House Price Prediction

Submitted By
Swati Kumari

June 24, 2022

Agenda:

Data preparation
Explanatory Data Analysis
Data Cleaning & Imputation
Remove Outliers
Feature engineering
Model selection
Model Prediction
Optimization
Conclusion

Data preparation

- Checking the shape of Datasets
- Checking the columns
- Checking the Data types Of independent features
- Checking the null values
- Checking and dropping the unwanted columns
- Checking Categorical columns and numerical columns

Address Null value fields

- Top columns with null values:

```
# have null columns with dtype = object
null_object_col = df_train.loc[:,df_train.isnull().sum() != 0].loc[:,df_train.loc[:,df_train.isnull().sum() != 0].dtypes ==
null_object_col
```

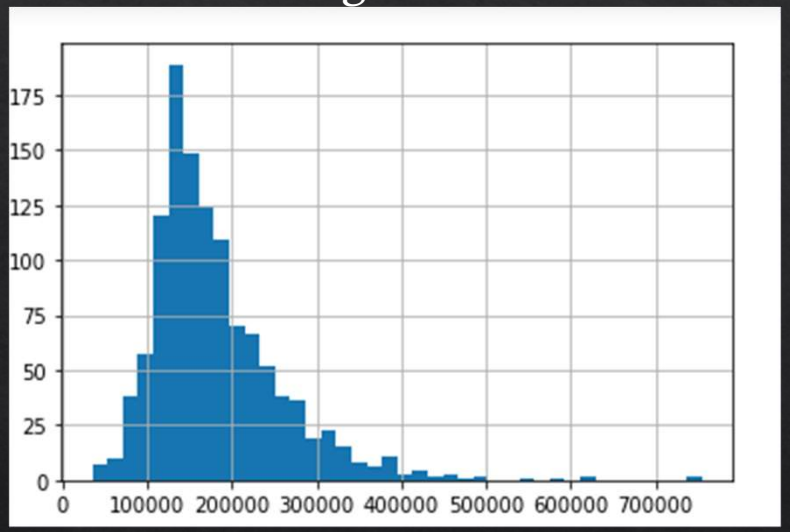
```
Index(['Alley', 'MasVnrType', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
       'BsmtFinType1', 'BsmtFinType2', 'FireplaceQu', 'GarageType',
       'GarageFinish', 'GarageQual', 'PoolQC', 'Fence', 'MiscFeature'],
      dtype='object')
```

- Most “Null” values actually mean “Not Available”
  - Example: PoolQC - Pool quality – Not Available means “No Pool”
  - Update all meaningful “Null_object col” to “Not Available”

- For numerical columns like LotFrontage, GarageYrBlt, MasVnrArea, mean is used to replace the null values

