HOUSE PRICE PREDICTION PROJECT

Submitted by:

Akanksha Padhye

ACKNOWLEDGMENT

I have gave my best in this project. I am highly indebted to Flip Robo Technologies Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I want to thank my SME Ms. Khushboo Garg for providing the Dataset and helping us to solve the problem and addressing out our Query in right time.

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.
- We are 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.

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 data is provided in the CSV file below.

- 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 a variable?
    - How do these variables describe the price of the house?

Motivation for the Problem Undertaken

- Our main objective of doing this project is to build a model to predict the house prices with the help of other supporting features. We are going to predict by using Machine Learning algorithms.
- The sample data is provided to us from our client database. In order to improve the selection of customers, the client wants some predictions that could help them in further investment and improvement in selection of customers.
- House Price Index is commonly used to estimate the changes in housing price. Since housing price is strongly correlated to other factors such as location, area, population, it requires other information apart from HPI to predict individual housing price.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem

- We are building a model in Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. So, this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house. This will help to determine the price of houses with the available independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.
- Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For specific mathematical reasons this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values.
- Regression analysis is also a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

  The different Mathematical/Analytical models that are used in this project are as below:

  1.Linear regression - is a linear model, e.g., a model that assumes a linear relationship between the input variables (x) and

the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

2. Lasso - In statistics and machine learning, lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

3. Ridge - regression is a way to create a parsimonious model when the number of predictor variables in a set exceeds the number of observations, or when a data set has multi co linearity (correlations between predictor variables).

4. Elastic Net - is a popular type of regularized linear regression that combines two popular penalties, specifically the L1 and L2 penalty functions. Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve on the regularization of statistical models.

5. K Neighbors Regressor - KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

6. Decision Tree - is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application. It is a tree-structured classifier with three types of nodes.

7. Random forest - is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and

control over-fitting. A Random Forest's nonlinear nature can give it a leg up over linear algorithms, making it a great option.

8. AdaBoost Regressor - is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

9. Gradient Boosting Regressor - GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

- First, use the train dataset and do the EDA process, fitting the best model and saving the model.
- Then, use the test dataset, load the saved model and predict the values over the test data.

Data Sources and their formats

- Dataset has 1168 rows and 81 columns.
- There are null values present in the dataset.
- Dataset contains categorial and continuous data.

In [2]:
```
#Loading the file
df=pd.read_csv('train.csv')
df
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

Data description

Data contains 1460 entries each having 81 variables. The details of the features are given below:
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

CHECKING THE NUMBER OF NULL VALUES & HANDELING THEM:

```
In [17]: 1 df.isnull().sum().sort_values().tail(20)
```

```
Out[17]: Exterior1st       0
         Exterior2nd       0
         MasVnrType        7
         MasVnrArea        7
         BsmtQual         30
         BsmtCond         30
         BsmtFinType1     30
         BsmtFinType2     31
         BsmtExposure     31
         GarageQual       64
         GarageFinish     64
         GarageYrBlt      64
         GarageCond       64
         GarageType       64
         LotFrontage     214
         FireplaceQu     551
         Fence           931
         Alley          1091
         MiscFeature    1124
         PoolQC         1161
         dtype: int64
```

```
for i in b:
    df[i].fillna('No_Basement',inplace=True)
    print(df[i].value_counts())

df['MiscFeature'].fillna('None',inplace=True)
print(df['MiscFeature'].value_counts())

df['Alley'].fillna('No_alley_access',inplace=True)
print(df['Alley'].value_counts())

df['Fence'].fillna('No_Fence',inplace=True)
print(df['Fence'].value_counts())

df['FireplaceQu'].fillna('No_Fireplace',inplace=True)
print(df['FireplaceQu'].value_counts())

df['LotFrontage'].fillna(df['LotFrontage'].median(),inplace=True)

g=['GarageType','GarageFinish','GarageQual','GarageCond']
for i in g:
    df[i].fillna('No_Garage',inplace=True)
    print(df[i].value_counts())

df["GarageYrBlt"]=df["GarageYrBlt"].fillna(df["YearBuilt"])
print(df['GarageYrBlt'].value_counts())

df['MasVnrArea'].fillna(0,inplace=True)
print(df['MasVnrArea'].value_counts())

df['MasVnrType'] = df['MasVnrType'].fillna('None')
```

DATA PREPROCESSING

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre- process our data before feeding it into our model.

STATISTICAL SUMMARY

In descriptive statistics, summary statistics are used to summarize a set of observations, in order to communicate the largest amount of information as simply as possible. Summary statistics summarize and provide information about your sample data. It tells something about the values in data set. This includes where the average lies and whether the data is skewed. The describe() function computes a summary of statistics pertaining to the Data Frame columns. This function gives the mean, count, max, standard deviation and IQR values of the dataset in a simple understandable way.

