1 Introduction

Purchasing a dream home remains one of the biggest buying decision that individuals make in their lifetime. But it depends on so many factors. Based on data, an intelligent model can predict house price accurately. If the dataset is smaller then it becomes easy to calculate. But when the dataset consists of many features that time, it's a bit tricky. Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. Based on Exploratory Data Analysis features can be selected to train a machine learning model.

This project aims to predict sale prices of houses by using various aspects of residential homes. The insights gathered could be used by individuals to complement their decision making process when purchasing a house.

This helps to maximise the value users can gain while keeping to a budget. Findings could also be used by a property agent to improve the probability of a sale through proper marketing of key variables. The dataset consists of mainly categorical variables stored as integers and factors. Focusing on descriptive and predictive analytics, with suggestions on how additional data could be obtained to conduct more experiments to optimise the purchase. The dataset used in this project is obtained from Kaggle. **There are 1460 instances of training data and 1459 of test data. Total number of attributes equals 81, of which 36 are numerical, 43 are categorical + Id and SalePrice.**

The models are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. (Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.)

2 Objectives and Learning Outcomes

1. Understanding the real meaning of Data, Information and Knowledge.
2. Applying Exploratory Data Analysis (EDA).
3. Applying Python Programming Language.
4. Generating different graphs for better analysis.
5. Building and Evaluating Machine Learning Models.

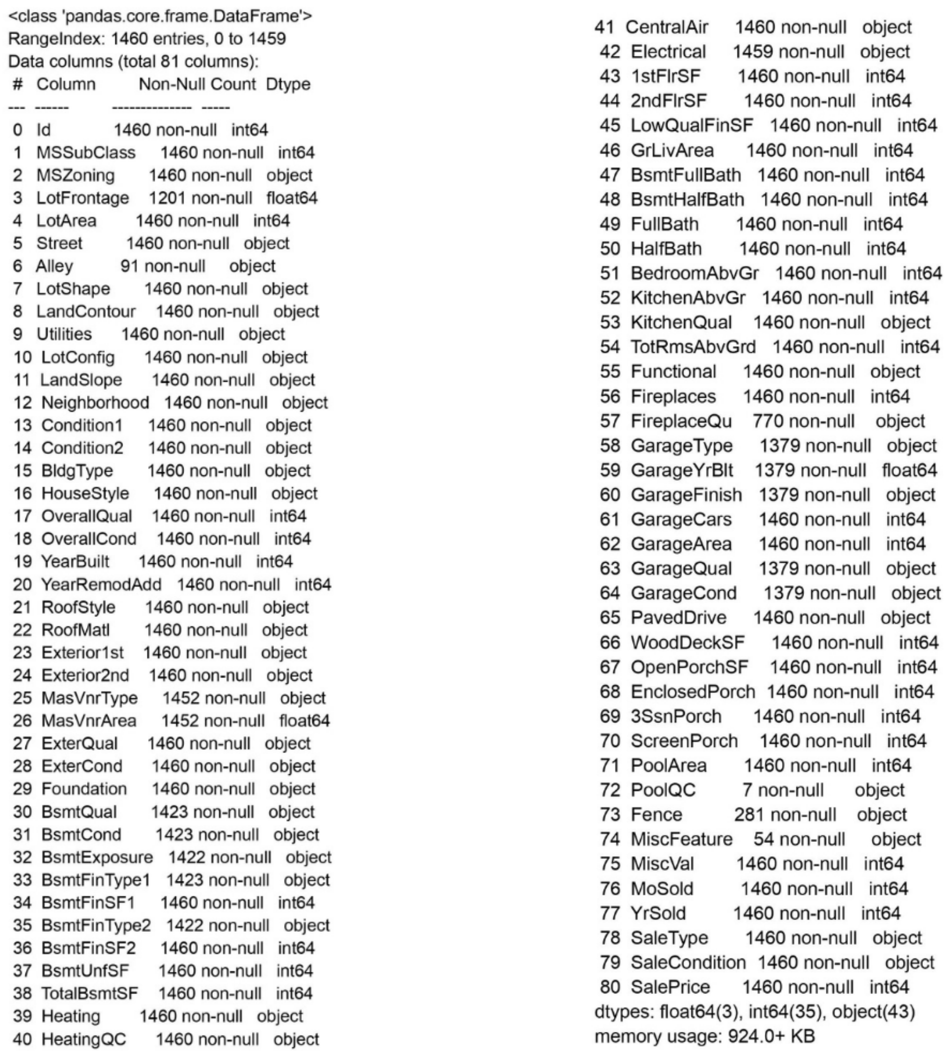
3 Tools and Programming Language

- Programming Language: Python
- Packages: pandas, numpy, seaborn, matplotlib, sklearn, scipy
- Integrated Development Environment (IDE): Google Colab
- Report Writing: Latex

4 Methodology

4.1 Data Collection

From https://www.kaggle.com/ the dataset has been collected. There are 1460 instances of training data and 1459of test data. Total number of attributes equals 81, of which 36 are numerical, 43 are categorical + Id and SalePrice.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #  Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0  Id            1460 non-null   int64
 1  MSSubClass    1460 non-null   int64
 2  MSZoning      1460 non-null   object
 3  LotFrontage   1201 non-null   float64
 4  LotArea       1460 non-null   int64
 5  Street        1460 non-null   object
 6  Alley         91 non-null     object
 7  LotShape      1460 non-null   object
 8  LandContour   1460 non-null   object
 9  Utilities     1460 non-null   object
 10 LotConfig     1460 non-null   object
 11 LandSlope     1460 non-null   object
 12 Neighborhood  1460 non-null   object
 13 Condition1    1460 non-null   object
 14 Condition2    1460 non-null   object
 15 BldgType      1460 non-null   object
 16 HouseStyle    1460 non-null   object
 17 OverallQual   1460 non-null   int64
 18 OverallCond   1460 non-null   int64
 19 YearBuilt     1460 non-null   int64
 20 YearRemodAdd  1460 non-null   int64
 21 RoofStyle     1460 non-null   object
 22 RoofMatl      1460 non-null   object
 23 Exterior1st   1460 non-null   object
 24 Exterior2nd   1460 non-null   object
 25 MasVnrType    1452 non-null   object
 26 MasVnrArea    1452 non-null   float64
 27 ExterQual     1460 non-null   object
 28 ExterCond     1460 non-null   object
 29 Foundation    1460 non-null   object
 30 BsmtQual      1423 non-null   object
 31 BsmtCond      1423 non-null   object
 32 BsmtExposure  1422 non-null   object
 33 BsmtFinType1  1423 non-null   object
 34 BsmtFinSF1    1460 non-null   int64
 35 BsmtFinType2  1422 non-null   object
 36 BsmtFinSF2    1460 non-null   int64
 37 BsmtUnfSF     1460 non-null   int64
 38 TotalBsmtSF   1460 non-null   int64
 39 Heating       1460 non-null   object
 40 HeatingQC     1460 non-null   object
 41 CentralAir    1460 non-null   object
 42 Electrical    1459 non-null   object
 43 1stFlrSF      1460 non-null   int64
 44 2ndFlrSF      1460 non-null   int64
 45 LowQualFinSF  1460 non-null   int64
 46 GrLivArea     1460 non-null   int64
 47 BsmtFullBath  1460 non-null   int64
 48 BsmtHalfBath  1460 non-null   int64
 49 FullBath      1460 non-null   int64
 50 HalfBath      1460 non-null   int64
 51 BedroomAbvGr  1460 non-null   int64
 52 KitchenAbvGr  1460 non-null   int64
 53 KitchenQual   1460 non-null   object
 54 TotRmsAbvGrd  1460 non-null   int64
 55 Functional    1460 non-null   object
 56 Fireplaces    1460 non-null   int64
 57 FireplaceQu   770 non-null    object
 58 GarageType    1379 non-null   object
 59 GarageYrBlt   1379 non-null   float64
 60 GarageFinish  1379 non-null   object
 61 GarageCars    1460 non-null   int64
 62 GarageArea    1460 non-null   int64
 63 GarageQual    1379 non-null   object
 64 GarageCond    1379 non-null   object
 65 PavedDrive    1460 non-null   object
 66 WoodDeckSF    1460 non-null   int64
 67 OpenPorchSF   1460 non-null   int64
 68 EnclosedPorch 1460 non-null   int64
 69 3SsnPorch     1460 non-null   int64
 70 ScreenPorch   1460 non-null   int64
 71 PoolArea      1460 non-null   int64
 72 PoolQC        7 non-null      object
 73 Fence         281 non-null    object
 74 MiscFeature   54 non-null     object
 75 MiscVal       1460 non-null   int64
 76 MoSold        1460 non-null   int64
 77 YrSold        1460 non-null   int64
 78 SaleType      1460 non-null   object
 79 SaleCondition 1460 non-null   object
 80 SalePrice     1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

Figure 1: Training Dataset Information

Numerical Features: 1stFlrSF, 2ndFlrSF, 3SsnPorch, BedroomAbvGr, BsmtFinSF1, BsmtFinSF2, BsmtFullBath, BsmtHalfBath, BsmtUnfSF, EnclosedPorch, Fireplaces, FullBath, GarageArea, GarageCars, GarageYrBlt, GrLivArea, HalfBath, KitchenAbvGr, LotArea, LotFrontage, LowQualFinSF, MSSubClass, MasVnrArea, MiscVal, MoSold, OpenPorchSF, OverallCond, OverallQual, PoolArea, ScreenPorch, TotRmsAbvGrd, TotalBsmtSF, WoodDeckSF, YearBuilt, YearRemodAdd, YrSold

Categorical Features: Alley, BldgType, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, BsmtQual, CentralAir, Condition1, Condition2, Electrical, ExterCond, ExterQual, Exterior1st, Exterior2nd, Fence, FireplaceQu, Foundation, Functional, GarageCond, GarageFinish, GarageQual, GarageType, Heating, HeatingQC, HouseStyle, KitchenQual, LandContour, LandSlope, LotConfig, LotShape, MSZoning, MasVnrType, MiscFeature, Neighborhood, PavedDrive, PoolQC, RoofMatl, RoofStyle, SaleCondition, SaleType, Street, Utilitif

Data Description:

- MSSubClass: Identifies the type of dwelling involved in the sale. Example:

