HOUSE PRICE PREDICTION

Submitted by:

SRIVIDYA.V

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Shubam Yadav as well as Flip Robo Technologies who gave me the opportunity to do this project on HOUSE PRICE PREDICTION & also helping me to gain in-depth knowledge of Machine Learning and DataScience to derive insights for organizational goals or meet business needs.

Also, I have utilized a few external resources that helped me to complete this project. All the external resources that were used in creating this project are listed below:

https://stackoverflow.com/questions

https://medium.com/

https://www.kaggle.com/

https://www.geeksforgeeks.org/

https://www.codegrepper.com/

https://www.analyticsvidhya.com/

https://towardsdatascience.com/

https://github.com/

INTRODUCTION

Business Problem Framing

Problem Overview

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 company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

Conceptual Background of the Domain Problem

MACHINE LEARNING AND DATA SCIENCE FOR BUSINESS:

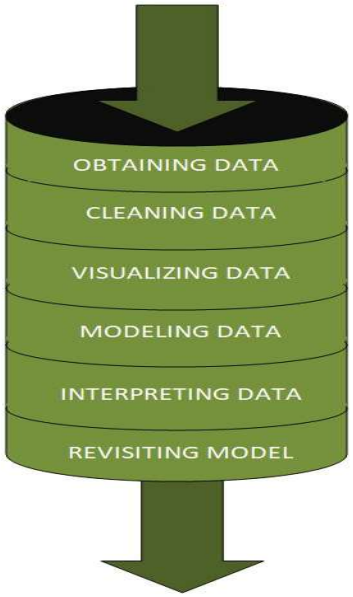
Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn from experience, make predictions and gradually improving its accuracy. It is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive

decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data science will increase, requires to assist in the identification of the most relevant business questions and subsequently the data to answer them. Following are the ways Data science can add value to Business :

- Empowering management and officers to make better decision
- Directing actions based on trends—which in turn help to define goals
- Challenging the staff to adopt best practices and focus on issues that matter
- Identifying opportunities
- Decision making with quantifiable, data-driven evidence
- Testing these decisions
- Identification and refining of target audiences

DATASCIENCE PIPELINE:

The data science pipeline is a collection of connected tasks that aims at delivering an insightful data science product or service to the business organization. The responsibilities include collecting, cleaning, exploring, modeling, interpreting the data, and other processes of the launching of the product. This final product can be used for to achieve Business Goals.



Exploratory Data Analysis:

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

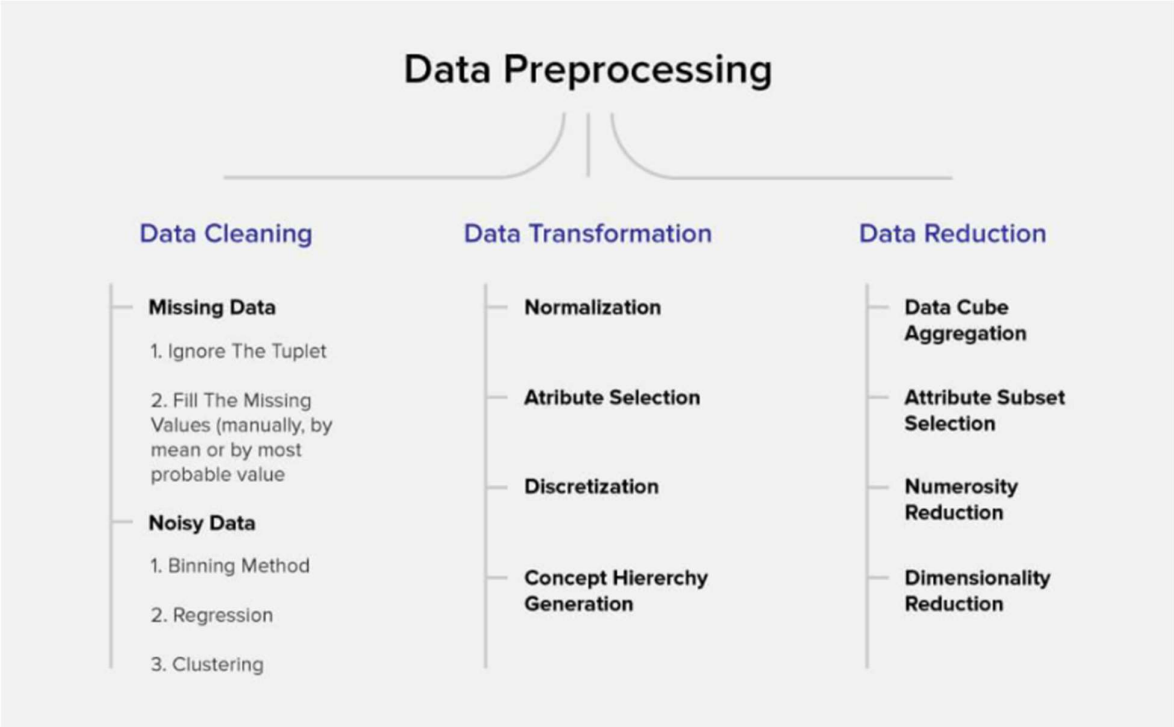
Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions

TYPES OF EXPLORATORY DATA ANALYSIS:

- Univariate Non-graphical
- Multivariate Non-graphical
- Univariate graphical
- Multivariate graphical

DATA PRE-PROCESSING & FEATURE ENGINEERING:

Preprocessing simply refers to perform series of operations to transform or change data. It is transformation applied to our data before feeding it to algorithm. When creating a machine learning project, and doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.



