HOUSE PRICE PREDICTION USING REGRESSION TECHNIQUE

A MINI PROJECT REPORT

Submitted by:

Raganee Verma

(Intern FlipRobo)

INTRODUCTION

House Price Is a very important topic for real state business. Thousands of houses are sold every day and There are some questions every buyer asks himself like: -

1. What is the actual price that this house deserves?
2. Am I paying a fair price?

Lots of confusion are coming in mind before buying a new house, So in this project, we proposed a machine learning model to predict a house price based on historical data of property markets.

Data Description: -

Our dataset contains two files:

1. Train data file for tanning a model (it contains 1460 rows and 81 columns)
2. Test data file for predict a model (it contains 1459 rows and 80 columns) Test data does not have Target column Sale Price.

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

```
#details of test data
df_test.head()
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Con
1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Feedr	
1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	NAmes	Norm	
1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	
1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	

Column name of dataset:

```
df.columns  # All column name of dataset
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Importing All Required Libraries:

Firstly, we import all the libraries which are used to predict our machine learning model.

```
#Importing required packages.
import pandas as pd
import numpy as np
import scipy.stats as st
#ploting libraries
import matplotlib.pyplot as plt
import seaborn as sns
#feature engineering
from sklearn.preprocessing import StandardScaler, LabelEncoder
#train test split and cross validation
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from scipy.stats import zscore
#metrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import plot_roc_curve
#ML models for regression
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

Data Analysis:

Before performing prediction task, we want to perform some Exploratory Data Analysis to gain a basic understanding of house price data.

Statical Analysis:

We use the describe function for statical analysis and it's showing the mean, std, max, median,25% and 75% value of numerical data and top frequent unique for categorical type data. We observed that data and analyse about missing value, unique value, skewed and outlies are present in our dataset.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Id	1460.0	NaN	NaN	NaN	730.5	421.610009	1.0	365.75	730.5	1095.25	1460.0
MSSubClass	1460.0	NaN	NaN	NaN	56.89726	42.300571	20.0	20.0	50.0	70.0	190.0
MSZoning	1460	5	RL	1151	NaN	NaN	NaN	NaN	NaN	NaN	NaN
LotFrontage	1201.0	NaN	NaN	NaN	70.049958	24.284752	21.0	59.0	69.0	80.0	313.0
LotArea	1460.0	NaN	NaN	NaN	10516.828082	9981.264932	1300.0	7553.5	9478.5	11601.5	215245.0
...	...	...	...	...	...	...	...	...	...	...	...
MoSold	1460.0	NaN	NaN	NaN	6.321918	2.703626	1.0	5.0	6.0	8.0	12.0
YrSold	1460.0	NaN	NaN	NaN	2007.815753	1.328095	2006.0	2007.0	2008.0	2009.0	2010.0
SaleType	1460	9	WD	1267	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SaleCondition	1460	6	Normal	1198	NaN	NaN	NaN	NaN	NaN	NaN	NaN
SalePrice	1460.0	NaN	NaN	NaN	180921.19589	79442.502883	34900.0	129975.0	163000.0	214000.0	755000.0

Check missing value Unique value and duplicate value in dataset:

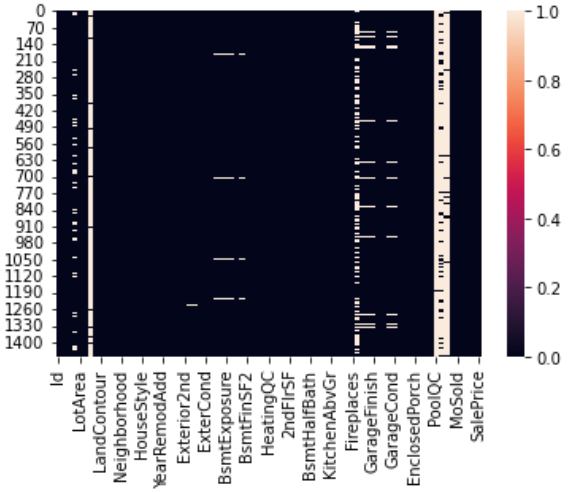
Lots of missing value is present but the column name is not shown there so we try to find out the column where missing value are present.

no duplicate present there because the row and column value is same.

