FLIP ROBO

NAME OF THE PROJECT

Surprise Housing Price Prediction Project

Submitted by:

Shilpi Mohanty

ACKNOWLEDGMENT

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 and also for their support in completing the project.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to Mr. Mohd. Kharif, SME for giving the dataset and clear instructions to perform the complete case study process and guided and advised me from time to time.

I perceive as this opportunity as a learning experience in my career development. I will strive to use gained skills and knowledge in the best possible way, and try to work on the improvement, in order to attain desired career objectives.

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

- **Review of Literature**

The sample dataset which is provided to us by client database where we came to know that the US housing company is looking at prospective properties to buy houses to enter the market.

Looking at the dataset which clearly says that this is a regression problem as we have to predict the target variable “SalePrice” which is continuous value so we need to build a model using Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not.

Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the house.

- ## Motivation for the Problem Undertaken

  Real estate sector is one of the most globally recognized sectors. It is increasingly expanding and developing, playing a more important role in the country's market economy thereby contributing to the socio-economic stability. It comprises of four sub sectors - housing, retail, hospitality, and commercial.

  Here we will be focusing about Housing. With the rapid population growth, the need is arising for housing. It is critical to provide accurate predictions of housing prices. Housing price Index is 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.

  Our main objective is to build a model to predict the house prices with the help of features given and predict it by using Machine Learning algorithms.

  The sample data is provided by 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.

  There has been large number of papers following the traditional machine learning approach to predict the housing prices and concerned very little for individual models' performance and neglect the complex models. So here we will be focussing on both traditional and advance machine learning models and compare them. This paper will also comprehensively validate multiple techniques in model implementation on regression and provide an optimistic result for housing price prediction.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

Referring to the problem statement which aim is to predict the sales price of the house which will help the housing company to invest or not in Australian market. And also, to find out which features are more important to the sales price and how these features describe the sales price. After understanding this, they will accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Considering the above which clearly states that this case is of Regression Analysis Problem as we have to predict the sales price which is of continuous nature.

Regression Analysis: Regression analysis refers to assessing the relationship between the outcome variable and one or more variables. The outcome variable is known as the dependent or response variable and the risk elements, and co-founders are known as predictors or independent variables. The dependent variable is shown by “y” and independent variables are shown by “x” in regression analysis.

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 helpful statistical method that can be leveraged across an organization to determine the degree to which particular independent variables are influencing dependent variables.

The possible scenarios for conducting regression analysis to yield valuable, actionable business insights are endless and thus make more informed business decisions, allocate resources more efficiently, and ultimately boost your bottom line.

- ## Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of data is 1168 rows and 81 columns. There are 2 data sets that are given. One is training data and one is testing data.

1) Train file will be used for training the model, i.e., the model will learn from this file. It contains all the independent variables and the target variable. Size of training set: 1168 records.

2) Test file contains all the independent variables, but not the target variable. We will apply the model to predict the target variable for the test data. Size of test set: 292 records.

Snapshot of the data as below:

Train dataset:

```
#Loading the Train dataset
df_train=pd.read_csv('train.csv')
df_train
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Edwards	Feedr	
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	NPkVill	Norm	
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	Inside	Gtl	IDOTRR	Feedr	
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	

1168 rows × 81 columns

Test Dataset:

```
#Loading the Test dataset
df_test=pd.read_csv('test.csv')
df_test
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr
5	650	180	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	Norm
6	1453	180	RM	35.0	3675	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Edwards	Norm
7	152	20	RL	107.0	13891	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	NridgHt	Norm
8	427	80	RL	NaN	12800	Pave	NaN	Reg	Low	AllPub	Inside	Mod	SawyerW	Norm
9	776	120	RM	32.0	4500	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Mitchel	Norm
10	30	30	RM	60.0	6324	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	BrkSide	Feedr

In the above test dataset,we can see there are 292 rows and 80 columns ,as this is used for prediction of model so target column is not there. therefore it consists of only the feature columns where the target label which is "SalePrice" to be predicted.we can see first five rows and last 5 rows here.

Data description

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

MSSubClass: Identifies the type of dwelling involved in the sale.

20	1-STORY 1946 & NEWER ALL STYLES
30	1-STORY 1945 & OLDER
40	1-STORY W/FINISHED ATTIC ALL AGES
45	1-1/2 STORY - UNFINISHED ALL AGES
50	1-1/2 STORY FINISHED ALL AGES
60	2-STORY 1946 & NEWER
70	2-STORY 1945 & OLDER
75	2-1/2 STORY ALL AGES
80	SPLIT OR MULTI-LEVEL
85	SPLIT FOYER
90	DUPLEX - ALL STYLES AND AGES
120	1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150	1-1/2 STORY PUD - ALL AGES
160	2-STORY PUD - 1946 & NEWER
180	PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190	2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

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

LotFrontage: Linear feet of street connected to property

LotArea	Lot size in square feet
Street	Type of road access to property
Grvl	Gravel
Pave	Paved

Alley: Type of alley access to property

Grvl	Gravel
Pave	Paved
NA	No alley access

LotShape: General shape of property

Reg	Regular
IR1	Slightly irregular
IR2	Moderately Irregular
IR3	Irregular

LandContour: Flatness of the property

Lvl	Near Flat/Level
Bnk	Banked - Quick and significant rise from street grade to building
HLS	Hillside - Significant slope from side to side
Low	Depression

Utilities: Type of utilities available

AllPub	All public Utilities (E,G,W,& S)
NoSewr	Electricity, Gas, and Water (Septic Tank)
NoSeWa	Electricity and Gas Only
ELO	Electricity only

LotConfig: Lot configuration

Inside	Inside lot
Corner	Corner lot
CulDSac	Cul-de-sac
FR2	Frontage on 2 sides of property
FR3	Frontage on 3 sides of property

LandSlope: Slope of property

Gtl	Gentle slope
Mod	Moderate Slope
Sev	Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn	Bloomington Heights
Blueste	Bluestem
BrDale	Briardale
BrkSide	Brookside
ClearCr	Clear Creek
CollgCr	College Creek
Crawfor	Crawford
Edwards	Edwards
Gilbert	Gilbert
IDOTRR	Iowa DOT and Rail Road
MeadowV	Meadow Village
Mitchel	Mitchell
Names	North Ames
NoRidge	Northridge
NPkVill	Northpark Villa
NridgHt	Northridge Heights
NWAmes	Northwest Ames
OldTown	Old Town
SWISU	South & West of Iowa State University
Sawyer	Sawyer
SawyerW	Sawyer West
Somerst	Somerset
StoneBr	Stone Brook
Timber	Timberland
Veenker	Veenker

Condition1: Proximity to various conditions

Artery	Adjacent to arterial street
Feedr	Adjacent to feeder street
Norm	Normal
RRNn	Within 200' of North-South Railroad
RRAn	Adjacent to North-South Railroad
PosN	Near positive off-site feature--park, greenbelt, etc.
PosA	Adjacent to postive off-site feature
RRNe	Within 200' of East-West Railroad
RRAe	Adjacent to East-West Railroad

Condition2: Proximity to various conditions (if more than one is present)

Artery	Adjacent to arterial street
Feedr	Adjacent to feeder street
Norm	Normal
RRNn	Within 200' of North-South Railroad
RRAn	Adjacent to North-South Railroad
PosN	Near positive off-site feature--park, greenbelt, etc.
PosA	Adjacent to postive off-site feature
RRNe	Within 200' of East-West Railroad
RRAe	Adjacent to East-West Railroad

BldgType: Type of dwelling

1Fam	Single-family Detached
2FmCon	Two-family Conversion; originally built as one-family dwelling
Duplx	Duplex
TwnhsE	Townhouse End Unit
TwnhsI	Townhouse Inside Unit

HouseStyle: Style of dwelling

1Story	One story
1.5Fin	One and one-half story: 2nd level finished
1.5Unf	One and one-half story: 2nd level unfinished
2Story	Two story
2.5Fin	Two and one-half story: 2nd level finished
2.5Unf	Two and one-half story: 2nd level unfinished
SFoyer	Split Foyer
SLvl	Split Level

OverallQual: Rates the overall material and finish of the house

10	Very Excellent
9	Excellent
8	Very Good
7	Good
6	Above Average
5	Average
4	Below Average
3	Fair
2	Poor
1	Very Poor

OverallCond: Rates the overall condition of the house

10	Very Excellent
9	Excellent
8	Very Good
7	Good
6	Above Average
5	Average
4	Below Average
3	Fair
2	Poor
1	Very Poor

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat	Flat
Gable	Gable
Gambrel	Gabrel (Barn)
Hip	Hip
Mansard	Mansard
Shed	Shed

RoofMatl: Roof material

ClyTile	Clay or Tile
CompShg	Standard (Composite) Shingle
Membran	Membrane
Metal	Metal
Roll	Roll
Tar&Grv	Gravel & Tar
WdShake	Wood Shakes
WdShngl	Wood Shingles

Exterior1st: Exterior covering on house

AsbShng	Asbestos Shingles
AsphShn	Asphalt Shingles
BrkComm	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
CemntBd	Cement Board
HdBoard	Hard Board
ImStucc	Imitation Stucco
MetalSd	Metal Siding
Other	Other
Plywood	Plywood
PreCast	PreCast
Stone	Stone
Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng	Asbestos Shingles
AsphShn	Asphalt Shingles
BrkComm	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
CemntBd	Cement Board
HdBoard	Hard Board
ImStucc	Imitation Stucco
MetalSd	Metal Siding
Other	Other
Plywood	Plywood
PreCast	PreCast
Stone	Stone
Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
None	None
Stone	Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

ExterCond: Evaluates the present condition of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

Foundation: Type of foundation

BrkTil	Brick & Tile
CBlock	Cinder Block
PConc	Poured Contrete
Slab	Slab
Stone	Stone
Wood	Wood

BsmtQual: Evaluates the height of the basement

Ex	Excellent (100+ inches)
Gd	Good (90-99 inches)
TA	Typical (80-89 inches)
Fa	Fair (70-79 inches)
Po	Poor (<70 inches
NA	No Basement

BsmtCond: Evaluates the general condition of the basement

Ex	Excellent
Gd	Good
TA	Typical - slight dampness allowed
Fa	Fair - dampness or some cracking or settling
Po	Poor - Severe cracking, settling, or wetness
NA	No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd	Good Exposure
Av	Average Exposure (split levels or foyers typically score average or above)
Mn	Mimimum Exposure
No	No Exposure
NA	No Basement

BsmtFinType1: Rating of basement finished area

GLQ	Good Living Quarters
ALQ	Average Living Quarters
BLQ	Below Average Living Quarters
Rec	Average Rec Room
LwQ	Low Quality
Unf	Unfinshed
NA	No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ	Good Living Quarters
ALQ	Average Living Quarters

BLQ	Below Average Living Quarters
Rec	Average Rec Room
LwQ	Low Quality
Unf	Unfinshed
NA	No Basement

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor	Floor Furnace
GasA	Gas forced warm air furnace
GasW	Gas hot water or steam heat
Grav	Gravity furnace
OthW	Hot water or steam heat other than gas
Wall	Wall furnace

HeatingQC: Heating quality and condition

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

CentralAir: Central air conditioning

N	No
Y	Yes

Electrical: Electrical system

SBrkr	Standard Circuit Breakers & Romex
FuseA	Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF	60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP	60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix	Mixed

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ	Typical Functionality
Min1	Minor Deductions 1
Min2	Minor Deductions 2
Mod	Moderate Deductions
Maj1	Major Deductions 1
Maj2	Major Deductions 2
Sev	Severely Damaged
Sal	Salvage only

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex	Excellent - Exceptional Masonry Fireplace
Gd	Good - Masonry Fireplace in main level
TA	Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
Fa	Fair - Prefabricated Fireplace in basement
Po	Poor - Ben Franklin Stove
NA	No Fireplace

GarageType: Garage location

2Types	More than one type of garage
Attchd	Attached to home
Basment	Basement Garage
BuiltIn	Built-In (Garage part of house - typically has room above garage)
CarPort	Car Port
Detchd	Detached from home
NA	No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin	Finished
RFn	Rough Finished
Unf	Unfinished
NA	No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

GarageCond: Garage condition

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

PavedDrive: Paved driveway

Y	Paved
P	Partial Pavement
N	Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
NA	No Pool

Fence: Fence quality

GdPrv	Good Privacy
MnPrv	Minimum Privacy
GdWo	Good Wood
MnWw	Minimum Wood/Wire
NA	No Fence

MiscFeature: Miscellaneous feature not covered in other categories

Elev	Elevator
Gar2	2nd Garage (if not described in garage section)
Othr	Other
Shed	Shed (over 100 SF)
TenC	Tennis Court
NA	None

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

WD	Warranty Deed - Conventional
CWD	Warranty Deed - Cash
VWD	Warranty Deed - VA Loan
New	Home just constructed and sold
COD	Court Officer Deed/Estate
Con	Contract 15% Down payment regular terms
ConLw	Contract Low Down payment and low interest
ConLI	Contract Low Interest
ConLD	Contract Low Down
Oth	Other

SaleCondition: Condition of sale

Normal	Normal Sale
Abnorml	Abnormal Sale - trade, foreclosure, short sale
AdjLand	Adjoining Land Purchase
Alloca	Allocation - two linked properties with separate deeds, typically condo with a garage unit
Family	Sale between family members
Partial	Home was not completed when last assessed (associated with New Homes)

Checking datatype and info data

```
: df_train.info() #checking info of the train dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
 14  Condition2     1168 non-null   object
 15  BldgType       1168 non-null   object
 16  HouseStyle     1168 non-null   object
 17  OverallQual    1168 non-null   int64
 18  OverallCond    1168 non-null   int64
 19  YearBuilt      1168 non-null   int64
 20  YearRemodAdd   1168 non-null   int64
 21  RoofStyle      1168 non-null   object
```

```
df_test.info() #chekcing info of the test dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292 entries, 0 to 291
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             292 non-null    int64
 1   MSSubClass     292 non-null    int64
 2   MSZoning       292 non-null    object
 3   LotFrontage    247 non-null    float64
 4   LotArea        292 non-null    int64
 5   Street         292 non-null    object
 6   Alley          14 non-null     object
 7   LotShape       292 non-null    object
 8   LandContour    292 non-null    object
 9   Utilities      292 non-null    object
 10  LotConfig      292 non-null    object
 11  LandSlope      292 non-null    object
 12  Neighborhood   292 non-null    object
 13  Condition1     292 non-null    object
 14  Condition2     292 non-null    object
 15  BldgType       292 non-null    object
 16  HouseStyle     292 non-null    object
 17  OverallQual    292 non-null    int64
 18  OverallCond    292 non-null    int64
 19  YearBuilt      292 non-null    int64
 20  YearRemodAdd   292 non-null    int64
 21  RoofStyle      292 non-null    object
 22  RoofMatl       292 non-null    object
 23  Exterior1st    292 non-null    object
```

- Data Preprocessing Done

Data pre-processing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analysed by computers and machine learning**.**

Raw, real-world data in the form of text, images, video, etc., is messy. It contain errors and inconsistencies, often incomplete, and doesn't have a regular, uniform design.

Machines understands and read data as 1s and 0s. So calculating structured data, like whole numbers and percentages is easy. However,

unstructured data, in the form of text and images must first be cleaned filtered and formatted before analysis.

- Steps in Data Pre-Processing

- **Getting the dataset**
- **Importing libraries**
- **Importing datasets**
- **Finding Missing Data-**
- **Encoding Categorical Data**
- **Splitting dataset into training and test set**
- **Feature scaling**

1.Checking and Handling missing values

```
#checking null values
```

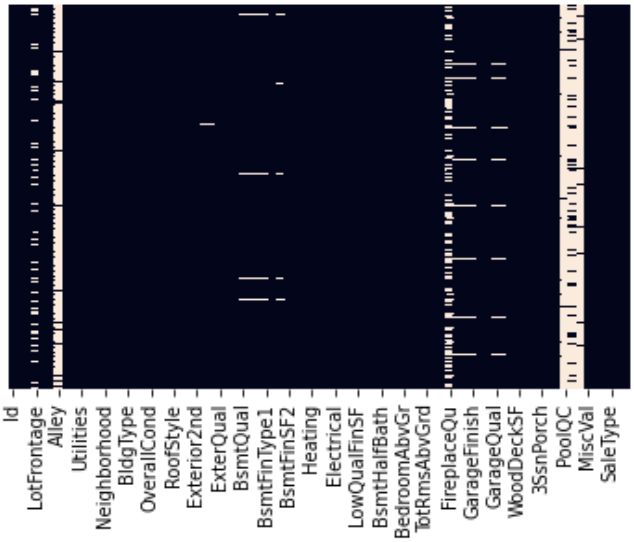
```
df_train.apply(lambda x: sum(x.isnull()))
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     214
LotArea           0
Street            0
Alley          1091
LotShape          0
LandContour       0
Utilities         0
LotConfig         0
LandSlope         0
Neighborhood      0
Condition1        0
Condition2        0
BldgType          0
HouseStyle        0
OverallQual       0
OverallCond       0
YearBuilt         0
YearRemodAdd      0
RoofStyle         0
RoofMatl          0
Exterior1st       0
Exterior2nd       0
MasVnrType        7
MasVnrArea        7
```

```
#checking null values using heatmap
```

```
sns.heatmap(df_train.isnull(),yticklabels=False,cbar=False)
```

```
<AxesSubplot:>
```



Observation: Thus we see that columns like MiscFeature,PoolQC,Alley,Fence are having missing value more than 50% ,so we will drop these columns, and for rest we will impute with mean or mode imputation technique as shown below.

Handling missing value by imputing mean in numerical datatype and mode imputation in categorical datatypes.

```
]: df_train.drop(['Alley','PoolQC','MiscFeature','Fence'],axis=1,inplace=True)
```

```
]: df_train["LotFrontage"].fillna(df_train["LotFrontage"].mean(),inplace=True)
```

```
]: df_train["MasVnrArea"].fillna(df_train["MasVnrArea"].mean(),inplace=True)
```

```
]: df_train["GarageYrBlt"].fillna(df_train["GarageYrBlt"].mean(),inplace=True)
```

```
]: df_train["BsmtQual"].fillna(df_train["BsmtQual"].mode()[0],inplace=True)
   df_train["BsmtCond"].fillna(df_train["BsmtCond"].mode()[0],inplace=True)
   df_train["BsmtExposure"].fillna(df_train["BsmtExposure"].mode()[0],inplace=True)
   df_train["BsmtFinType1"].fillna(df_train["BsmtFinType1"].mode()[0],inplace=True)
   df_train["BsmtFinType2"].fillna(df_train["BsmtFinType2"].mode()[0],inplace=True)
   df_train["FireplaceQu"].fillna(df_train["FireplaceQu"].mode()[0],inplace=True)
   df_train["GarageType"].fillna(df_train["GarageType"].mode()[0],inplace=True)
   df_train["GarageFinish"].fillna(df_train["GarageFinish"].mode()[0],inplace=True)
   df_train["GarageQual"].fillna(df_train["GarageQual"].mode()[0],inplace=True)
   df_train["GarageCond"].fillna(df_train["GarageCond"].mode()[0],inplace=True)
```

```
]: df_train["MasVnrType"].fillna(df_train["MasVnrType"].mode()[0],inplace=True)
```

Checking the statistical summary of the dataset

The Describe function returns the statistical summary of the dataframe or series. It Analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided.

For numeric data, the result's index will include count, mean, std, min, max as well as lower, 50 and upper percentiles. By default the lower percentile is 25 and the upper percentile is 75. The 50 percentile is the same as the median.

For object data (e.g. strings or timestamps), the result's index will include count, unique, top, and freq. The top is the most common value. The freq is the most common value's frequency. Timestamps also include the first and last items.

```
df_train.describe() #it provides statistical information about the numerical datatypes.
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	724.136130	56.767979	70.988470	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	46.647260
std	416.159877	41.940650	22.437056	8957.442311	1.390153	1.124343	30.145255	20.785185	182.047152	462.664785	163.520016
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000
25%	360.500000	20.000000	60.000000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000
50%	714.500000	50.000000	70.988470	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000
75%	1079.500000	70.000000	79.250000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000
max	1460.000000	190.000000	313.000000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000

Observation:

1.Maximum standard deviation of 8957.44 is observed in LotArea column.

2.Maximum SalePrice of a house seen is 755000 and minimum is 34900.

3.The columns where salesprice mean is considerably greatet than median are MSSubclass, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, Miscval,which shows that the columns are positively skewed.

4.The columns where Median is greater than mean are FullBath, BedroomAbvGr, Fireplaces, Garagecars, GarageArea, YrSold Median which shows that the columns are negatively skewed.

5.Thus we can see that high gap between max and 75% percentile in most of the columns like MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, MiscVal, SalePrice shows that outliers are present.

6.Most of the columns are not normally distributed or close to normal distribution.

7.id column is an identifier and irrelevant with respect to model training so we can drop this column.

```
# description of the categorcal features
df_train.describe(include=np.object)
```

	MSZoning	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle	RoofStyle
count	1168	1168	1168	1168	1168	1168	1168	1168	1168	1168	1168	1168	1168
unique	5	2	4	4	1	5	3	25	9	8	5	8	6
top	RL	Pave	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	Norm	1Fam	1Story	Gable
freq	928	1164	740	1046	1168	842	1105	182	1005	1154	981	578	915

Observation: Thus we see in utilities columns ,there only one constant value so we will drop this column as this will be irrelevant.

Checking Unique values in all Columns

```
df_train.nunique().to_frame("Unique Values") #gives all the distinct values of all the columns
```

	Unique Values
Id	1168
MSSubClass	15
MSZoning	5
LotFrontage	107
LotArea	892
Street	2
LotShape	4
LandContour	4
Utilities	1
LotConfig	5
LandSlope	3
Neighborhood	25

Thus we see that Id is a constant identifier and irrelevant for model training so dropping the same.

Dropping the unnecessary Columns

```
#Dropping Utilities column
df_train.drop(['Utilities'],axis=1,inplace=True)
```

```
#dropping Id column
df_train.drop('Id',axis=1,inplace=True)
```

Checking duplicate Values

```
: df_train.duplicated().sum()
```

```
: 0
```

Correlation Analysis:

Correlation analysis is a statistical method used to measure the strength of the linear relationship between two variables and compute their association.

A positive correlation indicates that the values tend to increase with one another and a negative correlation indicates that values in one set tend to decrease with an increase in the other set.

```
#Checking correlation of the dataset
corr=df_train.corr()  #corr() function provides the correlation value of each column
corr
```

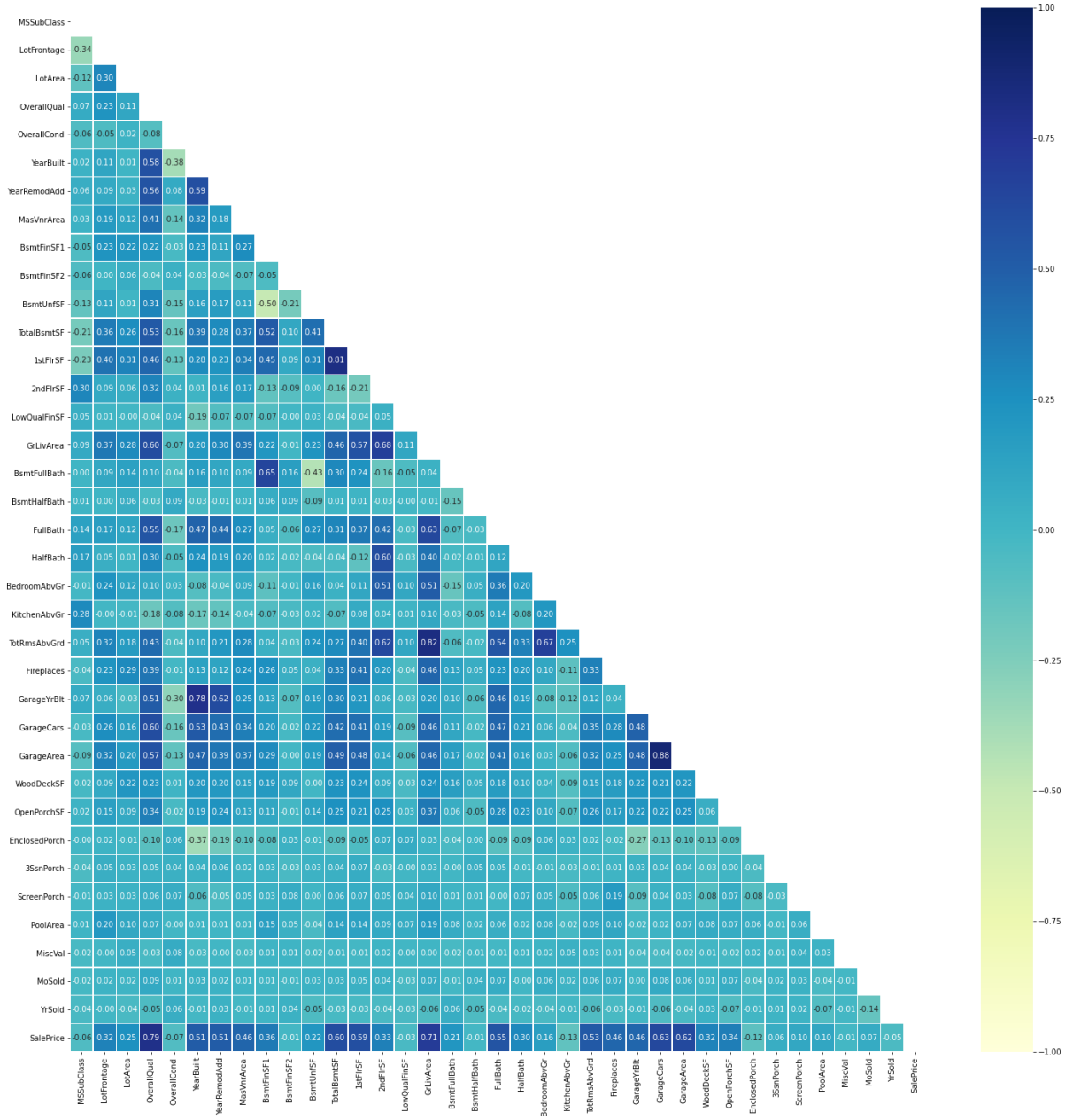
	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF
MSSubClass	1.000000	-0.336681	-0.124151	0.070462	-0.056978	0.023988	0.056618	0.027813	-0.052236	-0.062403	-0.134170
LotFrontage	-0.336681	1.000000	0.299452	0.229218	-0.047573	0.112655	0.088799	0.188758	0.228996	0.002159	0.113924
LotArea	-0.124151	0.299452	1.000000	0.107188	0.017513	0.005506	0.027228	0.121086	0.221851	0.056656	0.006600
OverallQual	0.070462	0.229218	0.107188	1.000000	-0.083167	0.575800	0.555945	0.407230	0.219643	-0.040893	0.308676
OverallCond	-0.056978	-0.047573	0.017513	-0.083167	1.000000	-0.377731	0.080669	-0.137475	-0.028810	0.044336	-0.146384
YearBuilt	0.023988	0.112655	0.005506	0.575800	-0.377731	1.000000	0.592829	0.321905	0.227933	-0.027682	0.155559
YearRemodAdd	0.056618	0.088799	0.027228	0.555945	0.080669	0.592829	1.000000	0.181385	0.114430	-0.044694	0.174732
MasVnrArea	0.027813	0.188758	0.121086	0.407230	-0.137475	0.321905	0.181385	1.000000	0.265735	-0.065707	0.109562
BsmtFinSF1	-0.052236	0.228996	0.221851	0.219643	-0.028810	0.227933	0.114430	0.265735	1.000000	-0.052145	-0.499861
BsmtFinSF2	-0.062403	0.002159	0.056656	-0.040893	0.044336	-0.027682	-0.044694	-0.065707	-0.052145	1.000000	-0.213580
BsmtUnfSF	-0.134170	0.113924	0.006600	0.308676	-0.146384	0.155559	0.174732	0.109562	-0.499861	-0.213580	1.000000
TotalBsmtSF	-0.214042	0.356107	0.259733	0.528285	-0.162481	0.386265	0.280720	0.365016	0.518940	0.098167	0.414186
1stFlrSF	-0.227927	0.403436	0.312843	0.458758	-0.134420	0.279450	0.233384	0.337143	0.445876	0.093442	0.307437
2ndFlrSF	0.300366	0.089675	0.059803	0.316624	0.036668	0.011834	0.155102	0.172741	-0.127656	-0.092049	0.002736
LowQualFinSF	0.053737	0.007506	-0.001915	-0.039295	0.041877	-0.189044	-0.072526	-0.070515	-0.070932	-0.000577	0.030088
GrLivArea	0.086448	0.374251	0.281360	0.599700	-0.065006	0.198644	0.295048	0.386791	0.217160	-0.007484	0.232920
BsmtFullBath	0.004556	0.094046	0.142387	0.101732	-0.039680	0.164983	0.104643	0.086444	0.645126	0.163518	-0.431740
BsmtHalfBath	0.008207	0.001389	0.059282	-0.030702	0.091016	-0.028161	-0.011375	0.014196	0.063895	0.093692	-0.090372

Correlation using a Heatmap

A heat map (or heatmap) is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions. The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space**.**

Correlation heatmaps are a type of plot that visualize the strength of relationships between numerical variables. Correlation plots are used to understand which variables are related to each other and the strength of this relationship.The rows represent the relationship between each pair of variables. The values in the cells indicate the strength of the relationship, with positive values indicating a positive relationship and negative values indicating a negative relationship**.**

```
mask = np.triu(np.ones_like(df_train.corr(), dtype=bool))
fig = plt.subplots(figsize=(25, 25))
sns.heatmap(df_train.corr(), annot=True,fmt='.2f',mask=mask,
            vmin=-1,vmax= +1,cmap="YlGnBu",linewidth=0.5)
plt.show()
```



Observation:

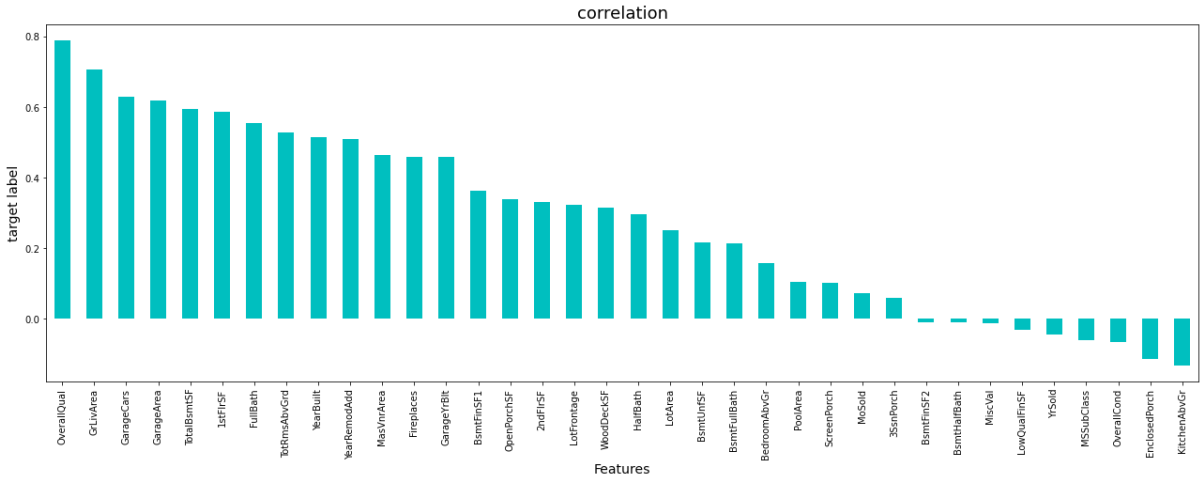
1.SalePrice is highly positively correlated with the features like OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars, GarageArea**.**

2.SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch, YrSold.

3.We observe multicollinearity in between columns ,we may PCA or VIF or feature selection using pearsons correlationm for handling the same

Correlation between target variable 'SalePrice' with Features.

```
#bar plot showing correlation bw target and features
plt.figure(figsize=(22,7))
df_train.corr()['SalePrice'].sort_values(ascending=False).drop(['SalePrice']).plot(kind='bar',color='c')
plt.xlabel('Features',fontsize=14)
plt.ylabel('target label',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



Observation:

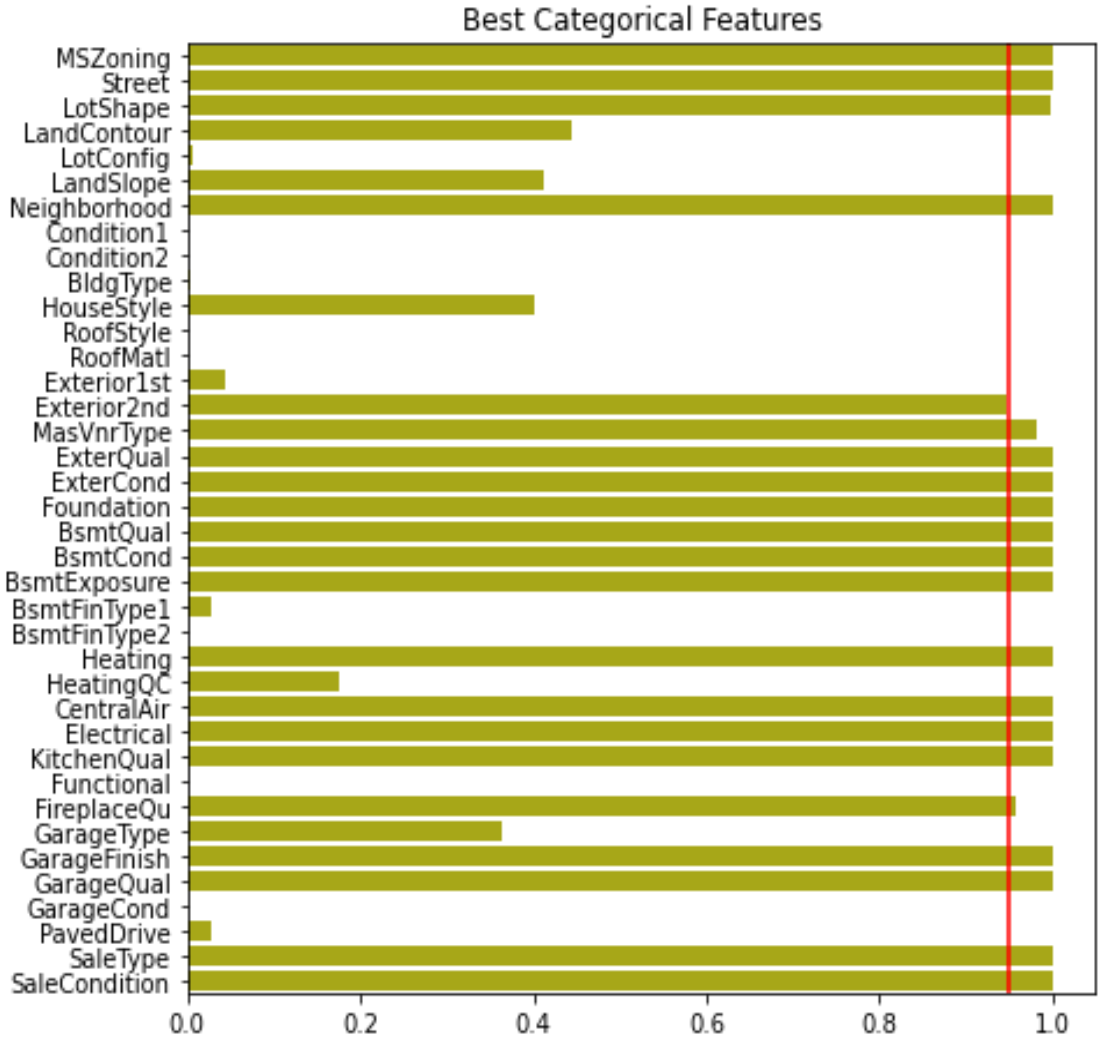
1.Target label Salesprice is highly positively correlated with OverallQual and GrLivArea.

2.Salesprice is negatively correlated with KitchenAbvGr,EnclosedPorch,OverallCond,Mssubclass,Yrsold,LowQualFinsf,Miscval,Bsmithhalfbatch and rest all positvely correlated.

Best Categorical Features using Chi-square test :

Chi Square test is used to identify significant categorical variables i.e. variables with p value less than 5%

```
#Chi Square test for Categorical Columns
from scipy.stats import chi2_contingency
l=[]
for i in obj:
    pvalue  = chi2_contingency(pd.crosstab(df_train['SalePrice'],df_train[i]))[1]
    l.append(1-pvalue)
plt.figure(figsize=(7,7))
sns.barplot(x=l, y=obj, color ='y')
plt.title('Best Categorical Features')
plt.axvline(x=(1-0.05),color='r')
plt.show()
```



Preliminary analysis shows that MsZoning, Street, Lotshape, Neighborhood,MasVnrType,ExterQual,ExterCond,Foundation,BsmtQual, bsmtCond,BsmtExposureHeating,CentralAir,Electrical,KitchenQual,FireplaceQu,GarageFinish,GarageQual,SaleType,SaleCondition seems to be having significant influence on Agent Bonus.

