FLIP ROBO

SURPRISE HOUSING DATA: PRICE PREDICTION

Submitted by:

SANTOSH H. HULBUTTI

ACKNOWLEDGMENT

This project is completed using knowledge/information available on internet.

Following are the websites & YouTube Channels, which were used to understand concepts related to ML, AI & Data Visualization.

Websites:

1. towardsdatascience.com
2. medium.com
3. analyticsvidya.com
4. DataTrained LMS Platform
5. Official documentation of ScikitLearn, Matplot library, Pandas Library & Seaborn library.
6. Kaggle.com
7. UCI ML Repository
8. Youtube Channels:
    a. Krish Naik
    b. Sidhdhardan
    c. Keith Galli

I would like to thank FlipRobo Technologies, for giving an opportunity to work as an intern during this project period. And also like to thank mentor Ms. Gulshana Chaudhary for assigning the project.

INTRODUCTION

• Business Problem Framing

Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world’s economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

• Conceptual Background of the Domain Problem

A US-based housing company named **Surprise Housing** has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

Which variables are important to predict the price of house?

How do these variables describe the price of the house?

• Review of Literature

Feature Selection - To avoid the curse of dimensionality, and also to avoid overfitting and under filling we should select features which are very important to the data. All of the features we find in the dataset might not be useful in building a machine learning model to make the necessary prediction. Using some of the features might even make the predictions worse. So, feature selection plays a huge role in building a machine learning model. I learned various methods to select the appropriate features: Variance, P-Value, Correlation, Co-Independence, Visualization.

Handling of missing values, Removal of outliers & skewness plays very important role in as it manipulates a fine percentage of data. Feature scaling is done to get make feature normally distributed. Different algorithms are used to check best fit model for the given dataset. Data Visualization is done using Matplotlib & Seaborn.

• Motivation for the Problem Undertaken

It was required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

Missing values are imputed based on the Mode method, as most of the data was categorical in nature. Those features with numerical data are filled using knn regression method.

Features grouped based on data types vis., Numerical, continuous numerical, discrete numerical & categorical.

Following features are removed as more than 75% of the data was 0 as observed value: **'BsmtFinSF2', 'LowQualFinSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', & 'MiscVal'.**

Some numbers are converted into categories.

MSold is converted in to cyclical number data, as moths are cyclical in nature.

Skewness is removed using Yeo-Johnson Power Transformer.

Outliers are removed using Z-score method. Data loss observed was 5.89%.

Standard scaling is applied on the entire train & test data.

- ## Data Sources and their formats

The data was shared in the CSV format. It had 1460 entries & 81 columns.

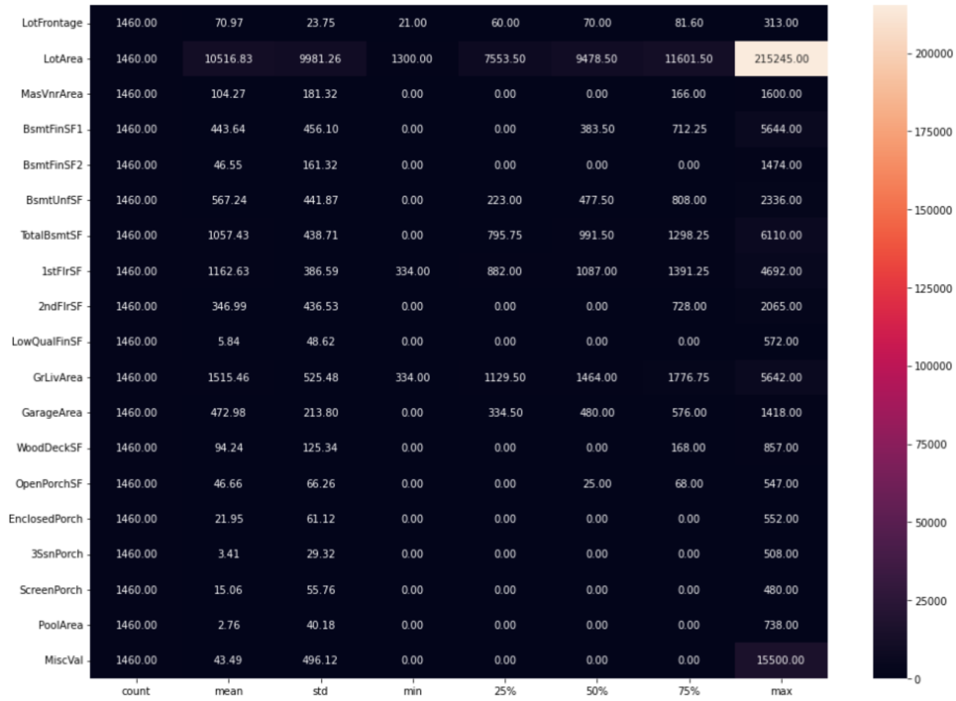
Following are the description of the columns/varibles in the data set

1. ***MSSubClass**: Identifies the type of dwelling involved in the sale.*
2. ***MSZoning**: Identifies the general zoning classification of the sale.*
3. ***LotFrontage**: Linear feet of street connected to property*
4. ***LotArea**: Lot size in square feet*
5. ***Street**: Type of road access to property*
6. ***Alley**: Type of alley access to property*
7. ***LotShape**: General shape of property*
8. ***LandContour**: Flatness of the property*
9. ***Utilities**: Type of utilities available*
10. ***LotConfig**: Lot configuration*
11. ***LandSlope**: Slope of property*
12. ***Neighborhood**: Physical locations within Ames city limits*
13. ***Condition1**: Proximity to various conditions*
14. ***Condition2**: Proximity to various conditions (if more than one is present)*
15. ***BldgType**: Type of dwelling*
16. ***HouseStyle**: Style of dwelling*
17. ***OverallQual**: Rates the overall material and finish of the house*
18. ***OverallCond**: Rates the overall condition of the house*
19. ***YearBuilt**: Original construction date*
20. ***YearRemodAdd**: Remodel date (same as construction date if no remodeling or additions)*
21. ***RoofStyle**: Type of roof*
22. ***RoofMatl**: Roof material*
23. ***Exterior1st**: Exterior covering on house*
24. ***Exterior2nd**: Exterior covering on house (if more than one material)*

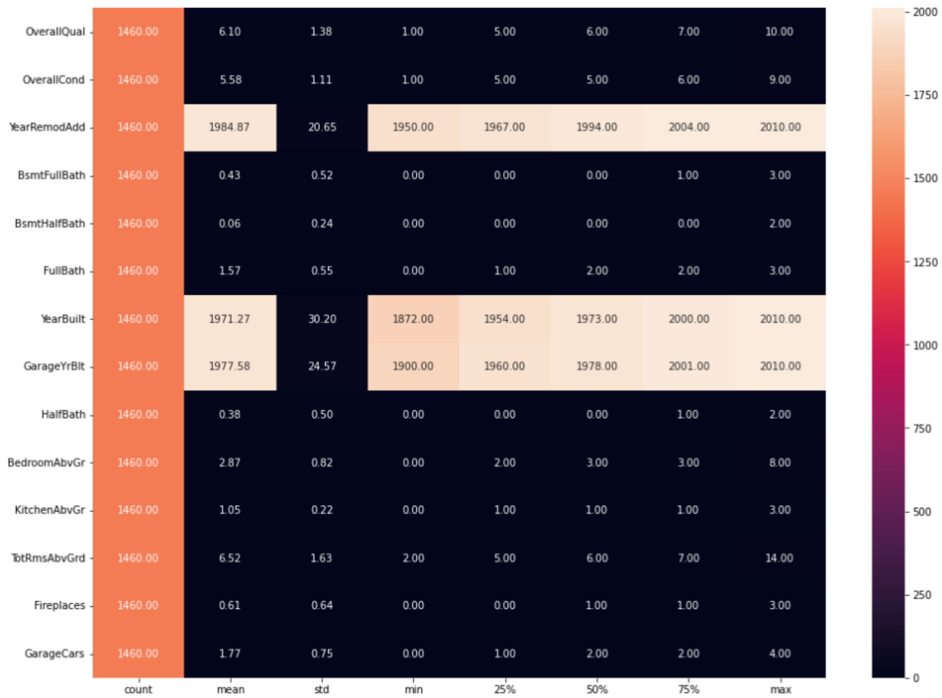
25. ***MasVnrType**: Masonry veneer type*
26. ***MasVnrArea**: Masonry veneer area in square feet*
27. ***ExterQual**: Evaluates the quality of the material on the exterior*
28. ***ExterCond**: Evaluates the present condition of the material on the exterior*
29. ***Foundation**: Type of foundation*
30. ***BsmtQual**: Evaluates the height of the basement*
31. ***BsmtCond**: Evaluates the general condition of the basement*
32. ***BsmtExposure**: Refers to walkout or garden level walls*
33. ***BsmtFinType1**: Rating of basement finished area*
34. ***BsmtFinSF1**: Type 1 finished square feet*
35. ***BsmtFinType2**: Rating of basement finished area (if multiple types)*
36. ***BsmtFinSF2**: Type 2 finished square feet*
37. ***BsmtUnfSF**: Unfinished square feet of basement area*
38. ***TotalBsmtSF**: Total square feet of basement area*
39. ***Heating**: Type of heating*
40. ***HeatingQC**: Heating quality and condition*
41. ***CentralAir**: Central air conditioning*
42. ***Electrical**: Electrical system*
43. ***1stFlrSF**: First Floor square feet*
44. ***2ndFlrSF**: Second floor square feet*
45. ***LowQualFinSF**: Low quality finished square feet (all floors)*
46. ***GrLivArea**: Above grade (ground) living area square feet*
47. ***BsmtFullBath**: Basement full bathrooms*
48. ***BsmtHalfBath**: Basement half bathrooms*
49. ***FullBath**: Full bathrooms above grade*
50. ***HalfBath**: Half baths above grade*
51. ***Bedroom**: Bedrooms above grade (does NOT include basement bedrooms)*
52. ***Kitchen**: Kitchens above grade*
53. ***KitchenQual**: Kitchen quality*
54. ***TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)*
55. ***Functional**: Home functionality (Assume typical unless deductions are warranted)*
56. ***Fireplaces**: Number of fireplaces*
57. ***FireplaceQu**: Fireplace quality*
58. ***GarageType**: Garage location*
59. ***GarageYrBlt**: Year garage was built*
60. ***GarageFinish**: Interior finish of the garage*
61. ***GarageCars**: Size of garage in car capacity*
62. ***GarageArea**: Size of garage in square feet*
63. ***GarageQual**: Garage quality*
64. ***GarageCond**: Garage condition*
65. ***PavedDrive**: Paved driveway*
66. ***WoodDeckSF**: Wood deck area in square feet*
67. ***OpenPorchSF**: Open porch area in square feet*
68. ***EnclosedPorch**: Enclosed porch area in square feet*
69. ***3SsnPorch**: Three season porch area in square feet*
70. ***ScreenPorch**: Screen porch area in square feet*
71. ***PoolArea**: Pool area in square feet*
72. ***PoolQC**: Pool quality*
73. ***Fence**: Fence quality*
74. ***MiscFeature**: Miscellaneous feature not covered in other categories*
75. ***MiscVal**: $Value of miscellaneous feature*