Data pre-processing is a very vital input to machine learning models, It is to prepare the raw data & make it suitable for efficient machine learning model. These are the methods of data preprocessing and we are going to use the required ones in our project.

FEATURE ENGINEERING:

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning. In order to make machine learning work well on new tasks, it might be necessary to design and train better features. As you may know, a “feature” is any measurable input that can be used in a predictive model.

Feature engineering, in simple terms, is the act of converting raw observations into desired features using statistical or machine learning approaches. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.

Feature Engineering Techniques for Machine Learning

- Imputation
- Handling Outliers
- Log Transform
- One-hot encoding/Label Encoding
- Scaling

Data Transformation:

Label Encoding:

As we mentioned above in library installation, Label Encoder is used to encode labels by assigning them numbers. It is used to encode single or multiple columns. Thus, if the feature is color with values such as ['white', 'red', 'black', 'blue']., using Label Encoder may encode color string label as [0, 1, 2, 3]

Handling Outliers:

The most important phase in Feature Engineering is handling outliers because it ensures that our model is trained on accurate data which leads to accurate models. An outlier may occur due to the variability in the data. It may indicate an experimental error or heavy skewness in the data(heavy-tailed distribution). We have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

Below are some of the techniques of detecting outliers

- Boxplots
- Z-score

Variance Inflation Factor (VIF)

Variance Inflation Factors (VIFs) measure the correlation among independent variables in least squares regression models. Statisticians refer to this type of correlation as multicollinearity. Excessive multicollinearity can cause problems for regression models. The stats models package has VIF library, Let us import the package.

SKEWNESS REMOVAL-(POWER-TRANSFORM):

Key step prior to initiating Machine learning models, optimizing, scaling the data to provide it as a input to start the modelling.

A power transform will make the probability distribution of a variable more Gaussian. This is often described as removing a skew in the distribution, although more generally is described as stabilizing the variance of the distribution. The log transform is a specific example of a family of transformations known as power transforms. The power_transform library present in the Sklearn. Pre-processing package.

MINMAX SCALER:

MinMax Scaler shrinks the data within the given range, usually of 0 to 1. It transforms data by scaling features to a given range. It scales the values to a specific value range without changing the shape of the original distribution.
Before scaling we have to train test split the data.since we have to do skewness removal and scaling only on input data.

TRAIN TEST SPLIT:

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train_test_split() function. The function takes a loaded dataset as input and returns the dataset split into two subsets.train_test_split() will split arrays data into random subsets. The ideal split is said to be 80:20 for training and testing.

Review of Literature

ABSTRACT:

In this House Price Prediction Project, we are going to predict the price of the houses in Australia with collection records of the sale of houses in Australia. We are doing this prediction for the US-based housing company named Surprise Housing.This company has decided to enter the Australian market to purchase houses at a price below their actual values and flip them at a higher price.So our detailed analysis, Machine Learning Model predictions done can be used for specific Business Requirements, Challenges and Improvements of the d Surprise Housing Company.

Motivation for the Problem Undertaken

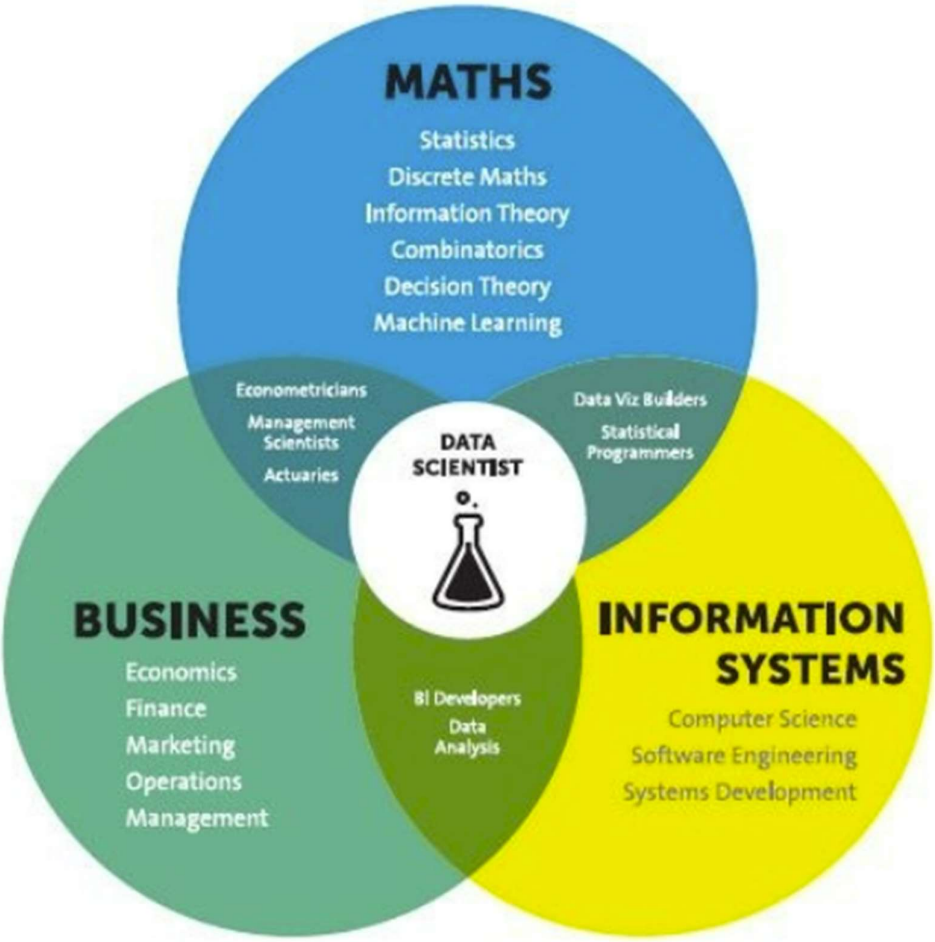
Business Goal:

We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Analytical Problem Framing

Mathematical/ Statistical /Analytical Modeling of the Problem

Mathematics, Statistics and Analytics are three of the most important concepts of Data Science. Data Science revolves around these three fields and draws their concepts to operate on the data.we will explore its practical usages in this field. So let's first explore how much these three are required for data science.



Mathematical Modelling

Mathematical models are important, selecting the right one to answer the business question can bring tremendous value to the organization. Machine Learning is a field that focuses on computers

having the ability to learn/operate without being programmed to do so.

Mathematics is playing an essential role in the latest technologies like Machine Learning, Artificial Intelligence, Data Science and Deep Learning, etc., It is because every algorithm built in the latest technologies has a mathematical function behind it and aid in identifying patterns.

The understanding of various notions of Statistics and Probability Theory are key for the implementation of such algorithms in data science. Notions include: Regression, Maximum Likelihood Estimation, the understanding of distributions (Binomial, Bernoulli, Gaussian (Normal)) and Bayes' Theorem.

The main reason for a greater significance of mathematics is because of its various concepts like: –

· Linear Algebra

· Probability

· Calculus

· Statistics

Linear Algebra & Calculus

Deep learning requires us to understand linear algebra & calculus, to understand how it works, for example forward propagation, backward propagation, parameters setting etc. For linear algebra, there are matrix operations (plus, minus, times, divide), scalar product, dot product, eigen-vectors and eigenvalues.

It is a branch of Mathematics for studying systems of equations. it can be one, two, and multi-dimensional equations. it helps us to solve numerical data or relations between two or more variables by establishing relations or equations between them. for example,

here' one basic algebraic equation:

$$y = a + bx + cx2$$

linear-algebra has a wide range of applications such as statics and matrices calculations, linear regression equations, descriptive statistics, graphic image vectors, Fourier series, graphs, and network establishment.

machine-learning algorithms like linear regression, logistic regression uses linear algebra to solve our target variables with given inputs/attributes or feature vectors given in the data set.

Calculus

Calculus is used essentially in optimization techniques. Using calculus, you can carry out mathematical modeling of artificial neural networks and also increase their accuracy and performance. For calculus, the data scientist need to understand various differentiation (to second-order derivative), integration, partial differentiation.

Differential Calculus

Differential Calculus studies the rate at which the quantities change. Derivates are most widely used for finding the maxima and minima of the functions. Derivates are used in optimization techniques where we have to find the minima in order to minimize the error function.

Integral Calculus

It is the mathematical study of the accumulation of quantities and for finding the area under the curve. Integrals are further divided into definite integrals and indefinite integrals.

Probability

The probability theory is very much helpful for making the prediction and Estimation.With the help of statistical methods, we make estimates for the further analysis. Thus, statistical methods are largely dependent on the theory of probability.

Probability is a very important mathematical concept for data science, used in validating hypothesis, bayes theorem and interpreting outputs in machine learning.

Bases on these we try to estimate various events, and the likelihood of the outcome. sometimes we wat graphical representations of probable outcomes which we call probability density functions or density curves.

Concepts of probability help us estimate expected value from given variables, to solve confusion matrix in classification algorithms, information entropy, evidence of particular attributes in naive Bayes classification, and even in statistics for hypothesis testings.

Statistics

A statistical model is a mathematical representation (or mathematical model) of observed data. When data analysts apply various statistical models to the data they are investigating, they are able to understand and interpret the information more strategically.

So the areas in statistics are simple statistics like measurement of centrality, distributions and different probability distributions (Weibull, Poisson etc), Baye's Theorem

statistics is divided into two –

- Descriptive Statistics
- Inferential Statistics

Descriptive Statistics

Descriptive Statistics or summary statistics is used for describing the data. It deals with the quantitative summarization of data. This summarization is performed through graphs or numerical representations.

Descriptive Statistics:

1) Mean, Median, Mode

2) IQR, percentiles

3) Std deviation and Variance

4) Normal Distribution

5) Z-statistics and T-statistics

6) correlation and linear regression

Inferential Statistics

It is the procedure of inferring or concluding from the data. Through inferential statistics, we make a conclusion about the larger population by running several tests and deductions from the smaller sample.

Inferential Statistics:

1) Sampling distributions

2) confidence interval

3) chi-square test

4) Advanced regression

5) ANOVA

The mathematical concepts noted above are key in understanding/implementing the following Machine Learning techniques.