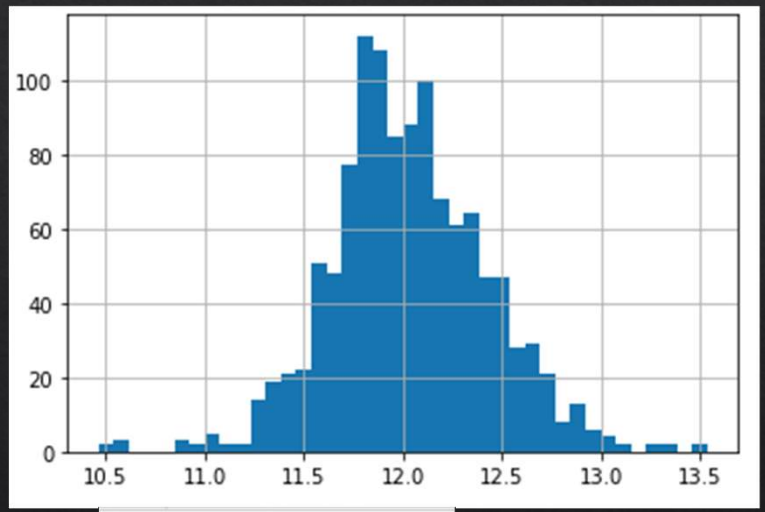
Analyzing target Column-SalePrice

Before removing skewness



Skewness: 1.953878
Kurtosis: 7.390657

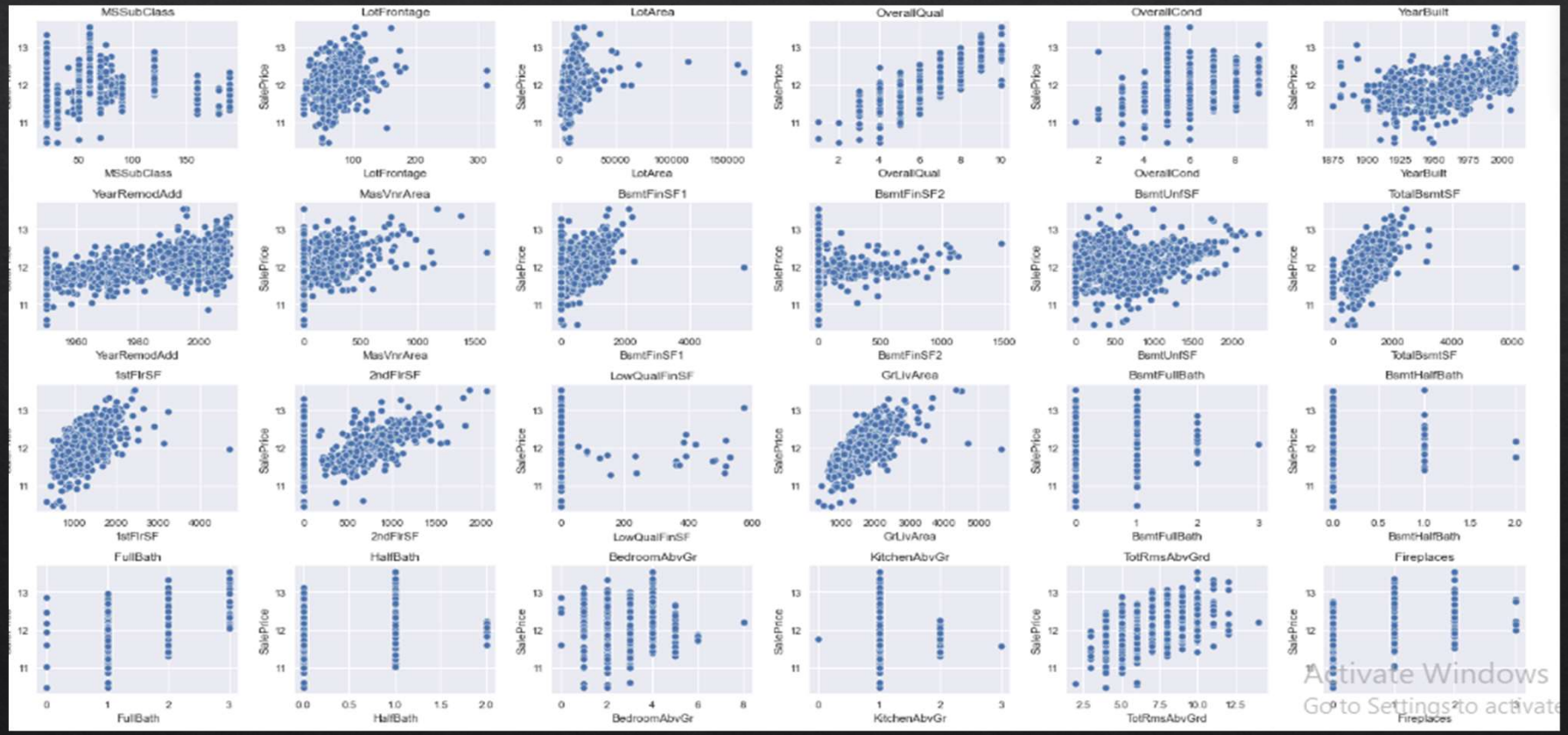
After removing skewness



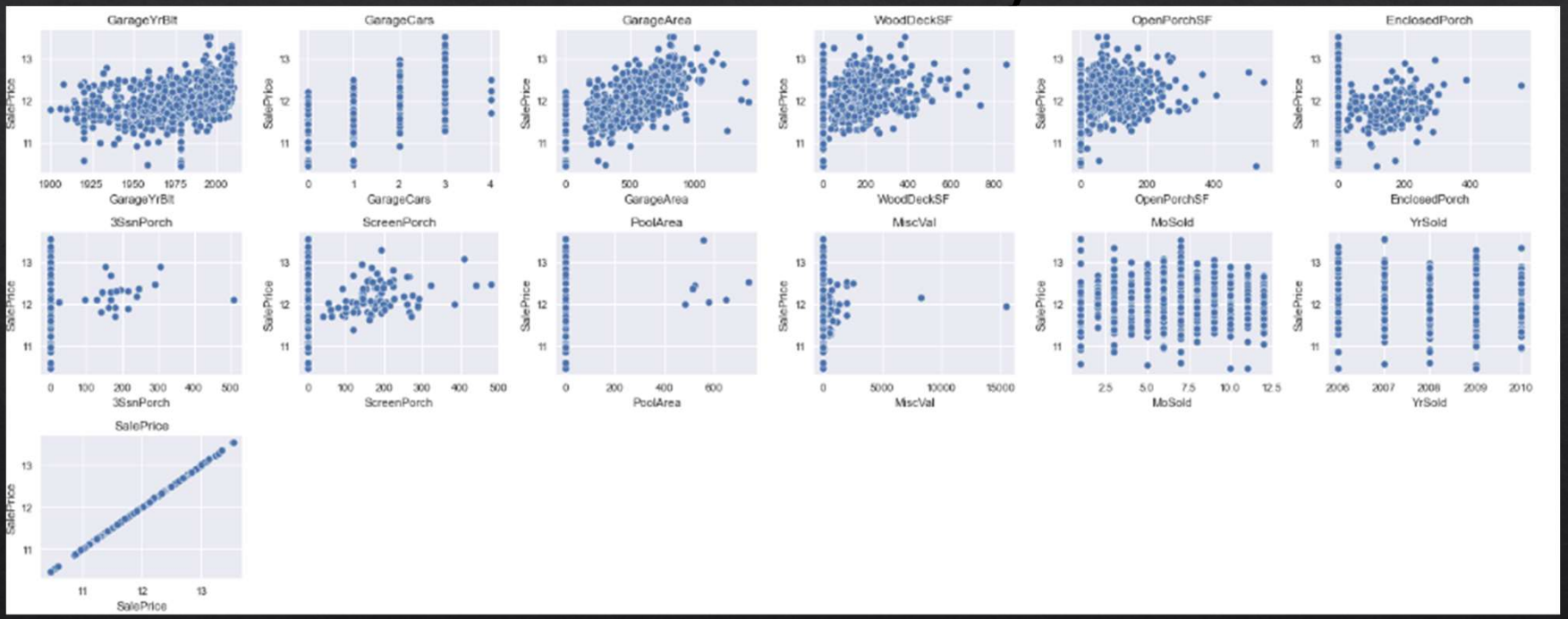
