HOUSING PRICE PREDICTION

Submitted by:

ADITI GUPTA

ACKNOWLEDGMENT

Gratitude takes three forms-"A feeling from heart, an expression in words and a giving in return". We take this opportunity to express our feelings.

I express my gracious gratitude to our project guide **Khushboo Garg,** SME of Flip Robo Technologies, for his valuable guidance and assistance. I am very thankful for his encouragement and inspiration that made project successful.

I express my gracious gratitude to our Trainer **Dr. Deepika Sharma,** Training Head of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, for her valuable guidance and assistance throughout the PG Program. I am very thankful for her encouragement and inspiration that made project successful.

I would like to thank **Vishal**, In House Data Scientist of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, for his constant support, encouragement, and guidance during project.

I would like to thank **Shankar**, In House Data Scientist of DataTrained - Data Analytics, Data Science Online Training, Bengaluru for his profound guidance throughout the training.

My special thanks to all the **Instructors** and **Subject Matter Experts** of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, who helped me during the live sessions and during doubt clearing scenarios from which I received a lots of suggestions that improved the quality of the work.

I express my deep sense of gratitude to my family for their moral support and understanding without which the completion of my project would not have been perceivable.

Place: Jaipur, Rajasthan Aditi Gupta

INTRODUCTION

Business Problem Framing

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company.

A US-based housing company named **Surprise Housing** has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

The company is looking at prospective properties to buy houses to enter the market. It is 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. So we are building a model that helps to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house.

This model will help to determine 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

As we are working on the Housing project dataset we can easily understand that the data belongs to the Housing and Real Estate which will eventually involve several Financial, Costing, Are & Neighbourhood, Statistical, and Technical terms in the dataset. As this data belongs to A US-based housing company named Surprise Housing as they have 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. So here we need to understand several words related to the core Domain.

The domain related concepts which are useful for better understanding of the project are.

Some relevant details of individual columns are,

1. 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

2. 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

3. LotFrontage: Linear feet of street connected to property

4. LotArea: Lot size in square feet

5. Street: Type of road access to property

Grvl	Gravel
Pave	Paved

6. Alley: Type of alley access to property

Grvl	Gravel
Pave	Paved
NA	No alley access

7. LotShape: General shape of property

Reg	Regular
IR1	Slightly irregular
IR2	Moderately Irregular
IR3	Irregular

8. 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

9. 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

10. 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

11. LandSlope: Slope of property

Gtl Gentle slope

Mod Moderate Slope

Sev Severe Slope

12. 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

MeadowVMeadow 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

13. 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

14. 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

15. 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

16. 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

17. 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

18. 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

19. YearBuilt: Original construction date

20. 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

21. 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

22. Exterior1st: Exterior covering on house

AsbShng Asbestos Shingles

AsphShn Asphalt Shingles

BrkCommBrick 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

23. 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

24. MasVnrType: Masonry veneer type

BrkCmn Brick Common

BrkFace Brick Face

CBlock Cinder Block

None None

Stone Stone

25. MasVnrArea: Masonry veneer area in square feet

26. ExterQual: Evaluates the quality of the material on the exterior

ExExcellent

GdGood

TA Average/Typical

Fa Fair

Po Poor

27. Exter Cond: Evaluates the present condition of the material on the exterior

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

28. Foundation: Type of foundation

BrkTil Brick & Tile

CBlock Cinder Block

PConc Poured Contrete

Slab Slab

Stone Stone

Wood Wood

29. 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

30. 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

31. 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

32. 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

33. BsmtFinSF1: Type 1 finished square feet

34. 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

35. BsmtFinSF2: Type 2 finished square feet

36. BsmtUnfSF: Unfinished square feet of basement area

37. TotalBsmtSF: Total square feet of basement area

38. 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

39. HeatingQC: Heating quality and condition

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

40. CentralAir: Central air conditioning

N No

Y Yes

41. 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

42. 1stFlrSF: First Floor square feet

43. 2ndFlrSF: Second floor square feet

44. LowQualFinSF: Low quality finished square feet (all floors)

45. GrLivArea: Above grade (ground) living area square feet

46. BsmtFullBath: Basement full bathrooms

47. BsmtHalfBath: Basement half bathrooms

48. FullBath: Full bathrooms above grade

49. HalfBath: Half baths above grade

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

51. Kitchen: Kitchens above grade

52. KitchenQual: Kitchen quality

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

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

54. 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

55. Fireplaces: Number of fireplaces

56. 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

57. 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

58. GarageYrBlt: Year garage was built

59. GarageFinish: Interior finish of the garage

Fin	Finished

RFn	Rough Finished

Unf	Unfinished

NA	No Garage

60. GarageCars: Size of garage in car capacity

61. GarageArea: Size of garage in square feet

62. GarageQual: Garage quality

Ex	Excellent

Gd	Good

TA	Typical/Average

Fa	Fair

Po	Poor

NA	No Garage

63. GarageCond: Garage condition

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

64. PavedDrive: Paved driveway

Y Paved

P Partial Pavement

N Dirt/Gravel

65. WoodDeckSF: Wood deck area in square feet

66. OpenPorchSF: Open porch area in square feet

67. EnclosedPorch: Enclosed porch area in square feet

68. 3SsnPorch: Three season porch area in square feet

69. ScreenPorch: Screen porch area in square feet

70. PoolArea: Pool area in square feet

71. PoolQC: Pool quality

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

NA No Pool

72. Fence: Fence quality

GdPrv Good Privacy

MnPrv Minimum Privacy

GdWo Good Wood

MnWw Minimum Wood/Wire

NA No Fence

73. 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

74. MiscVal: $Value of miscellaneous feature

75. MoSold: Month Sold (MM)

76. YrSold: Year Sold (YYYY)

77. 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

78. 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)

Review of Literature

This section discusses about the reported work carried out in various fields of Housing project & Real Estate sectors.

Chris Herbert et. Al., describes that household growth is now back from post-recession lows, but new home construction remains depressed, with additions to supply barely keeping pace with the number of new households. The most significant factors, however, are raising land prices and regulatory constraints on development. [Interactive Map] These constraints, largely imposed at the local level, raise costs and limit the number of homes that can be built in places where demand is highest. Meanwhile, a large percentage of new housing being built is intended primarily for the higher end of the market. The limited supply of smaller, more affordable homes in the face of rising demand suggests that the rising land costs and the difficult development environment make it unprofitable to build for the middle market. Also the number of homeowners rose sharply, even as the ratio of median home price to median household income rose from a low of 3.3 in 2011 to 4.1 in 2018, a sign of deteriorating affordability.

Gordon Davis et. Al., describes that housing affordability is an economic and social problem that affects rich and poor countries alike. No respecter of nationalities or cultures, it arises whenever fewer units of housing are available for sale or rent, for whatever reasons - in a given price or rent range than the number of households looking to buy or rent that can afford housing at the given price or rent. In fact, housing affordability problems tend to migrate down the price/rent scale. Regardless of the price range where a housing shortage first arises, buyers in that price range will tend to bid up less expensive units, the end result being to drive the shortage to the lower end of the housing price range. The rental market functions the same way: a shortage in supply of rental units in any rent range will migrate to the lower end. And, as alluded to above, as between purchases and rentals, rising purchase prices tend to nudge rents higher, and vice versa. The factors influencing the availability of affordable housing might be viewed as falling into two categories: market factors and individual household factors.

Nehal N Ghosalkar et. Al., describes that he real estate market is a standout amongst the most focused regarding pricing and keeps fluctuating. It is one of the prime fields to apply the ideas of machine learning on how to enhance and foresee the costs with high accuracy. There are three factors that influence the price of a house which includes physical conditions, concepts and location. The current framework includes estimating the price of houses without any expectations of market prices and cost increment. The objective of the paper is prediction of residential prices for the customers considering their financial plans and needs. By breaking down past market patterns and value ranges, and coming advancements future costs will be anticipated. This examination means to predict house prices in the city with Linear Regression. It will help clients to put resources into a bequest without moving toward a broker.

Elena Sliogeris et. Al., describes that incidence of the problem has spread from very low-income through low-income into moderate-income households. There is now a consistent call for housing schemes to retain 'key workers' and 'the working poor' in established areas to ensure access to employment, education, public transport and other facilities and amenities. Land com has a strategic position within this landscape and there exists a range of current and potential mechanisms land com might utilise to create and maintain a pool of affordable houses. Numerous interrelated factors have driven the loss of affordability, including an increased willingness and capacity to pay for housing due to increased incomes and more accessible lines of credit. Concurrent increases in population, decreases in household size and increases in house size have further compounded the problem. The role of supply-side impediments to housing development that contribute to a loss of affordability is strongly contested. The planning processes may have a role to play in addressing affordability concerns. However, the house prices are largely countered by recent market conditions in city areas have continued to increase in price while house prices in outer areas have stagnated or decreased in value. The landscape of affordability is influenced to a large extent by access to jobs, public transport and other social amenities. This highlights the need for future housing provision to address employment, transport and other infrastructure as well as the volume of housing supply.

George Earl et. Al., describes that markets are the central institutions of economies, allowing people to buy and sell goods and services in a manner that potentially makes everyone better off. However, markets can only be formed under certain conditions, and when these conditions are absent, markets may struggle to exist or the conditions may lead to market failures. This is the basic underlying principle of missing or incomplete markets; that is, failure to produce some goods and services despite being needed or wanted. It is well known that microeconomic equilibrium occurs when the demand for goods is equal to the supply. A missing market, therefore, is a sign that the market is out of equilibrium; a situation where markets do not exist or where the equilibrium price is not related to either marginal social benefits or marginal social costs. For decades, the market has been failing to meet the housing needs of its lowest income residents, and the situation is getting steadily worse. Many people on low incomes cannot afford to buy their own homes, and housing rents have also become increasingly unaffordable in recent years. Therefore, by definition, sustainable housing means that everyone should have the opportunity to live in a decent home at a price they can afford, in a place in which they want to live and work.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

We are building a model Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. So this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house. This will help to determine the price of houses with the available independent variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The most common form of regression analysis is linear regression, in which one finds the line (or a more complex linear combination) that most closely fits the data according to a specific mathematical criterion. For specific mathematical reasons this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values.

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables.

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

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

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

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

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

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

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

7. Random forest - is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A Random Forest's nonlinear nature can give it a leg up over linear algorithms, making it a great option.

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

9. GradientBoosting 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.

10. Extratrees Regressor - This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

11. XGBoost Regressor - XGBoost is an implementation of gradient boosted decision trees designed for speed and performance

Data Sources and their formats

The given dataset is in CSV format, now let’s load the dataset and do the analysis.

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

```
In [2]: df_train=pd.read_csv("housing_train.csv")
        df_test=pd.read_csv("housing_test.csv")
```

```
In [3]: df_train
```

Out[3]:

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

1168 rows × 81 columns

```
In [4]: df_test
```

Out[4]:

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

292 rows × 80 columns

Checking the data type & info of dataset

```
[5]: # checking for datatypes
     df_train.dtypes,df_test.dtypes
```

```
t[5]: (Id                 int64
       MSSubClass         int64
       MSZoning          object
       LotFrontage      float64
       LotArea            int64
                          ...
       MoSold             int64
       YrSold             int64
       SaleType          object
       SaleCondition     object
       SalePrice          int64
       Length: 81, dtype: object,
       Id                 int64
       MSSubClass         int64
       MSZoning          object
       LotFrontage      float64
       LotArea            int64
                          ...
       MiscVal            int64
       MoSold             int64
       YrSold             int64
       SaleType          object
       SaleCondition     object
       Length: 80, dtype: object)
```

Checking value_counts of each column in data frame.

```
[6]: # checking values of every column in train dataframe
     for col in df_train.columns:
         print(col)
         print(df_train[col].value_counts())
         print()
```

```
962     1
963     1
1       1
Name: Id, Length: 1168, dtype: int64

MSSubClass
20     428
60     244
50     113
120     69
70      53
30      52
160     47
80      43
90      41
190     26
85      19
75      14
45      10
180      6
```

```
[7]: # checking values of every column in test dataframe
     for col in df_test.columns:
         print(col)
         print(df_test[col].value_counts())
         print()
```

```
Id
56      1
1217    1
1185    1
162     1
676     1
       ..
340     1
855     1
858     1
1371    1
512     1
Name: Id, Length: 292, dtype: int64

MSSubClass
20     108
60      55
50      31
120     18
30      17
```

Data Pre-processing

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

Dropping unnecessary columns from datasets

```
n [8]: df_train.drop('Id',axis=1,inplace=True)  # id column not necessary for prediction
       df_train.drop('Utilities',axis=1,inplace=True) # contains same value in every row
       df_train.drop('PoolArea',axis=1,inplace=True) # contains same value in every row
       df_train.drop('PoolQC',axis=1,inplace=True) # contains same value in every row
       df_test.drop('Id',axis=1,inplace=True)   # id column not necessary for prediction
       df_test.drop('Utilities',axis=1,inplace=True)   # id column not necessary for prediction
       df_test.drop('PoolArea',axis=1,inplace=True)   # id column not necessary for prediction
       df_test.drop('PoolQC',axis=1,inplace=True)   #contains same value in every row
```

Handling the missing values in datasets

```
]: #checking for null values
   df_train.isnull().sum(),df_test.isnull().sum()
```

```
]: (MSSubClass        0
    MSZoning          0
    LotFrontage     214
    LotArea           0
    Street            0
                   ...
    MoSold            0
    YrSold            0
    SaleType          0
    SaleCondition     0
    SalePrice         0
    Length: 77, dtype: int64,
    MSSubClass       0
    MSZoning         0
    LotFrontage     45
    LotArea          0
    Street           0
                    ..
    MiscVal          0
    MoSold           0
    YrSold           0
    SaleType         0
    SaleCondition    0
    Length: 76, dtype: int64)
```

Missing Values in Percentage

Train Dataset

10]:

	Missing Values	% of Total Values
MiscFeature	1124	96.2
Alley	1091	93.4
Fence	931	79.7
FireplaceQu	551	47.2
LotFrontage	214	18.3
GarageType	64	5.5
GarageYrBlt	64	5.5
GarageFinish	64	5.5
GarageQual	64	5.5
GarageCond	64	5.5
BsmtExposure	31	2.7
BsmtFinType2	31	2.7
BsmtCond	30	2.6
BsmtFinType1	30	2.6
BsmtQual	30	2.6
MasVnrArea	7	0.6
MasVnrType	7	0.6

Test Dataset

11]:

	Missing Values	% of Total Values
MiscFeature	282	96.6
Alley	278	95.2
Fence	248	84.9
FireplaceQu	139	47.6
LotFrontage	45	15.4
GarageType	17	5.8
GarageYrBlt	17	5.8
GarageFinish	17	5.8
GarageQual	17	5.8
GarageCond	17	5.8
BsmtCond	7	2.4
BsmtExposure	7	2.4
BsmtFinType1	7	2.4
BsmtFinType2	7	2.4
BsmtQual	7	2.4
MasVnrArea	1	0.3
MasVnrType	1	0.3
Electrical	1	0.3

Comparing Garage year with Housing Year to fill null values of Garage year

	Garage_Year	Housing_Year
0	1977.0	1976
1	1970.0	1970
2	1997.0	1996
3	1977.0	1977
4	1977.0	1977
5	2006.0	2006
6	1957.0	1957
7	1957.0	1957
8	1965.0	1965
9	1947.0	1947

By analyzing the column of "YearBuilt & GarageYrBlt" we can see that both have similar Year values in the maximum number of columns. So replacing the Null/NaN values with frequently occurring or mode method will not give accurate input to the model. So replacing GarageYrBlt column null values with the actual built year of the house will result in better input.

