HOUSING:PRICE PREDICTION

Submitted by:

Shama Tanweer

ACKNOWLEDGMENT

I would like to thank my mentors at Data Trained, for their valuable and constructive suggestions during the planning and development of this project. For this particular task, I referred the following websites and articles when stuck:
https://stackoverflow.com/questions/43590489/gridsearchcv%20random-forest-regressor-tuning-best-params

https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/

https://www.mikulskibartosz.name/pca-how-to-choose-the-number-of-components/

INTRODUCTION

Business Problem Framing

The objective was to model the price of houses with the available independent variables. This model can 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

Houses are one of the necessary need 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. 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 variable?
- How do these variables describe the price of the house?

Technical Requirements:

- Data contains 1460 entries each having 81 variables.
- Data contains Null values
- Extensive EDA has to be performed to gain relationships of important variable and price.
- Data contains numerical as well as categorical variable
- You have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters.

Motivation for the Problem Undertaken

This model will 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 themanagement to understand the pricing dynamics of a new market

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

1.we used describe function,which tells us all the mathematical information about the numerical column

2. Mean,median,mode,Q1,Q2, Min, Max- methods are used to understand the skewness, , standard deviation and outliers.

3.Statistical model used- correlation amongst the columns and target variable to understand the correlation between different features..

4.Boxplot=We have used boxplot for outlier detection virtually

5.We have removed the outliers by Zscore which is used to limit the data within required standard deviation

6.StandardScaler: since different features have different scales, there is a chance that higher weightage is given to features with higher magnitude. This will impact the performance of the machine learning algorithm . Therefore, we scale our data by using **standard scaler** so that all the features contribute equally to the result.

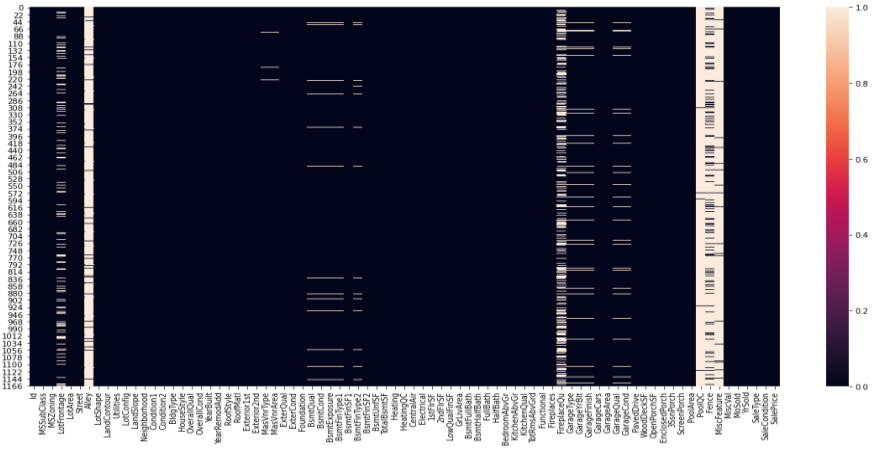
7.PCA: Principal component analysis (PCA) has been used to reduces the dimensionality of the data while retaining most of the variation in the data set.

- Data Sources and their formats

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.

Data Preprocessing

1 checking for null values in each column



The above heatmap shows there are many Null Values, which can't be processed.

As per data description ,NA has different meaning for different features.So,lets 1st replace NA with actual words.

For Example, • NA in feature 'Alley' means No_Alley

The percentage of Null values in Categorical features:

```
Alley: 93.41% missing values
MasVnrType: 0.6% missing values
BsmtQual: 2.57% missing values
BsmtCond: 2.57% missing values
BsmtExposure: 2.65% missing values
BsmtFinType1: 2.57% missing values
BsmtFinType2: 2.65% missing values
FireplaceQu: 47.17% missing values
GarageType: 5.48% missing values
GarageFinish: 5.48% missing values
GarageQual: 5.48% missing values
GarageCond: 5.48% missing values
PoolQC: 99.4% missing values
Fence: 79.71% missing values
MiscFeature: 96.23% missing values
```

Then I replaced all other categorical missing values with actual word given in data description.

#Removing irrelevant features as Id is of no use & PoolQc contains almost 99% of nan value

df.drop(["PoolQC","Id"],axis=1,inplace=True)

Imuting null values

```
# For Alley, NA means No_alley_access. Let's replace NAs with 'No_alley_access
'
df['Alley'].fillna('No_alley_access',inplace=True)
print(df['Alley'].value_counts())

basement=['BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2']
#NA means No_Basement for all i in basement. Let's replace NAs with 'No_Baseme
nt'
for i in basement:
    df[i].fillna('No_Basement',inplace=True)
    print(df[i].value_counts())
```

```
# As per given definition, NA means None. Let's replace NAs with 'None'
df['MiscFeature'].fillna('None',inplace=True)
print(df['MiscFeature'].value_counts())

# As per given definition, NA means No_Fence. Let's replace NAs with 'No_Fence
'
df['Fence'].fillna('No_Fence',inplace=True)
print(df['Fence'].value_counts())

#NA means No_Fireplace. Let's replace NAs with 'No_Fireplace'
df['FireplaceQu'].fillna('No_Fireplace',inplace=True)
print(df['FireplaceQu'].value_counts())

garage=['GarageType','GarageFinish','GarageQual','GarageCond']
# NA means No_Garage for all i in garage
for i in garage:
    df[i].fillna('No_Garage',inplace=True)
    print(df[i].value_counts())
```

Then I have checked missing values in all the coninuous features

```
LotFrontage: 18.32% missing values
MasVnrArea: 0.6% missing values
GarageYrBlt: 5.48% missing values
```

By using simple imputer all the missing data in contiuous feature has been filled with mean.

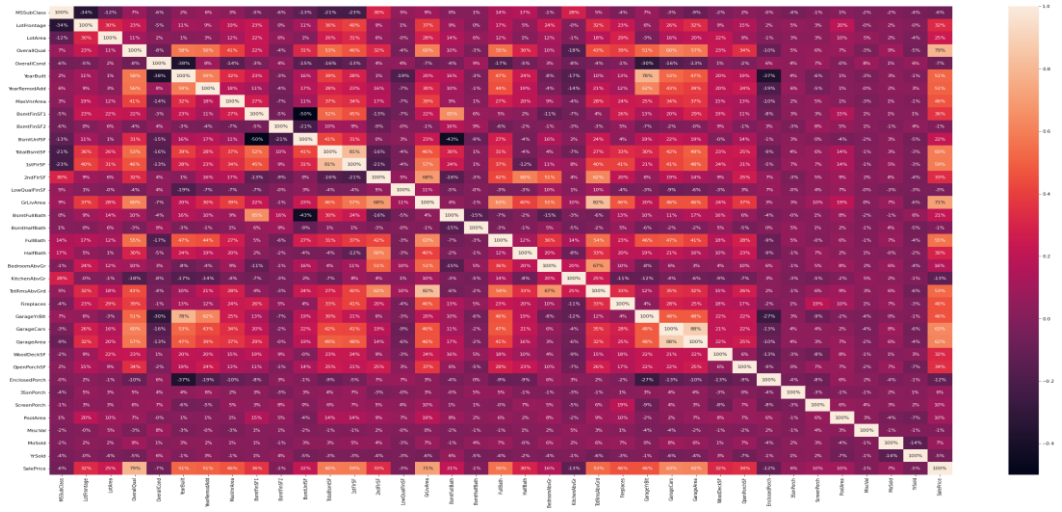
2. The data description revealed that :

1. Std is greater than mean for some features such as LowQualFinSF,PoolArea etc which means data is widely spread & indicate the presence of skewness.

