House Prices: Advanced Regression Techniques

Predict sales prices and practice feature engineering, RFs, and gradient boosting

by a1804938

School of Computer Science, The University of Adelaide

Report submitted for **4020_COMP_SCI_7209 Big Data Analysis and Project** at, University of Adelaide towards the Master of Data Science

Abstract

Testing accuracy of 87.04% was achieved using LGBM Regressor while a testing accuracy of 86.62% was achieved using XGB regressor. Model was trained on data acquired from Ames Housing Dataset compiled by Dean De Cock[1]. It is achieved by intensive Explanatory data analysis by dividing the data into categorical and numerical values, handling of missing and null data, applying encoding on the categorical data and then fitting the data into a model to calculate it's accuracy and root square result.

Introduction

In this world everyone dreams of owning a house and based on that dream is how the real estate market survives and hence making it the biggest and one of the most important decision an individual have to make in their lifetime. This project is aimed at predicting the housing sale prices in Ames, by using various features and aspects of any individual home. The various aspects and the process of training the data givens any individual insight on making the decision to make a purchase helping them to maximize their gain keeping in mind about their budget. These findings also help real estate agents in improving their sales based on marketing of some key features.

Main street in downtown Ames, Photo by Tim Kiser

The Ames Housing Data Set is an alternative to one of the most known and worked on dataset - Boston Housing Data Set. The Ames Housing Dataset was collected by a Professor at Truman State University, Dean De Cock, in 2011. This dataset comprises of residential properties in the town of Ames, ranging from year 2006 to 2010, consisting of 80 features with 2930 samples. This dataset is divided into training and test data sets. Training data set comprises of 1460 observations and testing dataset comprises of 1459 observations.

From the dataset following details can be understood:

- A total of 80 features of which is divided into categorical and numerical data.
- There are total of 46 categorical features ranging from 2 to 28 classes like largest neighborhood etc.
- There are 14 discrete items like kitchens, baths etc.
- There are 20 continuous items like square footage, lot size etc.

Categorical features [2] are the ones which might contain few distinct groups or finite number of categories and not necessarily have a logical order to the data. Categorical data examples are like gender or mode of payment. Gender maybe stored as male = 1 or female = 0, which can't be used to directly perform arithmetic operations.
Numerical data features are like quantitative characteristics which are expressed as quantitative values and can perform some arithmetic operations.
Numerical data is further divided into discrete and continuous variables.
Discrete variable is the data that have countable number of values between 2 values. For example, number of defects or issues.
Continuous variable is the data that have an infinite number of values between 2 values which either in the form of numbers or date time. For example, a transaction data and time.

The training dataset includes the sale price, hence making it a supervised regression machine learning task. A supervised task is one where we have access to both target and features. We have to train a model which can do mapping between both features and target. Sale price which is the target is a continuous variable. The model should be accurate i.e. it should be able to predict the sale price nearest to its actual value and the model predictions should be interpretable.

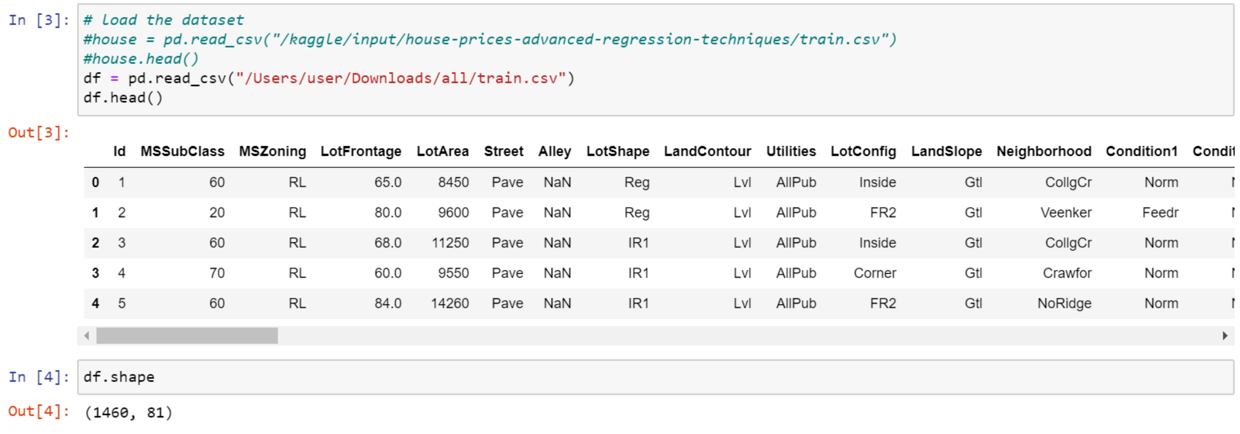
Step by step methodology that can be followed for the analysis of this dataset:

1. EDA - Exploratory data analysis
2. Data handling – formatting and cleaning
3. Feature engineering
4. Try to compare various machine learning models for performance
5. Perform parameter tuning for the selected model
6. Train models on data set
7. Submit predictions

Data formatting and cleaning take care of:

- Null or missing values
- Wrong datatype
- Wrong data
- Outliers
- Skewed distributions

Exploratory data analysis



```
In [3]: # Load the dataset
        #house = pd.read_csv("/kaggle/input/house-prices-advanced-regression-techniques/train.csv")
        #house.head()
        df = pd.read_csv("/Users/user/Downloads/all/train.csv")
        df.head()
```

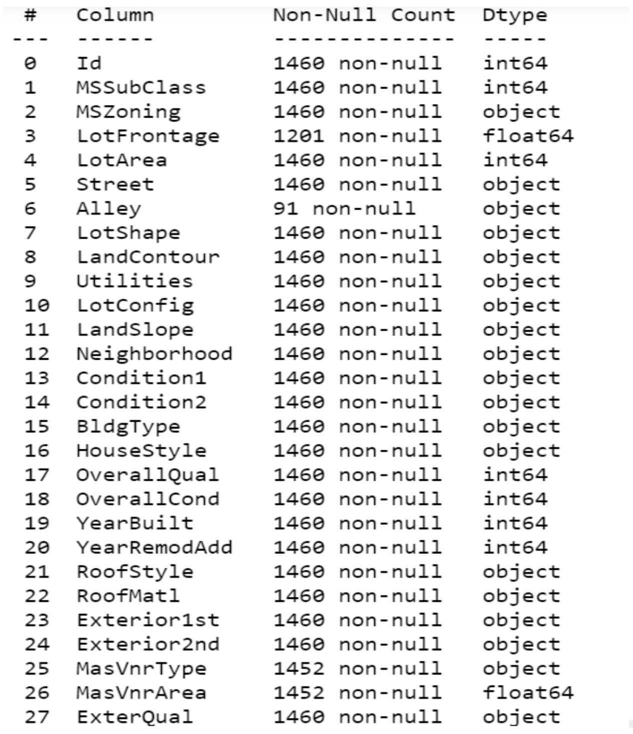
Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condi
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	N
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	N
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	N
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	N
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	N

