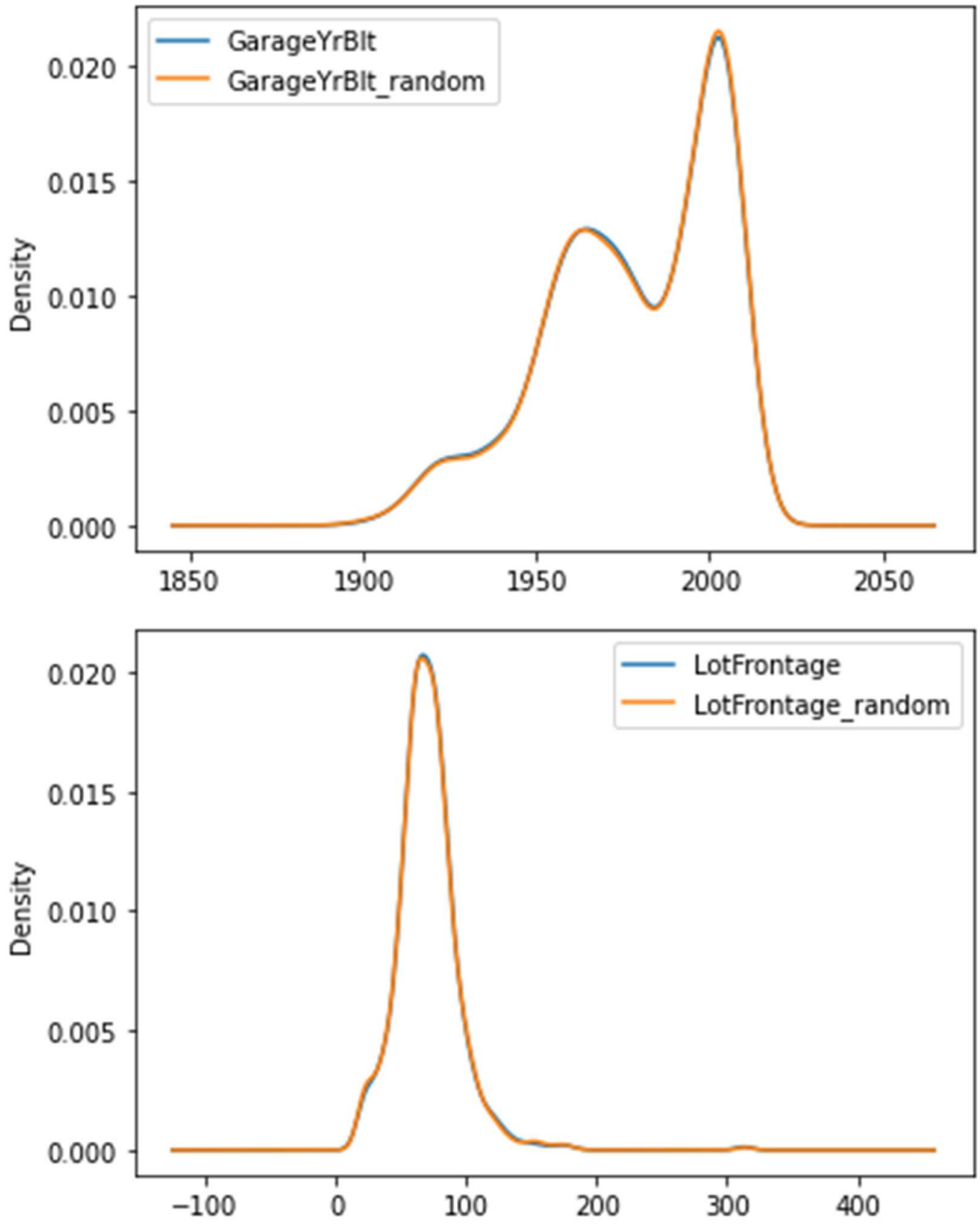
Analytical Problem Framing

The problem we have in this project is a supervised classification problem as the data provided has our target variable which we can use the first find train our algorithm and then use the trained model to predict the probability values.

The data for this project contains 1,168 rows and 81 columns. Out of the 81 columns, 38 columns hold numerical values while 39 columns are categorical columns.