Encoding the categorical object datatype column:

Label encoders transform non-numerical labels into numerical labels. Each category is assigned a unique label starting from 0 and going on till n_categories – 1 per feature.

```
#using Label encoder
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()

for i in df_train.columns:
    if df_train[i].dtypes== "object":
        df_train[i]= enc.fit_transform(df_train[i].values.reshape(-1,1))
```

```
df_train
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType
0	120	3	70.98847	4928	1	0	3	4	0	13	2	2	4
1	20	3	95.00000	15865	1	0	3	4	1	12	2	2	0
2	60	3	92.00000	9920	1	0	3	1	0	15	2	2	0
3	20	3	105.00000	11751	1	0	3	4	0	14	2	2	0
4	20	3	70.98847	16635	1	0	3	2	0	14	2	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	20	3	70.98847	9819	1	0	3	4	0	19	2	2	0
1164	20	3	67.00000	8777	1	3	3	4	0	7	1	2	0
1165	160	3	24.00000	2280	1	3	3	2	0	13	2	2	3
1166	70	0	50.00000	8500	1	3	3	4	0	9	1	2	0
1167	60	3	70.98847	7861	1	0	3	4	0	8	2	2	0

Skewness Analysis

skewness is a measure of asymmetry of the probability distribution about its mean and helps describe the shape of the probability distribution. Basically it measures the level of how much a given distribution is different from a normal distribution (which is symmetric). In other words,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

Distribution on the basis of skewness value:
Skewness = 0: Then normally distributed.
Skewness > 0: Then more weight in the left tail of the distribution,positively skewed .
Skewness < 0: Then more weight in the right tail of the distribution,negative skewed

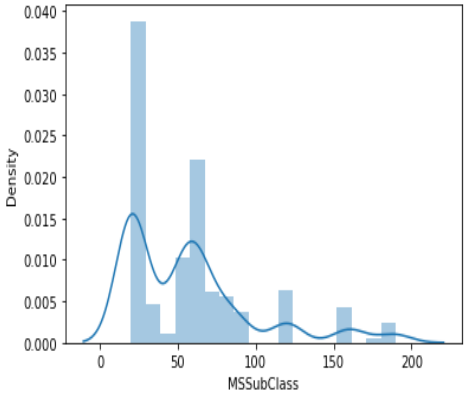
```
df_train.skew()
```

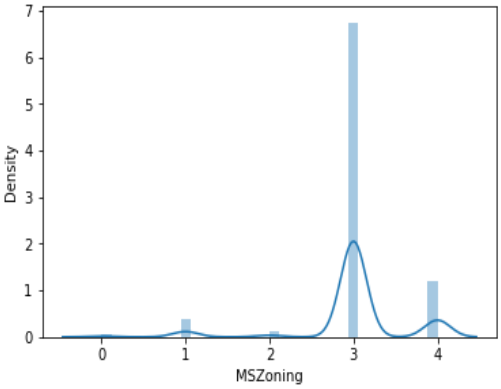
```
MSSubClass       1.422019
MSZoning        -1.796785
LotFrontage      2.710383
LotArea         10.659285
Street         -17.021969
LotShape        -0.603775
LandContour     -3.125982
LotConfig       -1.118821
LandSlope        4.812568
Neighborhood     0.043735
Condition1       3.008289
Condition2      11.514458
BldgType         2.318657
HouseStyle       0.285680
OverallQual      0.175082
OverallCond      0.580714
YearBuilt       -0.579204
YearRemodAdd    -0.495864
RoofStyle        1.498560
RoofMatl         7.577352
Exterior1st     -0.612816
Exterior2nd     -0.592349
MasVnrType      -0.104609
MasVnrArea       2.834658
ExterQual       -1.810843
ExterCond       -2.516219
```

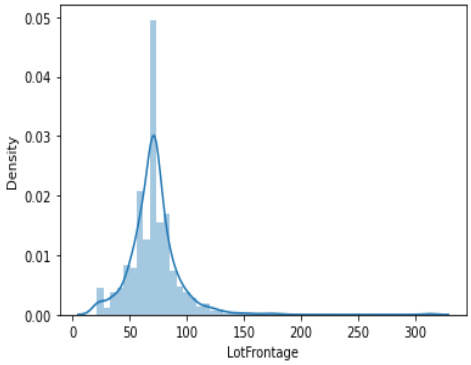
As a general rule of thumb,If skewness is less than -1 or greater than 1, the distribution is highly skewed.

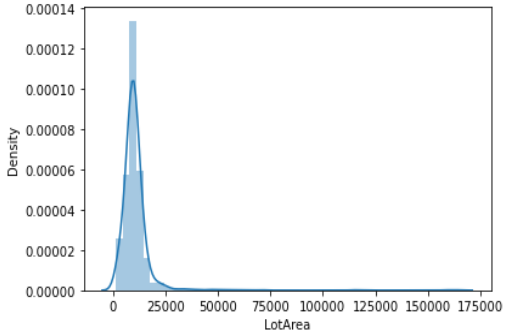
If skewness is between -1 and -0.5 or between 0.5 and 1, the distribution is moderately skewed.
If skewness is between -0.5 and 0.5, the distribution is approximately symmetric.
Considering the threshold as -0.5/+0.5,we can see most of the columns are skewed and skeweness will be removed later.

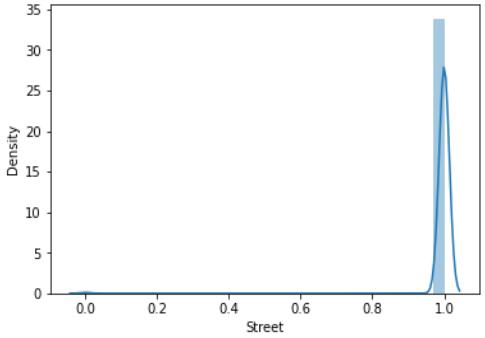
```
#Plotting distplot for checking the distribution of skewness
for col in df_train.describe().columns:
    sns.distplot(df_train[col])
    plt.show()
```

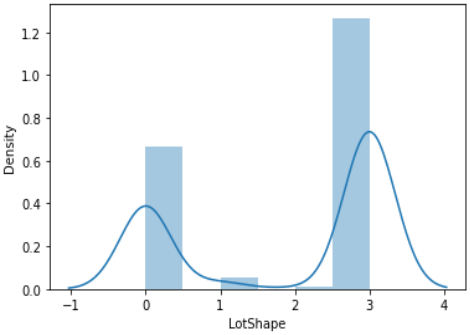

Density
MSSubClass

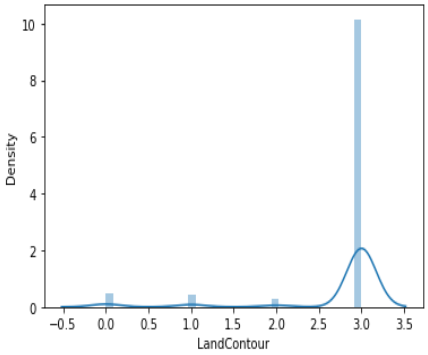

Density
MSZoning

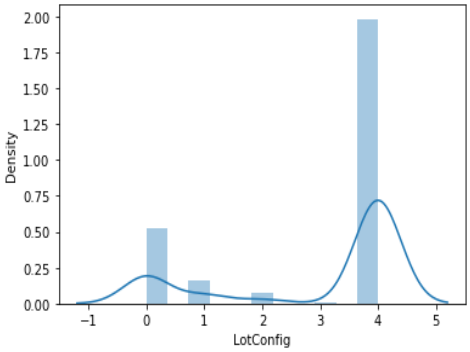

Density
LotFrontage

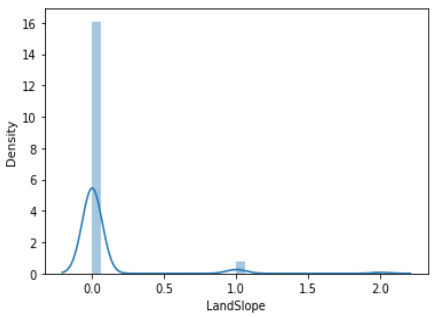

Density
LotArea

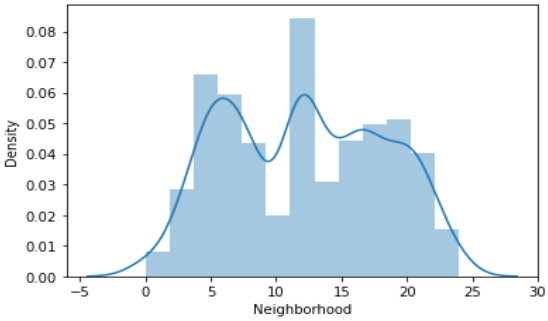

Density
Street


Density
LotShape

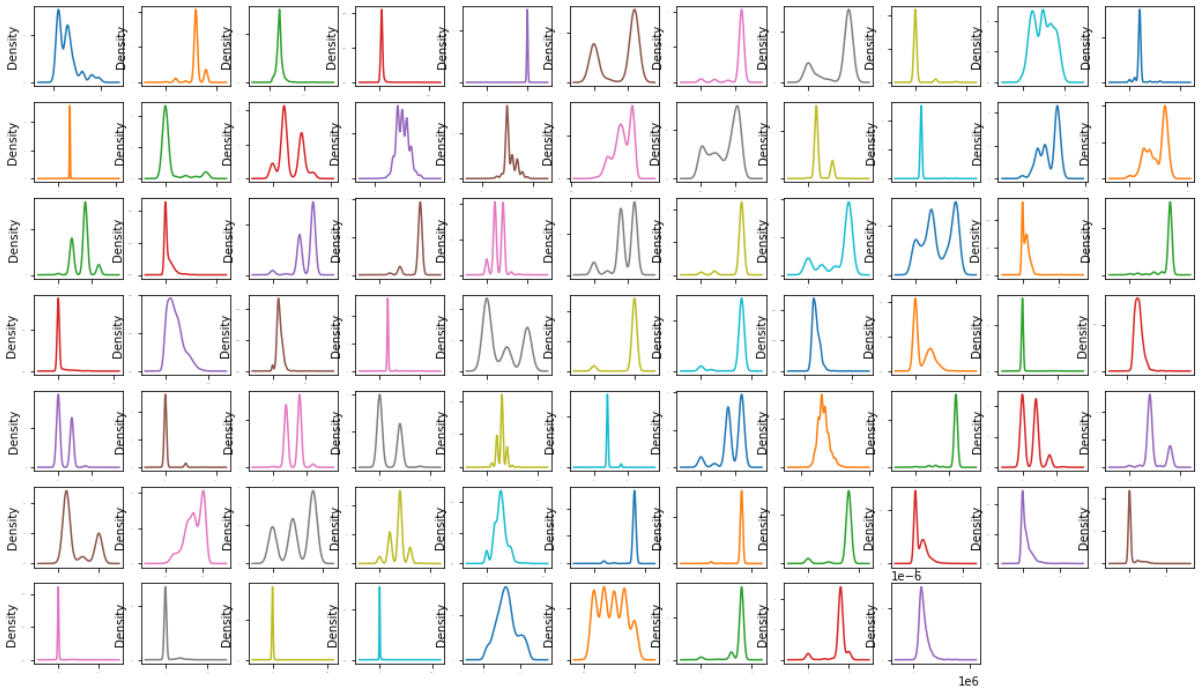

Density
LandContour


Density
LotConfig


Density
LandSlope


Density
Neighborhood

```
df_train.plot(kind="density",subplots=True,layout=(8,11),sharex=False,legend=False,fontsize=1,figsize=(18,12))
plt.show()
```



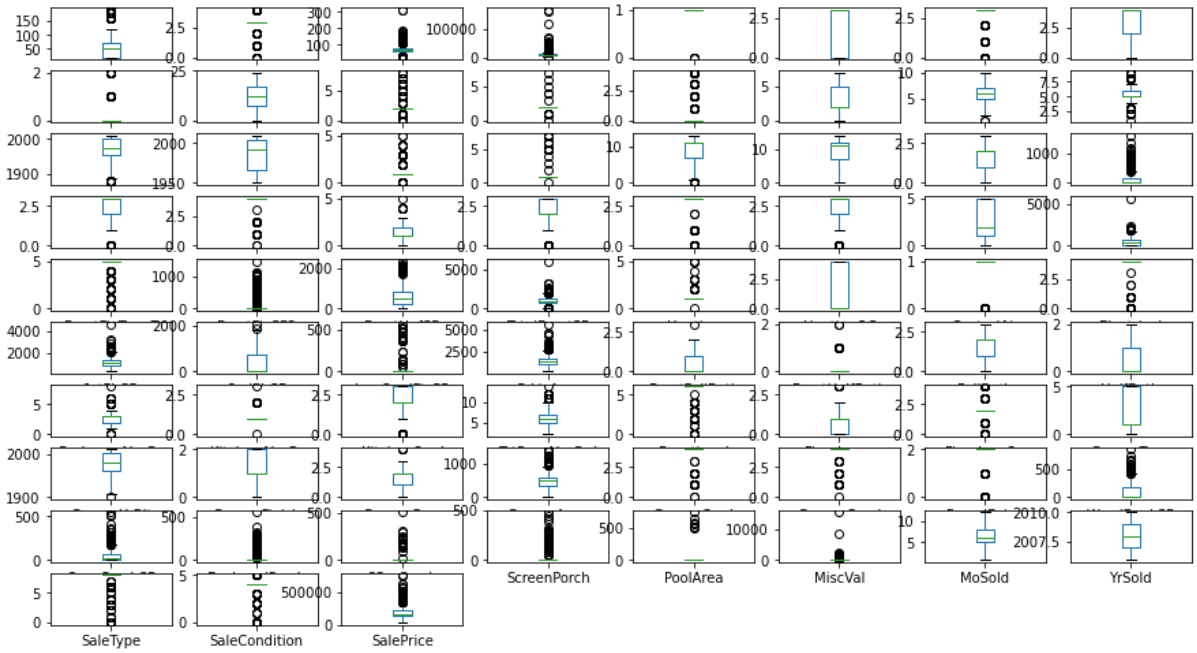
We can see that some of the curves are not normally distributed and it is due to the presence of high skewness and we need to handle them

- Checking outliers

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors.

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be “outliers” using a method that is a function of the inter-quartile range.

```
df_train.plot(kind='box',subplots=True,layout=(12,8),figsize=(15,10))
MSZoning        AxesSubplot(0.223936,0.826831;0.0824468x0.053169)
LotFrontage     AxesSubplot(0.322872,0.826831;0.0824468x0.053169)
LotArea         AxesSubplot(0.421809,0.826831;0.0824468x0.053169)
Street          AxesSubplot(0.520745,0.826831;0.0824468x0.053169)
LotShape        AxesSubplot(0.619681,0.826831;0.0824468x0.053169)
LandContour     AxesSubplot(0.718617,0.826831;0.0824468x0.053169)
LotConfig       AxesSubplot(0.817553,0.826831;0.0824468x0.053169)
LandSlope          AxesSubplot(0.125,0.763028;0.0824468x0.053169)
Neighborhood    AxesSubplot(0.223936,0.763028;0.0824468x0.053169)
Condition1      AxesSubplot(0.322872,0.763028;0.0824468x0.053169)
Condition2      AxesSubplot(0.421809,0.763028;0.0824468x0.053169)
BldgType        AxesSubplot(0.520745,0.763028;0.0824468x0.053169)
HouseStyle      AxesSubplot(0.619681,0.763028;0.0824468x0.053169)
OverallQual     AxesSubplot(0.718617,0.763028;0.0824468x0.053169)
OverallCond     AxesSubplot(0.817553,0.763028;0.0824468x0.053169)
YearBuilt          AxesSubplot(0.125,0.699225;0.0824468x0.053169)
YearRemodAdd    AxesSubplot(0.223936,0.699225;0.0824468x0.053169)
RoofStyle       AxesSubplot(0.322872,0.699225;0.0824468x0.053169)
```



We can see that there are outliers in the dataset and we can handle them by using z-score method

Handling outliers by using z-score method

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. Z-score is measured in terms of standard deviations from the mean. In most cases a threshold of 3 or - 3 is used i.e., if

the Z-score value is higher than or less than 3 or -3 respectively, that particular data point will be identified as outlier.

```
#removal of outliers
from scipy.stats import zscore
z = np.abs(zscore(df_train))
threshold = 6
dfnew = df_train[(z<6).all(axis = 1)]

print ("Dataframe Shape before removing outliers: ", df_train.shape)
print ("Dataframe Shape after removing outliers: ", dfnew.shape)
print ("Percentage of data loss after outlier removal: ", (df_train.shape[0]-dfnew.shape[0])/df_train.shape[0]*100)

df_train=dfnew.copy() # reassigning to our original dataframe name
```

```
Dataframe Shape before removing outliers:  (1168, 75)
Dataframe Shape after removing outliers:  (1068, 75)
Percentage of data loss after outlier removal:  8.561643835616438
```

Thus using threshold as 3 ,the data loss was huge that is 59%, so we will now keep the threshold limit at 6 and found that data loss percentage is around 8.56% which is acceptable.

Separating x and y as independent and dependent variable:

```
df_x=df_train.drop(columns=['SalePrice'])
y=df_train['SalePrice']
```

```
#Checking x data
df_x.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	Hc
0	120	3	70.98847	4928	1	0	3	4	0	13	2	2	4	
2	60	3	92.00000	9920	1	0	3	1	0	15	2	2	0	
3	20	3	105.00000	11751	1	0	3	4	0	14	2	2	0	
4	20	3	70.98847	16635	1	0	3	2	0	14	2	2	0	
5	60	3	58.00000	14054	1	0	3	4	0	8	2	2	0	

```
df_x.shape
```

```
(1068, 74)
```

```
#Checking y data after splitting
y.head()
```

```
0    128000
2    269790
3    190000
4    215000
5    219210
Name: SalePrice, dtype: int64
```

Treating skewness using Power transform

Power transform :Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like. The method i have used here is yeo-johnson as it works with both positive and negative values.

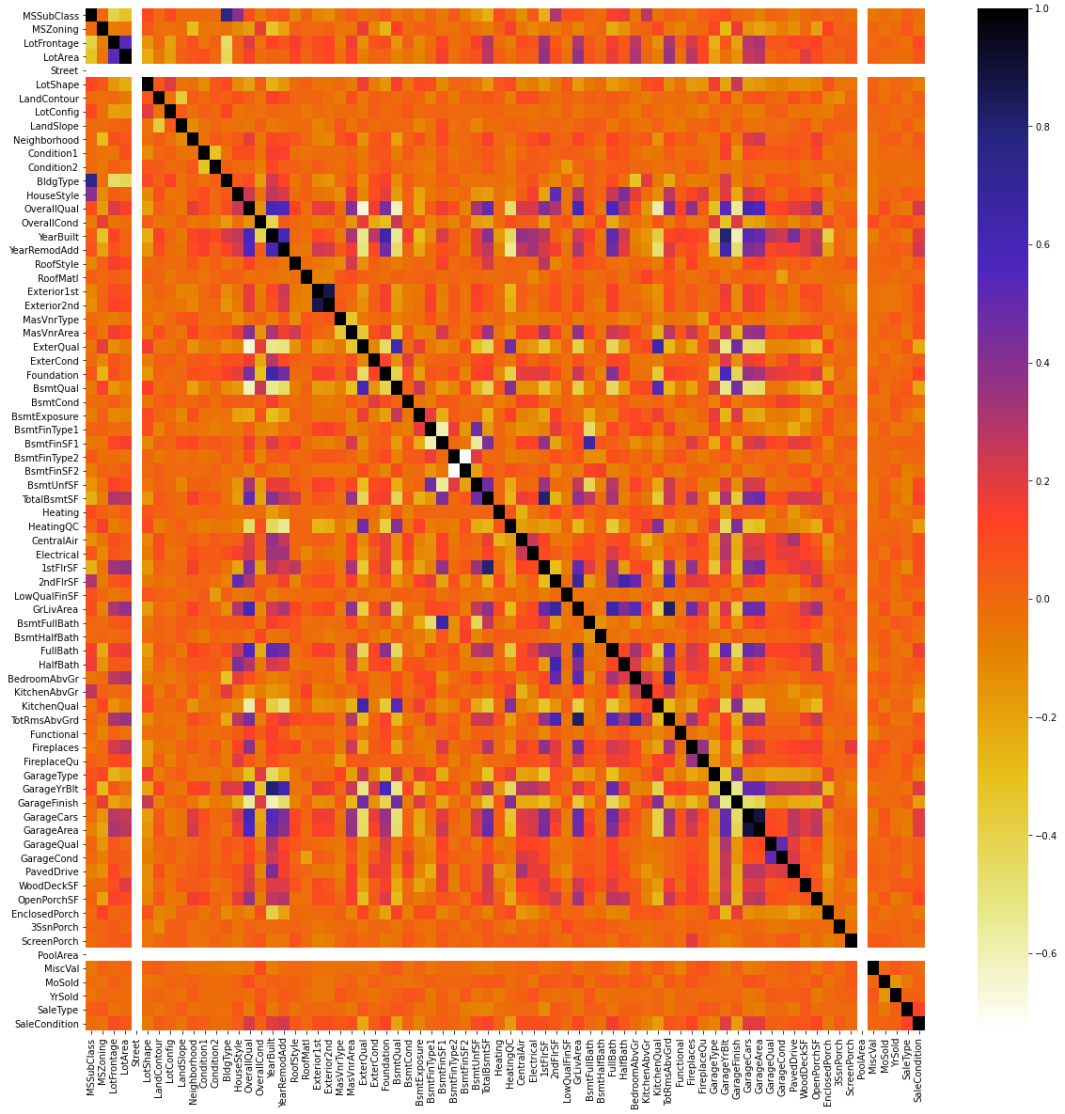
```
#using power transform it can work uoon both negative and positive skewed value
from sklearn.preprocessing import power_transform
df_x=power_transform(df_x,method='yeo-johnson')
df_x
```

```
array([[ 1.36219379, -0.15815015,  0.08669765, ..., -0.60796281,
         0.41408692,  0.0036483 ],
       [ 0.48722854, -0.15815015,  1.10613347, ..., -0.60796281,
         0.41408692,  0.0036483 ],
       [-1.16369229, -0.15815015,  1.70345337, ...,  1.63600961,
        -2.76356785,  0.0036483 ],
       ...,
       [ 1.69018156, -0.15815015, -2.60728681, ...,  0.88937435,
         0.41408692,  0.0036483 ],
       [ 0.69257825, -2.95461199, -1.02191951, ...,  0.14143777,
         0.41408692,  0.0036483 ],
       [ 0.48722854, -0.15815015,  0.08669765, ..., -1.35874604,
         0.41408692,  0.0036483 ]])
```

Feature Selection with Pearson Correlation

It will remove the features which are highly correlated and thereby helps in reducing multicollinearity problem.

```
#import essential libraries
import matplotlib.pyplot as plt
import seaborn as sns

#Using Pearson Correlation
corrmat = df_x.corr()
fig, ax = plt.subplots()
fig.set_size_inches(20,20)
sns.heatmap(corrmat,cmap="CMRmap_r")
```




```
# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

```
corr_features = correlation(df_x, 0.6) #setting the threshold values as 60%
len(set(corr_features))
```

```
17
```

```
corr_features
```

```
{'1stFlrSF',
 'BldgType',
 'BsmtFinSF2',
 'BsmtFullBath',
 'BsmtQual',
 'ExterQual',
 'Exterior2nd',
 'Foundation',
 'FullBath',
 'GarageArea',
 'GarageCars',
 'GarageFinish',
 'GarageYrBlt',
 'GrLivArea',
 'HalfBath',
 'KitchenQual',
 'TotRmsAbvGrd'}
```

Thus we see that 17 features are highly correlated

```
df_x.drop(['1stFlrSF',
 'BldgType',
 'BsmtFinSF2',
 'BsmtFullBath',
 'BsmtQual',
 'ExterQual',
 'Exterior2nd',
 'Foundation',
 'FullBath',
 'GarageArea',
 'GarageCars',
 'GarageFinish',
 'GarageYrBlt',
 'GrLivArea',
 'HalfBath',
 'KitchenQual',
 'TotRmsAbvGrd'],axis=1,inplace=True)
```

Observation: Thus correlated features have been removed using pearsons correlation technique setting the threshold as 60%.Now our features have reduced from 74 to 57 features.

Selecting Top Feature using Mutual Information for regression problem

Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

Mutual Information: It measures the amount of information one can obtain from one random variable given another. In other words, Information gain or mutual information measures how much information the presence/absence of a feature contributes to making the correct prediction on the target.

When having a big dataset with a big range of features, mutual information can help to select a subset of those features in order to discard the irrelevant ones.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df_x,y,test_size=0.3,random_state=0)
```

```
from sklearn.feature_selection import mutual_info_regression
#determine the mutual information
mutual_info=mutual_info_regression(x_train,y_train)
mutual_info
```

```
array([0.28238075, 0.07756537, 0.20564033, 0.10219909, 0.        ,
       0.09007086, 0.        , 0.02725412, 0.00199093, 0.51858399,
       0.04562502, 0.        , 0.11974973, 0.55933331, 0.10671893,
       0.37732706, 0.25727176, 0.02823151, 0.0037271 , 0.20959597,
       0.11255193, 0.11352154, 0.00396225, 0.02958767, 0.03690719,
       0.19350504, 0.13947299, 0.0230353 , 0.13864705, 0.32721927,
       0.00411637, 0.18290025, 0.06560626, 0.06175031, 0.24721473,
       0.        , 0.01289667, 0.06428619, 0.02851904, 0.01888944,
       0.17558996, 0.10855095, 0.11487068, 0.0021085 , 0.        ,
       0.05099407, 0.0623971 , 0.14026695, 0.02355047, 0.        ,
       0.00837846, 0.00210108, 0.        , 0.0168263 , 0.        ,
       0.06985207, 0.06787224])
```

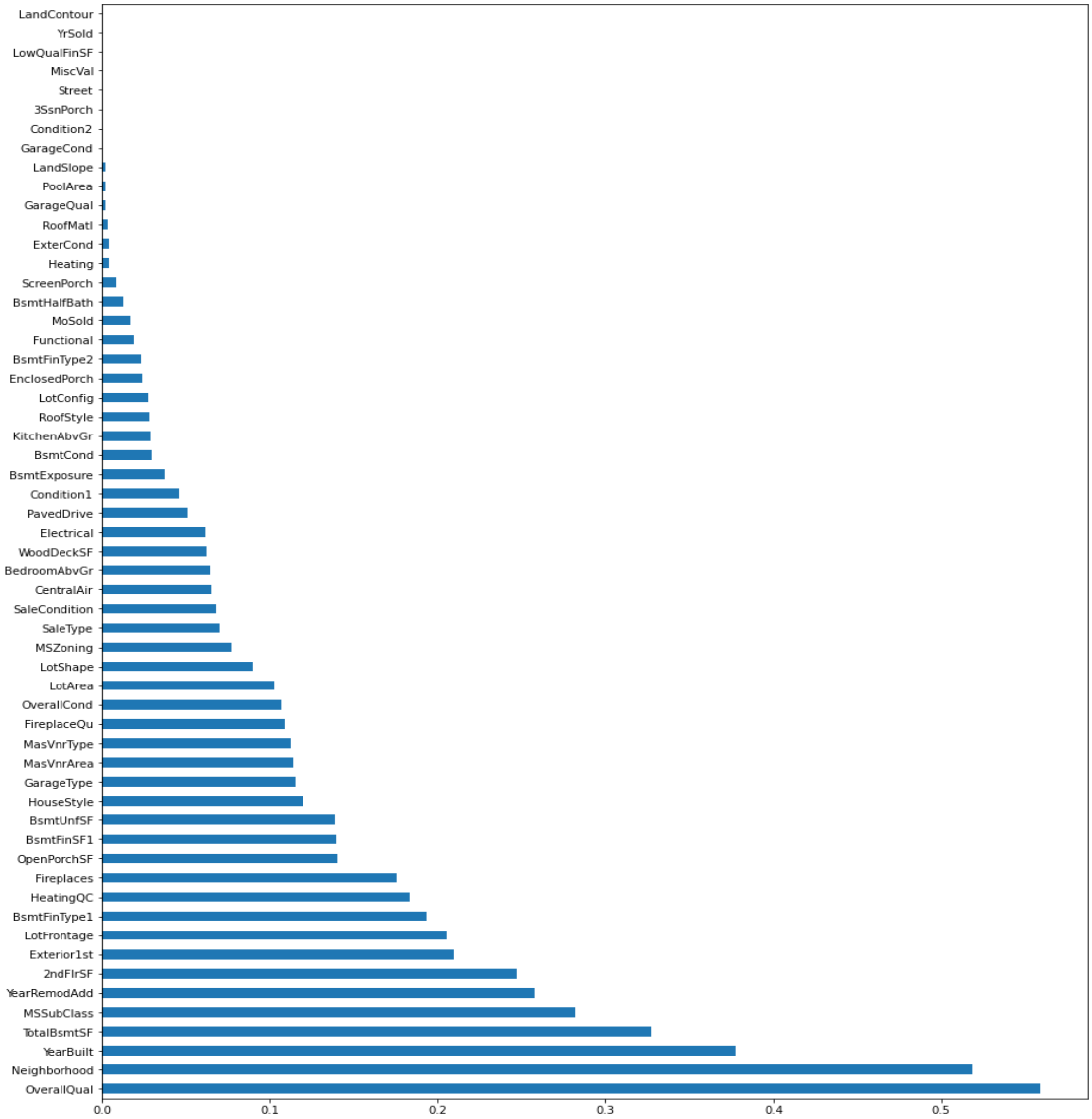
```
mutual_info=pd.Series(mutual_info)
```

```
mutual_info=pd.Series(mutual_info)
mutual_info.index=x_train.columns
mutual_info.sort_values(ascending=False)
```

```
OverallQual      0.559333
Neighborhood     0.518584
YearBuilt        0.377327
TotalBsmtSF      0.327219
MSSubClass       0.282381
YearRemodAdd     0.257272
2ndFlrSF         0.247215
Exterior1st      0.209596
LotFrontage      0.205640
BsmtFinType1     0.193505
HeatingQC        0.182900
Fireplaces       0.175590
OpenPorchSF      0.140267
BsmtFinSF1       0.139473
BsmtUnfSF        0.138647
HouseStyle       0.119750
GarageType       0.114871
MasVnrArea       0.113522
MasVnrType       0.112552
FireplaceQu      0.108551
OverallCond      0.106719
LotArea          0.102199
LotShape         0.090071
MSZoning         0.077565
SaleType         0.069852
SaleCondition    0.067872
CentralAir       0.065606
```

```
mutual_info.sort_values(ascending=False).plot.barh(figsize=(15,20))
```

```
<AxesSubplot:>
```



Observation: From the graph, we can infer that the OverallQual is having the highest mutual information gain(0.57) then Neighborhood(0.52) followed by GrLivArea(0.50), and so on. So OverallQual give 57% of the information about the target variable SalePrice in this case.

Selecting Top columns using SelectPercentile

```
from sklearn.feature_selection import SelectPercentile
selected_top_columns = SelectPercentile(mutual_info_regression, percentile=20)
selected_top_columns.fit(x_train, y_train)
selected_top_columns.get_support()
```

```
array([ True, False,  True, False, False, False, False, False, False,
        True, False, False, False,  True, False,  True,  True, False,
       False,  True, False, False, False, False, False,  True, False,
       False, False,  True, False,  True, False, False,  True, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False])
```

```
x_train.columns[selected_top_columns.get_support()]
```

```
Index(['MSSubClass', 'LotFrontage', 'Neighborhood', 'OverallQual', 'YearBuilt', 'YearRemodAdd', 'Exterior1st', 'BsmtFinType1',
'TotalBsmtSF', 'HeatingQC', '2ndFlrSF', 'Fireplaces'], dtype='object')
```

So these are the features in the top 20th percentile, which means that after Fireplaces the remaining features which are 80% have a dependence of at least less than that of Fireplaces.

Feature Scaling:

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.It is doen by Standardization or Normalization. In normalization,using MinMaxScaler which scales down the values of the features between 0 to 1

In standardaization, using StandardScaler(Z-score normalization),here all the features will be transformed in such a way that it will have properties of standard normal distribution mean =0 and standard deviation =1.

I have used here standardScaler to scaled the data values .

```
#Scaling the dataset using StandardScaler
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x= pd.DataFrame(ss.fit_transform(df_x))
```

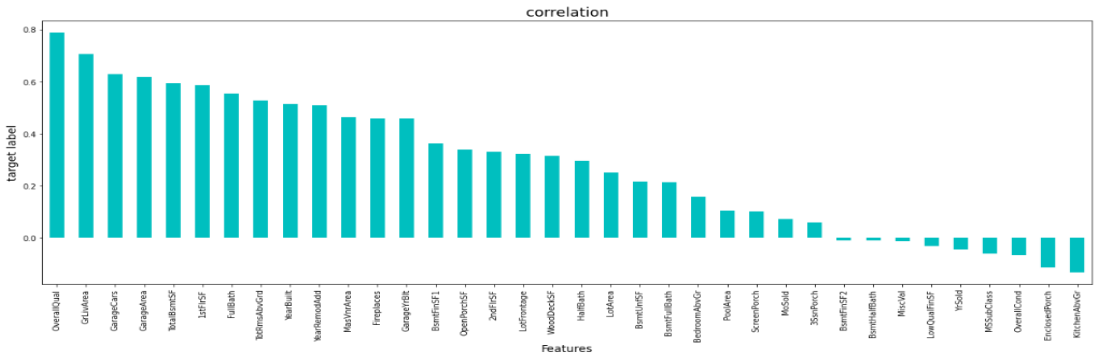
```
x
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.362194	-0.158150	0.086698	-1.267808	0.0	-1.400229	0.312444	0.593732	-0.202326	0.202843	0.058293	0.061314	-0.482468	-0.047826	-0.491382
1	0.487229	-0.158150	1.106133	0.173225	0.0	-1.400229	0.312444	-1.567712	-0.202326	0.517628	0.058293	0.061314	1.011590	0.685687	-0.491382
2	-1.163692	-0.158150	1.703453	0.596679	0.0	-1.400229	0.312444	0.593732	-0.202326	0.361518	0.058293	0.061314	-0.482468	-0.047826	0.437022
3	-1.163692	-0.158150	0.086698	1.574774	0.0	-1.400229	0.312444	-1.129051	-0.202326	0.361518	0.058293	0.061314	-0.482468	-0.047826	1.256657
4	0.487229	-0.158150	-0.586275	1.080995	0.0	-1.400229	0.312444	0.593732	-0.202326	-0.638710	0.058293	0.061314	1.011590	0.685687	-0.491382
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1063	-1.163692	-0.158150	0.086698	0.148672	0.0	-1.400229	0.312444	0.593732	-0.202326	1.120032	0.058293	0.061314	-0.482468	-0.812564	-0.491382
1064	-1.163692	-0.158150	-0.115948	-0.113127	0.0	0.733261	0.312444	0.593732	-0.202326	-0.819502	-1.496508	0.061314	-0.482468	-1.615471	-0.491382
1065	1.690182	-0.158150	-2.607287	-2.405066	0.0	0.733261	0.312444	-1.129051	-0.202326	0.202843	0.058293	0.061314	1.011590	-0.047826	0.437022
1066	0.692578	-2.954612	-1.021920	-0.185541	0.0	0.733261	0.312444	0.593732	-0.202326	-0.462767	-1.496508	0.061314	1.011590	-1.615471	-1.565335
1067	0.487229	-0.158150	0.086698	-0.357654	0.0	-1.400229	0.312444	0.593732	-0.202326	-0.638710	0.058293	0.061314	1.011590	-0.047826	-0.491382

1068 rows × 57 columns

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

  After loading of the training and testing dataset ,we check their datatypes ,well,it was consists of int,float and object datatype.Then we see that the our goal is to predict salesprice of the houses on the bases of which housing company will be able to make decision to investment or not,whether their investment is feasible or not.Also they want to see that what are most features affecting the salesprice and how they are related to sales price. To check this, we carry our correlation of features with target variable which was done in pre-processing stage above. We found that using mutual information gain technique, OverallQual has the highest mutual information gain(0.57) then Neighborhood(0.52) followed by GrLivArea(0.50), and so on. So OverallQual give 57% of the information about the target variable SalePrice . Graph showing correlation of target variable saleprice with the input features which has been also explained in pre-dataprocessing



And also Salesprice is negatively correlated with KitchenAbvGr,EnclosedPorch,OverallCond,Mssubclass,Yrsold,LowQualFinsf,Miscval,Bsmithhalfbatch and rest all positvely correlated.

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

1.Most of the Z_score problem handling outlier, we normally take thresholds as +/-3 but here I have taken threshold as 6 since data loss was 8% while taking 3, it was showing data loss more than 50%.

2. I have set the percentile as 20 using selectPercentile as I want to see top features which are under this percentile.

Hardware and Software Requirements and Tools Used

Hardware Used:

i. RAM: 8 GB
ii. CPU: Intel® Core™ i3-1005G1 CPU @1.20GHz
iii. GPU: Intel® UHD Graphics

Software Used:

i. Programming language: Python
ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

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. Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

The objective of the case study is to predict the SalePrice of the house which is as per the dataset is continuous is nature so this will be classified as Regression problem. As this is a regression problem, so we will be loading all the libraries related to it which is as below:

Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models.

Feature extraction − It is used to extract the features from data to define the attributes in image and text data.

Feature selection − It is used to identify useful attributes to create supervised models

So we will be using sklearn for regression Analysis :

Model Used & Description

1.Sklearn:Linear regression: It is one of the best statistical models that studies the relationship between a dependent variable (Y) with a given set of independent variables (X). The relationship can be established with the help of fitting a best line.

sklearn.linear_model.LinearRegression is the module used to implement linear regression.

2. LASSO (Least Absolute Shrinkage and Selection Operator)

LASSO is the regularisation technique that performs L1 regularisation. It modifies the loss function by adding the penalty (shrinkage quantity) equivalent to the summation of the absolute value of coefficients.

sklearn.linear_model.Lasso is a linear model, with an added regularisation term, used to estimate sparse coefficients.

3. Ridge regression or Tikhonov regularization is the regularization technique that performs L2 regularization. It modifies the loss function by adding the penalty (shrinkage quantity) equivalent to the square of the magnitude of coefficients.

sklearn.linear_model.Ridge is the module used to solve a regression model where loss function is the linear least squares function and regularization is L2.

4.Regression with decision trees(Sklearn.tree.DecisionTreeRegressor): Sklearn Module − The Scikit-learn library provides the module name DecisionTreeRegressor for applying decision trees on regression problems. Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

5. 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.Two families of ensemble methods are usually distinguished:

In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.Examples: Bagging methods, Forests of randomized trees.

By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.Examples: AdaBoost, Gradient Tree Boosting.

The different types of ensemble techniques used in the model are:

i).Random Forest Regressor: :It is an ensemble technique which works on the principle of bagging.It builds decision tree on different samples and final output decided upon majority voting for classification and averaging for regression.It takes care of overfitting issues.

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. Thus the loss gradient is minimized as the model is fit as the term gradient boosting signifies.

6.**XgBoost**: Extreme Gradient Boosting, or XGBoost for short, is an efficient open-source implementation of the gradient boosting algorithm. The two main reasons to use XGBoost are execution speed and model performance. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

7.sklearn.neighbors: Regression based on k-nearest neighbors.The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors.KNeighborsRegressor implements learning based on the nearest neighbors of each query point, where is an integer value specified by the user.

8.Important sklearn.metrics modules used in the project are:

1. **Root_mean_squared_error:** Root mean squared error (RMSE) is the square root of the mean of the square of all of the error. The use of RMSE is very common, and it is considered an excellent general purpose error metric for numerical predictions. RMSE is a good measure of accuracy, but only to compare forecasting errors of different models or model configurations for a particular variable and not between variables, as it is scale-dependent.

2. **r2_score**: It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. **MAPE:** Mean Absolute Percentage Error (MAPE) is a statistical measure to define the accuracy of a machine learning algorithm on a particular dataset. MAPE can be considered as a loss function to define the error termed by the model evaluation. Using MAPE, we can estimate the accuracy in terms of the differences in the actual v/s estimated values. MAPE can also be expressed in terms of percentage. Lower the MAPE, better fit is the model.

9. sklearn.model_selection : Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.It offers a lot of functionalities related to model selection and validation, including the following:

1.Cross-validation: Cross-validation is a technique used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

2.Hyperparameter tuning: Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning. We must select from a specific list of hyperparameters for a given model as it varies from model to model.

Hyper parameter optimisation in machine learning intends to find the hyper parameters of a given machine learning algorithm that deliver the best performance as measured on a validation set.

Using Scikit-Learn's GridSearchCV method, we can explicitly specify every combination of settings to try. We do this with GridSearchCV, a method that, instead of sampling randomly from a distribution, evaluates all combinations we define.

3.Train_Test_Split : Split arrays or matrices into random train and test subsets.

- ## Testing of Identified Approaches (Algorithms)

The algorithms used on training and test data are as follows:

A. Linear Regression Model
B. Ridge Regularization Regression Model
C. Lasso Regularization Regression Model
D. Decision Tree Regression Model
E. Random Forest Regression Model
F. Gradient Boosting Regression Model
G. Ada Boost Regression Model
H. XGBoost Regression Model
I. KNearest neighbors Model
J. Support Vector Regressor

- ## Run and Evaluate selected models

We have used here total of 10 Regression Models after choosing the random state amongst 1-100 number with test_train_split choosing 33% as test_size.We found random_state=22 where the accuracy was highest. Then defining a function for getting the regression model trained and evaluated. The code for the models is listed below.

Building the model