76. ***MoSold***: *Month Sold (MM)*
77. ***YrSold***: *Year Sold (YYYY)*
78. ***SaleType***: *Type of sale*
79. ***SaleCondition***: *Condition of sale*
80. ***Id:*** *Id of House*
81. ***SalePrice*** *: Price of House*

```
plt.figure(figsize=(16,12))
sns.heatmap(data3[num_cont_cols].describe().T , annot =True, fmt = ".2f");
```




```
plt.figure(figsize=(16,12))
sns.heatmap(data3[num_desc_cols].describe().T , annot =True, fmt = ".2f");
```

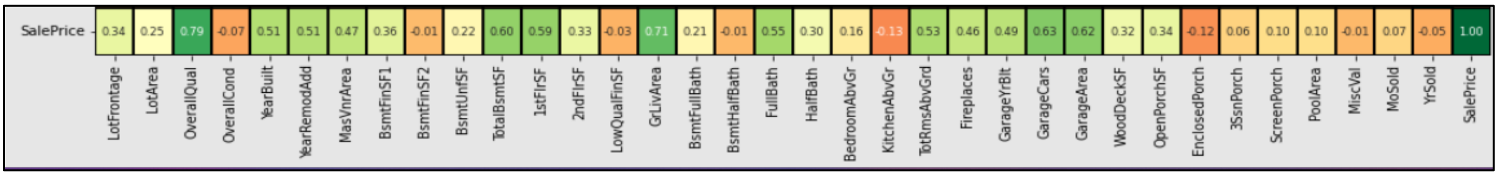


- Data Preprocessing Done

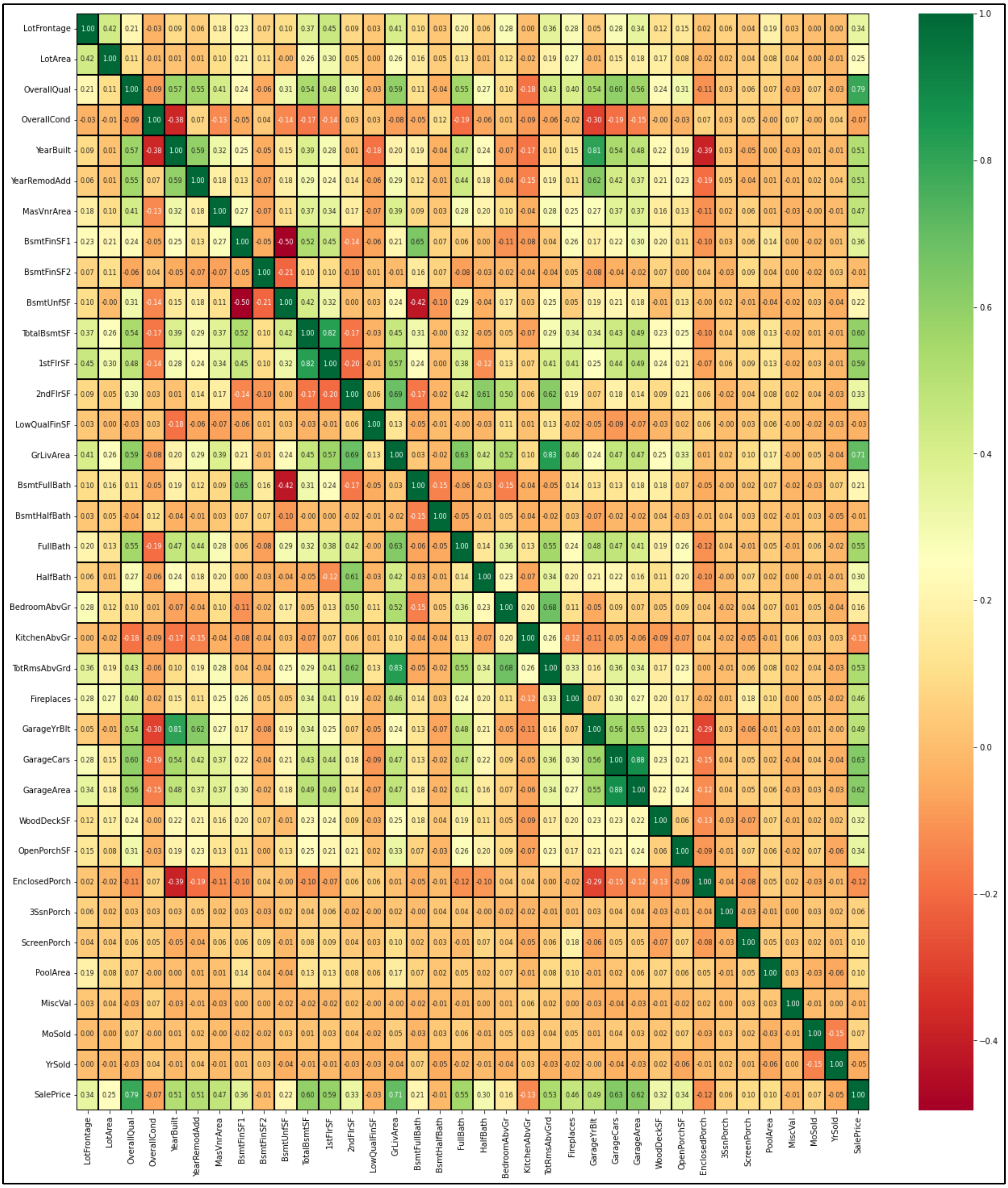
1. Data was given in two files, 1. Train set & 2. Test set. It was combined for Data Preprocessing.
2. Statistical summary of data was checked.
3. Missing values are imputed based on mode methos for categorical features & knn method for numerical features.
4. Features grouped based on data types vis., Numerical, continuous numerical, discrete numerical & categorical.
5. Following features are removed as more than 75% of the data was 0 as observed value: **'BsmtFinSF2', 'LowQualFinSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', & 'MiscVal'.**
6. ***MSSubClass*** numbers are converted into categories.
7. **MSold** is converted in to cyclical number data, as moths are cyclical in nature.
8. Skewness is removed using Yeo-Johnson Power Transformer.
9. Outliers are removed using Z-score method. Data loss observed was 5.89%.
10. Multicoliinearity checked using seaborn heat map & VIF. ***GrLivArea*** is removed from dataset after checking multicollinearity.
11. Categorical features are encoded using pd.get_dummies() method.
12. Standard scaling is applied on the entire train & test data.
13. We used train_test_split to split data for machine learning.

- # Data Inputs- Logic- Output Relationships

Following heat map shows the relation between numerical features and target variable ‘SalePrice’, using correlation coeffient.



Following heat map shows the relation between independent numerical features with each other & target variable using correlation coeffient.



- Hardware and Software Requirements and Tools Used

  a. Software

  i. ➔ Jupyter Notebook (Python 3.9)
  ii. ➔ Microsoft Office
  iii. ➔ Tableau

  b. Hardware

  i. ➔ Processor – AMD Ryzen 5
  ii. ➔ RAM - 8 GB
  iii. ➔ Graphic Memory - 4Gb , Nvidia GEFORCE RTX1650

  c. Python Libraries

  i. ➔ Pandas
  ii. ➔ Numpy
  iii. ➔ Matplotlib
  iv. ➔ Seaborn
  v. ➔ Scipy
  vi. ➔ Sklearn

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

The data set was was analysed both statistically and graphically. The statistical analysis showed that

a. data has outliers, skewness, null values & zero values
b. datas independent variable had numerical datas and categorical datas
c. The null values were replaced with mode & Knn method.
d. Outliers were removed using z-score method.. about 5.24% of data removed.
e. Skewness of some columns were transformed using yeo-Johnson method to have within allowed limits of +/-0.5.
f. Some features were dropped as the entries were 0 for more than 75% of the features.
g. Encoding was done using pd.get_dummies() method.

- ## Testing of Identified Approaches (Algorithms)

```python
#For Regression model
from sklearn.linear_model import LinearRegression, Ridge, Lasso, LassoCV
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor

#For Evaluation metrics for regression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

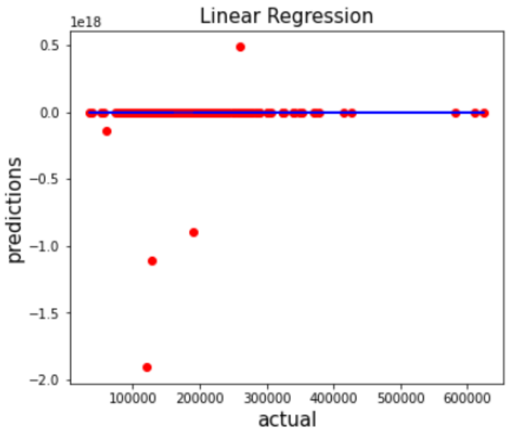
- ## Testing of Identified Approaches (Algorithms)

```python
lr = LinearRegression()
ls = Lasso()
rd = Ridge()
rfr = RandomForestRegressor()
abr = AdaBoostRegressor()
gbr = GradientBoostingRegressor()
dtr = DecisionTreeRegressor()
svr = SVR()
knr = KNeighborsRegressor()
xgb = XGBRegressor()
```

```python
models = [lr, ls, rd, abr, gbr, dtr,svr, knr, xgb, rfr]
models_name = ['Linear Regression', 'Lasso', 'Ridge',
               'Ada-Boost Regressor', 'Gradient Boosting Regressor',
               'Decision Tree Regressor','Support Vector Machine', 'KNeighbors Regressor', 'XGB Regressor', 'Random Forest
dummy_count = 0 #dummy variable for count purpose
for model in models:
    diff = []
    randomstate = []
    for i in range(0, 100):  ### loop to find best random state for splitting
        x_train, x_test, y_train, y_test = train_test_split(X, y_reg, test_size = 0.25, random_state = i)
        model.fit(x_train, y_train)
        pred_train = model.predict(x_train)
        pred_test = model.predict(x_test)
        diff.append(abs(r2_score(y_train, pred_train) - r2_score(y_test, pred_test)))
        randomstate.append(i)

    best_i = randomstate[diff.index(min(diff))]
    rs.append(best_i)

    ### splitting the train7 set with best random state
    x_train, x_test, y_train, y_test = train_test_split(X, y_reg, random_state=best_i, test_size=.25)
```