Skewness: 0.073610
Kurtosis: 0.995996

Since the target column is skewed towards right. So applying log transformation

Numerical Features analysis

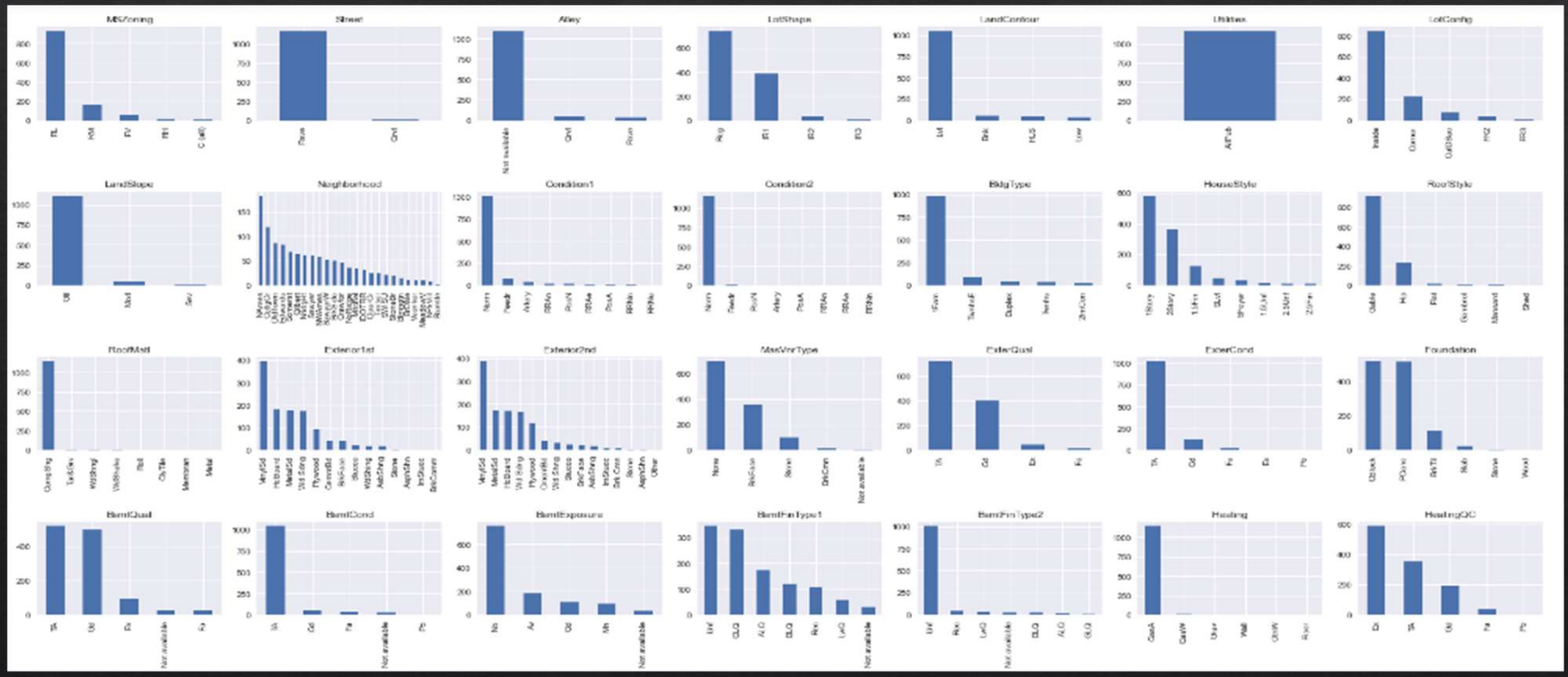


Numerical Features Analysis Cont..