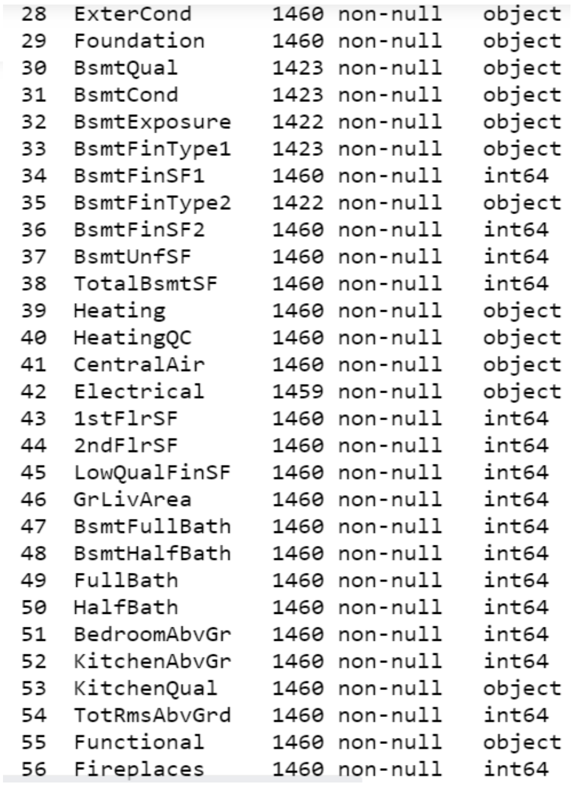
```
In [4]: df.shape

Out[4]: (1460, 81)
```

We have a total of 81 columns. Target variable is SalePrice that is to be predict. Id column is just an index which is to be ignored. So we have a total 79 input features to predict the output from.



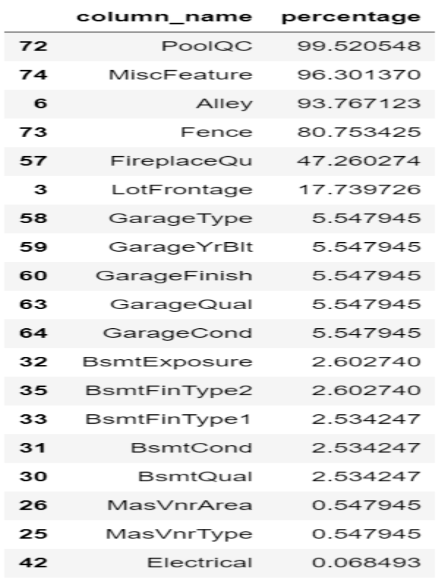
```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              1460 non-null   int64
 1   MSSubClass      1460 non-null   int64
 2   MSZoning        1460 non-null   object
 3   LotFrontage     1201 non-null   float64
 4   LotArea         1460 non-null   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
```

We can see that there's some logical order in the columns to go from general information to detailed ones and describe the neighborhood area and outside area first like Street, alley, LotArea, and then the inside information which goes from basement, heating, floors to rooms etc. with final information about the sale.

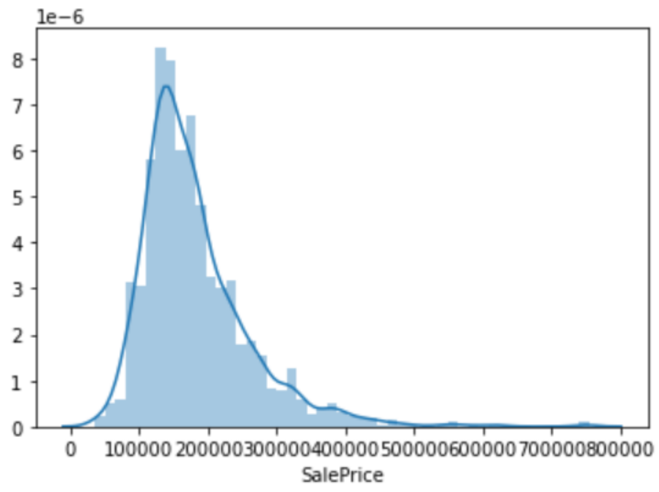
There are lots of null values present in the dataset. 4 of these features have missing values more than 50%.



	column_name	percentage
72	PoolQC	99.520548
74	MiscFeature	96.301370
6	Alley	93.767123
73	Fence	80.753425
57	FireplaceQu	47.260274
3	LotFrontage	17.739726
58	GarageType	5.547945
59	GarageYrBlt	5.547945
60	GarageFinish	5.547945
63	GarageQual	5.547945
64	GarageCond	5.547945
32	BsmtExposure	2.602740
35	BsmtFinType2	2.602740
33	BsmtFinType1	2.534247
31	BsmtCond	2.534247
30	BsmtQual	2.534247
26	MasVnrArea	0.547945
25	MasVnrType	0.547945
42	Electrical	0.068493