```
In [18]:  1  df.duplicated().sum()

Out[18]: 0
```

	Unique Values
Id	1460
MSSubClass	15
MSZoning	5
LotFrontage	110
LotArea	1073
...	...
MoSold	12
YrSold	5
SaleType	9
SaleCondition	6
SalePrice	663



Here we check the percentage of nan values present in each feature:

I observe that sale price has 49.9% missing value because we join both train and test data but test data do not have this column that's why missing value present here so no need to impute this column because after cleaning treatment, I separate both the data again.

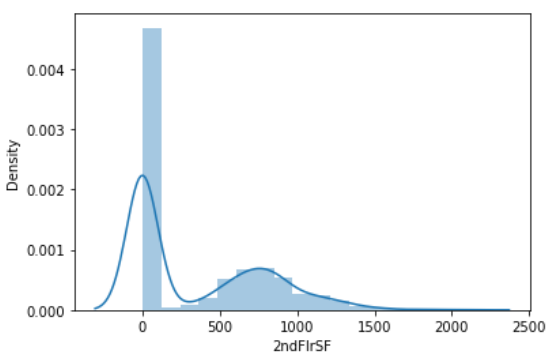
We might want to remove columns with more than 50% missing values

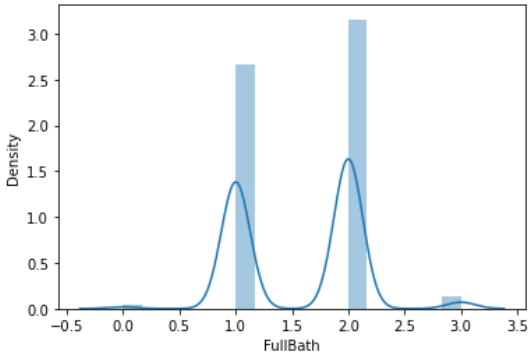
```
for features in feautres_na:
    print(features,np.round(df[features].isnull().mean(),4),'%')
```

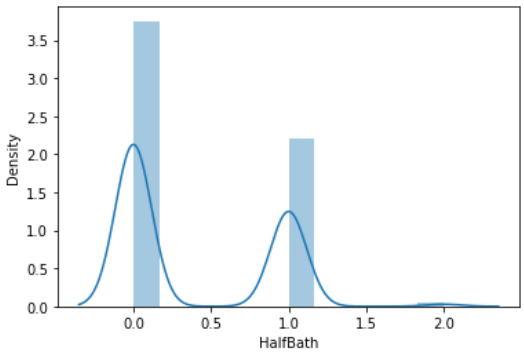
```
LotFrontage 0.1774 %
Alley 0.9377 %
MasVnrType 0.0055 %