Filling Missing Values in Datasets

Filling Null Values for Train Dataset

13]:
```
# As per given definition, NA means None. Let's replace NAs with 'None'
df_train['MiscFeature'].fillna('None',inplace=True)
print(df_train['MiscFeature'].value_counts())


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

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

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

basement=['BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2']
# As per given definition, NA means No_Basement. Let's replace NAs with 'No_Basement'
for i in basement:
    df_train[i].fillna('No_Basement',inplace=True)
    print(df_train[i].value_counts())

garage=['GarageType','GarageFinish','GarageQual','GarageCond']
for i in garage:
    print(df_train[i].value_counts())
# As per given definition, NA means No_Garage (Refer Variable Description at the end of the notebook). Let's replace NAs with 'No
for i in garage:
    df_train[i].fillna('No_Garage',inplace=True)
    print(df_train[i].value_counts())

# As per given values of MasVnrType,  Let's replace NAs with 'none' that is with mode value
df_train['MasVnrType'].fillna('None',inplace=True)
print(df_train['MasVnrType'].value_counts())

# As per given values of MasVnrArea,  Let's replace NAs with 'none' that is with mode value
df_train['MasVnrArea'].fillna(0,inplace=True)
print(df_train['MasVnrArea'].value_counts())

# As per given values of LotFrontage,  Let's replace NAs with 'median' of the same column
df_train['LotFrontage'].fillna(df_train['LotFrontage'].median(),inplace=True)
print(df_train['LotFrontage'].value_counts())

# As per dataframe "df" we can say that most of the rows of GarageYrBlt has same value as YearBuilt so we replace with that
df_train["GarageYrBlt"]=df_train["GarageYrBlt"].fillna(df_train["YearBuilt"])
print(df_train['GarageYrBlt'].value_counts())
```

```
None     1124
Shed       40
Gar2        2
Othr        1
TenC        1
Name: MiscFeature, dtype: int64
No_alley_access    1091
Grvl                 41
Pave                 36
Name: Alley, dtype: int64
No_Fence    931
MnPrv       129
GdPrv        51
GdWo         47
MnWw         10
Name: Fence, dtype: int64
No_Fireplace    551
Gd              301
TA              252
```

Filling Null values for Test Dataset

In [14]:
```
# As per given definition, NA means None. Let's replace NAs with 'None'
df_test['MiscFeature'].fillna('None',inplace=True)
print(df_test['MiscFeature'].value_counts())


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

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

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

basement=['BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2']
# As per given definition, NA means No_Basement. Let's replace NAs with 'No_Basement'
for i in basement:
    df_test[i].fillna('No_Basement',inplace=True)
    print(df_test[i].value_counts())

garage=['GarageType','GarageFinish','GarageQual','GarageCond']
for i in garage:
    print(df_test[i].value_counts())
# As per given definition, NA means No_Garage (Refer Variable Description at the end of the notebook). Let's replace NAs with 'No
for i in garage:
    df_test[i].fillna('No_Garage',inplace=True)
    print(df_test[i].value_counts())

# As per given values of MasVnrType,  Let's replace NAs with 'none' that is with mode value
df_test['MasVnrType'].fillna('None',inplace=True)
print(df_test['MasVnrType'].value_counts())

# As per given values of MasVnrArea,  Let's replace NAs with 'none' that is with mode value
df_test['MasVnrArea'].fillna(0,inplace=True)
print(df_test['MasVnrArea'].value_counts())

# As per given values of LotFrontage,  Let's replace NAs with 'median' of the same column
df_test['LotFrontage'].fillna(df_test['LotFrontage'].median(),inplace=True)
print(df_test['LotFrontage'].value_counts())

# As per dataframe "df" we can say that most of the rows of GarageYrBlt has same value as YearBuilt so we replace with that
df_test["GarageYrBlt"]=df_test["GarageYrBlt"].fillna(df_test["YearBuilt"])
print(df_test['GarageYrBlt'].value_counts())

# As per given values of Electrical,  Let's replace NAs with 'none' that is with mode value
df_test['Electrical'].fillna('SBrkr',inplace=True)
print(df_test['Electrical'].value_counts())
```

```
None     282
Shed       9
Othr       1
Name: MiscFeature, dtype: int64
No_alley_access    278
Grvl                 9
Pave                 5
Name: Alley, dtype: int64
No_Fence    248
MnPrv        28
GdPrv         8
GdWo          7
MnWw          1
Name: Fence, dtype: int64
No_Fireplace    139
Gd               79
TA               61
Fa                8
Ex                3
```

Checking Again for Missing Values

:
```
#checking again if missing values present in train dateset
df_train.isnull().values.any()
```

: False

:
```
#checking again if missing values present in test dataset
df_test.isnull().values.any()
```

: False

Data Inputs- Logic - Output Relationships

The given dataset has 77 columns after pre processing of the data in which the "SalesPrice" column is an output column. The below analysis describes the relationship behind the data input, its format, the logic in between and the output.

Replacing the Categorical values (alphabetic values) to numeric values

51]:
```
# encoding all to numeric values

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['MSZoning','Street','Alley', 'LotShape','LandContour','LotConfig','LandSlope', 'Neighborhood','Condition1','Condition2',
       'BldgType','HouseStyle','RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', 'ExterCond',
       'Foundation', 'BsmtQual','BsmtCond', 'BsmtExposure', 'BsmtFinType1','BsmtFinType2',  'Heating','HeatingQC','CentralAir',
       'Electrical', 'KitchenQual','Functional',  'FireplaceQu', 'GarageType','GarageFinish',  'GarageQual', 'GarageCond',
       'PavedDrive', 'Fence', 'MiscFeature',  'SaleType','SaleCondition', ]
for val in list1:
    df_train[val]=le.fit_transform(df_train[val].astype(str))
    df_test[val]=le.fit_transform(df_test[val].astype(str))
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1
0	127	120	3	70.98847	4928	1	1	0	3	0	4	0	13	2
1	889	20	3	95.00000	15865	1	1	0	3	0	4	1	12	2
2	793	60	3	92.00000	9920	1	1	0	3	0	1	0	15	2
3	110	20	3	105.00000	11751	1	1	0	3	0	4	0	14	2
4	422	20	3	70.98847	16635	1	1	0	3	0	2	0	14	2
5	1197	60	3	58.00000	14054	1	1	0	3	0	4	0	8	2
6	561	20	3	70.98847	11341	1	1	0	3	0	4	0	19	2
7	1041	20	3	88.00000	13125	1	1	3	3	0	0	0	19	2
8	503	20	3	70.00000	9170	1	1	3	3	0	0	0	7	1
9	576	50	3	80.00000	8480	1	1	3	3	0	4	0	12	2

Summary Statistics

In descriptive statistics, summary statistics are used to summarize a set of observations, in order to communicate the largest amount of information as simply as possible. Summary statistics summarize and provide information about your sample data. It tells something about the values in data set. This includes where the average lies and whether the data is skewed.

The describe() function computes a summary of statistics pertaining to the Data Frame columns. This function gives the mean, count, max, standard deviation and IQR values of the dataset in a simple understandable way.

```
#Checking the summary of the dataset

df_train.describe()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	LotConfig	LandSlope
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	56.767979	3.013699	70.988470	10484.749144	0.996575	0.995719	1.938356	2.773973	3.004281	0.064212
std	41.940650	0.633120	22.437056	8957.442311	0.058445	0.256832	1.412262	0.710027	1.642667	0.284088
min	20.000000	0.000000	21.000000	1300.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	20.000000	3.000000	60.000000	7621.500000	1.000000	1.000000	0.000000	3.000000	2.000000	0.000000
50%	50.000000	3.000000	70.988470	9522.500000	1.000000	1.000000	3.000000	3.000000	4.000000	0.000000
75%	70.000000	3.000000	79.250000	11515.500000	1.000000	1.000000	3.000000	3.000000	4.000000	0.000000
max	190.000000	4.000000	313.000000	164660.000000	1.000000	2.000000	3.000000	3.000000	4.000000	2.000000

Correlation Factor

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

Checking corelation in train dataset

In [52]:
```
df_train.corr()
```

Out[52]:

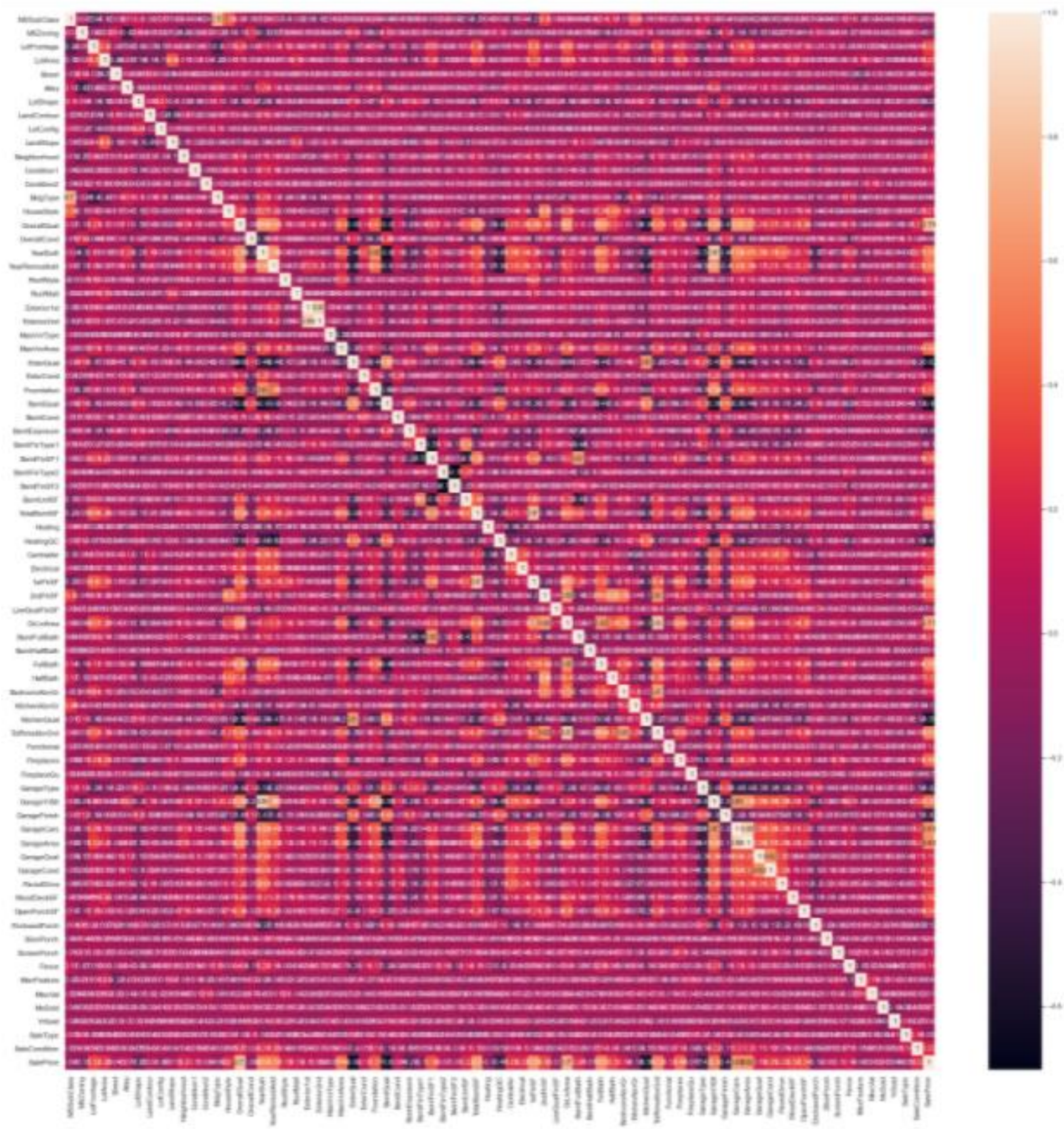
	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	LotConfig	LandSlope	...	3SsnPorch	ScreenPo
MSSubClass	1.000000	0.007478	-0.336234	-0.124151	-0.035981	0.107699	0.104485	-0.021387	0.076880	-0.014930	...	-0.043210	-0.013
MSZoning	0.007478	1.000000	-0.069476	-0.023328	0.140215	-0.415953	0.053655	0.001175	-0.027246	-0.023952	...	0.004409	0.030
LotFrontage	-0.336234	-0.069476	1.000000	0.296790	-0.035131	-0.084593	-0.138975	-0.073725	-0.189317	0.044283	...	0.050499	0.030
LotArea	-0.124151	-0.023328	0.296790	1.000000	-0.263973	-0.037048	-0.189201	-0.159038	-0.152063	0.395410	...	0.025794	0.025
Street	-0.035981	0.140215	-0.035131	-0.263973	1.000000	-0.000978	-0.012941	0.105226	0.000153	-0.141572	...	0.007338	0.016
...	...	...	...	...	...	...	...	...	...	...	...	...	
MoSold	-0.016015	-0.051646	0.022579	0.015141	-0.008860	-0.025186	-0.050418	-0.023872	0.019084	0.030526	...	0.020406	0.033
YrSold	-0.038595	-0.004964	-0.004162	-0.035399	-0.019635	0.010096	0.021421	0.009499	-0.009817	-0.005352	...	0.014440	0.017
SaleType	0.035050	0.079854	-0.036081	0.005421	0.025920	0.008918	-0.015161	-0.041763	-0.002039	0.056004	...	-0.013696	0.010
SaleCondition	-0.028981	0.004501	0.065439	0.034236	0.014176	-0.000467	-0.054905	0.047715	0.043692	-0.061461	...	0.001236	0.002
SalePrice	-0.060775	-0.133221	0.323851	0.249499	0.044753	0.076717	-0.248171	0.032836	-0.060452	0.015485	...	0.060119	0.100

77 rows × 77 columns

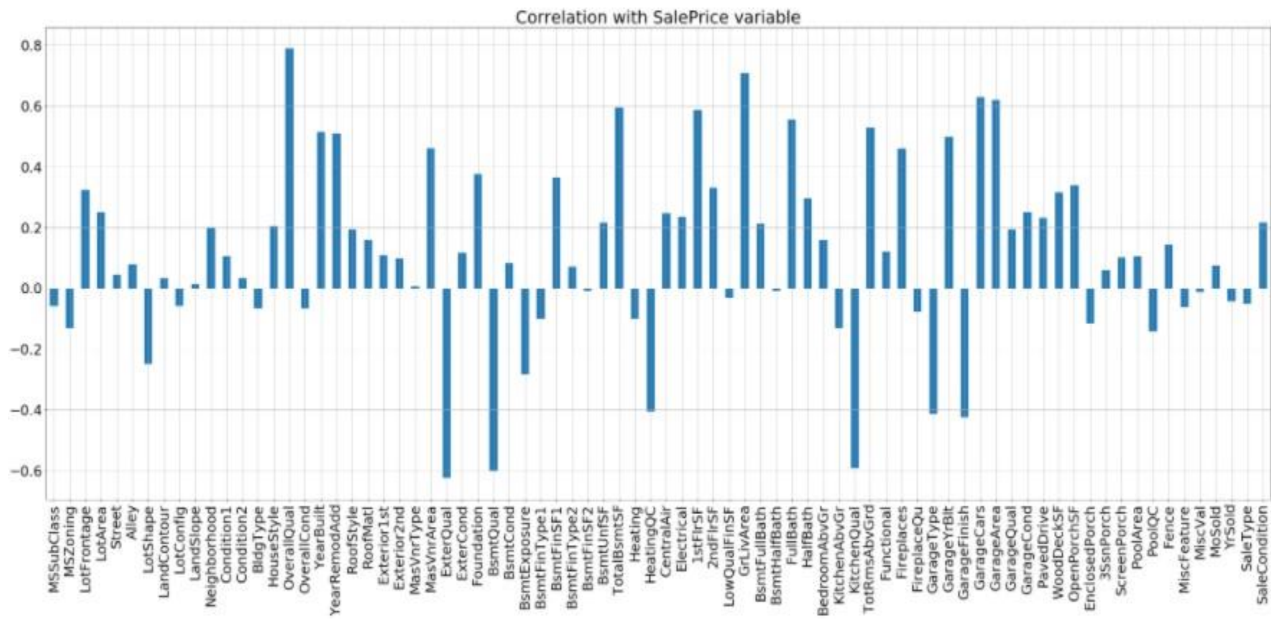
Correlation Matrix

A correlation matrix is a tabular data representing the ‘correlations’ between pairs of variables in a given dataset. It is also a very important pre-processing step in Machine Learning pipelines. The Correlation matrix is a data analysis representation that is used to summarize data to understand the relationship between various different variables of the given dataset.