2. There is large difference between maxm value and 75% ,which means that outliers are present .

3. MasVnrArea,BsmtFinSF1,BsmtFinSF2,2ndFlrSF,LowQualFinSF,BsmtFullBath,HalfBath,Fireplaces, WoodDeckSF,OpenPorchSF,EnclosedPorch,3SsnPorch,ScreenPorch,PoolArea,MiscVal have outliers

3.Further checked the correlation between columns



4.**Outliers** present in dataset was removed by Zscore,But when we apply zscore of 3,20% of the data was lost(due to widespread of data).We compromised by enlarging the zscore cut-off to 5 resulting in a loss of few data

5.**Removing of skewness**- using square root and cube root method skewness is removed.

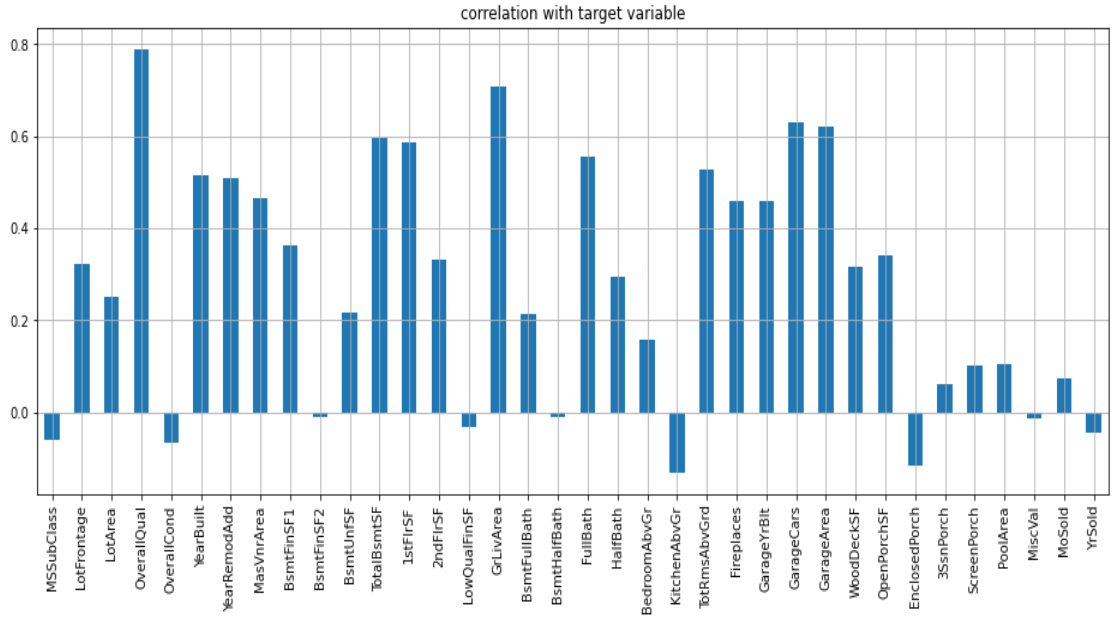
7.Scaling of the data- standard scaler is used to scale data

8.PCA technique is used for the analysis ,to reduce curse of dimensionality & at the same time minimizing information loss

- **Data Inputs- Logic- Output Relationships**

The SalePrice is the output variable here, rest all are the input variables.

Correlation with target column

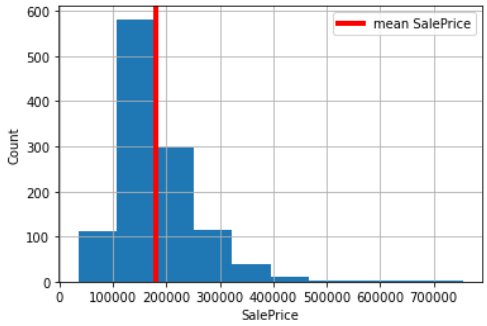


1. 'MSSubClass' ,'OverallCond' ,'OverallCond' ,'LowQualFinSF' ,'BsmtHalfBath' ,'KitchenAbvGr' ,'YrSold', 'EnclosedPorch', 'MiscVal' are negatively correlated with the target column,rest all are positively correlated

2. 'OverallQual' & 'GrLivArea' are highly positively correlated with target column

3. 'MSSubClass' ,'OverallCond' ,'OverallCond' ,'LowQualFinSF' ,'BsmtHalfBath' ,'YrSold', 'MiscVal', 'MoSold', '3SsnPorch' are least correlated with the target column

Data Visualisation

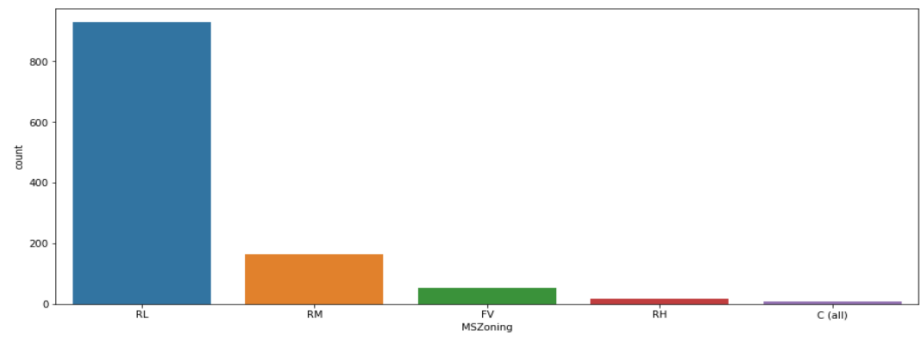


For better visualization I split categorical and numerical features and stored them in separate list.