20 1-STORY 1946 NEWER ALL STYLES, 30 1-STORY 1945 OLDER

- MSZoning: Identifies the general zoning classification of the sale. Example: A Agriculture, C Commercial
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access to property. Example: Grvl Gravel Pave Paved
- Alley: Type of alley access to property. Example: Grvl Gravel , Pave Paved
- LotShape: General shape of property. Example: Reg Regular, IR1 Slightly irregular
- LandContour: Flatness of the property. Example: Lvl Near Flat/Level
- Utilities: Type of utilities available. Example: AllPub All public Utilities (E,G,W, S)
- LotConfig: Lot configuration. Example: Inside Inside lot
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to various conditions
- Condition2: Proximity to various conditions (if more than one is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Rates the overall material and finish of the house
- OverallCond: Rates the overall condition of the house
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Evaluates the quality of the material on the exterior
- ExterCond: Evaluates the present condition of the material on the exterior

- Foundation: Type of foundation
- BsmtQual: Evaluates the height of the basement
- BsmtCond: Evaluates the general condition of the basement
- BsmtExposure: Refers to walkout or garden level walls
- BsmtFinType1: Rating of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Rating of basement finished area (if multiple types)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
- Kitchen: Kitchens above grade
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality (Assume typical unless deductions are warranted)
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location

- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: Value of miscellaneous feature
- MoSold: Month Sold (MM)
- YrSold: Year Sold (YYYY)
- SaleType: Type of sale
- SaleCondition: Condition of sale

4.2 Data Cleaning and Preparation

Looking at the information from the data train DataFrame, some columns have missing values. These missing values will be solved later. But before that some exploratory data analysis and feature engineering will be performed.

4.3 Exploratory Data Analysis and Feature Engineering

Before investigating the numerical and categorical features from both datasets, the "Id" column will be dropped from both datasets as well. The list of IDs from the test set will be saved so that it can be used later on.

Before dropping Id column:

Training set shape: (1460, 81)

Test set shape: (1459, 80)

After dropping Id column:

Training set shape: (1460, 80)

Test set shape: (1459, 79)

At first exploratory data analysis and some feature engineering will be performed on numerical features on the datasets.It's important for the following reasons.

- Investigate the distribution of these numerical features
- Investigate the correlation of these numerical features with "SalePrice" and drop any features that have low correlation;
- Deal with missing values through imputation

Imputation is the process of replacing missing data with substituted values.

Operation with Numerical Features:

Firstly, selecting the numerical columns from data train (including "SalePrice") by obtaining the list of numerical columns from data train in numerical cols, and using this to take out the numerical columns from data train and store them in a new DataFrame data train num (this process will repeat for the test dataset).

As a result, there are 37 numerical columns. It's time to plot the distribution of the numerical features.

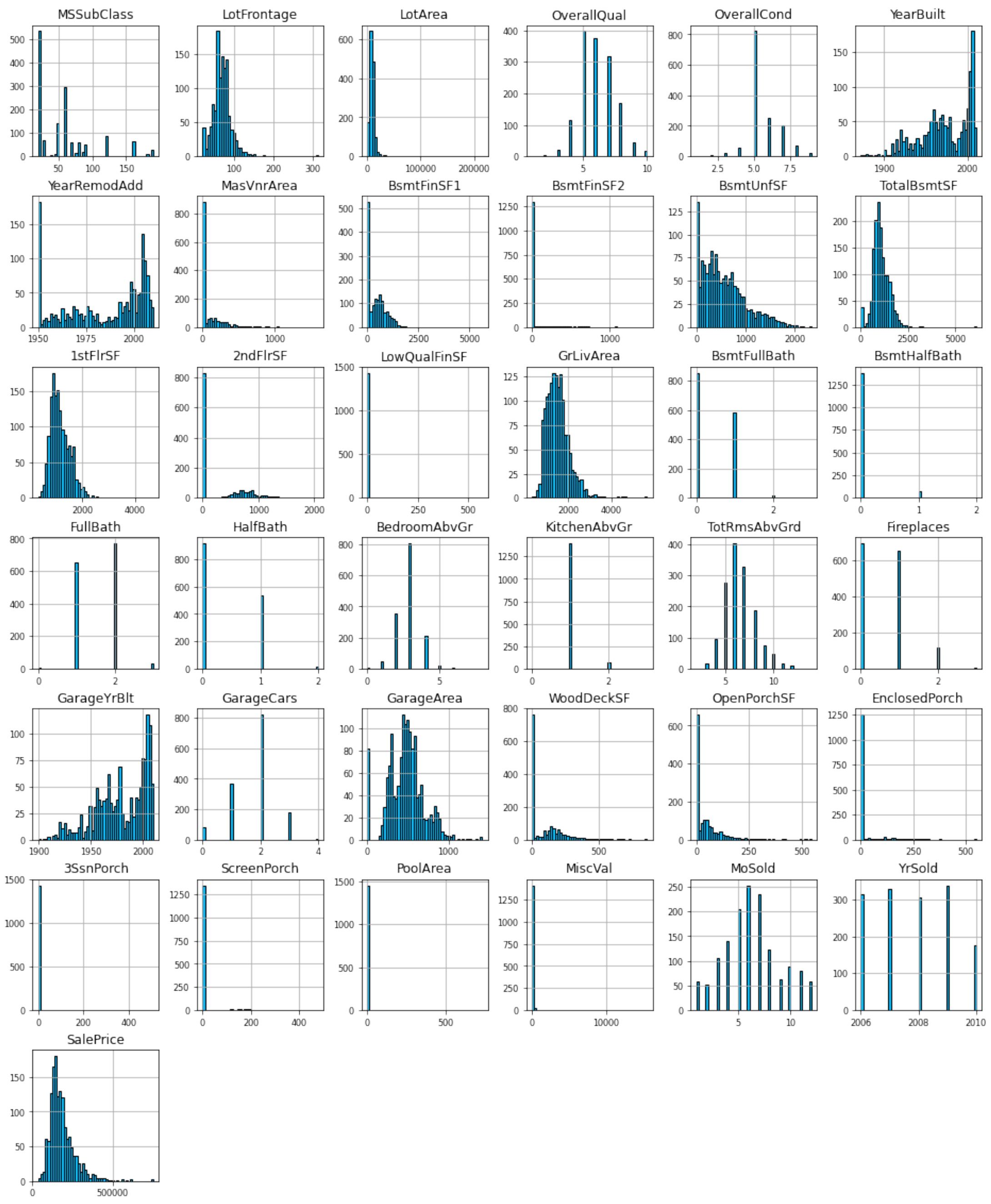


Figure 2: Distribution of All the Numerical Features

Looking at the distributions of each numerical feature, it is clear that thee is a mixture of discrete and continuous variables in the dataset.

One thing from these plots are variables that has small variation (i.e. they have very similar values) - these variables are likely to only have a small impact on the final price of the house. So it is better to drop variables where 95% of the values are similar or constant.

By using python code only one variable ”KitchenAbvGr” (Kitchens above the grade) has at least 95% of variables that are similar and constant, so this will be removed from both the training and test datasets.

At the moment we are working with Numerical Features. After dropping quasi-constant features from both datasets.

Training set shape (Numerical features): (1460, 36)
Test set shape (Numerical features): (1459, 35)

Now it's time to produce a correlation heatmap showing the correlation between all of the numerical variables including the correlation of each numerical feature with "SalePrice". Any variables that have a low correlation with "SalePrice" is likely not to have a huge impact on the final sale price of the house, any variables with correlation less than —0.3— will be replaced by 0 for simplicity.

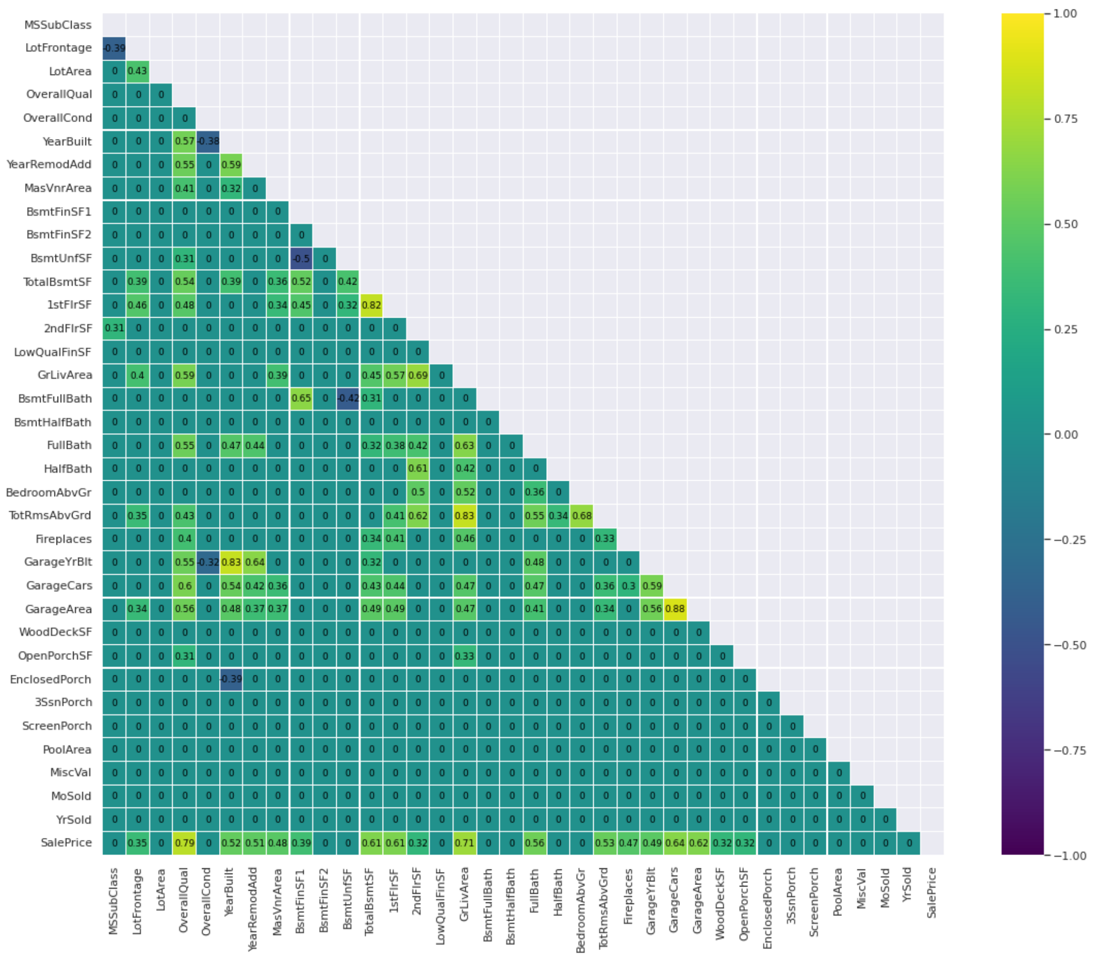


Figure 3: Heatmap for all the remaining numerical data including the taget 'SalePrice

From the heatmap 18 of the numerical features has some noticeable correlation with "SalePrice". There are also a few variables that have a very strong positive correlation (of at least 0.8) including

- "GarageArea" and "GarageCars" (0.88)
- "GarageYrBlt" and "YearBuilt" (0.83)

- "TotRmsAbvGrd" and "GrLivArea" (0.83)
- "1stFlrSF" and "TotalBsmtSF" (0.82)

The other correlations will be dealt with later, but first we will focus on features that have a correlaton of more than 0.3 with "SalePrice".