Correlation factor with visualization / Correlation matrix



Correlation with target column (SalesPrice)



Observation

1. In the Correlation with output column graph we can see that columns “LotShape, ExterQual, BsmtQual, BsmtExposure, HeatingQC” are negatively related with the Output column.

2. Also the columns “KitchenQual, GarageType, GarageFinish” are negatively related with the Output column.

3. All other columns are positively related with the output column and provide significant importance towards the model building.

4. However as we are predicting the price of Housing project and predicting the price of Test dataset, the negatively co related columns are not dropped from the dataset.

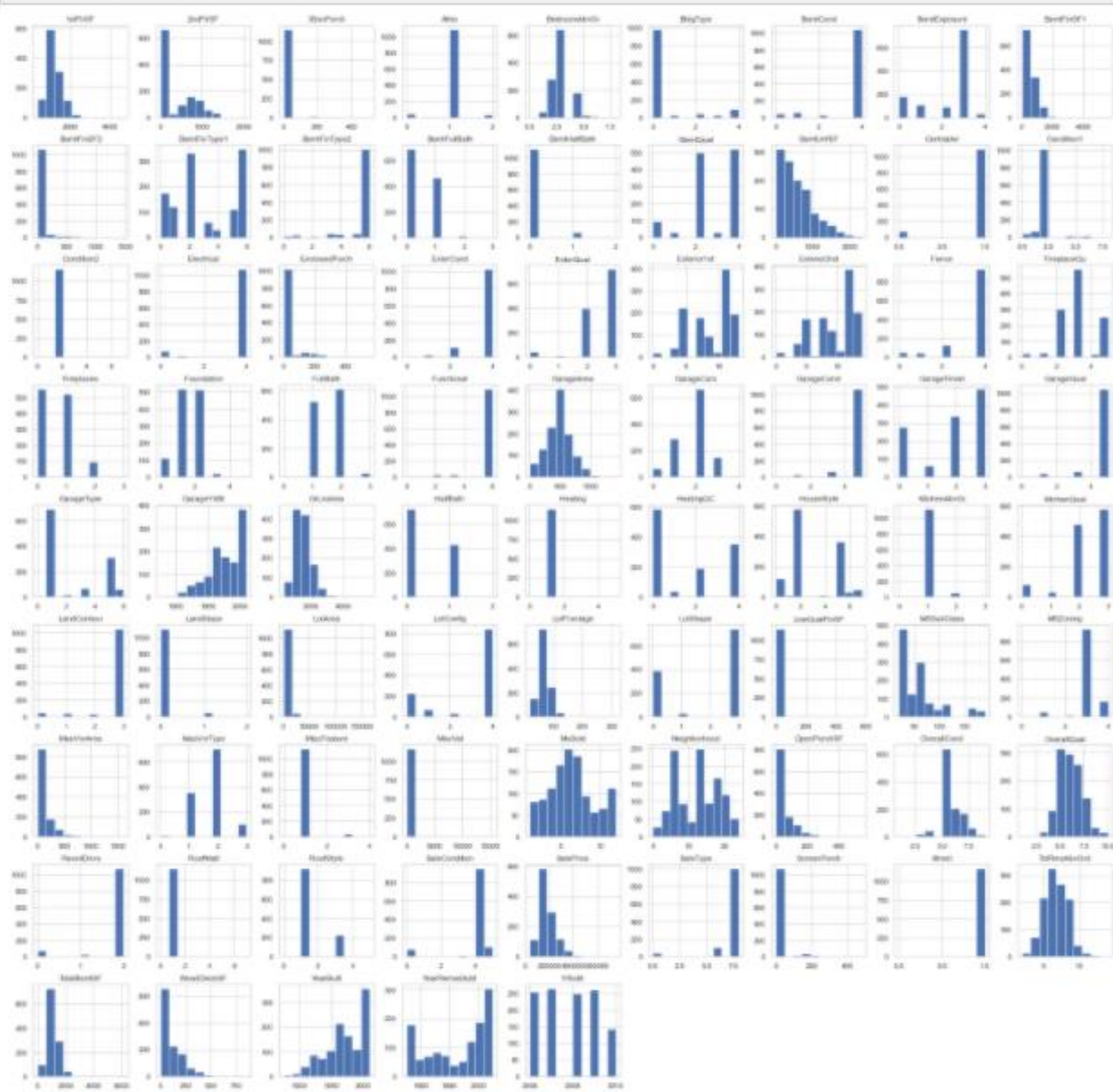
Skewness

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

Check skewness in datasets.

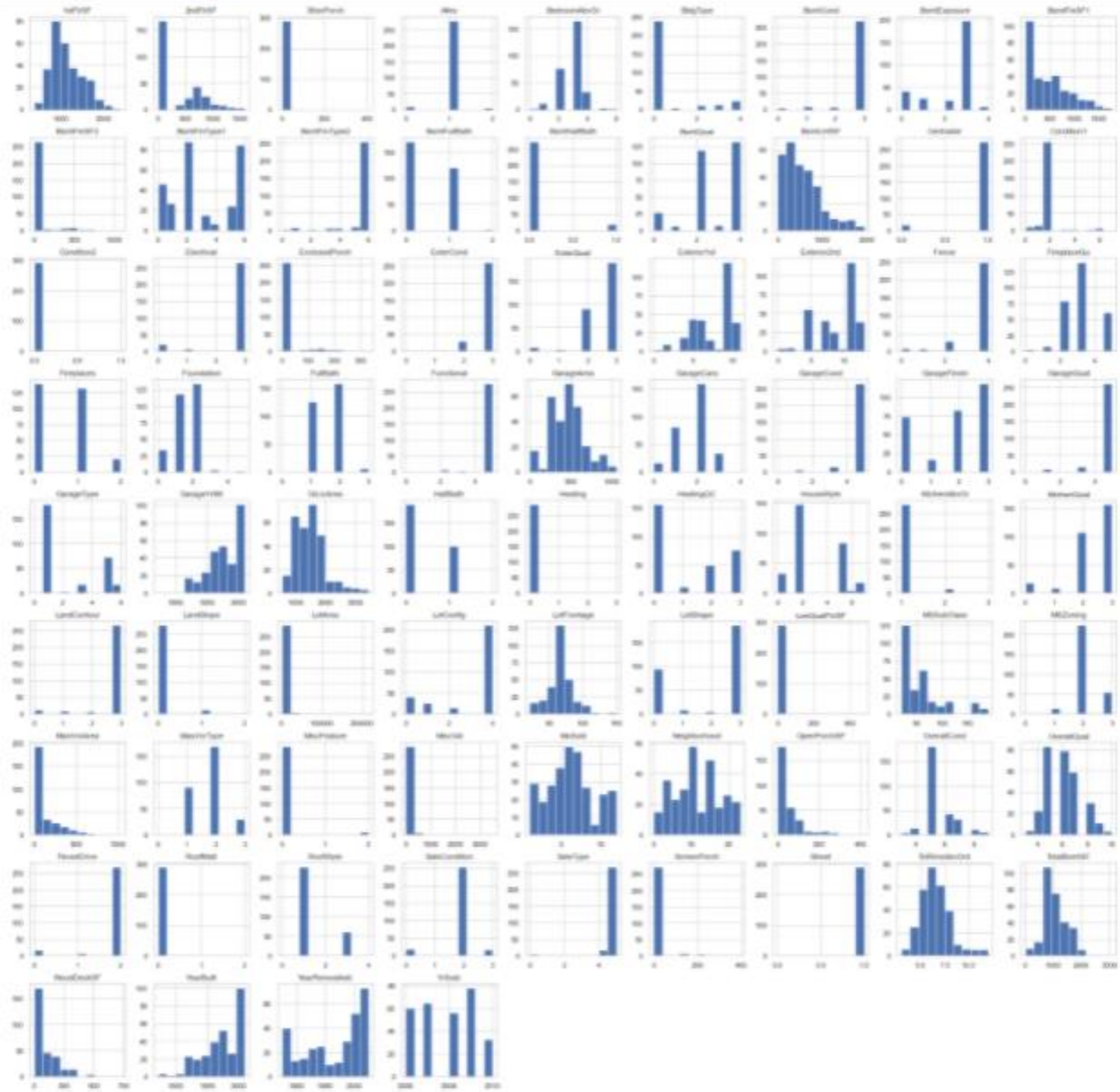
For Train Dataset

```
[59]: # Plotting Hist plot
plt.rcParams.update({'font.size': 10})
df_train.hist(figsize=(30, 30));
```



For Test Dataset

```
[60]: # Plotting Hist plot
plt.rcParams.update({'font.size': 10})
df_test.hist(figsize=(30, 30));
```



For Train Dataset

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

```
1]: MSSubClass           1.422019
    MSZoning            -1.796785
    LotFrontage          2.733440
    LotArea             10.659285
    Street             -17.021969
                          ...
    MoSold               0.220979
    YrSold               0.115765
    SaleType            -3.660513
    SaleCondition       -2.671829
    SalePrice            1.953878
    Length: 77, dtype: float64
```

For Test Dataset

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

```
55]: MSSubClass          1.358597
     MSZoning            0.187174
     LotFrontage         0.499491
     LotArea            12.781805
     Street            -12.020386
                          ...
     MiscVal            13.264758
     MoSold              0.186504
     YrSold              0.018412
     SaleType           -5.489874
     SaleCondition      -2.161104
     Length: 76, dtype: float64
```

Treating Skewness

In the Data Science it is just statistics and many algorithms revolve around the assumption that the data is normalized. So, the more the data is close to normal, the better it is for getting good predictions.There are many ways of transforming skewed data such as log transform, square-root transform, box-cox transform, etc.

1. Log Transform

Log transformation is a data transformation method in which it replaces each variable x with a log(x). The log transformation is, arguably, the most popular among the different types of transformations used to transform skewed data to approximately conform to normality

2. Square Root Transform

The square root, x to x^(1/2) = sqrt(x), is a transformation with a moderate effect on distribution shape: it is weaker than the logarithm and the cube root. It is also used for reducing right skewness, and also has the advantage that it can be applied to zero values. So applying a square root transform inflates smaller numbers but stabilises bigger ones.

3. Box-Cox Transform

In statistics, a power transform is a family of functions that are applied to create a monotonic transformation of data using power functions. This is a useful data transformation technique used to stabilize variance, make the data more normal distribution-like, improve the validity of measures of association such as the Pearson correlation between variables and for other data stabilization procedures.

```
In [63]: # Treating the skewness for train dataset
df_x.skew()
for col in df_x.skew().index:
    if col in df_x.describe().columns:
        if df_x[col].skew()>0.55:
            df_x[col]=np.sqrt(df_x[col])
        if df_x[col].skew()<-0.55:
            df_x[col]=np.cbrt(df_x[col])
```

```
[66]: # Treating the skewness
df_test.skew()
for col in df_test.skew().index:
    if col in df_test.describe().columns:
        if df_test[col].skew()>0.55:
            df_test[col]=np.sqrt(df_test[col])
        if df_test[col].skew()<-0.55:
            df_test[col]=np.cbrt(df_test[col])
```

Outliers

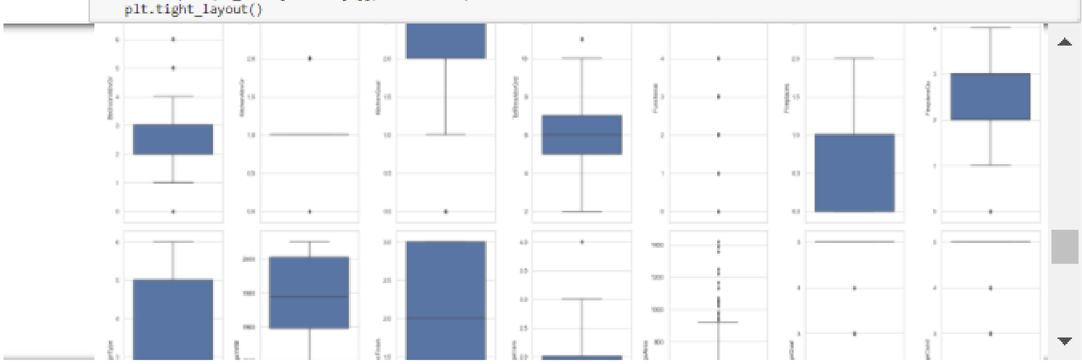
An outlier is a data point in a data set which is distant or far from all other observations available. It is a data point which lies outside the overall distribution which is available in the dataset. In statistics, an outlier is an observation point that is distant from other observations.

A box plot is a method or a process for graphically representing groups of numerical data through their quartiles. Outliers may also be plotted as an individual point. If there is an outlier it will plotted as point in box plot but other numerical data will be grouped together and displayed as boxes in the diagram. 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.

Plotting outliers in the Dataset

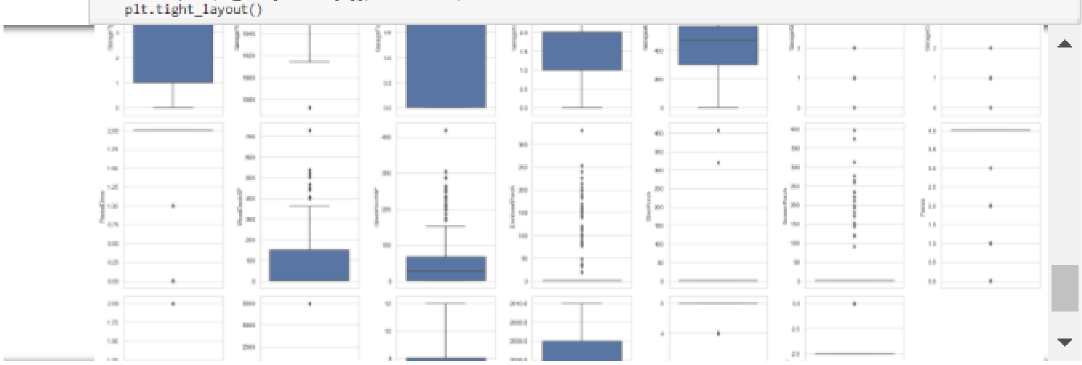
For Train Dataset

In [57]:
```
collist=df_train.columns.values
ncol=7
nrow=11

plt.figure(figsize=(25, 75))
plt.rcParams.update({'font.size': 10})
for i in range(0, len(collist)):
    plt.subplot(nrow, ncol, i+1)
    sns.boxplot(df_train[collist[i]], orient='v')
    plt.tight_layout()
```



For Test Dataset

In [58]:
```
collist=df_test.columns.values
ncol=7
nrow=11

plt.figure(figsize=(25, 50))
plt.rcParams.update({'font.size': 10})
for i in range(0, len(collist)):
    plt.subplot(nrow, ncol, i+1)
    sns.boxplot(df_test[collist[i]], orient='v')
    plt.tight_layout()
```



Hardware and Software Requirements and Tools Used

1. Software Tools used

Python 3.0

MS - Office

Operating System - Windows 10

2. Minimum Hardware Requirement

Processors: Intel Atom® processor or Intel® Core™ i3 processor

Disk space: 2 GB – 3 GB

Operating systems: Windows* 7 or later, macOS, and Linux

Python* versions: 2.7.X, 3.6.X

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

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

The different libraries used for the problem solving are

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

1. sklearn.linear_model

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

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regressions.