```
n [12]:
#Statistical summary
df.describe().transpose()
```

ut[12]:

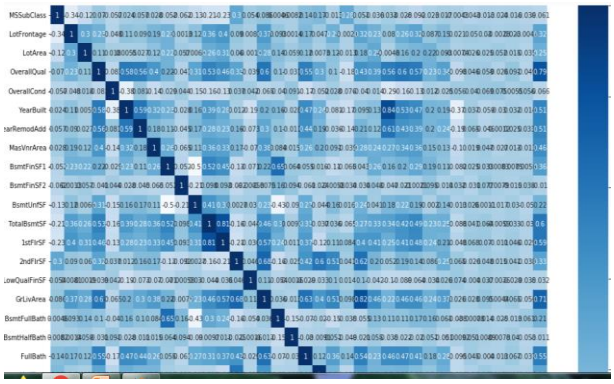
	count	mean	std	min	25%	50%	75%	max
MSSubClass	1168.0	56.767979	41.940650	20.0	20.00	50.0	70.00	190.0
LotFrontage	1168.0	70.807363	22.440317	21.0	60.00	70.0	79.25	313.0
LotArea	1168.0	10484.749144	8957.442311	1300.0	7621.50	9522.5	11515.50	164660.0
OverallQual	1168.0	6.104452	1.390153	1.0	5.00	6.0	7.00	10.0
OverallCond	1168.0	5.595890	1.124343	1.0	5.00	5.0	6.00	9.0
YearBuilt	1168.0	1970.930651	30.145255	1875.0	1954.00	1972.0	2000.00	2010.0
YearRemodAdd	1168.0	1984.758562	20.785185	1950.0	1966.00	1993.0	2004.00	2010.0
MasVnrArea	1168.0	101.696918	182.218483	0.0	0.00	0.0	160.00	1600.0
BsmtFinSF1	1168.0	444.726027	462.664785	0.0	0.00	385.5	714.50	5644.0
BsmtFinSF2	1168.0	46.647260	163.520016	0.0	0.00	0.0	0.00	1474.0
BsmtUnfSF	1168.0	569.721747	449.375525	0.0	216.00	474.0	816.00	2336.0
TotalBsmtSF	1168.0	1061.095034	442.272249	0.0	799.00	1005.5	1291.50	6110.0
1stFlrSF	1168.0	1169.860445	391.161983	334.0	892.00	1096.5	1392.00	4692.0
2ndFlrSF	1168.0	348.826199	439.696370	0.0	0.00	0.0	729.00	2065.0
LowQualFinSF	1168.0	6.380137	50.892844	0.0	0.00	0.0	0.00	572.0
GrLivArea	1168.0	1525.066781	528.042957	334.0	1143.25	1468.5	1795.00	5642.0
BsmtFullBath	1168.0	0.425514	0.521615	0.0	0.00	0.0	1.00	3.0

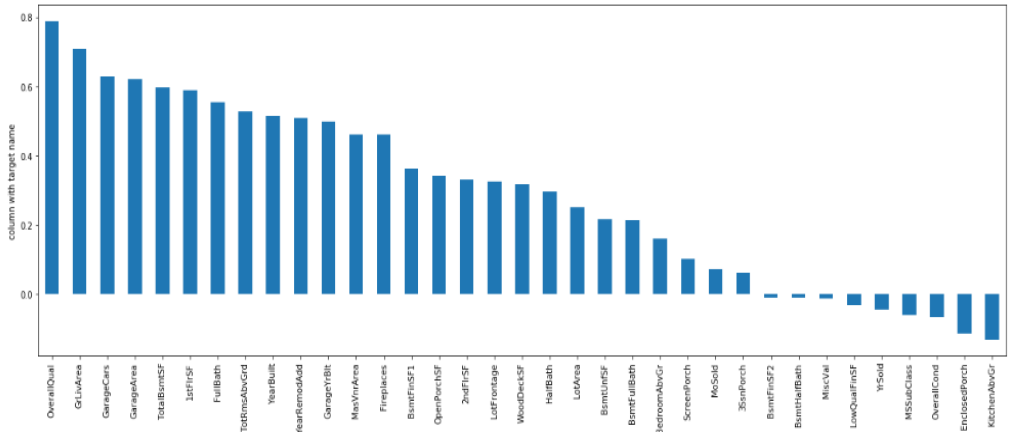
Observation:

- Count is same for all the columns.
- For some columns mean is greater than 50% and for some columns mean is lesser than 50%.
- For all the columns there is a difference between max and 75%.

Correlation Factor

The statistical relationship between two variables is referred to as their correlation. The correlation factor represents the relation between columns in a given dataset. A correlation can be positive, meaning both variables are moving in the same direction or it can be negative, meaning that when one variable's value increasing, the other variable's value is decreasing.





Observation:

- 'OverallQual' & 'GrLivArea' are highly positively correlated with target column
- 'EnclosedPorch' & 'KitchenAbvGr' are highly negatively correlated with target column.
- 'MSSubClass' ,'OverallCond' ,'OverallCond' ,'LowQualFinSF' ,'BsmtHalfBath' ,'YrSold', 'MiscVal', 'MoSold', '3SsnPorch' are least correlated with the target column.

DROPPING UNNECESSARY COLUMNS AND ENCODING NON-NUMERIC DATA USING LABEL ENCODER

```
#Dropping other unnecessary columns in the dataset
df.drop('Id',axis=1,inplace=True)
df.drop('PoolArea',axis=1,inplace=True)
df.drop('PoolQC',axis=1,inplace=True)
df.drop(['Utilities'],axis=1,inplace=True)
```

```
#Dropping the least correlated columns from the dataset
df.drop(['MSSubClass' ,'LowQualFinSF' ,'BsmtHalfBath' ,'BsmtUnfSF' ,'YrSold', 'MiscVal',
         'MoSold', '3SsnPorch'],axis=1,inplace=True)
```

```
le=LabelEncoder()
for i in df.columns:
    if df[i].dtypes=='object':
        df[i]=le.fit_transform(df[i].values.reshape(-1,1))
```