In order to prepare the data for model building, some pre-processing was necessary. Firstly, all columns for checked for missing values. The columns 'LotFrontage','MasVnrType','MasVnrArea','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','FireplaceQu','GarageType','GarageYrBlt','GarageFinish','GarageQual', and 'GarageCond'. Since there are missing values in both numeric and categorical columns, we will treat them differently. For numeric columns, after verifying that the values are MCAR (Missing Completely as Random), I chose to use random number imputation technique as it preserves the original distribution of the column. The diagram below represents the column distribution before and after random number imputation.



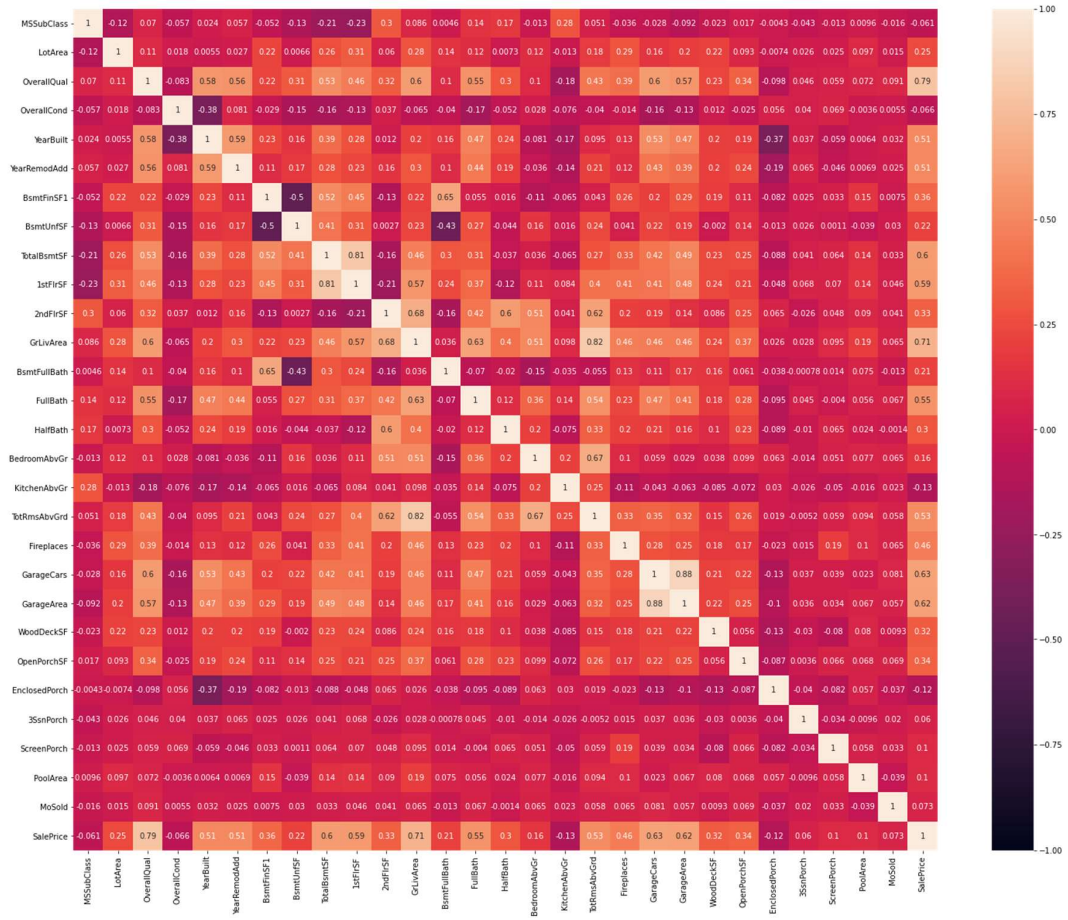


The column name with suffix '_random' represents the distribution after random number imputation while the column name represents the original column distribution. As evident, the original distribution is preserved in the imputed columns. For the categorical columns, the missing values were replaced with the mode of the column.

Since the dataset has 81 columns. The next step was to perform some data exploration and hypothesis testing to identify significant columns and exclude the ones which may not have any statistically significant relation with the target variable.