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

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of muticollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

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

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

The elastic net method improves on lasso's limitations, i.e., where lasso takes a few samples for high dimensional data, the elastic net procedure provides the inclusion of "n" number of variables until saturation. In a case where the variables are highly correlated groups, lasso tends to choose one variable from such groups and ignore the rest entirely.

To eliminate the limitations found in lasso, the elastic net includes a quadratic expression (||β||2) in the penalty, which, when used in isolation, becomes ridge regression. The quadratic expression in the penalty elevates the loss function toward being convex. The elastic net draws on best of both i.e., lasso and ridge regression. In the procedure for finding the elastic net method's estimator, there are two stages that involve both the lasso and regression techniques. It first finds the ridge regression coefficients and then conducts the second step by using a lasso sort of shrinkage of the coefficients.

2. sklearn.tree –

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

There are several advantages of using decision trees for predictive analysis:

- Decision trees can be used to predict both continuous and discrete values i.e. they work well for both regression and classification tasks.
- They require relatively less effort for training the algorithm.
- They can be used to classify non-linearly separable data.

- They're very fast and efficient compared to KNN and other algorithms.

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

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

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. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

3. sklearn.ensemble

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence. Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.

The different types of ensemble techniques used in the model are

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

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.

4. sklearn.metrics - The sklearn. metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

Important sklearn.metrics modules used in the project are

i. mean_absolute_error - In statistics, mean absolute error is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

ii. mean_squared_error - In statistics, the mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. Mean Square Error (MSE) is defined as Mean or Average of the square of the difference between actual and estimated values.

iii. r2_score - In statistics, the coefficient of determination, denoted R^2 or r^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable. R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determinations for multiple regressions.

5. sklearn.model_selection –

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

ii. cross_val_score - Cross validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross validation the 1^{st} part (20%) of the 5 parts will be kept out as a hold out set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a hold out set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross validation process to get the remaining estimate of the model quality.

cross_val_score estimates the expected accuracy of the model on out-of-training data (pulled from the same underlying process as the training data). The benefit is that one need not set aside any data to obtain this metric, and we can still train the model on all of the available data.

Testing of Identified Approaches (Algorithms)

Algorithm used for training and testing.

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

```
In [62]: #splitting the data for training and test

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

Scaling the input variables

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

For Train Dataset

```
]: from sklearn.preprocessing import StandardScaler
   scale=StandardScaler()
   x=scale.fit_transform(df_x)
```

For Test Dataset

```
]: from sklearn.preprocessing import StandardScaler
   scale=StandardScaler()
   df_test=scale.fit_transform(df_test)
```

Applying PCA

An important machine learning method for dimentionality reduction is called Principal Component Analysis. It is a method that uses simple matrix operations from linear algebra and statistics to calculate a projection of the original data into the same number or fewer dimentions.

For Train Dataset

```
0]: from sklearn.decomposition import PCA
    pca = PCA(n_components=10)
    x=pca.fit_transform(x)
```

For Test Dataset

```
1]: from sklearn.decomposition import PCA
    pca = PCA(n_components=10)
    df_test=pca.fit_transform(df_test)
```

Splitting the data into training and testing data

Train Test Split

Scikit-learn is a Python library that offers various features for data processing that can be used for classification, clustering, and 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. If we have one dataset, then it needs to be split by using the Sklearn train_test_split function first. By default, Sklearn train_test_split will make random partitions for the two subsets.

The train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, we don't need to divide the dataset manually. The train_test_split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

```
#Splitting the data into training and testing data

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=.20, random_state=42)
```

Run and Evaluate selected models

Before running the model & evaluating them, first we need to import all the necessary libraries which are required for the problem solving.

```
[72]: from sklearn.metrics import accuracy_score
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import mean_squared_error,mean_absolute_error
      from sklearn.metrics import r2_score
```

```
[73]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.22,random_state=42)
```

```
[74]: from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
      from sklearn.tree import DecisionTreeRegressor
      from sklearn.neighbors import KNeighborsRegressor
```

Now using for loop running all the models such as "Linear Regression, Lasso, Ridge, ElasticNet, KNeighborsRegressor, Decision Tree Regressor.

```
In [76]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.22,random_state=49)
         model=[LinearRegression(),Lasso(),Ridge(),ElasticNet(),DecisionTreeRegressor(),KNeighborsRegressor()]

         for m in model:
             m.fit(x_train,y_train)
             print("Results of ",m," is")
             predm=m.predict(x_test)
             print("R2 Score:",r2_score(y_test,predm))
             print("Error:")
             print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,predm)))
             print("******************************************************************************************")
             print('\n')
```

Output

```
Results of  LinearRegression()  is
R2 Score: 0.8663864650198234
Error:
Root Mean Squared Error: 28578.19095528328
*******************************************************************************


Results of  Lasso()  is
R2 Score: 0.8663870135280889
Error:
Root Mean Squared Error: 28578.132295844167
*******************************************************************************
```

```
Results of  Ridge()  is
R2 Score: 0.8663870717978961
Error:
Root Mean Squared Error: 28578.126064255146
**********************************************************************


Results of  ElasticNet()  is
R2 Score: 0.8615477666719318
Error:
Root Mean Squared Error: 29091.056212358308
**********************************************************************


Results of  DecisionTreeRegressor()  is
R2 Score: 0.7225304108082851
Error:
Root Mean Squared Error: 41182.92626470022
**********************************************************************


Results of  KNeighborsRegressor()  is
R2 Score: 0.8322815994476018
Error:
Root Mean Squared Error: 32018.417464162972
**********************************************************************
```

Using Some Ensemble Techniques to boost up the score

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from xgboost import XGBRegressor
```

```python
model=[RandomForestRegressor(),AdaBoostRegressor(),GradientBoostingRegressor(),ExtraTreesRegressor(),XGBRegressor()]

for m in model:
    m.fit(x_train,y_train)
    predm=m.predict(x_test)
    print("Results for Model ",m," is")
    print("R2 Score:",r2_score(y_test,predm))
    print("Error:")
    print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,predm)))
    print("**************************************************************************************************")
    print('\n')
```

```
Results for Model  RandomForestRegressor()  is
R2 Score: 0.8614876506796969
Error:
Root Mean Squared Error: 29097.371198661167
**************************************************************************************************


Results for Model  AdaBoostRegressor()  is
R2 Score: 0.8137546509972596
Error:
Root Mean Squared Error: 33740.55521409976
**************************************************************************************************


Results for Model  GradientBoostingRegressor()  is
R2 Score: 0.8737563241029982
Error:
Root Mean Squared Error: 27778.85382759111
**************************************************************************************************


Results for Model  ExtraTreesRegressor()  is
R2 Score: 0.8662884386269156
Error:
Root Mean Squared Error: 28588.67231611541
**************************************************************************************************
```

Cross Validation

```
Score of LinearRegression() is:


Score : [0.81903449 0.78102319 0.69934982 0.84887922 0.80723587]
Mean Score : 0.791104518310817
Standard deviation: 0.05079129269060956


**************************************************************************************************


Score of Lasso() is:


Score : [0.81903097 0.78102408 0.69935107 0.84887922 0.80723843]
Mean Score : 0.7911047545193821
Standard deviation: 0.05079058406089483


**************************************************************************************************


Score of Ridge() is:


Score : [0.81902083 0.78102667 0.69938095 0.848865   0.80724817]
Mean Score : 0.7911083265524222
Standard deviation: 0.05077595444545829


**************************************************************************************************


Score of ElasticNet() is:


Score : [0.80918293 0.77781431 0.70806647 0.83817777 0.80744757]
Mean Score : 0.7881378103913985
Standard deviation: 0.04435708805880142


**************************************************************************************************


Score of DecisionTreeRegressor() is:


Score : [0.79861754 0.61781672 0.569053   0.69398082 0.68392768]
Mean Score : 0.6726791520602853
Standard deviation: 0.07770332471372014
```

```
Results for Model  RandomForestRegressor()  is
R2 Score: 0.8614876506796969
Error:
Root Mean Squared Error: 29097.371198661167
****************************************************************************


Results for Model  AdaBoostRegressor()  is
R2 Score: 0.8137546509972596
Error:
Root Mean Squared Error: 33740.55521409976
****************************************************************************


Results for Model  GradientBoostingRegressor()  is
R2 Score: 0.8737563241029982
Error:
Root Mean Squared Error: 27778.85382759111
****************************************************************************


Results for Model  ExtraTreesRegressor()  is
R2 Score: 0.8662884386269156
Error:
Root Mean Squared Error: 28588.67231611541
****************************************************************************
```

Key Metrics for success in solving problem under consideration

As the Laaso, Ridge , Gradientboosting & Random Forest Regressor models are giving best results on the dataset further we can find out the best parameters of these models using GrdSearchCV. Using these best params we will be able to improve the overall score & accuracy of the models.

Using GridSearchCV to find out the best parameter in Lasso Regression

81]:
```
from sklearn.model_selection import GridSearchCV
```

82]:
```
#Lasso model is giving the best score so finding its best parameter using GridSearchCV

lasso=Lasso()
parameters={"alpha" :[0.001, 0.01, 0.1, 1], 'random_state':range(42, 100)}
clf = GridSearchCV(lasso, parameters)
clf.fit(x, y)
clf.best_params_
```

82]:
```
{'alpha': 1, 'random_state': 42}
```

83]:
```
# lasso model
lasso=Lasso(alpha= 1, random_state= 42)
lasso.fit(x_train,y_train)
pred=lasso.predict(x_test)
print("R2 Score:",r2_score(y_test,pred))
print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2 Score: 0.8663870135280889
Root Mean Squared Error: 28578.132295844167
```

Using GridSearchCV to find out the best parameter in Ridge Regression

4]:
```
#Ridge model is giving the GOOD score so finding its best parameter using GridSearchCV

ridge=Ridge()
parameters={"alpha" :[0.001, 0.01, 0.1, 1], 'random_state':range(42, 100)}
clf = GridSearchCV(ridge, parameters)
clf.fit(x_train, y_train)
clf.best_params_
```

4]:
```
{'alpha': 1, 'random_state': 42}
```

5]:
```
# APPLYING TO RIDGE ITS BEST PARAMETERS
ridge=Ridge(alpha= 1, random_state= 42)
ridge.fit(x_train,y_train)
pred=ridge.predict(x_test)
print("R2 Score:",r2_score(y_test,pred))
print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2 Score: 0.8663870717978961
Root Mean Squared Error: 28578.126064255146
```

Using GridSearchCV to find out the best parameter in Random Forest Regressor

```
[88]: rfr=RandomForestRegressor()
      parameters={'n_estimators':[50,100,150,200]}
      clf=GridSearchCV(rfr,parameters)
      clf.fit(x_train,y_train)
      clf.best_params_

[88]: {'n_estimators': 50}
```

```
[89]: rfr=RandomForestRegressor(n_estimators=50)
      rfr.fit(x_train,y_train)
      predrfr=rfr.predict(x_test)
      print("R2 Score:",r2_score(y_test,predrfr))
      print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,predrfr)))

      R2 Score: 0.8541875236889113
      Root Mean Squared Error: 29854.297034451127
```

Using GridSearchCV to find out the best parameter in Gradient Boosting Regressor

```
5]: gbr=GradientBoostingRegressor()
    estimator={'n_estimators':[100,200,300]}
    clf=GridSearchCV(gbr,estimator)
    clf.fit(x_train,y_train)
    clf.best_params_

5]: {'n_estimators': 100}
```

```
7]:
    gbr=GradientBoostingRegressor(n_estimators=100)
    gbr.fit(x_train,y_train)
    predgbr=gbr.predict(x_test)
    print("R2 Score:",r2_score(y_test,predgbr))
    print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,predgbr)))

    R2 Score: 0.8738522309889252
    Root Mean Squared Error: 27768.30007299185
```

Choosing the best Model

```
gbr=GradientBoostingRegressor(n_estimators=100)
gbr.fit(x_train,y_train)
predgbr=gbr.predict(x_test)
print("R2 Score:",r2_score(y_test,predgbr))
print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,predgbr)))

R2 Score: 0.8738522309889252
Root Mean Squared Error: 27768.30007299185
```

```
#cross validation

from sklearn.model_selection import cross_val_score

scores=cross_val_score(gbr,x,y,cv=5)
print(scores)
print(scores.mean(),scores.std())

[0.87737106 0.81992534 0.78333827 0.86206944 0.81540749]
0.8316223180891187 0.03391478828735006
```

Predicting the House Price of Test Dataset

Predicting for Test Data

n [92]:
```python
price_pred=gbr.predict(df_test)
```

n [93]:
```python
#lets make the dataframe for price_pred
price_pred=pd.DataFrame(price_pred,columns=["SalePrice"])
```

n [94]:
```python
price_pred
```

ut[94]:

	SalePrice
0	404506.736940
1	151376.344995
2	275187.209727
3	140927.518824
4	268022.092327
...	...
287	271420.403646
288	142333.037958
289	135465.786543
290	142006.745763
291	101607.098084

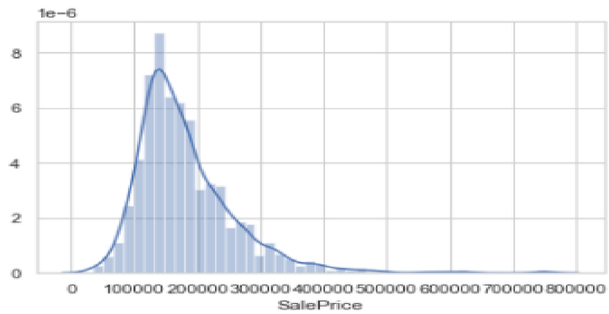
292 rows × 1 columns

Visualizations & Interpretation of the Results

Now we can analyze all the plots, graphs in the dataset and the inferences and observations obtained from them.

Univariate Analysis

Checking the distribution of Target column

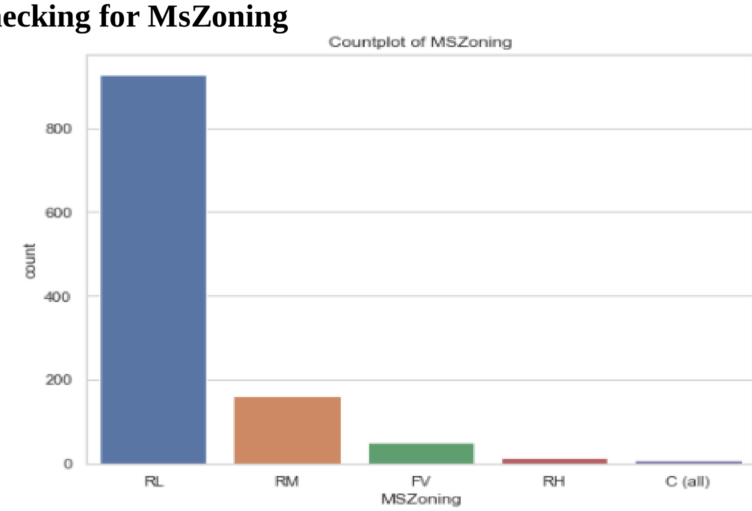



```
140000    18
135000    16
155000    12
139000    11
160000    11
          ..
126175     1
204000     1
186000     1
369900     1
105500     1
Name: SalePrice, Length: 581, dtype: int64
```

Observation:

Maximum number of SalePrice lies between 140000 and 230000.