10 strongly correlated values with SalePrice:

OverallQual 0.79

GrLivArea 0.71

GarageCars 0.64

GarageArea 0.62

TotalBsmtSF 0.61

1stFlrSF 0.61

FullBath 0.56

TotRmsAbvGrd 0.53

YearBuilt 0.52

8 slightly correlated values with SalePrice:

GarageYrBlt 0.49

MasVnrArea 0.48

Fireplaces 0.47

BsmtFinSF1 0.39

LotFrontage 0.35

WoodDeckSF 0.32

2ndFlrSF 0.32

OpenPorchSF 0.32

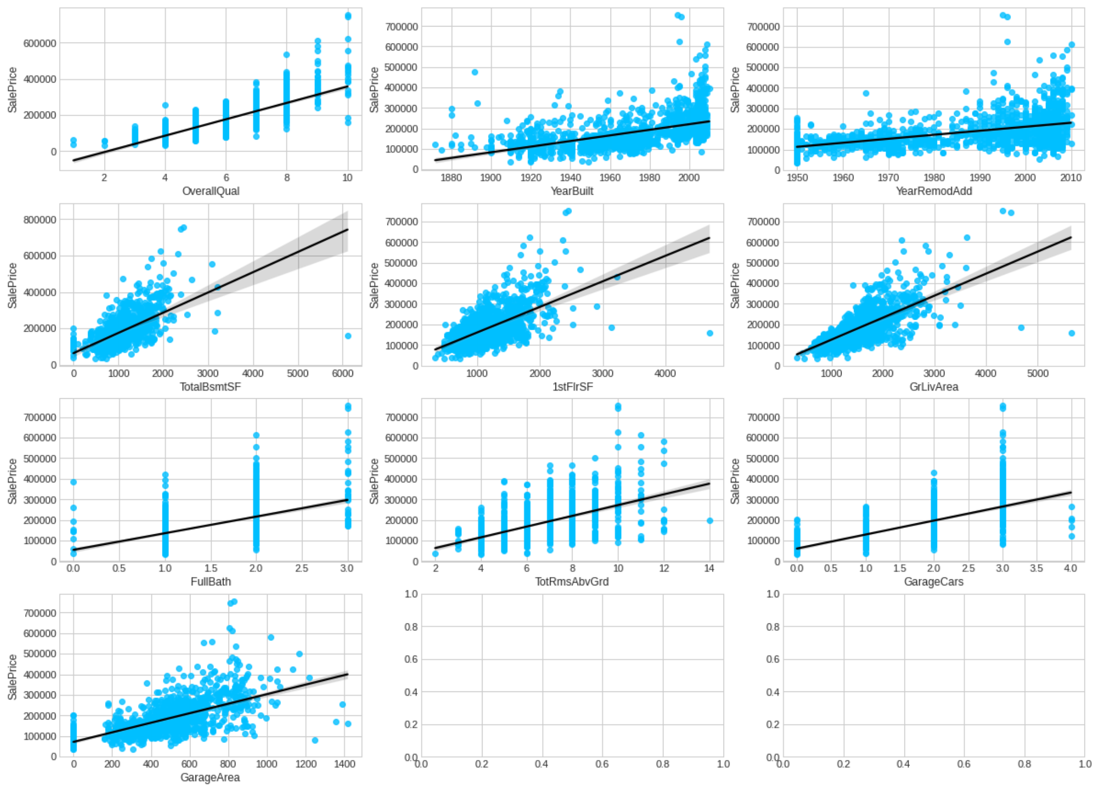


Figure 4: Features with high correlation (higher than 0.5)

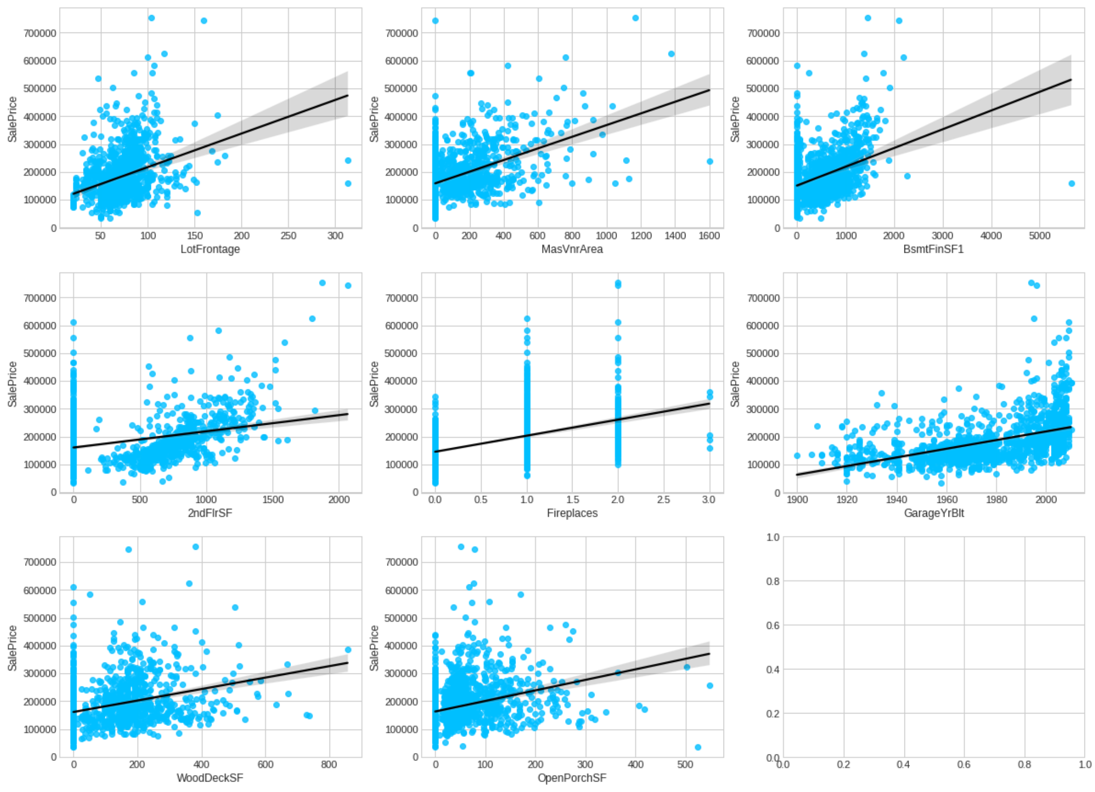


Figure 5: Features with low correlation (between 0.3 and 0.5)

Looking at some of these scatter plots related to the surface area of certain rooms in the house ("1stFlrSF", "TotalBsmtSF", "GrLivingArea"), there is at least one house whose price is relatively inexpensive given their surface area (located on the bottom right-hand-side of those graphs) which suggests the presence of outliers in the dataset.

List of features to be kept in the dataset: ['OverallQual', 'YearBuilt', 'YearRemodAdd', 'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'FullBath', 'TotRmsAbvGrd', 'GarageCars', 'GarageArea', 'LotFrontage', 'MasVnrArea', 'BsmtFinSF1', '2ndFlrSF', 'Fireplaces', 'GarageYrBlt', 'WoodDeckSF', 'OpenPorchSF', 'SalePrice']

Looking at the other correlation values beyond "SalePrice", since the correlation between "GarageCars" and "GarageArea" is so high, "GarageCars" will be dropped from both datasets as this is not likely to add in any relevant new information with regards to the final price of the house.

After dropping "GarageCars" from the dataset, the shape of the dataset is now

Training set shape (Numerical features): (1460, 18)

Test set shape (Numerical features): (1459, 17)

Remember we are now operating with the numerical features.

Handling Missing Values: From the updated numerical features training set, we see that there are three columns with missing values, with "LotFrontage" having around 18% missing values. The missing values from these columns will be imputed where they will be replaced with the median value from each column.