```
2]: #Importing required metrices and model for the dataset
    from sklearn.model_selection import train_test_split
```

Finding the Best Random State

```
3]: #Finding the best random state
    max_ran_score=0
    for ran_state in range(1,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=ran_state,test_size=0.33)
        lr=LinearRegression()
        lr.fit(x_train,y_train)
        y_pred=lr.predict(x_test)
        r2_scr=r2_score(y_test,y_pred)
        if r2_scr>max_ran_score:
            max_ran_score=r2_scr
            final_ran_state=ran_state
    print("max r2 score corresponding to",final_ran_state,"is",max_ran_score)
```

```
max r2 score corresponding to 22 is 0.8266426833945135
```

We can see that at random_state=22, the best r2_score is obtained so that we can create our train_test_split with this random state

```
4]: #Creating train_test_split using best random_state
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=22,test_size=.33)
```

Thus we select randon_state=22 according to our best r2_score having test_size at 33%..First we will import the required libraries . Then we will modelling all the models taken by defining function for algorithms and evaluating the metrics in the same.

Modeling without tuning

```
]: #Importing the algorithms and other parameters
   from sklearn.linear_model import LinearRegression
   from sklearn import linear_model
   from sklearn import tree
   from sklearn.ensemble import RandomForestRegressor
   from sklearn.ensemble import AdaBoostRegressor
   from sklearn.neighbors import KNeighborsRegressor
   from sklearn.tree import DecisionTreeRegressor
   from xgboost import XGBRegressor
   from sklearn.svm import SVR
   from sklearn.ensemble import GradientBoostingRegressor
   #importing required metrics
   from sklearn.metrics import mean_squared_error, mean_absolute_error,mean_absolute_percentage_error
   from sklearn.model_selection import cross_val_score
   from sklearn.metrics import r2_score
```

```
lr = LinearRegression()
lsr = linear_model.Lasso(random_state = 22)
rr = linear_model.Ridge(random_state=22)
rfr = RandomForestRegressor(random_state=22)
svr=SVR()
ada= AdaBoostRegressor()
gdb=GradientBoostingRegressor()
dtr = tree.DecisionTreeRegressor(random_state=22)
knnr = KNeighborsRegressor()
xg = XGBRegressor(random_state=22)


models=[lr,lsr,rr,svr,rfr,ada,gdb,dtr,knnr,xg]

rmse_train=[]
rmse_test=[]
scores_train=[]
scores_test=[]
mape=[]
cvs=[]


for i in models:
    i.fit(x_train,y_train)
    r2_train = round(i.score(x_train, y_train),3)
    r2_test = round(i.score(x_test, y_test),3)
    scores_train.append(round(r2_train,3))
    scores_test.append(round(r2_test,3))
    y_pred = i.predict(x_test)
    mape.append(round(mean_absolute_percentage_error(y_test, y_pred)*100,3))
    rmse_train.append(round(np.sqrt(mean_squared_error(y_train,i.predict(x_train))),3))
    rmse_test.append(round(np.sqrt(mean_squared_error(y_test,i.predict(x_test))),3))
```

```
print(pd.DataFrame({'Train RMSE': rmse_train,'Test RMSE': rmse_test,'Train R2': scores_train,'Test R2': scores_test, 'MAPE':mape}
           index=['Linear Regression','Lasso Regression','SVR Regression','AdaBoost Regression','GradientBoosting Regression','F
```

	Train RMSE	Test RMSE	Train R2	Test R2	MAPE
Linear Regression	30517.395	30426.378	0.828	0.827	13.790
Lasso Regression	30517.296	30422.160	0.828	0.827	13.786
SVR Regression	30517.713	30423.187	0.828	0.827	13.768
AdaBoost Regression	75257.252	75224.208	-0.047	-0.060	29.753
GradientBoosting Regression	11181.359	25436.678	0.977	0.879	9.841
Ridge Regression	25447.777	28149.318	0.880	0.852	12.534
Random Forest Regression	13130.617	23532.900	0.968	0.896	9.099
Decision Tree Regression	0.000	44777.956	1.000	0.625	14.978
KNN Regression	31972.840	37793.057	0.811	0.733	14.708
XGB Regression	372.932	29526.744	1.000	0.837	10.646

Observation: For Linear Models, the MAPE(Mean absolute percentage error) states that the predicted values are ~13% away from actuals.

For Non-Linear Models / ensemble models, Decision Tree models overfits. AdaBoost model performs very poor whereas Random forest model and Gradient Boosting Regression are performing the best among non-linear models, with approx lowest RMSE score and MAPE as 9.09% and 9.84% respectively .Comparing all the models,choosing Random Forest

Regressor and Gradient Boosting Regression model for hyperparameter tunning to optimse the model.

RMSE indicates the absolute fit of the model or spread out of residual. R2 indicates proportion of variance for a dependent variable that is explained by independent variables. Both RMSE and R2 are relative measure of fit. MAPE indicates to what extent predicated values are away from actual.

- ## Key Metrics for success in solving problem under consideration

The key metrics used here were r2_score, cross_val_score, RMSE and MAPE. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning through GridSearchCV method.

1.Root_mean_squared_error: Root mean squared error (RMSE) is the square root of the mean of the square of all of the error. The use of RMSE is very common, and it is considered an excellent general purpose error metric for numerical predictions. RMSE is a good measure of accuracy, but only to compare forecasting errors of different models or model configurations for a particular variable and not between variables, as it is scale-dependent.

2.r2_score: It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3.MAPE: Mean Absolute Percentage Error (MAPE) is a statistical measure to define the accuracy of a machine learning algorithm on a particular dataset. MAPE can be considered as a loss function to define the error termed by the model evaluation. Using MAPE, we can estimate the accuracy in terms of the differences in the actual v/s estimated values. MAPE can also be expressed in terms of percentage. Lower the MAPE, better fit is the model.

4. Cross-validation is a technique used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

5.Hyperparameter Tuning: Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning. We must select from a specific list of hyperparameters for a given model as it varies from model to model.

Hyper parameter optimisation in machine learning intends to find the hyper parameters of a given machine learning algorithm that deliver the best performance as measured on a validation set.

Using Scikit-Learn's GridSearchCV method, we can explicitly specify every combination of settings to try. We do this with GridSearchCV, a method that, instead of sampling randomly from a distribution, evaluates all combinations we define.

Cross validation of all the models:

```
#Random Forest Regressor  Model
from sklearn.model_selection import cross_val_score
rfscore=cross_val_score(rfr,x,y,cv=5)
rfc=rfscore.mean()
print('Cross Val Score:',rfc*100)
```

```
Cross Val Score: 85.41577616786012
```

```
#Decission Tree Regressor Model
dtscore=cross_val_score(dtr,x,y,cv=5)
dtc=dtscore.mean()
print('Cross Val Score:',dtc*100)
```

```
Cross Val Score: 70.49125245939842
```

```
#AdaBoost Regressor Model
adscore=cross_val_score(ada,x,y,cv=5)
adc=adscore.mean()
print('Cross Val Score:',adc*100)
```

```
Cross Val Score: 79.69716806071985
```

```
#Gradient Boosting Regressor Model
gbscore=cross_val_score(gdb,x,y,cv=5)
gbc=gbscore.mean()
print('Cross Val Score:',gbc*100)
```

```
Cross Val Score: 87.68988888738522
```

```
#Support Vector Regressor Model
svscore=cross_val_score(svr,x,y,cv=5)
svc=svscore.mean()
print('Cross Val Score:',svc*100)
```

```
Cross Val Score: -6.552844051027633
```

```
lassoscore=cross_val_score(lsr,x,y,cv=5)
lsc=lassoscore.mean()
print('Cross Val Score:',lsc*100)
```

```
Cross Val Score: 80.26462894486818
```

```
#Ridge Model
ridgescore=cross_val_score(rr,x,y,cv=5)
rdc=ridgescore.mean()
print('Cross Val Score:',rdc*100)
```

```
Cross Val Score: 80.27837095147481
```

```
#Knn Regressor Model
knnscore=cross_val_score(knnr,x,y,cv=5)
knnc=knnscore.mean()
print('Cross Val Score:',knnc*100)
```

```
Cross Val Score: 71.63881724895063
```

```
#XGBoost Model
xgbscore=cross_val_score(xg,x,y,cv=5)
xgbc=xgbscore.mean()
print('Cross Val Score:',xgbc*100)
```

```
Cross Val Score: 83.53848280311202
```

Observation: Thus we see that after doing Cross validation score ,the score of all the models were either almost same or have been reduced which means that cross validation is not decreasing the accuracy, it is rather giving us a better approximation for that accuracy, including less overfitting.On the basis of different evaluation metrics and cross validation score ,we choose Random Forest Regressor and Gradient Boosting Regressor for our hyperparameter tunning through GridsearchCV for optimization and boosting the accuracy.

HYPERPARAMETER TUNING THROUGH GRIDSEARCHCV

So now we are going to fine-tune the hyperparameters using GridSearchCV on Random Forest. GridSearchCV automatically tunes the hyperparameters with the parameters specified to find the best parameters and the best estimator,this helps us from manually having to tune, which would take a lot of time.

```
: from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
parameters = {
    #'criterion':['squared_error','absolute_error'],
    'max_depth':[2,5,10,20],
    'min_samples_leaf':[1,2,4],
    'n_estimators':[100,150],
    'min_samples_split':[2,5,10],
    #'bootstrap':[True,False]
    }
rfr=RandomForestRegressor()
clf=GridSearchCV(rfr,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
```

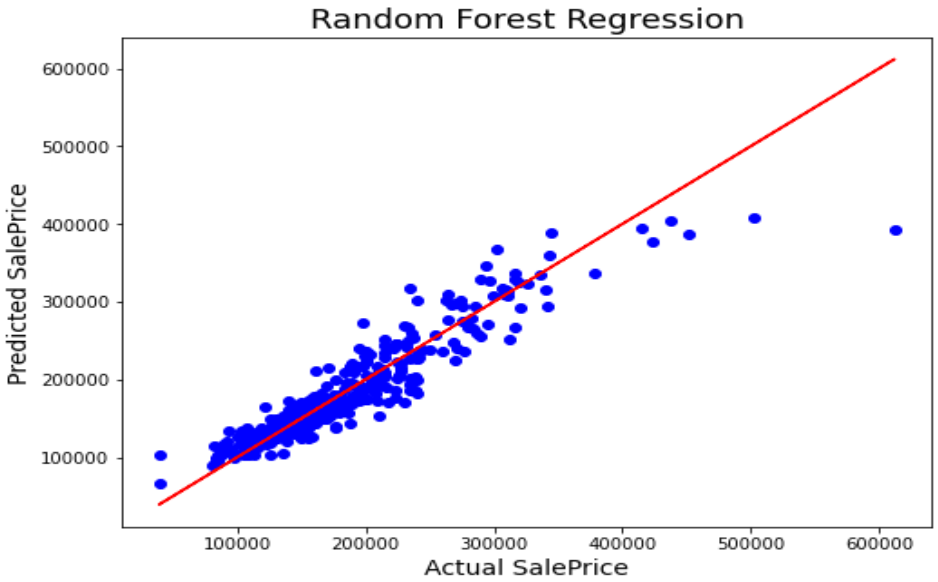
```
]: rfr=RandomForestRegressor(max_depth=10,min_samples_leaf=2,min_samples_split=2,n_estimators=100)
rfr.fit(x_train,y_train)
rfr.score(x_train,y_train)
pred_decision=rf.predict(x_test)

rfs=r2_score(y_test,pred_decision)
print('R2 Score',rfs*100)

rfscore=cross_val_score(rfr,x,y,cv=5)
rfc=rfscore.mean()
print('Cross Val Score:',rfc*100)
```

```
R2 Score 87.93604832589213
Cross Val Score: 85.0852996113281
```

Thus we see after hypertunning the model,getting r2 score as 87.93% and cross validation score as 85.08% which is quite good.

Best Fit Line Graph

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_decision,color='b')
plt.plot(y_test,y_test,color='r')
plt.xlabel("Actual SalePrice",fontsize=14)
plt.ylabel("Predicted SalePrice",fontsize=14)
plt.title("Random Forest Regression",fontsize=18)
plt.show()
```



Thus we can see the best fit line covering the datapoints and it is covering most of the points.There are some datapoint far away from best fit line,which can increase the mse error so we can apply more techniques and regularise and improve the r2_score.

Gradient Boosting Regression:

```
from sklearn.ensemble import GradientBoostingRegressor
parameters= { 'loss':['squared_error','absolute_error'],
              'learning_rate':[0.1,0.01],
              'n_estimators':[2,5,10,20],
              'criterion':['mae','mse'],
            }

gdb=GradientBoostingRegressor()
clf=GridSearchCV(gdb,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'criterion': 'mse', 'learning_rate': 0.1, 'loss': 'squared_error', 'n_estimators': 20}
```

```
gdb=GradientBoostingRegressor(learning_rate= 0.1, loss= 'squared_error',n_estimators= 20, criterion= 'mse')
gdb.fit(x_train,y_train)
gdb.score(x_train,y_train)
pred_decision=gdb.predict(x_test)

gbs=r2_score(y_test,pred_decision)
print('R2 Score',gbs*100)

gbscore=cross_val_score(gdb,x,y,cv=5)
gbc=gbscore.mean()
print('Cross Val Score:',gbc*100)
```

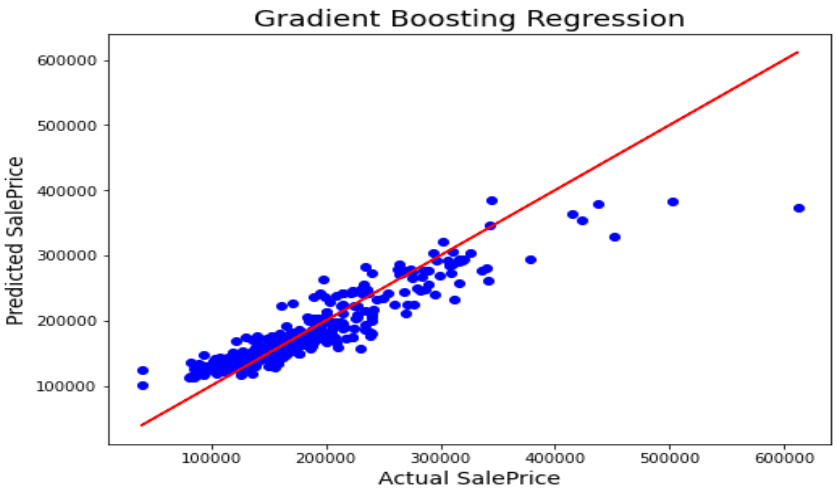
```
R2 Score 83.13672337870015
Cross Val Score: 80.48141472037122
```

Thus we see the R2 score is 83.13% and cross val score is 80% .

.

Best Fit Line Curve

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_decision,color='b')
plt.plot(y_test,y_test,color='r')
plt.xlabel("Actual SalePrice",fontsize=14)
plt.ylabel("Predicted SalePrice",fontsize=14)
plt.title("Random Forest Regression",fontsize=18)
plt.show()
```



Model Analysis using Hyperparameter:

Thus After applying hyperparameter search through GridSearchcv we find that r2_score accuracy and cross validation score of Random Forest is performing better than gradient boosting which is 87.47% and 85.07 % for random forest and 83.13 and 80.47 for gradient boosting. Though the scores has reduced very slight for random forest but this is the best approximation after reducing overtting and variance. So we will select Random Forest as our best performing model for further deployment process.

Saving the model :Randon Forest Regressor :Best Model

```
import pickle
filename='house_prediction.pkl'
pickle.dump(rfr,open(filename,'wb'))
```

After that I used the test_dataset and perform 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.

```
df_test.head(5)
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr

```
#checking null values
```

```
df_test.apply(lambda x: sum(x.isnull()))
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage      45
LotArea           0
Street            0
Alley           278
LotShape          0
LandContour       0
```

```
df_test.drop(['Alley','PoolQC','MiscFeature','Fence'],axis=1,inplace=True)
```

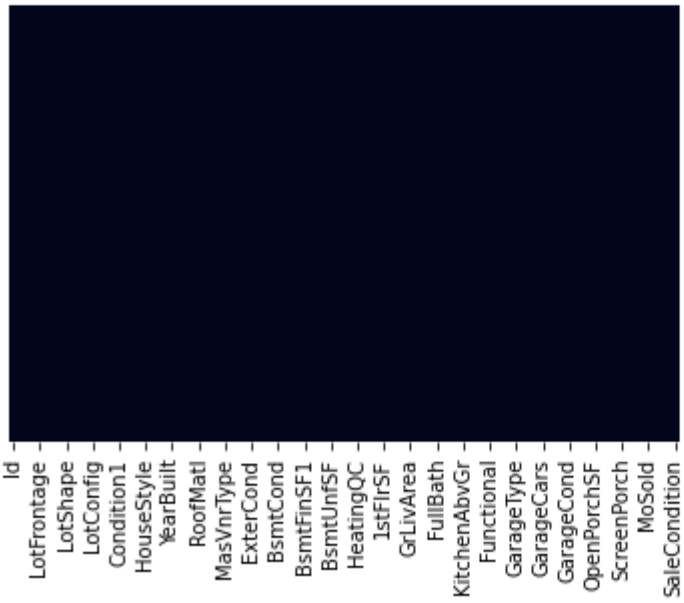
```
df_test["LotFrontage"].fillna(df_test["LotFrontage"].mean(),inplace=True)
df_test["MasVnrArea"].fillna(df_test["MasVnrArea"].mean(),inplace=True)
df_test["GarageYrBlt"].fillna(df_test["GarageYrBlt"].mean(),inplace=True)
```

```
df_test["MasVnrType"].fillna(df_test["MasVnrType"].mode()[0],inplace=True)
df_test["BsmtQual"].fillna(df_test["BsmtQual"].mode()[0],inplace=True)
df_test["BsmtCond"].fillna(df_test["BsmtCond"].mode()[0],inplace=True)
df_test["BsmtExposure"].fillna(df_test["BsmtExposure"].mode()[0],inplace=True)
df_test["BsmtFinType1"].fillna(df_test["BsmtFinType1"].mode()[0],inplace=True)
df_test["BsmtFinType2"].fillna(df_test["BsmtFinType2"].mode()[0],inplace=True)
df_test["FireplaceQu"].fillna(df_test["FireplaceQu"].mode()[0],inplace=True)
df_test["GarageType"].fillna(df_test["GarageType"].mode()[0],inplace=True)
df_test["GarageFinish"].fillna(df_test["GarageFinish"].mode()[0],inplace=True)
df_test["GarageQual"].fillna(df_test["GarageQual"].mode()[0],inplace=True)
df_test["GarageCond"].fillna(df_test["GarageCond"].mode()[0],inplace=True)
df_test["Electrical"].fillna(df_test["Electrical"].mode()[0],inplace=True)
```

```
#validating the missing value
sns.heatmap(df_test.isnull(),yticklabels=False,cbar=False)
```

```
#validating the missing value
sns.heatmap(df_test.isnull(),yticklabels=False,cbar=False)
```

```
<AxesSubplot:>
```




```
#dropping the utility column as its dropped in train data
df_test.drop(['Utilities'],axis=1,inplace=True)
```

```
df_test.duplicated().sum()
```

```
0
```

```
df_test.shape
```

```
(292, 74)
```

Feature Selection using Perasons correlation :

As we have already performed our analysis train dataset,we neednt perform again in test dataset to avoid redundancy so we simply delete those columns

```
#dropping the correlated feature as its dropped in train data
df_test.drop(['1stFlrSF', 'BldgType', 'BsmtFinSF2', 'BsmtFullBath', 'BsmtQual', 'ExterQual',
 'Exterior2nd', 'Foundation', 'FullBath', 'GarageArea', 'GarageCars', 'GarageFinish',
 'GarageYrBlt', 'GrLivArea', 'HalfBath', 'KitchenQual', 'TotRmsAbvGrd'],axis=1,inplace=True)
```

```
df_test.shape
```

```
(292, 57)
```

Encoding the categorical object datatype column

9]:
```
#using label encoder
#using label encoder
from sklearn.preprocessing import LabelEncoder
enc = LabelEncoder()

for i in df_test.columns:
    if df_test[i].dtypes== "object":
        df_test[i]= enc.fit_transform(df_test[i].values.reshape(-1,1))
```

0]:
```
df_test.shape
```

0]:
```
(292, 57)
```

1]:
```
df_test.skew()
```

1]:
```
MSSubClass        1.358597
MSZoning          0.187174
LotFrontage       0.466813
LotArea          12.781805
Street          -12.020386
LotShape         -0.639195
```

:
```
#treating skewness
#using power transform it can work uoon both negative and positive skewed value
from sklearn.preprocessing import power_transform
df_test=power_transform(df_test,method='yeo-johnson')
df_test
```

:
```
array([[-1.14976802, -0.28316498,  0.98225286, ..., -0.65028987,
         0.29921966,  0.01281114],
       [ 1.33766595, -0.28316498,  0.0357901 , ...,  0.8638488 ,
        -3.41487924, -2.55481962],
       [-1.14976802, -0.28316498,  0.0357901 , ...,  0.8638488 ,
         0.29921966,  0.01281114],
       ...,
       [-1.14976802, -0.28316498,  0.0357901 , ...,  1.61904342,
         0.29921966,  0.01281114],
       [ 0.262613  ,  1.7993534 , -0.80954055, ..., -1.40924184,
         0.29921966,  0.01281114],
       [ 1.63938246,  1.7993534 , -2.49342934, ..., -1.40924184,
         0.29921966,  0.01281114]])
```

Thus we see skewness has been taken care of by power transform method.

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

]:
```
array([[-1.14976802, -0.28316498,  0.98225286, ..., -0.65028939,
         0.29921966,  0.01281114],
       [ 1.33766595, -0.28316498,  0.0357901 , ...,  0.86384929,
        -3.41487924, -2.55481962],
       [-1.14976802, -0.28316498,  0.0357901 , ...,  0.86384929,
         0.29921966,  0.01281114],
       ...,
       [-1.14976802, -0.28316498,  0.0357901 , ...,  1.61904391,
         0.29921966,  0.01281114],
       [ 0.262613  ,  1.7993534 , -0.80954055, ..., -1.40924135,
         0.29921966,  0.01281114],
       [ 1.63938246,  1.7993534 , -2.49342934, ..., -1.40924135,
         0.29921966,  0.01281114]])
```

Loading the saved model

]:
```
loaded_model=pickle.load(open('house_prediction.pkl','rb'))
result=loaded_model.score(x_test,y_test)
print(result*100) #it gives us 87% accuracy which is good.
```

```
87.64261071317814
```

]:
```
loaded_model
```

]:
```
RandomForestRegressor(max_depth=20, min_samples_leaf=2, n_estimators=150)
```

We can see that RandomForestRegressor algorithm, which was finalized as best model and saved after we found that it was the best model performing, is loaded and also showing the best parameters we obtained after doing Hyperparameter Tuning.

Predictions over test data

```
]: #Making a dataframe for the SalePrice predictions
   conclusion=pd.DataFrame([loaded_model.predict(x)]).T
   conclusion
```

]:

	0
0	356674.499286
1	270180.980937
2	278641.219270
3	177417.904735
4	198893.371026
5	89947.155476
6	113713.628921
7	359640.448730
8	229704.333048
9	154930.269127
10	90706.787799

We have predicted the values using the test dataset and now we will save the predicted values separately.

Saving the predicted values

```
27]: test_results=pd.DataFrame(conclusion)
     test_results.to_csv('HP_testdataresult.csv')
```

- ## Visualizations

Now, we will see the different plots done with this dataset in order to know the insight of the data present. Below are the codes given for the plots and the output obtained:

Importing required libraries and plotting graphs for categorical data

```
#Loading the required libraries
import numpy as np
import pandas as pd
import scipy
import sklearn
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.expand_frame_repr', False)
```

Univariate Analysis

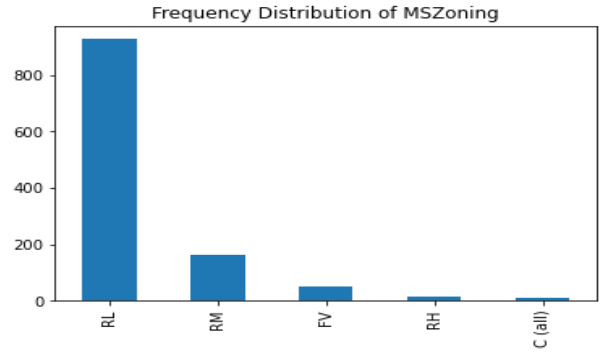
```
def univariateAnalysis_category(obj):
    print("Details of " + obj)
    print("------------------------------------------------------------------------")
    print(df_train[obj].value_counts())
    plt.figure()
    df_train[obj].value_counts().plot.bar(title="Frequency Distribution of " + obj)
    plt.show()
    plt.figure(figsize=(15,6))
    print("       ")
```

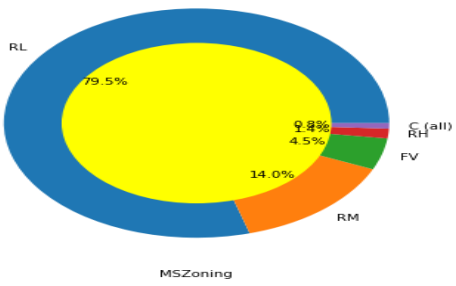
```
for x in df_train[obj]:
    univariateAnalysis_category(x)
```

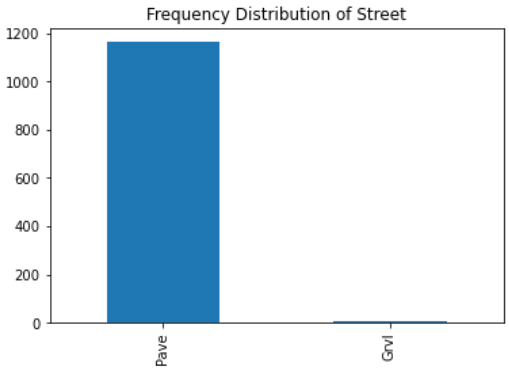
Code for pie-plot

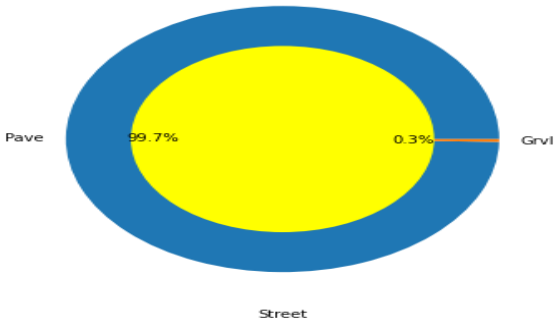
```
for i in df_train[obj]:
        plt.figure(figsize=(8,6))
        df_train[obj][i].value_counts().plot.pie(autopct='%1.1f%%')
        centre=plt.Circle((0,0),0.7,fc='yellow')
        fig=plt.gcf()
        fig.gca().add_artist(centre)
        plt.xlabel(i)
        plt.ylabel('')
        plt.figure()
```

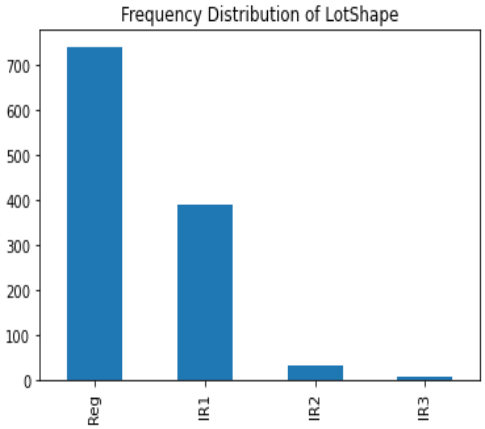
Thus below are the frequency distribution plot and pie-plot side by side:

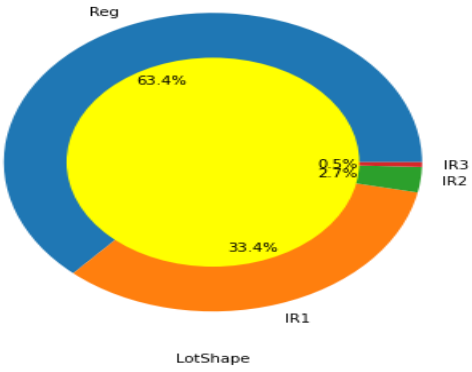


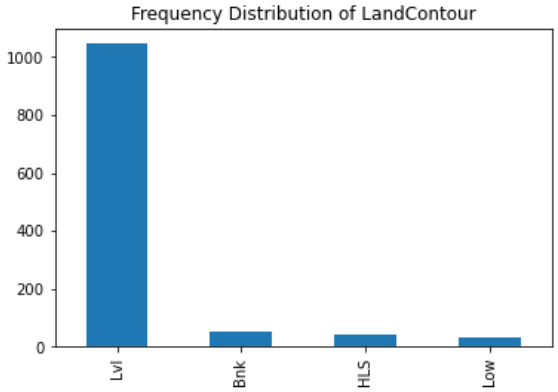


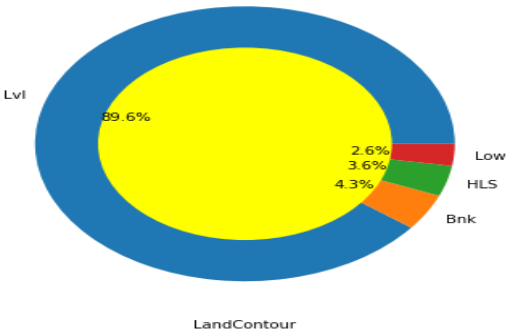

Frequency Distribution of Street
1200
1000
800
600
400
200
0
Pave
Grvl

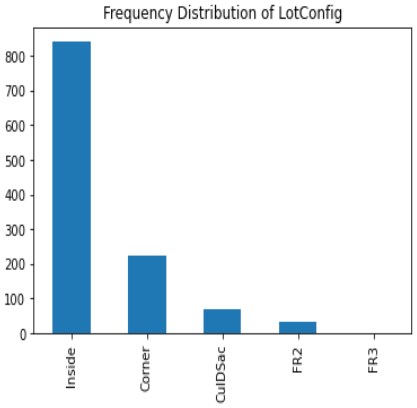

Pave
99.7%
0.3%
Grvl
Street

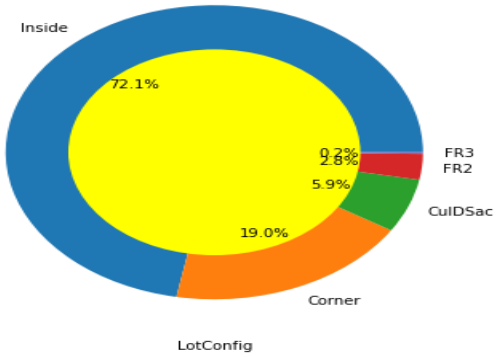

Frequency Distribution of LotShape
700
600
500
400
300
200
100
0
Reg
IR1
IR2
IR3

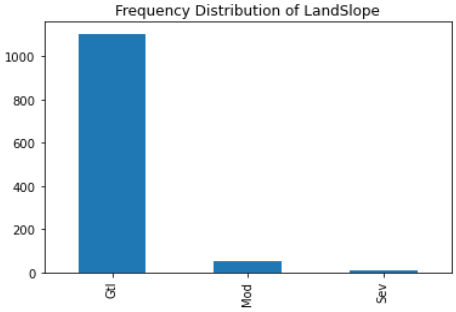

Reg
63.4%
0.5%
2.7%
IR3
IR2
33.4%
IR1
LotShape

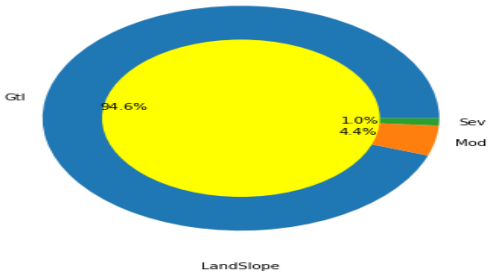

Frequency Distribution of LandContour
1000
800
600
400
200
0
Lvl
Bnk
HLS
Low

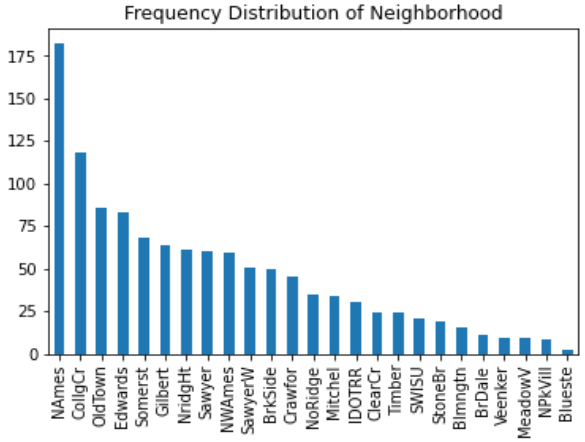

Lvl
89.6%
2.6%
3.6%
4.3%
Low
HLS
Bnk
LandContour

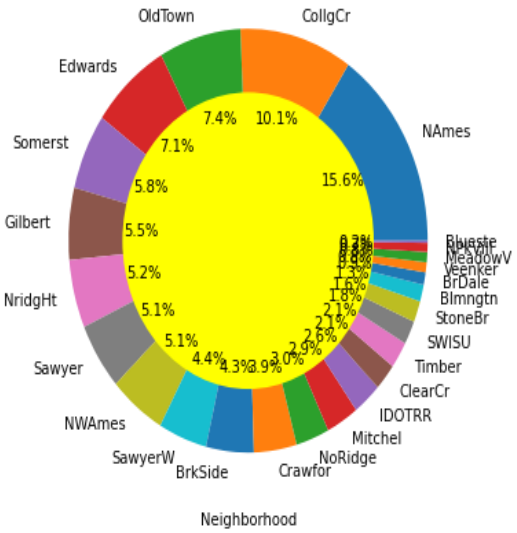

Frequency Distribution of LotConfig
800
700
600
500
400
300
200
100
0
Inside
Corner
CulDSac
FR2
FR3

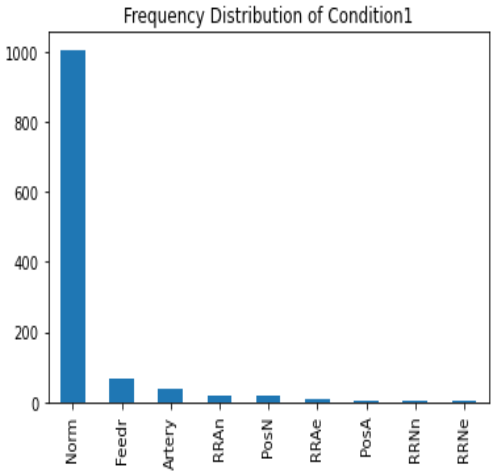

Inside
72.1%
0.2%
2.8%
FR3
FR2
5.9%
CulDSac
19.0%
Corner
LotConfig

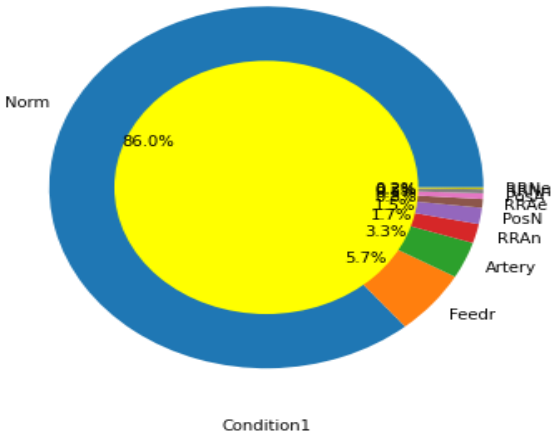

Frequency Distribution of LandSlope
1000
800
600
400
200
0
Gtl
Mod
Sev

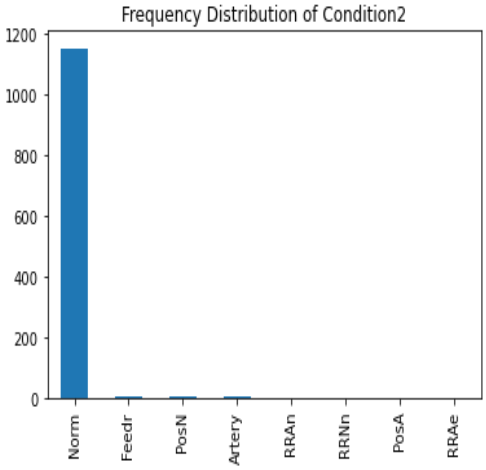

Gtl
94.6%
1.0%
4.4%
Sev
Mod
LandSlope

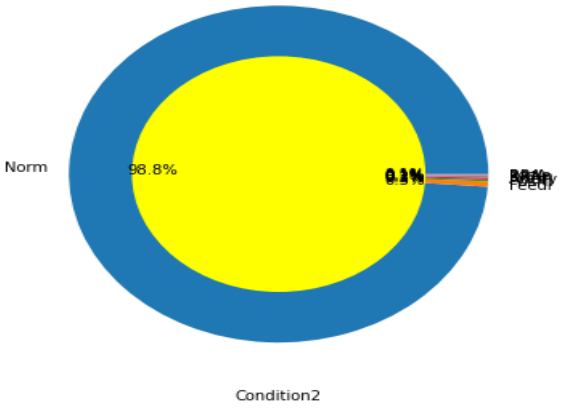

Frequency Distribution of Neighborhood
175
150
125
100
75
50
25
0
NAmes
CollgCr
OldTown
Edwards
Somerst
Gilbert
NridgHt
Sawyer
NWAmes
SawyerW
BrkSide
Crawfor
NoRidge
Mitchel
IDOTRR
ClearCr
Timber
SWISU
StoneBr
Blmngtn
BrDale
Veenker
MeadowV
NPkVill
Blueste