MasVnrArea 0.0055 %
BsmtQual 0.0253 %
BsmtCond 0.0253 %
BsmtExposure 0.026 %
BsmtFinType1 0.0253 %
BsmtFinType2 0.026 %
Electrical 0.0007 %
FireplaceQu 0.4726 %
GarageType 0.0555 %
GarageYrBlt 0.0555 %
GarageFinish 0.0555 %
GarageQual 0.0555 %
GarageCond 0.0555 %
PoolQC 0.9952 %
Fence 0.8075 %
MiscFeature 0.963 %
```

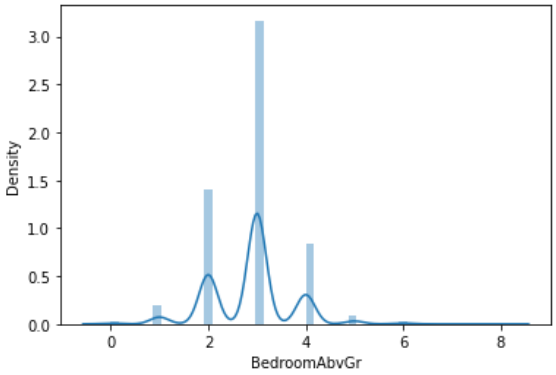
Data Visualization: Data visualization is the graphical representation of information and data.

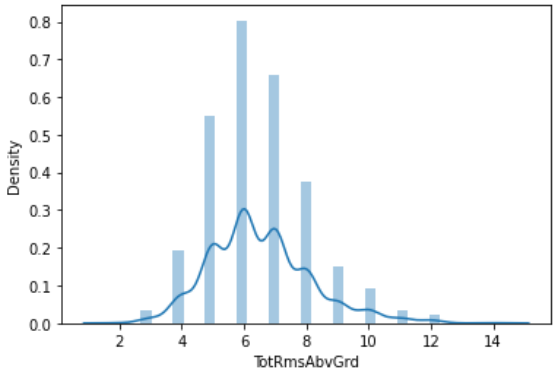
Skewness: distribution plot for each column that contain continues variable. We seen that Skewness is present so we need to remove skewness using any Transform Technique.

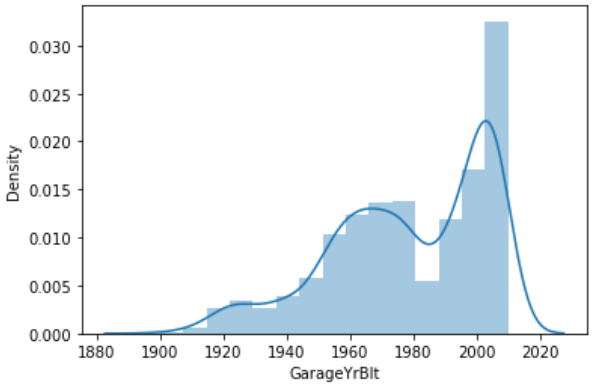




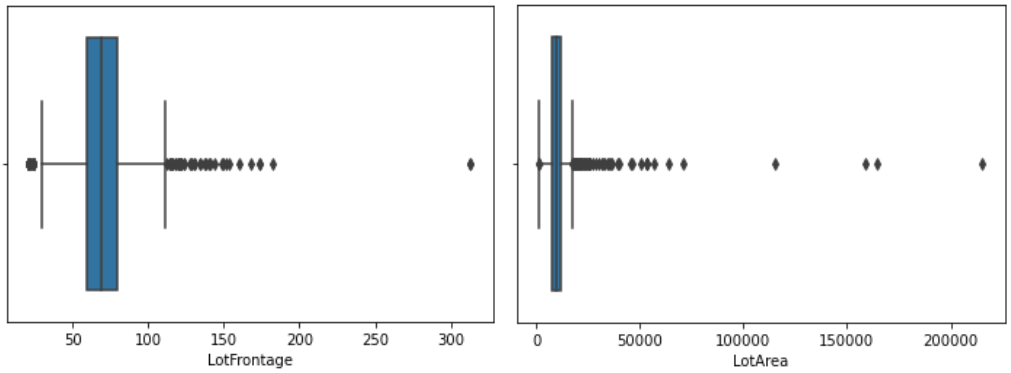


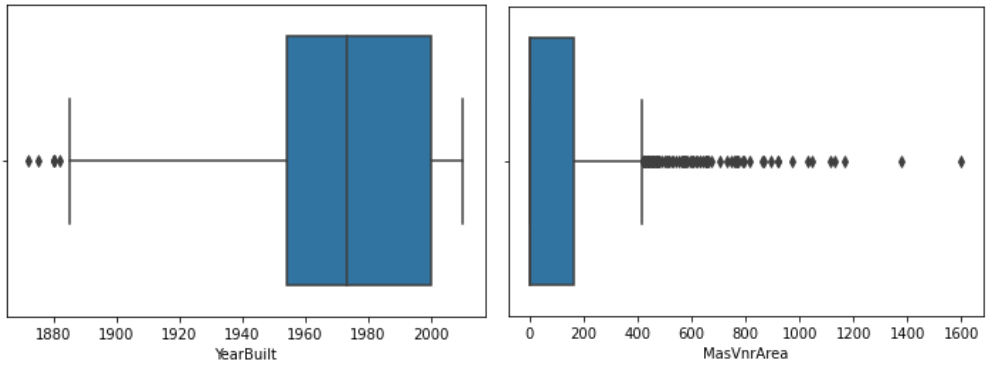






Outliers: Use box plot for visualizing outliers are present in our data. We observed that some outliers are there so we can remove it by using z-score or IQR technique.