Checking for MsZoning




```
8]:  RL         928
     RM         163
     FV          52
     RH          16
     C (all)      9
     Name: MSZoning, dtype: int64
```

Observation:

Maximum, 928 number of MSZoning are RL.

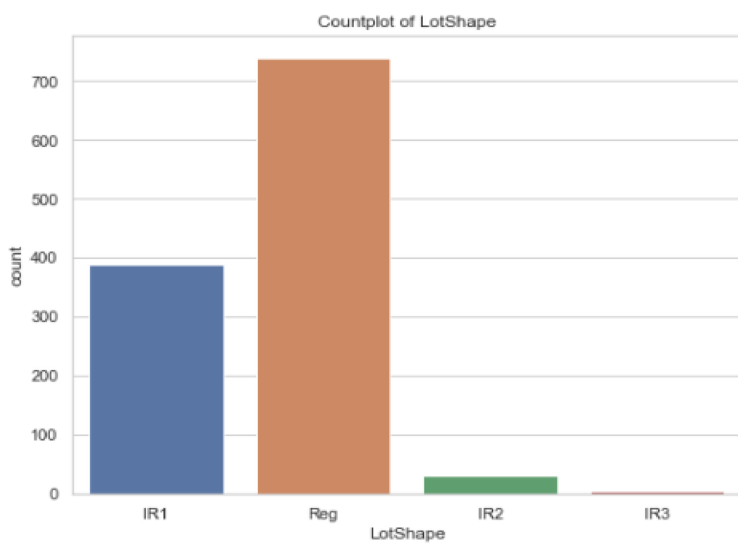
Checking for Street column


```
]: Pave     1164
   Grvl        4
   Name: Street, dtype: int64
```

Observation:

Maximum, 1164 number of Street are Pave where as only 4 are Grvl.

Checking for Lotshape column

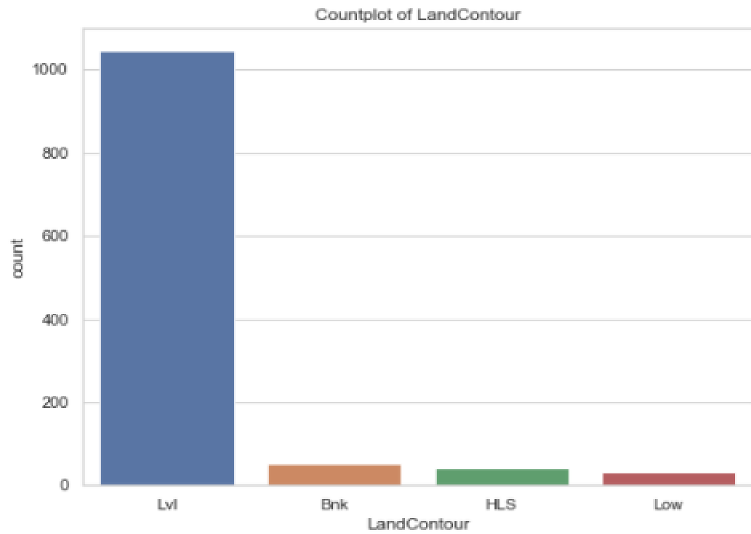



```
20]: Reg     740
     IR1     390
     IR2      32
     IR3       6
     Name: LotShape, dtype: int64
```

Observation:

Maximum, 740 number of LotShape are Reg.

Checking for LandCounter column

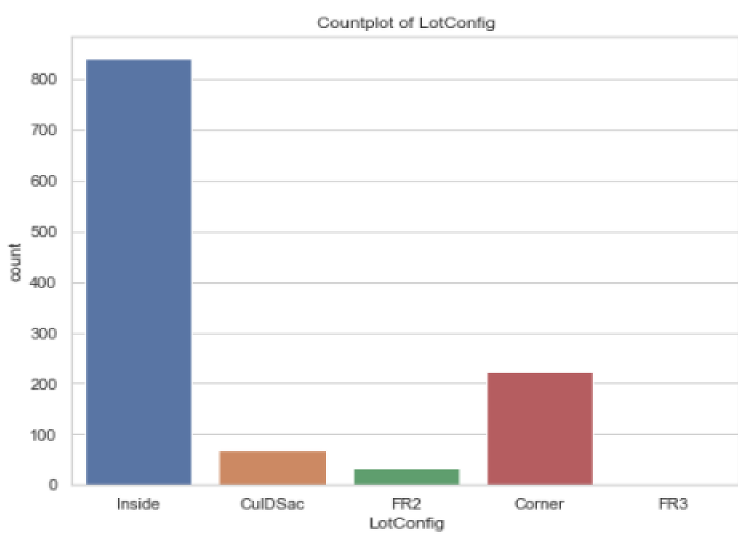



```
21]: Lvl     1046
     Bnk       50
     HLS       42
     Low       30
     Name: LandContour, dtype: int64
```

Observation:

Maximum, 1046 number of LandContour are Lvl.

Checking for Lotconfig column

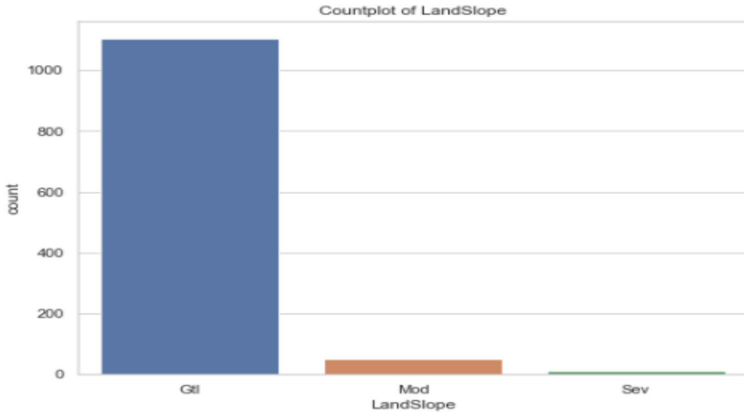



```
22]: Inside     842
     Corner     222
     CulDSac     69
     FR2         33
     FR3          2
     Name: LotConfig, dtype: int64
```

Observation:

Maximum, 842 number of LotConfig are Inside.

Checking for LandSlope column

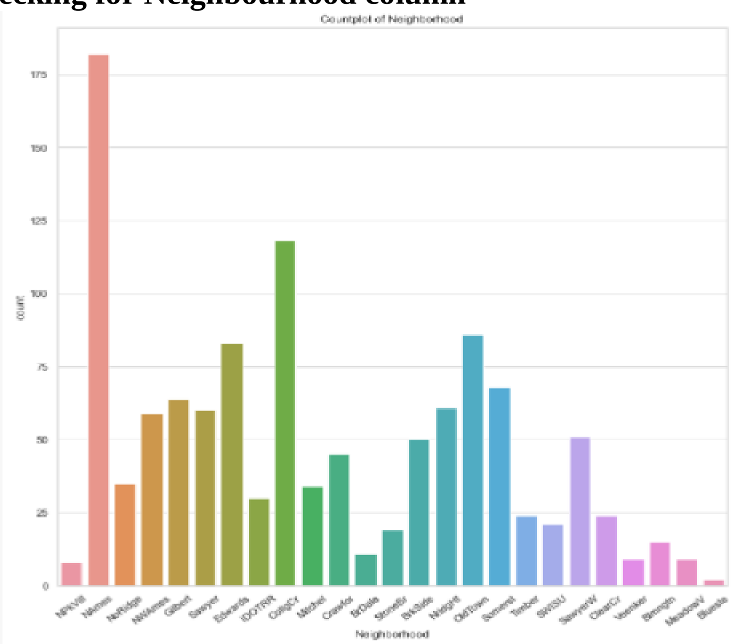



```
:3]: Gtl     1105
     Mod       51
     Sev       12
     Name: LandSlope, dtype: int64
```

Observation:

Maximum, 1105 number of LandSlope are Gtl.

Checking for Neighbourhood column

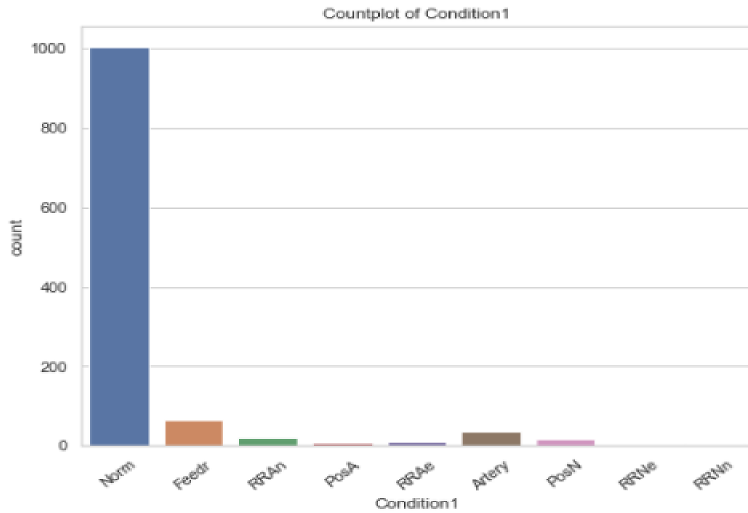



```
[24]: NAmes      182
      CollgCr    118
      OldTown     86
      Edwards     83
      Somerst     68
      Gilbert     64
      NridgHt     61
      Sawyer      60
      NWAmes      59
      SawyerW     51
      BrkSide     50
      Crawfor     45
      NoRidge     35
      Mitchel     34
      IDOTRR      30
      ClearCr     24
      Timber      24
      SWISU       21
      StoneBr     19
      Blmngtn     15
      BrDale      11
      MeadowV      9
      Veenker      9
      NPkVill      8
      Blueste      2
      Name: Neighborhood, dtype: int64
```

Observation:

Maximum, 182 number of Neighborhood are NAmes.

Checking for Condition1 column

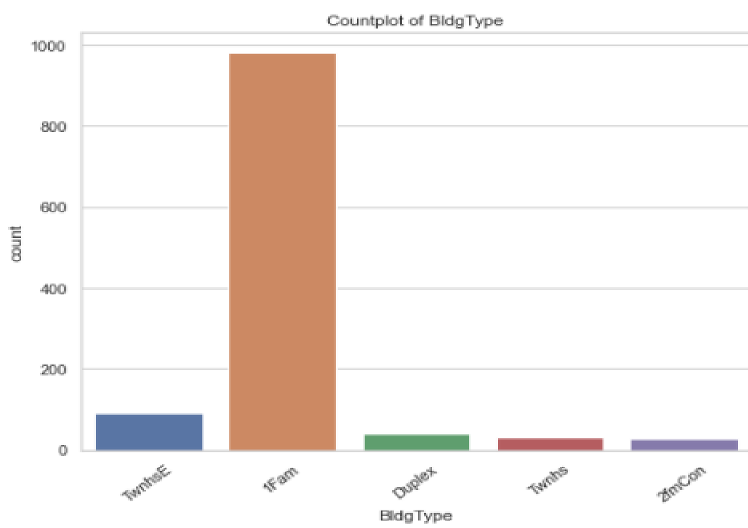



```
Norm      1005
Feedr       67
Artery      38
RRAn        20
PosN        17
RRAe         9
PosA         6
RRNn         4
RRNe         2
Name: Condition1, dtype: int64
```

Observation:

Maximum, 1005 number of Condition1 is Norm.

Checking for Bldgtype column

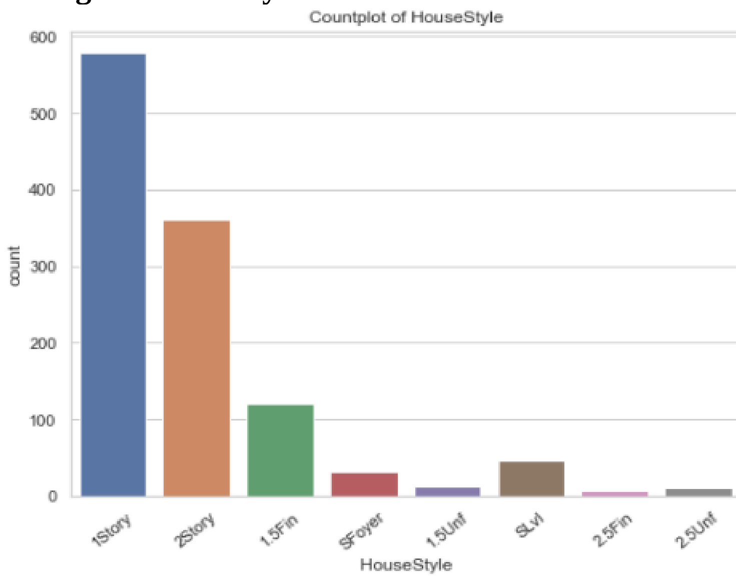



```
1Fam      981
TwnhsE     90
Duplex     41
Twnhs      29
2fmCon     27
Name: BldgType, dtype: int64
```

Observation:

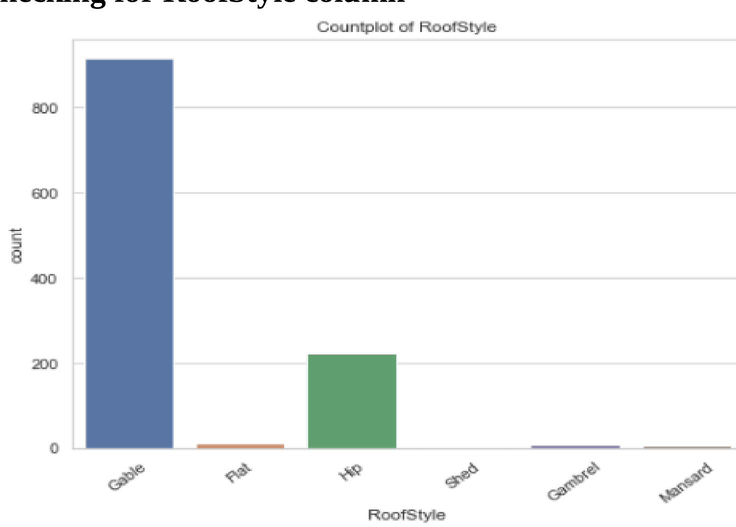
Maximum, 981 number of BldgType are 1Fam.

Checking for HouseStyle column




```
]: 1Story    578
   2Story    361
   1.5Fin    121
   SLvl       47
   SFoyer     32
   1.5Unf     12
   2.5Unf     10
   2.5Fin      7
   Name: HouseStyle, dtype: int64
```

Checking for RoofStyle column

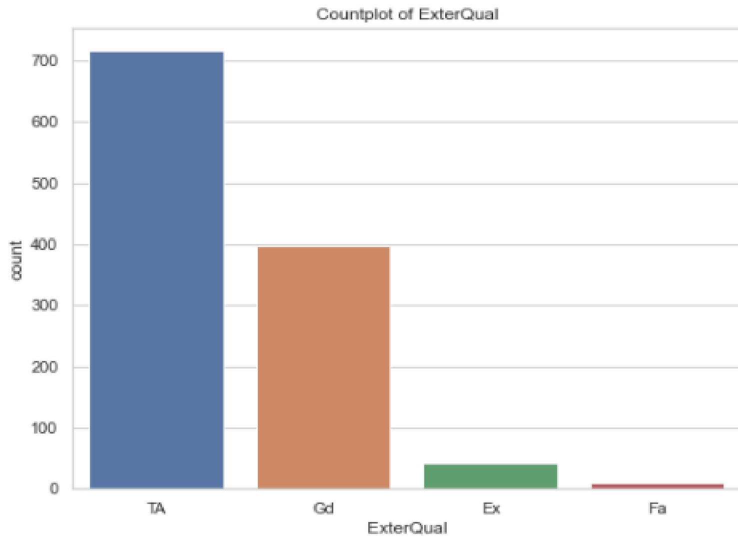



```
Gable      915
Hip        225
Flat        12
Gambrel      9
Mansard      5
Shed         2
Name: RoofStyle, dtype: int64
```

Observation:

Maximum, 915 number of RoofStyle are Gable.