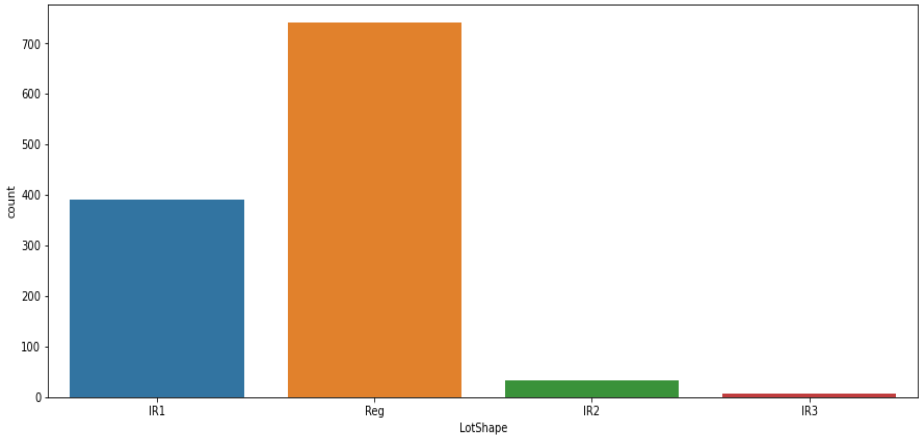
Univariatr Analysis

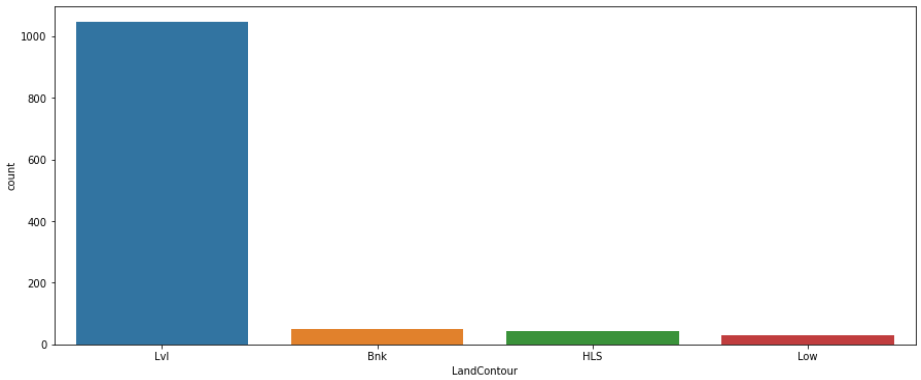
• I then used seaborn' countplot on all categorical features to check the count of each category present inside them. Below is the code which gave me countplot for each categorical attribute, where categorical contained only categorical features

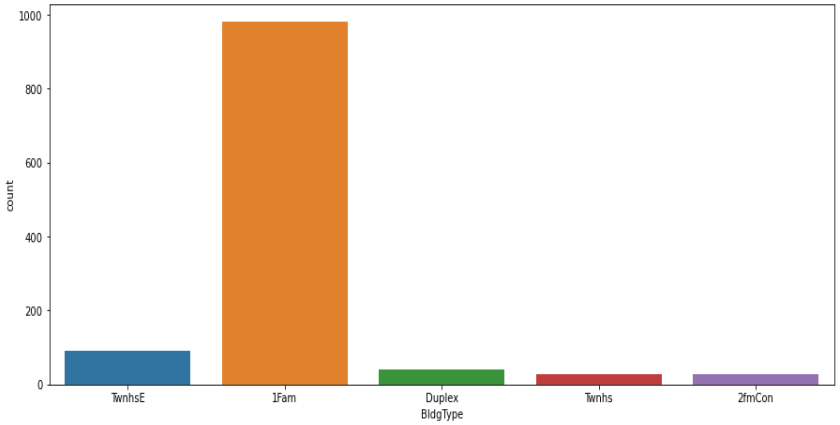
```
#visualizing each categorical column wrt count of each value
for i in categorical:
    plt.figure(figsize=(15,6))
    sns.countplot(df[i])
    plt.show()
#checking percentage of data classification in each string attribute
    print(round(df[i].value_counts()/1168*100),2)
```




```
RL         79.0
RM         14.0
FV          4.0
RH          1.0
C (all)     1.0
Name: MSZoning, dtype: float64 2
```







Observation from all the count plot¶

1. 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

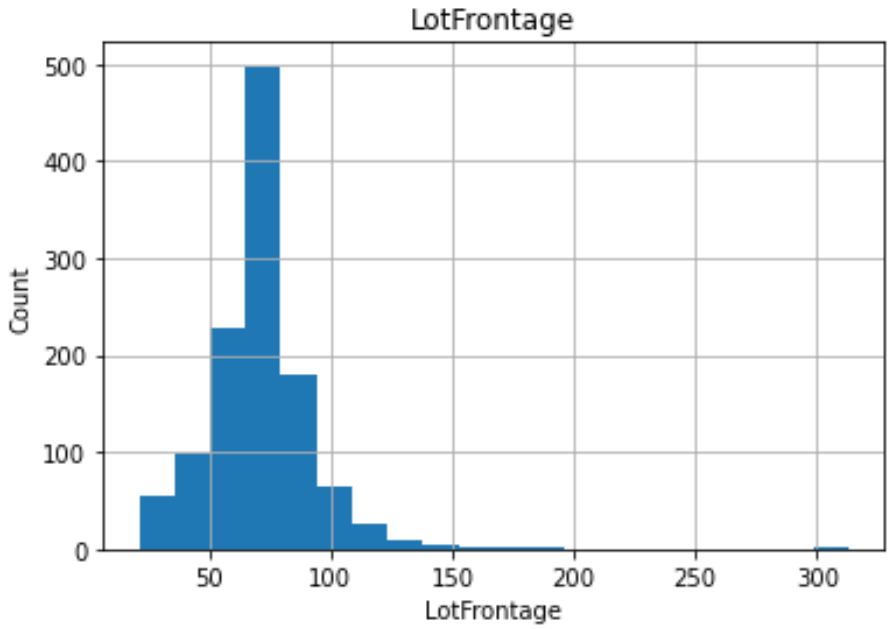
31. FireplaceQu: Fireplace quality-:47% of the houses donot have fireplace,25% houses have Gd(Good - Masonry Fireplace in main level) FireplaceQuality

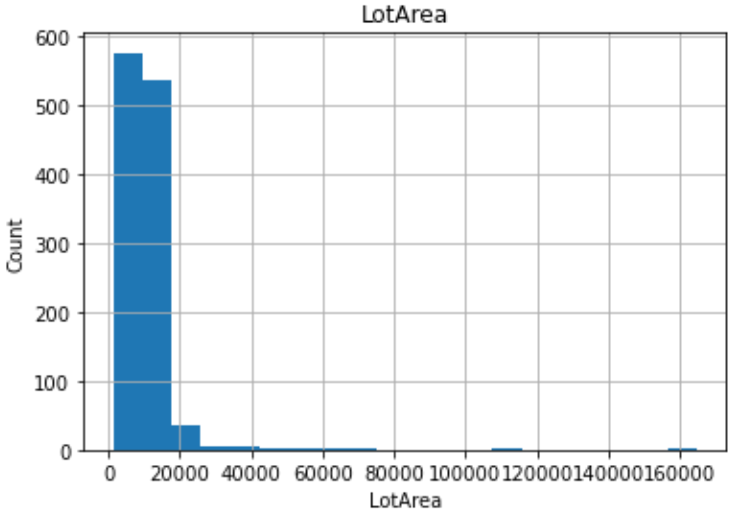
32. GarageType: Garage location-:57% houses have Attached garage type,while 29% have Detchd(Detached from home)