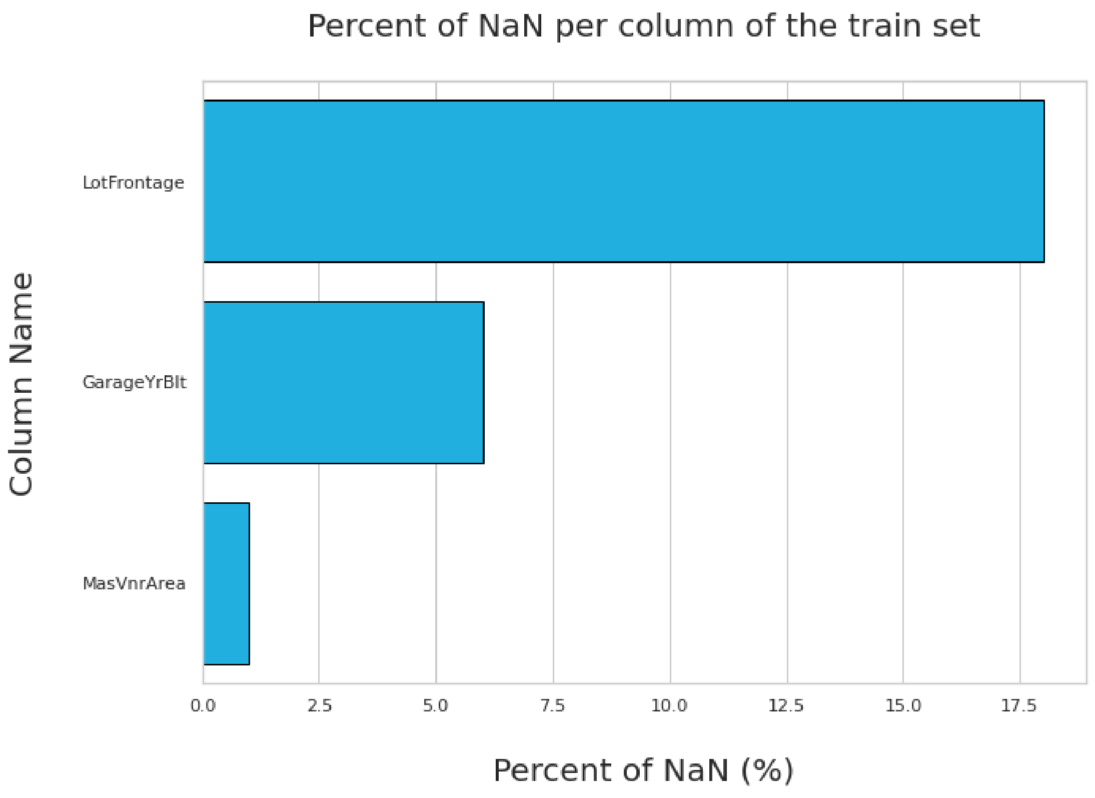


Figure 6: Names of columns with missing values (training set)

Imputation of missing values (NaNs) with SimpleImputer performed at this stage. Checking the distributions to ensure that the distribution of these variables after imputation is not heavily impacted by these changes.

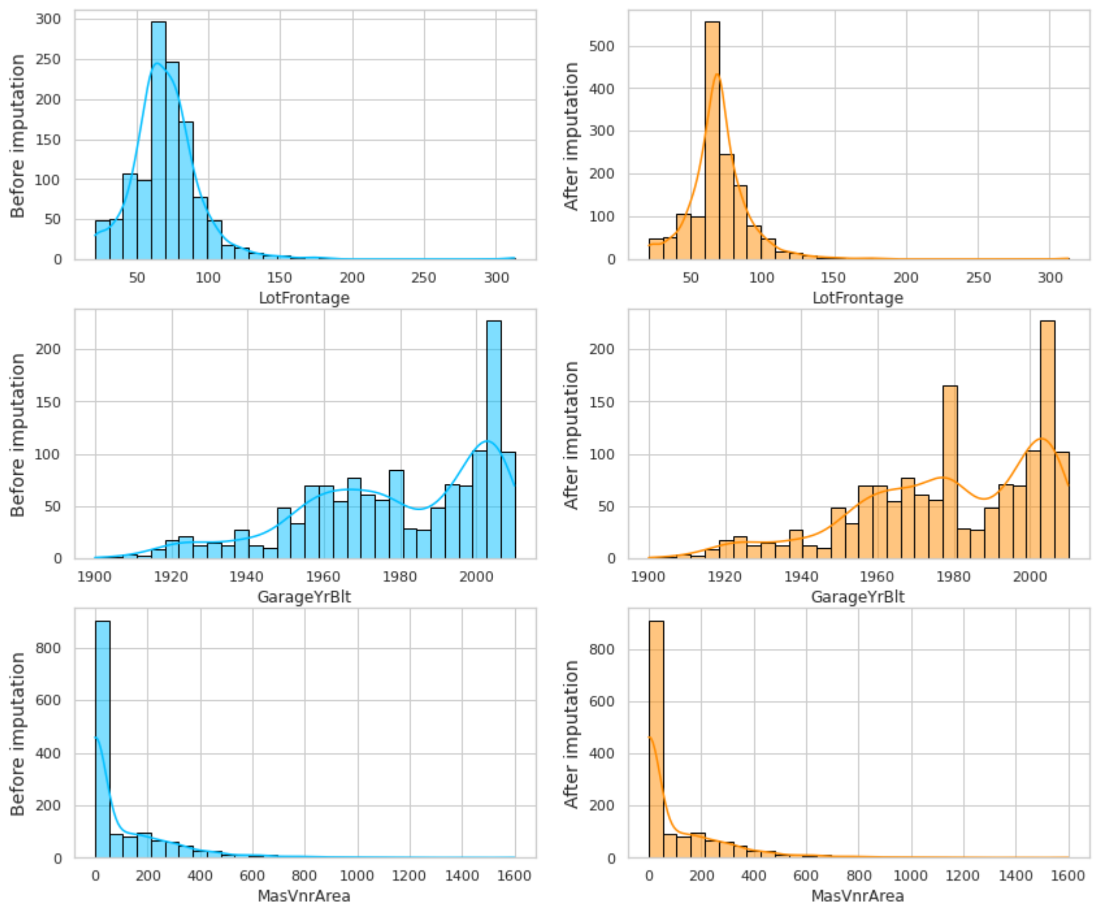


Figure 7: Distribution of each imputed feature before and after imputation

From distribution plots, it is clear that the distributions for "LotFrontage" and "GarageYrBlt" have been changed by imputation (with heavy bias towards the median for "LotFrontage"). Since there is noticeable bias towards the median class for both "LotFrontage" and "GarageYrBlt", these features will be removed from both datasets. The distribution of "MasVnrArea" has not changed that much at all after imputation, and will be kept on. Dropping these same features from test set also.Now we will deal with missing values from the test dataset.

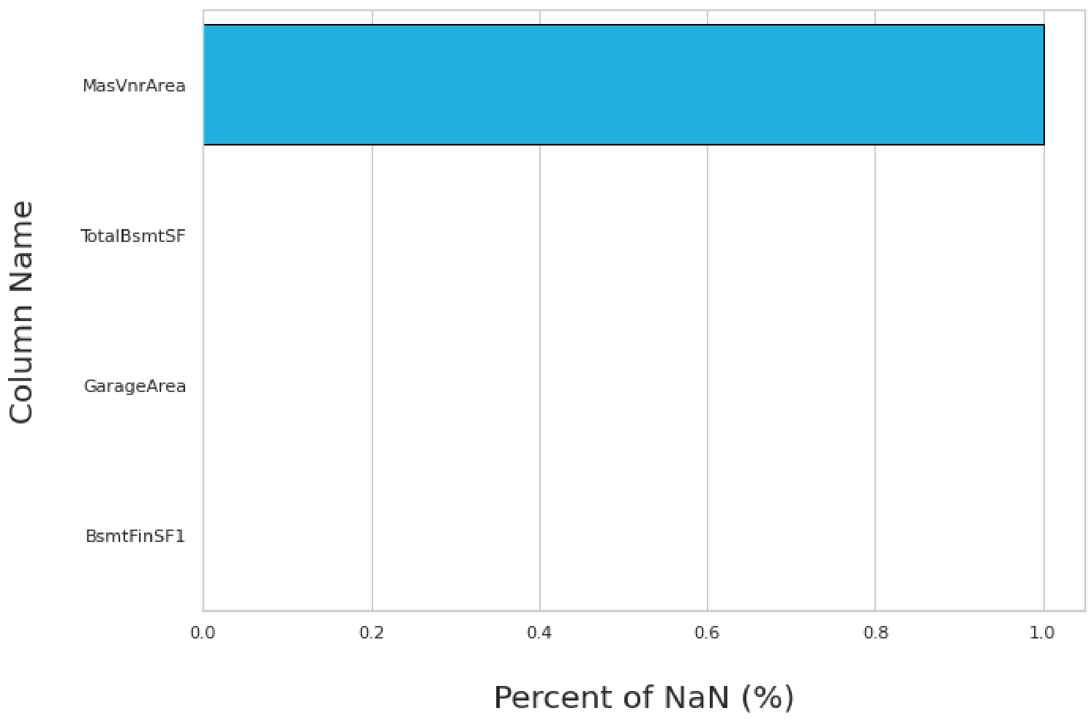


Figure 8: Names of columns with missing values (test set)

Here there are 4 feature variables that have missing values, but they all have a very small percentage of missing values. Each of these values will be filled in with the median from each perspective column.