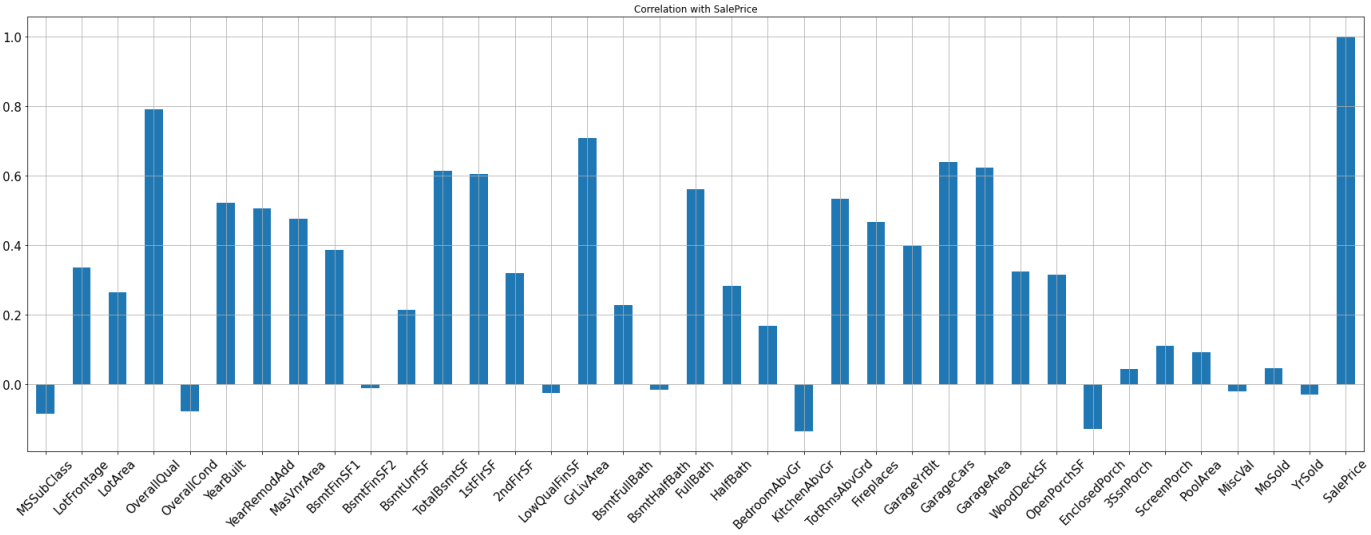




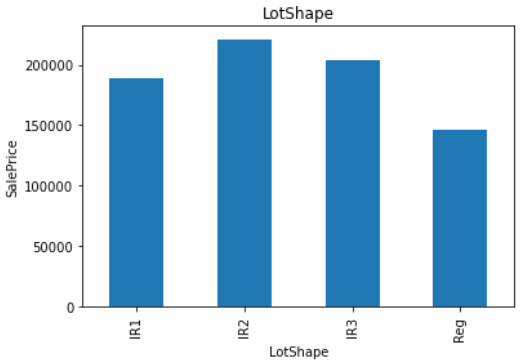
Correlation:

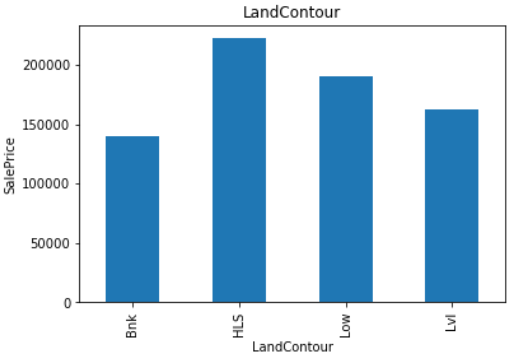
Checking the Correlation between the target value and rest columns.

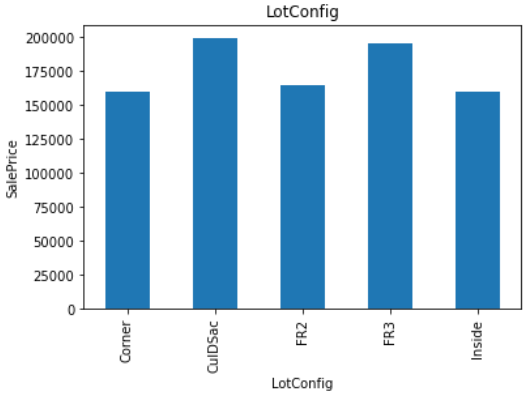
We observed that most of the column is positively correlated with target variable.

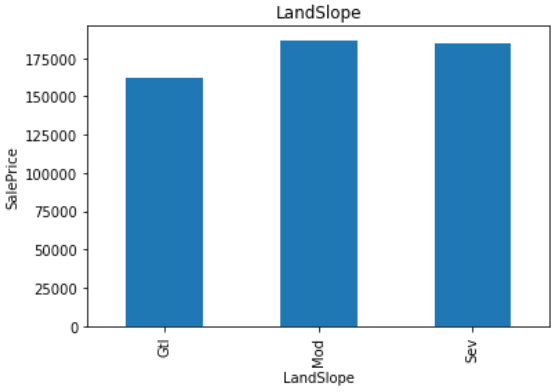


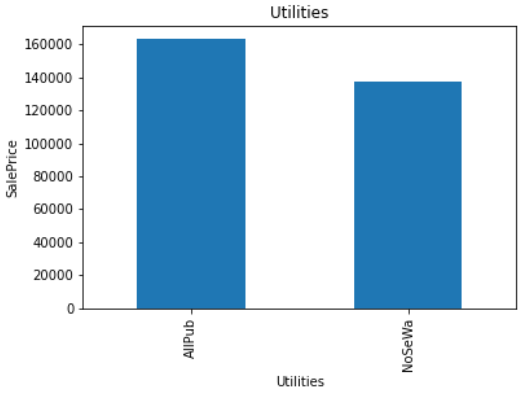
Bivariate Analysis: Data visualization between Discrete and Continues features and Target variable

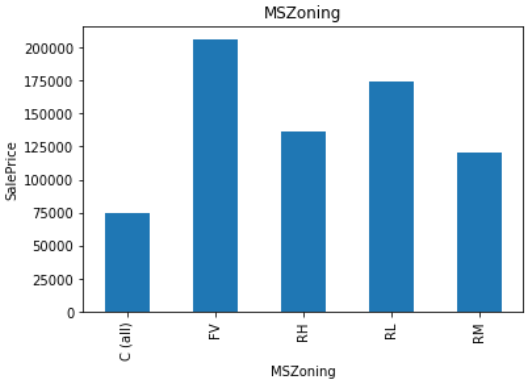

LotShape
SalePrice
0
50000
100000
150000
200000
IR1
IR2
IR3
Reg
LotShape

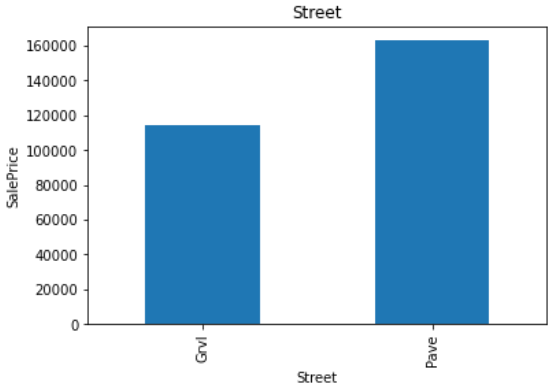

LandContour
SalePrice
0
50000
100000
150000
200000
Bnk
HLS
Low
Lvl
LandContour

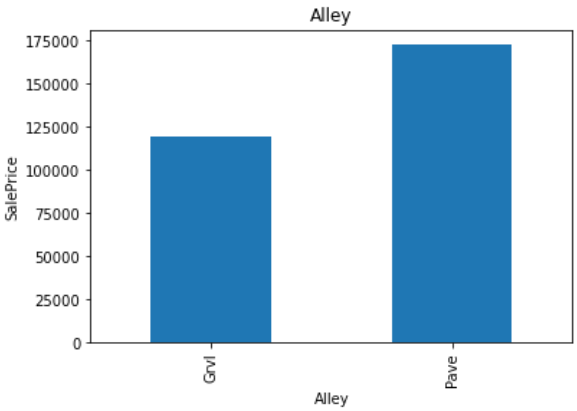

LotConfig
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