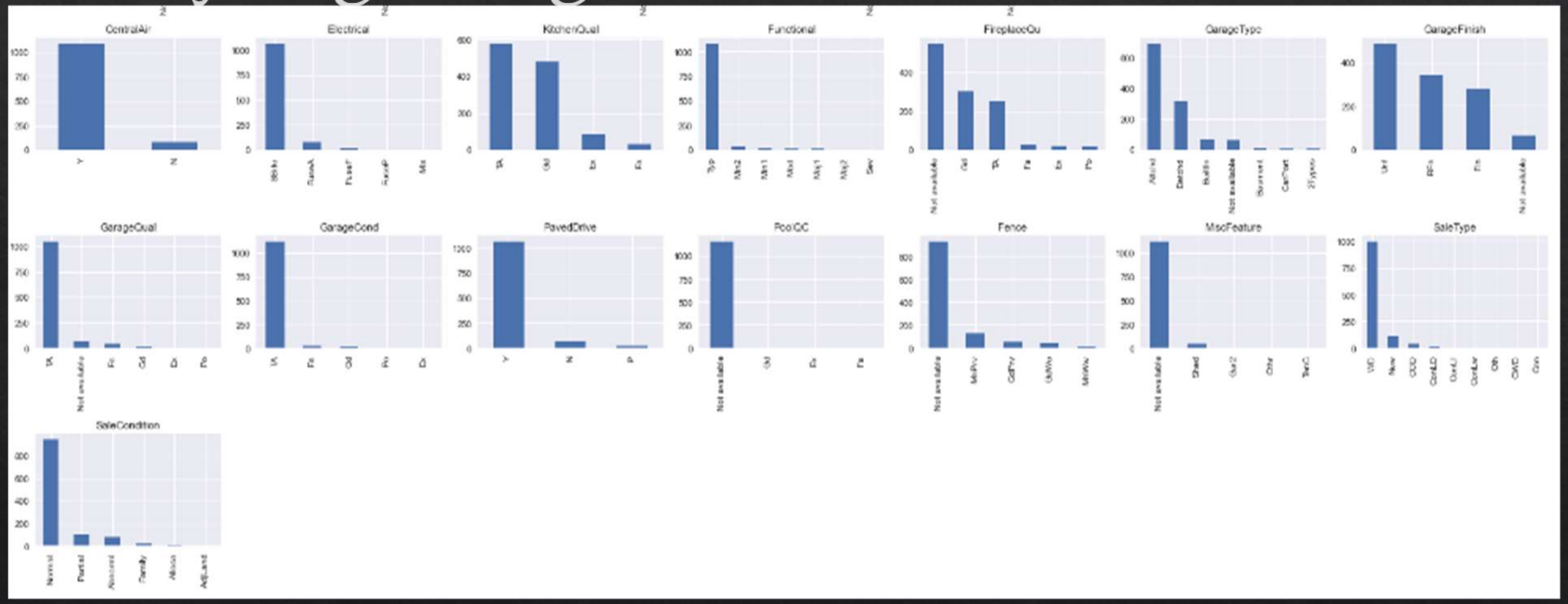


Here in all the numerical columns we can see through scatter plot that many numerical features are skewed on either the left side or right side. Also, there are so many outliers in the features.

Analyzing Categorical columns

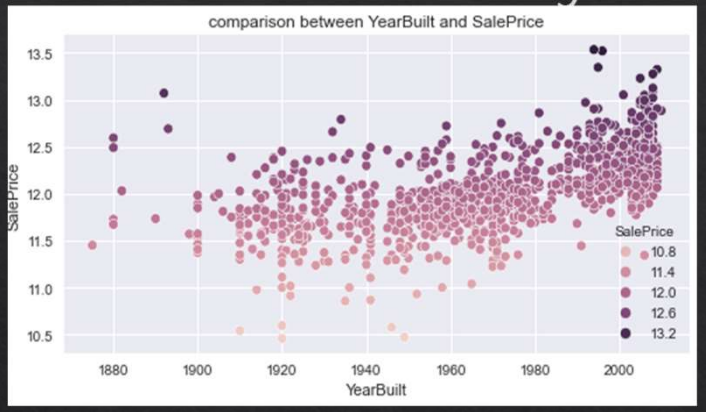


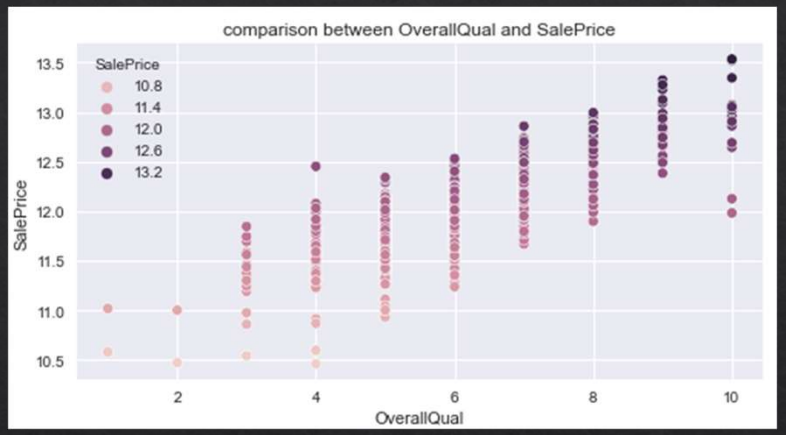
Analyzing Categorical columns cont.

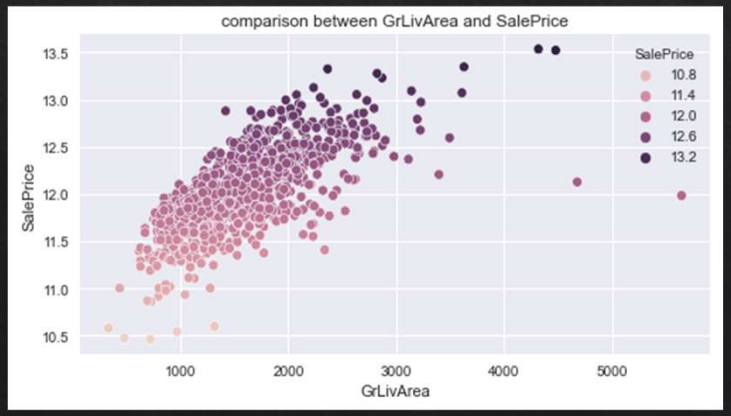


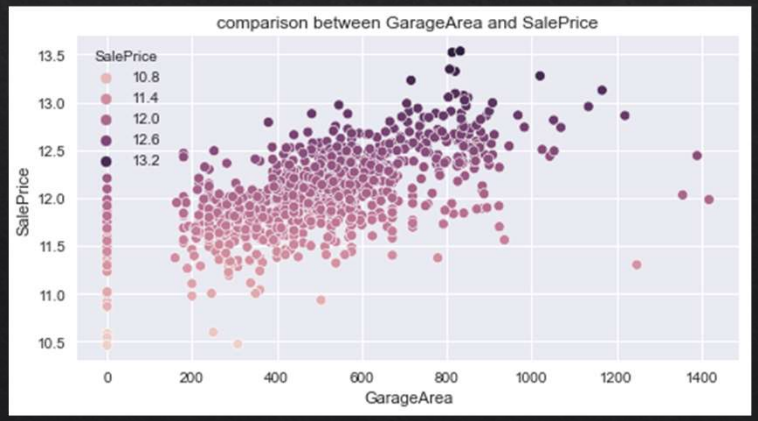
Through bar charts of different categorical columns, we can see unique values and the value counts for categorical features.