```
df
```

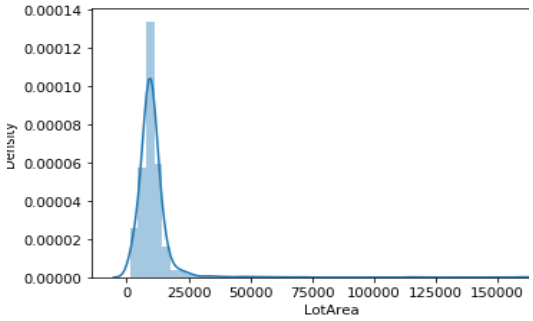
	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	...	PavedDrive	WoodDeckSF	OpenPorchS
0	3	70.0	4928	1	1	0	3	4	0	13	...	2	0	2(
1	3	95.0	15865	1	1	0	3	4	1	12	...	2	81	2(
2	3	92.0	9920	1	1	0	3	1	0	15	...	2	180	1:

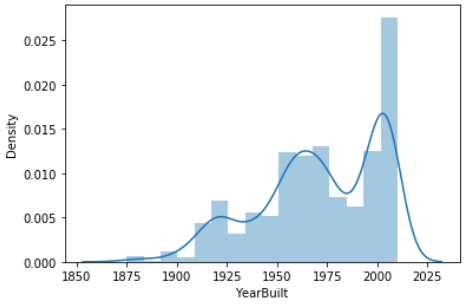
We have dropped the unnecessary columns from the dataset which is leastly correlated with the targeted column.
Using LabelEncoder() to convert the object datatype into integer datatype so that all the columns can go under the model building process.

CHECKING SKEWNESS

Skewness refers to distortion or asymmetry in a symmetrical bell curve, or normal distribution in a set of data. Besides positive and negative skew, distributions can also be said to have zero or undefined skew. The skewness value can be positive, zero, negative, or undefined.

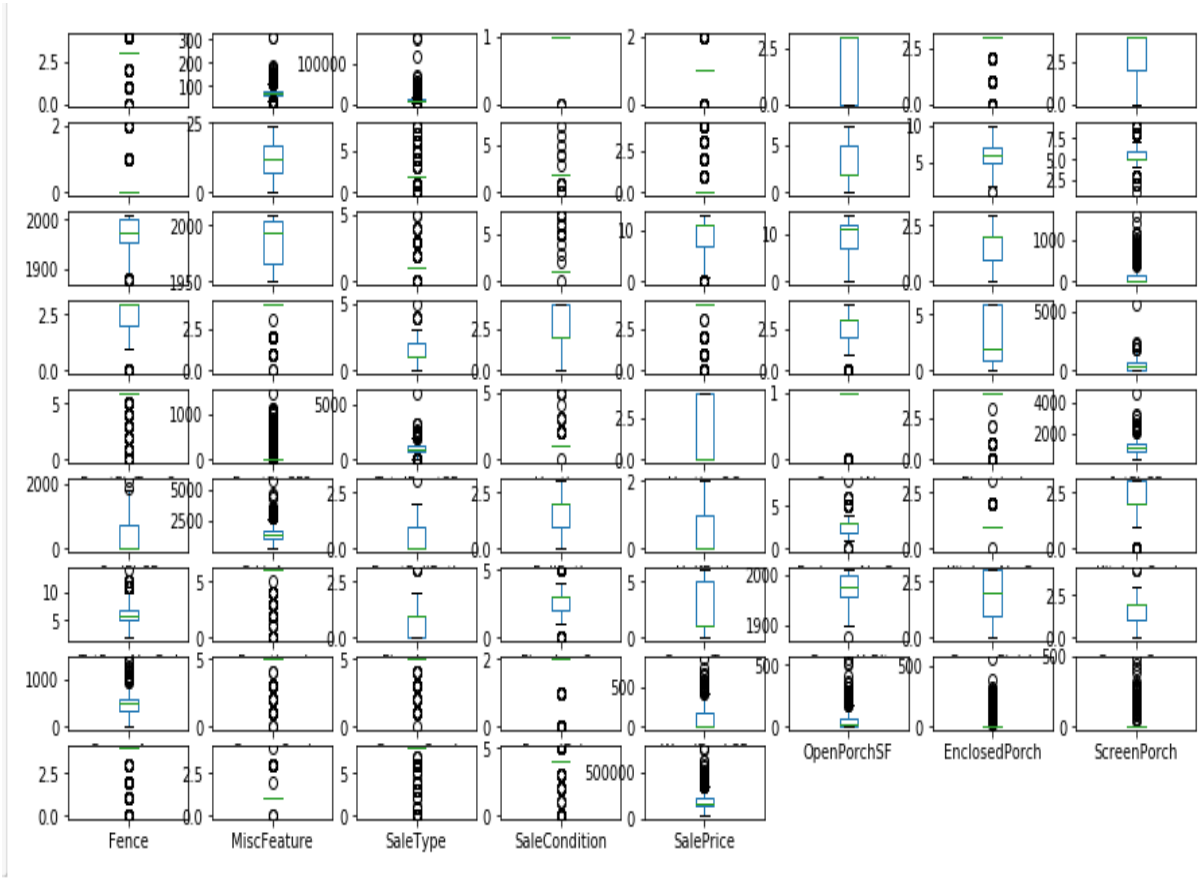
```
GarageYrBlt      -0.673163
YearBuilt        -0.579204
YearRemodAdd     -0.495864
GarageCars       -0.358556
FullBath          0.057809
OverallQual       0.175082
GarageArea        0.189665
BedroomAbvGr      0.243855
OverallCond       0.580714
BsmtFullBath      0.627106
TotRmsAbvGrd      0.644657
HalfBath          0.656492
Fireplaces        0.671966
2ndFlrSF          0.823479
GrLivArea         1.449952
WoodDeckSF        1.504929
1stFlrSF          1.513707
TotalBsmtSF       1.744591
BsmtFinSF1        1.871606
SalePrice         1.953878
OpenPorchSF       2.410840
LotFrontage       2.733440
MasVnrArea        2.835718
EnclosedPorch     3.043610
ScreenPorch       4.105741
KitchenAbvGr      4.365259
BsmtFinSF2        4.365829
LotArea          10.659285
```





CHECKING OUTLIERS

An outlier is a data point in a data set which is distant or far from all other observations available. It is a data point which lies outside the overall distribution which is available in the dataset.



TREATING SKEWNESS & OUTLIERS

```
#removing outliers
from scipy.stats import zscore
z=np.abs(zscore(df))
z
```

```
array([[0.02164599, 0.03599365, 0.62061571, ..., 0.33003329, 0.20793187,
        0.67631017],
       [0.02164599, 1.07854983, 0.60090318, ..., 0.33003329, 0.20793187,
        1.09423443],
       [0.02164599, 0.94480461, 0.06307504, ..., 0.33003329, 0.20793187,
        1.11687211],
       ...,
       [0.02164599, 2.08675364, 0.91636244, ..., 0.33003329, 0.20793187,
        0.41705186],
       [4.76211672, 0.92762843, 0.22167034, ..., 0.33003329, 0.20793187,
        1.78922393],
       [0.02164599, 0.03599365, 0.29303823, ..., 0.33003329, 0.20793187,
        0.02179027]])
```

```
threshold=3
print(np.where(z<6))
```

```
(array([   0,    0,    0, ..., 1167, 1167, 1167], dtype=int64), array([ 0,  1,
```

```
dfnew=df[(z<6).all(axis=1)]
dfnew
```

```
#removing skewness
import sklearn
from sklearn.preprocessing import power_transform
```

```
x=power_transform(x,method='yeo-johnson')
```

```
x
```

```
array([[-0.16315395,  0.04572657, -1.27273384, ..., -0.13043251,
         0.41408049,  0.00941838],
       [-0.16315395,  1.11881527,  0.17093382, ..., -0.13043251,
         0.41408049,  0.00941838],
       [-0.16315395,  1.71643041,  0.59454473, ..., -0.13043251,
        -2.76224319,  0.00941838],
       ...,
       [-0.16315395, -2.63178506, -2.41497978, ..., -0.13043251,
         0.41408049,  0.00941838],
       [-2.95160412, -1.02150011, -0.18816839, ..., -0.13043251,
         0.41408049,  0.00941838],
       [-0.16315395,  0.04572657, -0.36051361, ..., -0.13043251,
         0.41408049,  0.00941838]])
```