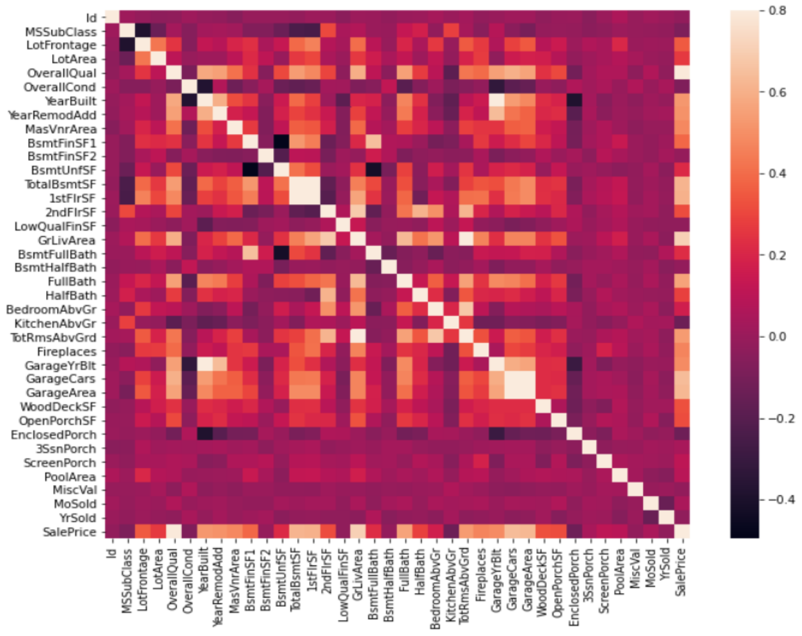
```python
#distribution of saleprice
sns.distplot(df.SalePrice)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x20a0b9f11c0>
```

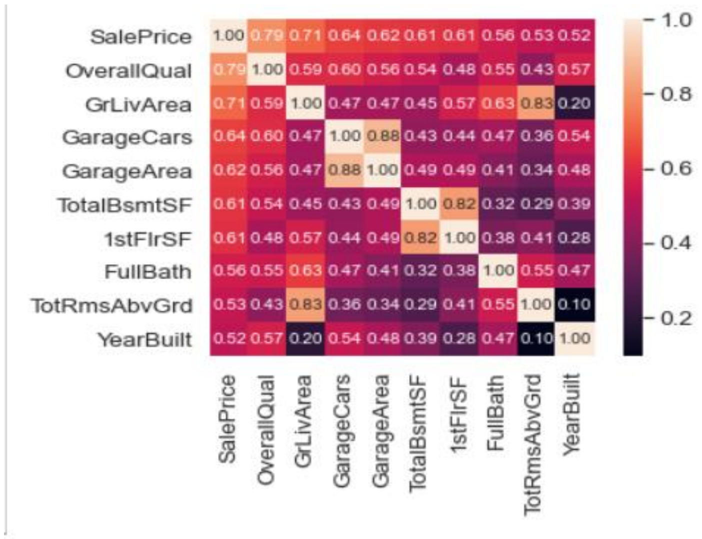


SalePrice seems to be skewed. Skewness is the amount of distortion when compared to symmetrical bell curve or normal distribution measuring the lack of symmetry in data distribution.[3] The values seem they aren't normally distributed.

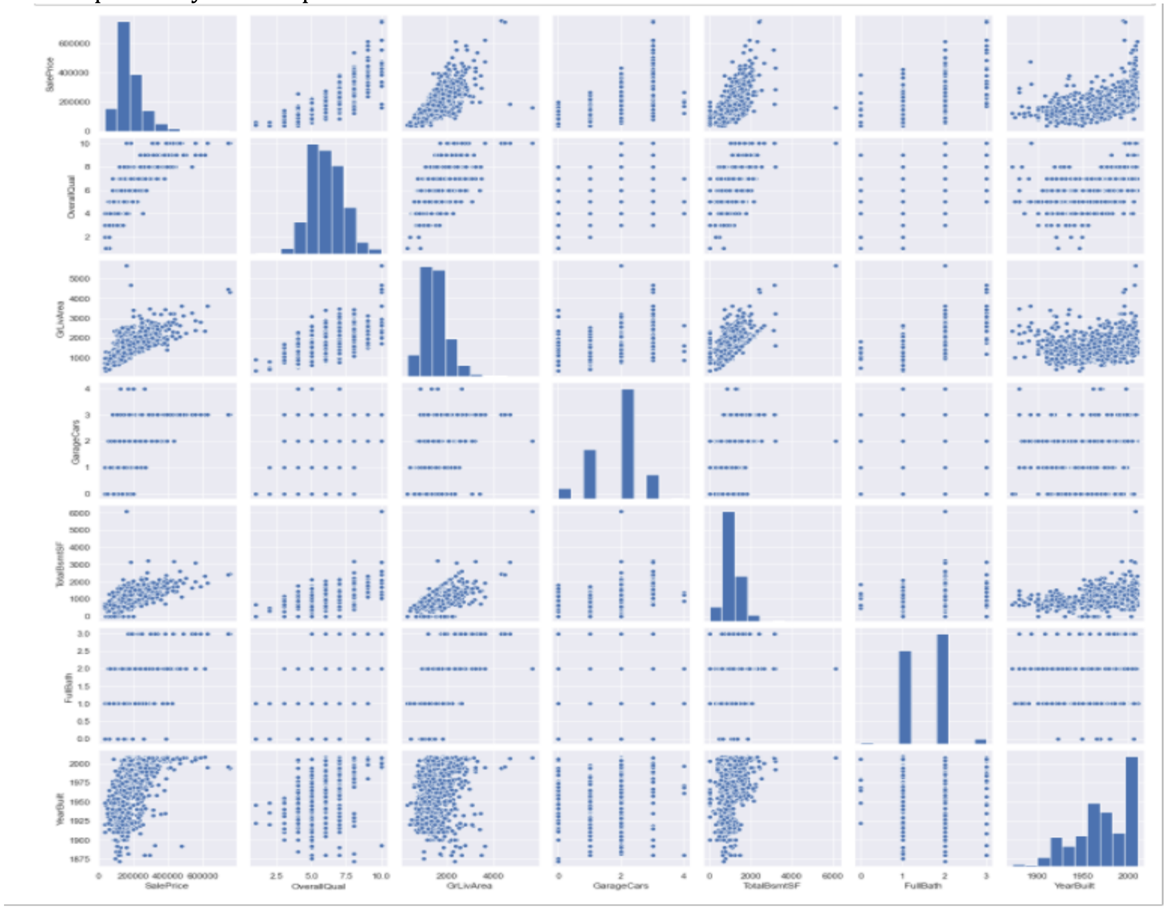
Correlation of the data can be seen as below:



The correlation of saleprice with 9 other important columns can be seen as below:

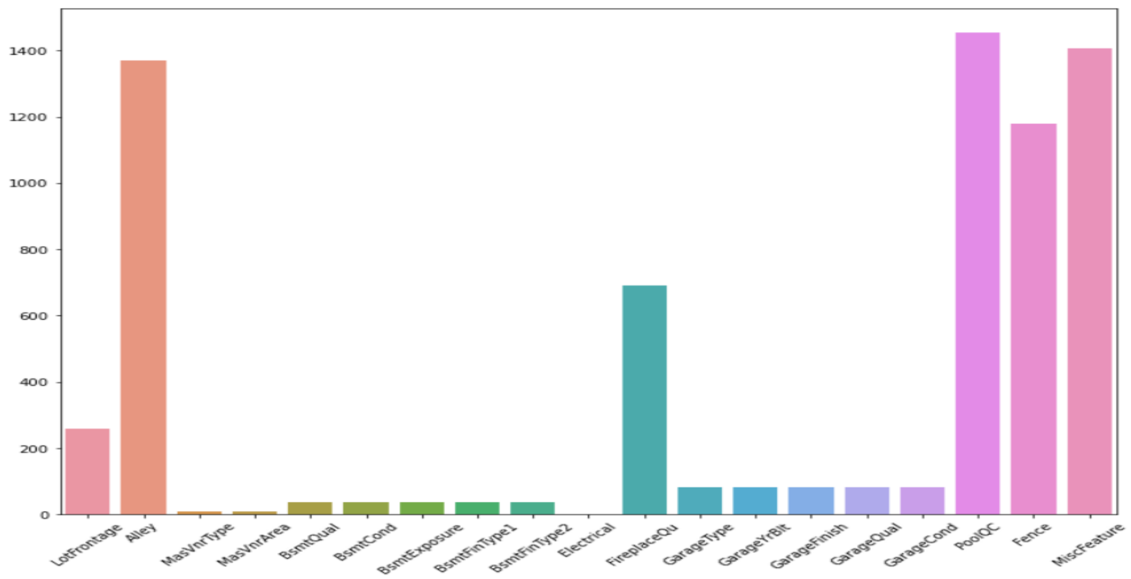


Then we can see the scatter plot of these variables with salesprice to have a better understanding of the data and their dependency on salesprice.



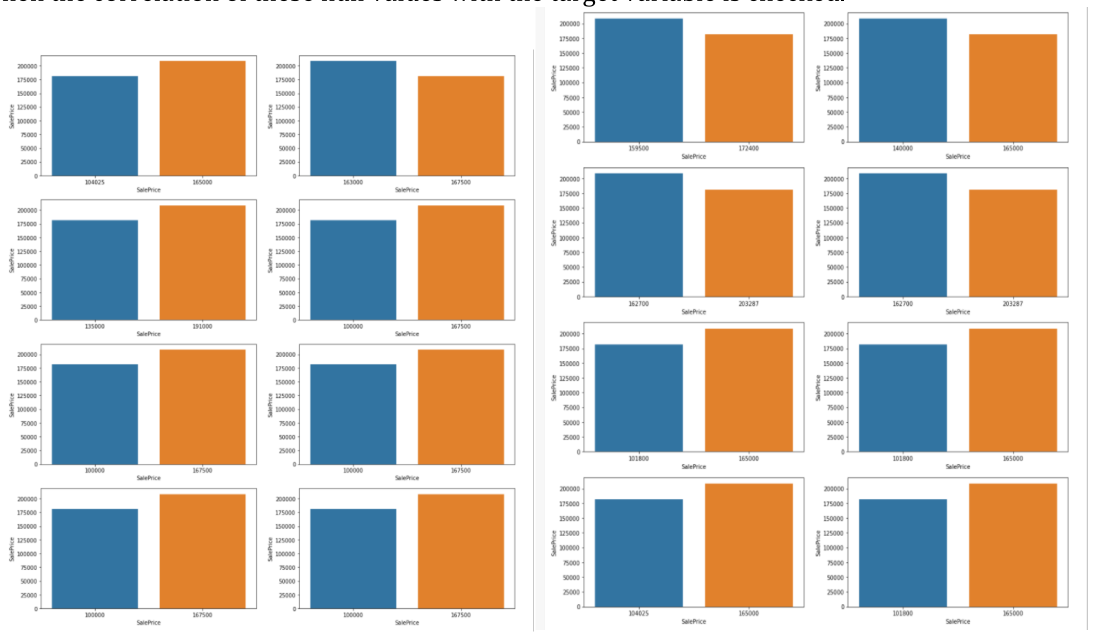
Then the Id column is dropped from the dataset as it is not required for the processing of the model.

Then null values handling is done. First checked the null values in the columns:



After checking the data dictionary, these are not actually the null values, rather these are the features which are not present in the house.
For example, let check the field Alley, Value "NA: here means house has "No Alley Access"

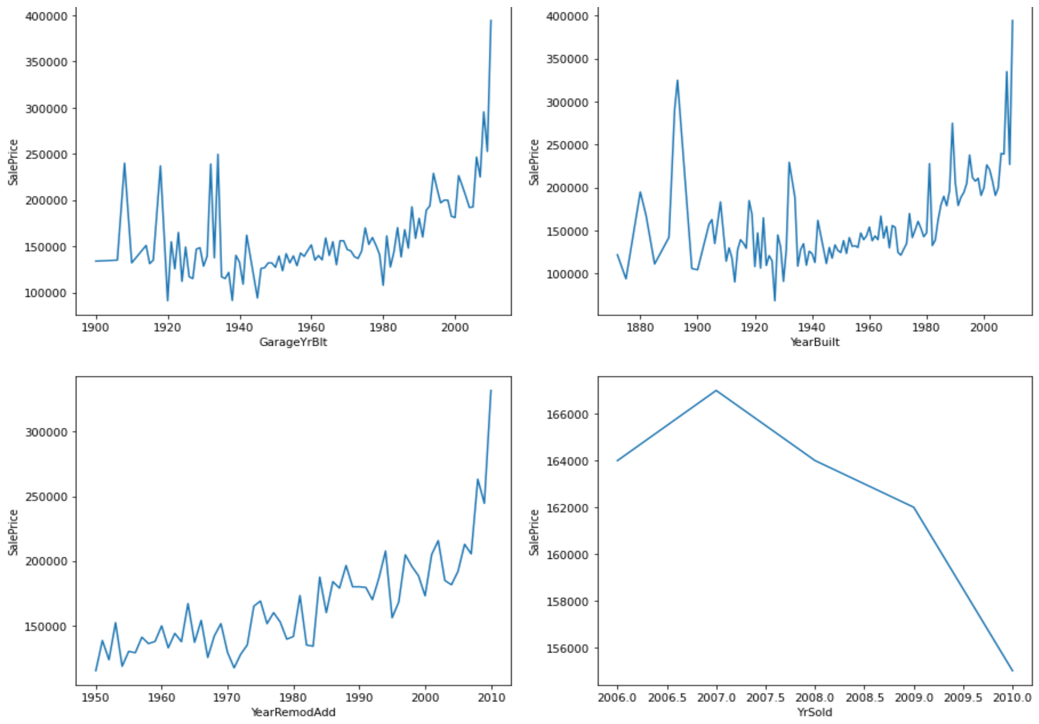
Then the correlation of these null values with the target variable is checked.