- Supervised learning, including regression and classification models.
- Unsupervised learning, including clustering algorithms and association rules.

Regression Models

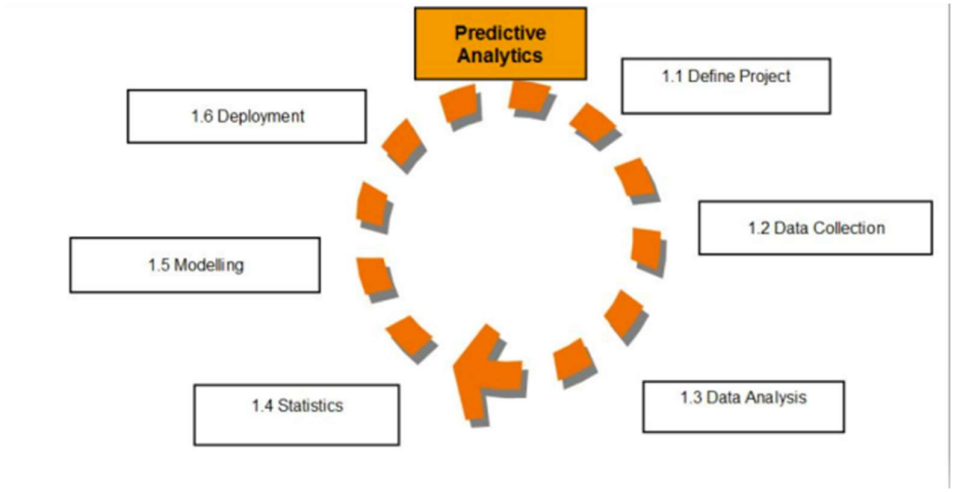
Data analysts use **regression models** to examine relationships between variables. Regression models are often used by organizations to determine which independent variables hold the most influence over dependent variables—information that can be leveraged to make essential business decisions.

Classification Models

Classification is a process in which an algorithm is used to analyze an existing data set of known points. The understanding achieved through that analysis is then leveraged as a means of appropriately classifying the data. Classification is a form of machine learning that can be particularly helpful in analyzing very large, complex sets of data to help make more accurate predictions.

Analytical Models:

An analytical model estimates or classifies data values by essentially drawing a line through data points. When applied to new data or records, a model can predict outcomes based on historical patterns.



. An analytical model is quantitative in nature, and used to answer a specific question or make a specific design decision. Different analytical models are used to address different aspects of the system, such as its performance, reliability, or mass properties.Data analysis comes with the fundamental types of data analytics encounter in data science: Descriptive, Diagnostic, Predictive, and Prescriptive.

- Descriptive analytics is a statistical method that is used to search and summarize historical data in order to identify patterns or meaning.
- Descriptive analysis is often used when reviewing any past or present data. This is because raw data is difficult to consume and interpret, while the metrics offered by descriptive analysis are much more focused.
- The example of descriptive statistics or analytics is to calculate the mean, median mode, standard deviation, and similar kinds of statistical calculation on finance or sales data.

- Diagnostic analytics takes it a step further to uncover the reasoning behind certain results. Diagnostic analytics is usually performed using such techniques as data discovery, drill-down, data mining, and different type of bivariant data analysis like correlations.etc.,
- Predictive Analytics is a **statistical method that utilizes algorithms and machine learning to identify trends in data and predict future behaviors**. Predictive Analytics can take both past and current data and offer predictions of what could happen in the future.
- Predictive models typically utilize variability in data to make the correct prediction and more variability of ingredient data that shows the relationship

with what is possible to predict that united together into a prediction or valid score.

- Prescriptive analytics automatically synthesizes big data, mathematical sciences, business rules, algorithms, and machine learning to make predictions and then suggests decision options to take advantage of the predictions. Prescriptive means (optimization and simulation).

Data Sources and their formats

Technical Requirements:

• Data contains 1460 entries each having 81 variables.

• Data contains Null values. We treated them using the domain knowledge and our own understanding.

• Extensive EDA has been performed to gain relationships of important variable and price.

• Data contains numerical as well as categorical variable. We handled them accordingly.

• We built Machine Learning models, applied regularization and determined the optimal values of Hyper Parameters.

• We found important features which affect the price positively or negatively.

• Two datasets are provided (test.csv, train.csv). We had done training on train.csv dataset and prediction on test.csv file.

The datasets are enclosed in notebook file

The dataset is provided to us by FlipRobo Technologies.And the dataset is in .csv file format.