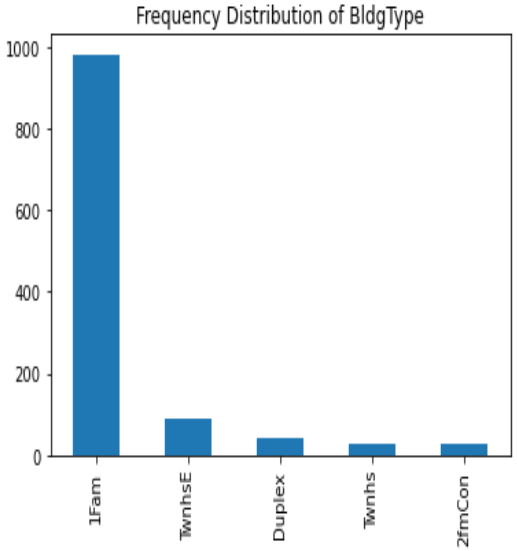

OldTown
CollgCr
Edwards
7.4%
10.1%
NAmes
Somerst
7.1%
15.6%
5.8%
Gilbert
5.5%
5.2%
NridgHt
5.1%
5.1%
4.4%
4.3%
3.9%
3.0%
2.9%
2.6%
2.1%
1.8%
1.6%
1.3%
Sawyer
NWAmes
SawyerW
BrkSide
Crawfor
NoRidge
Mitchel
IDOTRR
ClearCr
Timber
SWISU
StoneBr
Blmngtn
BrDale
Veenker
MeadowV
Neighborhood

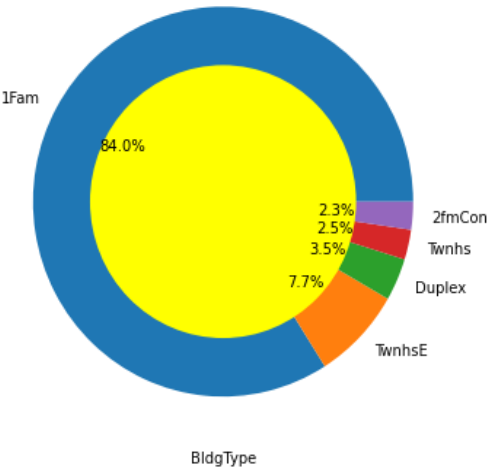

Frequency Distribution of Condition1
1000
800
600
400
200
0
Norm
Feedr
Artery
RRAn
PosN
RRAe
PosA
RRNn
RRNe

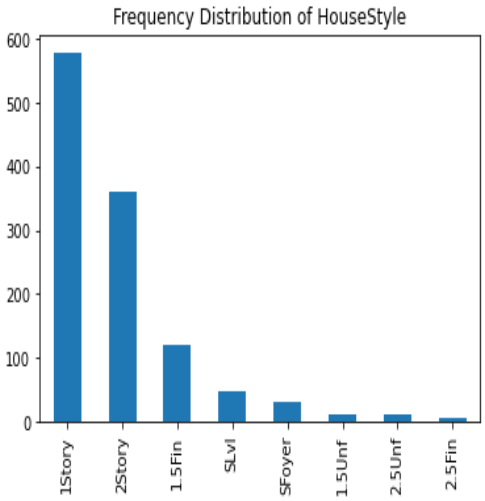

Norm
86.0%
5.7%
3.3%
1.7%
PosN
RRAn
Artery
Feedr
Condition1

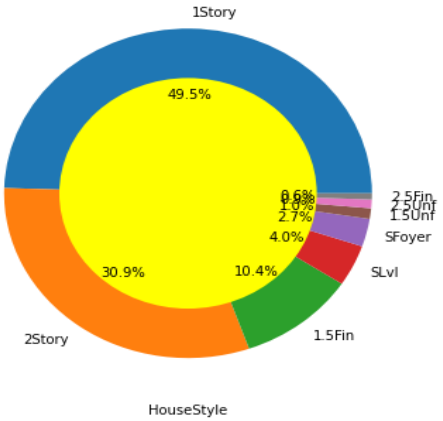

Frequency Distribution of Condition2
1200
1000
800
600
400
200
0
Norm
Feedr
PosN
Artery
RRAn
RRNn
PosA
RRAe

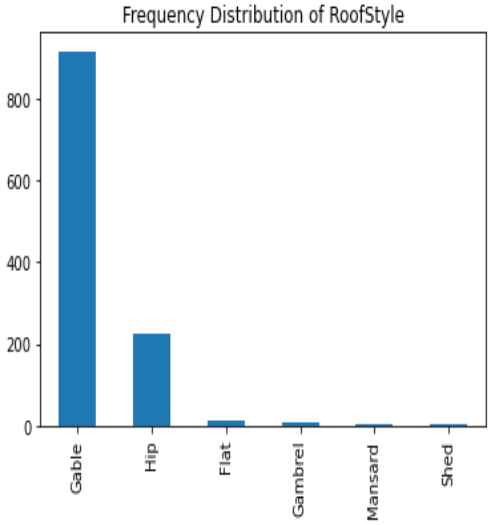

Norm
98.8%
Condition2

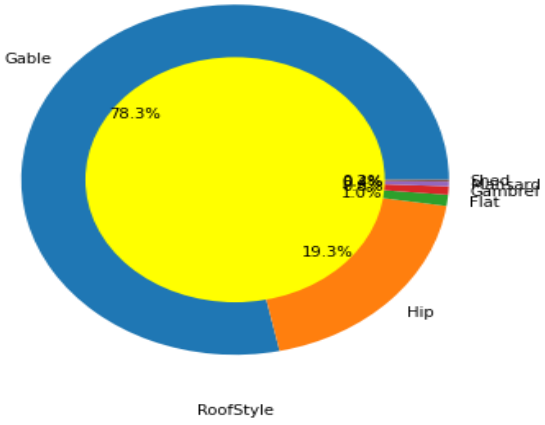

Frequency Distribution of BldgType
1000
800
600
400
200
0
1Fam
TwnhsE
Duplex
Twnhs
2fmCon

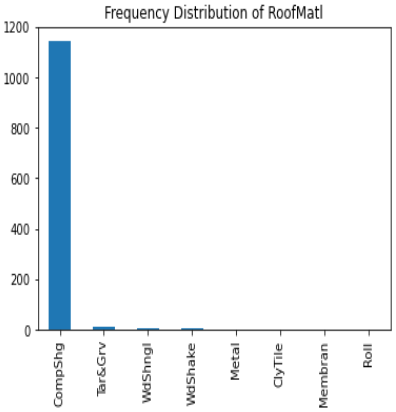

1Fam
84.0%
7.7%
3.5%
2.5%
2.3%
TwnhsE
Duplex
Twnhs
2fmCon
BldgType

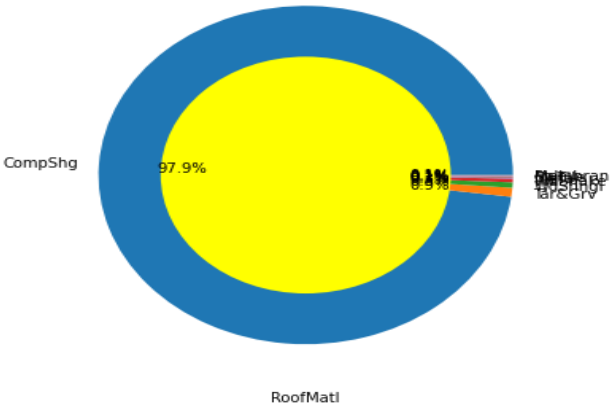

Frequency Distribution of HouseStyle
600
500
400
300
200
100
0
1Story
2Story
1.5Fin
SLvl
SFoyer
1.5Unf
2.5Unf
2.5Fin

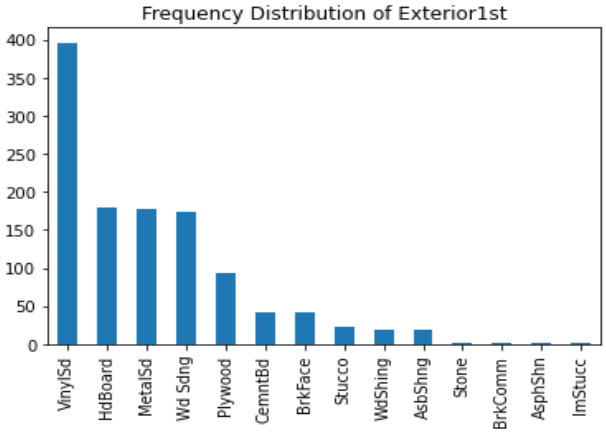

1Story
49.5%
2.5Fin
2.5Unf
1.5Unf
SFoyer
SLvl
4.0%
2.7%
1.0%
0.6%
30.9%
10.4%
2Story
1.5Fin
HouseStyle

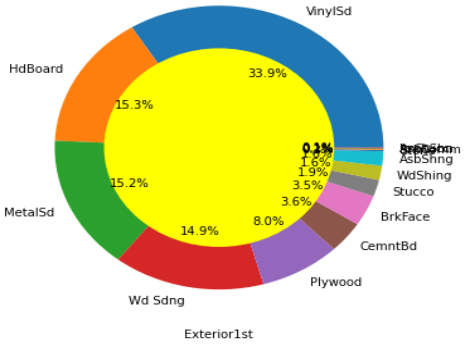

Frequency Distribution of RoofStyle
800
600
400
200
0
Gable
Hip
Flat
Gambrel
Mansard
Shed

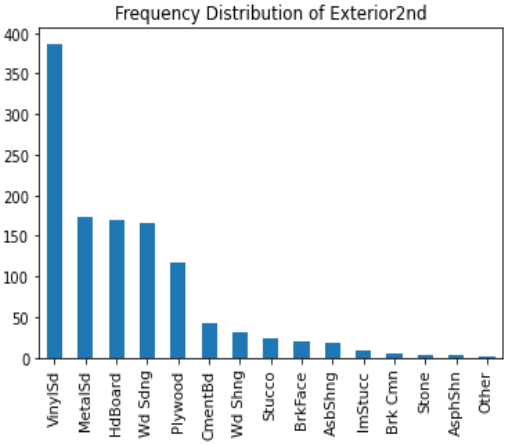

Gable
78.3%
Shed
Mansard
Gambrel
Flat
1.0%
19.3%
Hip
RoofStyle

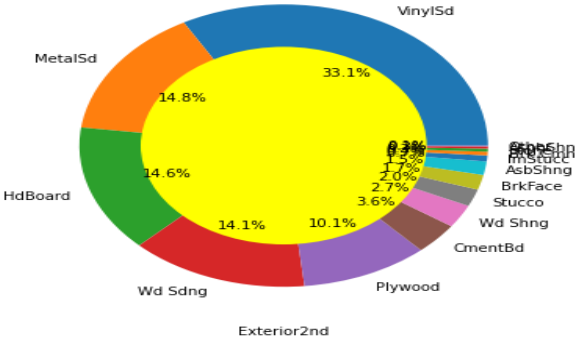

Frequency Distribution of RoofMatl
1200
1000
800
600
400
200
0
CompShg
Tar&Grv
WdShngl
WdShake
Metal
ClyTile
Membran
Roll

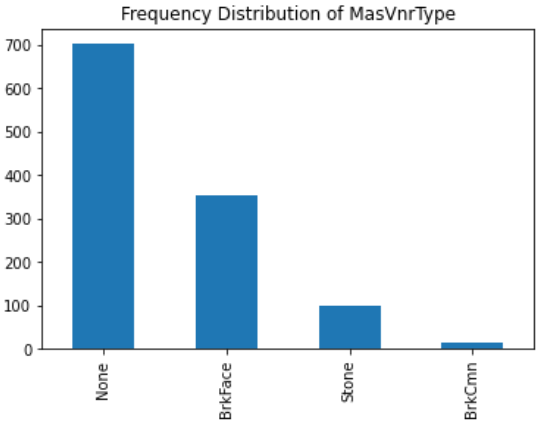

CompShg
97.9%
Tar&Grv
RoofMatl

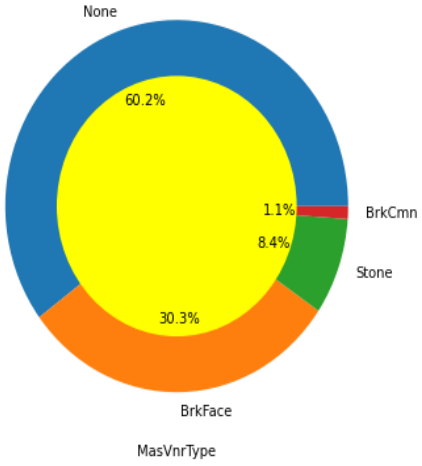

Frequency Distribution of Exterior1st
400
350
300
250
200
150
100
50
0
VinylSd
HdBoard
MetalSd
Wd Sdng
Plywood
CemntBd
BrkFace
Stucco
WdShing
AsbShng
Stone
BrkComm
AsphShn
ImStucc

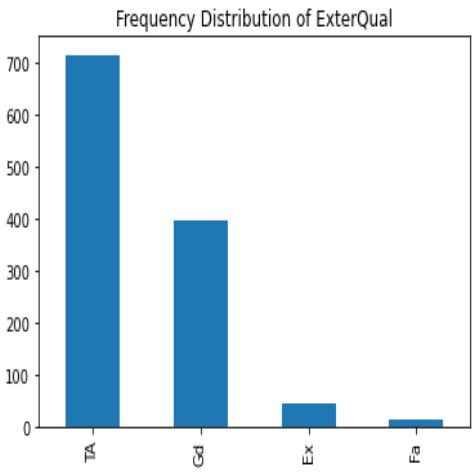

VinylSd
33.9%
HdBoard
15.3%
MetalSd
15.2%
Wd Sdng
14.9%
Plywood
8.0%
CemntBd
3.6%
BrkFace
3.5%
Stucco
1.9%
WdShing
1.6%
AsbShng
Exterior1st

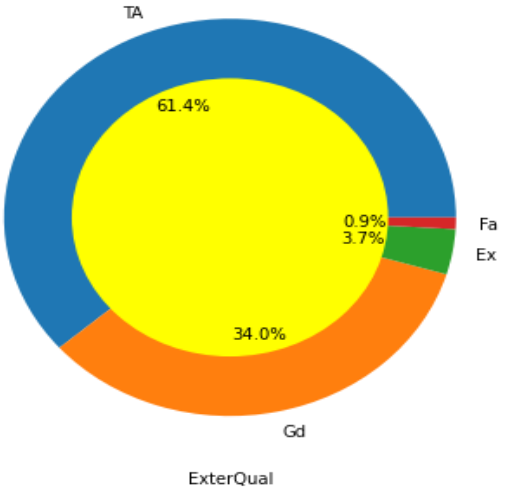

Frequency Distribution of Exterior2nd
400
350
300
250
200
150
100
50
0
VinylSd
MetalSd
HdBoard
Wd Sdng
Plywood
CmentBd
Wd Shng
Stucco
BrkFace
AsbShng
ImStucc
Brk Cmn
Stone
AsphShn
Other

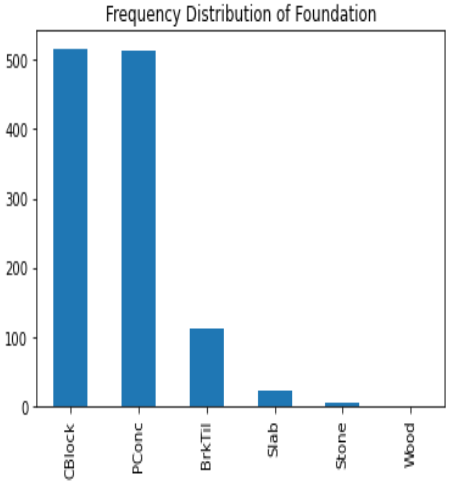

VinylSd
33.1%
MetalSd
14.8%
HdBoard
14.6%
Wd Sdng
14.1%
Plywood
10.1%
CmentBd
3.6%
Wd Shng
2.7%
Stucco
2.0%
BrkFace
1.7%
AsbShng
ImStucc
Exterior2nd

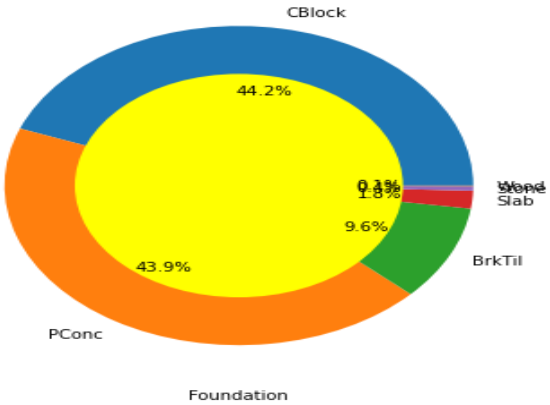

Frequency Distribution of MasVnrType
700
600
500
400
300
200
100
0
None
BrkFace
Stone
BrkCmn

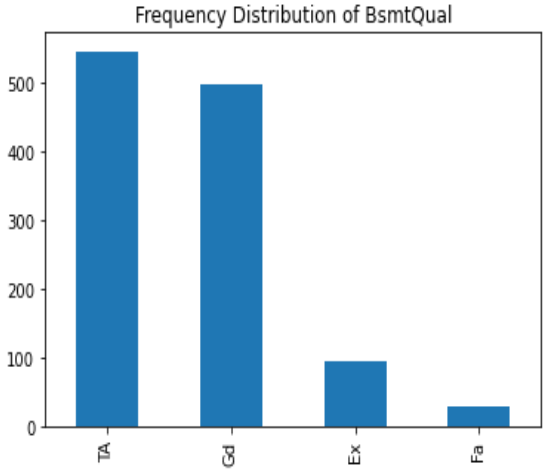

None
60.2%
BrkCmn
1.1%
Stone
8.4%
BrkFace
30.3%
MasVnrType

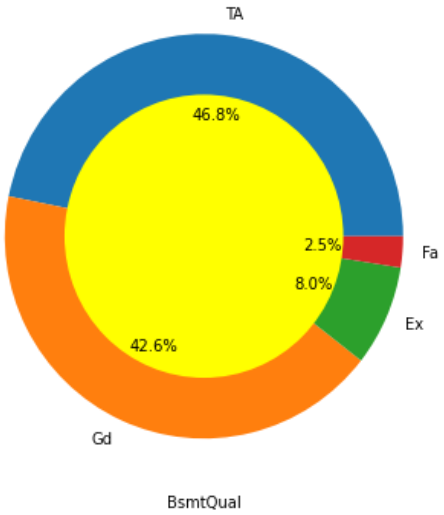

Frequency Distribution of ExterQual
700
600
500
400
300
200
100
0
TA
Gd
Ex
Fa

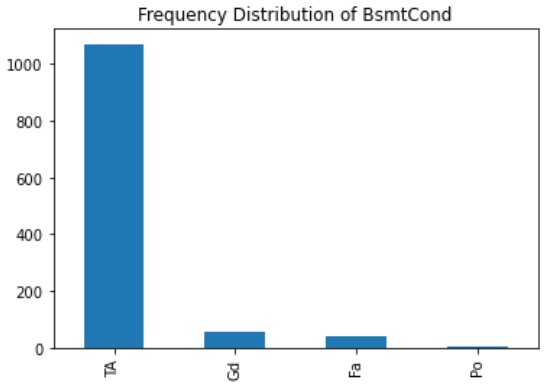

TA
61.4%
0.9%
3.7%
Fa
Ex
34.0%
Gd
ExterQual

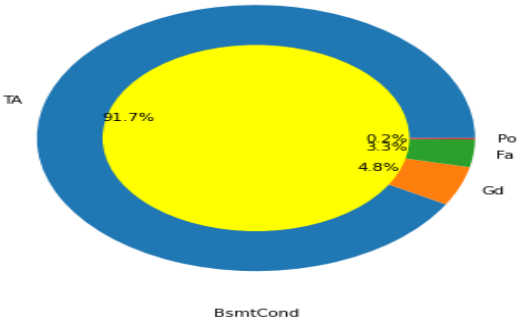

Frequency Distribution of Foundation
500
400
300
200
100
0
CBlock
PConc
BrkTil
Slab
Stone
Wood

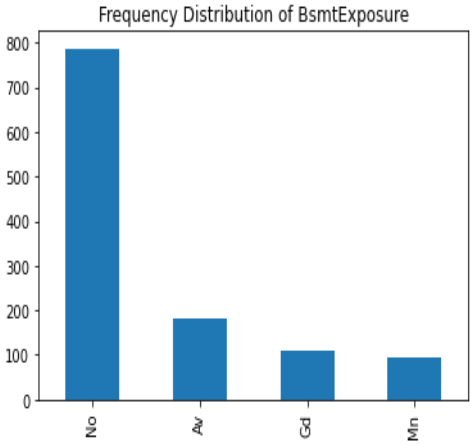

CBlock
44.2%
Slab
9.6%
BrkTil
43.9%
PConc
Foundation

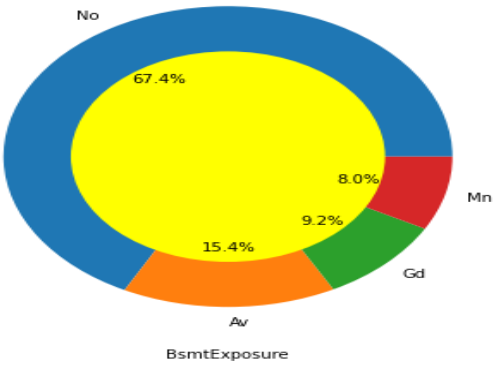

Frequency Distribution of BsmtQual
500
400
300
200
100
0
TA
Gd
Ex
Fa

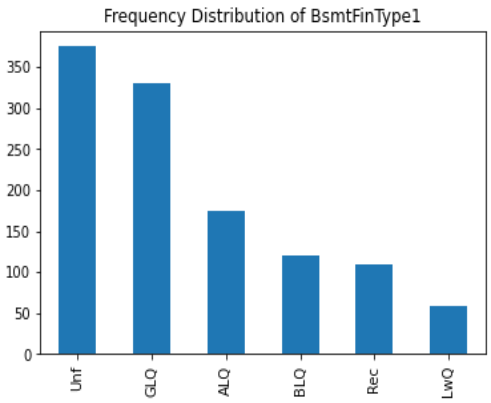

TA
46.8%
2.5%
Fa
8.0%
Ex
42.6%
Gd
BsmtQual

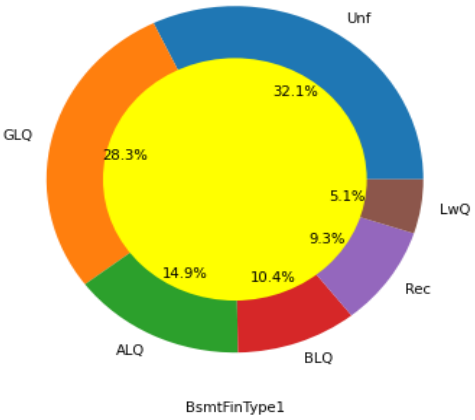

Frequency Distribution of BsmtCond
1000
800
600
400
200
0
TA
Gd
Fa
Po

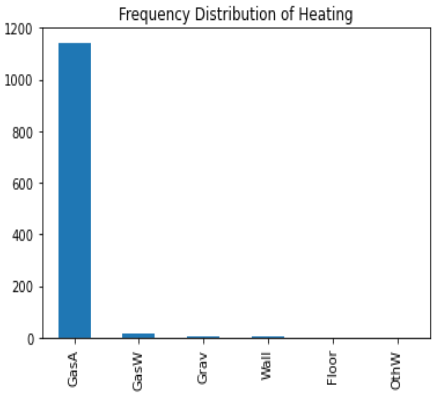

TA
91.7%
Po
Fa
Gd
4.8%
BsmtCond

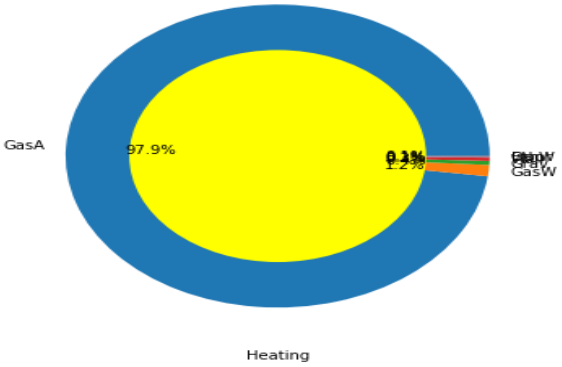

Frequency Distribution of BsmtExposure
800
700
600
500
400
300
200
100
0
No
Av
Gd
Mn

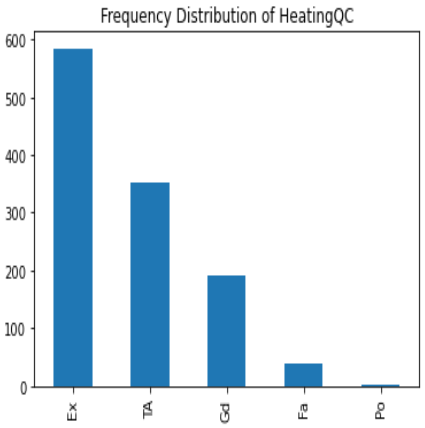

No
67.4%
8.0%
Mn
9.2%
15.4%
Gd
Av
BsmtExposure

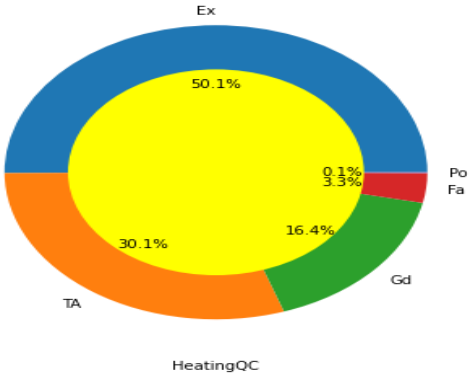

Frequency Distribution of BsmtFinType1
350
300
250
200
150
100
50
0
Unf
GLQ
ALQ
BLQ
Rec
LwQ

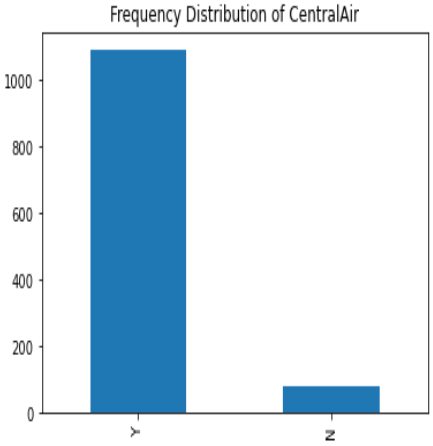

Unf
32.1%
GLQ
28.3%
5.1%
LwQ
9.3%
14.9%
10.4%
Rec
ALQ
BLQ
BsmtFinType1

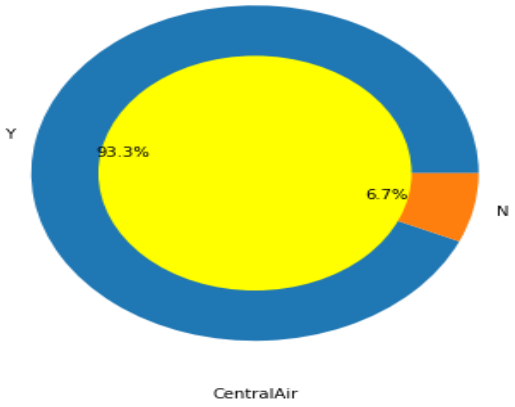

Frequency Distribution of Heating
1200
1000
800
600
400
200
0
GasA
GasW
Grav
Wall
Floor
OthW

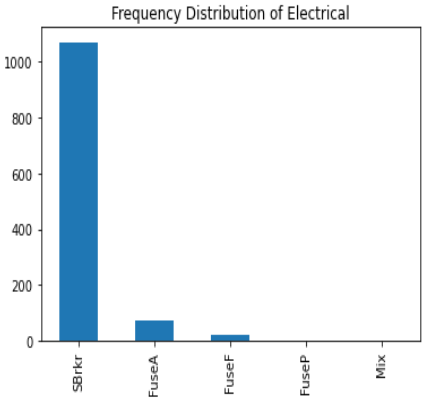

GasA
97.9%
GasW
Heating

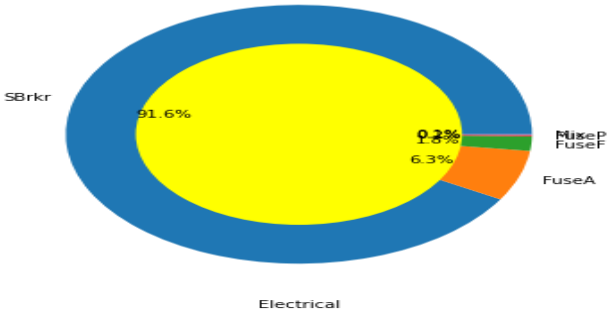

Frequency Distribution of HeatingQC
600
500
400
300
200
100
0
Ex
TA
Gd
Fa
Po

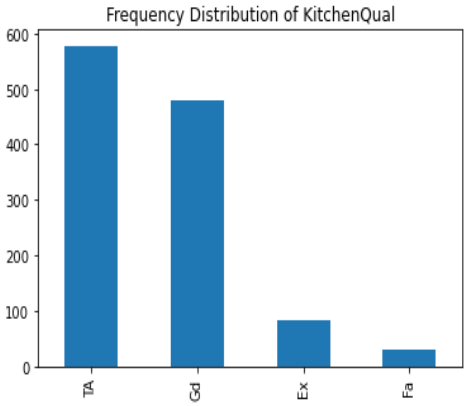

Ex
50.1%
Po
0.1%
Fa
3.3%
16.4%
30.1%
Gd
TA
HeatingQC

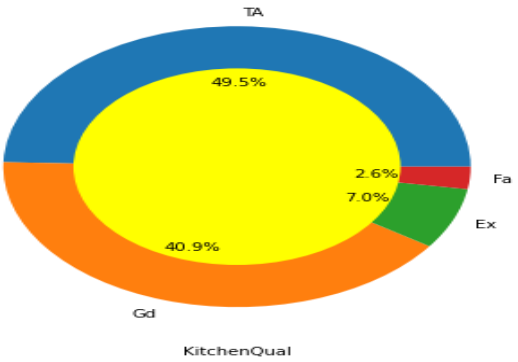

Frequency Distribution of CentralAir
1000
800
600
400
200
0
Y
N

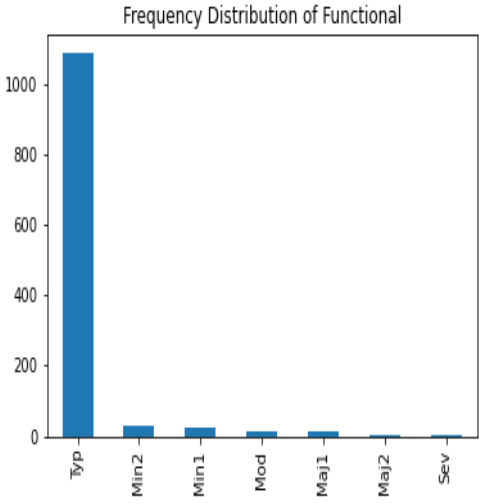

Y
93.3%
6.7%
N
CentralAir

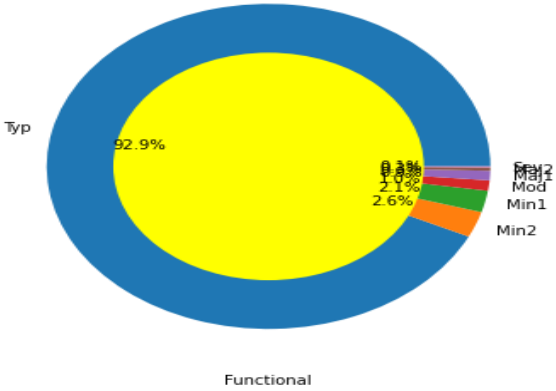

Frequency Distribution of Electrical
1000
800
600
400
200
0
SBrkr
FuseA
FuseF
FuseP
Mix

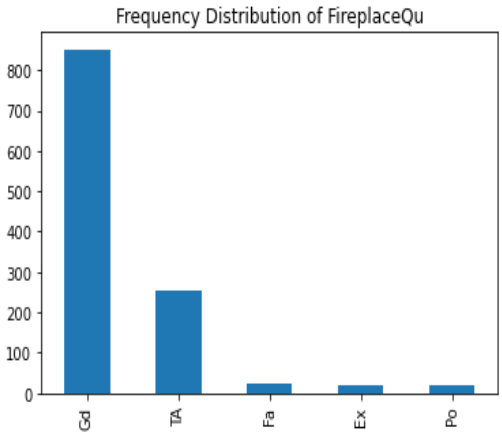

SBrkr
91.6%
6.3%
FuseA
FuseF
FuseP
Electrical

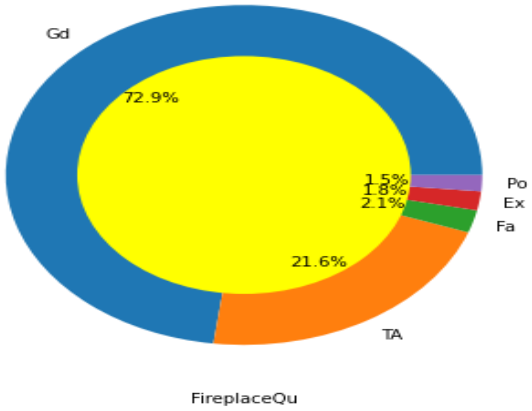

Frequency Distribution of KitchenQual
600
500
400
300
200
100
0
TA
Gd
Ex
Fa

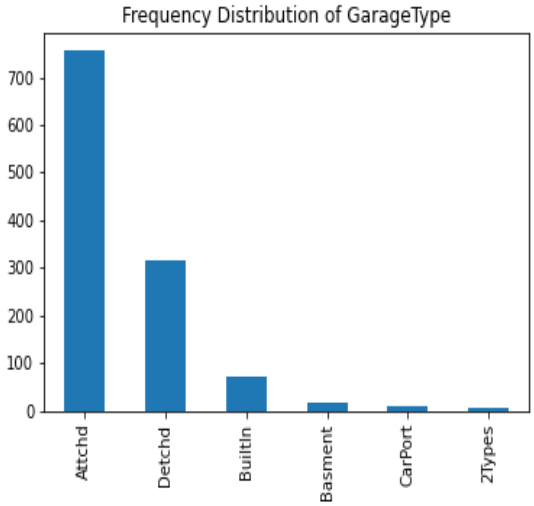

TA
49.5%
2.6%
Fa
7.0%
Ex
40.9%
Gd
KitchenQual

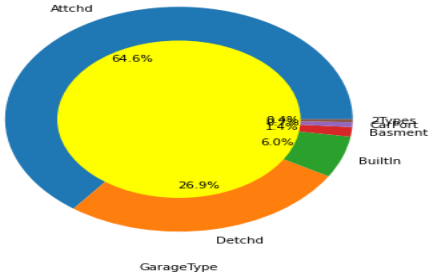

Frequency Distribution of Functional
1000
800
600
400
200
0
Typ
Min2
Min1
Mod
Maj1
Maj2
Sev

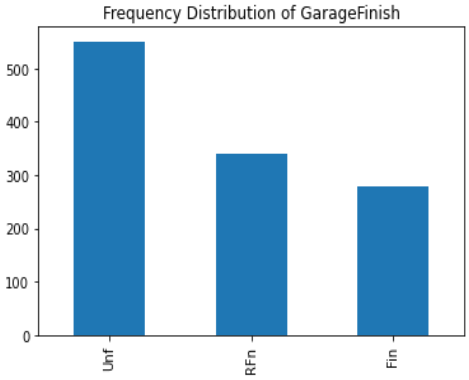

Typ
92.9%
2.1%
2.6%
Mod
Min1
Min2
Functional

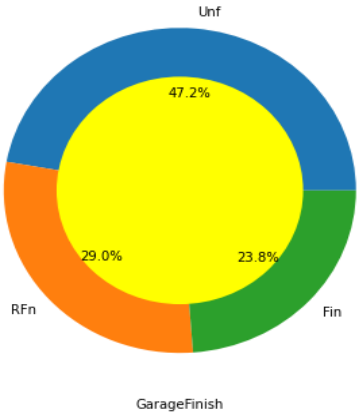

Frequency Distribution of FireplaceQu
800
700
600
500
400
300
200
100
0
Gd
TA
Fa
Ex
Po

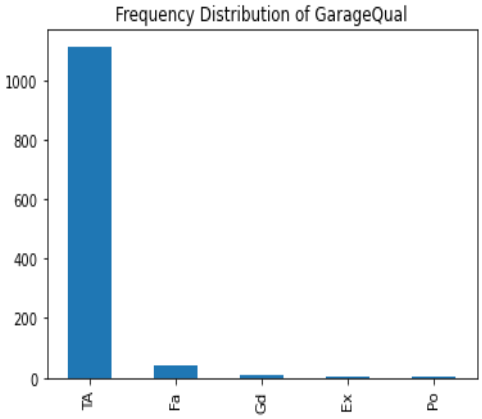

Gd
72.9%
1.5%
1.8%
2.1%
Po
Ex
Fa
21.6%
TA
FireplaceQu

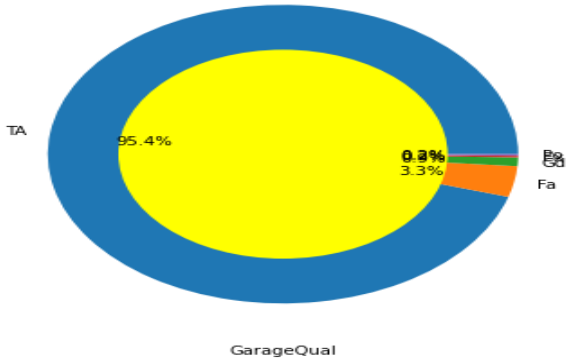

Frequency Distribution of GarageType
700
600
500
400
300
200
100
0
Attchd
Detchd
BuiltIn
Basment
CarPort
2Types

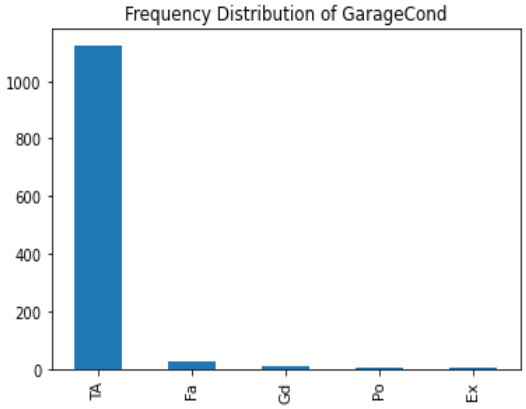

Attchd
64.6%
6.0%
BuiltIn
26.9%
Detchd
GarageType

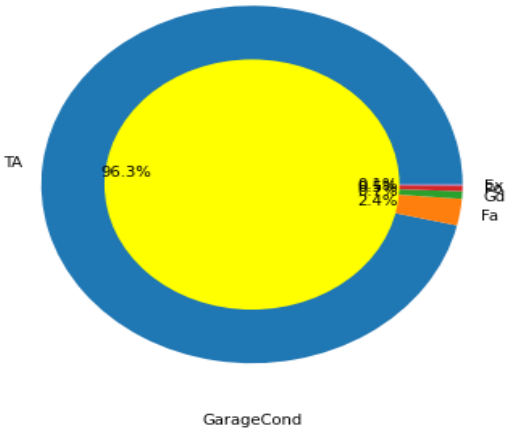

Frequency Distribution of GarageFinish
500
400
300
200
100
0
Unf
RFn
Fin

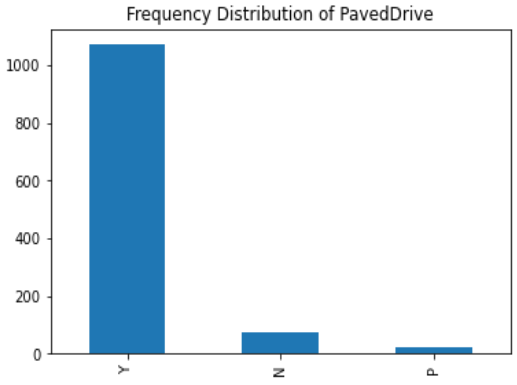

Unf
47.2%
29.0%
23.8%
RFn
Fin
GarageFinish

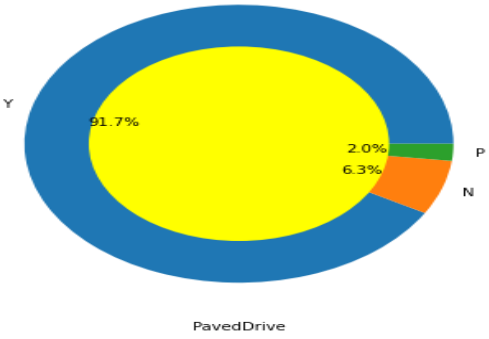

Frequency Distribution of GarageQual
1000
800
600
400
200
0
TA
Fa
Gd
Ex
Po

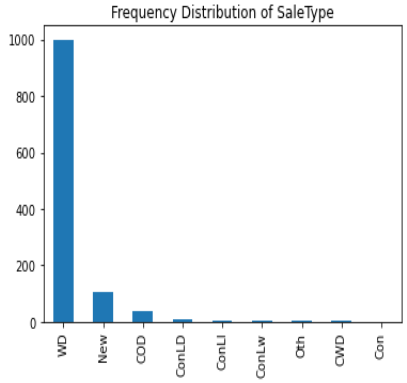

TA
95.4%
3.3%
Fa
GarageQual

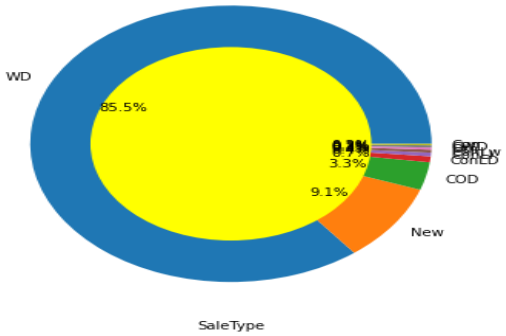

Frequency Distribution of GarageCond
1000
800
600
400
200
0
TA
Fa
Gd
Po
Ex

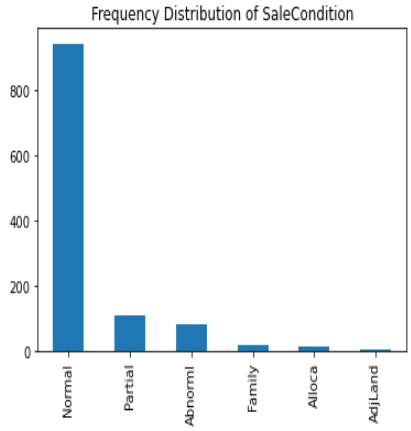

TA
96.3%
2.4%
Fa
GarageCond

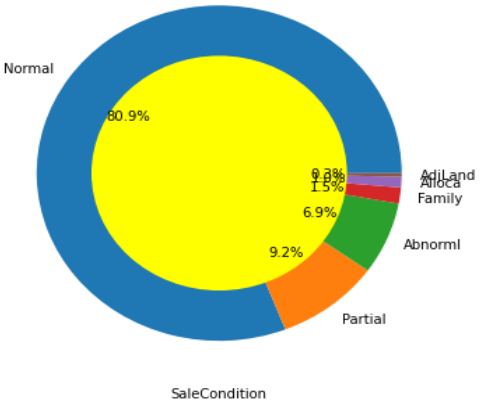

Frequency Distribution of PavedDrive
1000
800
600
400
200
0
Y
N
P


Y
91.7%
2.0%
P
6.3%
N
PavedDrive









Concepts: The frequency distribution plot shows that it is a graph or data set organized to represent the frequency of occurrence of each possible outcome of an event that is observed a specific number of times.