```
x=pd.DataFrame(x)  #coverting numpy to panda
```

Hardware and Software Requirements and Tools Used

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used Jupyter notebook to do my python programming and analysis.
For using a csv file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:
1. Pandas- a library which is used to read the data, visualisation and analysis of data.
2. NumPy- used for working with array and various mathematical techniques.
3. Seaborn- visualization tool for plotting different types of plot.
4. Matplotlib- It provides an object-oriented API for embedding plots into applications.
5. zscore- technique to remove outliers.
6. skew ()- to treat skewed data using various transformation like sqrt, log, cube, boxcox, etc.
7. Standard scaler- I used this to scale my data before sending it to model.
8. train_test_split- to split the test and train data.
9. Then I used different classification algorithms to find out the best model for predictions.
10. joblib- library used to save the model in either pickle or obj file.

MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible problem-solving approaches (methods)

From the given dataset it can be concluded that it is a Regression problem as the output column “SalesPrice” has continuous output. So, for further analysis of the problem, we have to import or call out the Regression related libraries in Python work frame.

The different libraries used for the problem solving are:
sklearn - Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

1. sklearn.linear_model
i. Linear Regression - Linear regression - is a linear model, e.g., a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

ii. Lasso - In statistics and machine learning, lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

iii. Ridge - The regression is a way to create a parsimonious model when the number of predictor variables in a set exceeds the number of observations, or when a data set has multicollinearity (correlations between predictor variables). Ridge regression is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias.

iv. Elastic Net - It is a popular type of regularized linear regression that combines two popular penalties, specifically the L1 and L2 penalty functions. Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve on the regularization of statistical models.

2. sklearn.tree –
Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a

model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
There are several advantages of using decision trees for predictive analysis:

- Decision trees can be used to predict both continuous and discrete values i.e., they work well for both regression and classification tasks.
- They require relatively less effort for training the algorithm.
- They can be used to classify non-linearly separable data.
- They're very fast and efficient compared to KNN and other algorithms.

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree to go from observations about an item to conclusions about the item's target value.

Decision Tree Regressor - Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application. It is a tree-structured classifier with three types of nodes.

3. sklearn.ensemble
The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.
The different types of ensemble techniques used in the model are:

i. Random Forest Regressor - It is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A Random Forest's nonlinear nature can give it a leg up over linear algorithms, making it a great option. Random forest is a type of supervised learning algorithm that uses ensemble methods (bagging) to solve both regression and classification problems.

ii. AdaBoost Regressor - It is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of

the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

iii. Gradient Boosting Regressor - GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

4. sklearn.metrics - The sklearn. metrics module implements several losses, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Important sklearn.metrics modules used in the project are:

- **mean_absolute_error**
- **mean_squared_error**
- **r2_score**

5. sklearn.model_selection –

i. GridSearchCV - It is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyper parameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters. GridSearchCV combines an estimator with a grid search preamble to tune hyper-parameters. The method picks the optimal parameter from the grid search and uses it with the estimator selected by the user.

ii. cross_val_score - Cross validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data.

Testing of Identified Approaches

After completing the required pre-processing techniques for the model building data is separated as input and output columns before passing it to the train_test_split.

Scaling the data using Standard Scaler

For each value in a feature, StandardScaler subtracts the minimum value in the feature and then divides by the range. The range is the difference between the original maximum and original minimum. StandardScaler preserves the shape of the original distribution.

```
#Scaling the dataset using StandardScaler
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
scale=sc.fit_transform(x)
```

```
x
```

	0	1	2	3	4	5	6	7	8	9	...	58	59	60	61	
0	-0.163154	0.045727	-1.272734	0.0	-0.005932	-1.404519	0.315244	0.597688	-0.204609	0.201831	...	0.302855	0.290100	-0.962270	1.427281	-0.410
1	-0.163154	1.118815	0.170934	0.0	-0.005932	-1.404519	0.315244	-1.554730	-0.204609	0.517647	...	0.302855	0.290100	1.047549	1.217369	-0.410
2	-0.163154	1.716430	0.594545	0.0	-0.005932	-1.404519	0.315244	0.597688	-0.204609	0.360987	...	0.302855	0.290100	-0.962270	1.188137	-0.410
3	-0.163154	0.045727	1.572109	0.0	-0.005932	-1.404519	0.315244	-1.113363	-0.204609	0.360987	...	0.302855	0.290100	1.140126	-1.052909	-0.410
4	-0.163154	-0.581695	1.078746	0.0	-0.005932	-1.404519	0.315244	0.597688	-0.204609	-0.640768	...	0.302855	0.290100	0.853621	0.295198	-0.410
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1090	-0.163154	0.045727	0.146364	0.0	-0.005932	-1.404519	0.315244	0.597688	-0.204609	1.122826	...	0.302855	0.290100	-0.962270	-1.052909	-0.410
1091	-0.163154	-0.107912	-0.115671	0.0	-0.005932	0.731100	0.315244	0.597688	-0.204609	-0.821399	...	0.302855	-3.450478	-0.962270	1.077947	-0.410
1092	-0.163154	-2.631785	-2.414980	0.0	-0.005932	0.731100	0.315244	-1.113363	-0.204609	0.201831	...	0.302855	0.290100	0.806760	-1.052909	-0.410
1093	-2.951604	-1.021500	-0.188168	0.0	4.206058	0.731100	0.315244	0.597688	-0.204609	-0.464838	...	-3.307612	-3.450478	-0.962270	0.814508	2.437
1094	-0.163154	0.045727	-0.360514	0.0	-0.005932	-1.404519	0.315244	0.597688	-0.204609	-0.640768	...	0.302855	0.290100	0.853621	0.964695	-0.410

1095 rows × 68 columns

CHECKING THE RANDOM STATE

```
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    print(f"At random state {i},the training accuracy is:- {r2_score(y_train,pred_train)}")
    print(f"At random state {i},the training accuracy is:- {r2_score(y_test,pred_test)}")

    print("\n")
```

```
At random state 0,the training accuracy is:- 0.8857565285988633
At random state 0,the training accuracy is:- 0.8447717396351766


At random state 1,the training accuracy is:- 0.8857565285988633
At random state 1,the training accuracy is:- 0.8447717396351766


At random state 2,the training accuracy is:- 0.8857565285988633
At random state 2,the training accuracy is:- 0.8447717396351766


At random state 3,the training accuracy is:- 0.8857565285988633
At random state 3,the training accuracy is:- 0.8447717396351766


At random state 4,the training accuracy is:- 0.8857565285988633
At random state 4,the training accuracy is:- 0.8447717396351766
```

Selecting random state as 4. We are getting same values in all the random states.

Train Test Split

The train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data.

FINDING THE BEST MODEL

We are using LinearRegressor, Lasso, Ridge, ElasticNet, Support vector, Decision Tree and Kneighbors Regressor.

We are calculating r2 score, cross_val_score, std, mean_absolute error, mean_squared error, root_mean_squared error for each models.