```python
    ##showing the results in output
    print('::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::')
    print(' ')
    print(' ')
    print(models_name[dummy_count] + ' Model')
    print(' ')
    print('for '+ models_name[dummy_count] + ' model, Best Random_state number for splitting the data is: ', best_i)
    print(' ')
    print('===scores for training set===')
    print('r2 score for training set', r2_score(y_train, pred_train))
    print('MAE for training set: ', mean_absolute_error(y_train, pred_train))
    print('MSE for training set: ', mean_squared_error(y_train, pred_train))
    print('SMSE for training set: ', np.sqrt(mean_squared_error(y_train, pred_train)))
    print(' ')
    print('===scores for testing set===')
    print('r2 score for testing set : ', r2_score(y_test, pred_test))
    print('MAE for testing set: ', mean_absolute_error(y_test, pred_test))
    print('MSE for testing set: ', mean_squared_error(y_test, pred_test))
    print('SMSE for testing set: ', np.sqrt(mean_squared_error(y_test, pred_test)))
    print(' ')
    print(' ')

    ##plotting the graph with bestfit line, actual & predicted values
    plt.figure(figsize = (6,5))
    plt.scatter(x = y_test, y=pred_test, color = 'r')
    plt.plot(y_test, y_test, color = 'b')
    plt.xlabel('actual', fontsize = 15)
    plt.ylabel('predictions', fontsize = 15)
    plt.title(models_name[dummy_count], fontsize = 15)
    plt.show()
```

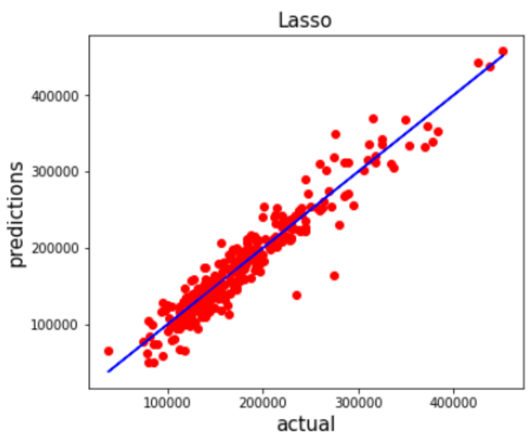
```
Linear Regression Model

for Linear Regression model, Best Random_state number for splitting the data is:  88

===scores for training set===
r2 score for training set 0.9385875877731483
MAE for training set:  12779.545454545454
MSE for training set:  309265588.14181817
SMSE for training set:  17585.94859943069

===scores for testing set===
r2 score for testing set :  -3.344017941116785e+24
MAE for testing set:  1.6512782940733286e+16
MSE for testing set:  2.1563723536941768e+34
SMSE for testing set:  1.4684591767203392e+17
```




```
Cross Validation score at best cv=5 is : -5226999404941303292272574464.00%
```

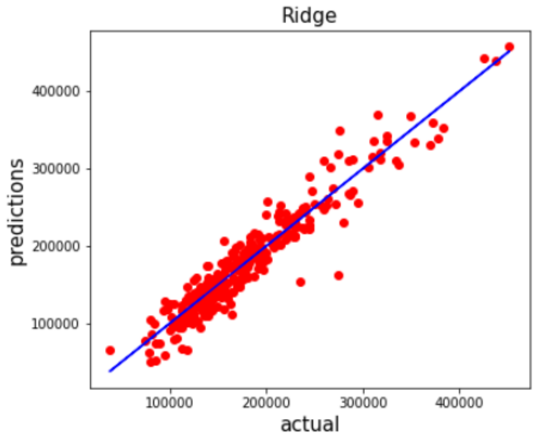
-

```
Lasso Model

for Lasso model, Best Random_state number for splitting the data is:  15

===scores for training set===
r2 score for training set 0.9267570312530975
MAE for training set:  13311.063256508824
MSE for training set:  410680204.28371346
SMSE for training set:  20265.246218186283

===scores for testing set===
r2 score for testing set :  0.8992135442426973
MAE for testing set:  16460.19633119665
MSE for testing set:  477805264.2436619
SMSE for testing set:  21858.75715230996
```




```
Cross Validation score at best cv=10 is : 86.17%
```

```
Ridge Model

for Ridge model, Best Random_state number for splitting the data is:  15

===scores for training set===
r2 score for training set 0.9267240132920888
MAE for training set:  13326.044343560221
MSE for training set:  410865339.09192896
SMSE for training set:  20269.813494256156

===scores for testing set===
r2 score for testing set :  0.9010570473918058
MAE for testing set:  16405.429290345983
MSE for testing set:  469065642.40980226
SMSE for testing set:  21657.92331710966
```

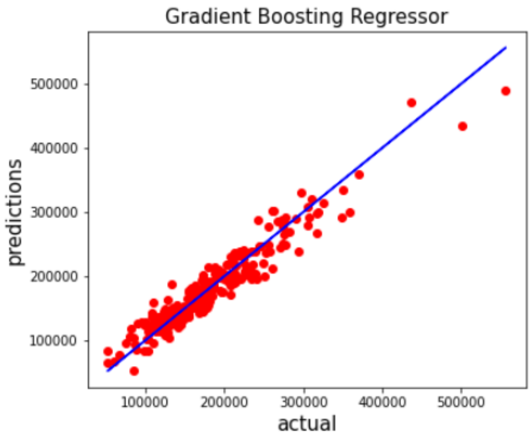



```
Cross Validation score at best cv=10 is : 86.28%
```

```
Ada-Boost Regressor Model

for Ada-Boost Regressor model, Best Random_state number for splitting the data is:  68

===scores for training set===
r2 score for training set 0.8666845550249671
MAE for training set:  20481.744050389087
MSE for training set:  675455845.4832253
SMSE for training set:  25989.533383330017

===scores for testing set===
r2 score for testing set :  0.8490217028214108
MAE for testing set:  22350.07426185275
MSE for testing set:  947386986.8303813
SMSE for testing set:  30779.652155772998
```

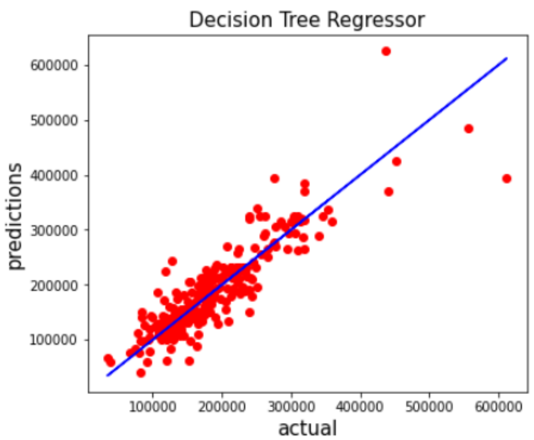

```
Cross Validation score at best cv=4 is : 79.48%
```

```
Gradient Boosting Regressor Model

for Gradient Boosting Regressor model, Best Random_state number for splitting the data is:  2

===scores for training set===
r2 score for training set 0.9720793935846473
MAE for training set:  9401.637559060135
MSE for training set:  157308437.94888383
SMSE for training set:  12542.26606115832

===scores for testing set===
r2 score for testing set :  0.9188833273066194
MAE for testing set:  14752.362195784086
MSE for testing set:  377143243.7625259
SMSE for testing set:  19420.176203179155
```



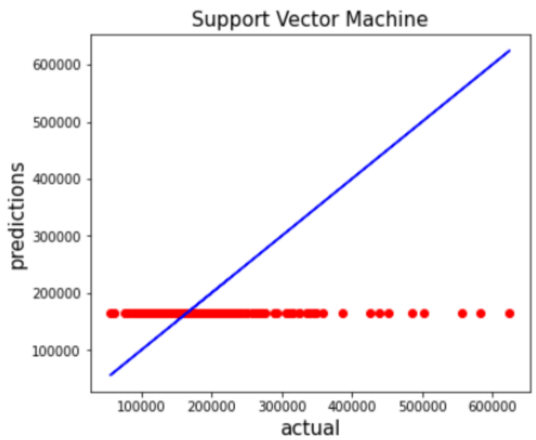

```
Cross Validation score at best cv=9 is : 88.15%
```

```
Decision Tree Regressor Model

for Decision Tree Regressor model, Best Random_state number for splitting the data is:  4

===scores for training set===
r2 score for training set 1.0
MAE for training set:  0.0
MSE for training set:  0.0
SMSE for training set:  0.0

===scores for testing set===
r2 score for testing set :  0.7718182046826112
MAE for testing set:  24491.52727272727
MSE for testing set:  1312868625.0181818
SMSE for testing set:  36233.52901689513
```



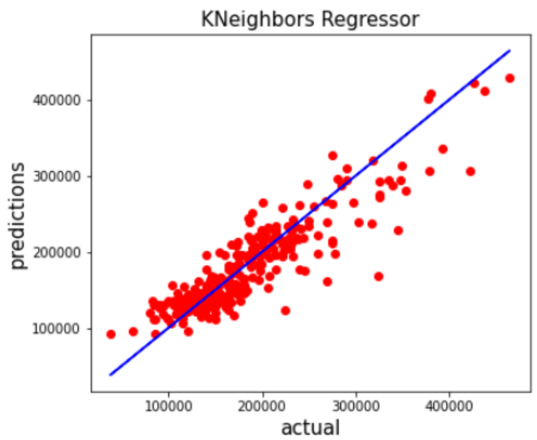

```
Cross Validation score at best cv=9 is : 71.34%
```

```
Support Vector Machine Model

for Support Vector Machine model, Best Random_state number for splitting the data is:  17

===scores for training set===
r2 score for training set -0.05090197714843714
MAE for training set:  52240.015941042795
MSE for training set:  5171790867.463673
SMSE for training set:  71915.1643776448

===scores for testing set===
r2 score for testing set :  -0.05069220062320534
MAE for testing set:  53682.12692041906
MSE for testing set:  7138021473.4767885
SMSE for testing set:  84486.81242345925
```




```
Cross Validation score at best cv=9 is : -5.29%
```

```
KNeighbors Regressor Model

for KNeighbors Regressor model, Best Random_state number for splitting the data is:  8

===scores for training set===
r2 score for training set 0.8044245510261278
MAE for training set:  21642.60606060606
MSE for training set:  1074527894.8282182
SMSE for training set:  32779.99229451127

===scores for testing set===
r2 score for testing set :  0.8036638584590726
MAE for testing set:  23063.272727272728
MSE for testing set:  996901320.937891
SMSE for testing set:  31573.744170400365
```

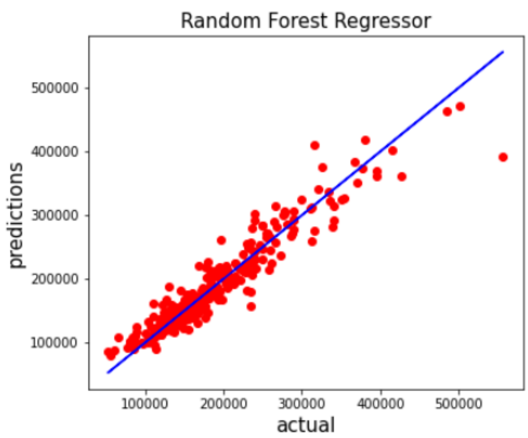