Data Description:

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

```
20    1-STORY 1946 & NEWER ALL STYLES

30    1-STORY 1945 & OLDER

40    1-STORY W/FINISHED ATTIC ALL AGES

45    1-1/2 STORY - UNFINISHED ALL AGES

50    1-1/2 STORY FINISHED ALL AGES

60    2-STORY 1946 & NEWER

70    2-STORY 1945 & OLDER
```

```
 75    2-1/2 STORY ALL AGES

 80    SPLIT OR MULTI-LEVEL

 85    SPLIT FOYER

 90    DUPLEX - ALL STYLES AND AGES

120    1-STORY PUD (Planned Unit Development) - 1946 & NEWER

150    1-1/2 STORY PUD - ALL AGES

160    2-STORY PUD - 1946 & NEWER

180    PUD - MULTILEVEL - INCL SPLIT LEV/FOYER

190    2 FAMILY CONVERSION - ALL STYLES AND AGES
```

MSZoning: Identifies the general zoning classification of the sale.

```
A    Agriculture

C    Commercial

FV    Floating Village Residential

I    Industrial

RH    Residential High Density

RL    Residential Low Density

RP    Residential Low Density Park

RM    Residential Medium Density
```

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

```
Grvl    Gravel

Pave    Paved
```

Alley: Type of alley access to property

```
Grvl    Gravel

Pave    Paved

NA     No alley access
```

LotShape: General shape of property

```
Reg    Regular
```

```
       IR1    Slightly irregular

       IR2    Moderately Irregular

       IR3    Irregular
```

LandContour: Flatness of the property

```
       Lvl    Near Flat/Level

       Bnk    Banked - Quick and significant rise from street grade
   to building

       HLS    Hillside - Significant slope from side to side

       Low    Depression
```

Utilities: Type of utilities available

```
       AllPub    All public Utilities (E,G,W,& S)

       NoSewr    Electricity, Gas, and Water (Septic Tank)

       NoSeWa    Electricity and Gas Only

       ELO    Electricity only
```

LotConfig: Lot configuration

```
       Inside    Inside lot

       Corner    Corner lot

       CulDSac    Cul-de-sac

       FR2    Frontage on 2 sides of property

       FR3    Frontage on 3 sides of property
```

LandSlope: Slope of property

```
       Gtl    Gentle slope

       Mod    Moderate Slope

       Sev    Severe Slope
```

Neighborhood: Physical locations within Ames city limits

```
       Blmngtn    Bloomington Heights

       Blueste    Bluestem

       BrDale    Briardale

       BrkSide    Brookside

       ClearCr    Clear Creek
```

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

Condition1: Proximity to various conditions

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

```
       RRAe    Adjacent to East-West Railroad
```

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

```
       Artery    Adjacent to arterial street

       Feedr    Adjacent to feeder street

       Norm    Normal

       RRNn    Within 200' of North-South Railroad

       RRAn    Adjacent to North-South Railroad

       PosN    Near positive off-site feature--park, greenbelt, etc.

       PosA    Adjacent to postive off-site feature

       RRNe    Within 200' of East-West Railroad

       RRAe    Adjacent to East-West Railroad
```

BldgType: Type of dwelling

```
       1Fam    Single-family Detached

       2FmCon    Two-family Conversion; originally built as one-fami
    ly dwelling

       Duplx    Duplex

       TwnhsE    Townhouse End Unit

       TwnhsI    Townhouse Inside Unit
```

HouseStyle: Style of dwelling

```
       1Story    One story

       1.5Fin    One and one-half story: 2nd level finished

       1.5Unf    One and one-half story: 2nd level unfinished

       2Story    Two story

       2.5Fin    Two and one-half story: 2nd level finished

       2.5Unf    Two and one-half story: 2nd level unfinished

       SFoyer    Split Foyer

       SLvl    Split Level
```

OverallQual: Rates the overall material and finish of the house

```
       10    Very Excellent

       9    Excellent
```

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

OverallCond: Rates the overall condition of the house

10 Very Excellent

9 Excellent

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

YearBuilt: Original construction date

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

RoofStyle: Type of roof

Flat Flat

Gable Gable

Gambrel Gabrel (Barn)

Hip Hip

Mansard Mansard

Shed Shed

RoofMatl: Roof material

ClyTile Clay or Tile

CompShg Standard (Composite) Shingle

Membran Membrane

Metal Metal

Roll Roll

Tar&Grv Gravel & Tar

WdShake Wood Shakes

WdShngl Wood Shingles

Exterior1st: Exterior covering on house

AsbShng Asbestos Shingles

AsphShn Asphalt Shingles

BrkComm Brick Common

BrkFace Brick Face

CBlock Cinder Block

CemntBd Cement Board

HdBoard Hard Board

ImStucc Imitation Stucco

MetalSd Metal Siding

Other Other

Plywood Plywood

PreCast PreCast

Stone Stone

Stucco Stucco

VinylSd Vinyl Siding

Wd Sdng Wood Siding

WdShing Wood Shingles

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

AsbShng Asbestos Shingles

AsphShn Asphalt Shingles

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

MasVnrType: Masonry veneer type

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

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

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

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

```
       Ex    Excellent

       Gd    Good

       TA    Average/Typical

       Fa    Fair

       Po    Poor
```

Foundation: Type of foundation

```
       BrkTil    Brick & Tile

       CBlock    Cinder Block

       PConc    Poured Contrete

       Slab    Slab

       Stone    Stone

       Wood    Wood
```

BsmtQual: Evaluates the height of the basement

```
       Ex    Excellent (100+ inches)

       Gd    Good (90-99 inches)

       TA    Typical (80-89 inches)

       Fa    Fair (70-79 inches)

       Po    Poor (<70 inches

       NA    No Basement
```

BsmtCond: Evaluates the general condition of the basement

```
       Ex    Excellent

       Gd    Good

       TA    Typical - slight dampness allowed

       Fa    Fair - dampness or some cracking or settling

       Po    Poor - Severe cracking, settling, or wetness

       NA    No Basement
```