Pieplot:A pie chart is a circular analytical chart, which is divided into region to symbolize numerical percentage. In px.pie, data anticipated by the sectors of the pie to set the values. All sector are classify in names.

Observation:
Both observing the frequency distribution and pie plot, we made the following observation as under:

1.MSZoning (general zoning classification of the sale): About 79.5% of houses were sold in low density residential zones.

2.Street(type of road access to property):Thus almost 99.7% of the houses sold had access to the paved road.

3.LotSahpe(General shape of property):Around 63.4% of the houses sold are regular shaped followed by 33.4% which were slightly irregular shape.

4.LandContour:89.6% of the houses sold were nearly flat level

5.LotConfig:Around 72.1% of the houses sold were having inside lot configuarion followed by 19% which were corner lot configuration.

6.LandSlope :Around 94.6% of the houses sold had gentle slope

7.Neighborhood(Physical locations within Ames city limits): 15.6% of sold houses has neighbourhood of NWAmes(Northwest Ames) followed by CollgCr(College Creek) and least houses were purchased in neighbour hood of Bluestem.

8.Condition1(Proximity to various conditions):86% of houses sold had normal proximity to various conditions1 and least had RRAe PosA,RRNn,RRNe proximity.

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

10.BldgType(Type of dwelling):84% of houses purchased were single family detached,followed by 7.7% 2FmCon(Two-family Conversion).

11.HouseStyle(Style of dwelling):49% of the houses purchased had 1story followed by 30.1% of 2story style.

12.RoofStyle(Type of roof):78.3% of houses had Gable roof style followed by 18.3% had Hip roof style.

13.RoofMatl(Roof material):97.9% houses had CompShg(Standard (Composite) Shingle) roof material.

14.Exterior1st(Exterior covering on house):33.9% houses had Vinylsiding covering on exteriors and approx 15% have hard board and metal siding covering.

15.Exterior2nd(Exterior covering on house (if more than one material))-:33.1% houses have VinylSd(Vinyl Siding) and approx 15% had hard board and metal siding covering.

16.MasVnrType(Masonry veneer type):60.2% of houses have no masonry veneer type followed by BrkFace(Brick Face) (30.3%).

17.ExterQual(Evaluates the quality of the material on the exterior):61.4% of the sold hoUse have TA(Average/Typical) quality material on exterior followed by Gd(Good) 34%.

18.ExterCond(Evaluates the present condition of the material on the exterior):87.5% houses are currently in TA(average) condition of exterior material.

19.Foundation(Type of foundation):44.2% houses have foundation CBlock(Cinder Block) & 43.9% have PConc(Poured Contrete)

20.BsmtQual(Evaluates the height of the basement):46.8% of houses have TA(typical) (80-89 inches) basement height followed by Gd(Good) (90-99 inches)(42.6%)

21.BsmtCond(Evaluates the general condition of the basement):91.7% of houses have TA(Typical-slight dampness allowed)basment.

22.BsmtExposure(Refers to walkout or garden level walls):67.4% of houses have No(No Exposure) followed by Av(Average Exposure) 15.4%

23.BsmtFinType1(Rating of basement finished area):(32.1%) have Unf(Unfinshed) basement area and 28.3% comes under GLQ(good living quarters)

24.Heating(Type of heating):97.9% houses have GasA(Gas forced warm air furnace) heating type.

25.HeatingQC(Heating quality and condition):50.1% houses have excelent quality heating followed by 30.1% having avergae/Typical heating quality.

26.CentralAir(Central air conditioning)-:93.3% houses were hvaing central air conditioning.

27.Electrical(Electrical system):91.6% of houses had SbrKr(Standard Circuit Breakers & Romex) type of electrical systems.

28.KitchenQual(Kitchen quality):49.5% houses have average (TA) kitchen quality followed by 40.9% had good (Gd) kitchen quality.

29.Functional(Home functionality (Assume typical unless deductions are warranted)):92.9% houses have typical (TA) home functionality.

30.FireplaceQu(Fireplace quality):72.9% of the houses have Good - Masonry Fireplace in main level followed by 21.6% houses had Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement.

31.GarageType(Garage location):64.6% houses have Attached garage type,while 26.9% have Detchd(Detached from home)

32.GarageFinish(Interior finish of the garage):47.2% of houses have unfinished garage while 29% have RFn(Rough Finished)

33 GarageQual(Garage quality):95.4% of houses have average garage quality.

34.GarageCond(Garage condition):96.3% of houses have TA(average garage condition)

35.PavedDrive(Paved driveway):91.7% of houses have Y(paved drive) way

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

37.SaleType(Type of sale):85.5% houses have sale type WD(warranty deed - conventional)

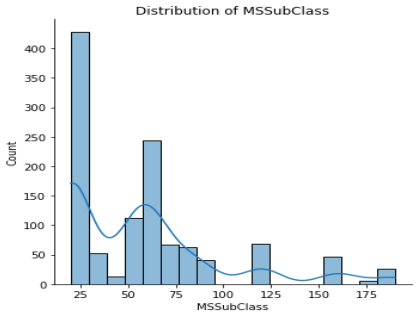
38.SaleCondition:80.9% of houses are in normal sale condition.

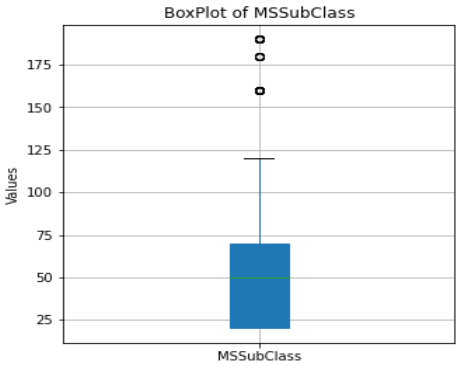
Univariate Analysis for non_object data i.e continuous data:

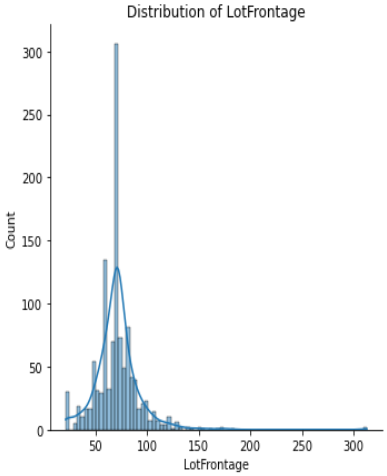
```
def univariateAnalysis_numeric(non_obj,nbins):
    print("Description of " + non_obj)
    print("----------------------------------------------------------------------------")
    print(df_train[non_obj].describe(),end=' ')

    plt.figure(figsize=(5,5))
    print("----------------------------------------------------------------------------")
    sns.displot(df_train[non_obj], kde=True)#, color='blue')
    plt.title("Distribution of " + non_obj)
    plt.show()

    plt.figure(figsize=(5,5))
    print("----------------------------------------------------------------------------")
    ax = df_train.boxplot(non_obj,patch_artist=True)
    #for box in ax['boxes']:
    #    box.set(color='y', linewidth=2)
    #    box.set(facecolor = 'green')
        #box.set(hatch = '/')
    plt.title("BoxPlot of " + non_obj)
    # plt.xlabel(non_obj)
    plt.ylabel('Values')
    plt.show()
```

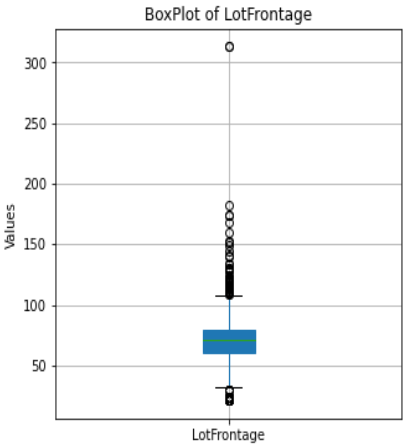
```
for x in non_obj:
    univariateAnalysis_numeric(x, 50);
```

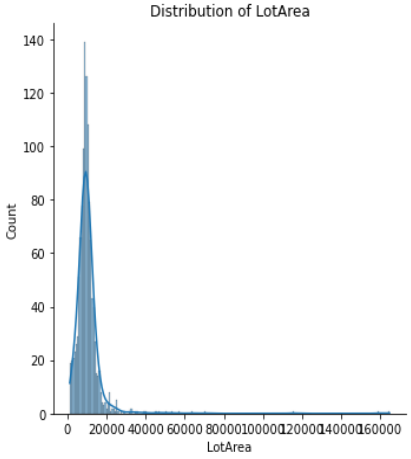
Below are the distribution plot,boxplot of the continuous data :

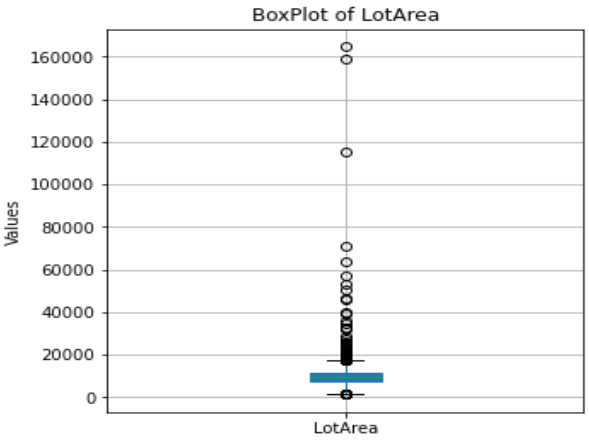


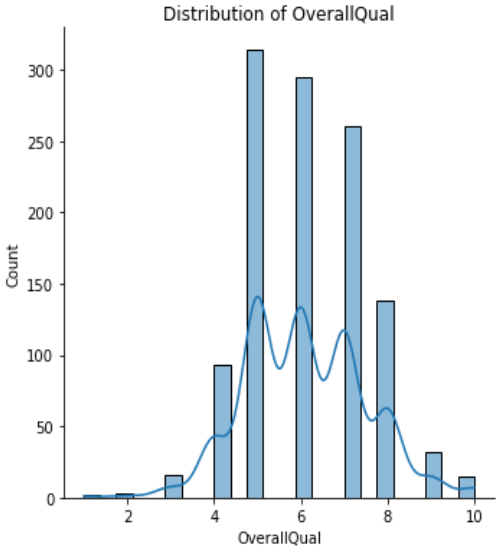


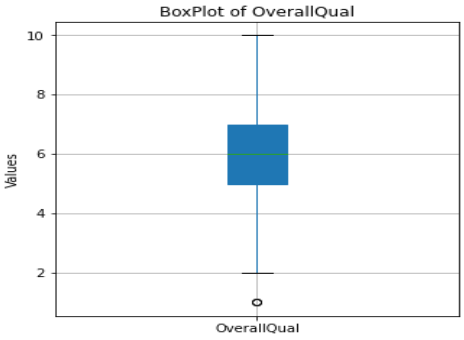

Distribution of LotFrontage
Count
LotFrontage

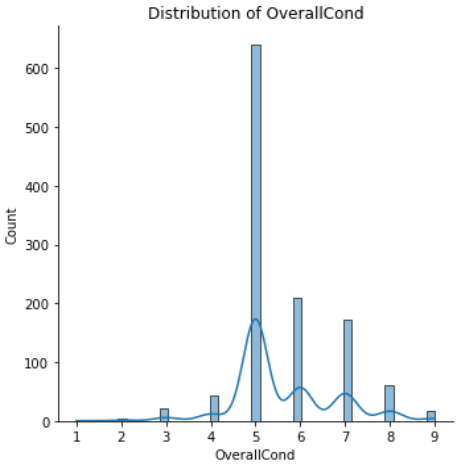

BoxPlot of LotFrontage
Values
LotFrontage

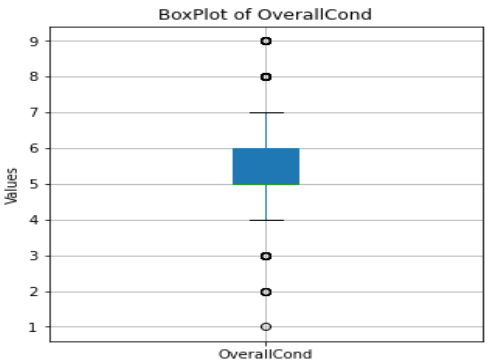

Distribution of LotArea
Count
LotArea

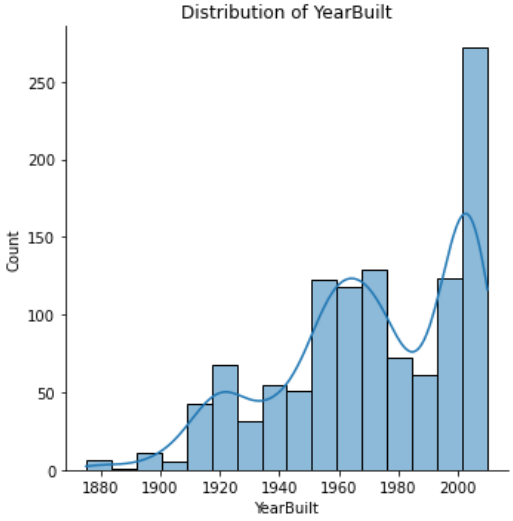

BoxPlot of LotArea
Values
LotArea

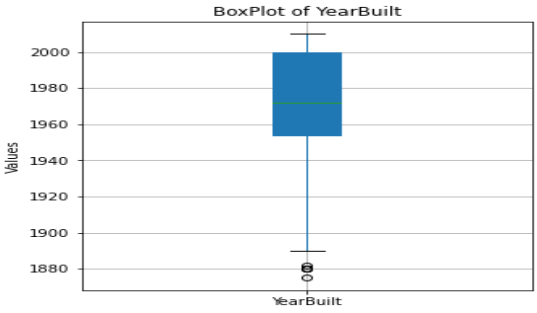

Distribution of OverallQual
Count
OverallQual

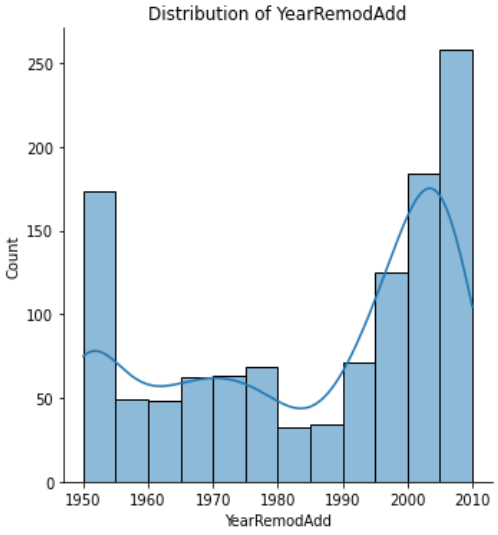

BoxPlot of OverallQual
Values
OverallQual

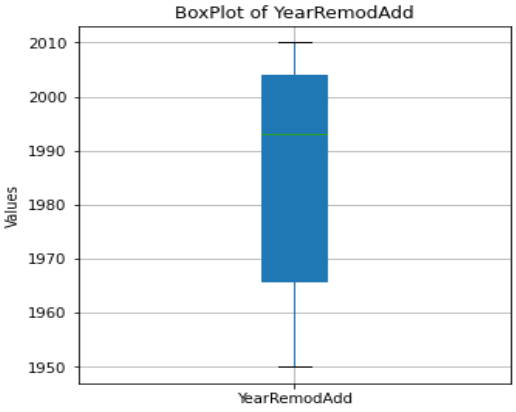

Distribution of OverallCond
Count
OverallCond

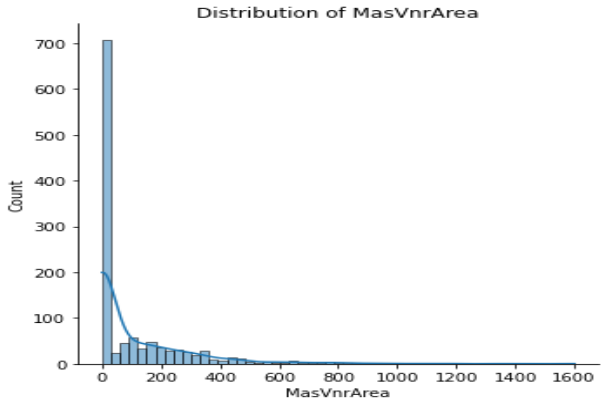

BoxPlot of OverallCond
Values
OverallCond

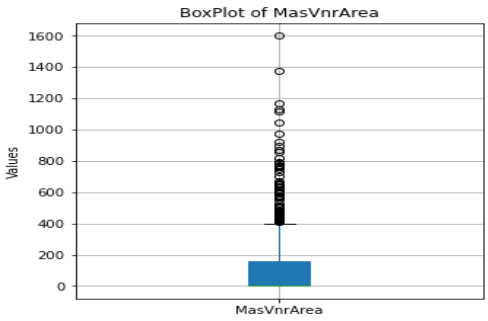

Distribution of YearBuilt
Count
YearBuilt

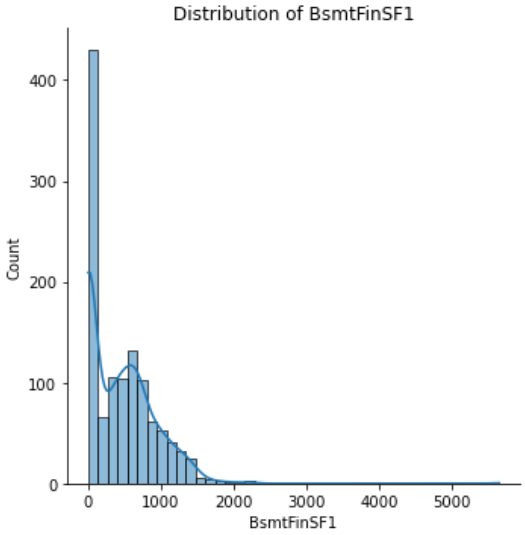

BoxPlot of YearBuilt
Values
YearBuilt

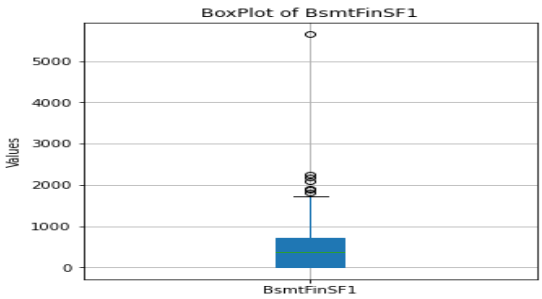

Distribution of YearRemodAdd
Count
YearRemodAdd

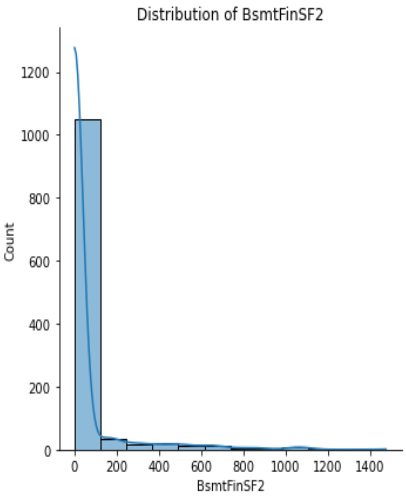

BoxPlot of YearRemodAdd
Values
YearRemodAdd

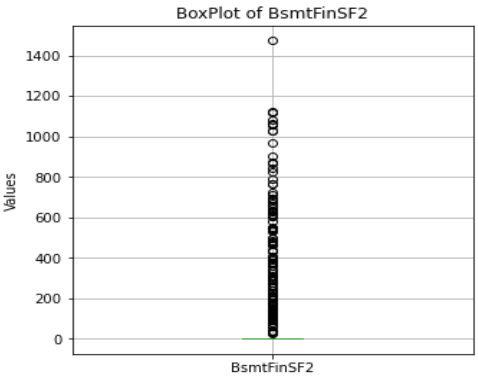

Distribution of MasVnrArea
Count
MasVnrArea
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200
1400
1600

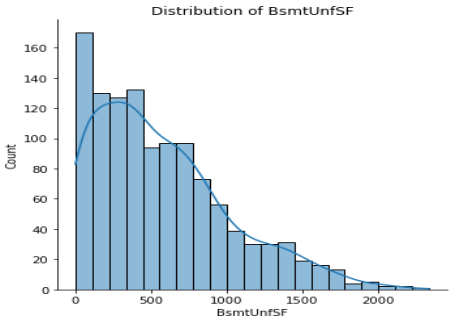

BoxPlot of MasVnrArea
Values
MasVnrArea
0
200
400
600
800
1000
1200
1400
1600

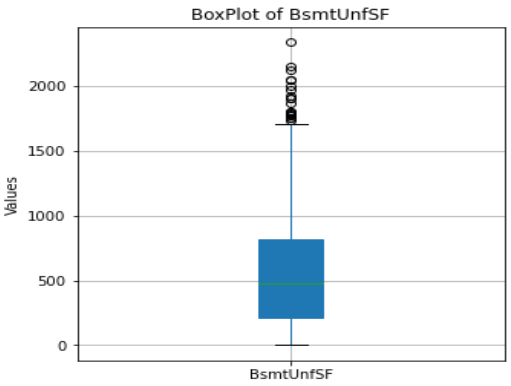

Distribution of BsmtFinSF1
Count
BsmtFinSF1
0
100
200
300
400
0
1000
2000
3000
4000
5000

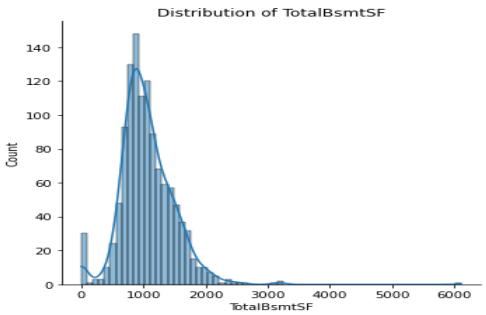

BoxPlot of BsmtFinSF1
Values
BsmtFinSF1
0
1000
2000
3000
4000
5000

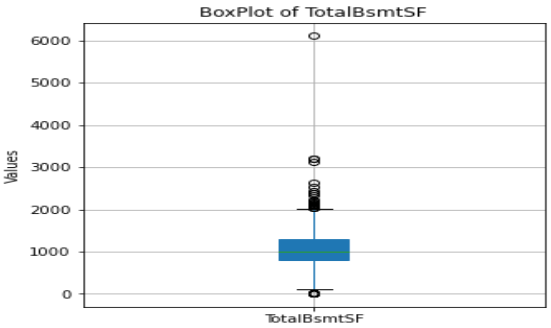

Distribution of BsmtFinSF2
Count
BsmtFinSF2
0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
1400

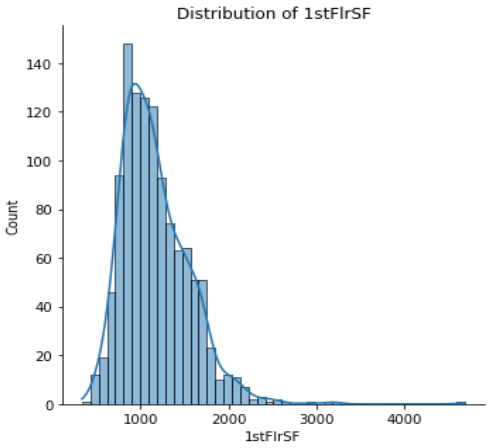

BoxPlot of BsmtFinSF2
Values
BsmtFinSF2
0
200
400
600
800
1000
1200
1400

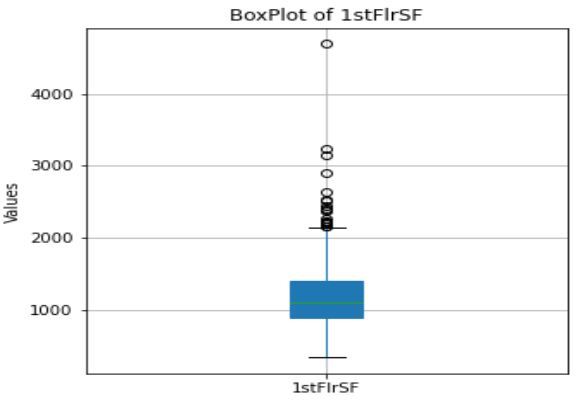

Distribution of BsmtUnfSF
Count
BsmtUnfSF

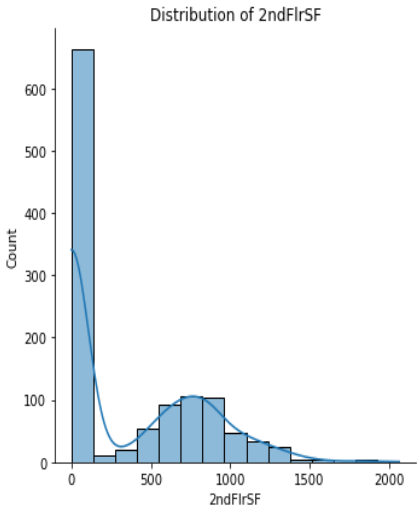

BoxPlot of BsmtUnfSF
Values
BsmtUnfSF

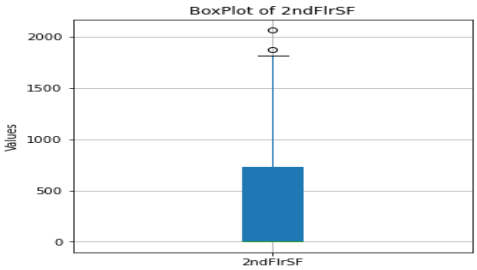

Distribution of TotalBsmtSF
Count
TotalBsmtSF

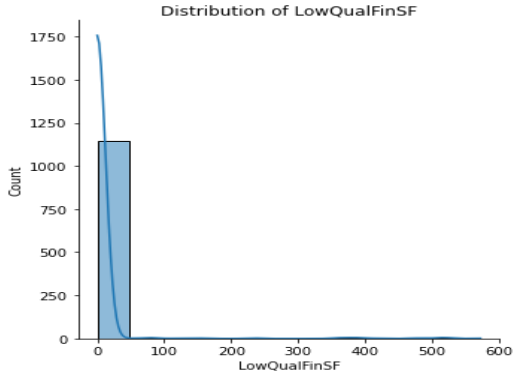

BoxPlot of TotalBsmtSF
Values
TotalBsmtSF

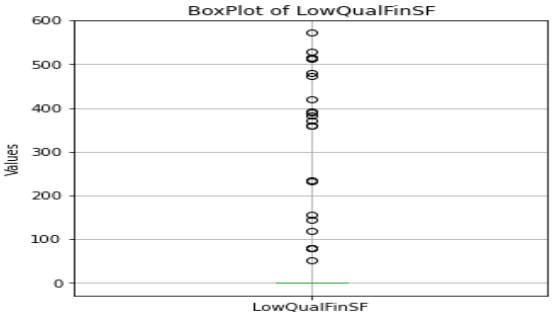

Distribution of 1stFlrSF
Count
1stFlrSF

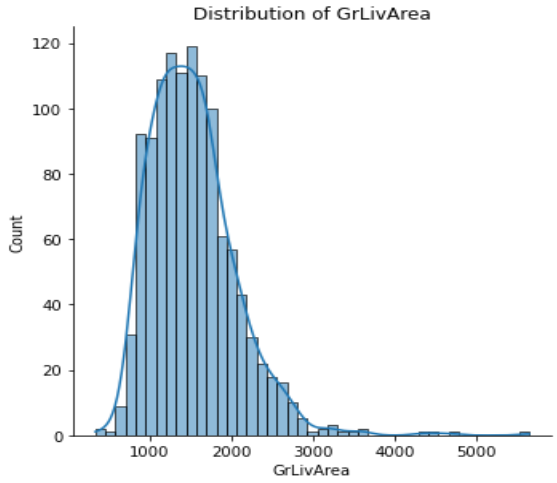

BoxPlot of 1stFlrSF
Values
1stFlrSF

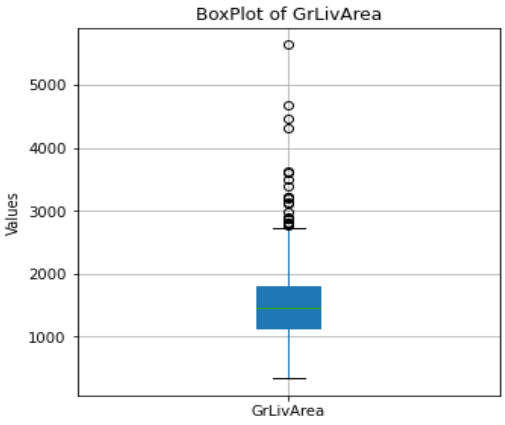

Distribution of 2ndFlrSF
Count
2ndFlrSF
0
100
200
300
400
500
600
0
500
1000
1500
2000

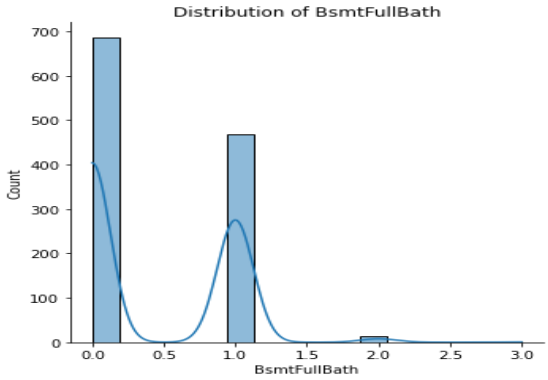

BoxPlot of 2ndFlrSF
Values
2ndFlrSF
0
500
1000
1500
2000

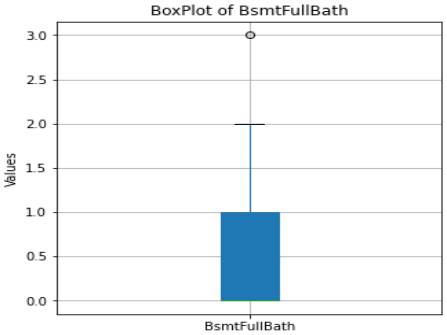

Distribution of LowQualFinSF
Count
LowQualFinSF
0
250
500
750
1000
1250
1500
1750
0
100
200
300
400
500
600

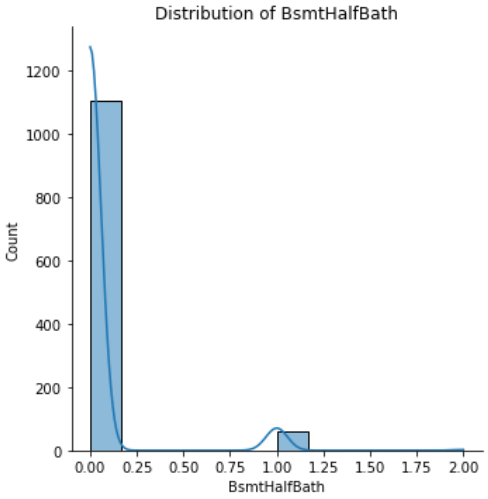

BoxPlot of LowQualFinSF
Values
LowQualFinSF
0
100
200
300
400
500
600

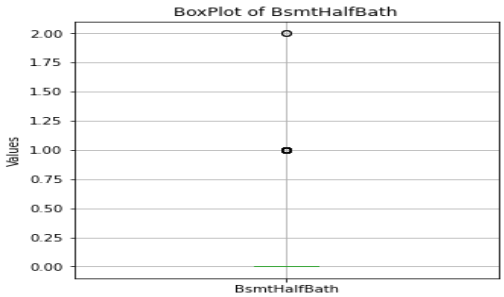

Distribution of GrLivArea
Count
GrLivArea
0
20
40
60
80
100
120
1000
2000
3000
4000
5000