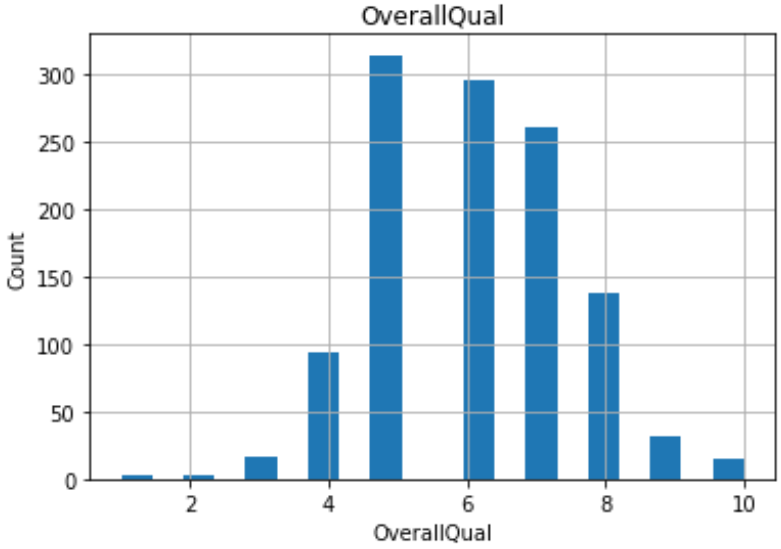
33. GarageFinish: Interior finish of the garage:42% of houses have unfinished garage while 29% have RFn(Rough Finished)

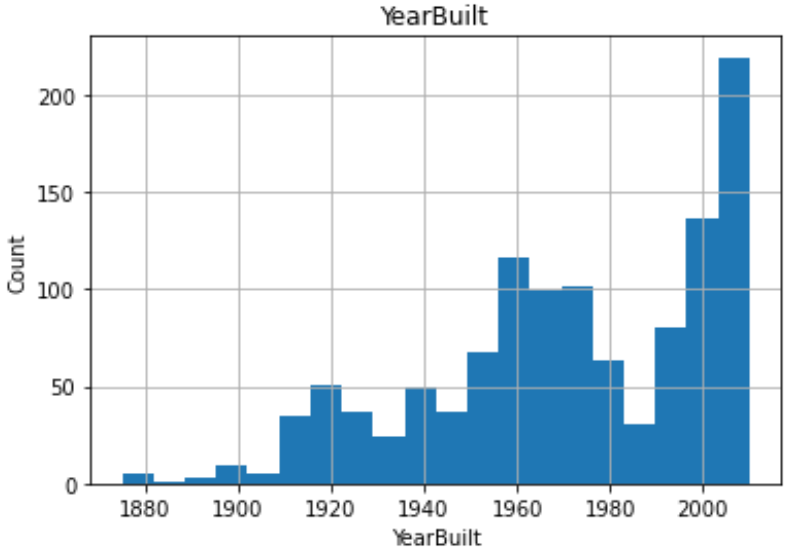
34. GarageQual: Garage quality-:90% of houses have average garage quality
35. GarageCond: Garage condition-:91% of houses have TA(average garage condition)
36. PavedDrive: Paved driveway-:92% of houses have Y(paved drive) way
37. PoolQC: Pool quality-:99% houses donot have pool
38. Fence: Fence quality-:89% houses have NA(no fence).
39. MiscFeature: Miscellaneous feature-:96% houses have no miscellaneous features
40. SaleType: Type of sale-:85% houses have sale type WD(warranty deed -conventional)
41. SaleCondition:81% of houses are in normal sale condition

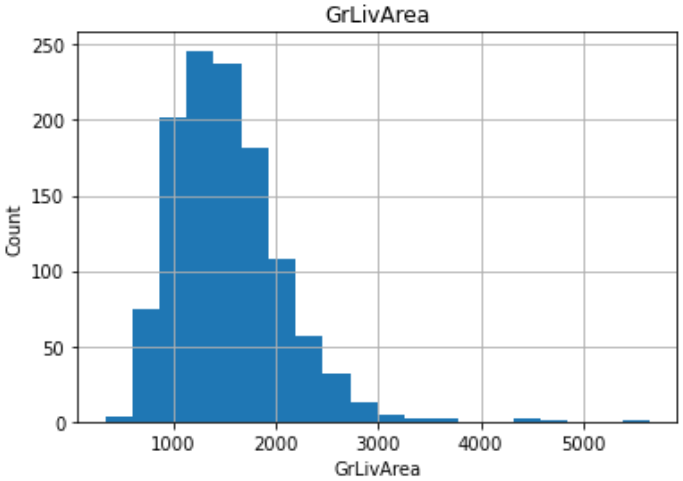
```
#visualizing each continuous feature
for feature in cont:
    data=df.copy()
    data[feature].hist(bins=20)
    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title(feature)
    plt.show()
```




LotArea
Count
LotArea
0
100
200
300
400
500
600
0
20000
40000
60000
80000
100000
120000
140000
160000


OverallQual
Count
OverallQual
0
50
100
150
200
250
300
2
4
6
8
10


YearBuilt
Count
YearBuilt
0
50
100
150
200
1880
1900
1920
1940
1960
1980
2000



0bservation:

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

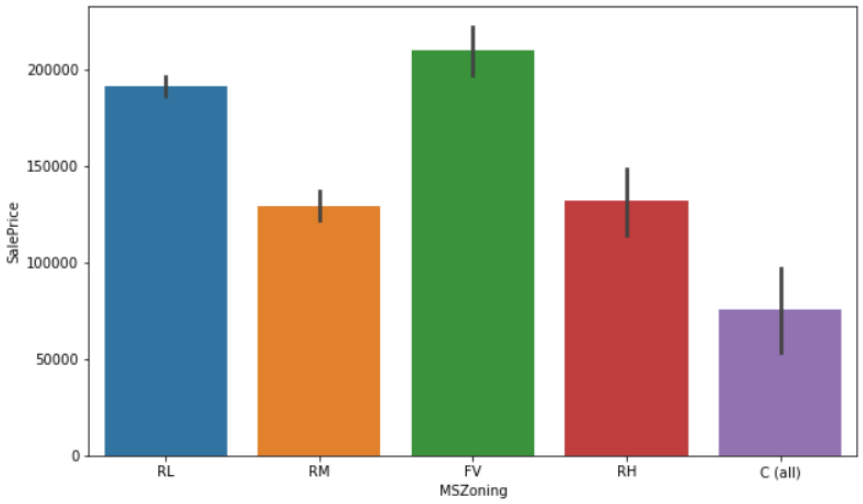
23. ScreenPorch: Screen porch area in square feet-:Almost all houses have 0 screen porch area sq ft

24. PoolArea: Pool area in square feet-:Almost all houses have 0 sq ft of pool area

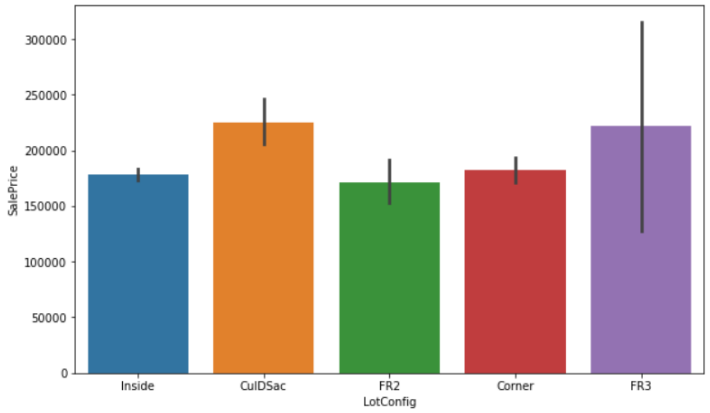
25. Sale Price-:Around 500 house have sale price in between 100000 to 200000.Very few houses have sale price of 600000 & 700000