For the numerical columns, Pearson correlation test was done for independent column with the dependant column. All numeric columns with a P-values of less the 0.05 was considered for analysis. This revealed that the columns 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'BsmtFinSF1', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'GrLivArea', 'BsmtFullBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', and 'MoSold' had statistically significant relation with the target column.

For the categorical columns, Chi-square test of significance was performed. Like the numeric columns, those with P-value of less the 0.05 was considered. However, none of the categorical columns had any statistically significant relation with the target column and hence excluded from the analysis.



Next, the correlation matrix was checked for the significant numeric columns to ensure there is no multi-collinearity. There was no multi-collinearity found as evident of the correlation heat map above.

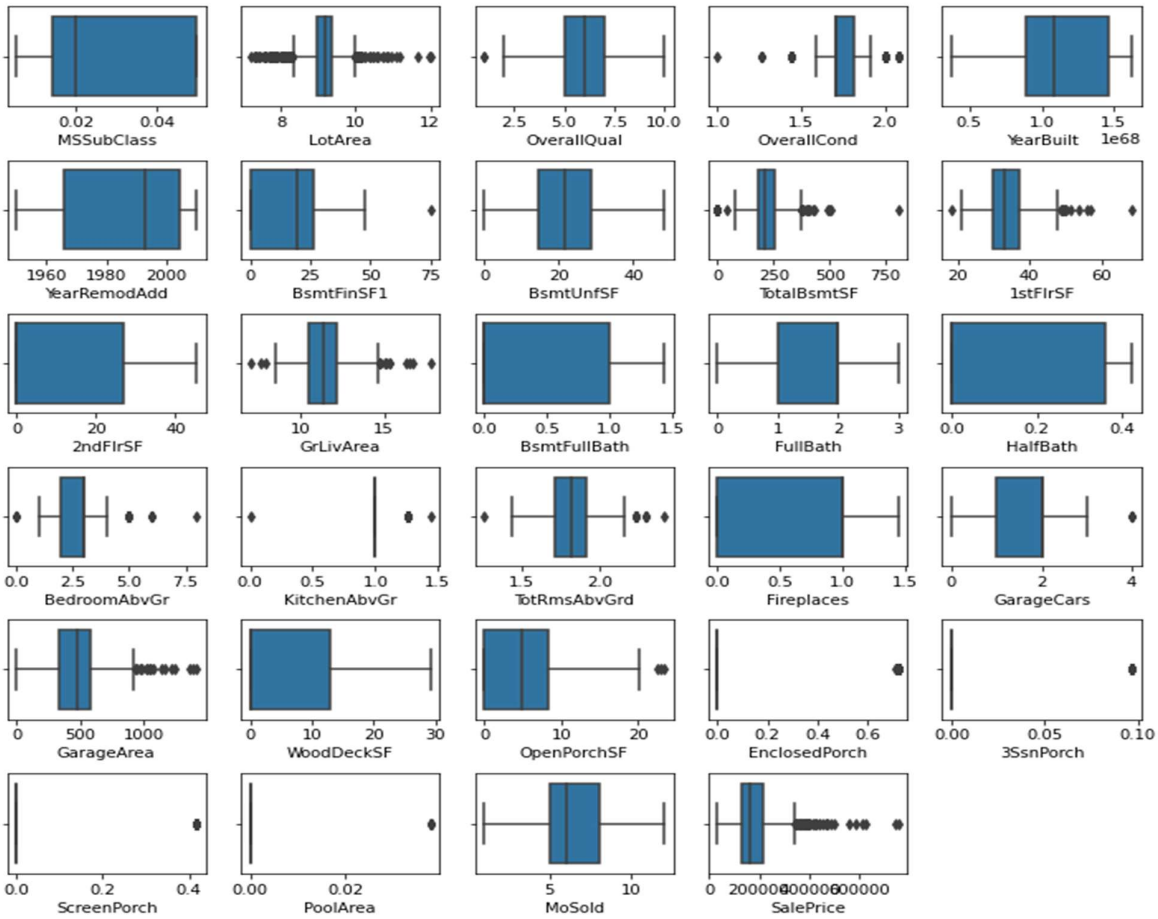
The numeric columns were then checked for skewness. The columns 'MSSubClass','LotArea','OverallCond','YearBuilt','BsmtFinSF1','BsmtUnfSF', 'TotalBsmtSF','1stFlrSF','2ndFlrSF','GrLivArea','BsmtFullBath','HalfBath','KitchenAbvGr','TotRmsAbvGrd','Fireplaces','WoodDeckSF','OpenPorchSF','EnclosedPorch','3SsnPorch','ScreenPorch', and 'PoolArea' had skewness which was greater (or less) than the acceptable limit of ± 0.5. The following is the skewness of the aforementioned columns:

```
MSSubClass       1.422019
LotArea         10.659285
OverallCond      0.580714
YearBuilt       -0.579204
BsmtFinSF1       1.871606
BsmtUnfSF        0.909057
TotalBsmtSF      1.744591
1stFlrSF         1.513707
2ndFlrSF         0.823479
GrLivArea        1.449952
BsmtFullBath     0.627106
HalfBath         0.656492
KitchenAbvGr     4.365259
TotRmsAbvGrd     0.644657
Fireplaces       0.671966
WoodDeckSF       1.504929
OpenPorchSF      2.410840
EnclosedPorch    3.043610
3SsnPorch        9.770611
ScreenPorch      4.105741
PoolArea        13.243711
SalePrice        1.953878
dtype: float64
```