```
    print('\n')
    std=cross_val_score(model,x,y,cv=5,scoring='r2').std()
    print('Standard Deviation: ',std)
    sd.append(std)
    print('\n')
    MAE=mean_absolute_error(y_test,pre)
    print('Mean Absolute Error: ',MAE)
    mae.append(MAE)
    print('\n')
    MSE=mean_squared_error(y_test,pre)
    print('Mean Squared Error: ',MSE)
    mse.append(MSE)
    print('\n')
    RMSE=np.sqrt(mean_squared_error(y_test,pre))
    print('Root Mean Squared Error: ',RMSE)
    rmse.append(RMSE)
    print('\n\n')
```

```
r2_score:  0.8893622578983588

cross_val_score:  0.8541079525089728

Standard Deviation:  0.018812259300546526

Mean Absolute Error:  17241.69327883252
```

```
r2_score:  0.889345331073221

cross_val_score:  0.8541861240049167

Standard Deviation:  0.01871037050075954

Mean Absolute Error:  17247.321611625473

Mean Squared Error:  593950884.6956929

Root Mean Squared Error:  24371.107580405387
```

We can see that Ridge and Lasso Regression algorithms are performing well as compared to other models.

5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.** These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

```
parameters={'alpha' :[0.001, 0.01, 0.1, 1], 'random_state':range(42, 100), 'selection':['cyclic','ran
```

```
l=Lasso()
grid=GridSearchCV(l,parameters,cv=4,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)
print(grid.best_score_)
```

```
{'alpha': 1, 'random_state': 88, 'selection': 'random'}
0.8505113251946173
```

```
l=Lasso(alpha=1, random_state=65, selection='random')
l.fit(x_train,y_train)
pred=l.predict(x_test)
print('Final r2_score after tuning is: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(l,x,y,cv=4,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(l,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Final r2_score after tuning is:  88.93624234637868
Cross validation score:  85.98438354177253
Standard deviation:  0.018812161240908537


Mean absolute error:  17241.71478603178
Mean squared error:  593859139.4435803
Root Mean squared error:  24369.225253248824
```

```
parameters={'alpha' :[0.001, 0.01, 0.1, 1], 'random_state':range(42, 100), 'solver':['auto',
```

```
rd=Ridge()
grid=GridSearchCV(rd,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)
print(grid.best_score_)
```

```
{'alpha': 1, 'random_state': 42, 'solver': 'lsqr'}
0.8521109152220575
```

```
rd=Ridge(alpha=1, random_state=42, solver='lsqr')
rd.fit(x_train,y_train)
pred=rd.predict(x_test)
print('Final r2_score after tuning is: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(rd,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(rd,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Final r2_score after tuning is:  88.90970876230963
Cross validation score:  85.4288383516901
Standard deviation:  0.018502809040402354


Mean absolute error:  17250.237012913756
Mean squared error:  595283358.2212249
Root Mean squared error:  24398.429421198918
```

```
RF=RandomForestRegressor(random_state=4, n_estimators=500)
RF.fit(x_train,y_train)
pred=RF.predict(x_test)
print('r2_score: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(RF,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(RF,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2_score:  89.90786352806568
Cross validation score:  87.6627972359073
Standard deviation:  0.01824037150533999


Mean absolute error:  15590.938410958905
Mean squared error:  541706323.2949969
Root Mean squared error:  23274.58535173069
```

```
adr=AdaBoostRegressor(random_state=4, n_estimators=1000, learning_rate=1, loss='square')
adr.fit(x_train,y_train)
pred=adr.predict(x_test)
print("r2_score: ",r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(adr,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(adr,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2_score:  82.55640862465795
Cross validation score:  82.4893203446529
Standard deviation:  0.024590686743963752


Mean absolute error:  22694.919778475793
Mean squared error:  936303603.8280749
Root Mean squared error:  30599.078480046992
```

```
gbr=GradientBoostingRegressor(random_state=4, n_estimators=500)
gbr.fit(x_train,y_train)
pred=gbr.predict(x_test)
print("r2_score: ",r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(gbr,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(gbr,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2_score:  91.01421803172161
Cross validation score:  89.43333216962823
Standard deviation:  0.01160250434230876


Mean absolute error:  15189.185430545615
Mean squared error:  482321550.5956791
Root Mean squared error:  21961.820293310822
```

After applying Ensemble Techniques, we can see that GradientBoostingRegressor is the best performing algorithm among all other algorithms as it is giving a r2_score of 91.01 and cross validation score of 89.43

SAVING THE BEST MODEL

```
#finalizing the model
gbr_prediction=gbr.predict(x)
```

```
#Saving the model
import pickle
filename='houseprice.pkl'
pickle.dump(gbr,open(filename,'wb'))
```

```
#saving predicted values
train_results=pd.DataFrame(gbr_prediction)
train_results.to_csv('housingprice_TrainDataResults.csv')
```

USING THE TEST DATASET

```
#loading the file
df1=pd.read_csv('test.csv')
df1
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFe
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	...	0	0	NaN	NaN	
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	

292 rows × 80 columns

Here, we will be doing the same steps as we did for training dataset like handling missing data, dropping unnecessary columns, encoding non-categorical data, treating skewness, etc. Then, we will scale the data according to the best model requirements.

PREDICTING OVER TEST DATA

In [86]:
```
#loading the save model
fitted_model=pickle.load(open('houseprice.pkl','rb'))
```

In [87]:
```
fitted_model
```

Out[87]:
```
GradientBoostingRegressor(n_estimators=500, random_state=4)
```

In [88]:
```
#prediction over test data
test_predictions=fitted_model.predict(x)
```

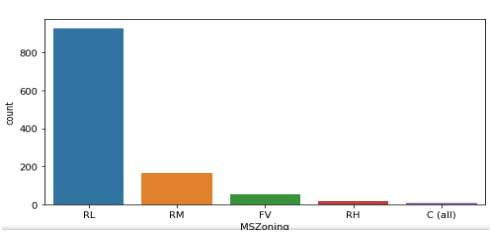
In [89]:
```
test_predictions=pd.DataFrame(test_predictions,columns=['SalePrice'])
test_predictions
```

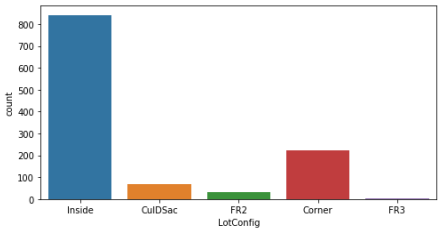
Out[89]:

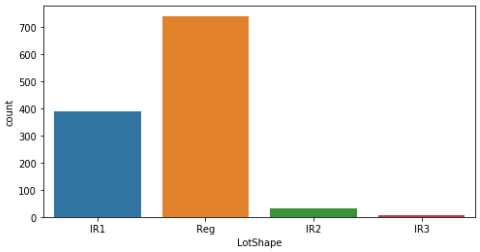
	SalePrice
0	317120.499666
1	229243.440986
2	245538.927747
3	173851.743308
4	202997.989554
...	...
287	242792.172320
288	139078.722197
289	149021.026822

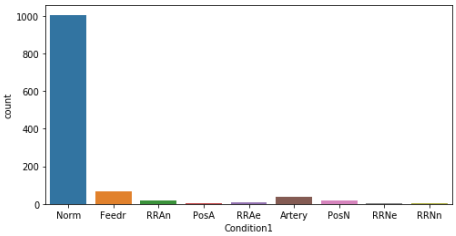
DATA VISUALIZATION

PLOTTING GRAPHS FOR CATEGORICAL DATA









Observations:

1. By looking at countplot of MSZoning ,which Identifies the general zoning classification of the sale,we find that 79 % of houses were sold in Low density resedential Areas.

2. For street ,which states:Type of road access to property,we observe that almost 100% of house which were sold had access to paved roads so we can consider that no houses were purchased which had gravel road access.

3. For Alley,93% of the purchased house do not have access to alley.Only 4% have gravel & 3% have paved alley.

4. LotShape : 63% of the sold property was of Regular shape followed by slightly irregular type (33%).It means Australian gives priority to regular shaped houses.

5. LandContour :90% of sold houses were neary flat level.

6. LotConfig : 72% of purchased houses had Inside lot of the property.

7. LandSlope :Around 95% of the sold property had gentle slope

8. Neighborhood: Physical locations within Ames city limits-:highest 16% of purhcased houses has neighbourhood of NWAmes(Northwest Ames) followed by CollgCr(College Creek) and least houses were purchased in neighbour hood of Bluestem

9. Condition1: Proximity to various conditions-:86% of purchased houses had normal proximity to various conditions1 and least 0.00 had RRne,RRNn proximity

10. Condition2: Proximity to various conditions (if more than one is present)-:99% of purchased houses had normal proximity to various conditions2

11. BldgType: Type of dwelling-:84% purchased houses were single family detached,followed by 8% 2FmCon(Two-family Conversion)

12. HouseStyle: Style of dwelling-:49% houses had 1story followed by 2story style (31%)

13. RoofStyle: Type of roof-:78% of houses have Gable roof style and 19% have Hip roof style

14. RoofMatl: Roof material-:98% houses have CompShg(Standard (Composite) Shingle) roof material

15. Exterior1st: Exterior covering on house-:34% houses have Vinylsiding covering on exteriors 15% have hard board and metal siding

16. Exterior2nd: Exterior covering on house (if more than one material)-:33% houses have VinylSd(Vinyl Siding) 15% have hard board and metal siding

17. MasVnrType: Masonry veneer type-:60% of houses have no masonry veneer type followed by BrkFace(Brick Face) (30%)

18. ExterQual: Evaluates the quality of the material on the exterior-:61% of the sold hoUse have TA(Average/Typical) quality material on exterior followed by Gd(Good) 34%

19. ExterCond: Evaluates the present condition of the material on the exterior-:88% houses are currently in TA(average) condition of exterior material

20. Foundation: Type of foundation-:44% houses have foundation CBlock(Cinder Block) & 44% have PConc(Poured Contrete)

21. BsmtQual: Evaluates the height of the basement-:44% of houses have TA(typical) (80-89 inches) basement height followed by Gd(Good) (90-99 inches)

22. BsmtCond: Evaluates the general condition of the basement-:89% of houses have TA(Typical - slight dampness allowed) basment

23. BsmtExposure: Refers to walkout or garden level walls-:64% of houses have No(No Exposure) followed by Av(Average Exposure) 15%

24. BsmtFinType1: Rating of basement finished area-:(30%) have Unf(Unfinshed) basement area and 28% comes under GLQ(good living quarters)

25. Heating: Type of heating-:98% houses have GasA(Gas forced warm air furnace) heating type

26. HeatingQC: Heating quality and condition-:30% houses have average quality heating

27. CentralAir: Central air conditioning-:93% houses are central air

28. Electrical: Electrical system-:92% houses have SbrKr(Standard Circuit Breakers & Romex) type of electrical systems

29. KitchenQual: Kitchen quality-:49% houses have average (TA) kitchen quality

30. Functional: Home functionality (Assume typical unless deductions are warranted)-:92% houses have typical (TA) home functionality

31. FireplaceQu: Fireplace quality-:47% of the houses donot have fireplace,25% houses have Gd(Good - Masonry Fireplace in main level) FireplaceQuality

32. GarageType: Garage location-:57% houses have Attached garage type,while 29% have Detchd(Detached from home)

33. GarageFinish: Interior finish of the garage:42% of houses have unfinished garage while 29% have RFn(Rough Finished)

34. GarageQual: Garage quality-:90% of houses have average garage quality

35. GarageCond: Garage condition-:91% of houses have TA(average garage condition)

36. PavedDrive: Paved driveway-:92% of houses have Y(paved drive) way

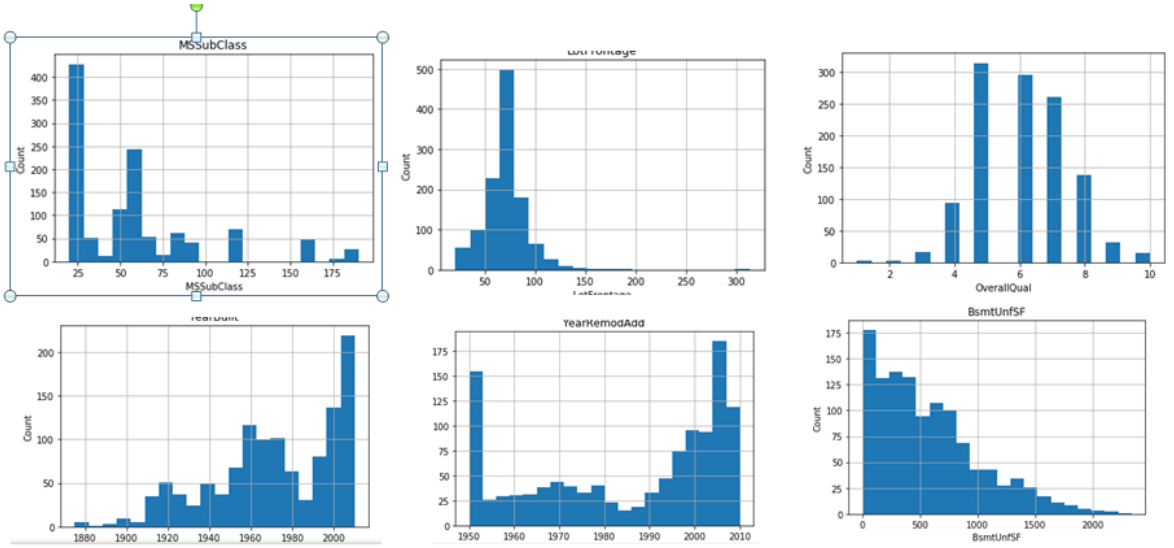
37. Fence: Fence quality-:89% houses have NA(no fence).