Corner
CulDSac
FR2
FR3
Inside
LotConfig

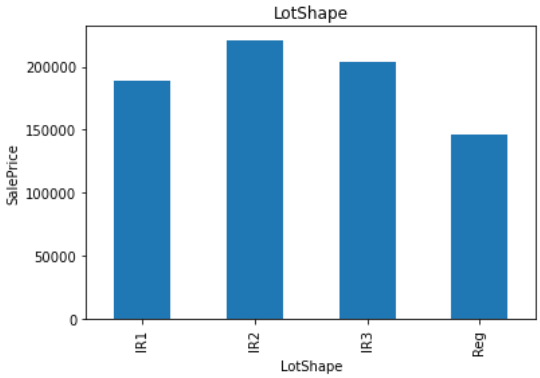

LandSlope
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
Gtl
Mod
Sev
LandSlope

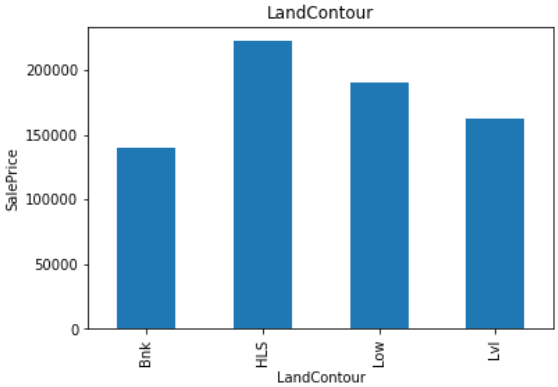

Utilities
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
AllPub
NoSeWa
Utilities

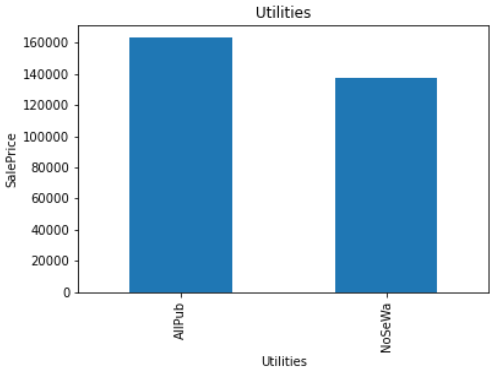

MSZoning
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
C (all)
FV
RH
RL
RM
MSZoning

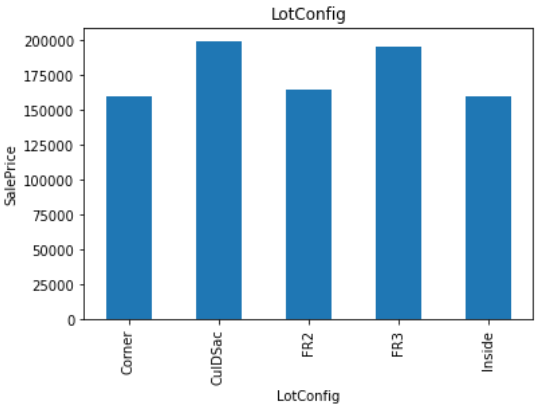

Street
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
Grvl
Pave
Street

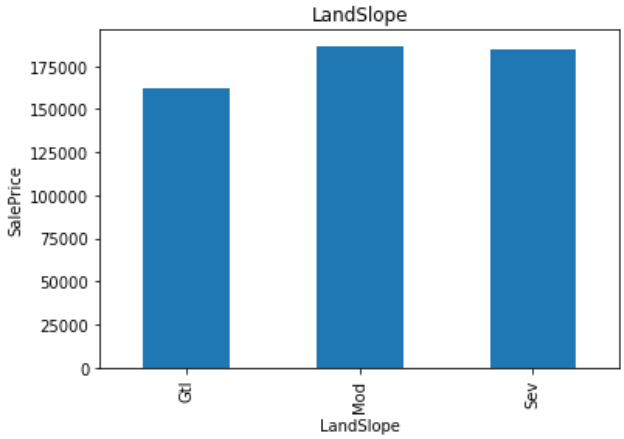

Alley
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
Grvl
Pave
Alley

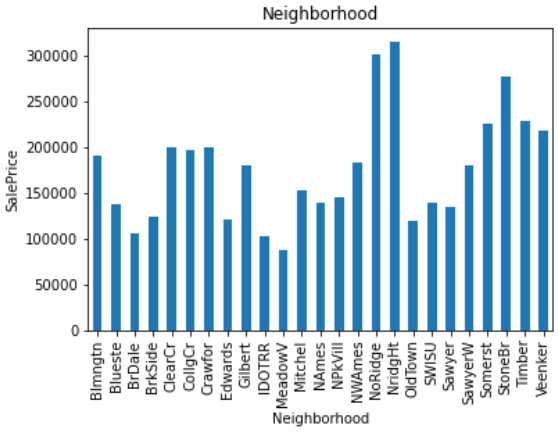

LotShape
SalePrice
0
50000
100000
150000
200000
IR1
IR2
IR3
Reg
LotShape

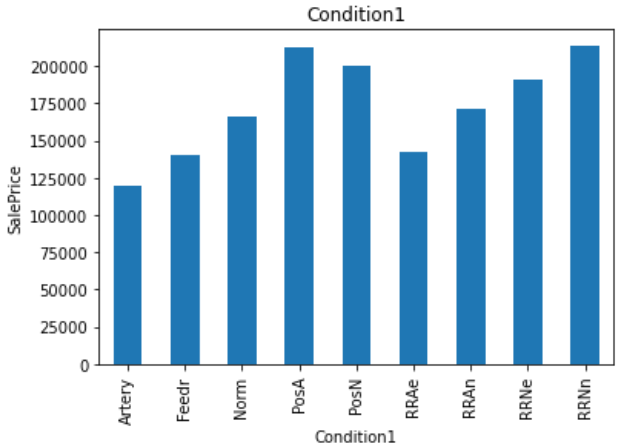

LandContour
SalePrice
0
50000
100000
150000