Bivariate analysis



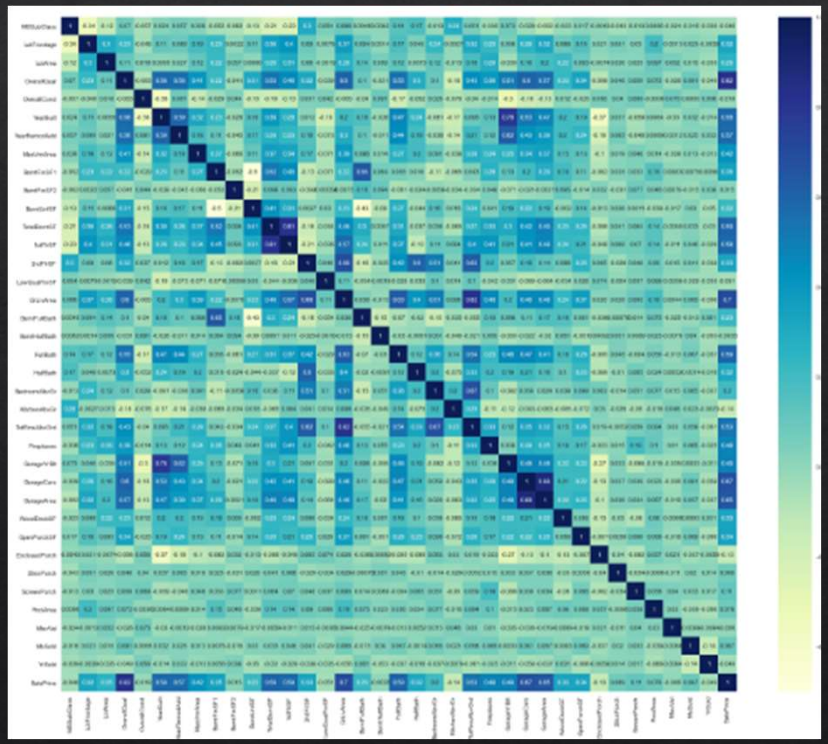


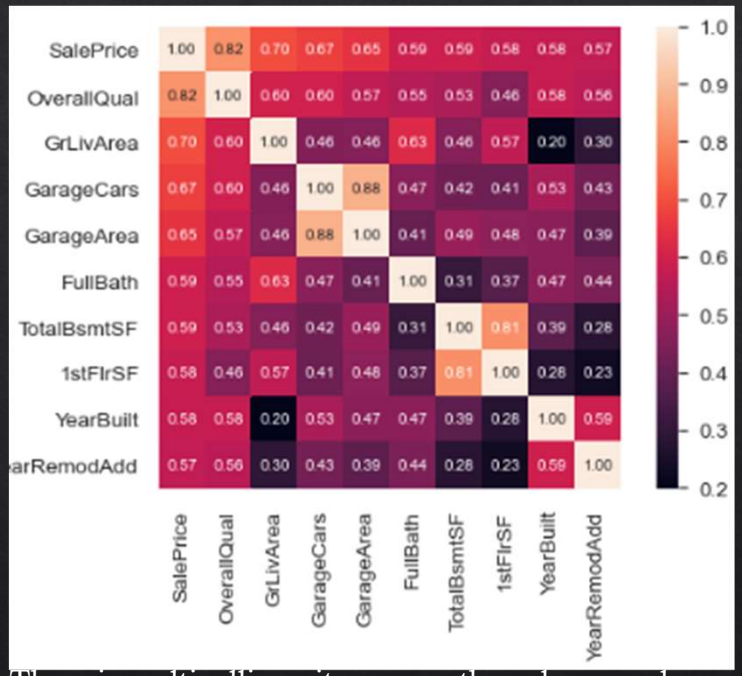




For all the above variables, Sale price increases with their

Checking the correlation

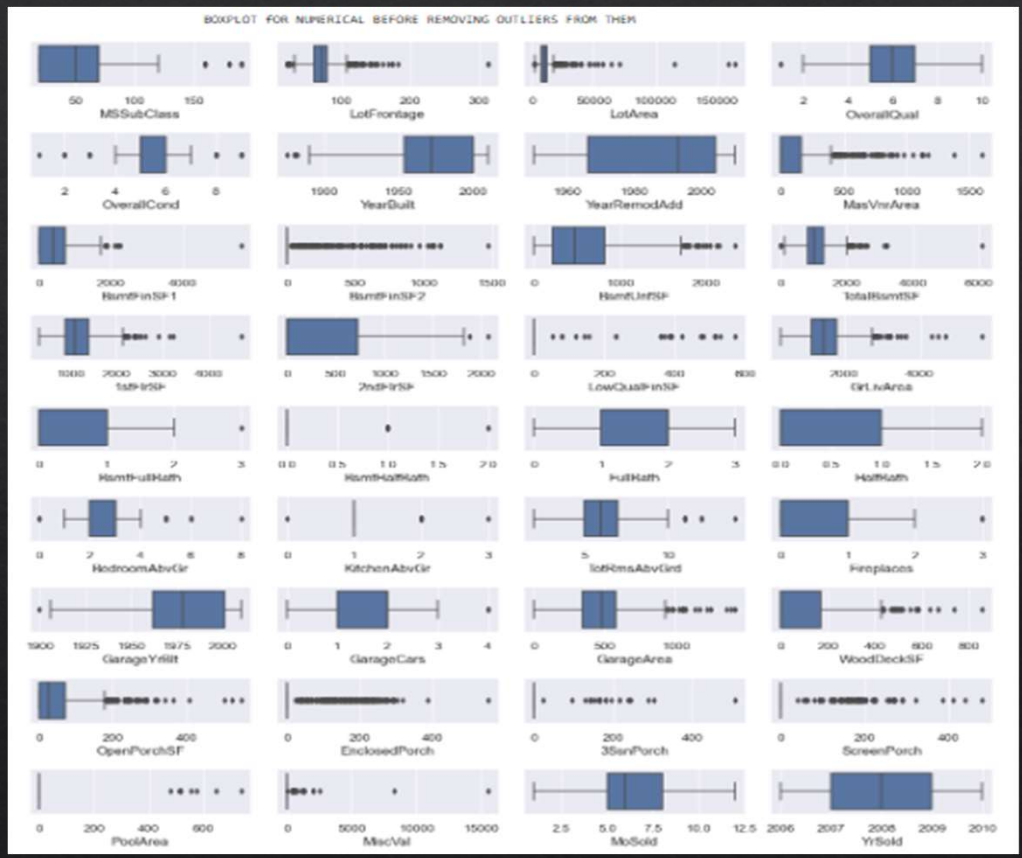




There is multicollinearity among the columns sale price and OverallQual, GarageCars And GarageArea,TotalBsmtSf and 1stFlrSF. But these columns have high correlation with target also

Outliers check

So many outliers are there in all the features. We removed the outliers using the interquartile method.

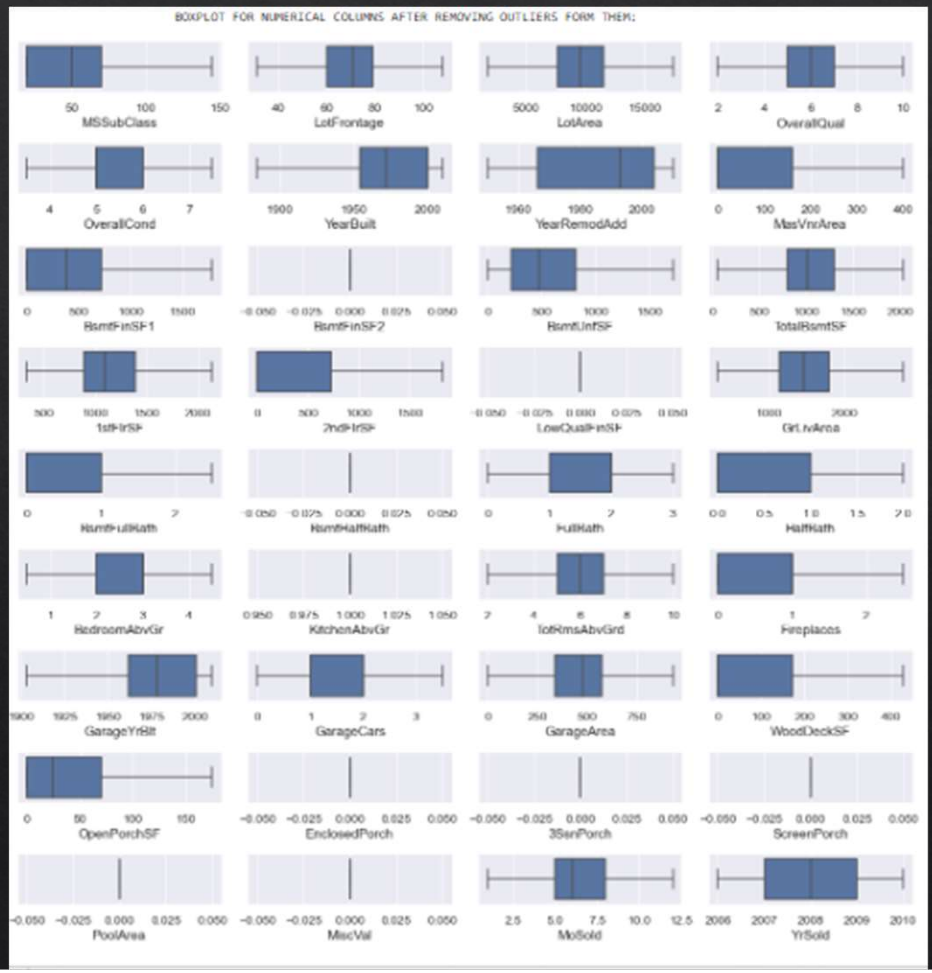


Removing Outliers

```python
def removeOutliers(new_num_var):
    global df_train
    for i in range(len(new_num_var)):
        q1 = df_train[new_num_var[i]].quantile(0.25)
        q3 = df_train[new_num_var[i]].quantile(0.75)
        IQR = q3-q1
        minimum = q1 - 1.5 * IQR
        maximum = q3 + 1.5 * IQR
        df_train.loc[(df_train[new_num_var[i]] <= minimum), new_num_var[i]] = minimum
        df_train.loc[(df_train[new_num_var[i]] >= maximum), new_num_var[i]] = maximum
```

```python
removeOutliers(new_num_var)
```

We can see in the box plots, the outliers are removed



Label Encoding and Feature Scaling

Converted all the categorical columns to numerical columns using Label encoder

By using a standard scaler, I have scaled the data in one range.

```
# Performing Standard scaler
#For train data
sc = StandardScaler()
X = sc.fit_transform(X)

#For test data
price_test = sc.fit_transform(df_test_le)
```

```
X
```

```
array([[ 1.70759409, -0.02164599,  0.02879392, ..., -0.60548713,
         0.33003329,  0.20793187],
       [-1.02115826, -0.02164599,  1.45626837, ..., -0.60548713,
         0.33003329,  0.20793187],
       [ 0.21918372, -0.02164599,  1.28497144, ..., -0.60548713,
         0.33003329,  0.20793187],
       ...,
       [ 1.95566248, -0.02164599, -2.19806622, ...,  0.8992128 ,
         0.33003329,  0.20793187],
       [ 0.46725211, -4.76211672, -1.17028462, ...,  0.14686284,
         0.33003329,  0.20793187],
       [ 0.21918372, -0.02164599,  0.02879392, ..., -1.3578371 ,
         0.33003329,  0.20793187]])
```