38. MiscFeature: Miscellaneous feature-:96% houses have no miscellaneous features

39. SaleType: Type of sale-:85% houses have sale type WD(warranty deed -conventional)

40. SaleCondition:81% of houses are in normal sale condition.

PLOTTING GRAPHS FOR CONTINUOUS DATA



Observations:

1. lotFrontage:Almost all houses have LotFrontage between 20 to 150

2. lotArea:Around 580 house have lot Area between (0-10000)sqft.Very few houses have lot area aound 120000sqft & around 160000sqft

3. OverallQual: Rates the overall material and finish of the house-:Around 300 houses sold were in average condition.Only 10-15 houses were in excellent condition.

4. YearBuilt: Original construction date-:More number of people have brought the houses build after 1990

5. MasVnrArea: Masonry veneer area in square feet-:50% of houses have Masonry veneer area as '0-50' and out of rest 50% houses most houses have Masonry veneer area 50-1200

6. BsmtFinSF1: Type 1 finished square feet-:MOst houses have Type 1 finished square feet area of basement between 0 and 1500

7. BsmtFinSF2: Type 2 finished square feet-:Around 1000 houses have Type 2 finished square feet area of 0

8. BsmtUnfSF: Unfinished square feet of basement area-:Around 130 houses have unfinished basesent of area around 100-500 sqft

9. 1stFlrSF: First Floor square feet-:Around 280 houses have 1st floor square feet area between 800-1200sqft

10. GrLivArea: Above grade (ground) living area square feet-:Most houses have above ground living sq ft area in between 800 to 3000

11. BsmtFullBath: Basement full bathrooms-:50% houses have no full bathrooms in basement and in remaining houses most have 1 full bathrooms in basement and very few has 2 full bathrooms

12. FullBath: Full bathrooms above grade-:25% houses have 1 full bathrooms above ground and 50% have 2 full bathrooms located above ground and very less have 3

13. HalfBath: Half baths above grade-: around 700 houses have no half bathrooms very few has 1 half bathroom

14. Bedroom: Bedrooms above grade (does NOT include basement bedrooms)-:Most houses have 3 bedrooms above ground followed by 2 and 4

15. Kitchen: Kitchens above grade-:Maximum houses have 1 Kitchen .very few have 2

16. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)-:Around 300 houses have 6 rooms ,around 200 have 5,&250 have 7. Very few have 12 & 14 rooms

17. Fireplaces: Number of fireplaces-:Most houses have 0 fireplaces followed by 1

18. GarageCars: Size of garage in car capacity-:Most houses have garage with 2 car capcity

19. GarageArea: Size of garage in square feet-:Most houses have Garage area in between 200 to 800

20. woodDeckSF: Wood deck area in square feet-:More than 50% of houses have 0 Wood Deck sq ft area and rest have in between 0 to 400

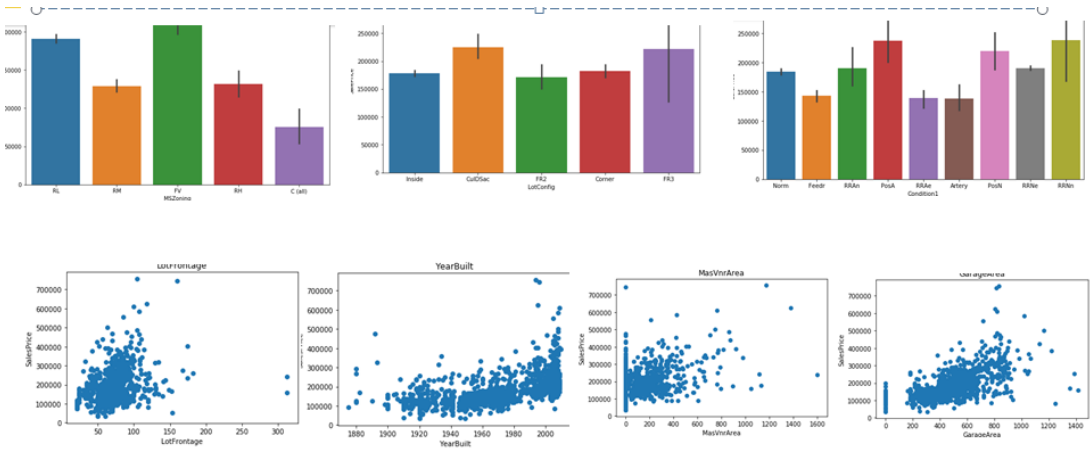
21. OpenPorchSF: Open porch area in square feet-:25% of houses have 0 open porch sq ft area and rest have in between 0 to 300

22. EnclosedPorch: Enclosed porch area in square feet-:Almost all houses have 0 enclosed porch sq ft area

23. ScreenPorch: Screen porch area in square feet-:Almost all houses have 0 screen porch area sq ft

24. Sale Price-:Around 500 house have sale price in between 100000 to 200000.Very few houses have sale price of 600000 & 700000.