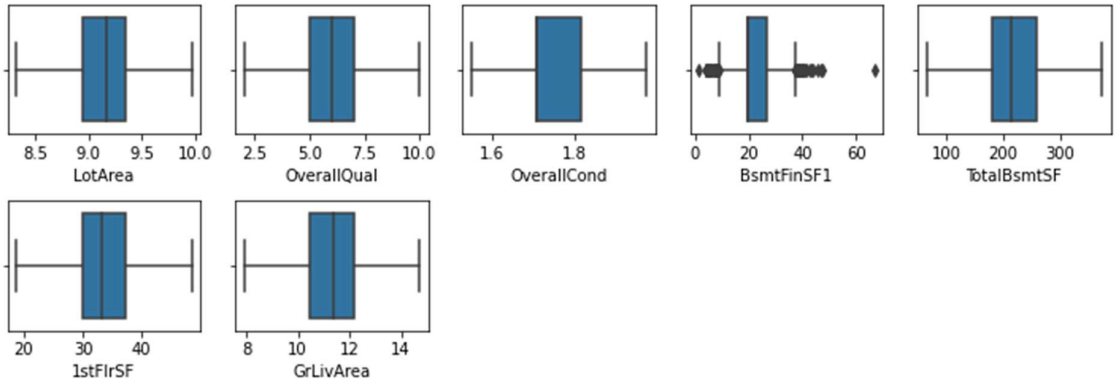
After applying various transformation like, inverse, log, square-root, cube-root and YeoJohnson transformation, the following are the results:

```
MSSubClass        0.313316
LotArea          -0.169680
OverallCond      -0.194972
YearBuilt        -0.126641
BsmtFinSF1       -0.011789
BsmtUnfSF        -0.233752
TotalBsmtSF       0.286779
1stFlrSF          0.686774
2ndFlrSF          0.425547
GrLivArea         0.406535
BsmtFullBath      0.368254
HalfBath          0.498003
KitchenAbvGr     -0.540012
TotRmsAbvGrd      0.091655
Fireplaces       -0.049047
WoodDeckSF        0.458303
OpenPorchSF       0.606878
EnclosedPorch     2.022616
3SsnPorch         7.087955
ScreenPorch       3.067153
PoolArea         12.817372
SalePrice         1.953878
dtype: float64
```

Next, the columns were checked for outliers.



From the boxplot, we see that 'LotArea', 'OverallQual', 'OverallCond', 'BsmtFinSF1', 'TotalBsmtSF', '1stFlrSF', and 'GrLivArea' have outliers. The outliers were removed using the IQR-method where any datapoints outside 1.5 times outside the inter-quartile range (IQR) of the column is replaced with the values of 'upperBridge'(which is 75th percentile value + 1.5 time the IQR) for values higher than the 'upperBridge' and values of the 'lowerBridge' (which is 25th percentile value - 1.5 time the IQR) for values lower than the 'lowerBridge'. After outlier handling, the boxplot for the columns with outliers are shown below:



This project uses the Anaconda Distribution of Python and Jupyter Notebook for analysis. The system requirements are:

- License: Free use and redistribution under the terms of the ../eula.
- Operating system: Windows 8 or newer, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 6+, and others.
- System architecture: Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit Power8/Power9.
- Minimum 5 GB disk space to download and install.