From the above graphs, we can clearly see that the null values have strong relation with the SalePrice, hence we can niether drop the columns with null values, nor we can drop the rows with null values.

To handle these null and missing values below steps are performed:
1) All missing values for the categorical columns will be replaced by "None"
2) All missing values for the numeric columns will be replaced by median of that field

Then the date variables are examined. The following are the date variables:
GarageYrBlt', 'YearBuilt', 'YearRemodAdd', 'YrSold'
The relation of these variables are examined with the salesprice.

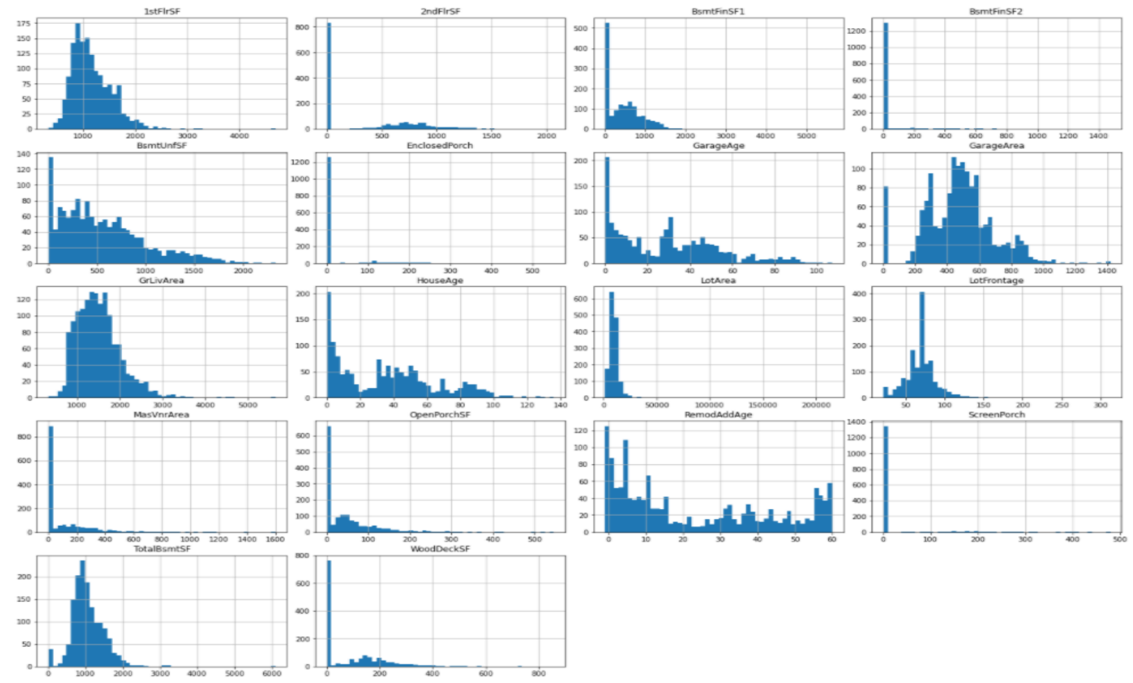


The trend of sale price with the feature "YrSold", it shows a decreasing trend. In real estate this seems to be unreal because as time passes the price is expected to increase which seems to be opposite over here. This is handled by creating a variable "Age" from these variables.

3 new parameters are created:
Age variable:- 'HouseAge' = 'YrSold' - 'YearBuilt'
Age of master after remodeling:- 'RemodAddAge' = 'YrSold' - 'YearRemodAdd'
Creating age of the garage from year built of the garage to the sale of the master
'GarageAge' = 'YrSold' - 'GarageYrBlt'
And then the original variables: "YearBuilt","YearRemodAdd","GarageYrBlt" are dropped from the dataset.

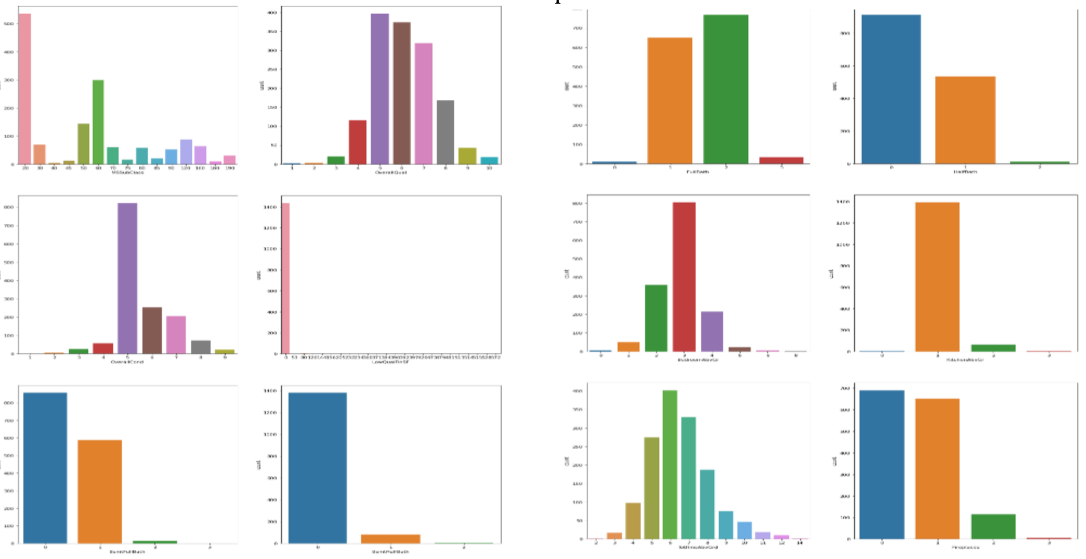
Then a separate lists of categorical and numeric columns are created based on the datatypes. Then the numeric features are separated into continuous and discrete numeric features.