Finding the best random state

```
maxScore = 0
maxRS = 0

for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=i)
    lr = LinearRegression()
    lr.fit(x_train,y_train)
    pred_train = lr.predict(x_train)
    pred_test = lr.predict(x_test)
    acc=r2_score(y_test,pred_test)
    if acc>maxScore:
        maxScore=acc
        maxRS=i
print('Best score is',maxScore,'on Random State',maxRS)
```

```
Best score is 0.9240826608036915 on Random State 84
```

Applying on 5 different algorithms

- LinearRegression(),
- Lasso()
- Ridge()
- DecisionTreeRegressor()
- KNeighborsRegressor()]

Train And test Scores of 5 Different Algorithms

```
Train Score of LinearRegression() is: 0.8949500885534561
r2_score 0.9240826608036915
mean_squred_error 1818.6729876007134
RMSE 42.645902354161926

Train Score of Lasso() is: 0.8917401227582257
r2_score 0.9261040022874991
mean_squred_error 1770.249805304877
RMSE 42.074336659118906

Train Score of Ridge() is: 0.8949527937715197
r2_score 0.9242053679462536
mean_squred_error 1815.7334198033445
RMSE 42.61142358339304

Train Score of DecisionTreeRegressor() is: 1.0
r2_score 0.7182627273349806
mean_squred_error 6749.28774928775
RMSE 82.15404889162646

Train Score of KNeighborsRegressor() is: 0.8761395853867664
r2_score 0.8369131698871278
mean_squred_error 3906.9021082621084
RMSE 62.505216648389506
```

The Decision tree regressor model is overfitting. Other models are working well. But lasso regression is having less train and test score difference with least mean square error and least RMSE.

Ensemble Technique to boost up score

- **Random Forest Regressor:**

```
Train Score of RandomForestRegressor(random_state=84) is: 0.9813767323691924
r2_score 0.8930452761631832
mean_squred_error 2562.203433333333
RMSE 50.61821246679236
```

- **AdaBoostRegressor:**

```
Train Score of AdaBoostRegressor(base_estimator=Lasso(), random_state=84) is: 0.8622186362608182
r2_score 0.8681156446260296
mean_squred_error 3159.4167701999227
RMSE 56.208689454566745
```

- **GradientBoostingRegressor:**

```
Train Score of GradientBoostingRegressor() is: 0.9679385176546844
r2_score 0.9188795601235165
mean_squred_error 1943.3182762656959
RMSE 44.08308378806655
```

RandomForestRegressor and GradientBoostRegressor is overfitting. I have found that AdaBoostRegressor() is working well on the dataset with the least train score and test score difference and have given less RMSE score . So i am selecting AdaBoostRegressor for final Model.

Hyperparameter tuning to find best parameters of AdaBoost Regressor

```
Ada_Boost = AdaBoostRegressor()
Para ={'n_estimators' : [50, 100, 150, 200],
        'learning_rate' : [0.001, 0.01, 0.1, 1],
        'loss' : ["linear", "square", "exponential"],
        'random_state' : [21, 42, 104, 111]
        }
Rand_search = RandomizedSearchCV(Ada_Boost,Para,cv = 5,scoring = "r2",n_jobs =-1,verbose = 2)
Rand_search.fit(X_train1,y_train1)
print(Rand_search.best_params_)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
{'random_state': 21, 'n_estimators': 200, 'loss': 'linear', 'learning_rate': 1}
```

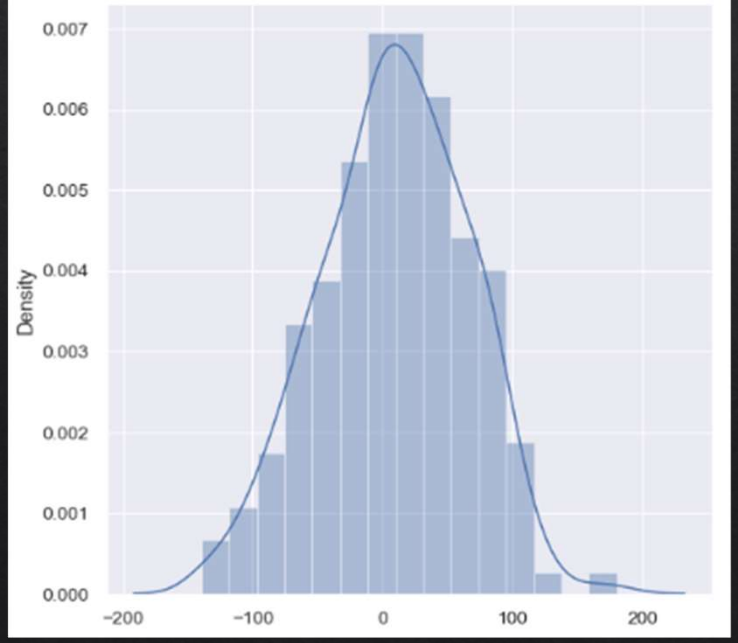
```
prediction = Rand_search.predict(X_test1)
```

```
SalePrice = AdaBoostRegressor(n_estimators= 200, loss= 'linear', learning_rate =1, random_state=21)
SalePrice.fit(x_train, y_train)
pred = SalePrice.predict(x_test)
print('R2_Score:',r2_score(y_test,pred)*100)
print("RMSE value:",np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2_Score: 86.21189353986094
RMSE value: 56.714676962897954
```