```
Cross Validation score at best cv=9 is : 73.48%
```

```
XGB Regressor Model

for XGB Regressor model, Best Random_state number for splitting the data is:  68

===scores for training set===
r2 score for training set 0.9999461787452694
MAE for training set:  360.0444412878788
MSE for training set:  272690.6933088083
SMSE for training set:  522.1979445658593

===scores for testing set===
r2 score for testing set :  0.905238463445502
MAE for testing set:  17500.65697443182
MSE for testing set:  594627494.5569751
SMSE for testing set:  24384.985022693272
```


```
Cross Validation score at best cv=10 is : 85.95%
```

```
Random Forest Regressor Model

for Random Forest Regressor model, Best Random_state number for splitting the data is:  68

===scores for training set===
r2 score for training set 0.9761357134113182
MAE for training set:  6718.856012121212
MSE for training set:  120910760.7721737
SMSE for training set:  10995.942923286466

===scores for testing set===
r2 score for testing set :  0.9050281445805635
MAE for testing set:  17149.144327272727
MSE for testing set:  595947242.8880346
SMSE for testing set:  24412.030699801166
```




```
Cross Validation score at best cv=8 is : 85.56%
```

Sr. No.	Model	Best_Random_State	Train_r2_Score	Test_r2_Score	Train_MAE	Train_MSE	Train_SMSE	Test_MAE	Test_MSE	Test_SMSE	Best_CV_Fold	Cross_Val_Score
5	Gradient Boosting Regressor	2	0.97	9.200000e-01	9401.64	1.573084e+08	12542.27	1.475236e+04	3.771432e+08	1.942018e+04	9	8.800000e-01
3	Ridge	15	0.93	9.000000e-01	13326.04	4.108653e+08	20269.81	1.640543e+04	4.690656e+08	2.165792e+04	10	8.600000e-01
2	Lasso	15	0.93	9.000000e-01	13311.06	4.106802e+08	20265.25	1.646020e+04	4.778053e+08	2.185876e+04	10	8.600000e-01
10	Random Forest Regressor	68	0.98	9.100000e-01	6718.86	1.209108e+08	10995.94	1.714914e+04	5.959472e+08	2.441203e+04	8	8.600000e-01
9	XGB Regressor	68	1.00	9.100000e-01	360.04	2.726907e+05	522.20	1.750066e+04	5.946275e+08	2.438499e+04	10	8.600000e-01
4	Ada-Boost Regressor	68	0.87	8.500000e-01	20481.74	6.754558e+08	25989.53	2.235007e+04	9.473870e+08	3.077965e+04	4	7.900000e-01
8	KNeighbors Regressor	8	0.80	8.000000e-01	21642.61	1.074528e+09	32779.99	2.306327e+04	9.969013e+08	3.157374e+04	9	7.300000e-01
6	Decision Tree Regressor	4	1.00	7.700000e-01	0.00	0.000000e+00	0.00	2.449153e+04	1.312869e+09	3.623353e+04	9	7.100000e-01
7	Support Vector Machine	17	-0.05	-5.000000e-02	52240.02	5.171791e+09	71915.16	5.368213e+04	7.138021e+09	8.448681e+04	9	-5.000000e-02
1	Linear Regression	88	0.94	-3.344018e+24	12779.55	3.092656e+08	17585.95	1.651278e+16	2.156372e+34	1.468459e+17	5	-5.226999e+25

- # Key Metrics for success in solving problem under consideration

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

  Mean Squared Error (MSE) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss.

  Cross Validation Score - to check if our model is overfitting or not we use cross validation score, higher the cross validation score higher the cross validation score means the model is not overfitting.

Hyperparameter Tuning:

Hyper parameter tuning using GridSearchCV

In [78]:
```
param_grid  = {'loss':['ls', 'lad'],
               'criterion':['friedman_mse', 'squared_error', 'mae'],
               'max_depth':[1,2,3],
               'n_estimators':[100, 200, 300, 400],
               'learning_rate':[0.01,0.02,0.1],
               'min_samples_split':[2,3,4]
              }

x_train, x_test, y_train, y_test = train_test_split(X, y_reg, test_size = 0.25, random_state = 2)


grid = GridSearchCV(estimator = gbr, param_grid = param_grid,verbose = 2,scoring = 'r2')
grid.fit(x_train,y_train)
```

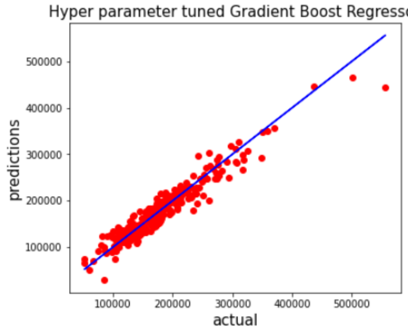
Using hyper parameters obtained from gridsearch..

In [81]:
```
gbr_tune_final = GradientBoostingRegressor(criterion='squared_error',loss='ls',learning_rate= 0.1, max_depth= 2, min_samples
```

In [84]:
```
gbr_tune_final.fit(x_train,y_train)
y_pred = gbr_tune_final.predict(x_test)

print('r2 score for testing set : ', r2_score(y_test, y_pred))
print('MAE for testing set:  ', mean_absolute_error(y_test, y_pred))
print('MSE for testing set:  ', mean_squared_error(y_test, y_pred))
print('SMSE for testing set:  ', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
r2 score for testing set :  0.9201405939235497
MAE for testing set:   14342.768284221345
MSE for testing set:   371297717.8744535
SMSE for testing set:   19269.087105373037
```

```python
In [86]:
##plotting the graph with bestfit line, actual & predicted values
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('Hyper parameter tuned Gradient Boost Regressor', fontsize = 15)
plt.show()
```




```python
In [87]:
cross_val_score(gbr_tune_final, X, y_reg, cv = 9, scoring ='r2').mean()
```

```
Out[87]: 0.89322749246374
```

Saving & predictions of the model on Test data provided

Saving & predicting regression model

```python
In [88]:
filename='Housing_data_reg.pkl'
pickle.dump(gbr_tune_final,open(filename,'wb'))
```