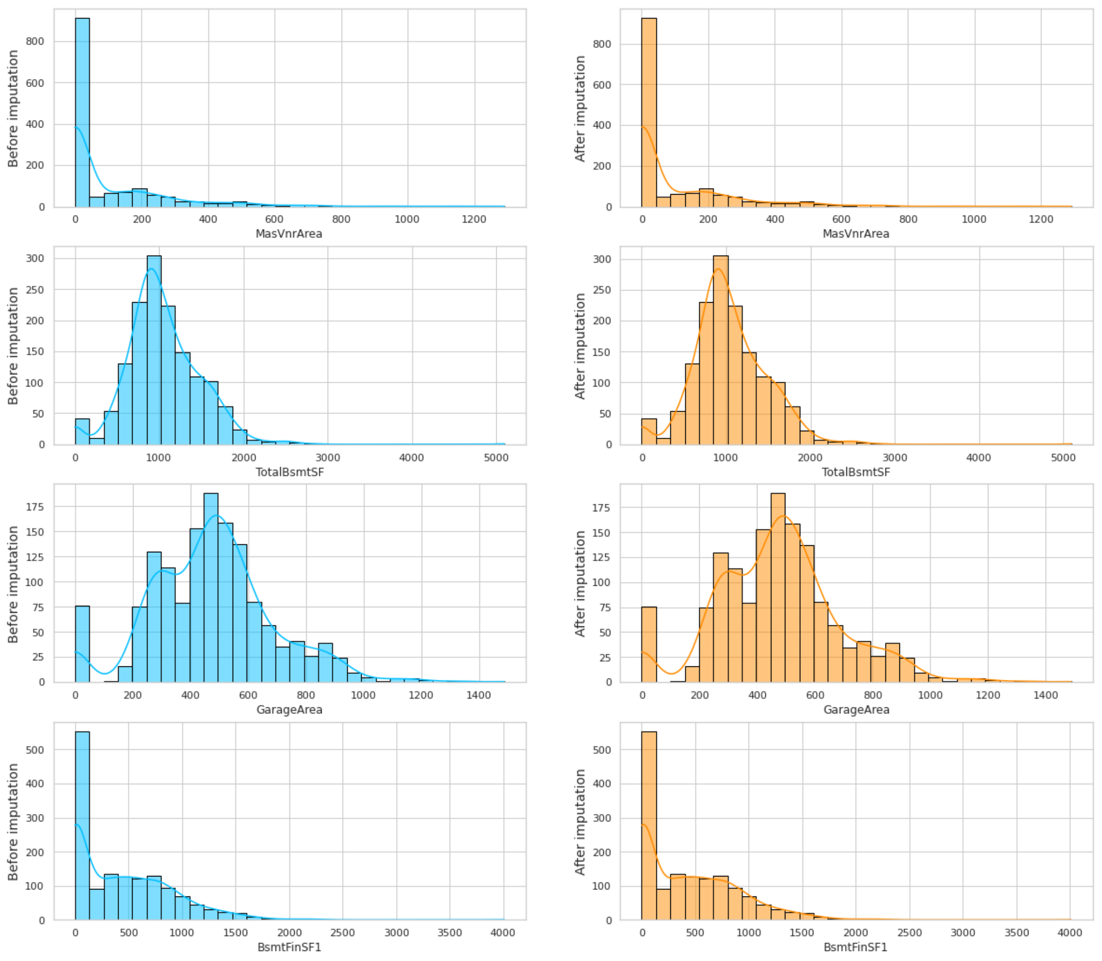


Figure 9: Checking the distribution of each imputed feature before and after imputation, test set.

The distribution of each of these variables before and after imputation does not really change at all, so all of these variables will be kept on.

Finally the shape of the Numerical Features

Training set shape (Numerical features): (1460, 16)
Test set shape (Numerical features): (1459, 15)

Operation with Categorical Features:

Firstly we separate the categorical features from both datasets into their own DataFrames data train categ and data test categ.

Training set shape (Categorical features): (1460, 44)

Test set shape (Categorical features): (1459, 43)

Now we need to determine which categorical features are dominated by one outcome. Countplot for each of the categorical features in the training set.

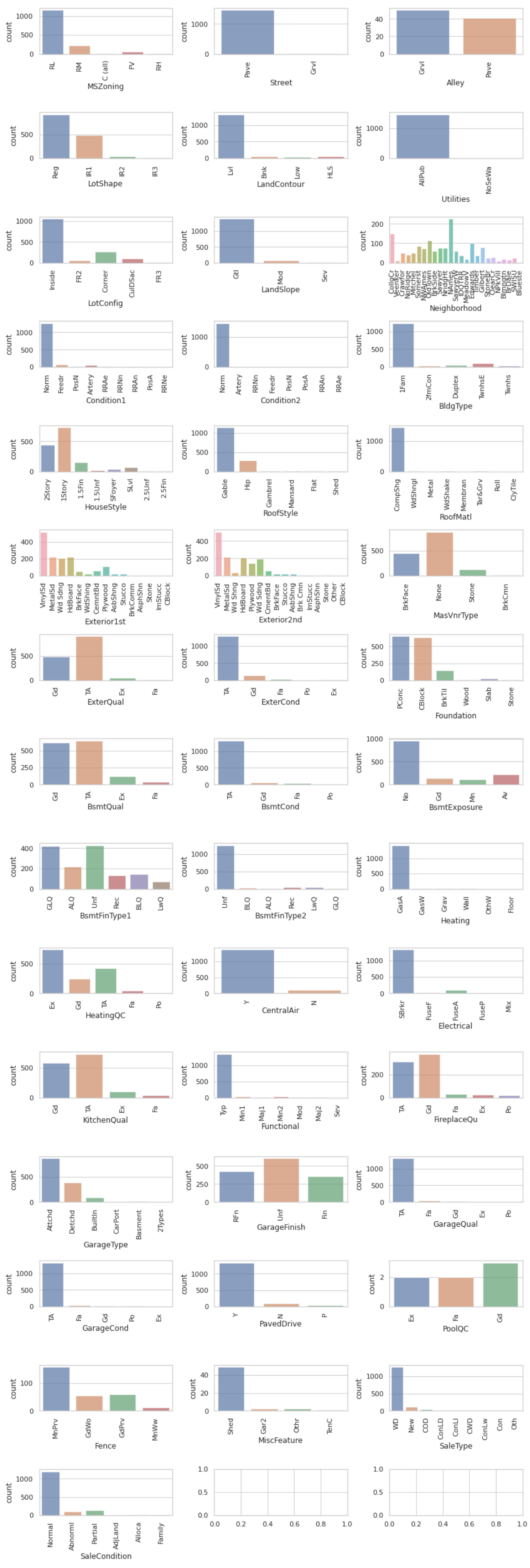


Figure 10: Countplot for categorical features (train set).

Looking at the Count Plots above, we see that some of these variables are highly dominated by one feature. Since those variables will have minimal impact on the final house prices, those variables dominated by one outcome will be removed from both dataset.

After Removing The Data Shape

Training set shape (Categorical features): (1460, 29)

Test set shape (Categorical features): (1459, 28)

Now With the boxplot we can see the variation of the target 'SalePrice' in each of the categorical features.

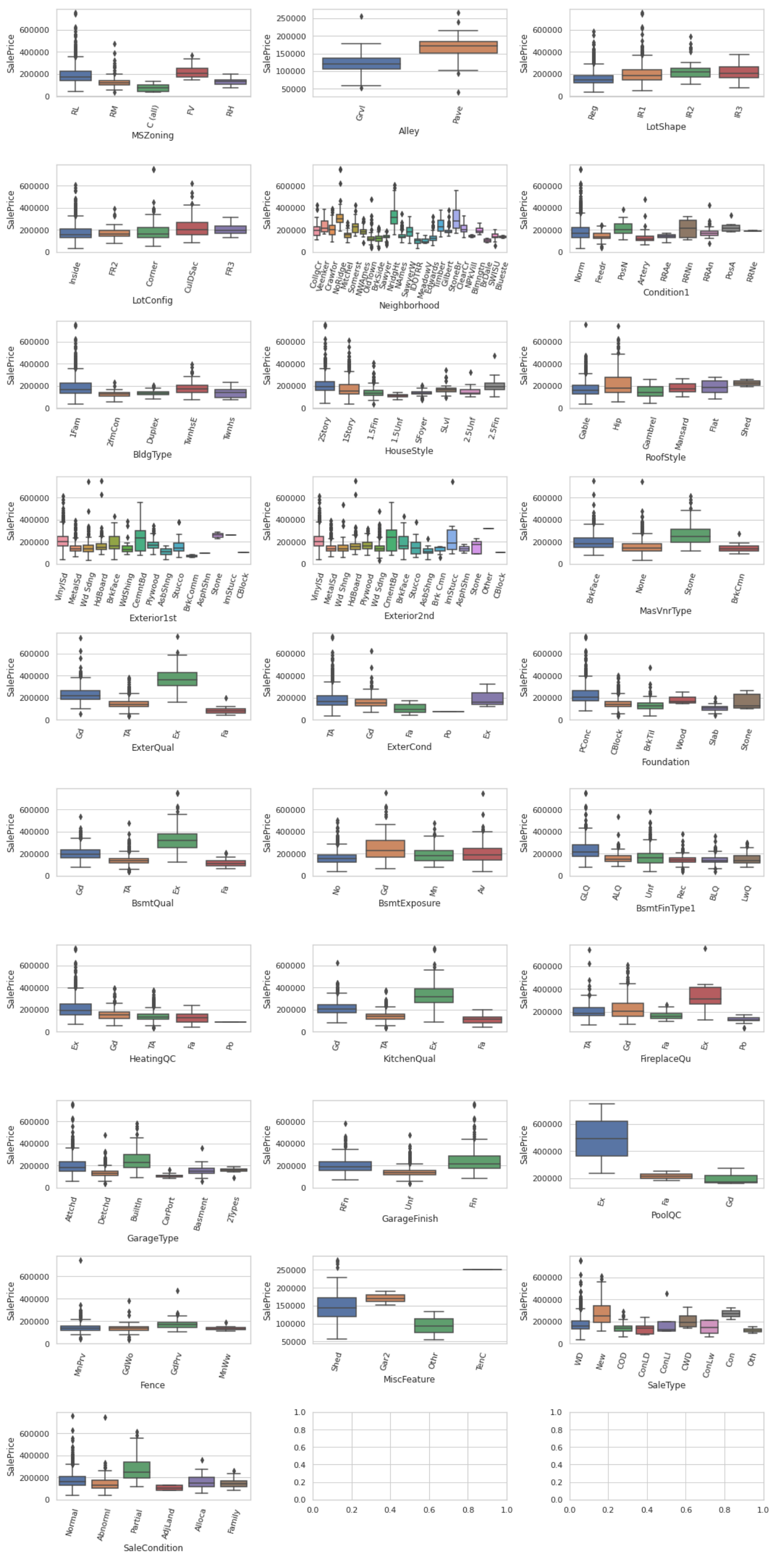
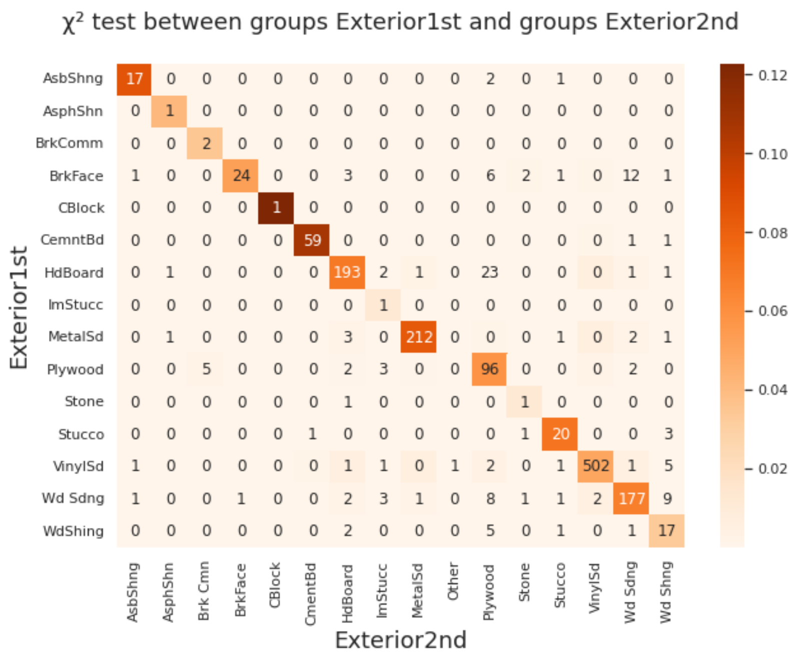


Figure 11: Variation of the target 'SalePrice'

Looking at these box plots, we see that the distribution of sale price for certain categorical variables are similar to each other (like "Exterior1st" and "Exterior2nd") which suggests that certain categorical variables are co-dependent on each other. There are three pairs of categorical variables for which the distribution of sale price is very similar

- "Exterior1st" and "Exterior2nd"
- "ExterQual" and "MasVnrType"
- "BsmtQual" and "BsmtExposure"

We will perform a Chi-squared test for each pair of variables at a 5% significance level to determine whether or not there is a strong dependency between these variables.