BsmtExposure: Refers to walkout or garden level walls

```
       Gd    Good Exposure

       Av    Average Exposure (split levels or foyers typically scor
   e average or above)
```

```
Mn    Mimimum Exposure

No    No Exposure

NA    No Basement
```

BsmtFinType1: Rating of basement finished area

```
GLQ    Good Living Quarters

ALQ    Average Living Quarters

BLQ    Below Average Living Quarters

Rec    Average Rec Room

LwQ    Low Quality

Unf    Unfinshed

NA    No Basement
```

BsmtFinSF1: Type 1 finished square feet

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

```
GLQ    Good Living Quarters

ALQ    Average Living Quarters

BLQ    Below Average Living Quarters

Rec    Average Rec Room

LwQ    Low Quality

Unf    Unfinshed

NA    No Basement
```

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

```
Floor    Floor Furnace

GasA    Gas forced warm air furnace

GasW    Gas hot water or steam heat

Grav    Gravity furnace

OthW    Hot water or steam heat other than gas
```

```
       Wall    Wall furnace
```

HeatingQC: Heating quality and condition

```
       Ex    Excellent

       Gd    Good

       TA    Average/Typical

       Fa    Fair

       Po    Poor
```

CentralAir: Central air conditioning

```
       N    No

       Y    Yes
```

Electrical: Electrical system

```
       SBrkr    Standard Circuit Breakers & Romex

       FuseA    Fuse Box over 60 AMP and all Romex wiring (Average)

       FuseF    60 AMP Fuse Box and mostly Romex wiring (Fair)

       FuseP    60 AMP Fuse Box and mostly knob & tube wiring (poor)

       Mix    Mixed
```

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

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

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

```
       Ex    Excellent

       Gd    Good
```

```
       TA    Typical/Average

       Fa    Fair

       Po    Poor
```

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

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

```
       Typ    Typical Functionality

       Min1    Minor Deductions 1

       Min2    Minor Deductions 2

       Mod    Moderate Deductions

       Maj1    Major Deductions 1

       Maj2    Major Deductions 2

       Sev    Severely Damaged

       Sal    Salvage only
```

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

```
       Ex    Excellent - Exceptional Masonry Fireplace

       Gd    Good - Masonry Fireplace in main level

       TA    Average - Prefabricated Fireplace in main living area o
   r Masonry Fireplace in basement

       Fa    Fair - Prefabricated Fireplace in basement

       Po    Poor - Ben Franklin Stove

       NA    No Fireplace
```

GarageType: Garage location

```
       2Types    More than one type of garage

       Attchd    Attached to home

       Basment    Basement Garage

       BuiltIn    Built-In (Garage part of house - typically has roo
   m above garage)

       CarPort    Car Port

       Detchd    Detached from home
```

NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished

RFn Rough Finished

Unf Unfinished

NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

GarageCond: Garage condition

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

PavedDrive: Paved driveway

Y Paved

P Partial Pavement

N Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

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

Fence: Fence quality

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

MiscFeature: Miscellaneous feature not covered in other categories

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

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

```
WD     Warranty Deed - Conventional
CWD    Warranty Deed - Cash
```

VWD Warranty Deed - VA Loan

New Home just constructed and sold

COD Court Officer Deed/Estate

Con Contract 15% Down payment regular terms

ConLw Contract Low Down payment and low interest

ConLI Contract Low Interest

ConLD Contract Low Down

Oth Other

SaleCondition: Condition of sale

Normal Normal Sale

Abnorml Abnormal Sale - trade, foreclosure, short sale

AdjLand Adjoining Land Purchase

Alloca Allocation - two linked properties with separate de eds, typically condo with a garage unit

Family Sale between family members

Partial Home was not completed when last assessed (associa ted with New Homes)

DATA ACQUISITION

```
In [4]: df.to_csv("HOUSING_PRICE _PREDICTION.csv",sep='\t')

In [5]: df
```

Out[5]:

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

1168 rows × 81 columns

FEATURE DESCRIPTION:

Here Id,MSSubClass,LotFrontage,LotArea,Neighborhood,Condition1,Condition2,HouseStyle,OverallQua,OverallCond,YearBuilt,YearRemodAdd,Exterior1st,Exterior2nd,MasVnrArea,BsmtFinSF1,BsmtFinType2,BsmtFinSF2,BsmtUnfSF,TotalBsmtSF,1stFlrSF,2ndFlrSF,LowQualFinSF,GrLivArea,BedroomAbvGr,TotRmsAbvGrd,Functional,GarageType,GarageYrBlt,GarageArea,WoodDeckSF,OpenPorchSF,EnclosedPorch,3SsnPorch,ScreenPorch,PoolArea,MiscVal,MoSold,SaleType are all the columns which has categorical ordinal data type.

MSZoning,Street,Alley,LotShape,LandContour,Utilities,LotConfig,LandSlope,BldgType,RoofStyle,RoofMatl,MasVnrType,ExterQual,ExterCond,Foundation,BsmtQual,BsmtCond,BsmtExposure,BsmtFinType1,Heating,HeatingQC,CentralAir,Electrical,BsmtFullBath,BsmtHalfBath, FullBath,HalfBath,KitchenAbvGr,KitchenQual,Fireplaces,FireplaceQu,GarageFinish,GarageCars,GarageQual,GarageCond,PavedDrive, PoolQC,Fence,MiscFeature,YrSold,SaleCondition are all the columns which has categorical nominal data type.