200000
Bnk
HLS
Low
Lvl
LandContour


Utilities
SalePrice
0
20000
40000
60000
80000
100000
120000
140000
160000
AllPub
NoSeWa
Utilities

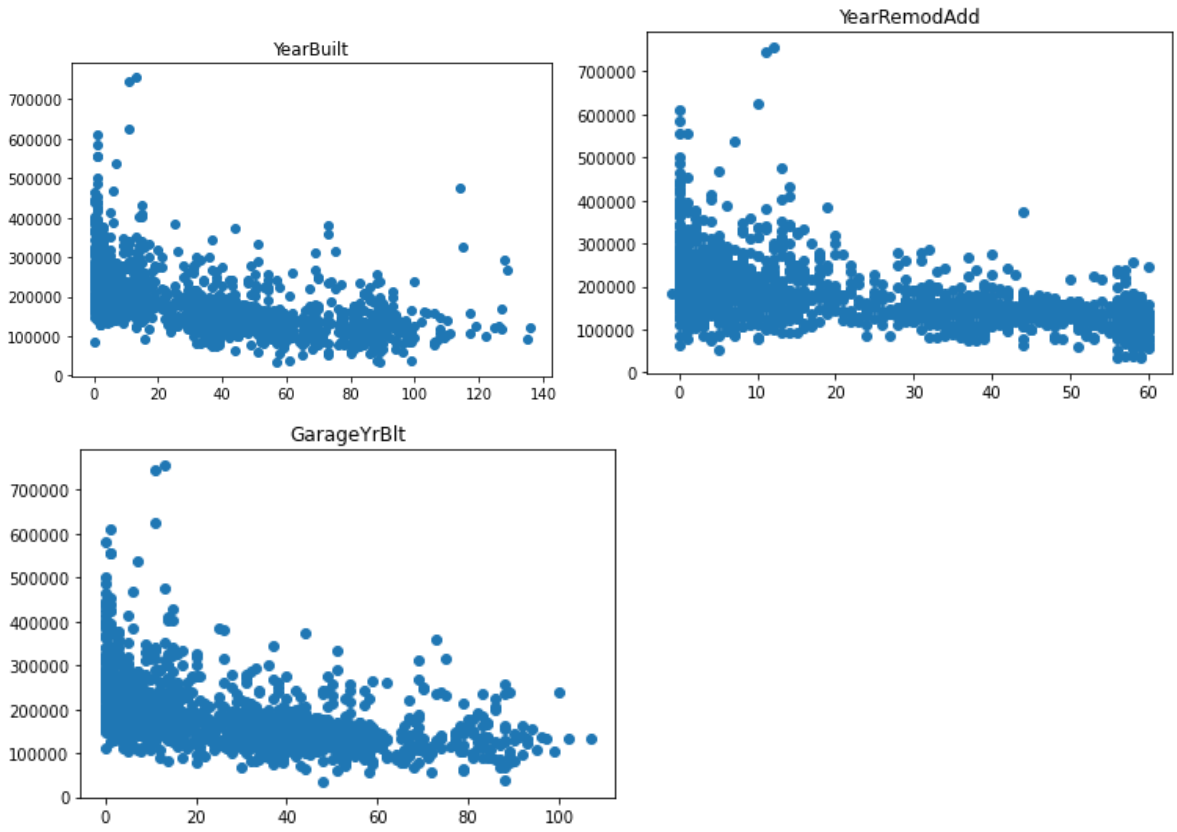

LotConfig
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
Corner
CulDSac
FR2
FR3
Inside
LotConfig


LandSlope
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
Gtl
Mod
Sev
LandSlope


Neighborhood
SalePrice
0
50000
100000
150000
200000
250000
300000
Blmngtn
Blueste
BrDale
BrkSide
ClearCr
CollgCr
Crawfor
Edwards
Gilbert
IDOTRR
MeadowV
Mitchel
NAmes
NPkVill
NWAmes
NoRidge
NridgHt
OldTown
SWISU
Sawyer
SawyerW
Somerst
StoneBr
Timber
Veenker
Neighborhood


Condition1
SalePrice
0
25000
50000
75000
100000
125000
150000
175000
200000
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNe
RRNn
Condition1

Data visualization between Year Sold and Target variable:



Feature Engineering:

Handling missing value