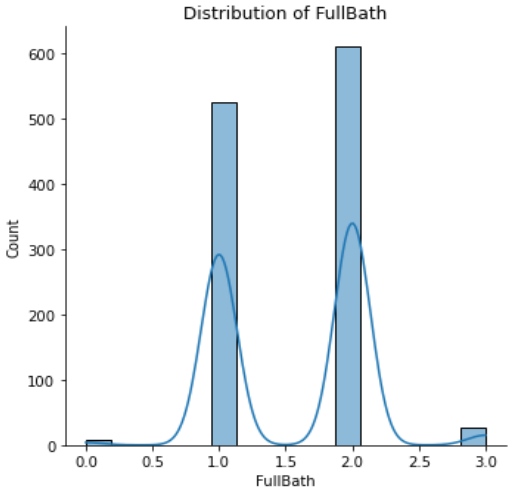

BoxPlot of GrLivArea
Values
GrLivArea
1000
2000
3000
4000
5000

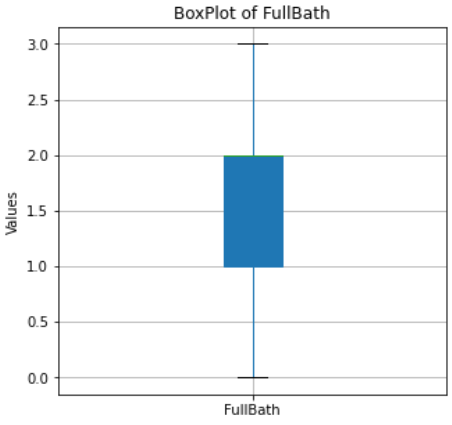

Distribution of BsmtFullBath
Count
BsmtFullBath

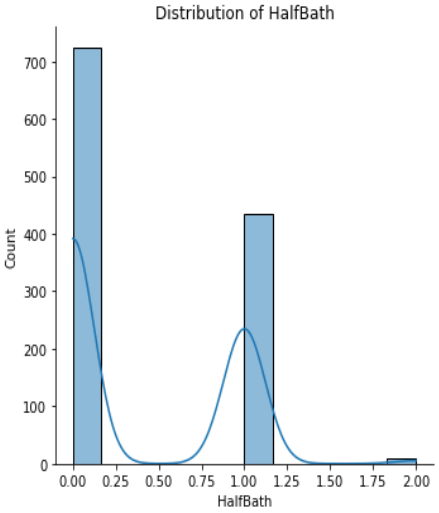

BoxPlot of BsmtFullBath
Values
BsmtFullBath

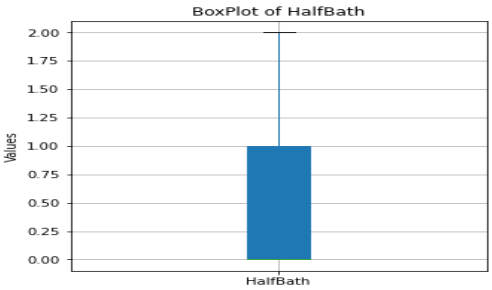

Distribution of BsmtHalfBath
Count
BsmtHalfBath

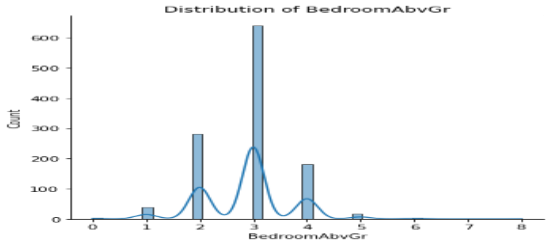

BoxPlot of BsmtHalfBath
Values
BsmtHalfBath

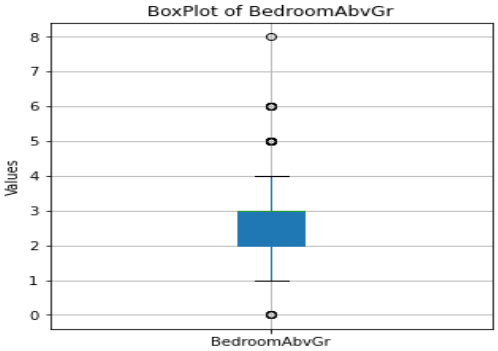

Distribution of FullBath
Count
FullBath

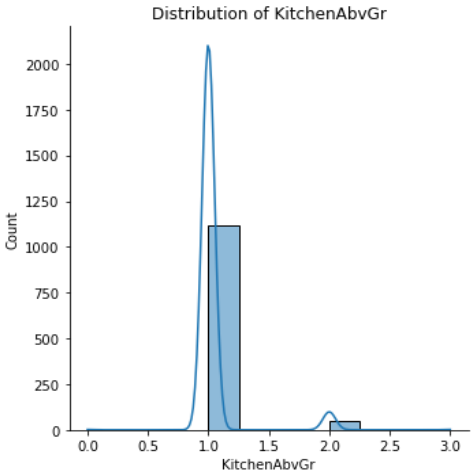

BoxPlot of FullBath
Values
FullBath

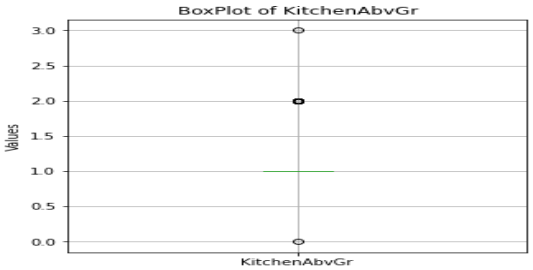

Distribution of HalfBath
Count
HalfBath

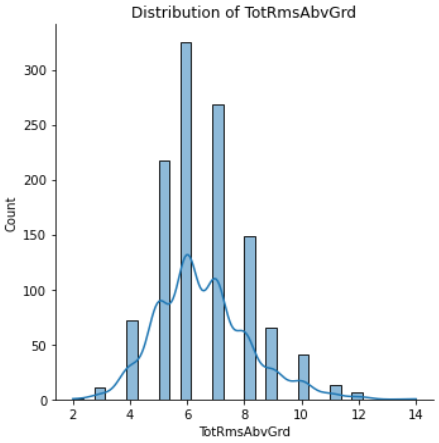

BoxPlot of HalfBath
Values
HalfBath

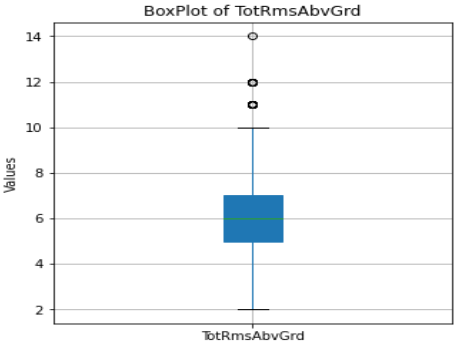

Distribution of BedroomAbvGr
Count
BedroomAbvGr

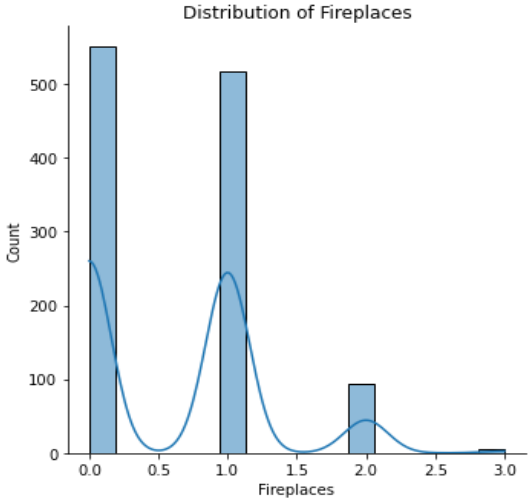

BoxPlot of BedroomAbvGr
Values
BedroomAbvGr

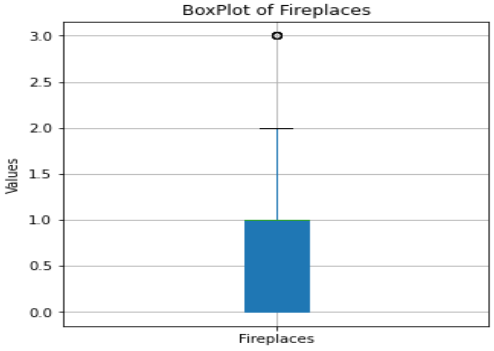

Distribution of KitchenAbvGr
Count
KitchenAbvGr

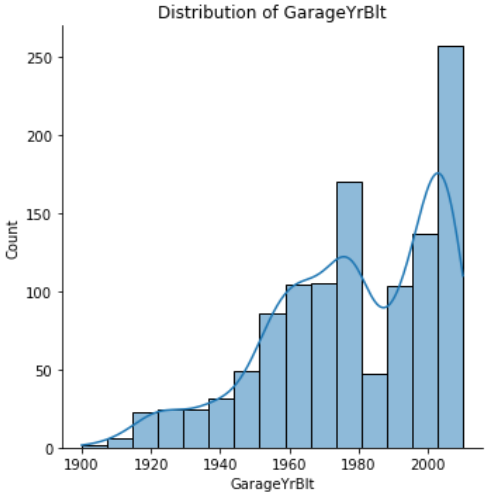

BoxPlot of KitchenAbvGr
Values
KitchenAbvGr

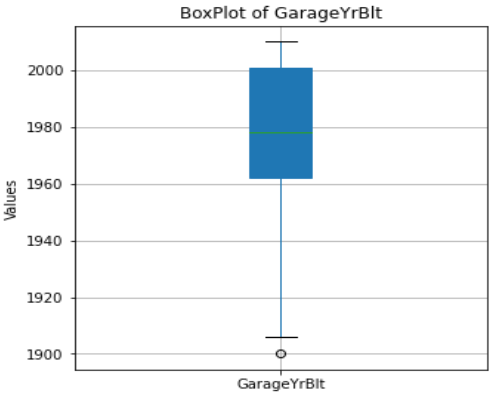

Distribution of TotRmsAbvGrd
Count
TotRmsAbvGrd

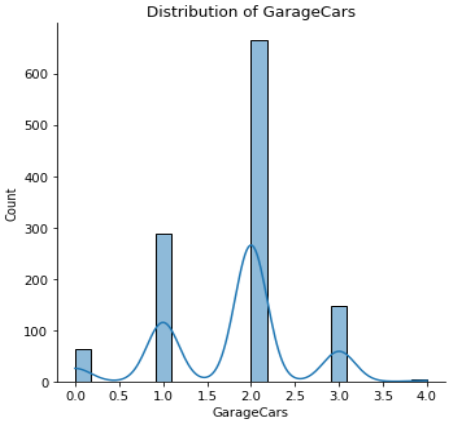

BoxPlot of TotRmsAbvGrd
Values
TotRmsAbvGrd

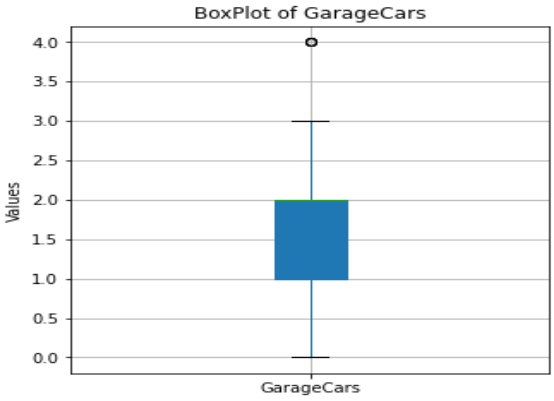

Distribution of Fireplaces
Count
Fireplaces

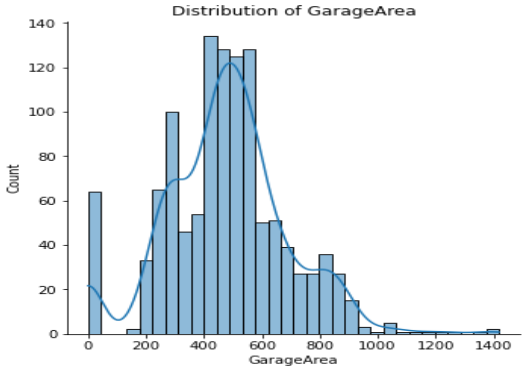

BoxPlot of Fireplaces
Values
Fireplaces

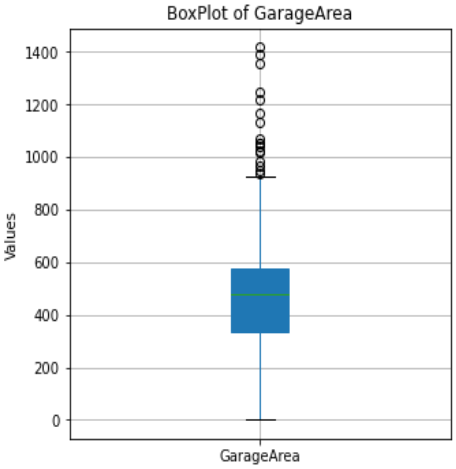

Distribution of GarageYrBlt
Count
GarageYrBlt

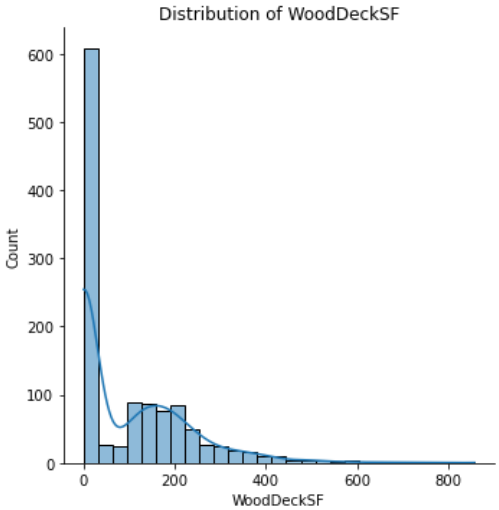

BoxPlot of GarageYrBlt
Values
GarageYrBlt

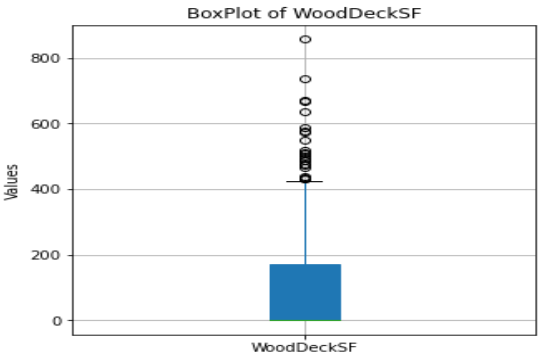

Distribution of GarageCars
Count
GarageCars

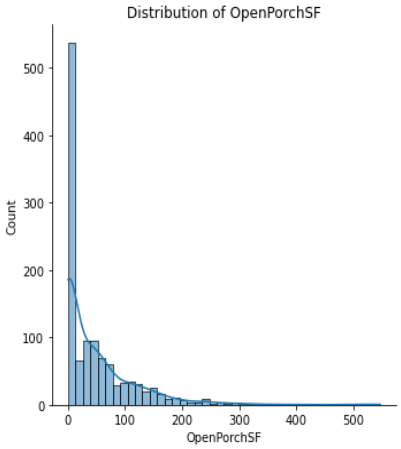

BoxPlot of GarageCars
Values
GarageCars

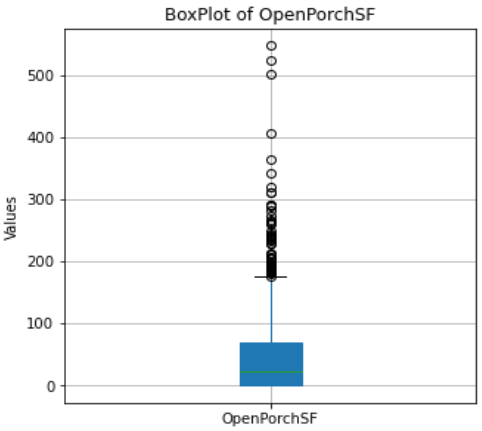

Distribution of GarageArea
Count
GarageArea

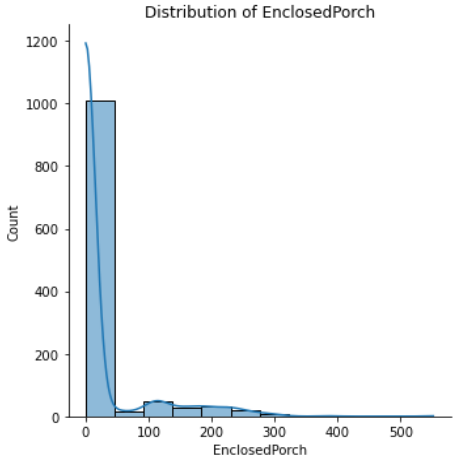

BoxPlot of GarageArea
Values
GarageArea

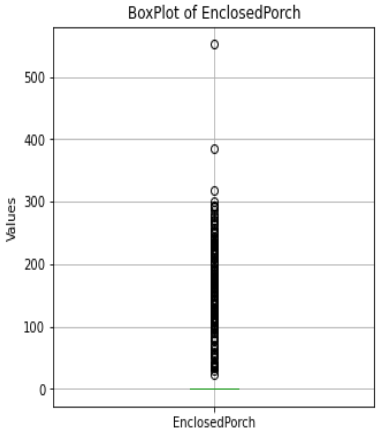

Distribution of WoodDeckSF
Count
WoodDeckSF

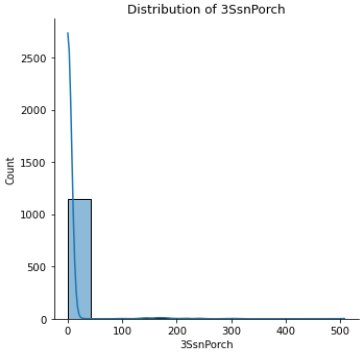

BoxPlot of WoodDeckSF
Values
WoodDeckSF

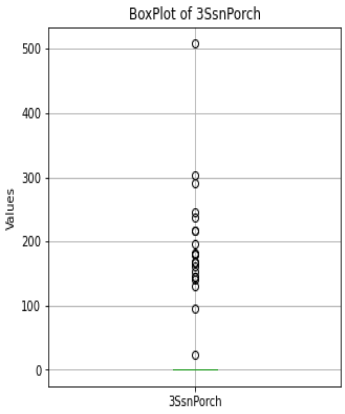

Distribution of OpenPorchSF
Count
OpenPorchSF
0
100
200
300
400
500

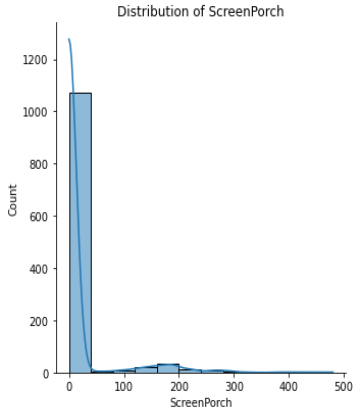

BoxPlot of OpenPorchSF
Values
OpenPorchSF
0
100
200
300
400
500

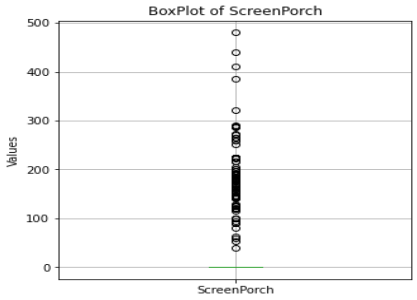

Distribution of EnclosedPorch
Count
EnclosedPorch
0
200
400
600
800
1000
1200
0
100
200
300
400
500

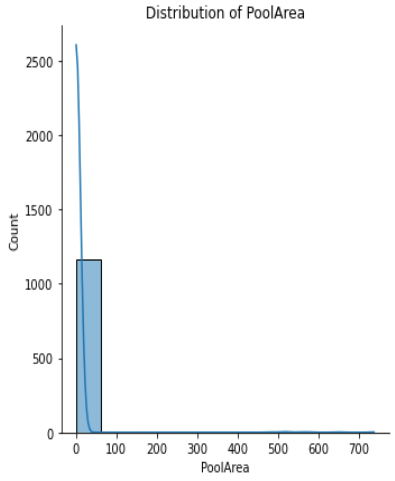

BoxPlot of EnclosedPorch
Values
EnclosedPorch
0
100
200
300
400
500

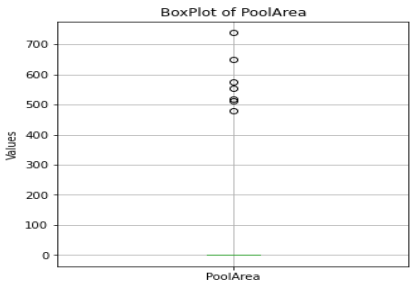

Distribution of 3SsnPorch
Count
3SsnPorch
0
500
1000
1500
2000
2500
0
100
200
300
400
500

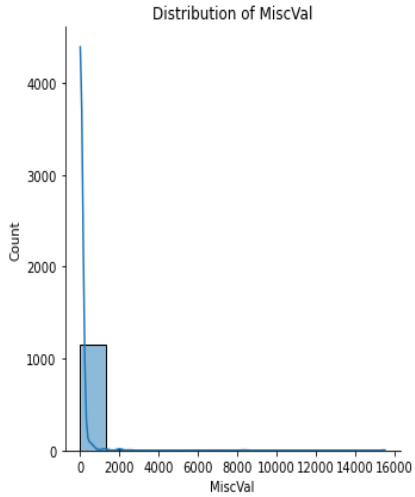

BoxPlot of 3SsnPorch
Values
3SsnPorch
0
100
200
300
400
500

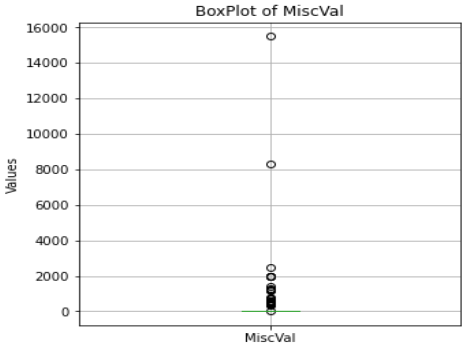

Distribution of ScreenPorch
Count
ScreenPorch
0
100
200
300
400
500
0
200
400
600
800
1000
1200

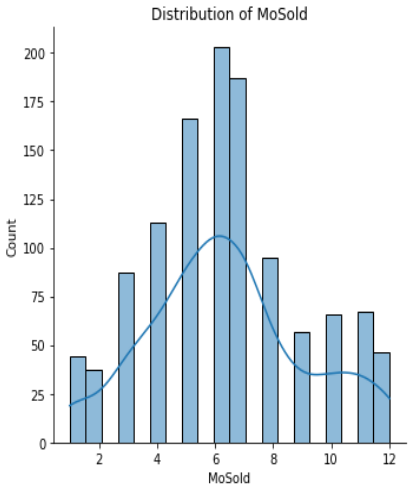

BoxPlot of ScreenPorch
Values
ScreenPorch
0
100
200
300
400
500

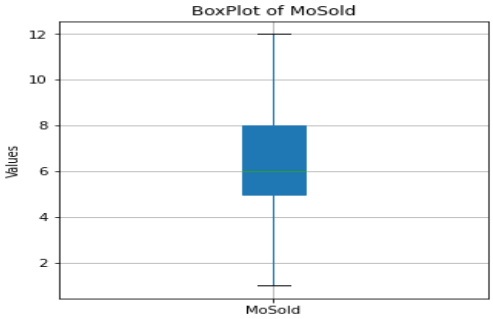

Distribution of PoolArea
Count
PoolArea
0
100
200
300
400
500
600
700
0
500
1000
1500
2000
2500

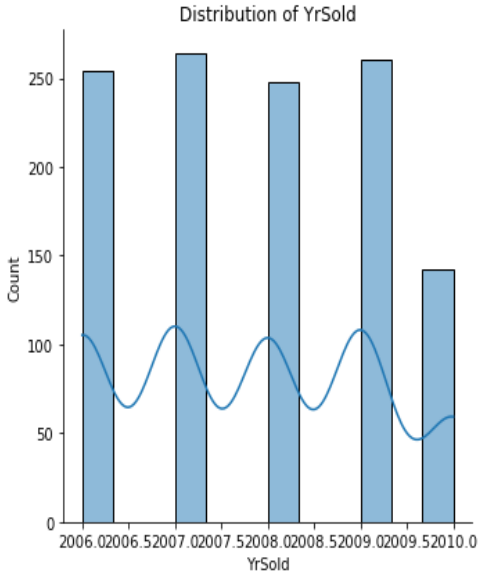

BoxPlot of PoolArea
Values
PoolArea
0
100
200
300
400
500
600
700

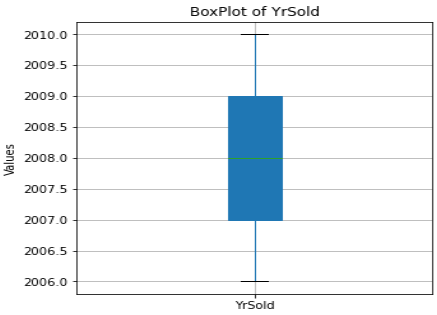

Distribution of MiscVal
Count
MiscVal
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1000
2000
3000
4000


BoxPlot of MiscVal
Values
MiscVal
0
2000
4000
6000
8000
10000
12000
14000
16000









Observation:

ThushHere we defined a function for describe,distribution plot and boxplot for non object data of train set and calling the same at one place. We observe the following:

1.MsSubClass:It is not normal distributed and presence of outliers.

2.LotFrontage:It is almost normally distributed and there is presence of outliers.Almost all houses were having Lotfrontage between approx 20 to 150.

3.LotArea:This is rightly skewed and having presence of outlier.Around 140 houses having LotArea between 0-20000 sqft.Few houses have lotarea above 40000 sqft.

4.OverallQual:Rates the overall material and finish of the house:Around 300 houses were sold in average condition. Only 20-30 houses were in excellent condition.The is normal distribution and one outliers can be seen and can be avoided as its near to whiskers.

5.OverallCond:Rates the overall condition of the house:Aroung 650 houses were sold in average condition overall.It shows normal distribution curve and presence of outliers is there.

6.YearBuilt:Original construction date: More number of people have brought the houses build after 2000.It is not normally distributed and having outliers.

7.YearRemodeAdd:Remodel date (same as construction date if no remodeling or additions):Not normally distributed and no outliers present. It showing houses have been remodel highest in the year 2005-2010.

8.MasVnrArea:Masonry veneer area in square feet-:Most of houses have Masonry veneer area as 0-100 and rest houses have Masonry veneer area 100-1000.It is rightly skewed and having outliers.

9.BsmtFinSF1:Type 1 finished square feet-:MOst houses have Type 1 finished square feet area of basement between 0 and 1500.It is not normally distributed and having outliers.

10.BsmtFinSF2: Type 2 finished square feet-:Around 1000 houses have Type 2 finished square feet area of 0 -100.Not normally distributed and have lots of outliers.

11.BsmtUnfSF: Unfinished square feet of basement area-:Around 120 houses have unfinished basesent of area around 200-500 sqft. Not normally distributed and having outliers.

12.1stFlrSF: First Floor square feet-:Around 140 houses have 1st floor square feet area between 900-1200sqft.Almost normally distributed and having outliers.

13.GrLivArea: Above grade (ground) living area square feet-:Most houses have above ground living sq ft area in between 800 to 3000.Almost normal distribution and hvaing outliers.

14.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

15.FullBath: Full bathrooms above grade-:42% houses have 1 full bathrooms above ground and 50% have 2 full bathrooms located above ground and very less have 3 and no full bathroom.Not normally distributed and no outliers.

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

17.BedroomAvgr: Bedrooms above grade (does NOT include basement bedrooms)-:Most houses have 3 bedrooms above ground followed by 2 and 4.Not normally distributed and having outliers.

18.Kitchen: Kitchens above grade-:Maximum houses have 1 Kitchen .very few have 2.Not normally distributed and having outliers.

19.TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)-:Around 350 houses have 6 rooms ,around 200 have 5,& 250 have 7. Very few have 12 & 14 rooms.Very near to normal distribution and having outliers.

20.Fireplaces: Number of fireplaces-:Most houses have 0 fireplaces followed by 1.Not normally distributed and having outliers.

21.Garageyrbuit:Most of the garages were built after 2000.Not normally distributed and having outliers.

22.GarageCars: Size of garage in car capacity-:Most houses have garage with 2 car capacity followed by 1 car capacity.Not normally distributed and having outlier.

23.GarageArea: Size of garage in square feet-:Most houses have Garage area in between 400 to 800.Nearly normal distributed and having outliers.

24.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.Not normally distributed and having outliers.

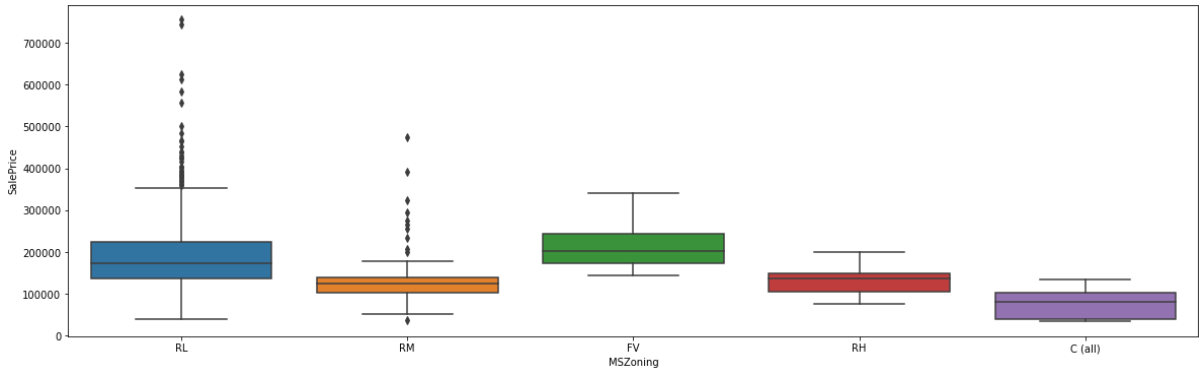
25.OpenPorchSF: Open porch area in square feet-:Most of houses have 0 open porch sq ft area and rest have in between 0 to 300

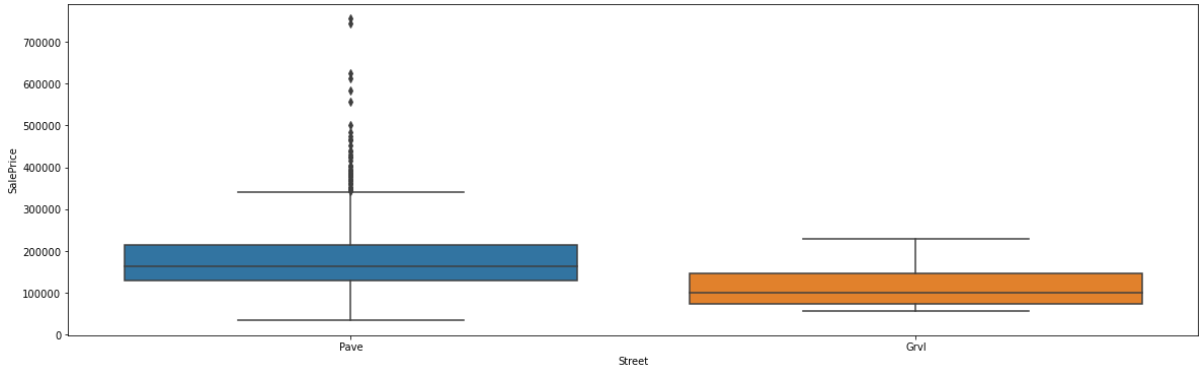
26.EnclosedPorch: Enclosed porch area in square feet-:Almost all houses have 0 enclosed porch sq ft area.Not normally distributted pattern and having outliers.

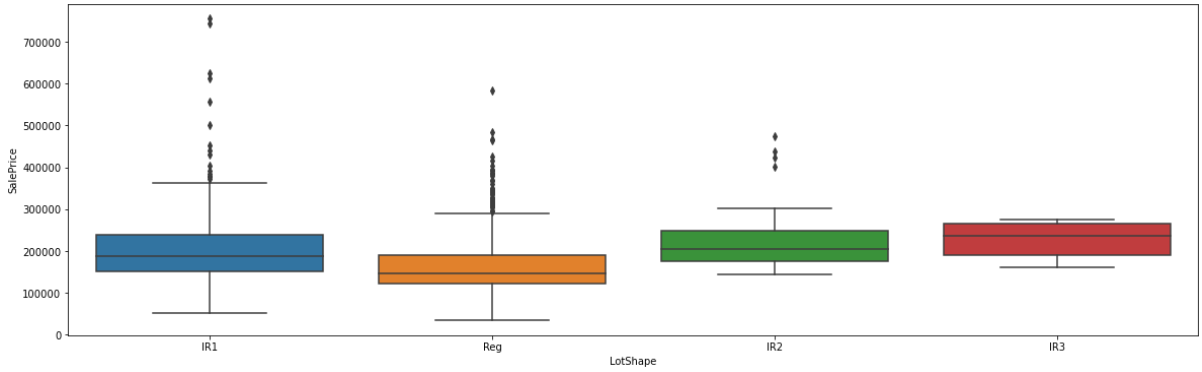
27.ScreenPorch: Screen porch area in square feet-:Almost all houses have 0 screen porch area sq ft.Not normally distributed and having outliers.

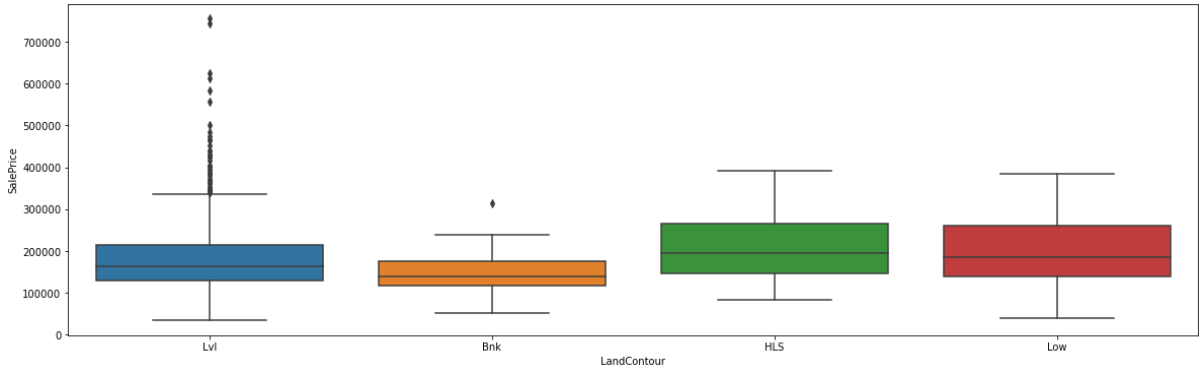
28.Sale Price-:Around 160 house have sale price in between 100000 to 300000.Very few houses have sale price of 600000 & 700000

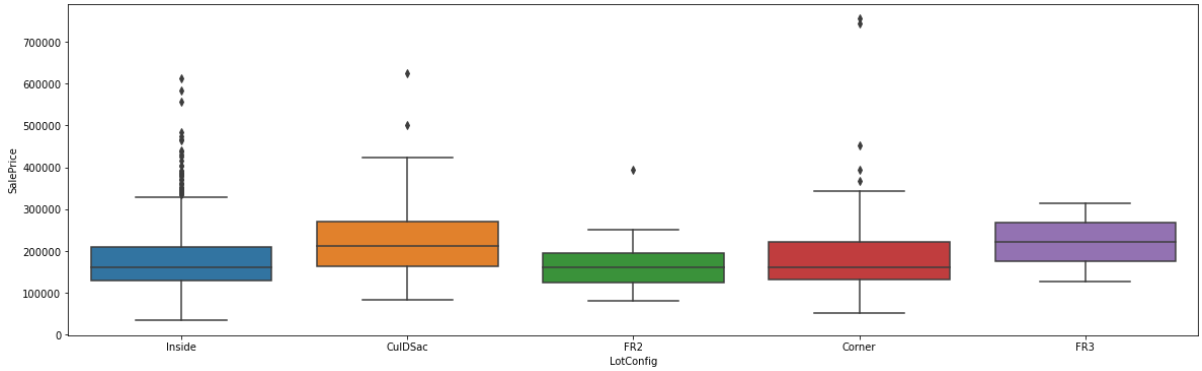
Bivariate Analysis of features with respect to SalePrice as our Target Variable

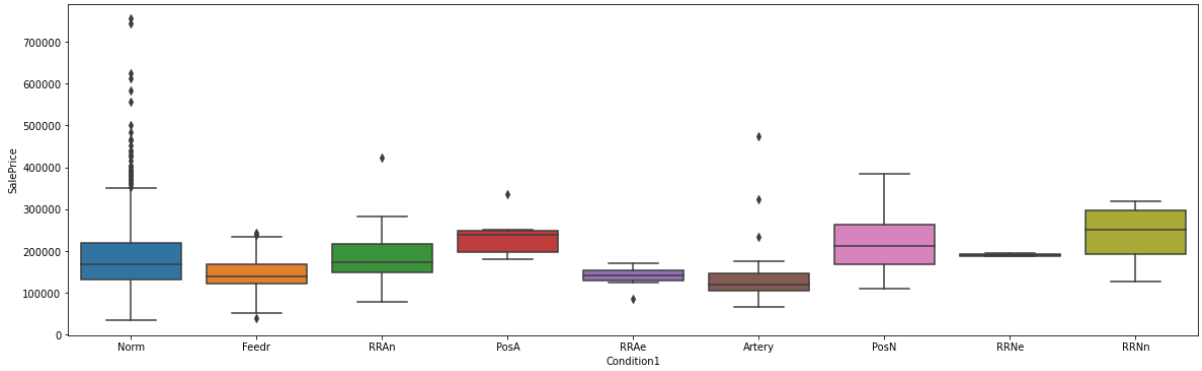


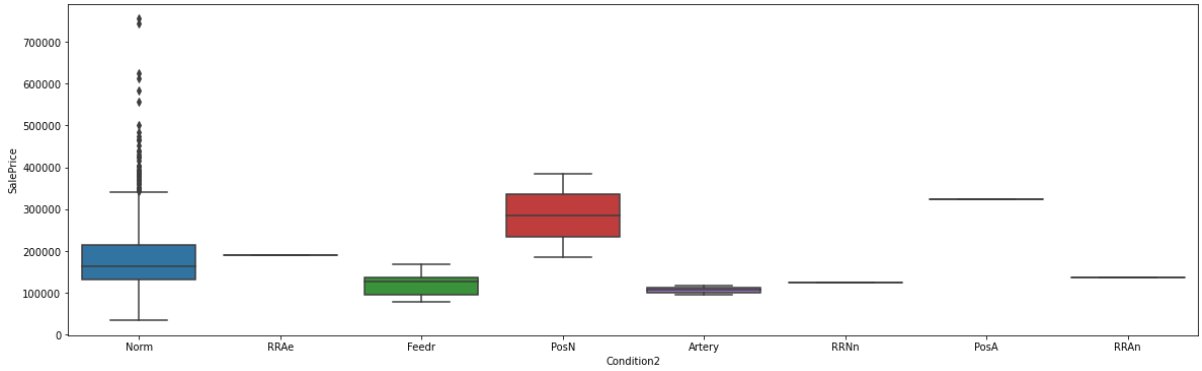

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Pave
Grvl
Street

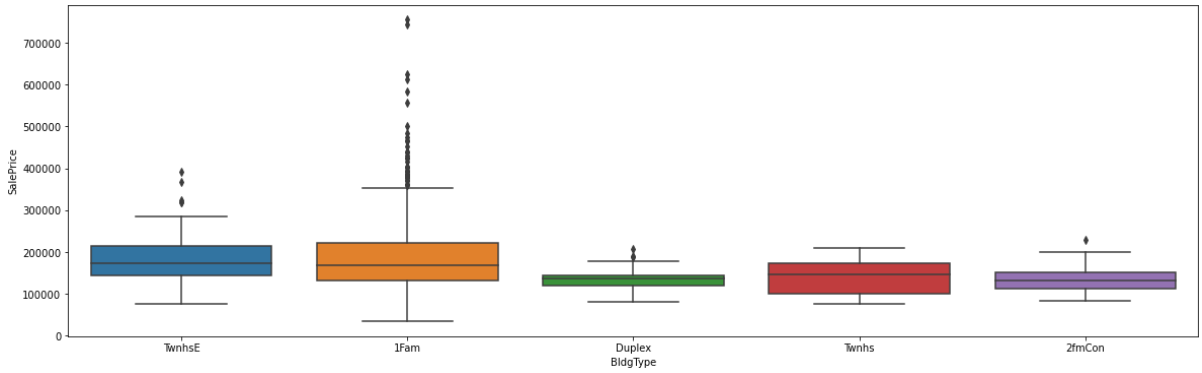

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
IR1
Reg
IR2
IR3
LotShape

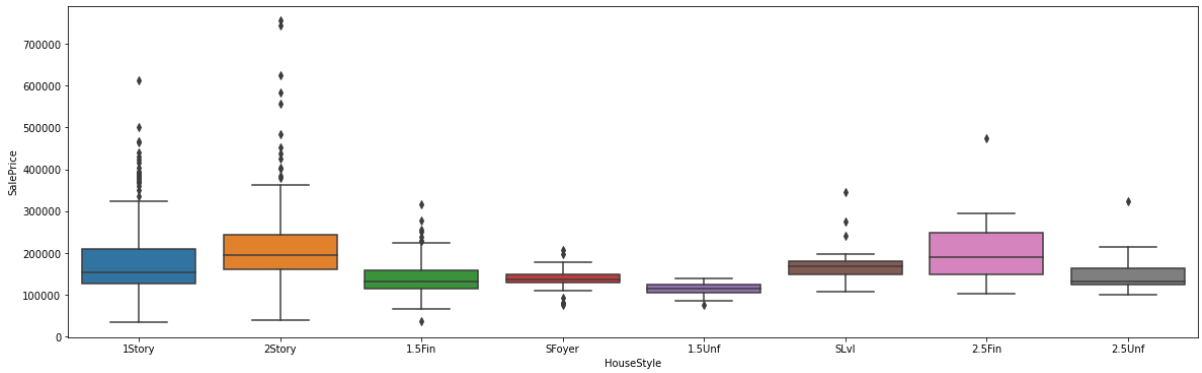

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Lvl
Bnk
HLS
Low
LandContour

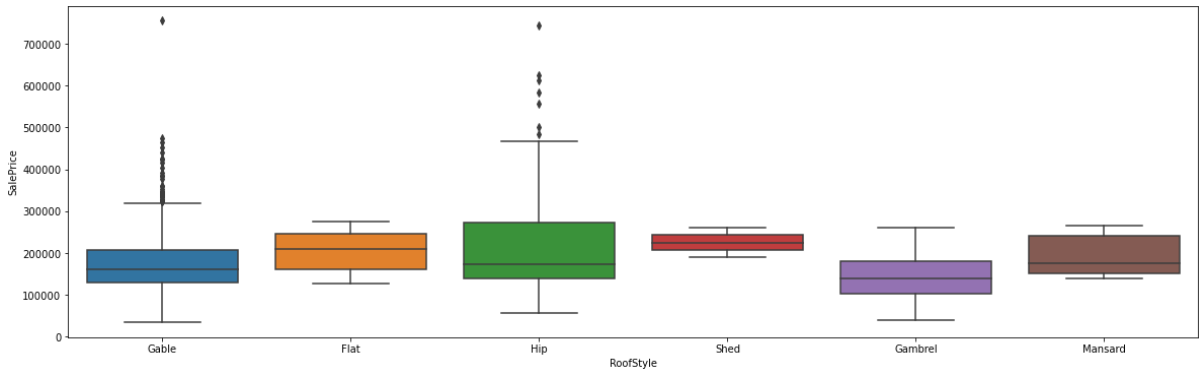

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Inside
CulDSac
FR2
Corner
FR3
LotConfig