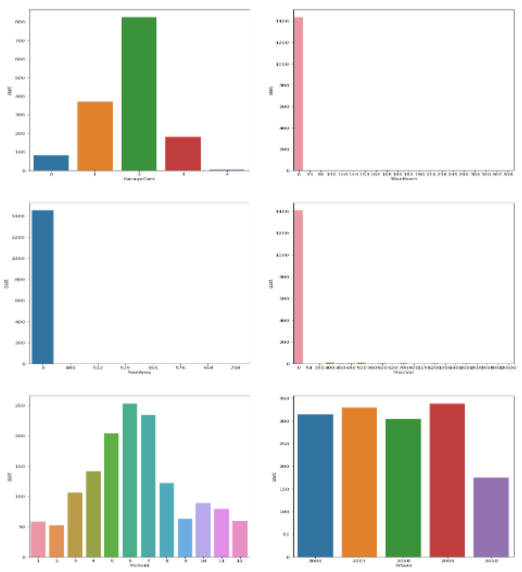
Then the variance in the different continuous numeric columns present in the dataset is examined which can be seen in the below scatter plot.



Then the variance in numbers is examined to find out the list of features that doesn't have much effect on the target variables by checking the count of the values of those column values. Following variables seems to have low variance:
MasVnArea, BsmtFinSF2, 2ndFlrSF, EnclosedPorch, ScreenPorch.

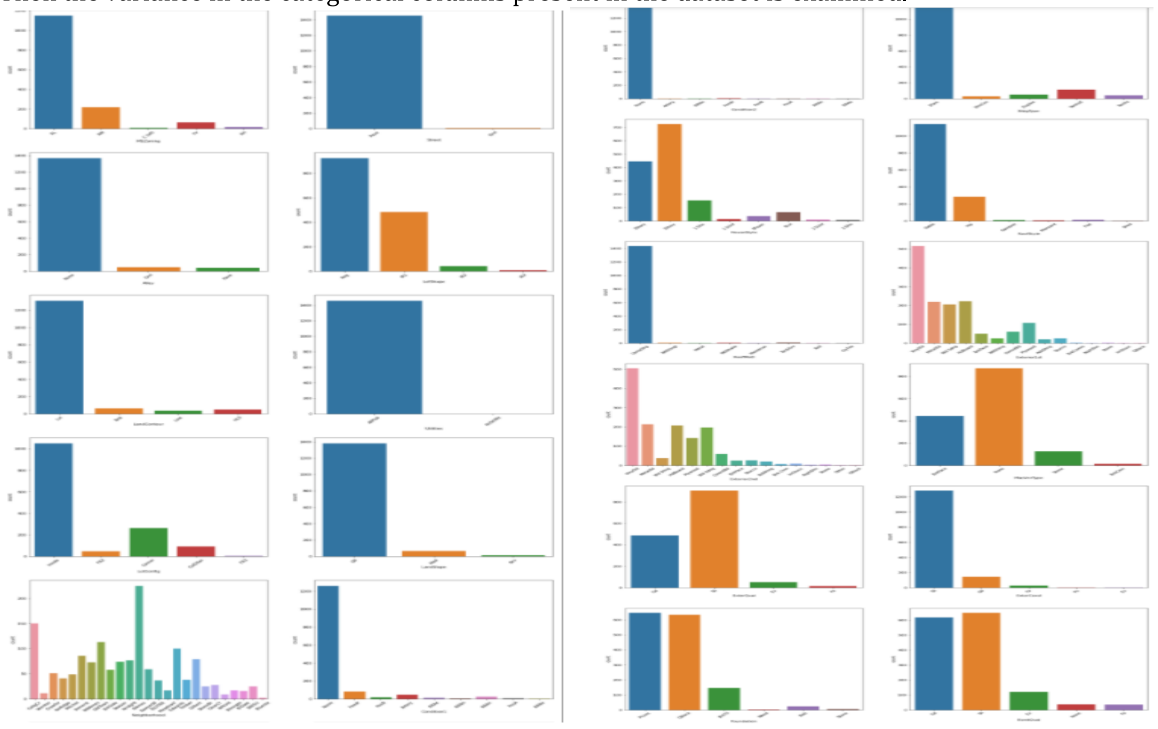
Then the variance in the discrete numeric columns present in the dataset is examined.





The following variables seems to have low variance:
LowQualFinSF, BsmtHalfBath, KitchenAbvGr, 3SsnPorch, PoolArea, MiscVal.

Then the variance in the categorical columns present in the dataset is examined.



Following variables seems to have low variance:

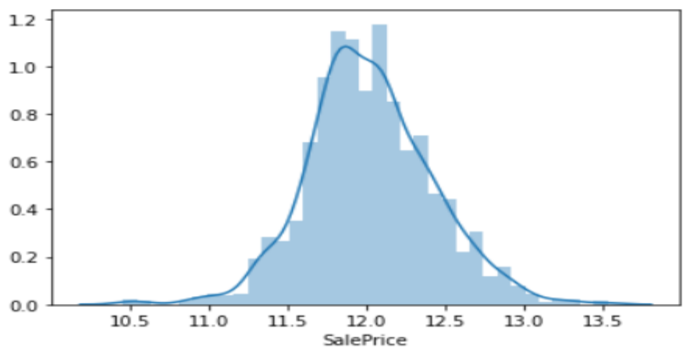
MSZoning Street, Alley LandContour, Utilities, LotConfig Condition1 LandSlope Condition2, BldgType RoofStyle, RoofMatl, ExterCond, BsmtCond, BsmtFinType2, Heating, CentralAir, Electrical Functional GarageQual, GarageCond, PavedDrive, PoolQC, Fence, MiscFeature, SaleType, SaleCondition.

All the columns which have low variables as identified are dropped from the dataset.

Then the skewness in saleprice, is handled by taking the log to get normal distribution which becomes like below.