```
for feature in categorical_features:
    df[feature] = df[feature].fillna(df[feature].mode()[0])                #train
    df_test[feature] = df_test[feature].fillna(df_test[feature].mode()[0])  #test
```

```
for feature in discrete_features:
    df[feature] = df[feature].fillna(df[feature].mode()[0])                #train
    df_test[feature] = df_test[feature].fillna(df_test[feature].mode()[0])  #test
```

```
for feature in year_features:
    df[feature] = df[feature].fillna(df[feature].mode()[0])                #train
    df_test[feature] = df_test[feature].fillna(df_test[feature].mode()[0])  #test
```

```
# as sales price is absent in test data so we want to remove it from the feature list
continous_numerical_features = [feature for feature in continous_features if feature not in 'SalePrice']
```

```
for feature in continous_numerical_features:
    df[feature] = df[feature].fillna(df[feature].mean())                #train
    df_test[feature] = df_test[feature].fillna(df_test[feature].mean())  #test
```

Label Encoder: Convert numerical variable into categorical variable.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
#for changing all catigorical value into numeric value

final_df=final_df.apply(le.fit_transform)
```

```
final_df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	Bldg1
0	5	3	41	619	1	3	3	0	4	0	5	2	2	
1	0	3	58	895	1	3	3	0	2	0	24	1	2	
2	5	3	44	1266	1	0	3	0	4	0	5	2	2	
3	6	3	36	883	1	0	3	0	0	0	6	2	2	
4	5	3	62	1670	1	0	3	0	2	0	15	2	2	

Drop Unnecessary columns:

Drop the column which is not useful for prediction purpose

```
for feature in more_than_50_percent_misssing_value_features:
    df.drop([feature],axis = 1, inplace = True)
    df_test.drop([feature],axis = 1, inplace = True)
```

We will also drop the id column as it is not useful for prediction

```
df.drop(['Id'],axis = 1, inplace = True)
df_test.drop(['Id'],axis = 1, inplace = True)
```

Scaling using Standard Scaler:

By using standard scaler, we arrange the value between 0 and 1.

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x=pd.DataFrame(scaler.fit_transform(x), columns=x.columns)
```

```
x.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
0	0.177030	-0.045532	-0.290681	-0.508928	0.064238	0.750731	0.314667	-0.02618	0.604670	-0.225716	-1.206215	-0.036289	-0.03174
1	-0.983043	-0.045532	0.546875	0.000388	0.064238	0.750731	0.314667	-0.02618	-0.628316	-0.225716	1.954302	-1.188074	-0.03174
2	0.177030	-0.045532	-0.142877	0.685012	0.064238	-1.378933	0.314667	-0.02618	0.604670	-0.225716	-1.206215	-0.036289	-0.03174
3	0.409045	-0.045532	-0.537022	-0.021756	0.064238	-1.378933	0.314667	-0.02618	-1.861302	-0.225716	-1.039872	-0.036289	-0.03174
4	0.177030	-0.045532	0.743947	1.430533	0.064238	-1.378933	0.314667	-0.02618	-0.628316	-0.225716	0.457215	-0.036289	-0.03174

Model Building: Our target variable is Continues type of variable so we can use many Regressions model.

LinearRegression, Ridge, Lass,DecisionTreeRegressor

,RandomForestRegressor,KNeighborsRegressor,AdaBoostRegressor, ExtraTreesRegressor,GradientBoostingRegressor.

Now we use many regressions algorithm but the performance of ExtraTreesRegressor is best so this is our final model and perform hyperparameter tunning for this model.

The r2 value and cross validation score of ExtraTreesRegressor model is 90% .