	AsbShng	AsphShn	Brk Cmn	BrkFace	CBlock	CmentBd	HdBoard	ImStucc	MetalSd	Other	Plywood	Stone	Stucco	VinylSd	Wd Sdng	Wd Shng
AsbShng	17	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0
AsphShn	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BrkComm	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
BrkFace	1	0	0	24	0	0	3	0	0	0	6	2	1	0	12	1
CBlock	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
CemntBd	0	0	0	0	0	59	0	0	0	0	0	0	0	0	1	1
HdBoard	0	1	0	0	0	0	193	2	1	0	23	0	0	0	1	1
ImStucc	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
MetalSd	0	1	0	0	0	0	3	0	212	0	0	0	1	0	2	1
Plywood	0	0	5	0	0	0	2	3	0	0	96	0	0	0	2	0
Stone	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
Stucco	0	0	0	0	0	1	0	0	0	0	0	1	20	0	0	3
VinylSd	1	0	0	0	0	0	1	1	0	1	2	0	1	502	1	5
Wd Sdng	1	0	0	1	0	0	2	3	1	0	8	1	1	2	177	9
WdShing	0	0	0	0	0	0	2	0	0	0	5	0	1	0	1	17

Figure 12: Chi-squared test "Exterior1st" and "Exterior2nd"

P-Value of the ChiSq Test bewteen Exterior1st and Exterior2nd is: 0.0
significance=0.050, p=0.000 , Dependent (reject H0)

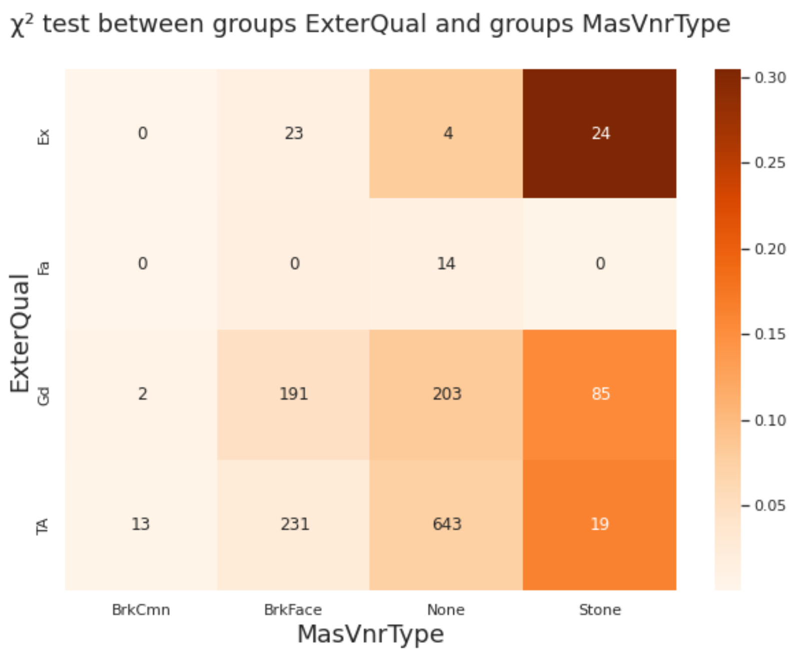


Figure 13: Chi-squared test "ExterQual" and "MasVnrType"

P-Value of the ChiSq Test bewteen ExterQual and MasVnrType is: 1.0187554679218715e-54

significance=0.050, p=0.000 , Dependent (reject H0)

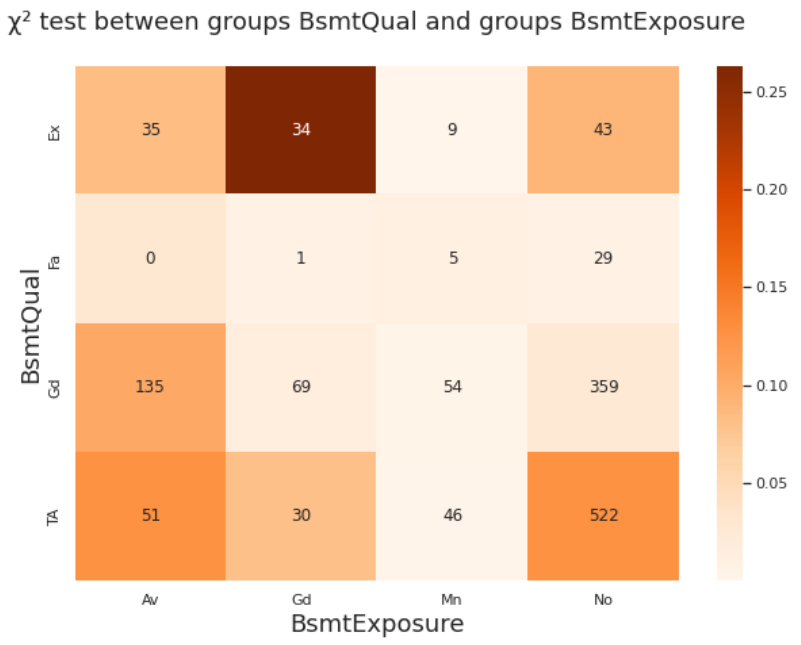


Figure 14: Chi-squared test "BsmtQual" and "BsmtExposure"

P-Value of the ChiSq Test bewteen BsmtQual and BsmtExposure is: 3.879215036512606e-32 significance=0.050, p=0.000 , Dependent (reject H0)

Here, the contingency tables for each pair of categorical variables considered are printed as a heatmap, with the colors representing the relative error between the actual value of the table with its expected value. After performing the Chi-squared test for each pairs of variables considered, we see that there is strong co-dependency for each of these variables. Since highly dependent/correlated variables do not add much relevant new information with regards to the value of the target variable, we will drop one of each co-dependent variable from the dataset.

Dataset Shape

Training set shape (Categorical features): (1460, 26)

Test set shape (Categorical features): (1459, 25)

Missing Values: In the training set, there are five categorical variables that have a significant amount of missing values. To help reduce the error, we will remove any columns with more than 30% NaN entries from both datasets. Imputation will be used to fill in the missing entries from the remaining columns in the training set using the modal class.

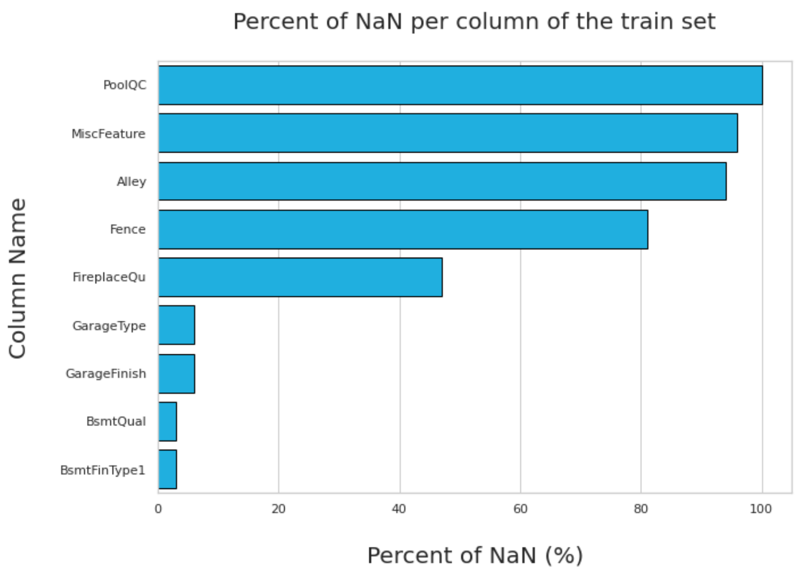


Figure 15: Categorical Columns with missing (NA) values in train data

There are a few more columns in the test dataset that have missing values, but none of them have more than 5

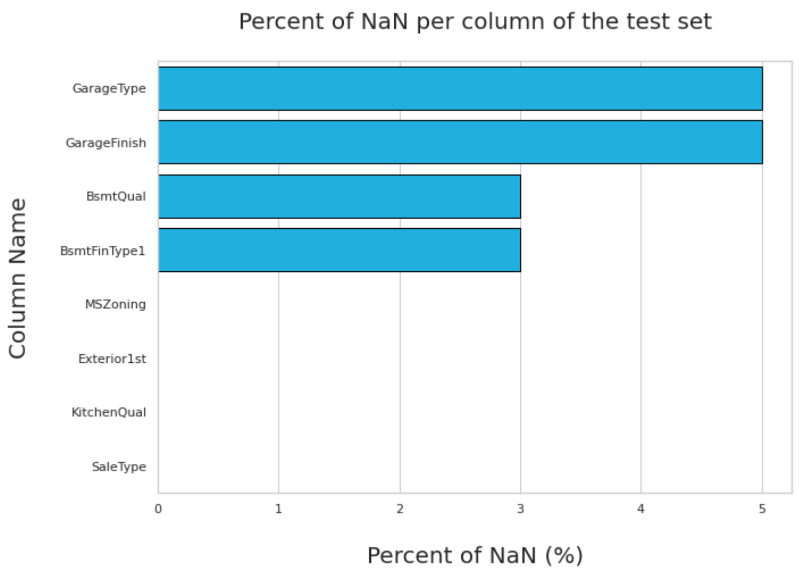


Figure 16: Categorical Columns with missing (NA) values in test data

Finally the shape of each dataset (containing categorical features only) after imputation is shown below.