Bivariate analysis

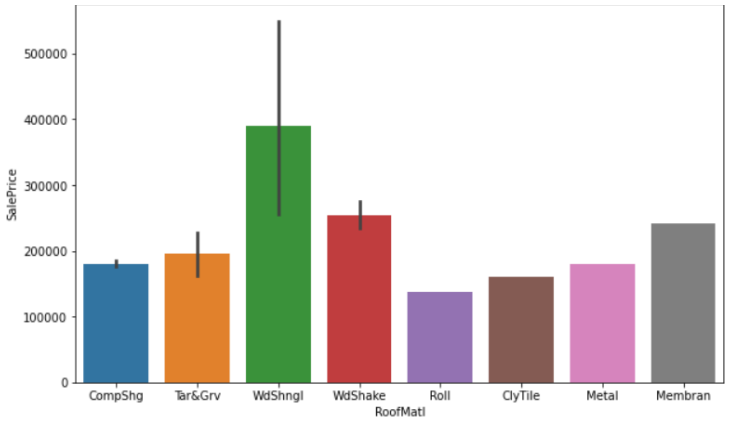
```
#visualizing each categorical column wrt sale price
for i in categorical:
    plt.figure(figsize=(10,6))
    sns.barplot(x=df[i],y=df['SalePrice'])
    plt.show()
    print(df.groupby(df[i])['SalePrice'].mean())
```



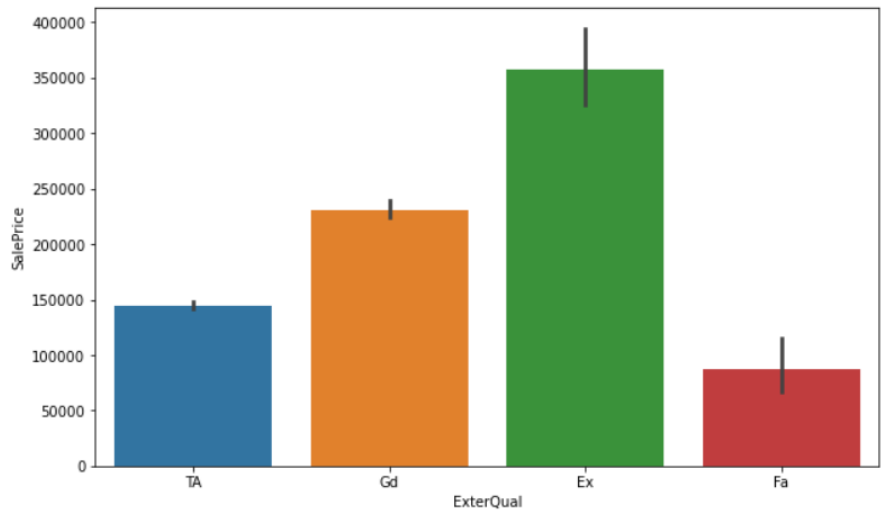

```
MSZoning
C (all)     75208.888889
FV         209478.461538
RH         131558.375000
RL         191004.181034
RM         129070.975460
Name: SalePrice, dtype: float64
```



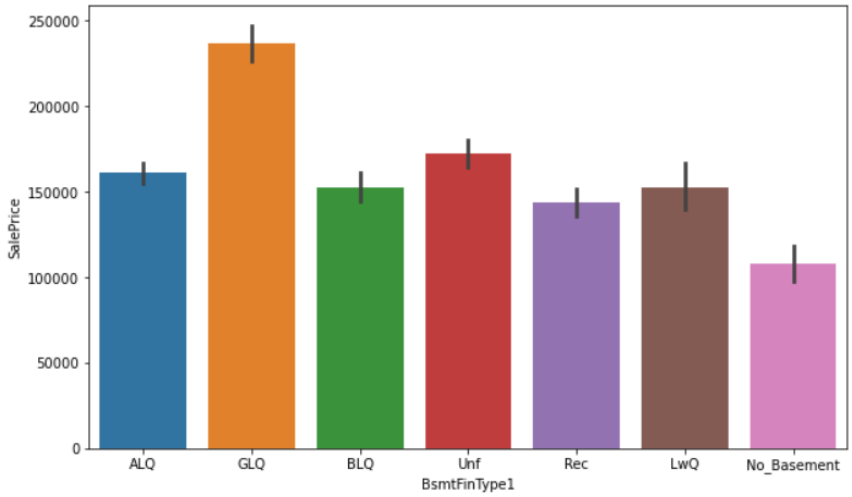

```
LotConfig
Corner     182427.635135
CulDSac    224594.463768
FR2        171138.636364
FR3        221500.000000
Inside     178003.105701
```



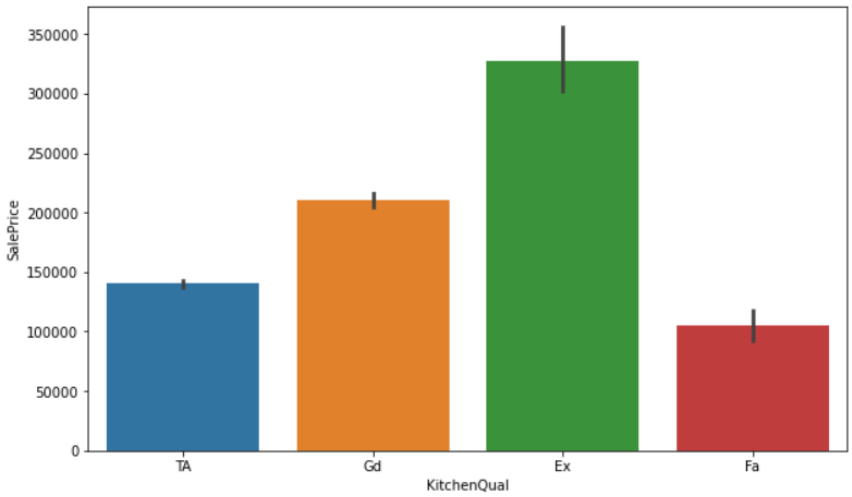

```
RoofMatl
ClyTile    160000.000000
CompShg    180009.329545
Membran    241500.000000
Metal      180000.000000
Roll       137000.000000
Tar&Grv    195747.000000
WdShake    254250.000000
```




```
ExterQual
Ex    357817.651163
Fa     87435.727273
Gd    231009.743073
TA    144918.163180
```




```
BsmtFinType1
ALQ            161003.879310
BLQ            152803.396694
GLQ            236871.596970
LwQ            152712.711864
No_Basement    107897.500000
Rec            143592.844037
```




```
KitchenQual
Ex    327812.329268
Fa    105028.100000
Gd    211076.234310
TA    140206.313149
Name: SalePrice, dtype: float64
```

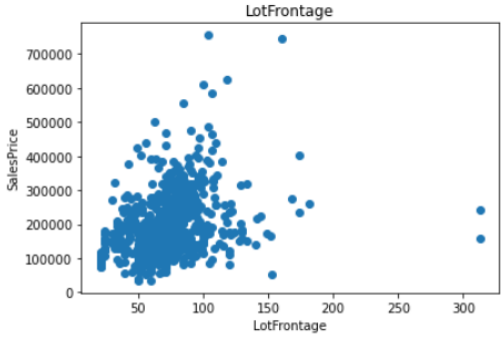
Observation:

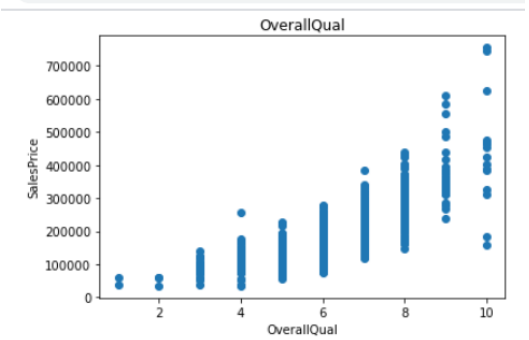
1. MSZoning:The avg sale price of the house is maximum in FV(Floating Village Residential) foloowed by RL(Residential Low Density) zone
2. Street:The property that have access to paved road have much higher average sale price as compared to that with gravel street
3. Alley:houses that do not have access 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)

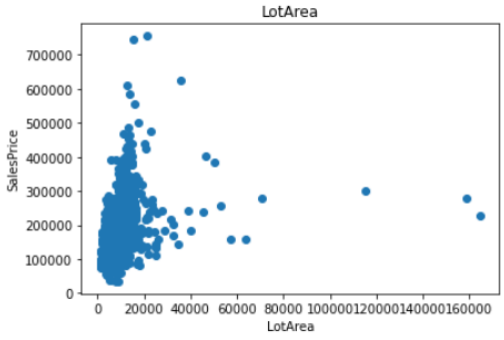
8. Condition1:house that is RRAn(Adjacent to North-South Railroad) has hightest avg sale price followed by PosA(Adjacent to postive off-site feature) while houses that is Artery(Adjacent to arterial street) has a minimum average sale price.
9. BldgType: Type of dwelling-:TwnhsE(Townhouse End Unit) & 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

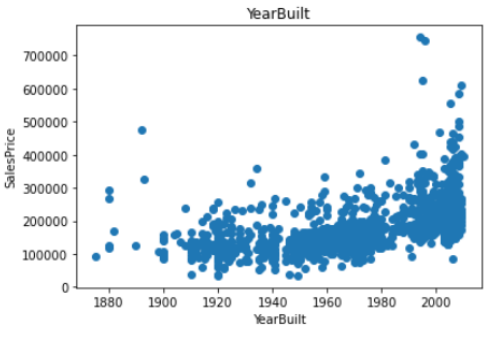
 # visualising sale price wrt to each continuos feature

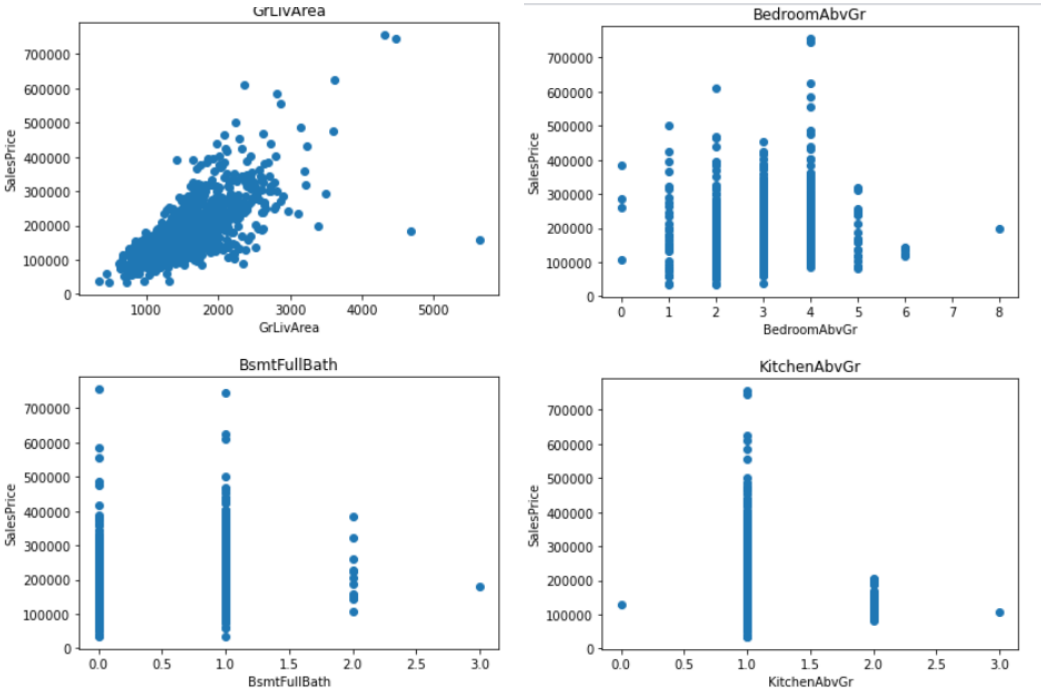
```
for i in cont:
    plt.scatter(df[i],df['SalePrice'])
    plt.xlabel(i)
    plt.ylabel('SalesPrice')
    plt.title(i)
    plt.show()
```











Observation:

- 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
- 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
- OverallQual: Rates the overall material and finish of the house-:Overall quality is directly proportional to the sale price of houses
- 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
- 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
- Kitchen: Kitchens above grade-:houses with 1 kitchen above ground have high sale price in comparison to those having 2 kitchens
- Fireplaces: Number of fireplaces-:Houses with 1 and 2 fireplaces have higher prices in comparion to houses having 0 or 3 fireplaces
- Wood deck,Enclosed porch,Three season porch, screen porch,pool area,Miscval do not have impact on sale price

State the set of assumptions (if any) related to the problem under consideration

- **Assumption while replacing NaN values for each continuous variable with mean value considering it will give accurate predictions.**
- **Some attributes like 'Id' were dropped since they were irrelevant looking at the problem statement.**
- **Attributes like 'MSSubClass' ,'OverallCond' ,'OverallCond' ,'LowQualFinSF' ,'BsmtHalfBath' ,'YrSold', 'MiscVal', 'MoSold', '3SsnPorch' were removed since they were least correlated with target variable.**
- **Attributes which had more than 99% NaN values .ie , 'PoolQC',' were dropped so that my predictions can be more .**

- Hardware and Software Requirements and Tools Used

• Framework-Annconda
• IDE-Jupyter –Notebook
• Coding Language-Python
• Hardware used: system memory 8GB,
• Processor: core i3 Libraries used: Below are the python library used in this project.
• Pandas: To read the Data file in form of data.
• Matplotlib: This library is typically used to plot the figures for better visualisation of data.
• Seaborn: A advanced version of Matplotlib
• Scikit Learn: This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this it contains very useful joblib library for serialization purpose using which final model have been saved in this project

Model/s Development and Evaluation

Identification of possible problem

First I have train data on train.csv dataset and then predict on test.csv file.

Understanding the problem is a crucial first step in solving any problem.Since it was clear that it's a regression type problem. Therefore I decided to run my preprocessed dataset on 9 regression algorithms and then to get observationsfrom their results.