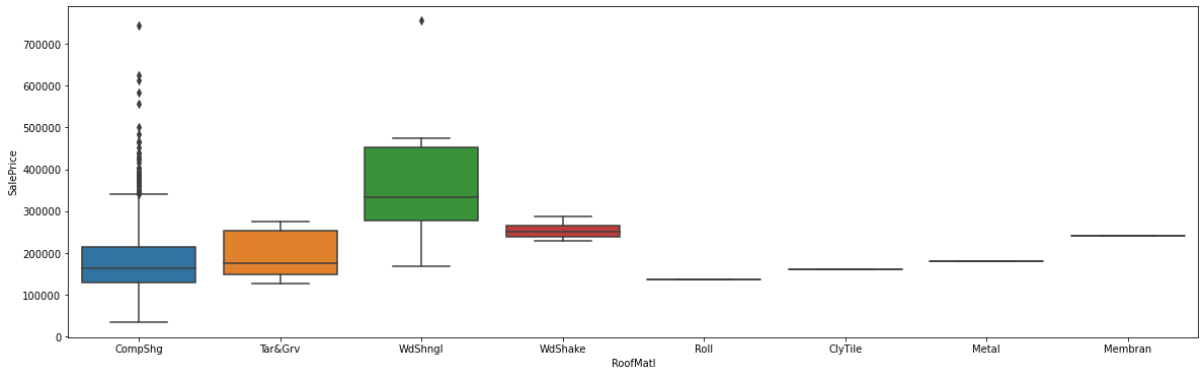

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1

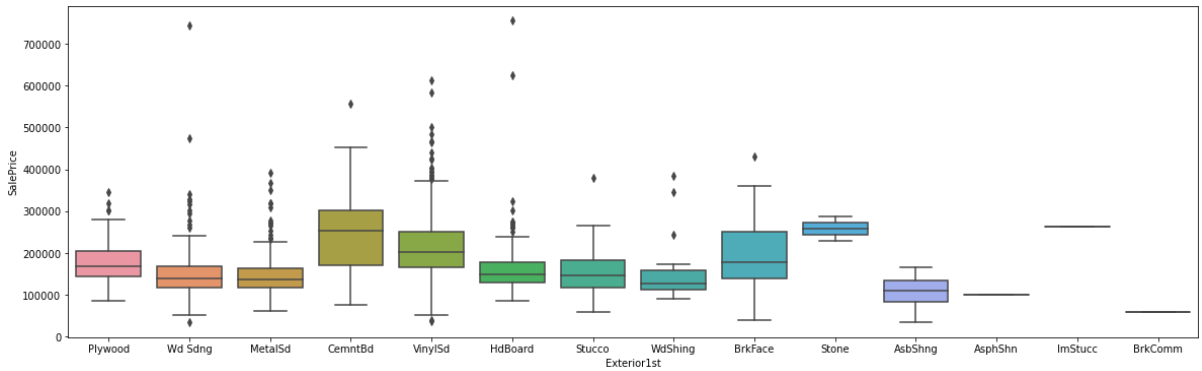

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
Condition2

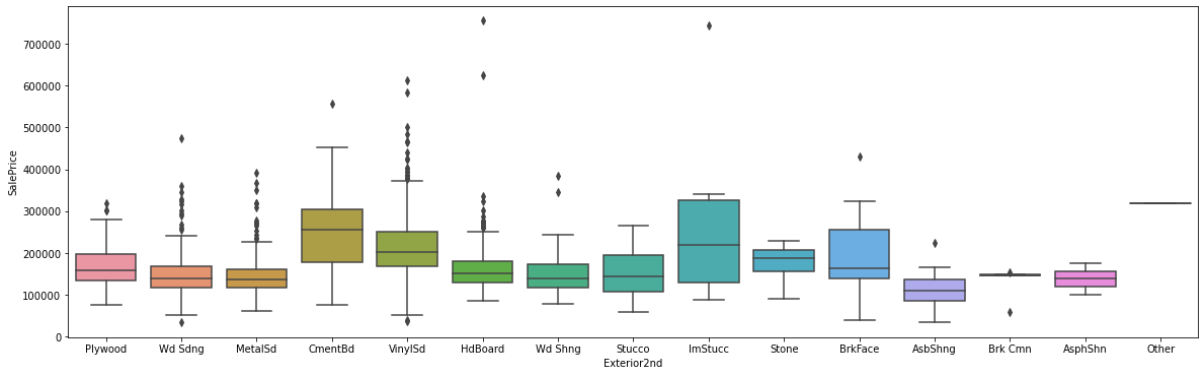

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType

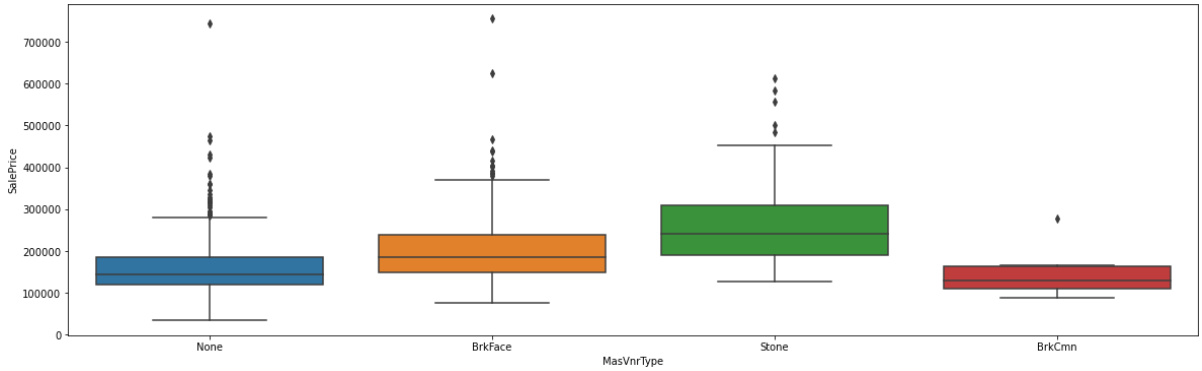

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle

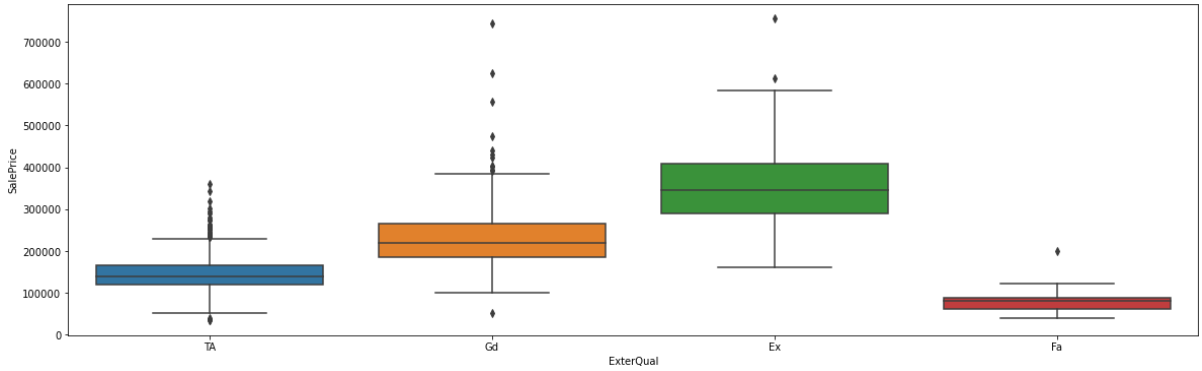

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofStyle

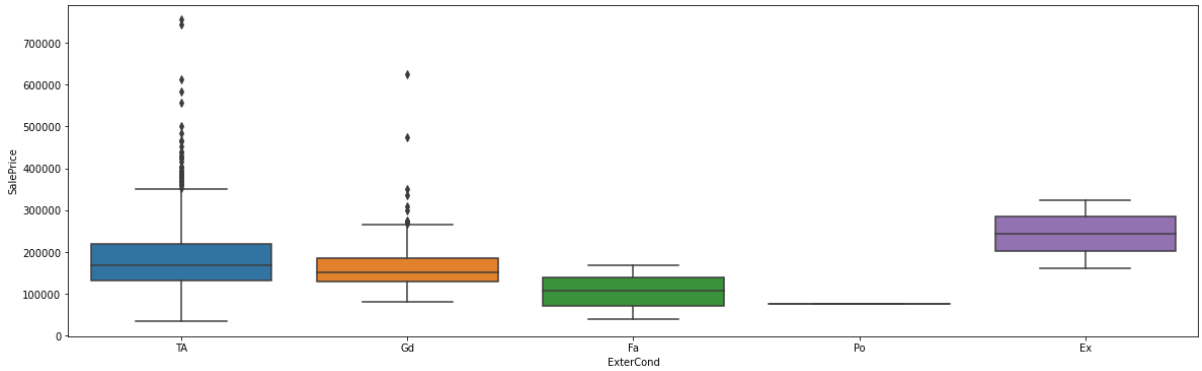

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
CompShg
Tar&Grv
WdShngl
WdShake
Roll
ClyTile
Metal
Membran
RoofMatl

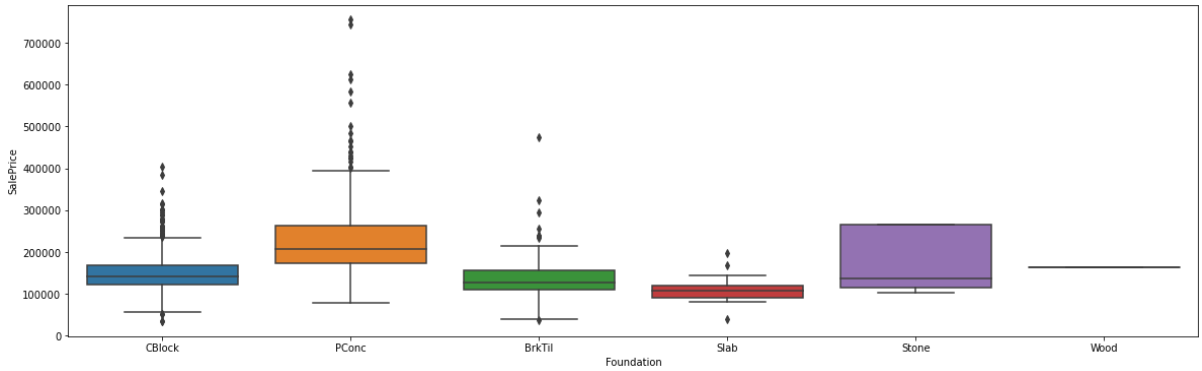

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Plywood
Wd Sdng
MetalSd
CemntBd
VinylSd
HdBoard
Stucco
WdShing
BrkFace
Stone
AsbShng
AsphShn
ImStucc
BrkComm
Exterior1st

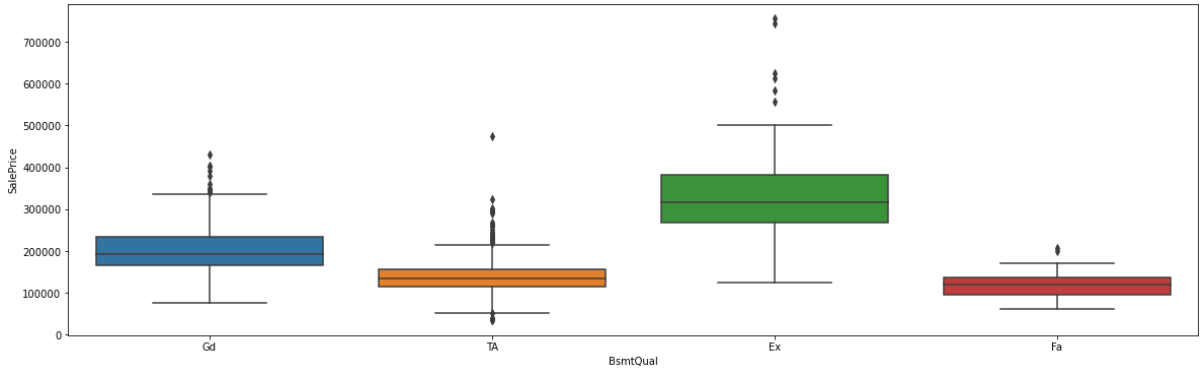

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Plywood
Wd Sdng
MetalSd
CmentBd
VinylSd
HdBoard
Wd Shng
Stucco
ImStucc
Stone
BrkFace
AsbShng
Brk Cmn
AsphShn
Other
Exterior2nd

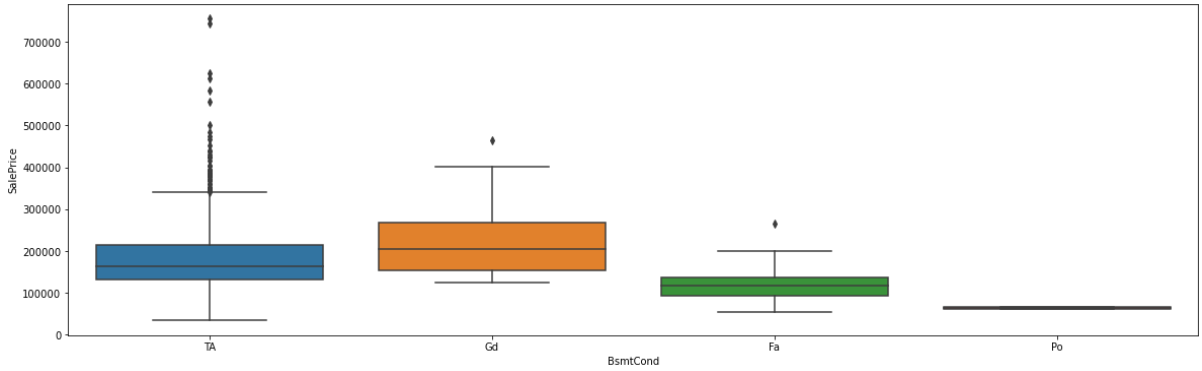

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
None
BrkFace
Stone
BrkCmn
MasVnrType

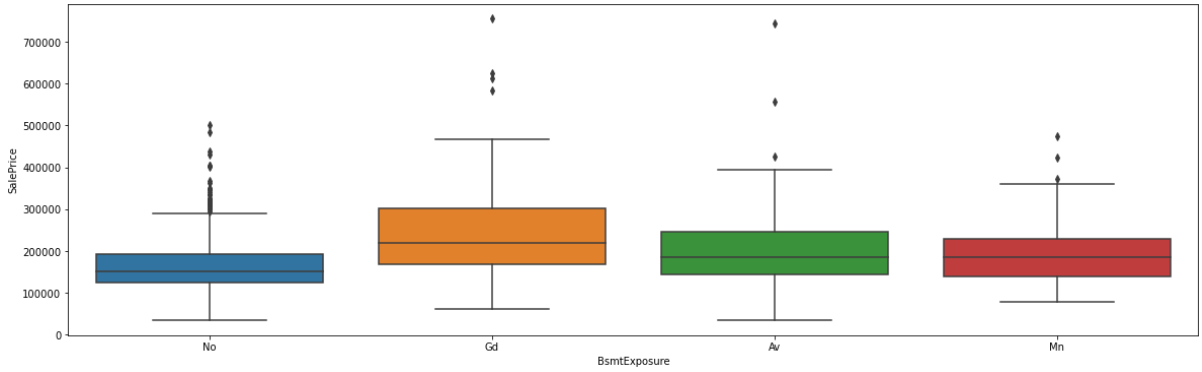

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Ex
Fa
ExterQual

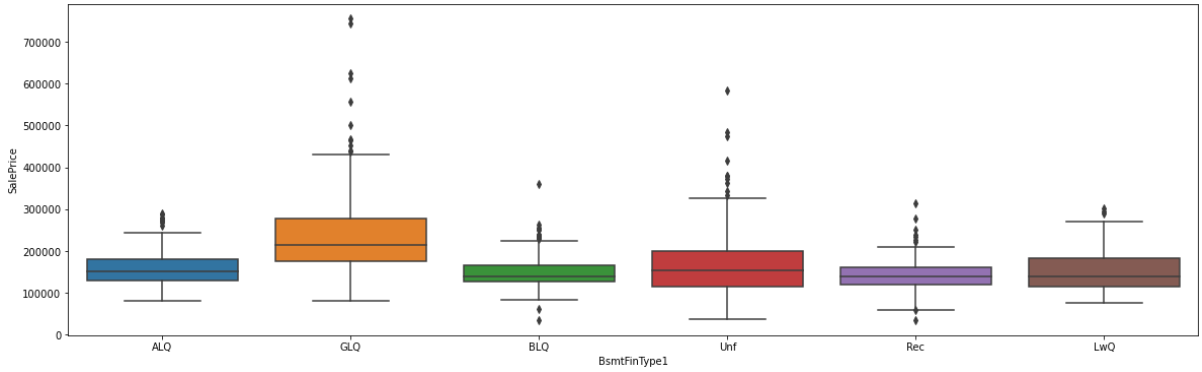

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
Po
Ex
ExterCond

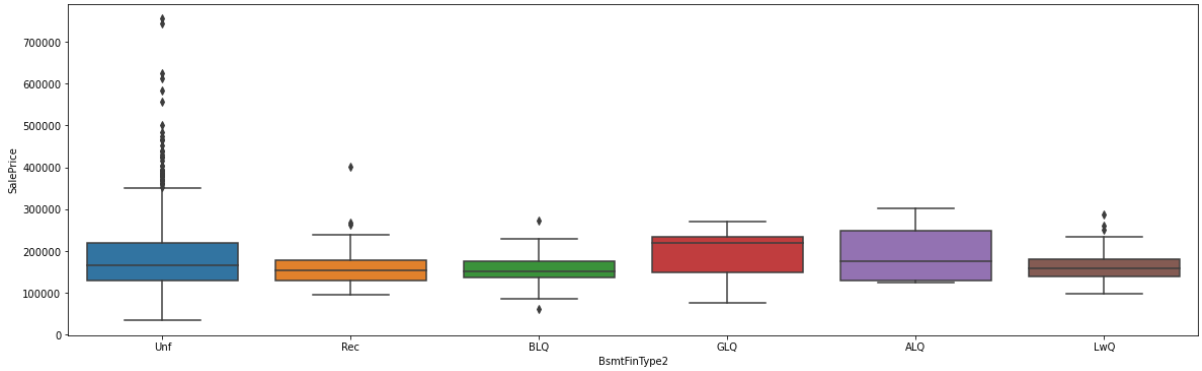

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
CBlock
PConc
BrkTil
Slab
Stone
Wood
Foundation

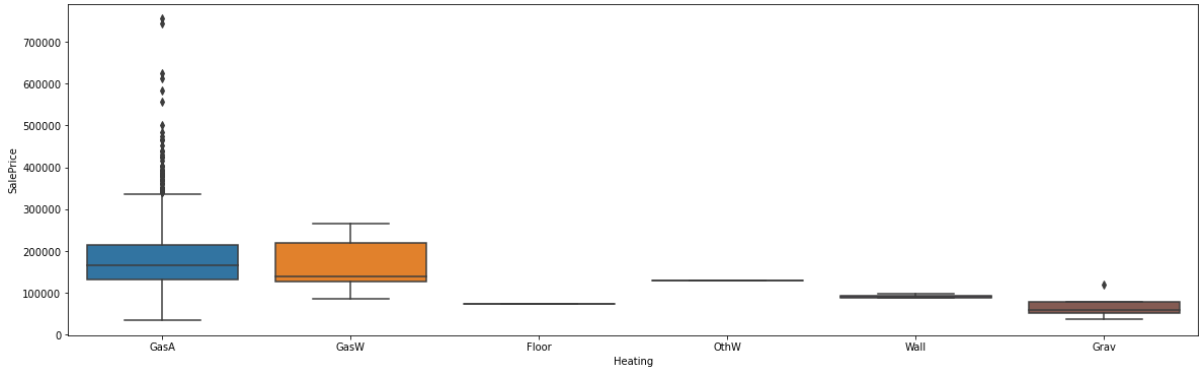

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Gd
TA
Ex
Fa
BsmtQual

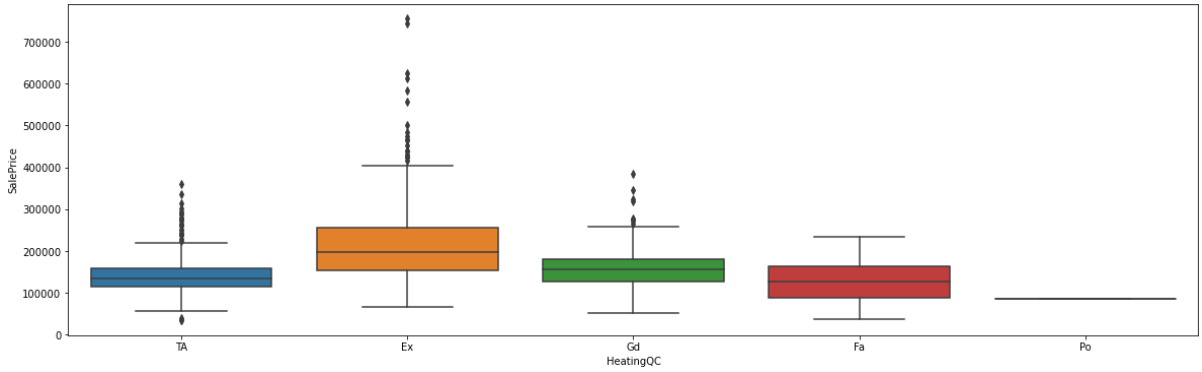

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
Po
BsmtCond

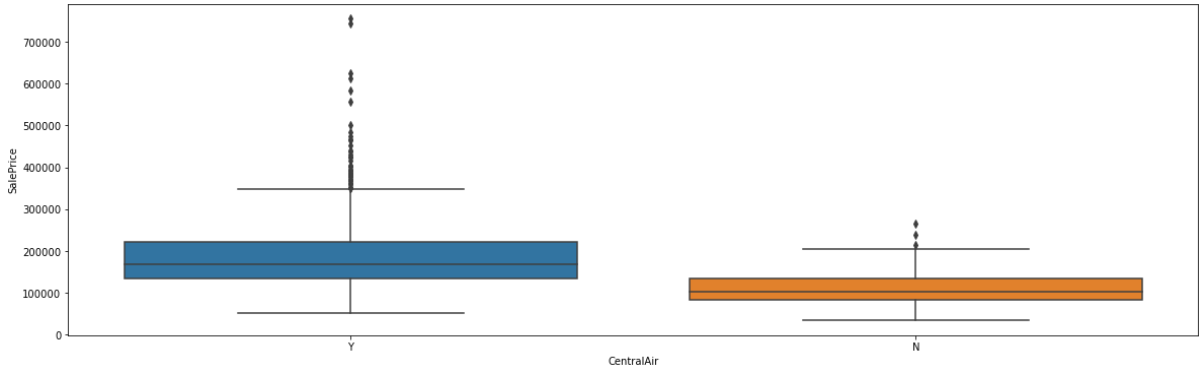

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
No
Gd
Av
Mn
BsmtExposure

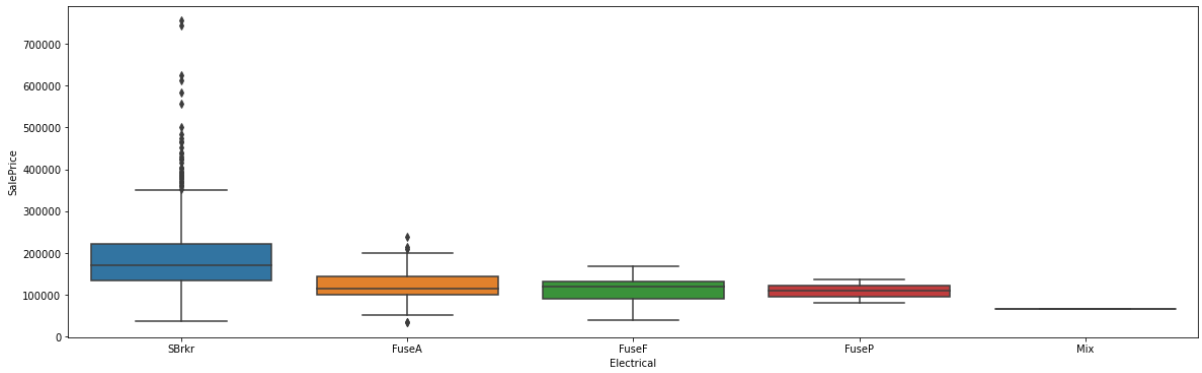

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
ALQ
GLQ
BLQ
Unf
Rec
LwQ
BsmtFinType1

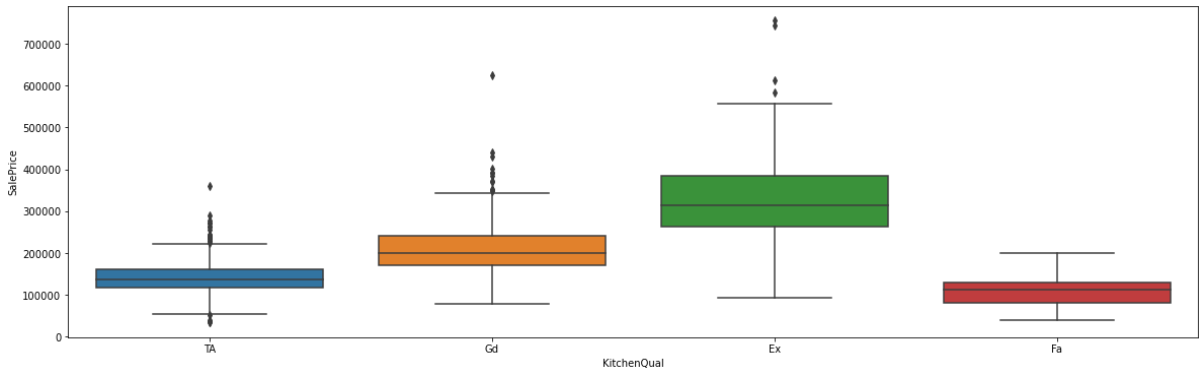

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Unf
Rec
BLQ
GLQ
ALQ
LwQ
BsmtFinType2

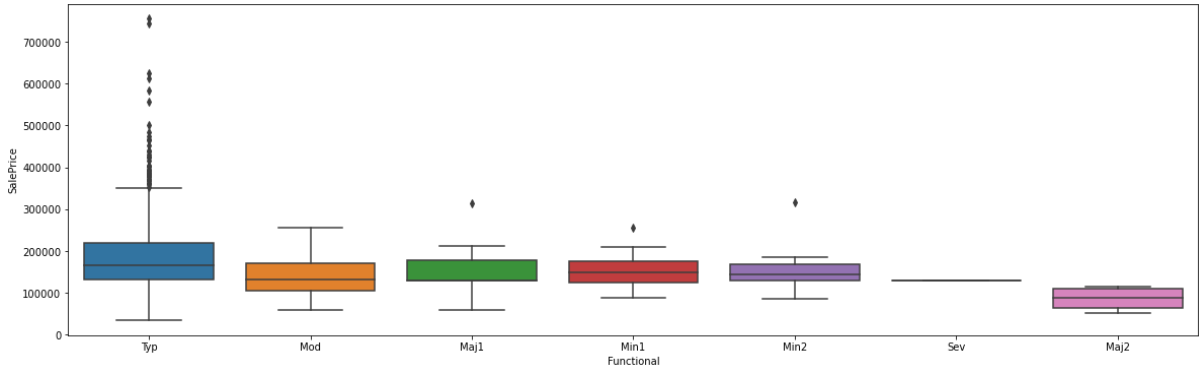

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
GasA
GasW
Floor
OthW
Wall
Grav
Heating

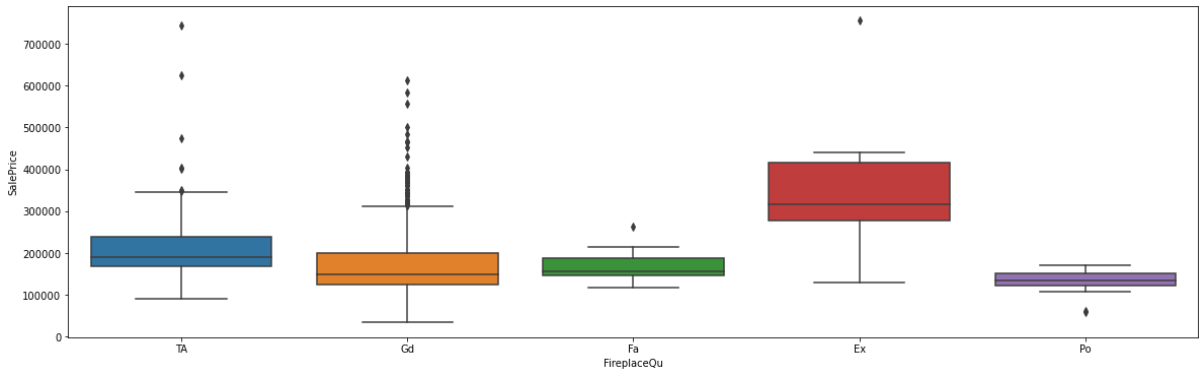

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Ex
Gd
Fa
Po
HeatingQC

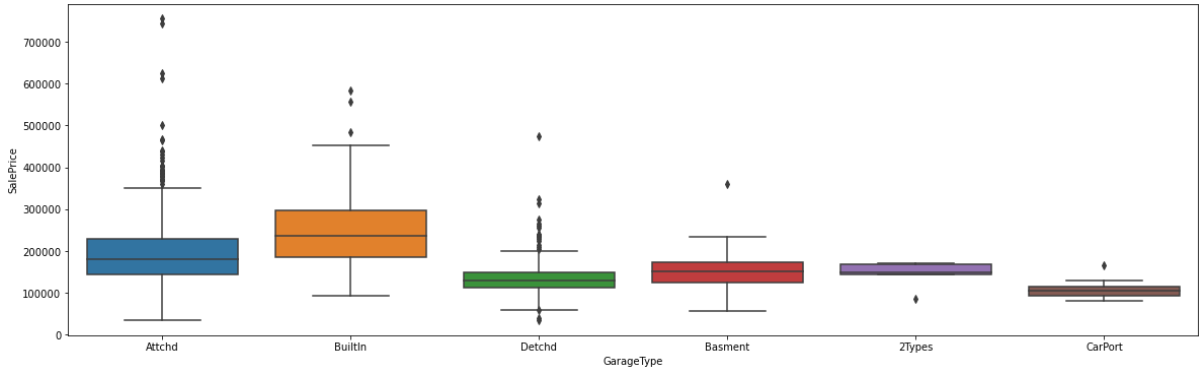

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Y
N
CentralAir

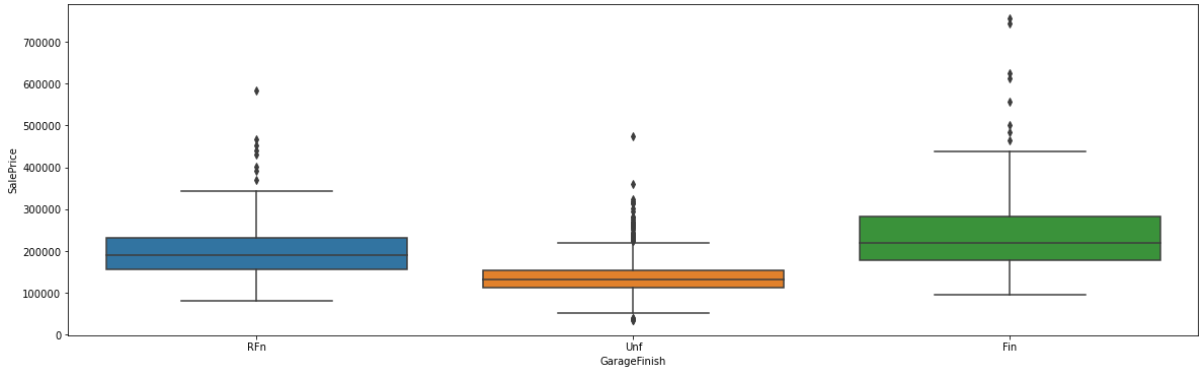

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

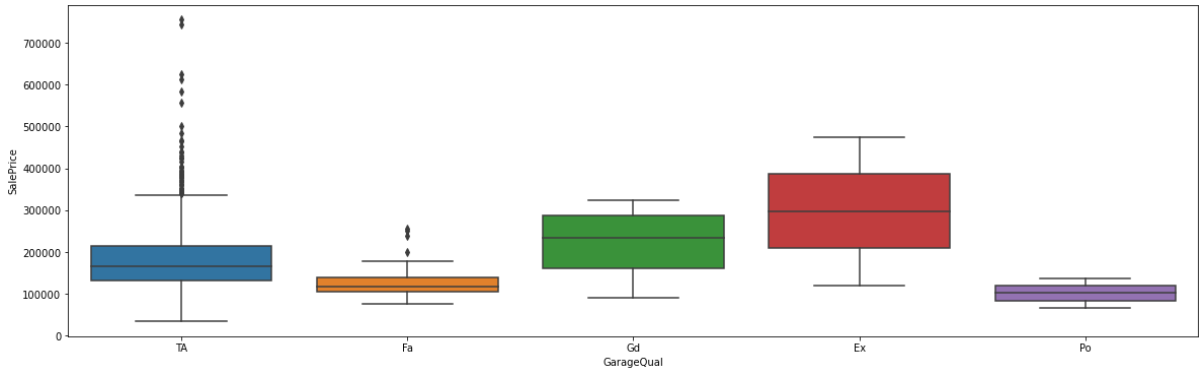

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Ex
Fa
KitchenQual

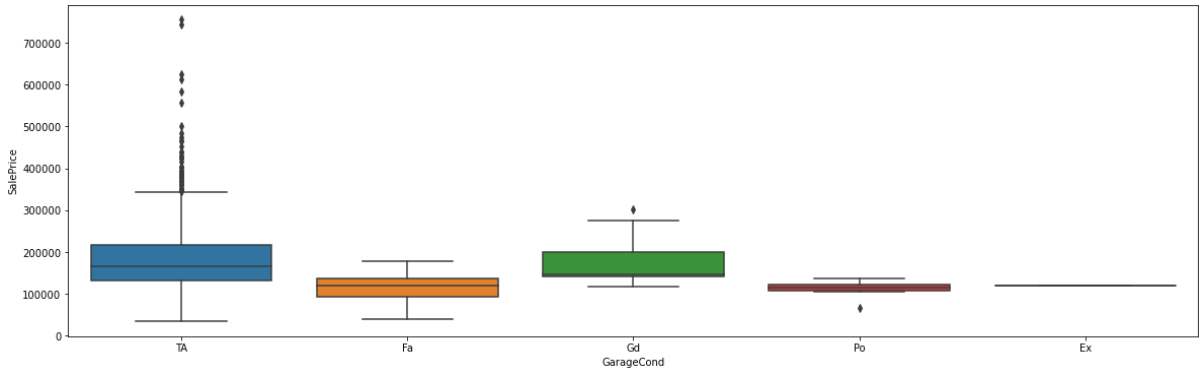

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
Functional

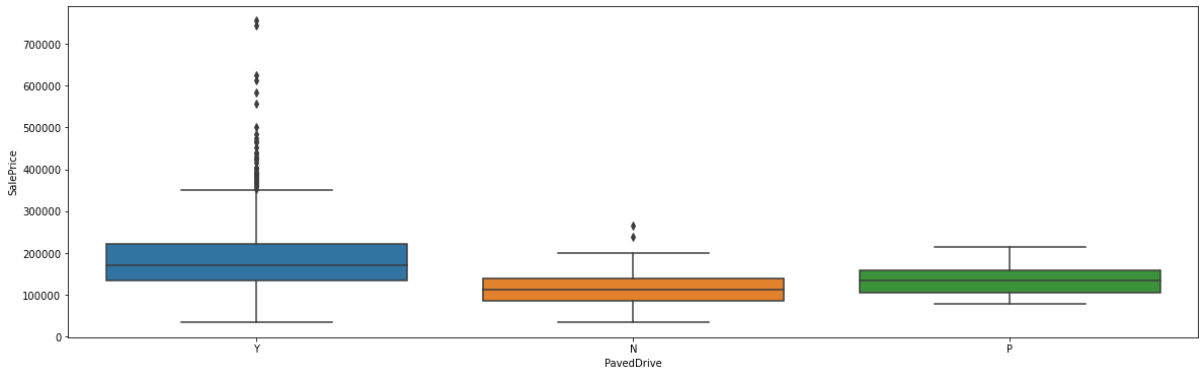

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Gd
Fa
Ex
Po
FireplaceQu

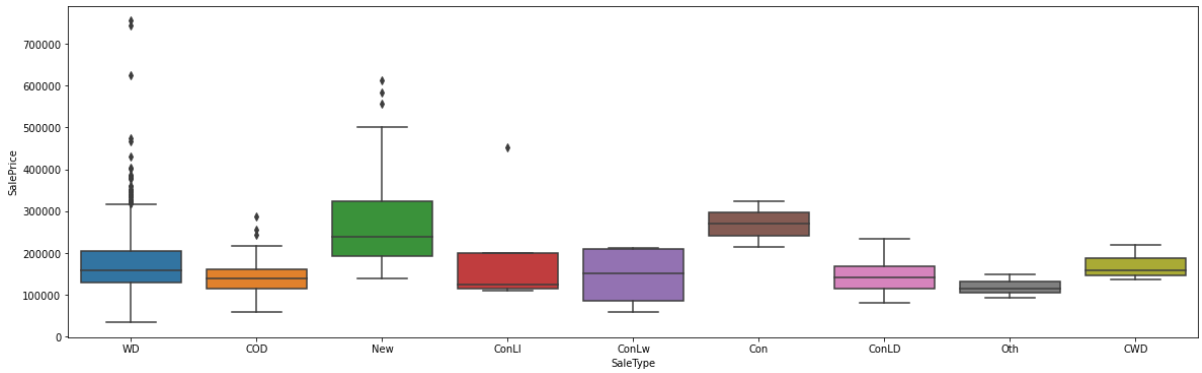

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort
GarageType

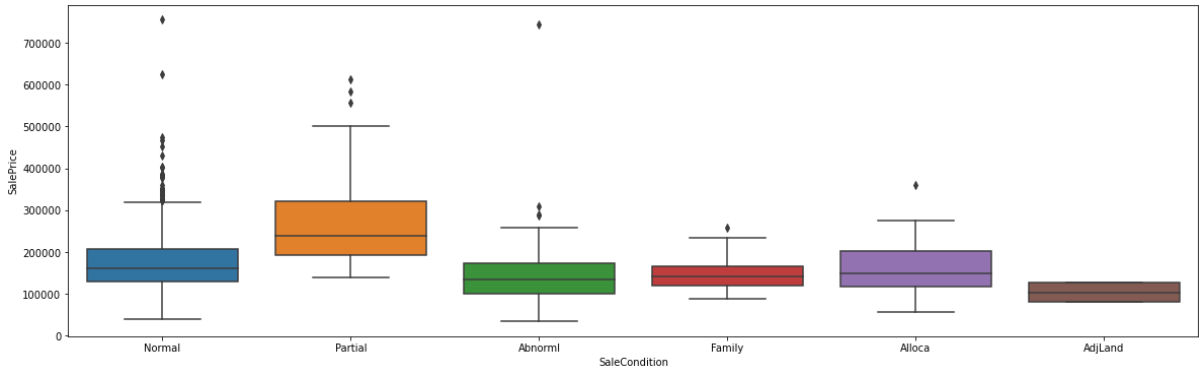

SalePrice
0
100000
200000
300000
400000
500000
600000
700000
RFn
Unf
Fin
GarageFinish


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Fa
Gd
Ex
Po
GarageQual


SalePrice
0
100000
200000
300000
400000
500000
600000
700000
TA
Fa
Gd
Po
Ex
GarageCond







OBSERVATION:

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

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

3.LotShape:IR3(Irregular) have a bit higher price compared to other while Reg(Regular) have lowest avg sale price.