Here Our Target column is the SalePrice column which is we are going to predict.Hence our problem is the Regression problem.

```
In [7]: print("Train data size is : {}".format(df.shape))
        print("Test data size is : {}".format(df1.shape))

        Train data size is : (1168, 81)
        Test data size is : (292, 80)
```

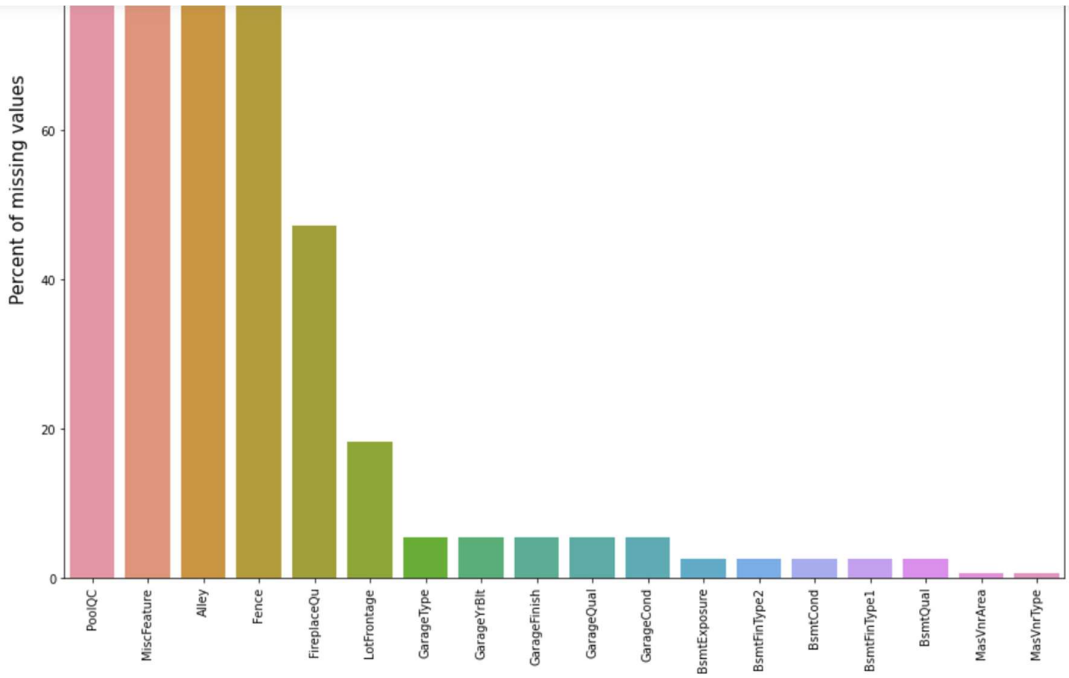
```
In [6]: df.columns

Out[6]: Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
               'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
               'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
               'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
               'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
               'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
               'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
               'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
               'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
               'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
               'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
               'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
               'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
               'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
               'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
               'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
               'SaleCondition', 'SalePrice'],
              dtype='object')
```

Data Preprocessing Done

In [16]:
```
# Find Missing Ratio of Dataset
df_null= (df.isnull().sum() / len(df)) * 100
df_null= df_null.drop(df_null[df_null == 0].index).sort_values(ascending=False)
missing_data = pd.DataFrame({'Missing Ratio' :df_null})
missing_data
```

Out[16]:

	Missing Ratio
PoolQC	99.400685
MiscFeature	96.232877
Alley	93.407534
Fence	79.708904
FireplaceQu	47.174658
LotFrontage	18.321918
GarageType	5.479452
GarageYrBlt	5.479452
GarageFinish	5.479452
GarageQual	5.479452
GarageCond	5.479452
BsmtExposure	2.654110
BsmtFinType2	2.654110
BsmtCond	2.568493
BsmtFinType1	2.568493
BsmtQual	2.568493
MasVnrArea	0.599315
MasVnrType	0.599315



In [29]:
```
# Create a new column named IsRemodelled - This column would determine whether the house has been remodelled or not based on
# the difference between remodelled and built years

def checkForRemodel(row):
    if(row['YearBuilt'] == row['YearRemodAdd']):
        return 0
    elif(row['YearBuilt'] < row['YearRemodAdd']):
        return 1
    else:
        return 2
df['IsRemodelled'] = df.apply(checkForRemodel, axis=1)
df.head()
```

Out[29]:

F	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice	IsRemodelled
5	0	0	0	0	NaN	NaN	NaN	0	2	2007	WD	Normal	128000	0
7	0	0	224	0	NaN	NaN	NaN	0	10	2007	WD	Normal	268000	0
0	0	0	0	0	NaN	NaN	NaN	0	6	2007	WD	Normal	269790	1
2	0	0	0	0	NaN	MnPrv	NaN	0	1	2010	COD	Normal	190000	0
0	0	0	0	0	NaN	NaN	NaN	0	6	2009	WD	Normal	215000	1