Training set shape (Categorical features): (1460, 21)

Test set shape (Categorical features): (1459, 20)

Transformation into Numerical Values: Before we combine the categorical data back with the numerical data, we need to transform the categorical entries into numerical entries. This will be done using panda's the get dummies() function where each categorical feature will be transformed into a binary feature.

But after using get dummies ['HouseStyle 2.5Fin', 'Exterior1st ImStucc', 'Exterior1st Stone'] these three features were missing in the test dataset. Thus they will be dropped from the training dataset to ensure that both datasets have exactly the same features.

Now,

Training set shape (Categorical features): (1460, 137)

Test set shape (Categorical features): (1459, 137)

Before going for Model Building, it's time to join numerical and categorical data together.

So finally
Train set: (1460, 153)
Test set: (1459, 152)

The Year of construction and the Year of Remodelling variables will be transformed into new variables representing the Age of the House and the Age since the house was remodelled - this will enable us to apply a log transform to normalize those variables. After the transformation, the variables ”YearBuilt” and ”YearRemodAdd” will be removed.

A Log transformation will be applied to the skewed numerical variables to help mitigate the strong variation of some variables, and to reduce redundancy.

To obtain the skewed features, we take out variables that are more than 50% skewed. A log transformation will be applied to ”SalePrice” as well.

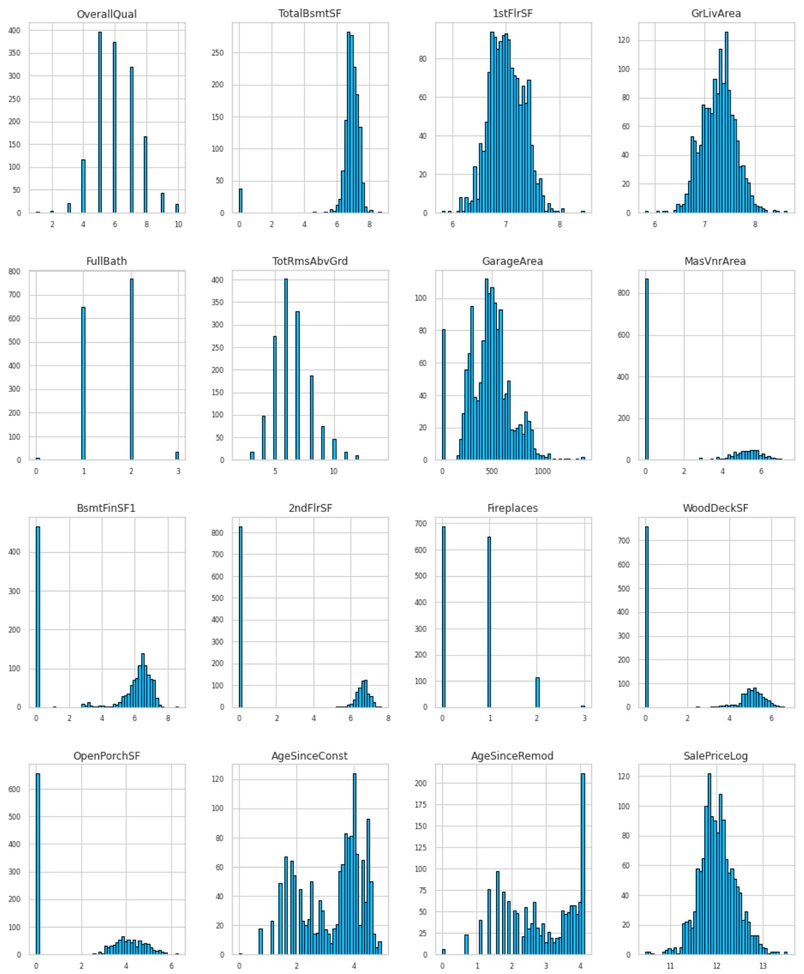


Figure 17: The distribution of all the numerical features

Looking at the distribution of the numerical features, we notice that most of the previously skewed variables have a more normal distribution (excluding zero values) with exception of the Age variables, which should result in better predictions.

4.4 Model Building

For building the model, X will denote as Feature variable and y will denote as Target Variable.

Splitting the data into training and test sets to configure model.
X train, X test, y train, y test = train test split(X, y, test size = 0.2, random

state = 42)

For this project, three supervised learning models will be considered:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor

To measure model performance and their predicitons the RMSE and R square scores will be used.

Linear Regression: Linear Regression is a supervised machine learning model that attempts to model a linear relationship between dependent variables (Y) and independent variables (X). Every evaluated observation with a model, the target (Y)'s actual value is compared to the target (Y)'s predicted value, and the major differences in these values are called residuals.

Based on the processed data, the linear regression model generated

R^2: 0.906023717952908, RMSE: 0.13242826218245224

Decision Tree: Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

Based on the processed data, the decision tree model generated

R^2: 0.7886620096370631, RMSE: 0.1985913540944503

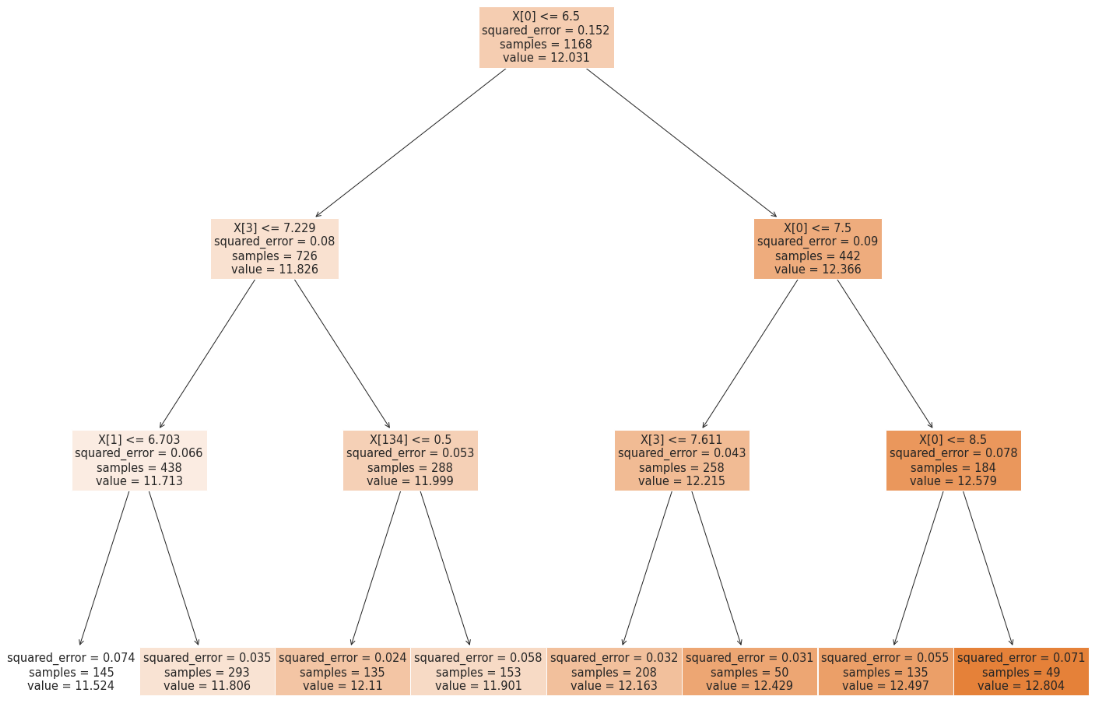


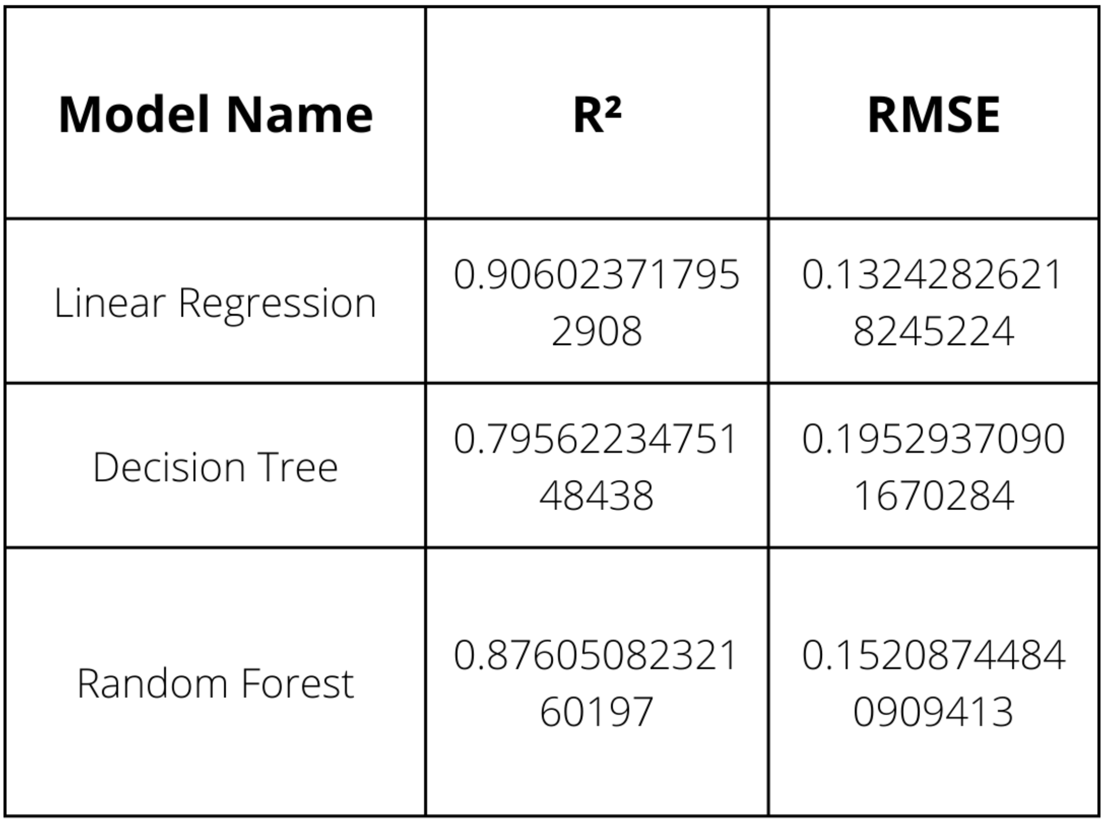
Figure 18: Decision Tree Visual

Random Forest: The bootstrapping Random Forest algorithm combines ensemble learning methods with the decision tree framework to create multiple randomly drawn decision trees from the data, averaging the results to output a new result that often leads to strong predictions/classifications.