4.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.

5.LotConfig:Lot configuration:Cul-de-sac have and FR3(Frontage on 3 sides of property) having high average sales price and FR2 and Corner having lowest.

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) and MeadowV having lowest.

8.Condition1:house that is RRNn(Within 200' of 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 2Story(Two story) is maximum followed by 2.5F(Two and one-half story). 1.5Unf(One and one-half story: 2nd level unfinished) have lowest avg selling price.

11.RoofStyle: Type of Roof: Houses with roofstyle having shed having high average selling price and Gambrel has lowest sale price.

12.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.

13.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

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

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

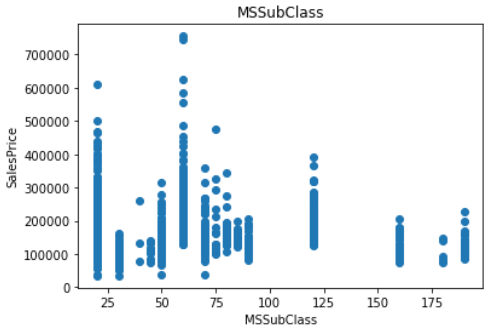
16.FirePlaceQual:Houses with Ex(Excellent) fireplace quality have higher sale price.

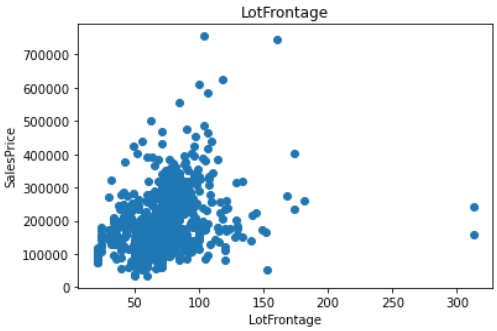
17.Garagetype:Builtin garage type having maximum average selling price and carport is having lowest.

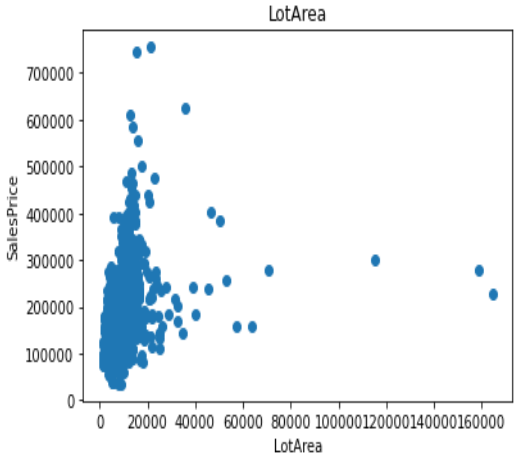
18.Saletype:Con (Contract 15% Down payment regular terms) having maximum average selling price and oth(other) is having lowest.

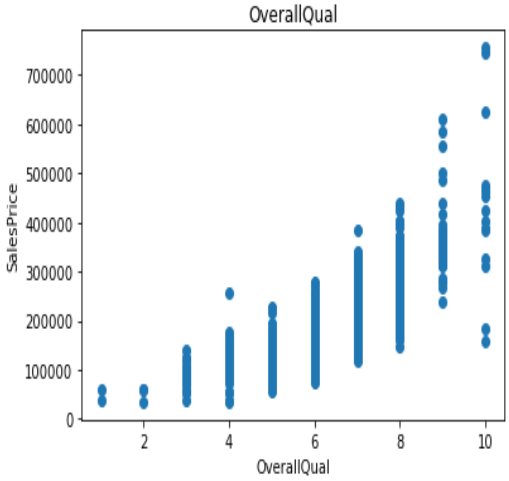
Scatter Plot (Sales price with respect to features)

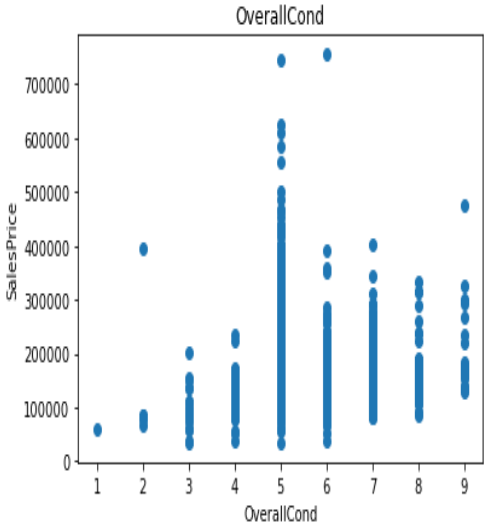
```
#Visualising sale price wrt to non_obj feature
for i in non_obj:
    plt.scatter(df_train[i],df_train['SalePrice'])
    plt.xlabel(i)
    plt.ylabel('SalesPrice')
    plt.title(i)
    plt.show()
```

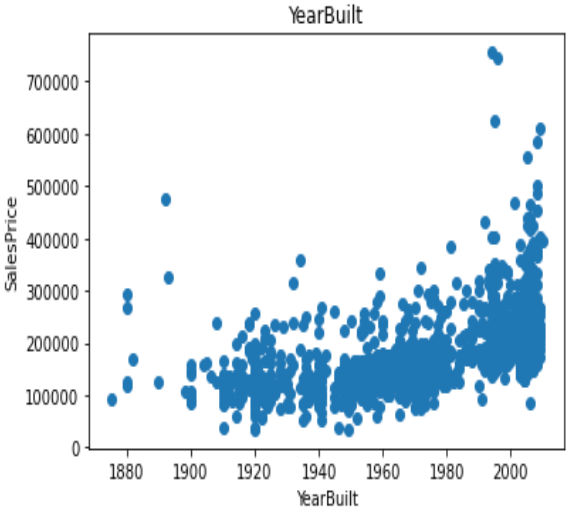

MSSubClass
SalesPrice
MSSubClass

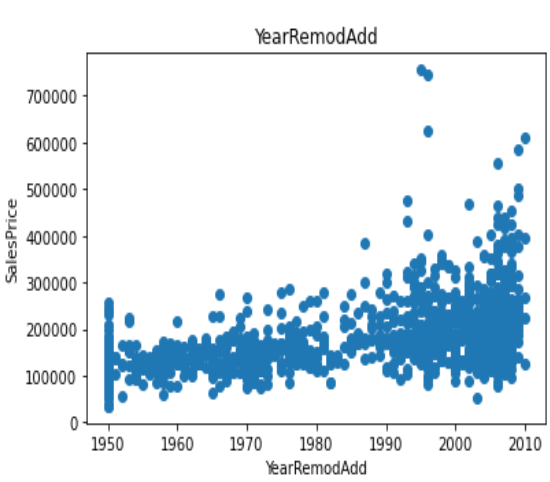

LotFrontage
SalesPrice
LotFrontage

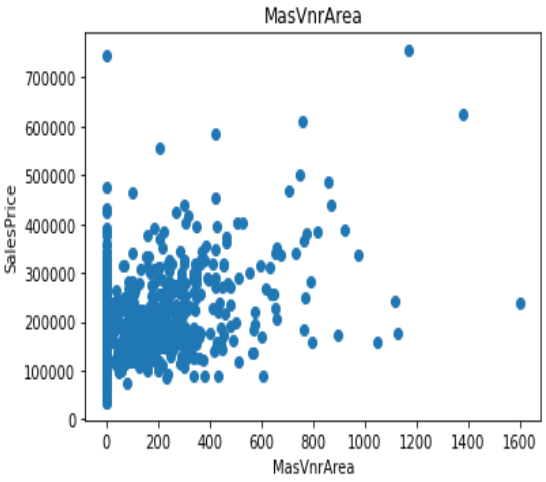

LotArea
SalesPrice
LotArea

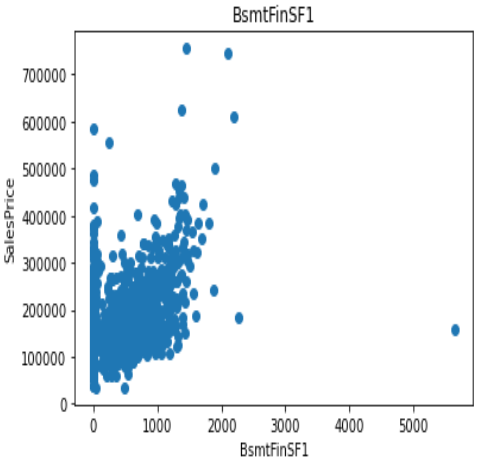

OverallQual
SalesPrice
OverallQual

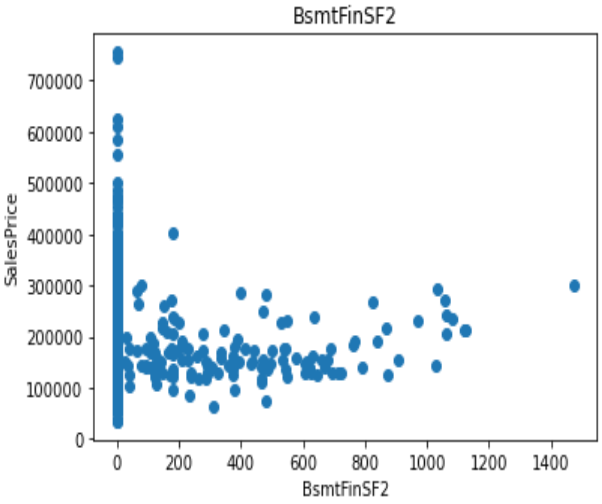

OverallCond
SalesPrice
OverallCond

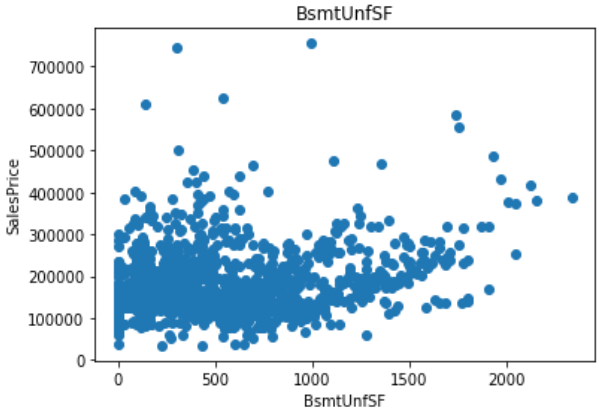

YearBuilt
SalesPrice
YearBuilt

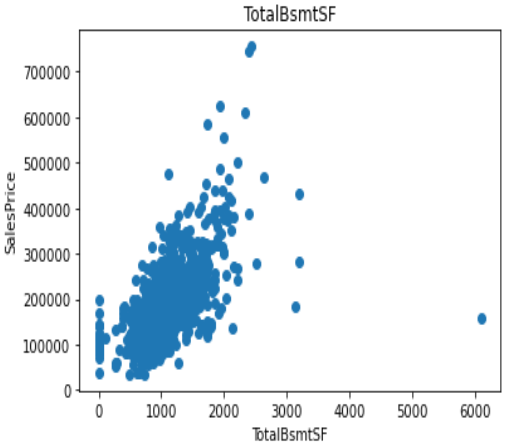

YearRemodAdd
SalesPrice
YearRemodAdd
0
100000
200000
300000
400000
500000
600000
700000
1950
1960
1970
1980
1990
2000
2010

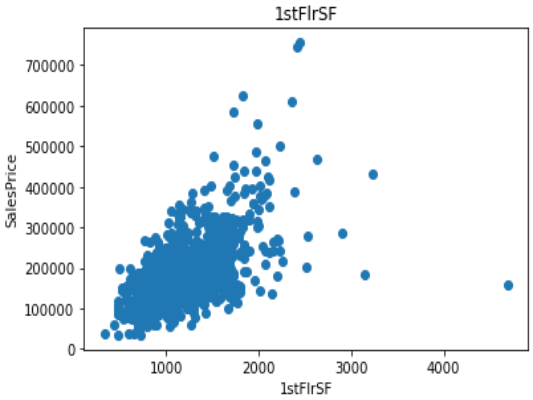

MasVnrArea
SalesPrice
MasVnrArea
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400
1600

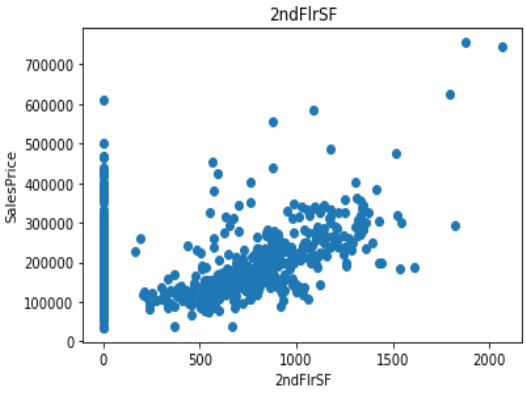

BsmtFinSF1
SalesPrice
BsmtFinSF1
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000

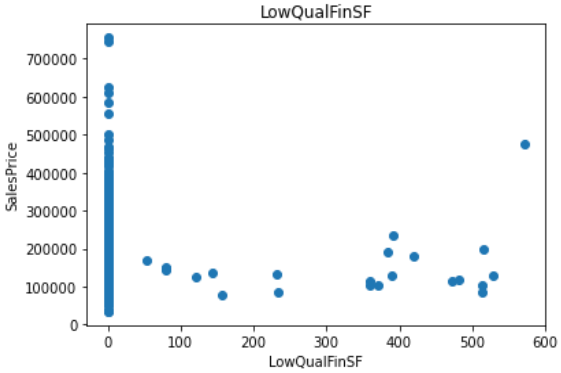

BsmtFinSF2
SalesPrice
BsmtFinSF2
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400

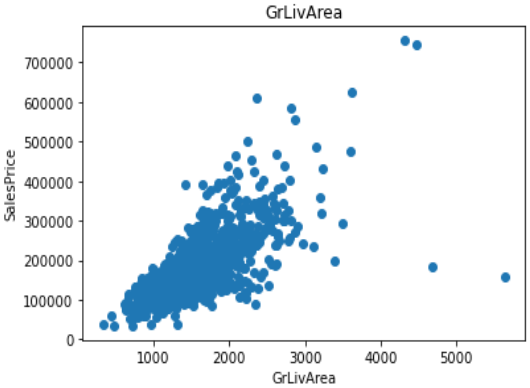

BsmtUnfSF
SalesPrice
BsmtUnfSF
0
100000
200000
300000
400000
500000
600000
700000
0
500
1000
1500
2000

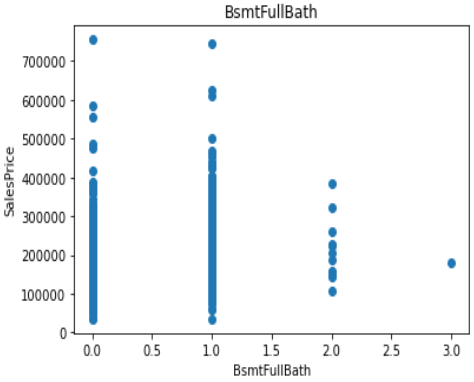

TotalBsmtSF
SalesPrice
TotalBsmtSF
0
100000
200000
300000
400000
500000
600000
700000
0
1000
2000
3000
4000
5000
6000

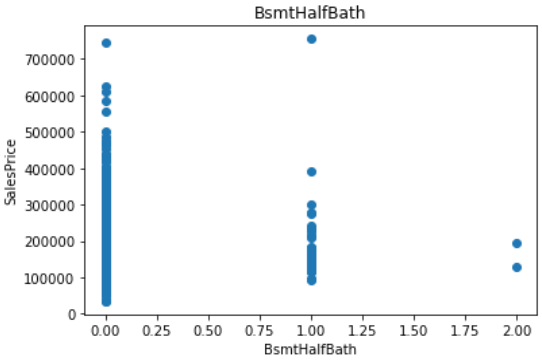

1stFlrSF
SalesPrice
1stFlrSF

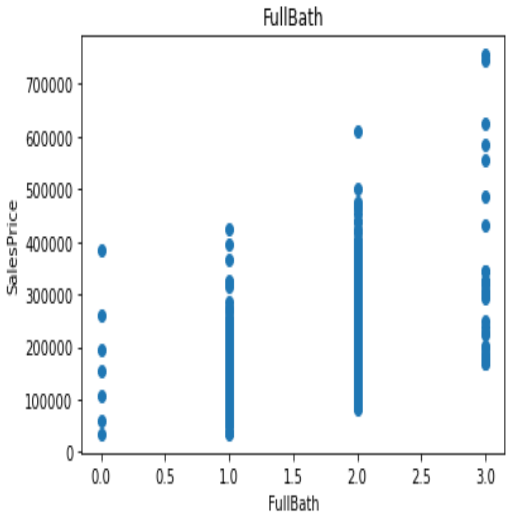

2ndFlrSF
SalesPrice
2ndFlrSF

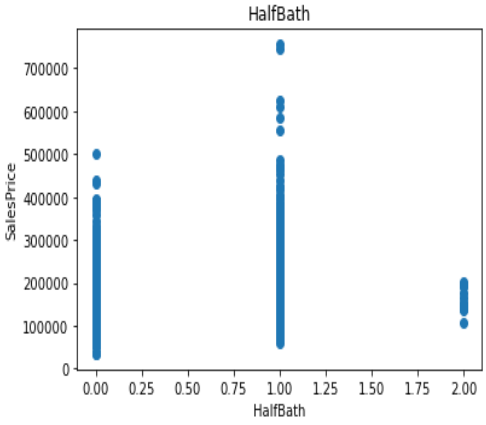

LowQualFinSF
SalesPrice
LowQualFinSF

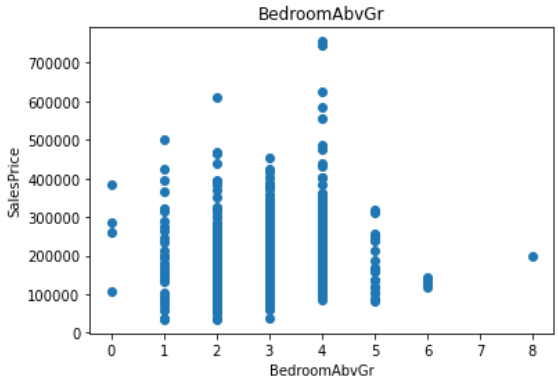

GrLivArea
SalesPrice
GrLivArea

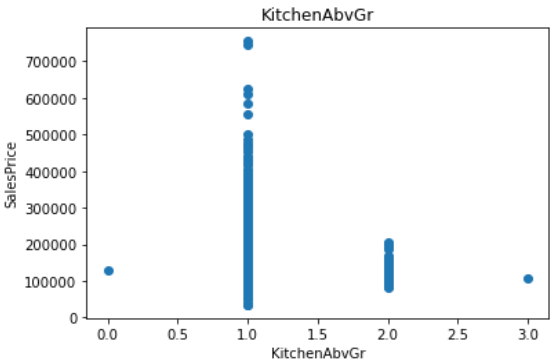

BsmtFullBath
SalesPrice
BsmtFullBath

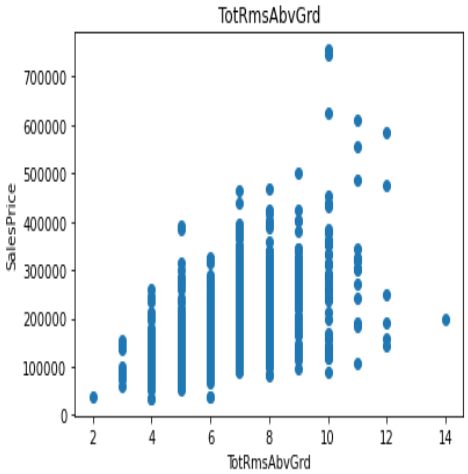

BsmtHalfBath
SalesPrice
BsmtHalfBath

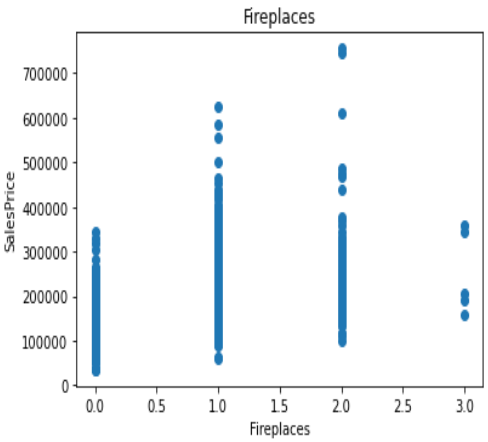

FullBath
SalesPrice
FullBath

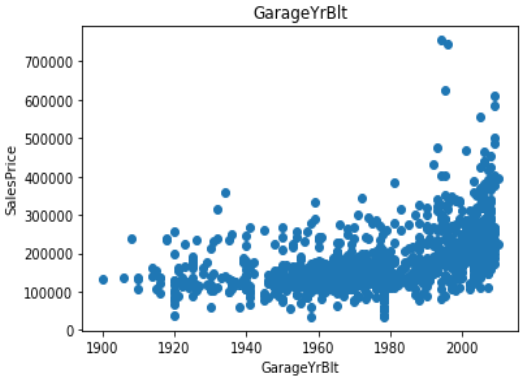

HalfBath
SalesPrice
HalfBath

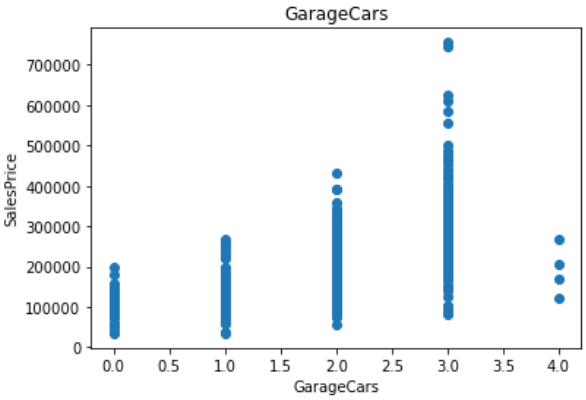

BedroomAbvGr
SalesPrice
BedroomAbvGr

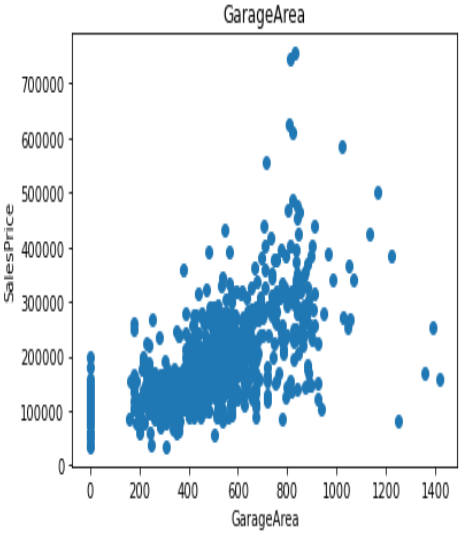

KitchenAbvGr
SalesPrice
KitchenAbvGr

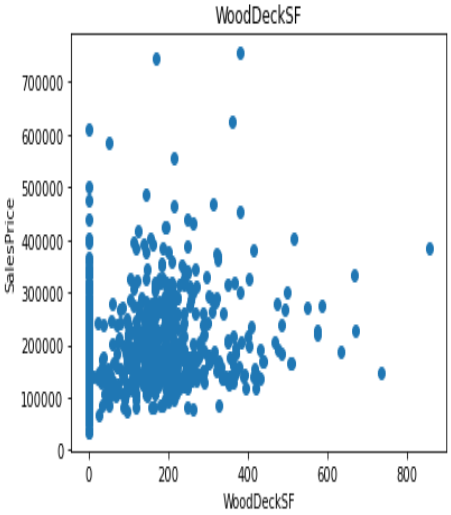

TotRmsAbvGrd
SalesPrice
TotRmsAbvGrd

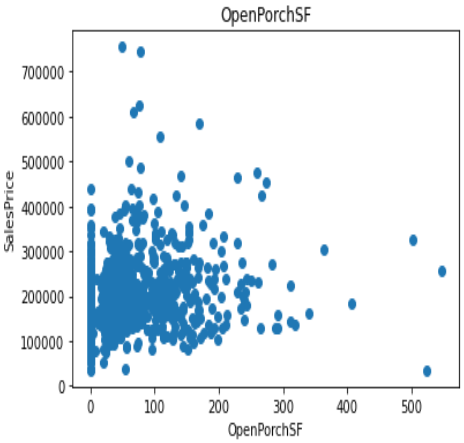

Fireplaces
SalesPrice
Fireplaces

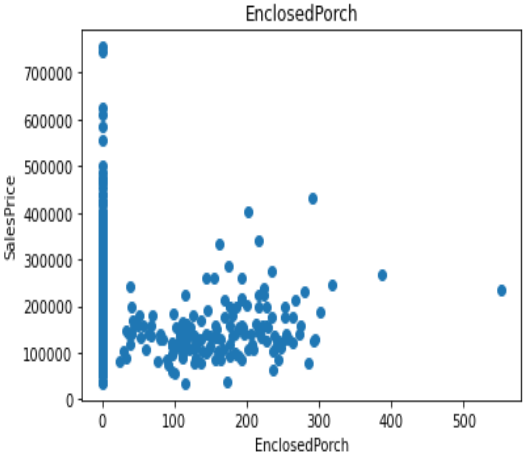

GarageYrBlt
SalesPrice
GarageYrBlt
0
100000
200000
300000
400000
500000
600000
700000
1900
1920
1940
1960
1980
2000

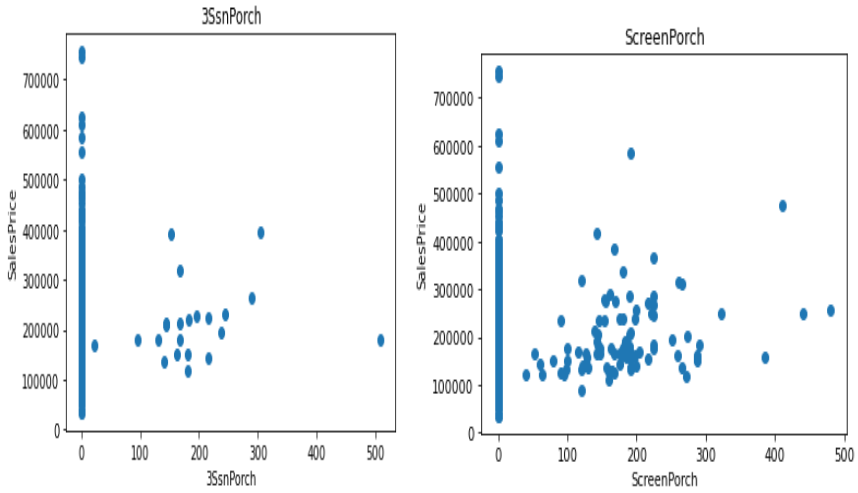

GarageCars
SalesPrice
GarageCars
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

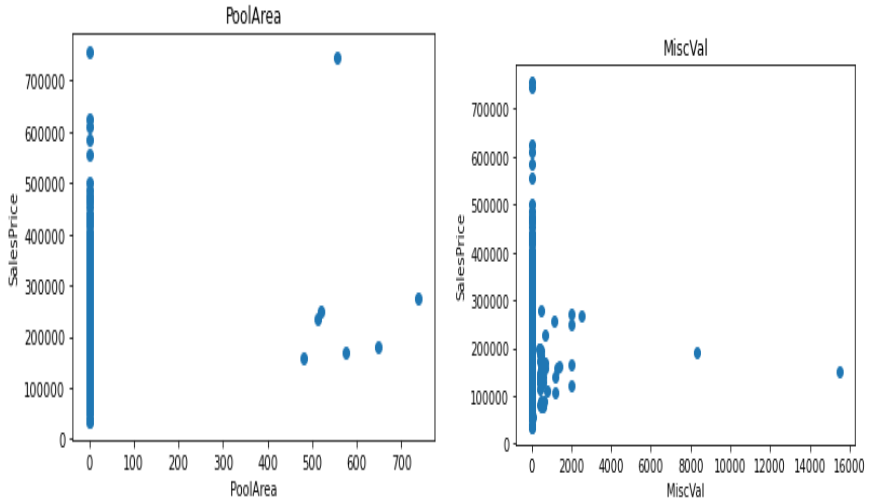

GarageArea
SalesPrice
GarageArea
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800
1000
1200
1400

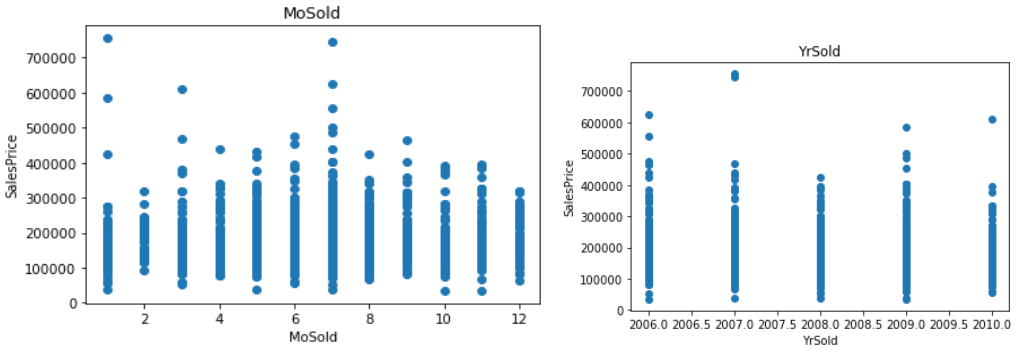

WoodDeckSF
SalesPrice
WoodDeckSF
0
100000
200000
300000
400000
500000
600000
700000
0
200
400
600
800


OpenPorchSF
SalesPrice
OpenPorchSF
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500


EnclosedPorch
SalesPrice
EnclosedPorch
0
100000
200000
300000
400000
500000
600000
700000
0
100
200
300
400
500







Observation

1.Mssubclass: Identifies the type of dwelling involved in the sale:Thus we see that dwelling type 60 (2-STORY 1946 & NEWER) are selling most with a price above 100k and it has gone above 700k too. Those which are price less than

below 100k having dwelling type (1-STORY 1946 & NEWER ALL STYLES) are also selling more comparison to other dwelling .

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

2.LotArea: It doesnot impact the sales price much as from 0-20000 it is having different prices.

3.OverallQual:OverallQuality is directly proportional to sales price.

4.YearBuilt:Houses that are build in recent years having high sales price with respect to houses built in early years.

5.YearRemodAdd:Houses which are remodelled in recent years are having high sales price with respect to houses remodelled in early years.

6.BsmtFinSF1,BsmtUnfSf,TotalBsmtSF are directly proportional to sale price.

7.1stflrSF,2ndFlrSF are directly proportional to sale price.

8.GrLivArea is directly proportional to sales price.

9.BsmtFullBath,BsmtHalfBath are inversely proportional to sales price.

10.Fullbath is directly proportional to sales price.

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

12.TotRmsabvground are directly proportional to sales price.

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

14.Garagecars:Size of garage in car capacity:Houses having garagecars in car capacity 3 having high selling price followed by 2.

15.Garagearea is directly proportional to salesprice.

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

CONCLUSIONS

Key Findings and Conclusions of the Study

The objective of the case was to create a predictive model that predicts saleprice of houses with the available independent variables which will be used by the management to understand how exactly the prices vary with the variables. So first I started with loading the dataset and carry out data analysis and then did the EDA process with visualization patterns using pie-plot, box plot, distribution plot, scatter plot and learnt about different relationship between the features and target variable.

After that I did pre-processing techniques like checking outliers, removal of skewness, encoding of categorical column, handling multicollinearity using Pearson correlation technique and find out the most important features using mutual information gain and selectPercentile with respect to target variable and finally scaling of data for standardization.

Then I did the model training, building the model and finding out the best model out of several models on the basis of different evaluation metrices scores like Mean Absolute Error, Mean squared Error, Root Mean Squared Error,MAPE etc.

We find that Random Forest Regressor was the best fitted model as it was having less MAPE(which means how much predicted results and actual results are away,Less Mape means good fit model) ,more R2 score.Then I did cross validation to reduce overfitting problem and I found random forest regressor and gradient boosting are performing well. So I did hypertunning through GridsearchCv on these two models taking the parameters to improve the model. As the score was bit less which means that it may have reduced the overfitting problem. Thus concluded that Random Forest was

performing well than gradient Boosting Model with the help of best fit line curve which was showing most the datapoints are covered by the best fit line and some are away from the line showing error which can be improved by doing selecting more parameters or taking more models.

Thus finally concluding saying that Random Forest model was having the highest precision accuracy for prediction the sales price of the house with machine learning data. Hence by implementation of this model company can strategise the plans for firm growth and concentrate on areas that will yield high returns in short duration of time and limited manual intervention.

I saved the best model using pickle method and loaded the model for prediction test and find 87% accuracy which is very good. Thus this model can be used in further deployment process.

Then I have used the test dataset and performed all the pre-processing methods to it and then loaded the saved model that we have earlier obtained from train dataset and did the predictions over the test data and then saving the predictions separately in a csv file.

Overall, this dataset is good for predicting the Housing prices based on regression analysis using Random Forest Regressor as the best suited model.

Learning Outcomes of the Study in respect of Data Science

The case study pertains to the real-life scenario of the real estate -Housing category. Houses are not only shelters but also a relatively profitable investment tool. At this point, a realistic prediction of real estate sales prices is important for buyers/sellers and for those who want to invest in real estate (Bin, 2004). However, it is quite difficult to predict the exact value of a residential real estate because it includes a variety of features (environmental, spatial, structural, quality of materials used inside a house, etc.) that are difficult to measure and gather information about.

With the help of dataset provide to us ,we came to know the how saleprice of the houses influences different features of the house ,neighbourhoods, pavement, amenities ,basement, garage etc and how these features impact the cost using different visualization technique like distribution plot,scatter

plot,,box plot ,distribution plot,pie-plot for feature study related category. We also came to know which are the top most features out of the lot influencing sales price of the house. We handled the raw data having nan values,missing values,outliers with the suitable imputation technique and zscore method respectively.We also removed the skewness through power transform method and finally scaled the data for further building of model.

Therefore in this project,we present various algorithms while predicting housing price with good accuracy. We tested various regression models such as linear regression, lasso regression, ridge regression, Random Forest regression, support vector regression , gradient boosting regression,XGBoost regression,AdaBoost regression,Descisiion Tree Regressor and selected the best fit among models and also done hyperparameter tuning. This project directs us that it can be the best application of the machine learning model in order to optimise the result.

Therfore we Therefore in this

Housing price prediction can help developer to determine the selling price of the house and can help customer to arrange right time to purchase or invest in the house.

Limitations of this work and Scope for Future Work

1.The dataset contains more than 50% of the missing value in many columns which we have to drop and rest we imputed with mean and mode imputation technique which can lead to biasness.

2.High skewed data can reduce the effectiveness of the model.

3.The case study did not take into consideration economic factors, interest rate, government subsidies and population.

4.For future work, we recommend that working on large dataset would yield a better and real picture about the model.

5. We can still improve error, model accuracy with some feature engineering and by doing some extensive hyperparameter tuning on it.