Checking for ExterQual column

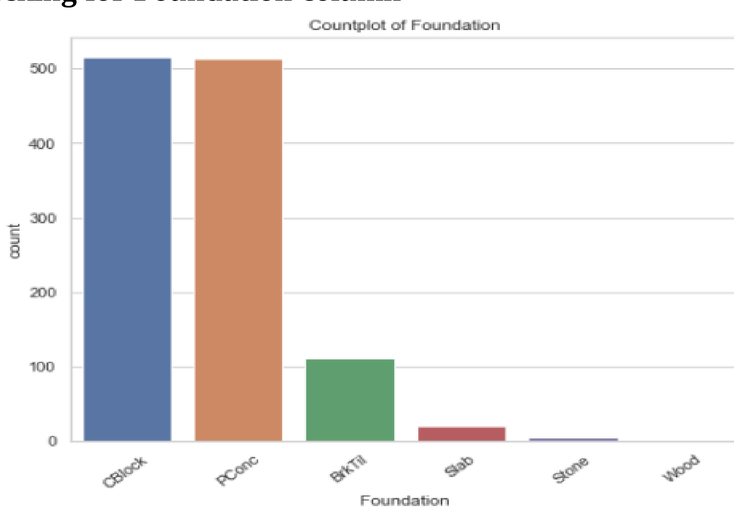



```
TA     717
Gd     397
Ex      43
Fa      11
Name: ExterQual, dtype: int64
```

Observation:

Maximum, 717 number of ExterQual is TA.

Checking for Foundation column

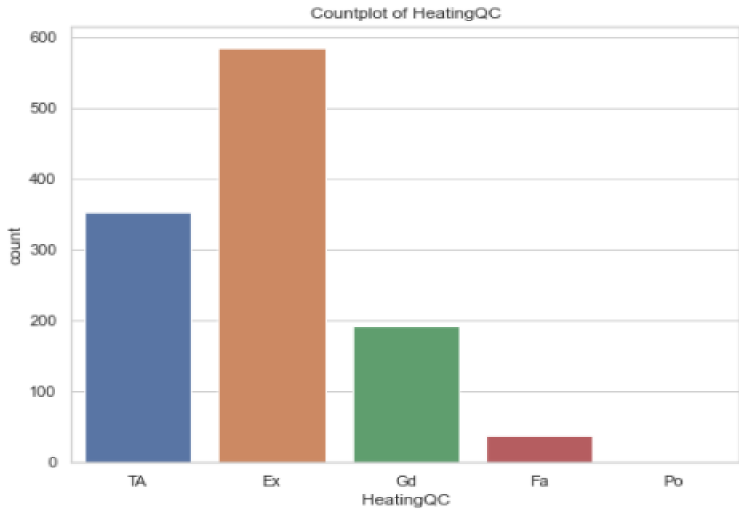



```
0]: CBlock    516
    PConc     513
    BrkTil    112
    Slab       21
    Stone       5
    Wood        1
    Name: Foundation, dtype: int64
```

Observation:

Maximum, 516 number of Foundation are CBlock.

Checking for HeatingQC column

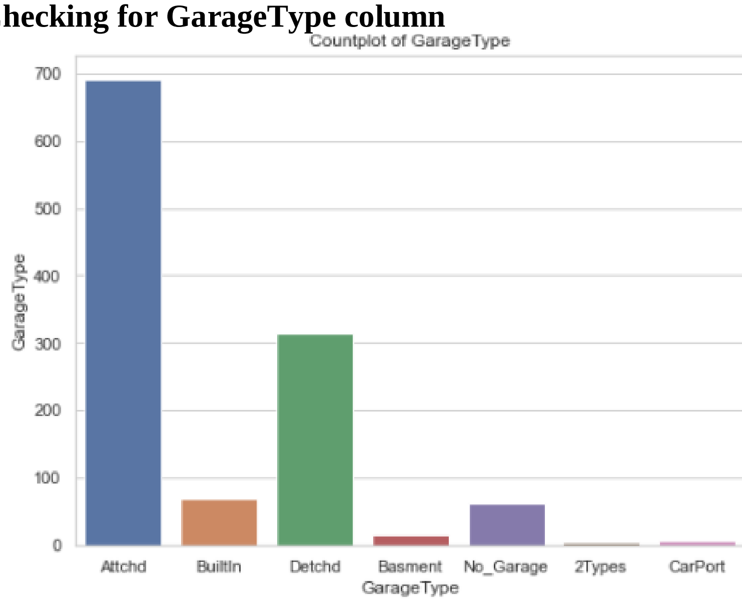



```
Ex     585
TA     352
Gd     192
Fa      38
Po       1
Name: HeatingQC, dtype: int64
```

Observation:

Maximum, 585 number of HeatingQC is Ex.

Checking for GarageType column

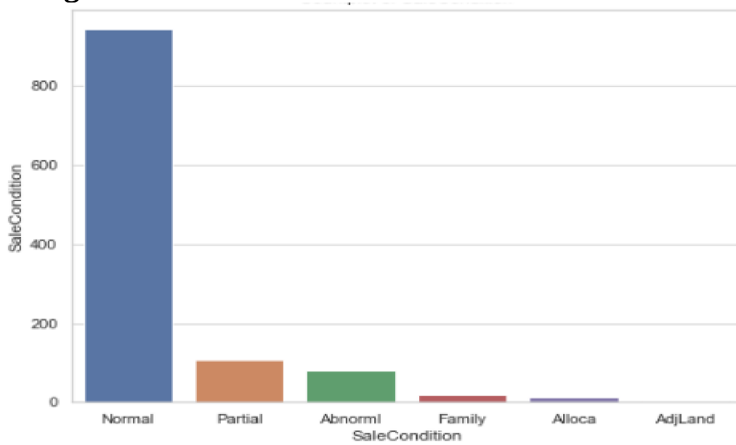



```
Attchd       691
Detchd       314
BuiltIn       70
No_Garage     64
Basment       16
CarPort        8
2Types         5
Name: GarageType, dtype: int64
```

Observation:

Maximum, 691 number of GarageType are Attchd.

Checking for SaleCondition column

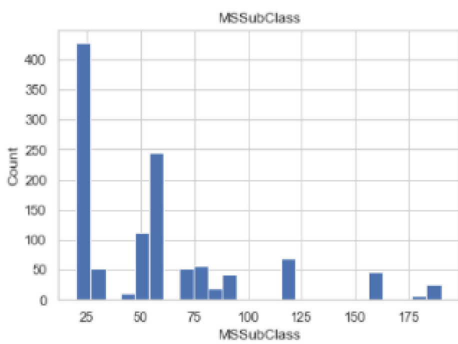


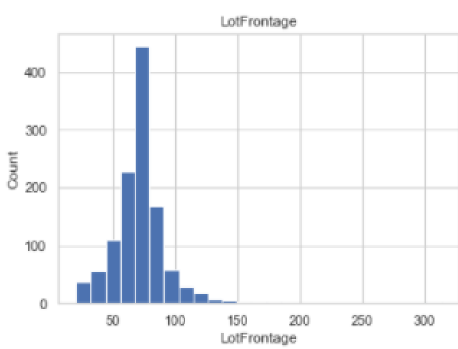

```
3]: Normal      945
    Partial     108
    Abnorml      81
    Family       18
    Alloca       12
    AdjLand       4
    Name: SaleCondition, dtype: int64
```

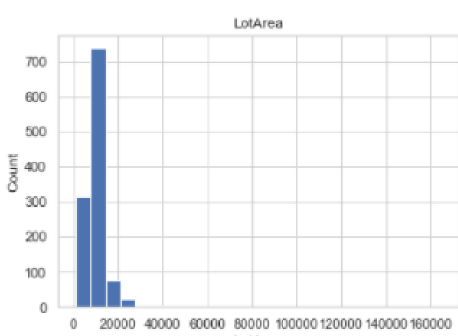
Observation:

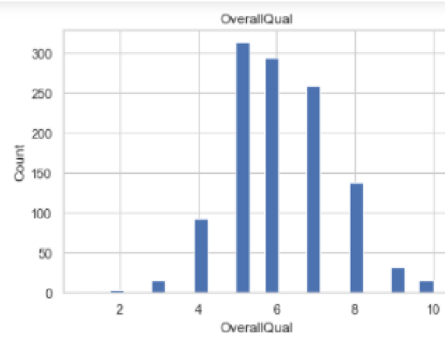
Maximum, 945 number of SaleCondition is normal.