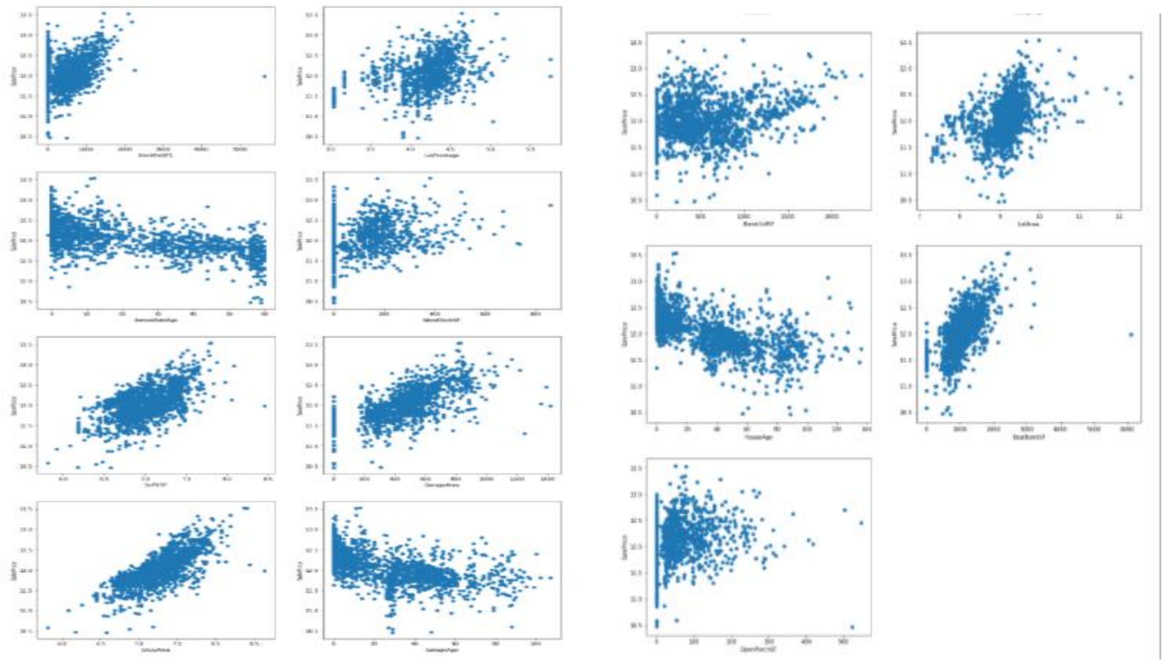
```
sns.distplot(df.SalePrice)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13e87c97c
```



Also the log of few numeric variables ‘LotFrontage', 'LotArea', '1stFlrSF', 'GrLivArea' to is done to handle skewness.

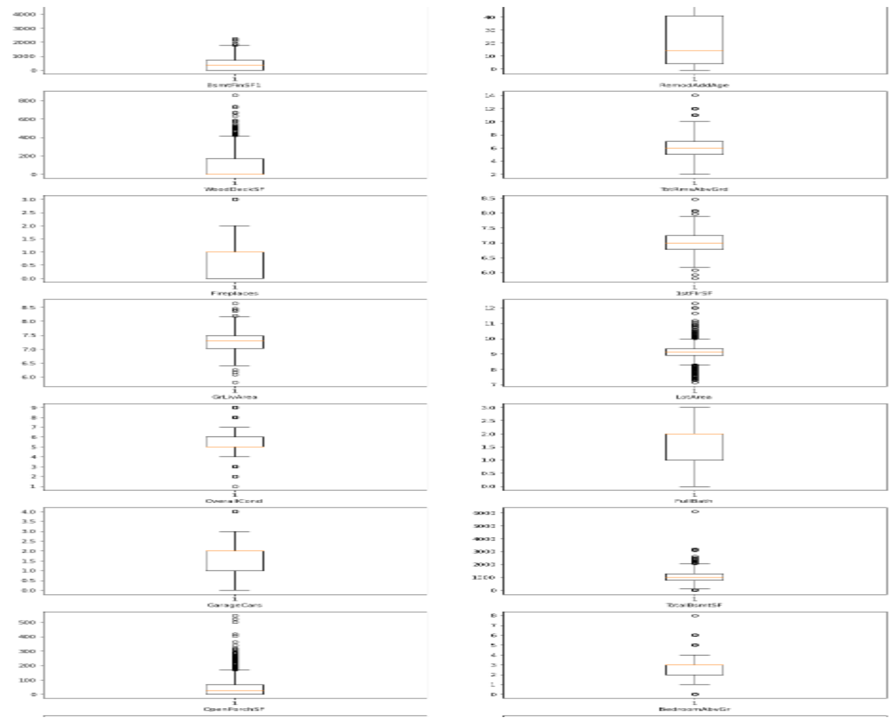
Then we examine how SalePrice varies with Continous numeric features in dataset.



Most of the features above seems to have a good relation with SalePrice but there are some outliers present which need to be treated

After checking Discrete numeric variables with respect to salesprice in the dataset we can drop MSSubClass, YrSold & MoSold as they have no impact on SalePrice.

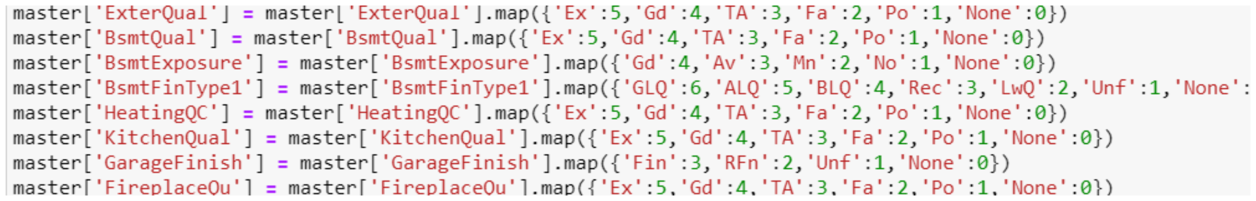
After all the columns are dropped from the dataset which have very less variance, then we then examine the outliers.



There are outliers in the dataset which are treated by taking the quantile of those variables.
OpenPorchSF, GarageArea, TotalBsmtSF, BsmtUnfSF, WoodDeckSF and BsmtFinSF1.
The outiers are handled by the quantile measurement of position. These numbers indicate where a specified proportion of data lies. Median is the middle position. Similarly, 25% of the data have values less than the 1st quartile whereas the 75% of data have values less than 3rd quartile.

After the outliers are handled, feature engineering on the test dataset is done where the steps like removing the id column, dividing the features into categorical and numerical data and then further dividing the numerical data into discrete and continuous features is done. Then the datetime features are handled in the same way and handling the skewness in the numeric variables is done.