The following packages were used in this analysis:

1. NumPy: For basic mathematical operations
2. Pandas: For DataFrame manipulation
3. Scipy: For statistical analysis and hypothesis testing

4. Matplotlib & Seaborn: For data visualization
5. Scikit-Learn: For data pre-processing, data modelling and model evaluation.

Model/s Development and Evaluation

The following algorithms were taken into consideration for this analysis:

1. Linear Regression
2. Random Forest Regression
3. Decision Tree Regression
4. Support Vector Regression
5. K-Neighbors Regression
6. Multi-Layer Perceptron Regression

```
In [34]: #Fitting various regression models to the train data and printing their perfomance metrics one-by-one
         lr = LinearRegression()
         rfr = RandomForestRegressor()
         dtr = DecisionTreeRegressor()
         svr = SVR()
         knr = KNeighborsRegressor()
         mlp = MLPRegressor()

         models = [lr, rfr, dtr, svr, knr, mlp]

         for model in models:
             model.fit(x_train, y_train)
             pred = model.predict(x_test)
             predTrain = model.predict(x_train)
             print(f'{model}\n\tAccuracy: {model.score(x_test, y_test)}\n\tMSE: {mean_squared_error(y_test, pred)}\n\tRMSE: {mean_squared_
```

The snapshot above shows the coding done to implement the algorithms used in this analysis. First the object for the regression algorithms are initialised. The list ‘model’ contains all the regression objects and is used in the for-loop below to fit and print the evaluation metrics of each of the algorithms one-by-one. The following are the evaluation metrics for the algorithms:

```
LinearRegression()
        Accuracy: 0.7713151755838868
        MSE: 1526358451.6307666
        RMSE: 39068.6376986806
        R2 Score: 0.7713151755838868
        Train-Test difference: 0.04639578059305127
RandomForestRegressor()
        Accuracy: 0.8175617563043698
        MSE: 1217685326.8531876
        RMSE: 34895.34821223579
        R2 Score: 0.8175617563043698
        Train-Test difference: 0.16276771433286885
DecisionTreeRegressor()
        Accuracy: 0.6226211018950369
        MSE: 2518818081.0
        RMSE: 50187.82801636269
        R2 Score: 0.6226211018950369
        Train-Test difference: 0.37737889810496306
SVR()
        Accuracy: -0.06588159280467965
        MSE: 7114234107.002918
        RMSE: 84345.91932632496
        R2 Score: -0.06588159280467965
        Train-Test difference: 0.016114250650999473
KNeighborsRegressor()
        Accuracy: 0.7241039710788493
        MSE: 1841469964.5696864
        RMSE: 42912.352121151394
        R2 Score: 0.7241039710788493
        Train-Test difference: 0.1561581458977006
MLPRegressor()
        Accuracy: -5.004063047894965
        MSE: 40074160586.20083
        RMSE: 200185.31561081304
        R2 Score: -5.004063047894965
        Train-Test difference: -0.2924251293053084
```

Judging by the evaluation metrics, RandomForestRegressor seems to be the best algorithm because:

1. It has the highest Accuracy score.
2. The has the lowest RSME indicating that the errors are not widely spread.
3. Test-Train score difference is not high indicating there is no overfitting.

Due to these reasons, our final model algorithm will be RandomForestRegressor. After selecting the algorithm for the final model, the model is fitted again with all the data in order to give it more data

points to consider. Finally, the test set is imported, pre-processed the same was as the train set and predictions are made and stored in the variable ‘pred’.

Conclusion

The accuracy score for the final model was 81.76%. While this is not a very low score, there may be still some room for improvement. One way this can be done is by incorporating the categorical columns which were ignored after their Chi-square independence test P-values suggested no statistical significance with the target column. Since Chi-square test of significance did suggest any significance with the target variable, one way to incorporate these categorical variables is to train a model first and then perform model explainability analysis like Permutation importance (which tells us which features the model thinks is important), and/or observe partial plots (tells us how each features affect the prediction), etc. Once this analysis is done, this new information can be corroborated with existing domain information in order to incorporated those categorical features with makes sense.