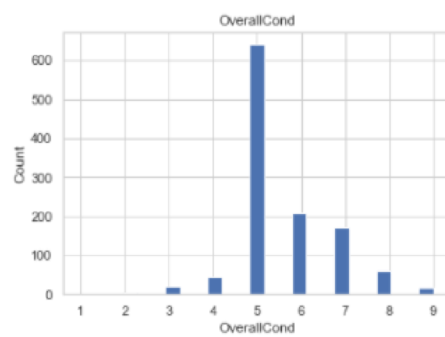
Plotting histogram for numeric column

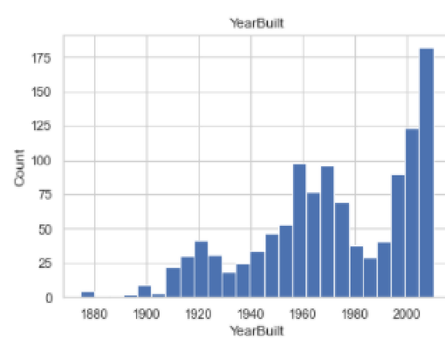


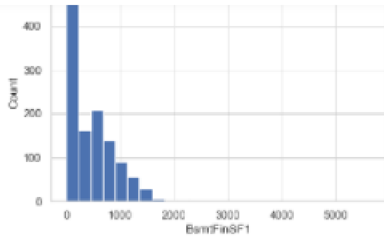


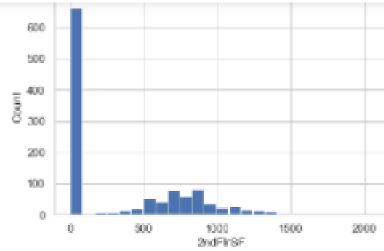


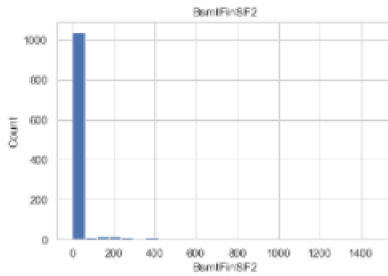


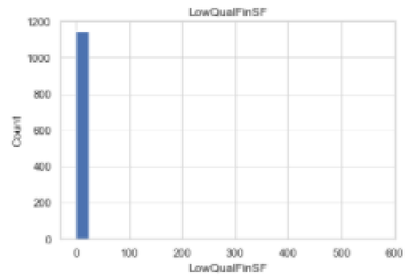


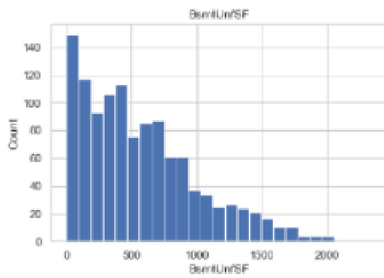


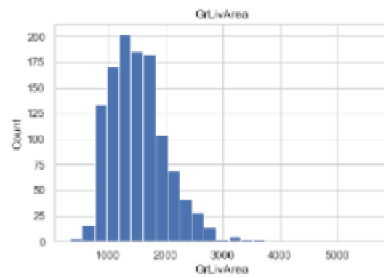

Count
BsmtFinSF1

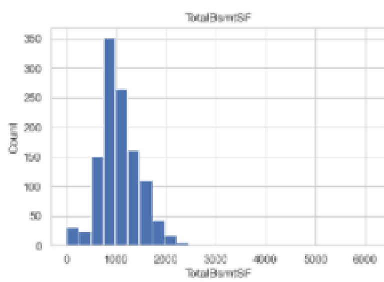

Count
2ndFlrSF

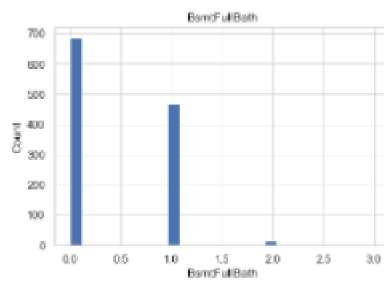

BsmtFinSF2
Count
BsmtFinSF2

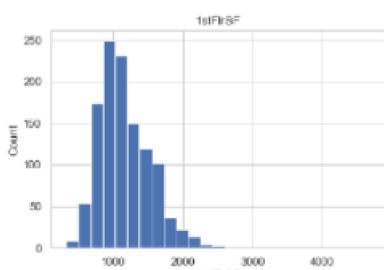

LowQualFinSF
Count
LowQualFinSF

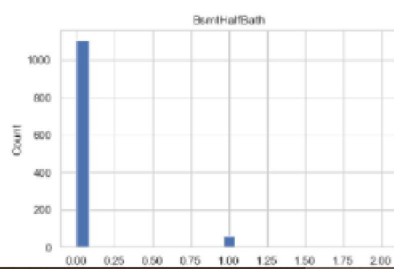

BsmtUnfSF
Count
BsmtUnfSF

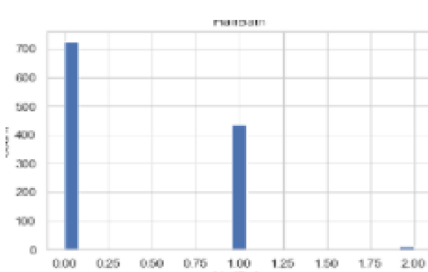

GrLivArea
Count
GrLivArea

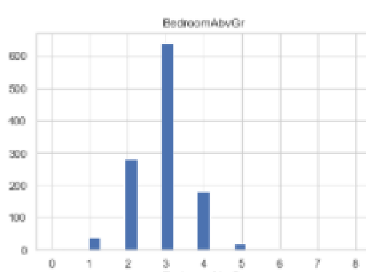

TotalBsmtSF
Count
TotalBsmtSF

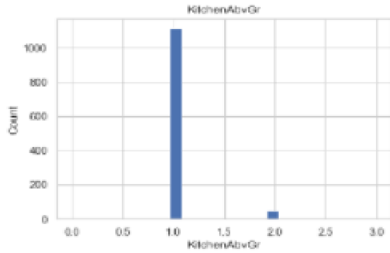

BsmtFullBath
Count
BsmtFullBath

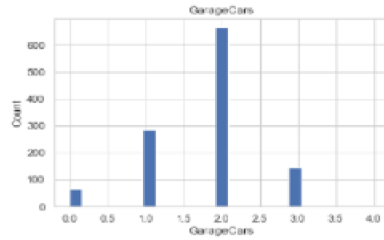

1stFlrSF
Count

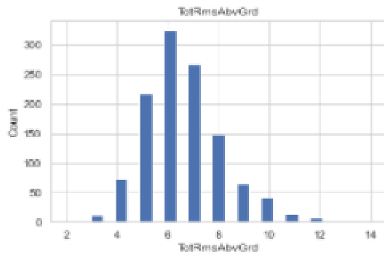

BsmtHalfBath
Count

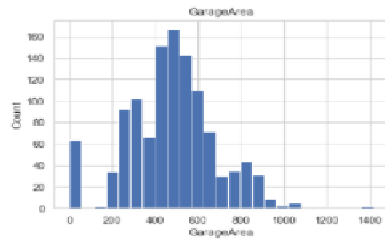

HalfBath

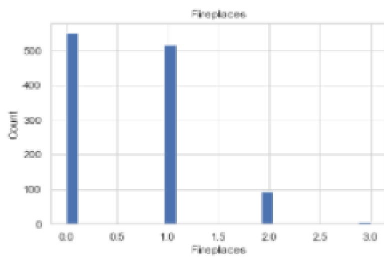

BedroomAbvGr
BedroomAbvGr

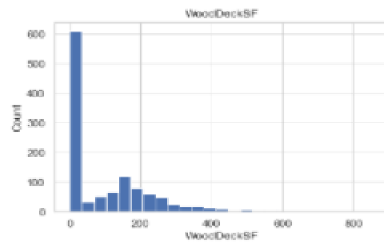

KitchenAbvGr
Count
KitchenAbvGr

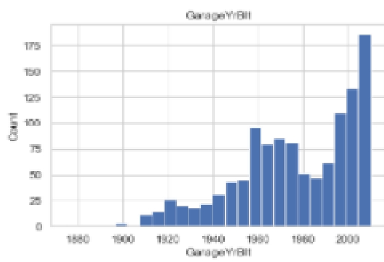

GarageCars
Count
GarageCars

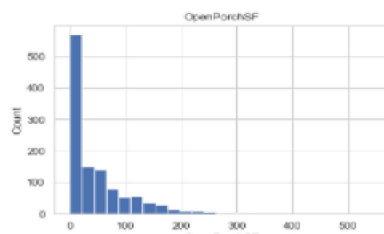

TotRmsAbvGrd
Count
TotRmsAbvGrd

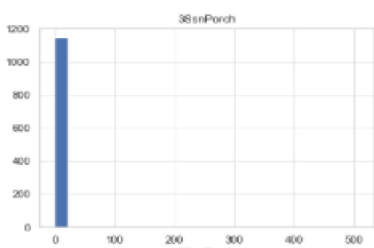

GarageArea
Count
GarageArea

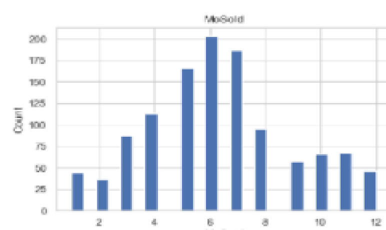

Fireplaces
Count
Fireplaces

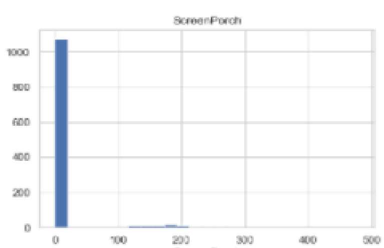

WoodDeckSF
Count
WoodDeckSF

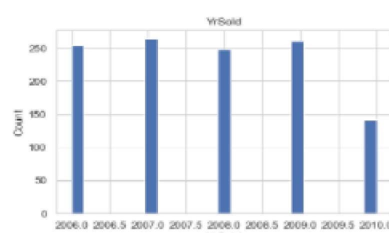

GarageYrBlt
Count
GarageYrBlt

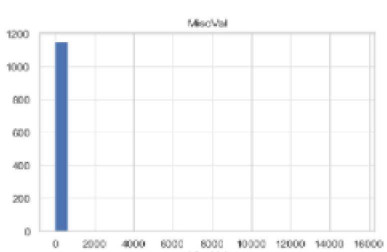

OpenPorchSF
Count
OpenPorchSF


3SsnPorch
3SsnPorch


MoSold
Count
MoSold


ScreenPorch
ScreenPorch

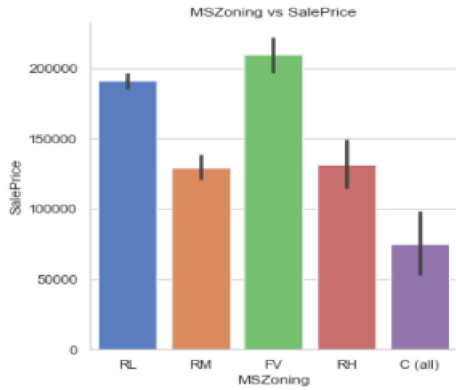

YrSold
Count
YrSold


MiscVal
MiscVal


SalePrice
Count
SalePrice

Bivariate Analysis

Factor plot of MSZoning vs SalePrice

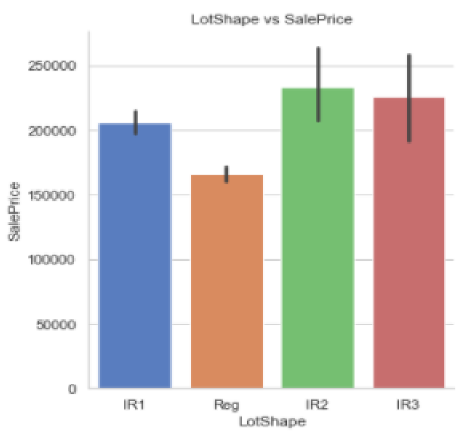



```
SalePrice  MSZoning
34900      C (all)      1
35311      C (all)      1
37900      RM           1
39300      RL           1
40000      C (all)      1
                       ..
582933     RL           1
611657     RL           1
625000     RL           1
745000     RL           1
755000     RL           1
Name: MSZoning, Length: 697, dtype: int64
```

Observation:

SalePrice is maximum with FV MSZOning.

Factor plot of LotShape vs SalePrice

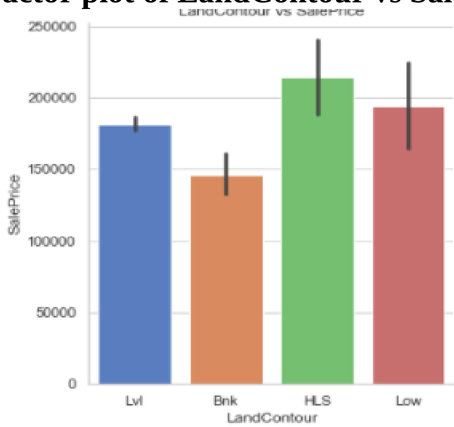



```
SalePrice  LotShape
34900      Reg          1
35311      Reg          1
37900      Reg          1
39300      Reg          1
40000      Reg          1
                       ..
582933     Reg          1
611657     IR1          1
625000     IR1          1
745000     IR1          1
755000     IR1          1
Name: LotShape, Length: 733, dtype: int64
```

Observation:

SalePrice is maximum with IR2 LotShape.

Factor plot of LandContour vs SalePrice

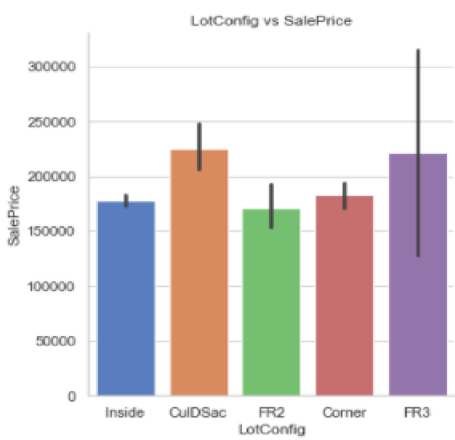



```
SalePrice  LandContour
34900      Lvl            1
35311      Lvl            1
37900      Lvl            1
39300      Low            1
40000      Lvl            1
                         ..
582933     Lvl            1
611657     Lvl            1
625000     Lvl            1
745000     Lvl            1
755000     Lvl            1
Name: LandContour, Length: 655, dtype: int64
```

Observation:

SalePrice is maximum with HLS LandContour.

Factor plot of LotConfig vs SalePrice

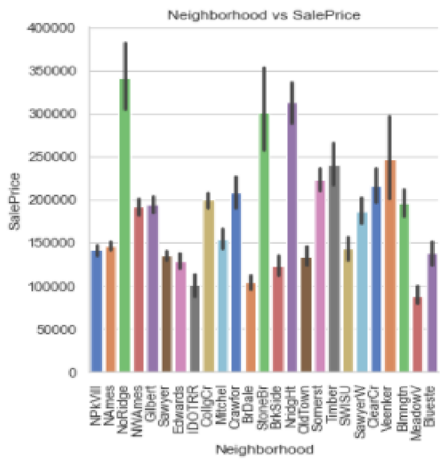



```
SalePrice  LotConfig
34900      Inside        1
35311      Inside        1
37900      Inside        1
39300      Inside        1
40000      Inside        1
                        ..
582933     Inside        1
611657     Inside        1
625000     CulDSac       1
745000     Corner        1
755000     Corner        1
Name: LotConfig, Length: 743, dtype: int64
```

Observation:

SalePrice is maximum with CulDsac LotConfig.

Factor plot of Neighbourhood vs SalePrice

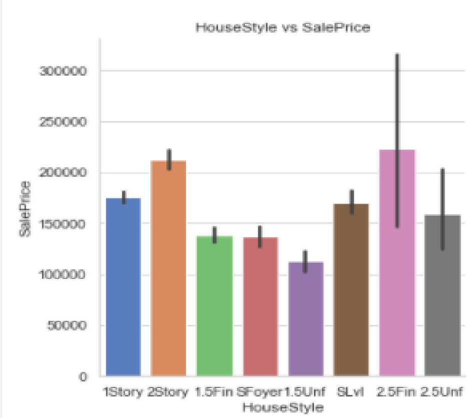



```
SalePrice  Neighborhood
34900      IDOTRR          1
35311      IDOTRR          1
37900      OldTown         1
39300      BrkSide         1
40000      IDOTRR          1
                          ..
582933     NridgHt         1
611657     NridgHt         1
625000     NoRidge         1
745000     NoRidge         1
755000     NoRidge         1
Name: Neighborhood, Length: 1013, dtype: int64
```

Observation:

SalePrice is maximum with NoRidge Neighborhood.

Factor plot of HouseStyle vs SalePrice

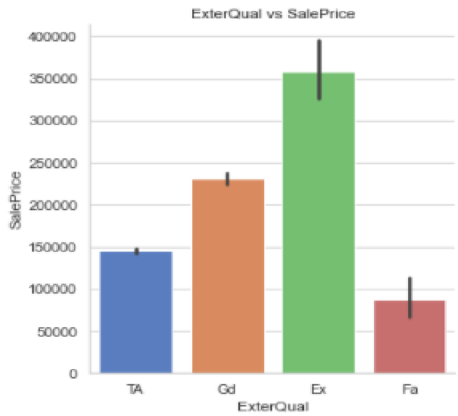



```
SalePrice  HouseStyle
34900      1Story        1
35311      1Story        1
37900      1.5Fin        1
39300      1Story        1
40000      2Story        1
                        ..
582933     2Story        1
611657     1Story        1
625000     2Story        1
745000     2Story        1
755000     2Story        1
Name: HouseStyle, Length: 840, dtype: int64
```

Observation:

SalePrice is maximum with 2.5Fin HouseStyle.

Factor plot of ExterQual vs SalePrice

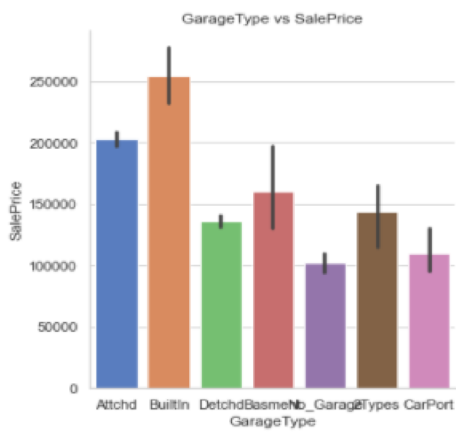



```
SalePrice  ExterQual
34900      TA           1
35311      TA           1
37900      TA           1
39300      Fa           1
40000      TA           1
                       ..
582933     Ex           1
611657     Ex           1
625000     Gd           1
745000     Gd           1
755000     Ex           1
Name: ExterQual, Length: 679, dtype: int64
```

Observation:

SalePrice is maximum with Ex ExterQual.

Factor plot of GarageType vs SalePrice

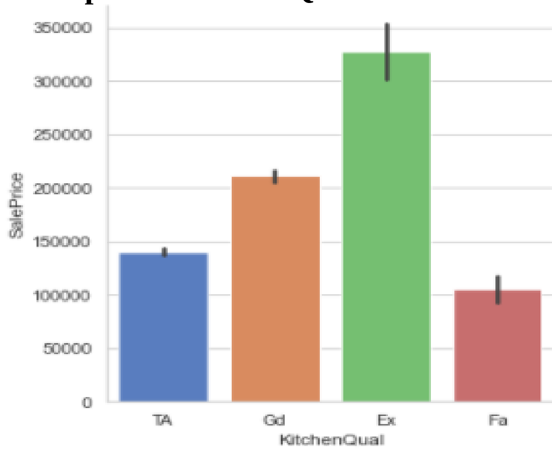



```
SalePrice  GarageType
34900      No_Garage     1
35311      Detchd        1
37900      No_Garage     1
39300      No_Garage     1
40000      Detchd        1
                        ..
582933     BuiltIn       1
611657     Attchd        1
625000     Attchd        1
745000     Attchd        1
755000     Attchd        1
Name: GarageType, Length: 762, dtype: int64
```

Observation:

SalePrice is maximum with BuiltIn GarageType.

Factor plot of KitchenQual vs SalePrice

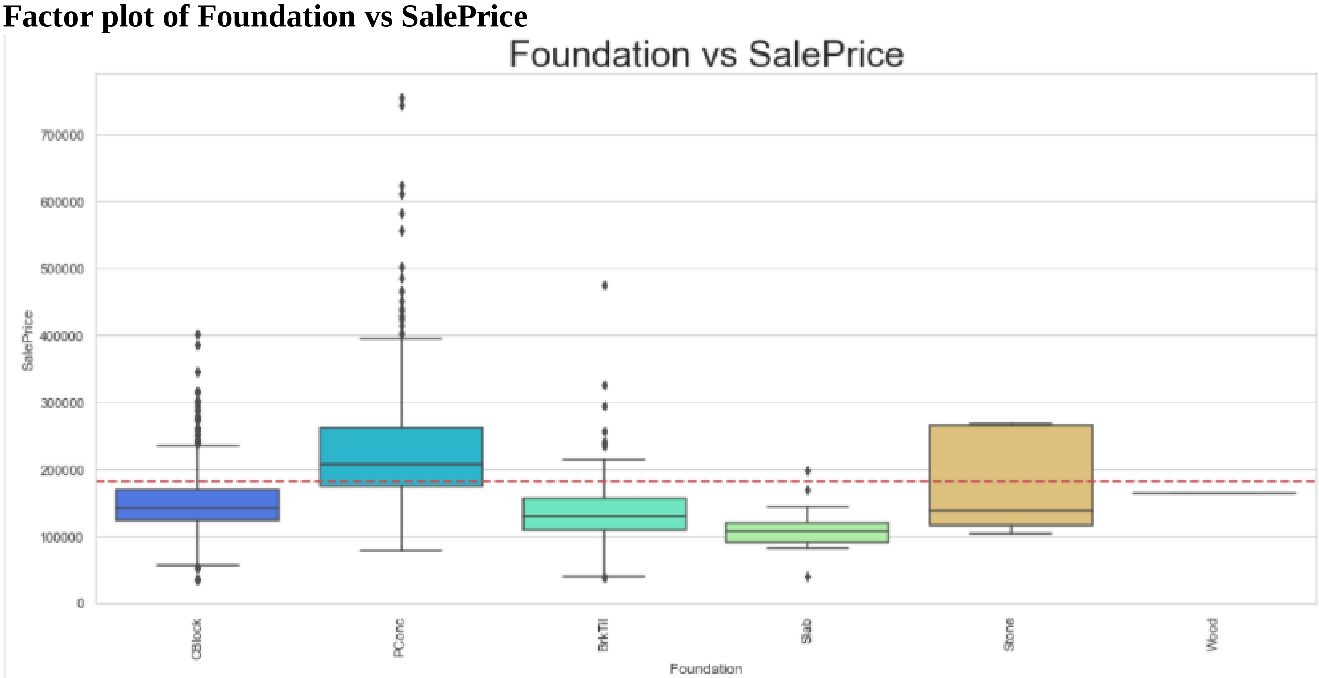



```
SalePrice  KitchenQual
34900      TA            1
35311      TA            1
37900      TA            1
39300      Fa            1
40000      TA            1
                        ..
582933     Ex            1
611657     Ex            1
625000     Gd            1
745000     Ex            1
755000     Ex            1
Name: KitchenQual, Length: 710, dtype: int64
```

Observation:

SalePrice is maximum with Ex PoolQC.