```python
In [97]:
model =pickle.load(open('Housing_data_reg.pkl','rb'))
pred =model.predict(test_final)
result = pd.DataFrame(list(pred), columns = ['Prediction'])
result
```

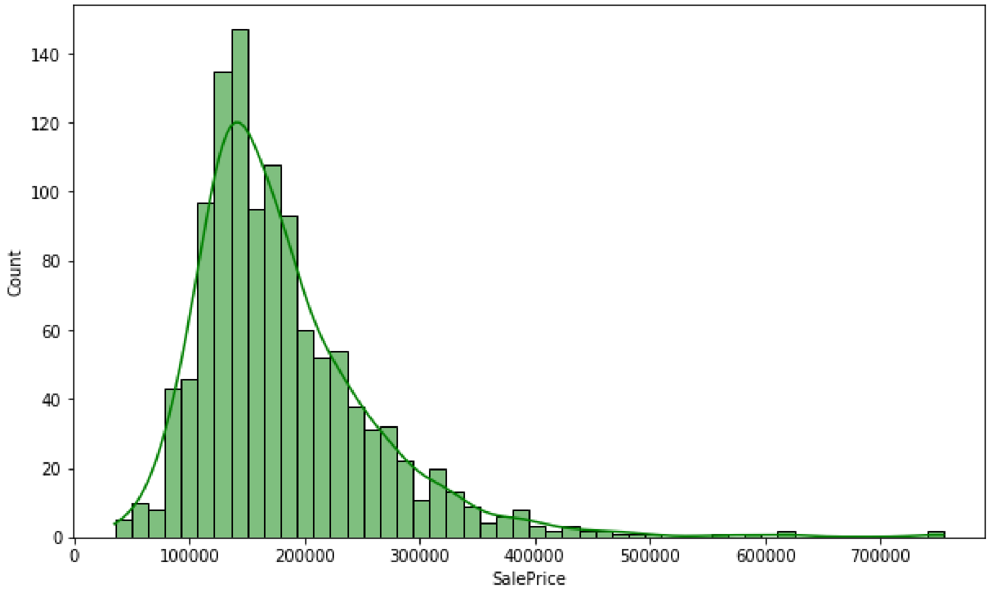
Out[97]:

	Prediction
0	380600.482555
1	228632.036343
2	244368.907465
3	188527.317230
4	207937.620070
...	...
269	235741.082497
270	130661.943554
271	155066.246241
272	150795.489327
273	100082.328790

274 rows × 1 columns

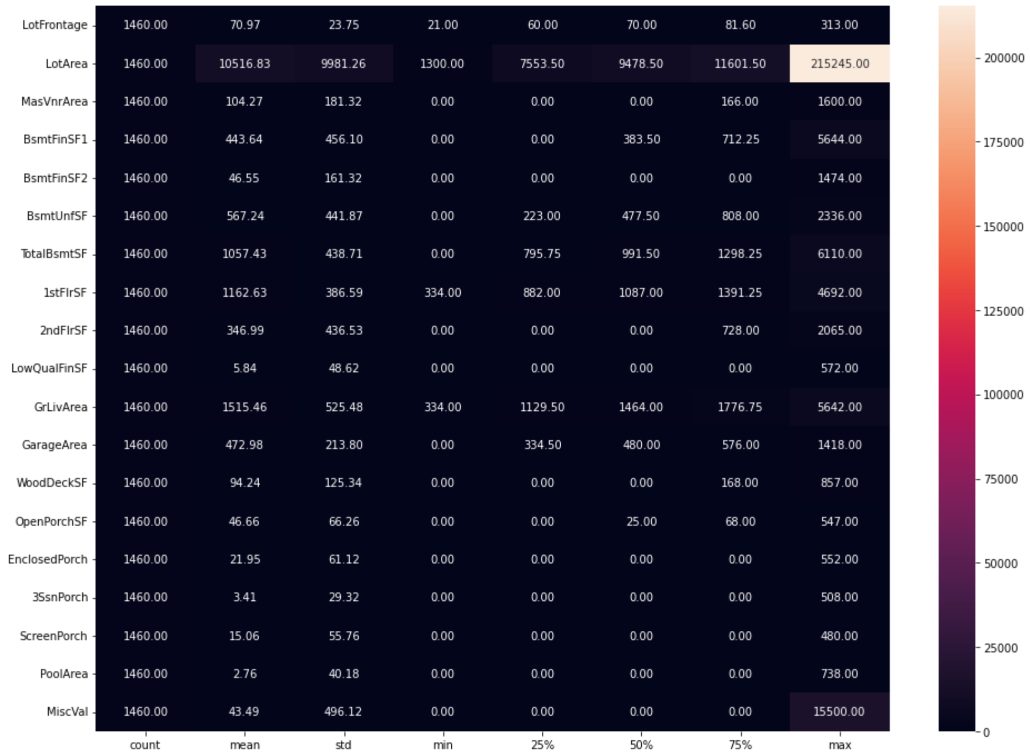
Visualizations & EDA

Continuous Numerical Features



Observation:

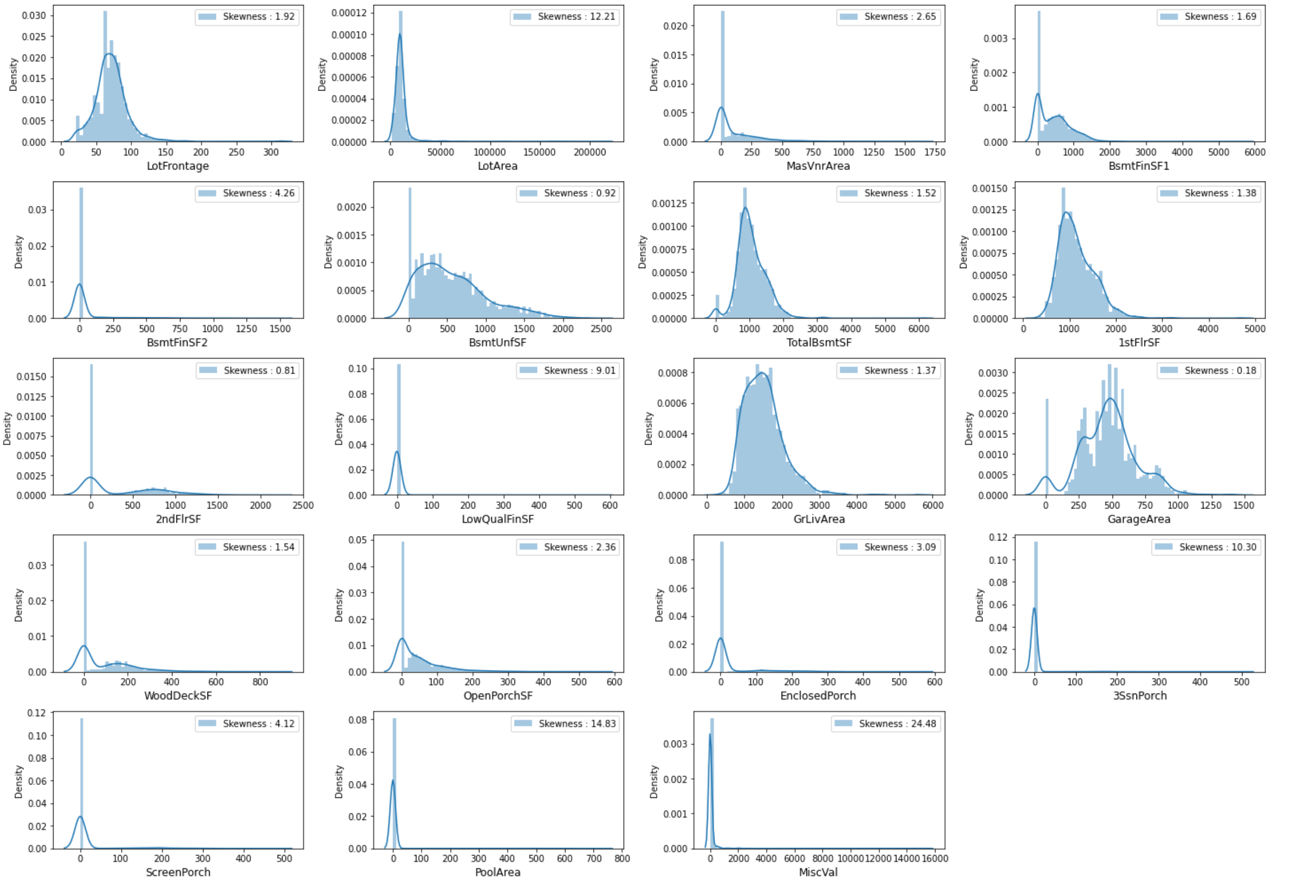
1. The Sale Price feature has right skewed distribution in which most Sales are between 80K and 340K.

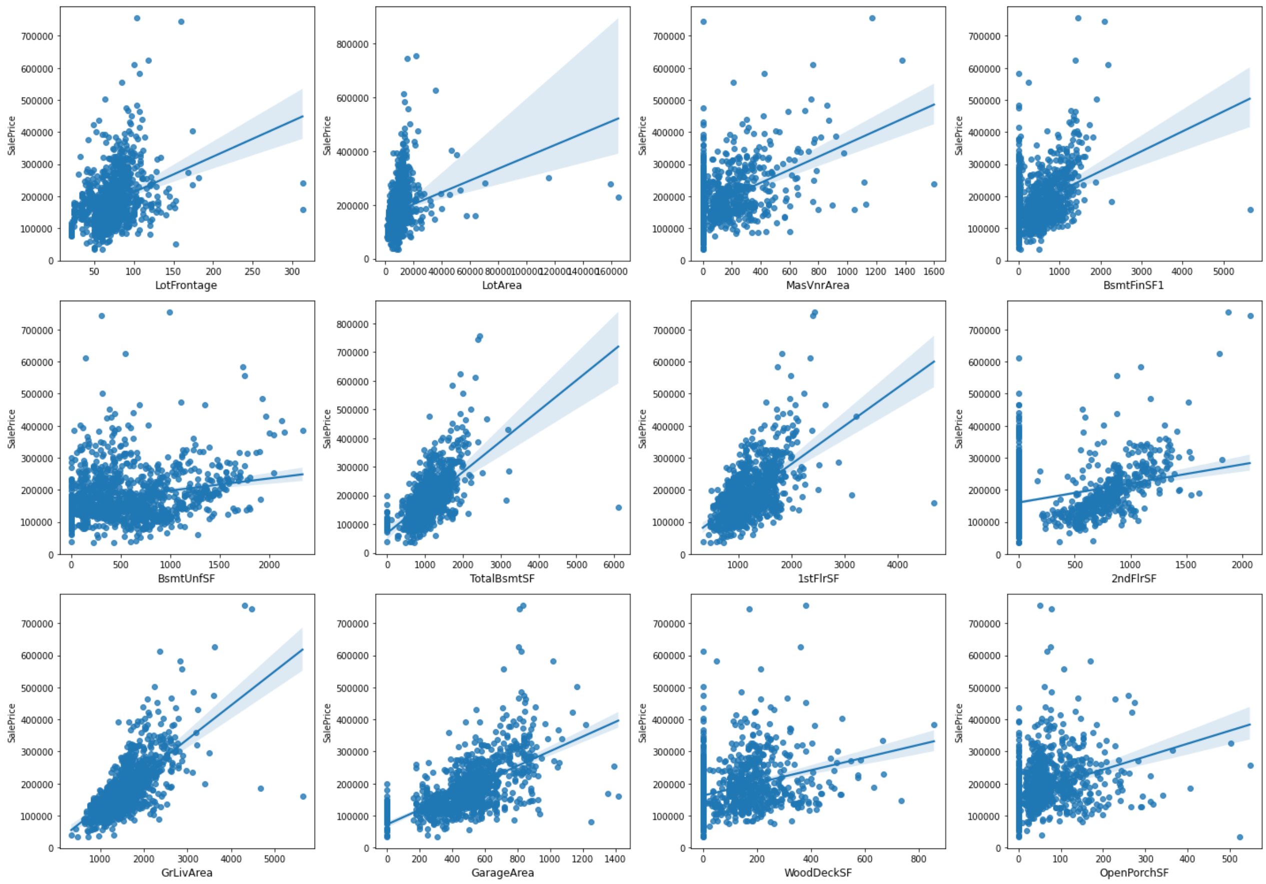


	count	mean	std	min	25%	50%	75%	max
LotFrontage	1460.00	70.97	23.75	21.00	60.00	70.00	81.60	313.00
LotArea	1460.00	10516.83	9981.26	1300.00	7553.50	9478.50	11601.50	215245.00
MasVnrArea	1460.00	104.27	181.32	0.00	0.00	0.00	166.00	1600.00
BsmtFinSF1	1460.00	443.64	456.10	0.00	0.00	383.50	712.25	5644.00
BsmtFinSF2	1460.00	46.55	161.32	0.00	0.00	0.00	0.00	1474.00
BsmtUnfSF	1460.00	567.24	441.87	0.00	223.00	477.50	808.00	2336.00
TotalBsmtSF	1460.00	1057.43	438.71	0.00	795.75	991.50	1298.25	6110.00
1stFlrSF	1460.00	1162.63	386.59	334.00	882.00	1087.00	1391.25	4692.00
2ndFlrSF	1460.00	346.99	436.53	0.00	0.00	0.00	728.00	2065.00
LowQualFinSF	1460.00	5.84	48.62	0.00	0.00	0.00	0.00	572.00
GrLivArea	1460.00	1515.46	525.48	334.00	1129.50	1464.00	1776.75	5642.00
GarageArea	1460.00	472.98	213.80	0.00	334.50	480.00	576.00	1418.00
WoodDeckSF	1460.00	94.24	125.34	0.00	0.00	0.00	168.00	857.00
OpenPorchSF	1460.00	46.66	66.26	0.00	0.00	25.00	68.00	547.00
EnclosedPorch	1460.00	21.95	61.12	0.00	0.00	0.00	0.00	552.00
3SsnPorch	1460.00	3.41	29.32	0.00	0.00	0.00	0.00	508.00
ScreenPorch	1460.00	15.06	55.76	0.00	0.00	0.00	0.00	480.00
PoolArea	1460.00	2.76	40.18	0.00	0.00	0.00	0.00	738.00
MiscVal	1460.00	43.49	496.12	0.00	0.00	0.00	0.00	15500.00