BIVARIATE ANALYSIS WITH SALESPRICE FOR CATEGORICAL & CONTINUOUS DATA



Observations:

1. MSZoning:The avg sale price of the house is maximum in FV(Floating Village Residential) foloowed by RL(Residential Low Density) zone

2. Street:The property that have access to paved road have much higher average sale price as compared to that with gravel street

3. Alley:houses that do not have access to alley have higher sale price as compared to those with paved or gravel alley

4. LotShape:sale price is not much affected by lotshape,however IR2(Moderately Irregular) have a bit higher price compared to other while Reg(Regular) have lowest avg sale price

5. LandContour:Flatness of the property-:HLS(Hillside - Significant slope from side to side) have maximum average sale price & Bnk(Banked - Quick and significant rise from street grade to building) have miimum average sale price

6. LandSlope: It doesn't affect the average sale price of house

7. Neighborhood:The houses that has a neighbourhood of NoRidge(Northridge) has the maximum sale price followed by that with a neighbourhood of NridgHt(Northridge Heights)

8. Condition1:house that is RRAn(Adjacent to North-South Railroad) has hightest avg sale price followed by PosA(Adjacent to postive off-site feature) while houses that is Artery(Adjacent to arterial street) has a minimum average sale price.

9. BldgType: Type of dwelling-:TwnhsE(Townhouse End Unit) & 1Fam(Single-family Detached) type house have hightset selling price.

10. HouseStyle: Style of dwelling-:The average sale price of 2.5Fin(Two and one-half story) is maximum followed by 2Story(Two story). 1.5Unf(One and one-half story: 2nd level unfinished) have lowest avg selling price

11. RoofMatl: Roof material-:House with roof material WdShngl(Wood Shingles) have a very high average selling price,followed by that with roof of WdShake(Wood Shakes),while house with roof material Roll(Roll) have lowest sale price

12. Exterior1st: Exterior covering on house-:House with exterior covering of ImStucc(Imitation Stucco) have maximum selling price while that with exterior coverng of BrkComm(Brick Common) have minimum average selling price

13. ExterQual: Evaluates the quality of the material on the exterior-:Houses with exterior material of excellent quality have highest saelling price followed by that of gd(good) quality

14. KitchenQual: Kitchen quality-:Houses with Ex(Excellent) kitchen quality have higher sale price while that with Fa(Fair) kitchen quality of lower selling price

15. LotFrontage: Linear feet of street connected to property-: Lot frontage doed not impact much on sale price since houses with different sale price are having same Lot frontage area

16. LotArea: Lot size in square feet-: LotArea doesn't affect sale price of the houses much, as can be seen different sale price are availble within the Lot area range of 0 to 20000.In fact some houses where Lot Area is very large have moderate sale price

17. OverallQual: Rates the overall material and finish of the house-:Overall quality is directly proportional to the sale price of houses

18. YearBuilt: & YearRemodAdd: Houses which are build latest have high sale price in comparison to those build in early years.similar is the case with remodelling date

19. BsmtFinSF1: Type 1 finished square feet-:Total sq ft of basement area is directly proportional to sale price. Houses with higher number of full bathrooms seems having high sale price

20. Kitchen: Kitchens above grade-:houses with 1 kitchen above ground have high sale price in comparison to those having 2 kitchens

21. Fireplaces: Number of fireplaces-:Houses with 1 and 2 fireplaces have higher prices in comparion to houses having 0 or 3 fireplaces

22. Wood deck,Enclosed porch,Three season porch, screen porch,pool area,Miscval do not have impact on sale price.

CONCLUSION

- After getting an insight of this dataset, we were able to understand that the Housing prices are done on basis of different features.
- First, we loaded the train dataset and did the EDA process and other pre-processing techniques like skewness check and removal, handling the outliers present, filling the missing data, visualizing the distribution of data, etc.
- Then we did the model training, building the model and finding out the best model on the basis of different metrices scores we got like Mean Absolute Error, Mean squared Error, Root Mean Squared Error, etc.
- We got Lasso Regressor as the best algorithm among all as it gave more r2_score and cross_val_score. Then for finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.
- As the scores were not increased, we also tried using Ensemble Techniques like RandomForestRegressor, AdaBoostRegressor and GradientBoostingRegressor algorithms for boosting up our scores. Finally, we concluded that GradientBoostingRegressor was the best performing algorithm, although there were more errors in it and it had less RMSE compared to other algorithms. It gave an

r2_score of 91.01 and cross validation score of 89.43 which is the highest scores among all.

- We saved the model in a pickle with a filename in order to use whenever we require.
- We predicted the values obtained and saved it separately in a csv file.
- Then we used the test dataset and performed all the pre-processing pipeline methods to it.
- After treating skewness, we loaded the saved model that we obtained and did the predictions over the test data and then saving the predictions separately in a csv file.
- From this project, we learnt that how to handle train and test data separately and how to predict the values from them. This will be useful while we are working in a real-time case study as we can get any new data from the client we work on and we can proceed our analysis by loading the best model we obtained and start working on the analysis of the new data we have.
- The final result will be the predictions we get from the new data and saving it separately.
- Overall, we can say that this dataset is good for predicting the Housing prices using regression analysis and GradientBoostingRegressor is the best working algorithm model we obtained.
- We can improve the data by adding more features that are positively correlated with the target variable, having less outliers, normally distributed values, etc.

Learning Outcomes of the Study in respect of Data Science

1. Price Prediction modeling – This allows predicting the prices of houses & how they are varying in nature considering the different factors affecting the prices in the real time scenarios.

2. Prediction of Sale Price – This helps to predict the future revenues based on inputs from the past and different types of factors related to real estate & property related cases. This is best done using predictive data analytics to calculate the future values of houses. This helps in segregating houses, identifying the ones with high future value, and investing more resources on them.

3. Deployment of ML models – The Machine learning models can also predict the houses depending upon the needs of the buyers and recommend them, so customers can make final decisions as per the needs.

4. We see how to deal with outliers when all the rows have at least one value Z>3.

5. To do a visualisation when data has high standard deviation and no Classification

6. Ways to select features and to do hyperparameter tuning efficiently

7. Ways of removing skewness and what are the best methods still not versatile when it comes to data with 0 value

8. How to make a model using a pipeline.

Limitations of this work and Scope for Future Work

1. The biggest limitation I observed was that not all categories of a particular feature were available in the training data. So, if there were new category in the test data the model would not be able to identify that.

Example: MSZoning has 8 categories

A Agriculture

C Commercial

FV Floating Village Residential

I Industrial

RH Residential High Density

RL Residential Low Density

RP Residential Low-Density Park

RM Residential Medium Density

2. However in the Training dataset only 5 categories are present, what happen if other 3 categories will present in test data in future. It would be difficult for machine to identify and predict.

3. The high skewness of data reduces the effectivity

4.Many features have NaN values more than 50%, and imputation of them can decrease the effectiveness. And dropping them had the loss of data.

5.we can increase the efficiency of a model by selecting a better method to remove outliers and skewness also how to make the search of perfect model in a way that if we want to change some parameters in model then we don't have to run all the model again

THANK YOU