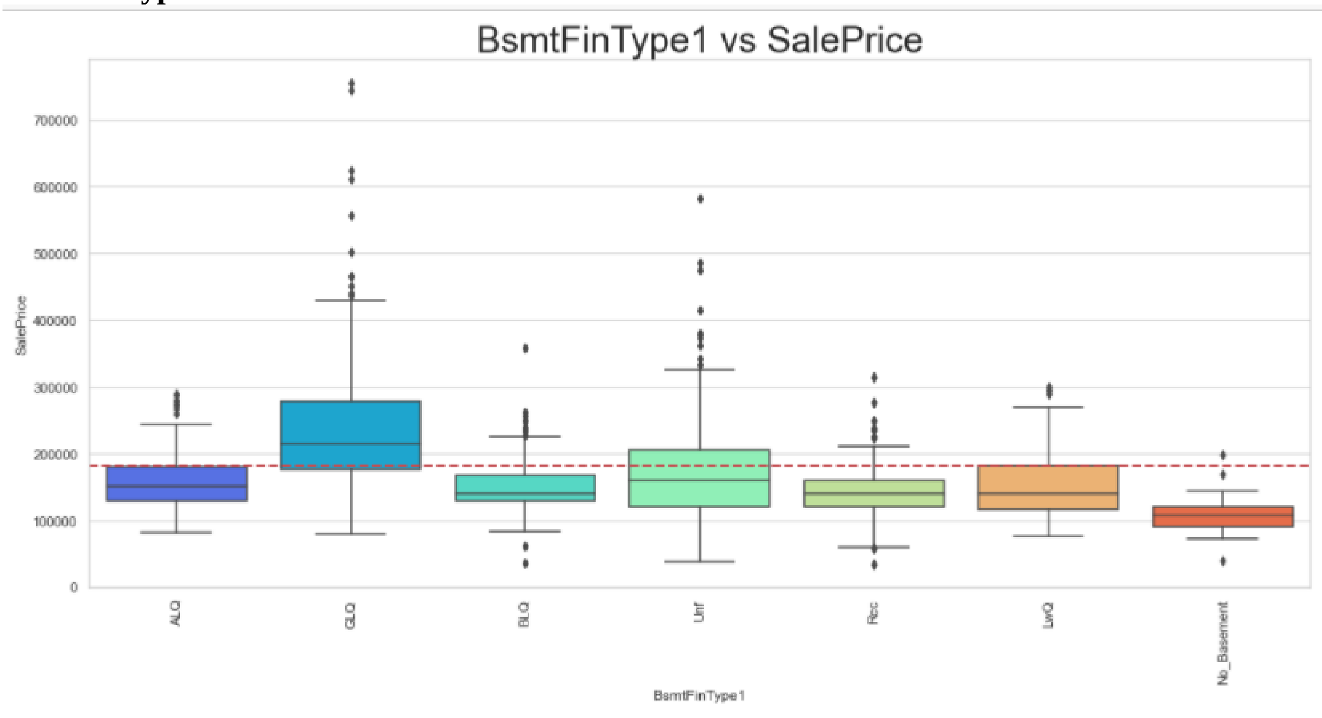
Factor plot of Foundation vs SalePrice



Observation:

SalePrice is maximum with PConc.

BsmtFinType1 vs SalePrice

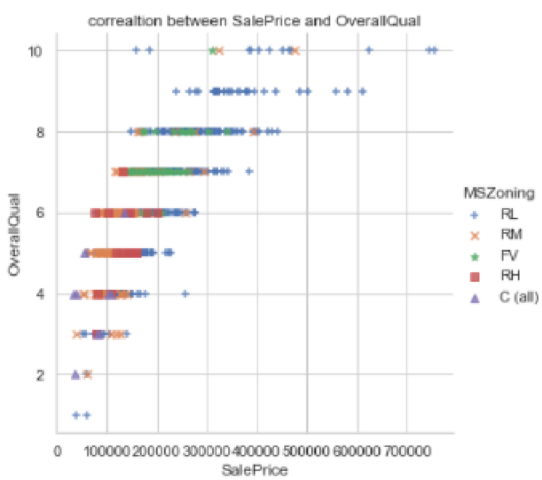


Observation:

SalePrice is maximum with GLQ BsmtFinType1.

Multivariate Analysis

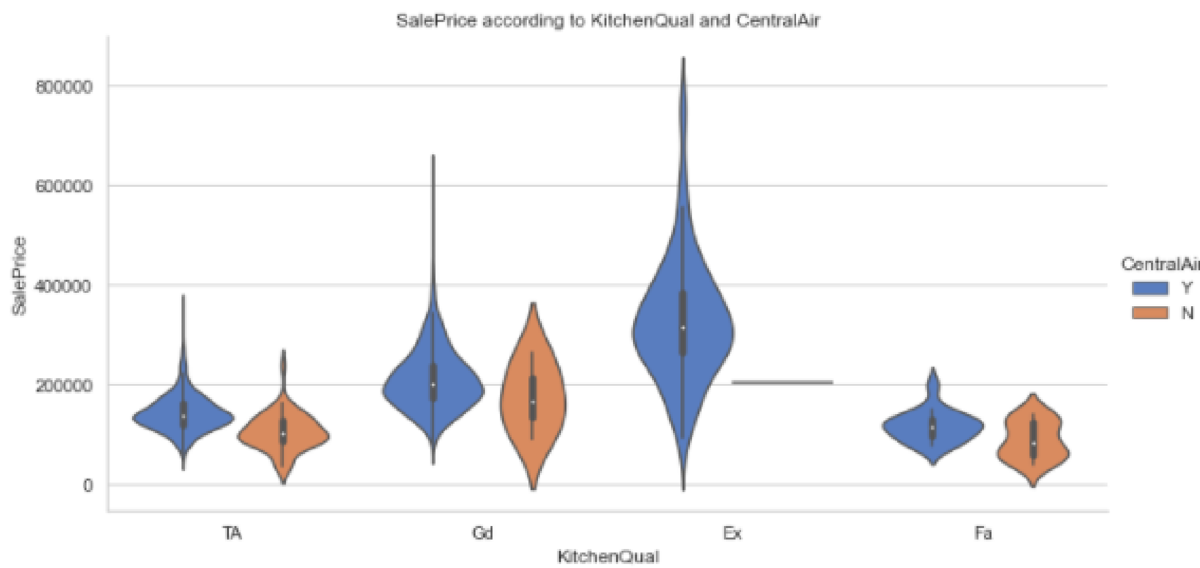
Scatter plot between SalePrice and OverallCond with respect to MSZoning



Observation:

With MSZoning RL and increase in OverallQual the SalePrice of a house increases.

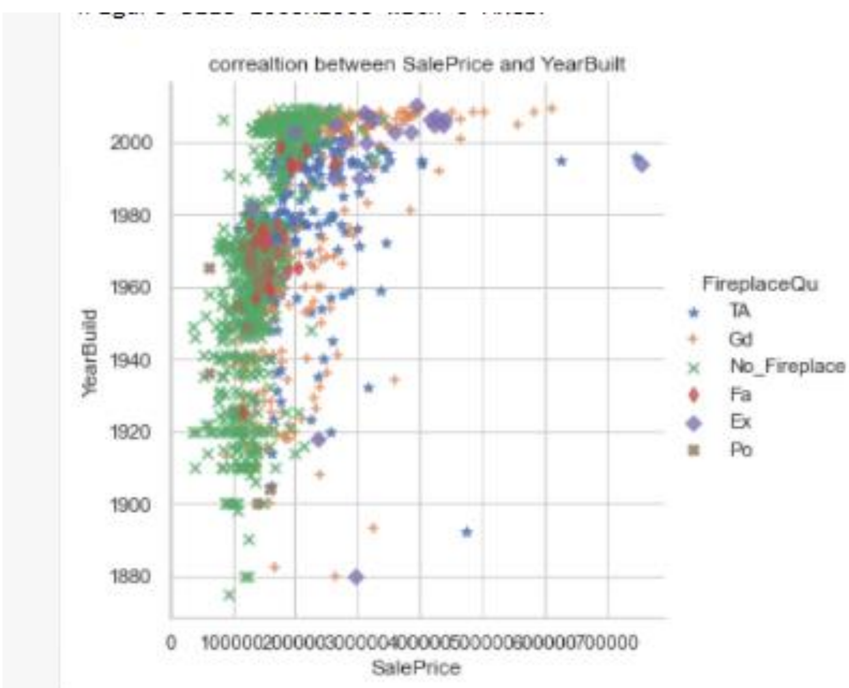
Checking KitchenQual and CentralAir with respect to SalePrice



Observation:

SalePrice is maximum with Ex kitchenQual and CentralAir.

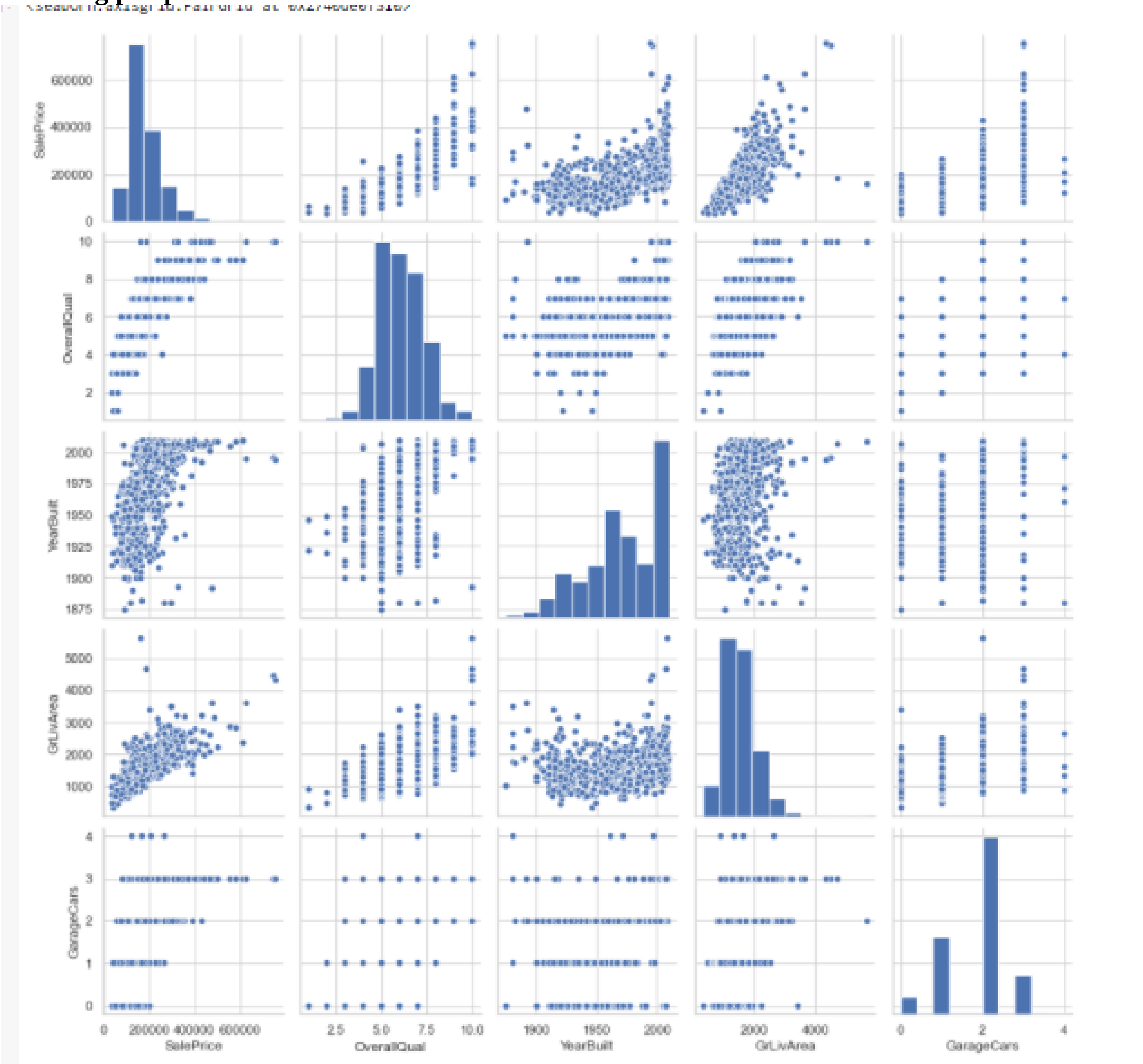
Scatter plot between SalePrice and OverallCond with respect to MSZoning



Observation:

As the YearBuilt is increasing SalePrice is also increasing.

Plotting pairplot



Observation:

SalePrice is highly positively correlated with GrLivArea and OverallQual.

CONCLUSION

- The GradientBoosting Regressor model is working with highest R2 score of 87.385.
- The cross_val_score of model is 83.299359 which again show that cross val score is better compared to other models in the table.
- All these points prove that GradientBoosting Regressor model is working best and can beconsidered as finalised model.

Learning Outcomes of the Study in respect of Data Science

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

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

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

Limitations of this work and Scope for Future Work

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

Example: MSZoning has 8 categories

```
A  Agriculture
C  Commercial
FV Floating Village Residential
I  Industrial
RH Residential High Density
RL Residential Low Density
RP Residential Low Density Park
RM Residential Medium Density
```

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

References

1. State of the Nation's housing report – US housing supply falls short of what is needed – Chris Herbert

2. Affordable Housing in the United States - By Gordon Davis, Jaime Bordenave, Roger Williams, Richard A. Hanson, and Richard Shields - June 12, 2006

3. Real Estate Value Prediction Using Linear Regression - Nehal N Ghosalkar, Sudhir N Dhage

4. Housing Affordability Literature Review and Affordable Housing Program Audit - Elena Sliogeris, Louise Crabtree, Peter Phibbs and Kate Johnston

5. Social Housing Finance in Australia as a Missing or Incomplete Market - George Earl, Judy Kraatz, Benjamin Liu, Sherif Mohamed, Eduardo Roca.

6. Investment in Social Housing Finance in Australia - Nirodha Jayawardena & Griffith University, Australia

THANKS