Converting categorical feature into numeric

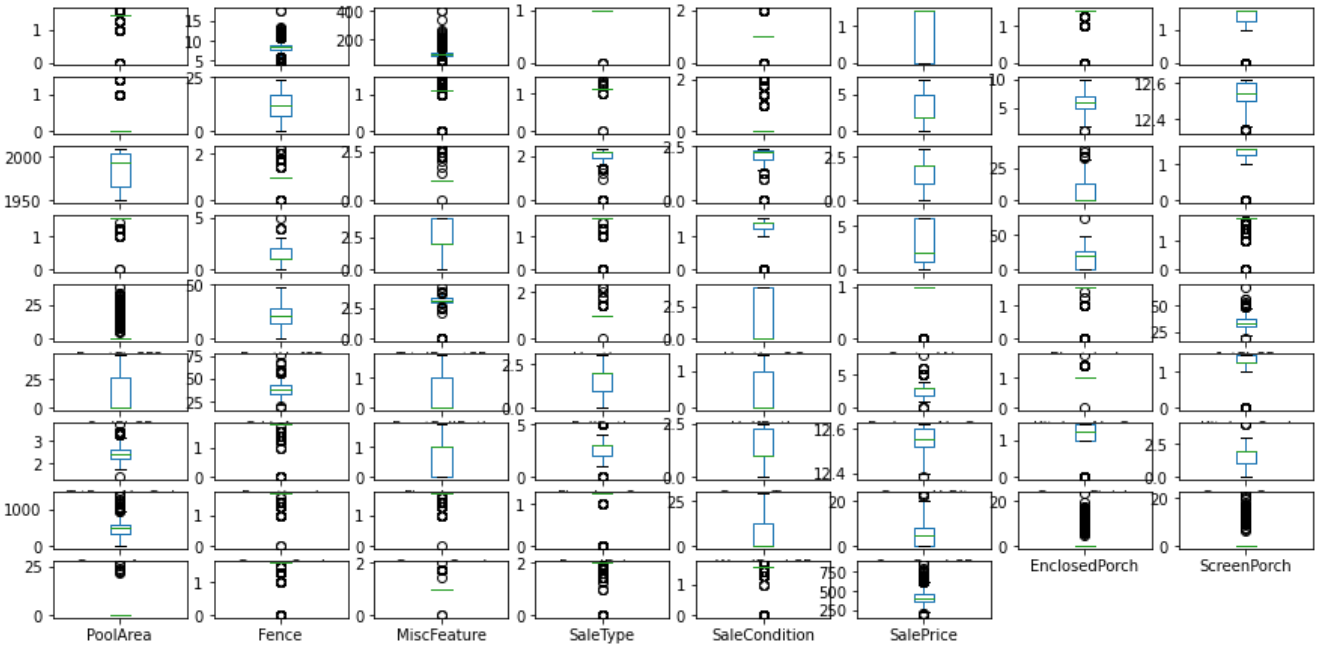
```
#transform non numeric column into numeric one
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in df.columns:
    if df[col].dtype==np.number:
        continue
    df[col]=le.fit_transform(df[col])
```

```
#checking & removing skewness
df.skew()
```

```
#treating skewness via squareroot method and cube root method
df.skew()
for col in df.skew().index:
    if col in df.describe().columns:
        if df[col].skew()>0.55:
            df[col]=np.sqrt(df[col])
        if df[col].skew()<-0.55:
            df[col]=np.cbrt(df[col])
```

checking outliers

```
df.plot(kind='box',subplots=True,layout=(12,8),figsize=(15,10))
```




```
#removing outliers
from scipy.stats import zscore
z_score=abs(zscore(df))
print(df.shape)

df_new=df.loc[(z_score<6).all(axis=1)]
print(df_new.shape)
```

```
(1168, 70)
(1084, 70)
```

Feature Selection

```
#spliting the data into input and output variable
x=df_new.drop('SalePrice',axis=1)
y=df_new['SalePrice']


from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(df_new)
x=pd.DataFrame(x,columns=df_new.columns)
x.head()


# PCA is required for the analysis to reduce curse of Dimensionality & at the same time minimizing information loss
from sklearn.decomposition import PCA
for i in range(20,50):
    pca = PCA(n_components=i)
    x_pca=pca.fit_transform(x)
    print(i," variance :{}".format(np.sum(pca.explained_variance_ratio_)))
# Selecting 49 features, as it explains 94% of data
```

```
pca = PCA(n_components=49)
x=pca.fit_transform(x)
```

Testing of Identified Approaches (Algorithms)

```
from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor

#Importing boosting model
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
import xgboost as xgb

#import error metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.model_selection import GridSearchCV,cross_val_score
```

Selecting best parameter after doing Hyperparameter tuning using gridsearchcv

```
#models with is best parameters
lg= LinearRegression()
ridge=Ridge(alpha= 1, random_state= 42)
lasso=Lasso(alpha= 1, random_state= 42)
knn=KNeighborsRegressor(algorithm= 'auto', n_neighbors= 20)
svr=SVR(C= 10, kernel= 'linear')
dtc=DecisionTreeRegressor(criterion= 'mse', random_state= 66, splitter='best')
rfr=RandomForestRegressor(bootstrap= True, max_features= 'auto', min_samples_leaf= 1, min_samples_spli
XGB = xgb.XGBRegressor(base_score= 0.5, booster= 'gblinear')
gbr=GradientBoostingRegressor(learning_rate=0.01, max_depth= 4, n_estimators= 1000, subsample=0.5)
```

All algorithm by using for loop

```
#All Algorithm by using for loop

model=[lg,knn,svr,dtc,lasso,ridge,
       rfr,gbr,XGB]
rmse = []
cvs=[]
r2score=[]
mse=[]
mae=[]
for m in model:
    m.fit(x_train,y_train)
    score=m.score(x_train,y_train)
    predm=m.predict(x_test)
    print('score of ',m,'is',score)
    r2s=r2_score(y_test,predm)
    print("r2 score is: ",r2s)
    r2score.append(r2s)
    MAE=mean_absolute_error(y_test,predm)
    print("Mean_absolute_error: ",MAE)
    mae.append(MAE)
    MSE=mean_squared_error(y_test,predm)
    print("Mean_squared_error: ",MSE)
    mse.append(MSE)
    rmse1=np.sqrt(mean_squared_error(y_test,predm))
    print("root Mean squared error: ",rmse1)
    rmse.append(rmse1)
    print('**********************************************************************************************')
```

```
result = pd.DataFrame({'Model': model, 'r2_score':r2score,'Mean_absolute_error':mae,'Mean_squared_error':mse,'root_mean_squa
result
```

	Model	r2_score	Mean_absolute_error	Mean_squared_error	root_mean_squared_error
0	LinearRegression()	0.950907	13.670331	320.299132	17.896903
1	KNeighborsRegressor(n_neighbors=20)	0.871234	21.440186	840.109923	28.984650
2	SVR(C=10, kernel='linear')	0.950951	13.938226	320.008542	17.888783
3	DecisionTreeRegressor(random_state=66)	0.788942	27.654674	1377.003834	37.108002
4	Lasso(alpha=1, random_state=42)	0.950587	13.759894	322.388010	17.955167
5	Ridge(alpha=1, random_state=42)	0.950912	13.670454	320.263146	17.895897
6	(DecisionTreeRegressor(max_features='auto', ra...	0.929631	15.948460	459.105811	21.426755
7	([DecisionTreeRegressor(criterion='friedman_ms...	0.945414	14.065580	356.135703	18.871558
8	XGBRegressor(base_score=0.5, booster='gblinear...	0.950907	13.670335	320.299245	17.896906