Then a master dataset is created with the concatenation of training dataset and test dataset to apply encoding for the categorical variables. Encoding will basically convert the categorical variables to numeric variables. First the ordinal variables are treated first by applying integer encoding which is used to convert ordinal variables to numerical data.



```
master['ExterQual'] = master['ExterQual'].map({'Ex':5,'Gd':4,'TA':3,'Fa':2,'Po':1,'None':0})
master['BsmtQual'] = master['BsmtQual'].map({'Ex':5,'Gd':4,'TA':3,'Fa':2,'Po':1,'None':0})
master['BsmtExposure'] = master['BsmtExposure'].map({'Gd':4,'Av':3,'Mn':2,'No':1,'None':0})
master['BsmtFinType1'] = master['BsmtFinType1'].map({'GLQ':6,'ALQ':5,'BLQ':4,'Rec':3,'LwQ':2,'Unf':1,'None':
master['HeatingQC'] = master['HeatingQC'].map({'Ex':5,'Gd':4,'TA':3,'Fa':2,'Po':1,'None':0})
master['KitchenQual'] = master['KitchenQual'].map({'Ex':5,'Gd':4,'TA':3,'Fa':2,'Po':1,'None':0})
master['GarageFinish'] = master['GarageFinish'].map({'Fin':3,'RFn':2,'Unf':1,'None':0})
master['FireplaceQu'] = master['FireplaceQu'].map({'Ex':5,'Gd':4,'TA':3,'Fa':2,'Po':1,'None':0})
```

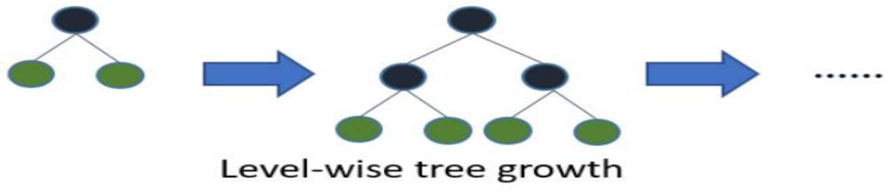
Then for the remaining categorical variables are encoded. Dummy Variable Encoding which his is almost same as One-Hot Encoding, with one less column. There is some redundancy in One-Hot encoding. For instance, a person is either male or female. So we only need to use one of these two dummy-coded variables as a predictor. More generally, the number of dummy-coded variables needed is one less than the number of possible values, which is K-1. In statistics, this is called a dummy encoding variable, or dummy variable.
By default, the get_dummies() does not do dummy encoding, but One-Hot encoding. To produce an actual dummy encoding from a DataFrame, we need to pass drop_first=True. [4]

Since all the necessary operations on the train and test datasets are performed, then the dataset is again separated into train and test subset. The testing subset is to build the model. The testing subset is to use the model on unknown data for evaluation of the performance of the selected model.
Then the train_test_split function is used to divide the train set into train and test data in the ratio of 70:30.

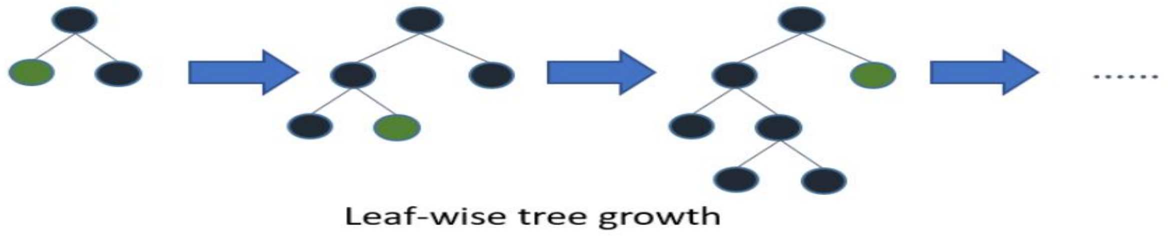
Once the train data has been split into test and train subset, a model is applied on the dataset.
First XGBoost is applied. XGBoost stands for "Extreme Gradient Boosting" and is an application of gradient boosting trees algorithm. The XGBoost is a popular SVM learning model with features like speed, parallelization, and performance. Booster stands for the regression booster type. In the model applied gbtree was used as this is a tree-based booster which gives better result compared to a linear model. Learning rate stands for how quick the model learns. The slower the learning rate, more better and generalized predictions are going to be and hence, 0.1 is the learning rate. Then max_depth will be the maximum depth of the tree which is 3 in our model. Then N_estimators is the count of trees generated to get the output from the model. Number need to be average as we can face overfitting or underfitting of data. Reg_alpha and reg_lambda these are the regularization values. num_leaves will be the maximum number of leaves for weaker model.

Light GBM is a fast, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification. It is a decision tree algo as well, which splits the tree leaf wise with the best available fit. So when growing on the same leaf in Light GBM, it reduces more loss than the level-wise algorithm which ultimately results in higher accuracy which isn't achieved by existing boosting algorithms that's why it has the name Light.

Before is a diagrammatic representation by the makers of the Light GBM to explain the difference clearly.



Level-wise tree growth in XGBOOST.



Leaf wise tree growth in Light GBM.

Analysis:

The coefficient of determination, which is r2 or R2 also known as R squared is applied to predict the score of the models. It is the proportion of the variance of dependent features from the target features.
The R squared prediction from XGB model is 86.62%.
The R squared prediction from LGBM model is 87.04%.
The output is saved as a csv file and before the final output is printed, the exponential values of XGB and LGBM is passed as to handle the skewness logarithmic evaluation of data was done.
In this experiment, we used 50% of XGB and 50% of LGBM values.
The kaggle score for the output file was 0.13424 ranking 2041 on the leader board.

References

[1] https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[2]https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/what-are-categorical-discrete-and-continuous-variables/.

[3] https://towardsdatascience.com/skewed-data-a-problem-to-your-statistical-model-9a6b5bb74e37#:~:text=A%20data%20is%20called%20as%20skewed%20when%20curve,For%20example%2C%20below%20is%20the%20Height%20Distribution%20graph.

[4] https://towardsdatascience.com/what-is-one-hot-encoding-and-how-to-use-pandas-get-dummies-function-922eb9bd4970

https://www.thoughtco.com/what-is-a-quantile-3126239

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

https://harrymoreno.com/2018/10/15/gradient-boosting-decisions-trees-xgboost-vs-lightgbm.html