```
# Choosing Extra Trees Regressor

param = {'criterion' : ['mse', 'mae'],
               'min_samples_split' : [2, 3],
               'n_jobs' : [-1, 1]
              }
```

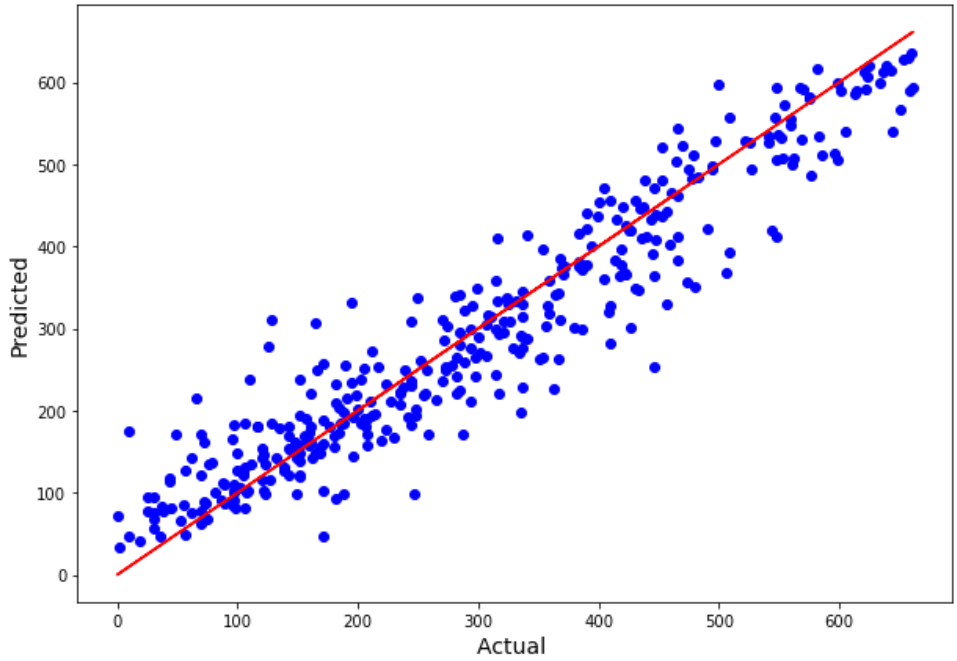
```
GSCV = GridSearchCV(ExtraTreesRegressor(),param, cv=5)
print(GSCV.fit(x_train,y_train))
print(GSCV.best_params_)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
             param_grid={'criterion': ['mse', 'mae'],
                         'min_samples_split': [2, 3], 'n_jobs': [-1, 1]})
{'criterion': 'mse', 'min_samples_split': 2, 'n_jobs': -1}
```

```
final_model = RandomForestRegressor(criterion = 'mse',min_samples_split = 2, n_jobs=-1)
final_model.fit(x_train,y_train)
predfm = final_model .predict(x_test)
scr = cross_val_score(final_model ,x,y,cv=5)
print("r2 score= ",r2_score(y_test,predfm)*100)
print('Cross validation score for RandomForestRegresso Model is ',scr.mean()*100)
print("MSE= ",mean_squared_error(y_test,predfm))
print("RMSE= ",np.sqrt(mean_squared_error(y_test,predfm)))
```

```
r2 score=  90.01634760346646
Cross validation score for RandomForestRegresso Model is  89.22800059636457
MSE=  2948.367931780822
RMSE=  54.29887597161493
```

Best fit Line for Final Model:



Saving and Loading Final Model:

```
import pickle
filename_1="house_price.pkl"
pickle.dump(final_model,open(filename_1,"wb"))
```

```
loaded_model=pickle.load(open("house_price.pkl","rb"))
result=loaded_model.score(x_test,y_test)
print(result*100)
```

```
90.01634760346646
```

Prediction and Conclusion:

Now we predict our saved model by using test data.

```
conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],predfm[:]],index=["predicted","original"])
```

```
conclusion.T
```

	predicted	original
0	180.92	180.92
1	310.60	310.60
2	236.21	236.21
3	321.58	321.58
4	181.14	181.14
...	...	...
360	421.84	421.84
361	556.73	556.73
362	110.51	110.51
363	249.68	249.68
364	411.04	411.04

365 rows × 2 columns

Conclusion/ Results

As a result, we can say, ExtraTreesRegressor is the best fit to this dataset, gives 90% accuracy. Gradient, lasso, ridge Models working very well on this Dataset. Ensembled all the models together for better results.

In [113]:

ffgg

```
import pickle
```

.