Observation:

1. We see most of the entries as 0, which means no such type of features in corresponding house.
2. Minimum LoaArea is of 1300 sqft, & max being 215245 sqft.
3. 50% & 75% quantile values from heat maps indicate skewness & outlier presence.

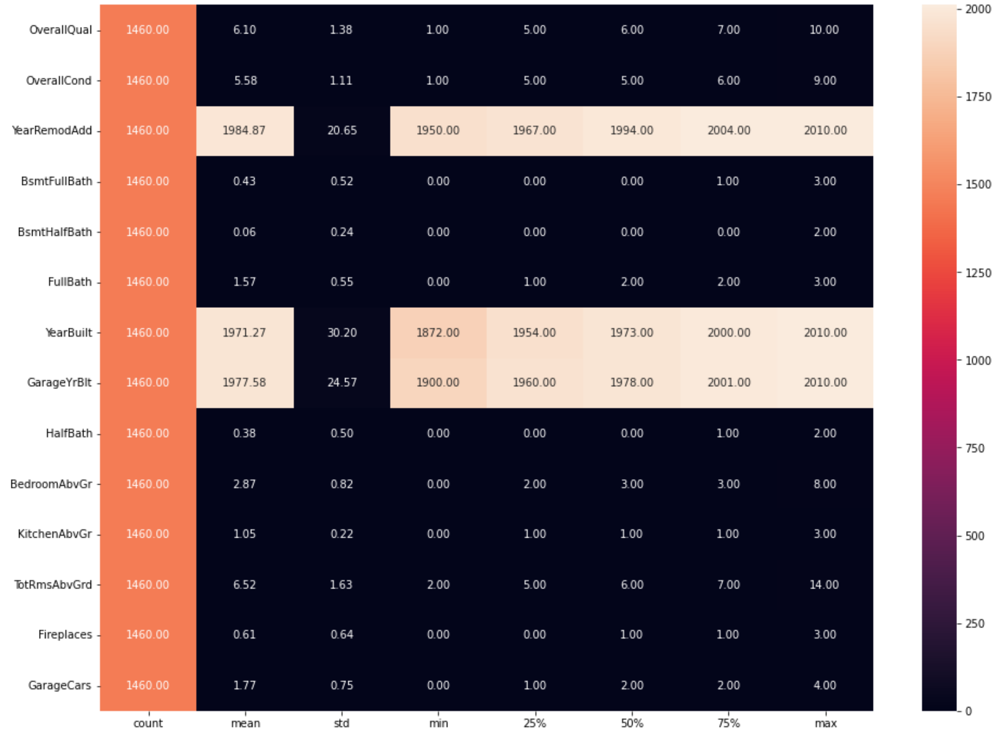




Observation:

1. We see most of the entries as 0, which means no such type of features in corresponding house.
2. Minimum LoaArea is of 1300 sqft, & max being 215245 sqft.
3. 50% & 75% quantile values from heat maps indicate skewness & outlier presence.

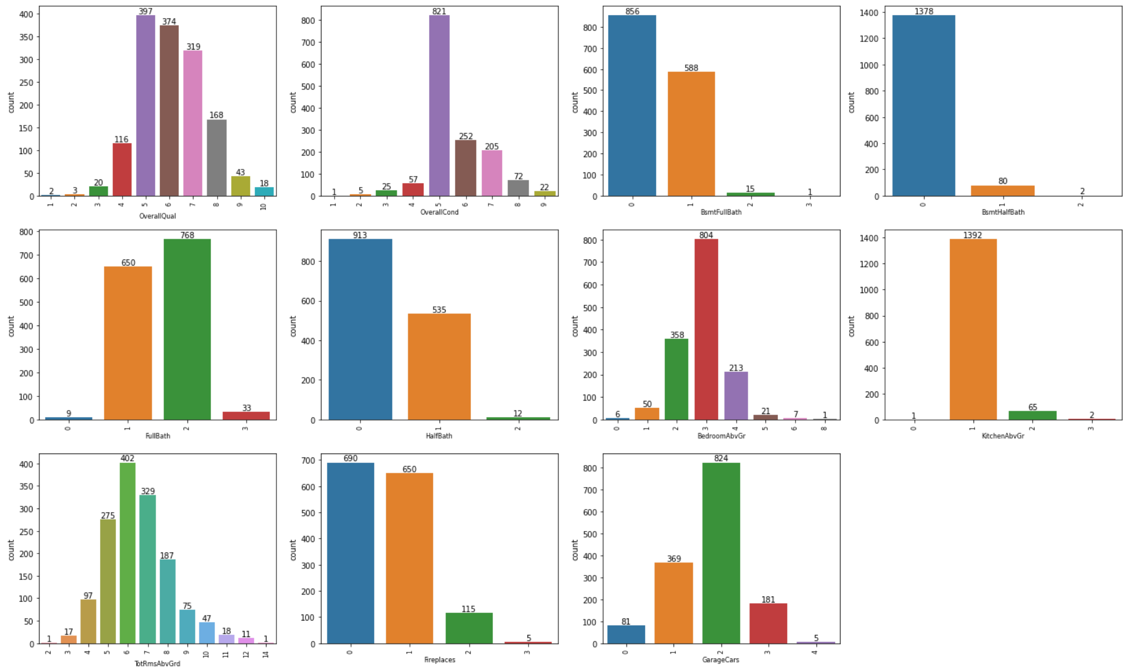
Discrete Numerical Features



	count	mean	std	min	25%	50%	75%	max
OverallQual	1460.00	6.10	1.38	1.00	5.00	6.00	7.00	10.00
OverallCond	1460.00	5.58	1.11	1.00	5.00	5.00	6.00	9.00
YearRemodAdd	1460.00	1984.87	20.65	1950.00	1967.00	1994.00	2004.00	2010.00
BsmtFullBath	1460.00	0.43	0.52	0.00	0.00	0.00	1.00	3.00
BsmtHalfBath	1460.00	0.06	0.24	0.00	0.00	0.00	0.00	2.00
FullBath	1460.00	1.57	0.55	0.00	1.00	2.00	2.00	3.00
YearBuilt	1460.00	1971.27	30.20	1872.00	1954.00	1973.00	2000.00	2010.00
GarageYrBlt	1460.00	1977.58	24.57	1900.00	1960.00	1978.00	2001.00	2010.00
HalfBath	1460.00	0.38	0.50	0.00	0.00	0.00	1.00	2.00
BedroomAbvGr	1460.00	2.87	0.82	0.00	2.00	3.00	3.00	8.00
KitchenAbvGr	1460.00	1.05	0.22	0.00	1.00	1.00	1.00	3.00
TotRmsAbvGrd	1460.00	6.52	1.63	2.00	5.00	6.00	7.00	14.00
Fireplaces	1460.00	0.61	0.64	0.00	0.00	1.00	1.00	3.00
GarageCars	1460.00	1.77	0.75	0.00	1.00	2.00	2.00	4.00

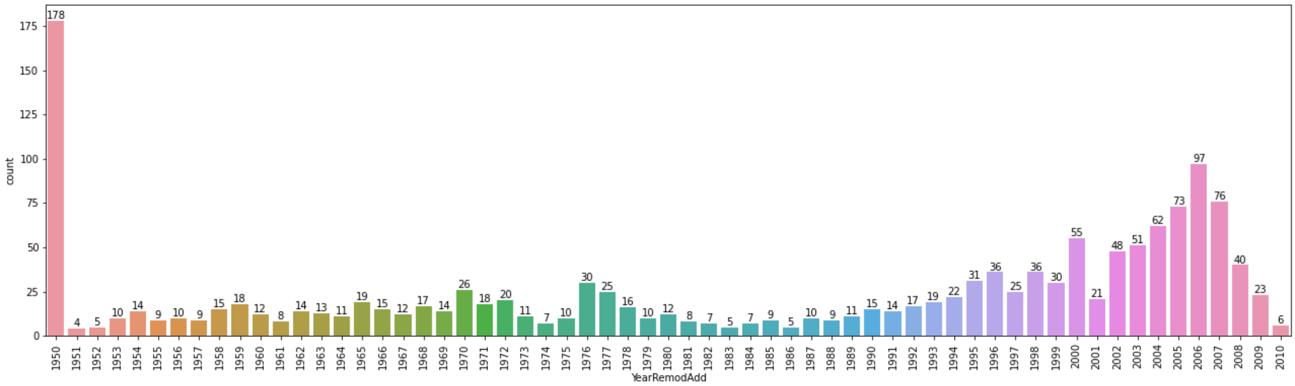
Observation:

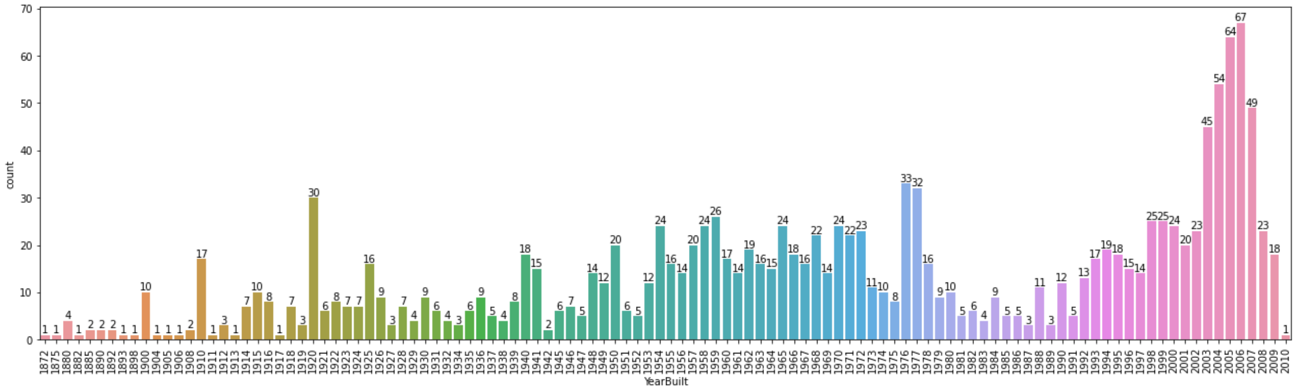
1. We see most of the entries as 0, which means no such type of features in corresponding house.
2. Oldest house built is in 1872, & newest being in 2010.
3. Oldest modification year was 1950, & recent modification year was 2010.



Observation:

1. 5, 6, 7 & 8 are the most common ratings given for Overall Condition & Quality.
2. 856 units have no basement with full bath feature, 588 houses have 1 full bath in basement.