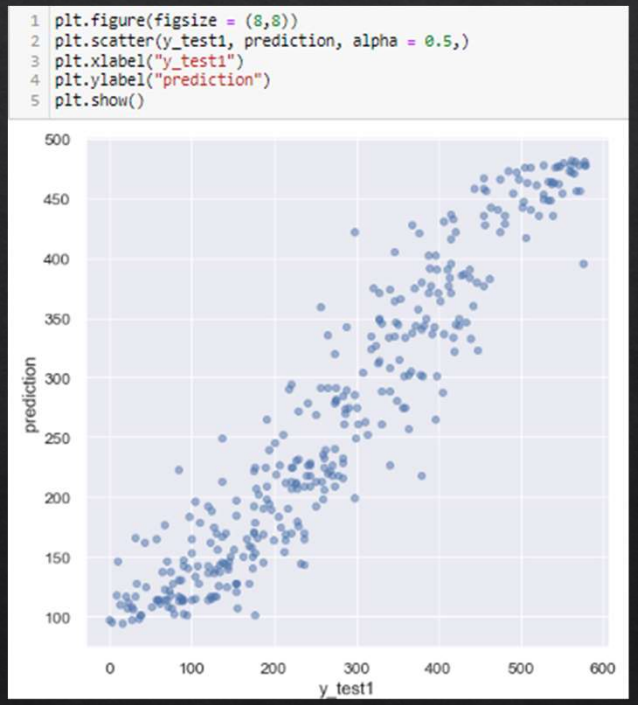
Plotting the residuals.

```
1 plt.figure(figsize = (8,8))
2 sns.distplot(y_test1-prediction)
3 plt.show()
```

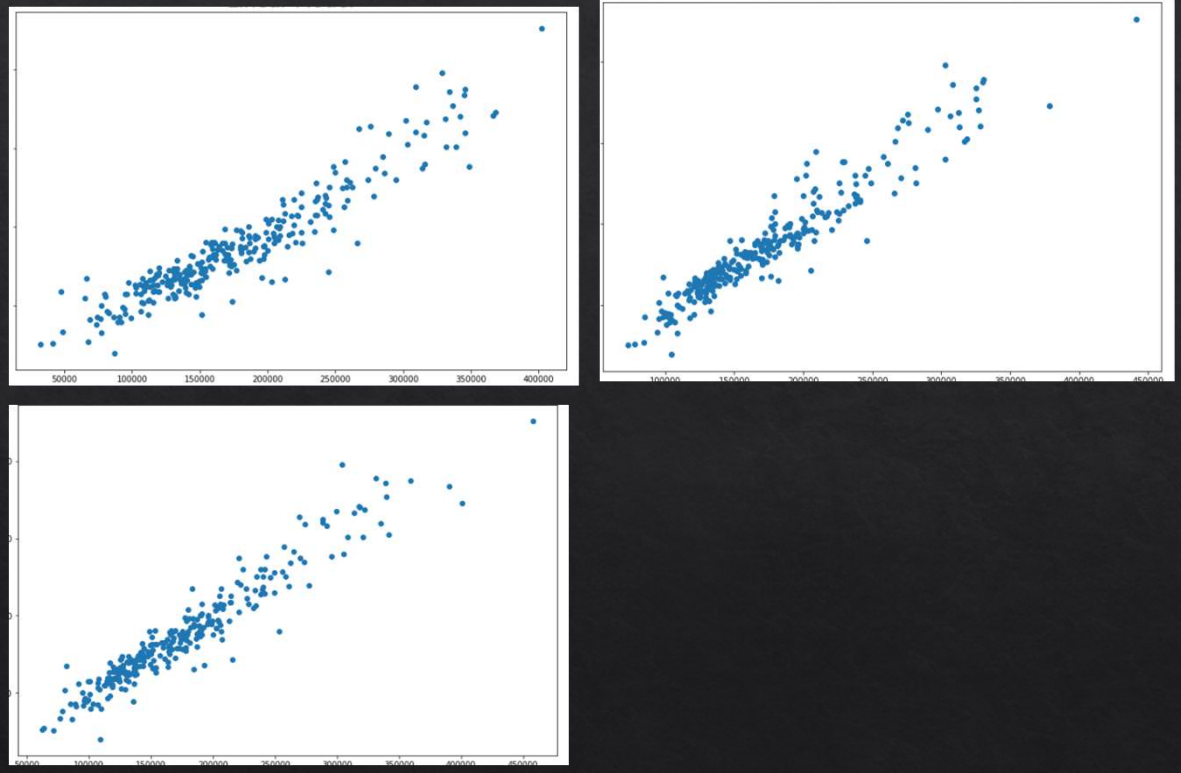


Plotting y_test vs predictions.

```
1 plt.figure(figsize = (8,8))
2 plt.scatter(y_test1, prediction, alpha = 0.5,)
3 plt.xlabel("y_test1")
4 plt.ylabel("prediction")
5 plt.show()
```



Model Stacking - Graphically



Conclusion:

- We have done a complete EDA process, getting data insights, feature engineering, and data visualization as well so after all these steps one can go for the prediction using machine learning model-making steps.
- We have training and test file separately available with us. we have both numerical and categorical data types features in both datasets and the dependent variable of train data i.e. the price is the numerical data type. So, I applied the regression method for prediction.
- Once data has been cleaned for both test and train datasets, Label encoding is applied to them to convert them into Numerical ones. I trained the model on five different algorithms but for most of the models, train and test data was having a high variance, and the model was overfitting.
- Only Ada Boost regressor worked well out of all the models, as there was less difference between train score and test score and RMSE was also low hence I used it as the final model and have done further processing.
- **After applying hyperparameter tuning I got an accuracy(r2_score) of 86% from the Ada Boost Regressor model which is a good score. Then I applied that score to the test dataset to get the target variable which is price.**

Thanks!