Based on the processed data, the random forest model generated

R^2: 0.8800193164516512, RMSE: 0.1496329424917678

4.5 Model Evaluation



Model Name	R²	RMSE
Linear Regression	0.906023717952908	0.13242826218245224
Decision Tree	0.7956223475148438	0.19529370901670284
Random Forest	0.8760508232160197	0.15208744840909413

Figure 19: Model Evaluation

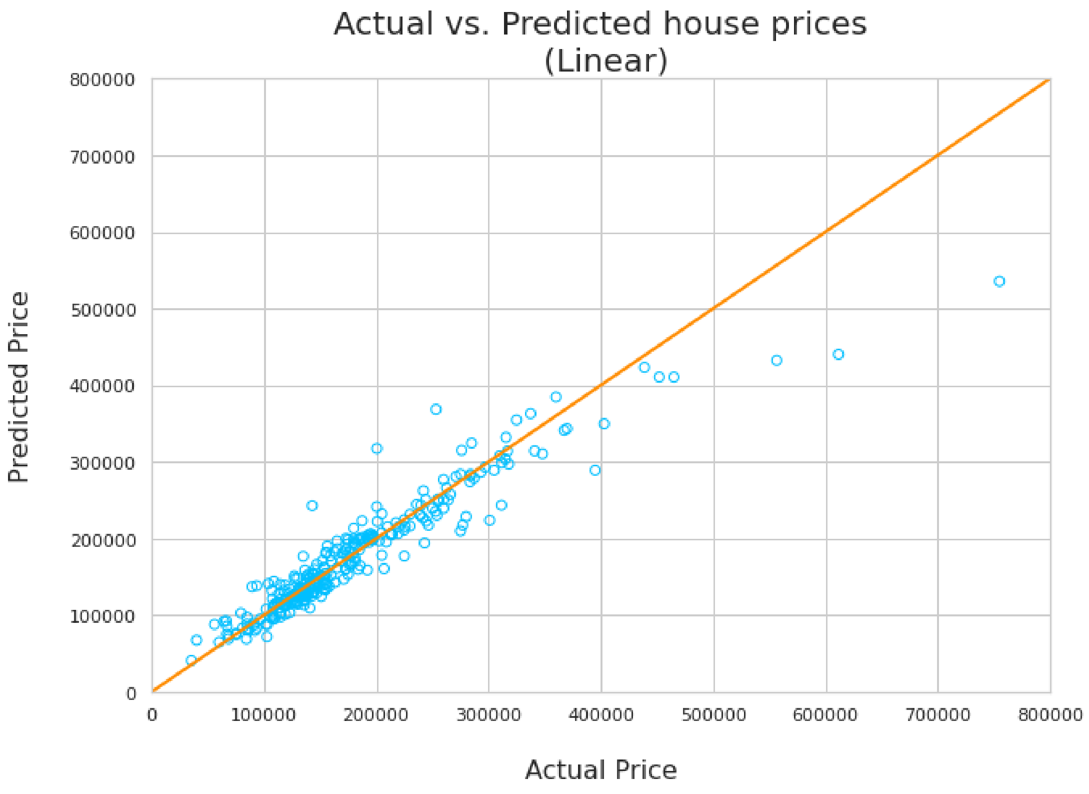


Figure 20: Actual vs Predicted House Prices (Linear Regression)

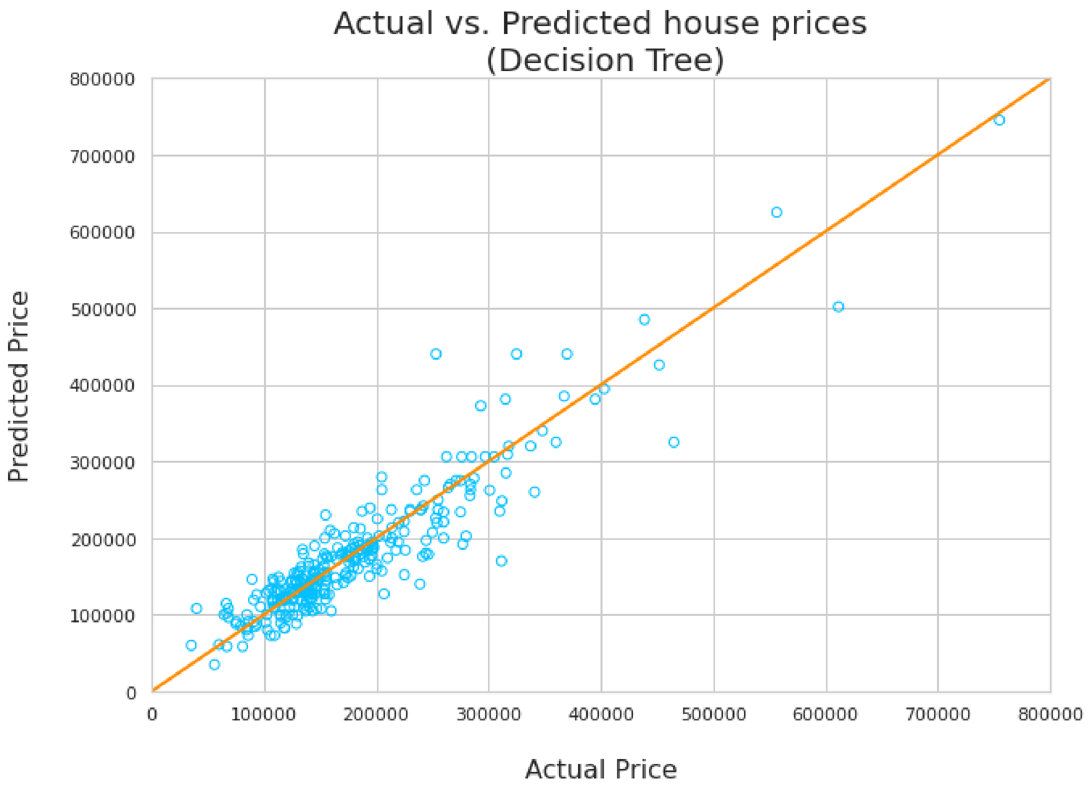


Figure 21: Actual vs Predicted House Prices (Decision Tree)

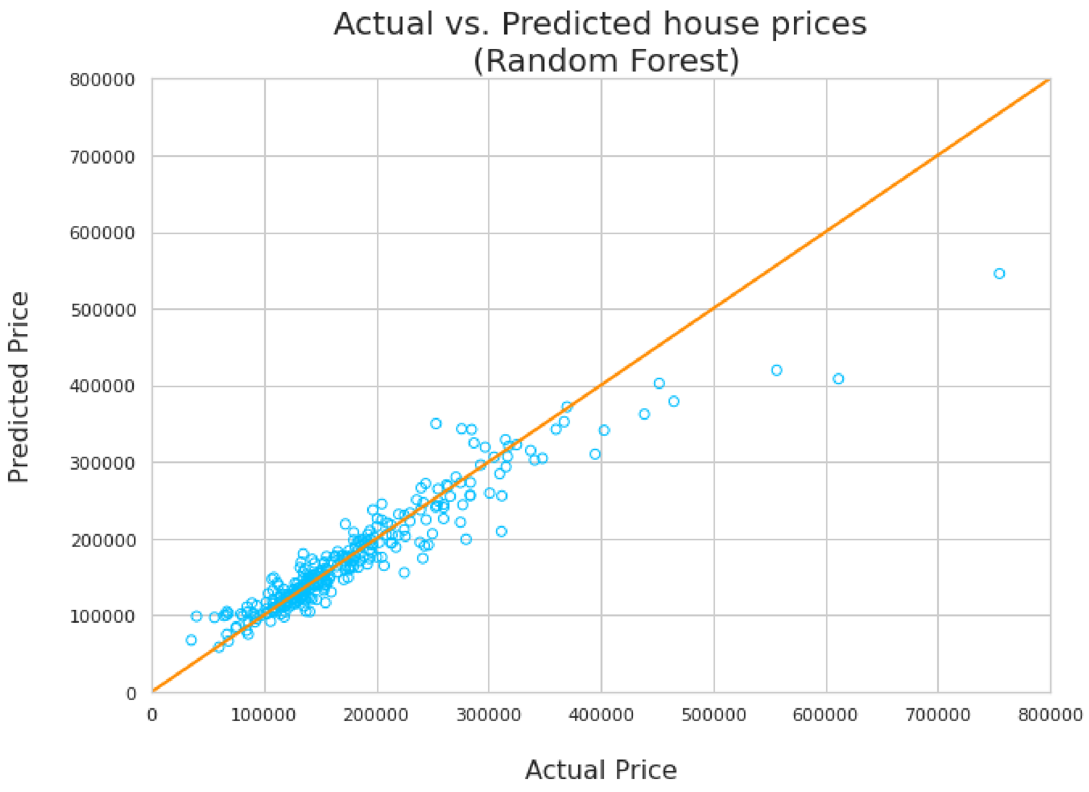


Figure 22: Actual vs Predicted House Prices (Random Forest)

The RMSE value of Linear Regression Model is smaller.

5 Discussion

While removing features with a high degree of skewness may reduce noise in the model, it is preferable to error on the side of caution of too many features than too few, and to enable the models themselves to attenuate overfitting via hyperparameter tuning.