3. 82 houses had half baths in basement area.
4. There were 1469 houses with kitchen above ground, 1442 houses with more than 4 rooms above ground.

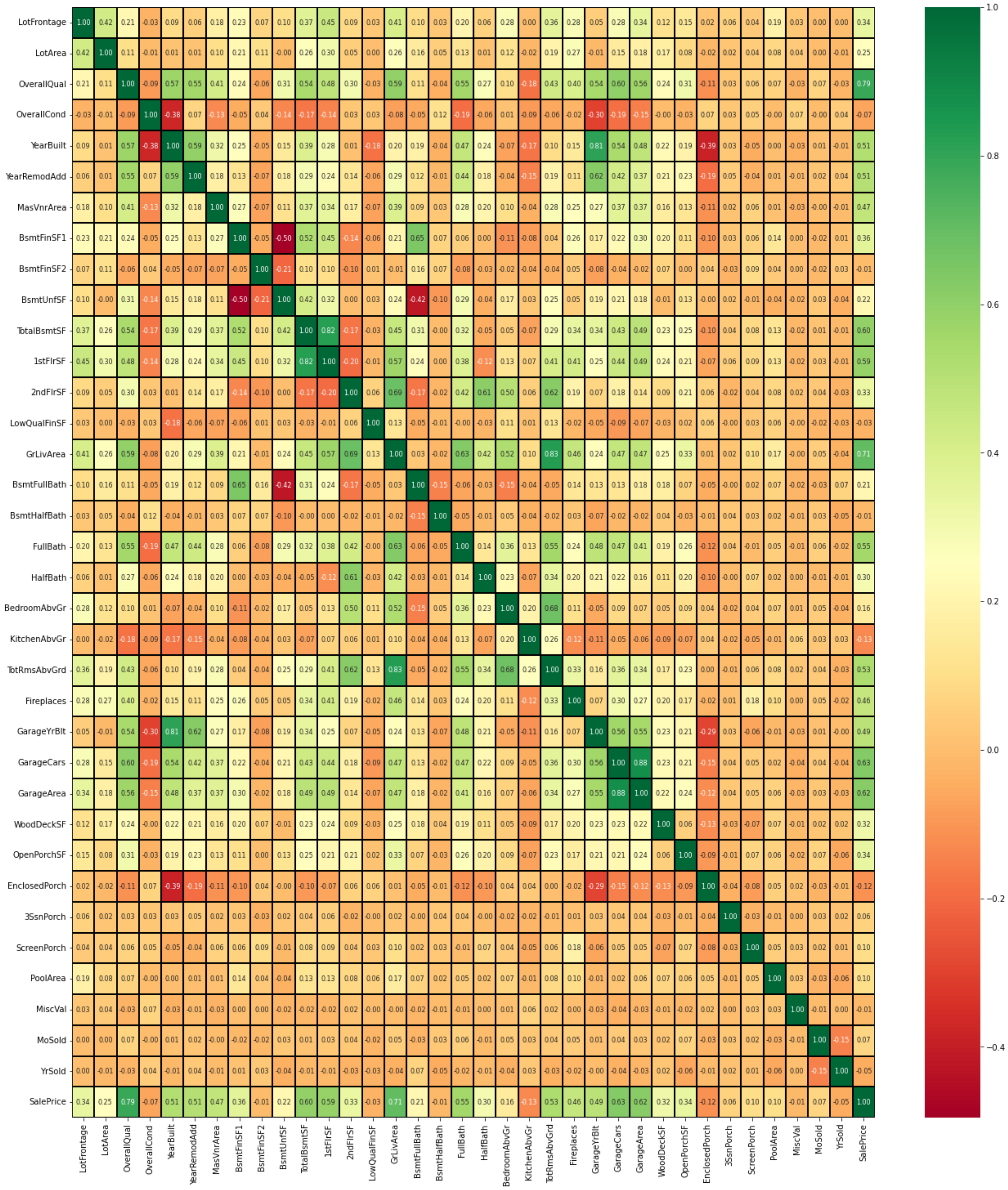




Observation:

1. Most of the Hoses were modified recently.
2. Most of the houses were built in the year 1940 to 1980 & 1990 to 2010.

All Numerical Features’ Correlation with Target variable SalePrice



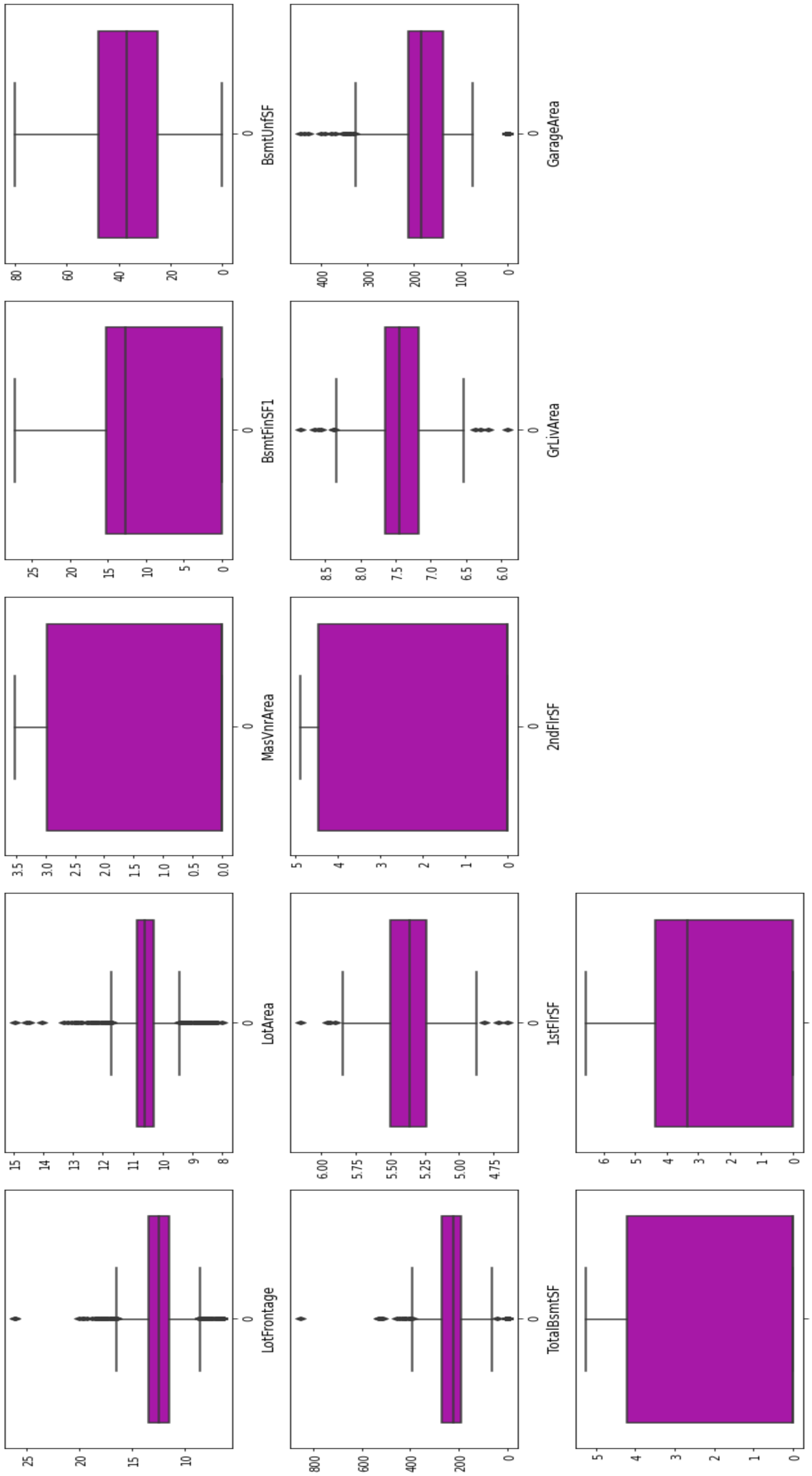
Observation:

1. Following are the numerical features with positive correlation:

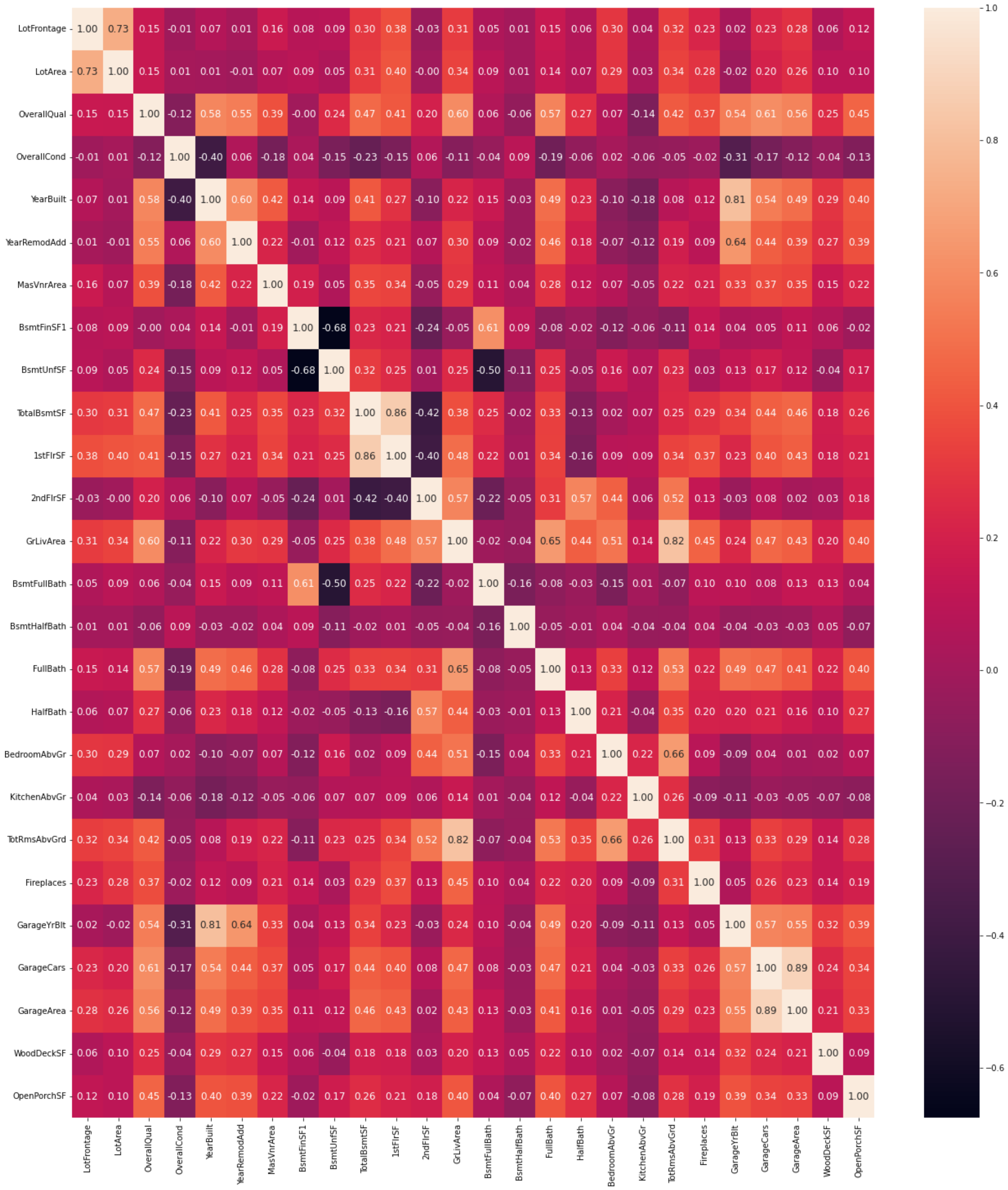
   Lotfrontage, Overall Quality, Year built, Remodification year, MasVnrArea, BsmtFinSF1, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, GrLivArea, FullBath, HalfBath, TotRmsAbvGrd, FirePlaces, GarageYrblt, GarageCars, GarageArea, WoodDeckSF & OpenPorchSF

2. Rest all numerical features were having no or very low correlation with the Sale price.

Outliers in given dataset



Correlation after data pre-processing & scaling

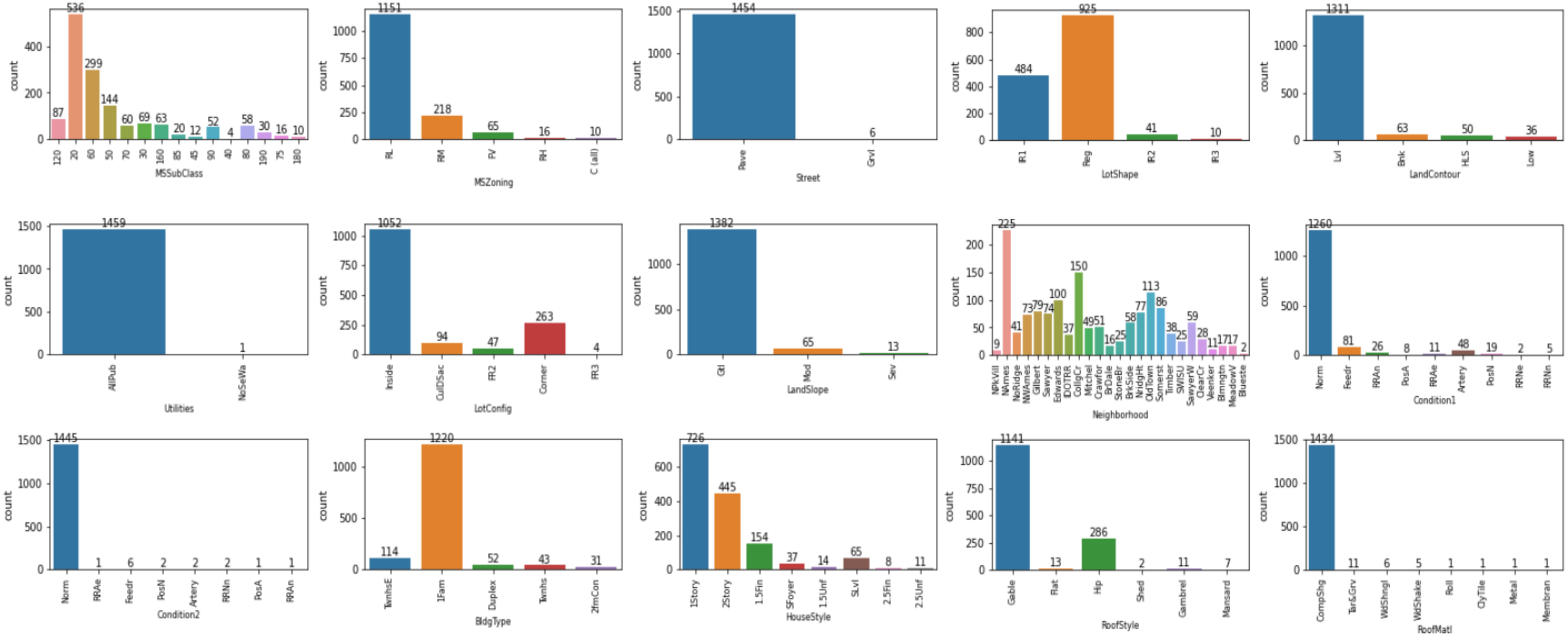


Observation:

Above Heatmap shows correlation between independent numerical features.

Most of the features were having multi collinearity issue. It was Overcome by using Variance influence factor. & We removed GrLivArea. The limit of VIF considered is 10.

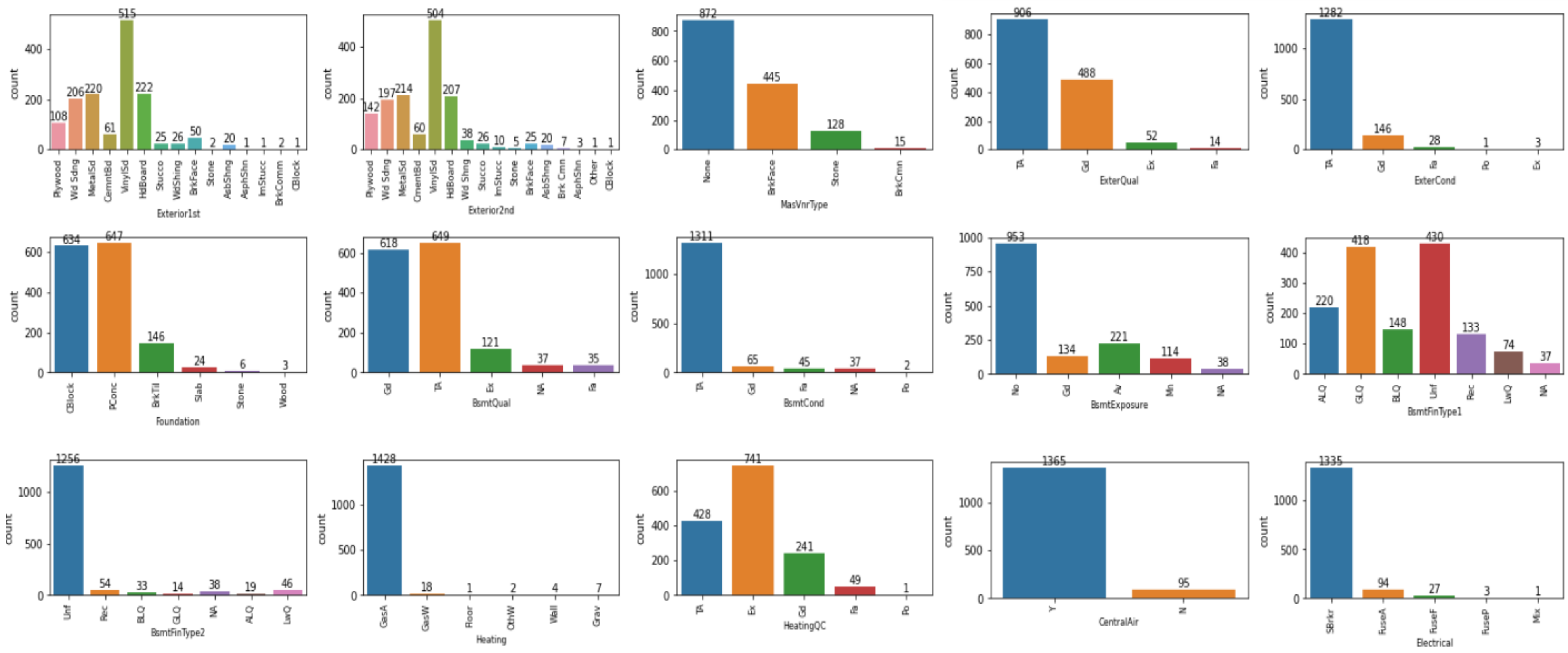
Categorical Features



Feature	Most occurring Entries
MSSubClass	20, 50 & 60
MSZoning	Residential Low Density
Street	Paved
LotShape	Regular
LandContour	Near Flat/Level

Feature	Most occurring Entries
Utilities	AllPublic Utilities
LotConfig	Inside
Landslope	Gentle slope
Neighborhood	NWAmes
Condition1	Normal

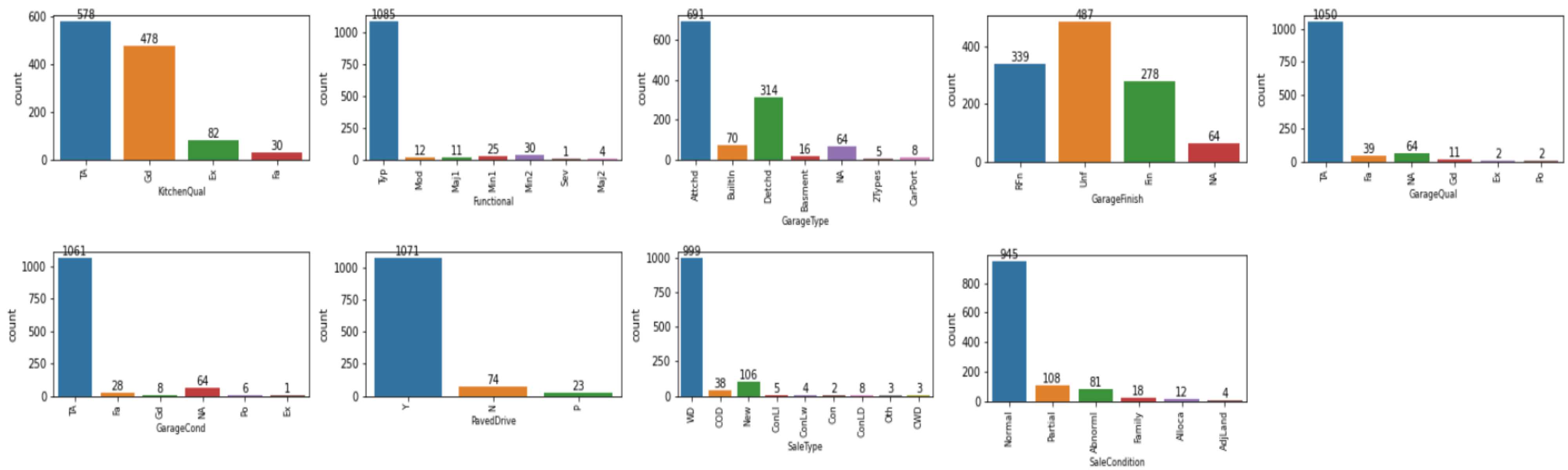
Feature	Most occurring Entries
Condition2	Normal
BldgType	Single-family Detached
HouseStyle	1Story, 2Story
RoofStyle	Gable
RoofMaterial	Standard (Comp) Shingle



Feature	Most occurring Entries
Exterior1st	Vinyl Siding
Exterior2nd	Vinyl Siding
MasVnrType	None