Cross validation

```python
#cross validate the models
from sklearn.model_selection import cross_val_score
model=[lg,knn,svr,dtc,Lasso(),Ridge(),
       rfr,gbr,XGB]
for m in model:
    score=cross_val_score(m,x,y,cv=5,scoring='r2')
    print('score of ',m,'is:')
    print('score:',score)
    print('mean score:',score.mean())
    print('standard deviation:',score.std())
    print('**************************************************')
    print('\n')
```

```
score of  LinearRegression() is 0.9099198680112668
r2 score is:  0.9509066950646884
Mean_absolute_error:  13.670330927577726
Mean_squared_error:  320.29913177588764
root Mean squared error:  17.896902854289834
****************************************************************************************


score of  KNeighborsRegressor(n_neighbors=20) is 0.8209529424805423
r2 score is:  0.8712335796109671
Mean_absolute_error:  21.44018593211921
Mean_squared_error:  840.1099234781913
root Mean squared error:  28.984649790504477
****************************************************************************************


score of  SVR(C=10, kernel='linear') is 0.9048481094097038
r2 score is:  0.9509512346428354
Mean_absolute_error:  13.938226393247984
Mean_squared_error:  320.00854249433576
root Mean squared error:  17.88878258838023
****************************************************************************************


score of  DecisionTreeRegressor(random_state=66) is 1.0
r2 score is:  0.7889420781400341
Mean_absolute_error:  27.654673900162148
Mean_squared_error:  1377.0038341327054
root Mean squared error:  37.10800229239921
****************************************************************************************
```

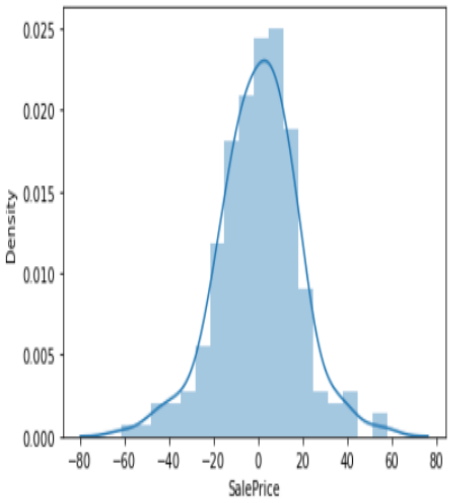
```
score of  Lasso(alpha=1, random_state=42) is 0.9038449239949256
r2 score is:  0.9505865258538466
Mean_absolute_error:  13.759893790230072
Mean_squared_error:  322.388009686819
root Mean squared error:  17.95516665717194
**********************************************************************************************


score of  Ridge(alpha=1, random_state=42) is 0.9099197334396052
r2 score is:  0.9509122106545963
Mean_absolute_error:  13.670453956578092
Mean_squared_error:  320.26314644833496
root Mean squared error:  17.895897475352694
**********************************************************************************************


score of  RandomForestRegressor() is 0.9808116879710892
r2 score is:  0.9296313372026026
Mean_absolute_error:  15.948459615015931
Mean_squared_error:  459.10581143264625
root Mean squared error:  21.426754570691433
**********************************************************************************************


score of  GradientBoostingRegressor(learning_rate=0.01, max_depth=4, n_estimators=1000,
                          subsample=0.5) is 0.9894883465311678
r2 score is:  0.945413905605343
Mean_absolute_error:  14.065579598368311
Mean_squared_error:  356.1357025093972
root Mean squared error:  18.871558930797052
```

4 models have approximately same r2_score ,therefore I choose ridge as the final model

FINAL MODEL

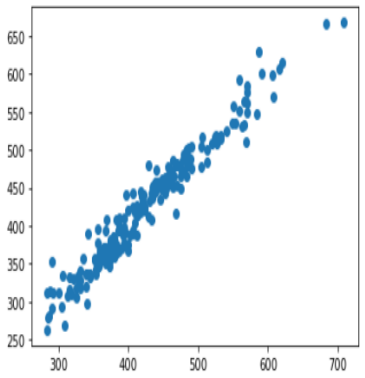
```
ridge = Ridge(alpha=1,random_state=42)
ridge.fit(x_train,y_train)

y_train_pred = ridge.predict(x_train)
print(r2_score(y_train,y_train_pred))
y_test_pred = ridge.predict(x_test)
print(r2_score(y_test,y_test_pred))
```

```
0.9099197334396052
0.9509122106545963
```




```
plt.scatter(y_test,y_test_pred)
```

```
<matplotlib.collections.PathCollection at 0x2a6ab955
```



#Saving the best model

```
import joblib
joblib.dump(ridge,'House_Price_Prediction.pkl')
```

```
['House_Price_Prediction.pkl']
```

Predicting for Test Data

All the preprocessing step performed fot train data .ie df has been performed for test data .ie df1 also

```
model = joblib.load('House_Price_Prediction.pkl')
sale_price = model.predict(df1)
print(sale_price)
```

```
[545.01090686 455.78795526 517.66650704 383.95205608 528.91901647
 302.84542319 404.64193705 502.62511555 443.99290948 440.28701782
 211.02571117 362.91847806 314.07999814 364.71674726 523.35723926
 327.91223358 384.13471194 342.61046898 479.40793957 458.19937532
 340.26347932 404.37554562 400.8670139  332.76539424 283.98831272
 387.07798183 456.00743359 361.24591693 432.29009113 258.67457125
 422.67117226 457.04347587 482.83600423 388.06553913 257.15971854
 472.57957002 484.27096361 352.85331732 436.30671736 411.15355443
 290.31876264 514.50989474 513.86607834 476.84181045 434.82093512
 340.97906146 347.84698442 297.76209416 493.13810044 561.28163835
 371.16668558 435.13274097 302.71964078 276.14426963 522.75539241
 362.73953833 336.50082976 481.85899304 313.9855975  516.77387598
 346.14658105 461.01375595 352.00080328 420.84477863 466.2803982
 335.15393513 427.72952432 429.10642388 405.67338715 351.47076189
 502.68426327 403.55264329 348.17159607 404.67971105 384.00016118
 520.47878267 529.82165977 479.67226687 513.80228367 386.84664607
 523.11411751 352.36417301 421.06571538 380.94549468 441.30434974
 459.93284662 272.82292065 550.1546655  405.091617   476.14353015
 500.72507139 382.80043985 371.04038768 395.59185044 471.94505569
 424.12817295 520.40364912 471.05311449 559.22100878 384.11276377
```

Conclusion

- we built regression models to predict the price of some house given some of the house features. We evaluated and compared each model to determine the one with highest performance.
- The saved model now can help to give an estimate of House prices with respect to the the data fed to the model and hence will be helpful in making decision