ExterQual	Good & Average/Typical
ExterCond	Average/Typical

Feature	Most occurring Entries
Foundation	Cinder Block & Poured Concrete
BsmtQual	Good & Typical
BsmtCond	Typical
BsmtExposure	No Exposure
BsmtFinType1	Unfinished & Good Living Quarters

Feature	Most occurring Entries
BsmtFinType2	Unfinished
Heating	Gas forced warm air furnace
HeatingQC	Excellent
CentralAir	Yes
Electrical	Standard Circuit Breakers & Romex



Feature	Most occurring Entries
KitchenQual	Good & Typical/Average
Functional	Typical Functionality
GarageType	Attached to home
GarageFinish	Unfinished
GarageQual	Average/Typical

Feature	Most occurring Entries
GarageCond	Typical/Average
PavedDrive	Paved
SaleType	Warranty Deed - Conventional
SaleCondition	Normal Sale

CONCLUSION

- ## Key Findings and Conclusions of the Study

  a. Most of the numerical features were having Positive relation with the Sale Price.
  b. Following features have high impact on Sale Price predictions:
     i. OverallQuality
     ii. GarageCars
     iii. GarageArea
     iv. FullBath
     v. GarageYrBlt
     vi. TotalRooms above ground
     vii. Sq ft area in basement
     viii. Sq ft area in first floor
  c. Our Model can predict with 92% accuracy, with mean absolute error of $14324AUD

- ## Learning Outcomes of the Study in respect of Data Science

  a. Dealing with outliers & skewness, filling missing values based on other independent features.
  b. Visualization techniques.
  c. Tuning hyperparameter efficiently

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

  1. Dataset is very small in size.
  2. Huge number of null values in some features
  3. Zero values are more than 75% of the data. Had to remove these features to achieve better performing model
  4. Skewness limits the model accuracy.
  5. Outlier removal takes out more than 5 % of the data.
  6. Multi collinearity issues in some of the features.
  7. Huge number of categorical features there by increases columns after encoding, resulting in model run time.