In [30]:
```
# Create a new column named BuiltOrRemodelledAge and determine the age of the building at the time of selling

def getBuiltOrRemodelAge(row):
    if(row['YearBuilt'] == row['YearRemodAdd']):
        return row['YrSold'] - row['YearBuilt']
    else:
        return row['YrSold'] - row['YearRemodAdd']

df['BuiltOrRemodelAge'] = df.apply(getBuiltOrRemodelAge, axis=1)
df.head()
```

Out[30]:

SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice	IsRemodelled	BuiltOrRemodelAge
0	0	0	NaN	NaN	NaN	0	2	2007	WD	Normal	128000	0	31
0	224	0	NaN	NaN	NaN	0	10	2007	WD	Normal	268000	0	37
0	0	0	NaN	NaN	NaN	0	6	2007	WD	Normal	269790	1	10
0	0	0	NaN	MnPrv	NaN	0	1	2010	COD	Normal	190000	0	33
0	0	0	NaN	NaN	NaN	0	6	2009	WD	Normal	215000	1	9

Fill GarageYrBlt with zero values

In [31]:
```
df['GarageYrBlt']= df['GarageYrBlt'].fillna(0)
```

In [32]:
```
# Create a new column which would indicate if the Garage is old or new.
# Garage Yr Built less than 2000 will be considered as old (1) else new(2).
# For GarageYrBuilt , where we have imputed the value as 0 will also be treated as 'No Garrage'

def getGarageConstructionPeriod(row):
    if row == 0:
        return 0
    elif row >= 1900 and row < 2000:
        return 1
    else:
        return 2

df['OldOrNewGarage'] =df['GarageYrBlt'].apply(getGarageConstructionPeriod)
df.head()
```

Out[32]:

'orch	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice	IsRemodelled	BuiltOrRemodelAge	OldOrNewGarage
0	0	NaN	NaN	NaN	0	2	2007	WD	Normal	128000	0	31	1
224	0	NaN	NaN	NaN	0	10	2007	WD	Normal	268000	0	37	1
0	0	NaN	NaN	NaN	0	6	2007	WD	Normal	269790	1	10	1
0	0	NaN	MnPrv	NaN	0	1	2010	COD	Normal	190000	0	33	1
0	0	NaN	NaN	NaN	0	6	2009	WD	Normal	215000	1	9	1

In [33]:
```
# Since we have created new features from YearBuilt, YearRemodAdd, YrSold and GarageYrBlt, we can drop these columns as we
# would only be using the derived columns for further analysis

df.drop(['YearBuilt', 'YearRemodAdd', 'YrSold', 'GarageYrBlt'], axis = 1, inplace = True)
```

In [35]:
```
# we have to do features engineering on numerical that are not rational for modeling
# combine '1st Flr SF', '2nd Flr SF','LowQualFinSF' - as 'GrLivArea'
# Combine columns 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF'-as 'TotalBsmtSF'
df['TotalBsmtSF'] = df['BsmtFinSF1'] + df['BsmtFinSF2']+df['BsmtUnfSF']
df['GrLivArea'] = df['1stFlrSF'] + df['2ndFlrSF'] + df['LowQualFinSF']
```

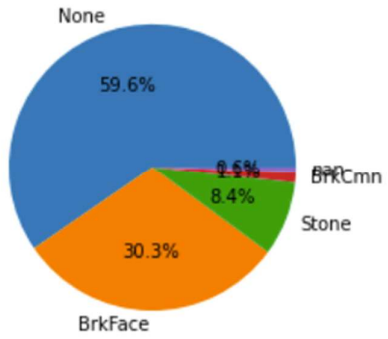
In [36]:
```
# drop 'BsmtUnfSF','BsmtFinSF1', 'BsmtFinSF2' since there is TotalBsmtSF
df.drop(columns = ['BsmtUnfSF','BsmtFinSF1', 'BsmtFinSF2','1stFlrSF','2ndFlrSF','LowQualFinSF'], axis=1, inplace = True)
```

In [37]:
```
missing_count=df.isnull().sum()
missing_count[missing_count>0].sort_values(ascending=False)
```

Out[37]:
```
PoolQC          1161
MiscFeature     1124
Alley           1091
Fence            931
FireplaceQu      551
LotFrontage      214
GarageType        64
GarageFinish      64
GarageQual        64
GarageCond        64
BsmtExposure      31
BsmtFinType2      31
BsmtFinType1      30
BsmtCond          30
BsmtQual          30
MasVnrArea         7
MasVnrType         7
dtype: int64
```

In [41]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
df['MasVnrType'].value_counts(dropna=False).plot.pie(autopct='%1.1f%%')
plt.ylabel('')
df['MasVnrType'].value_counts(dropna=False)
```

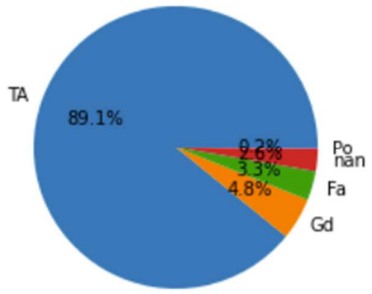
Out[41]:
```
None       696
BrkFace    354
Stone       98
BrkCmn      13
NaN          7
Name: MasVnrType, dtype: int64
```



more than 59% values are none so we can drop this column

In [42]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
df['BsmtCond'].value_counts(dropna=False).plot.pie(autopct='%1.1f%%')
plt.ylabel('')
df['BsmtCond'].value_counts(dropna=False)
```

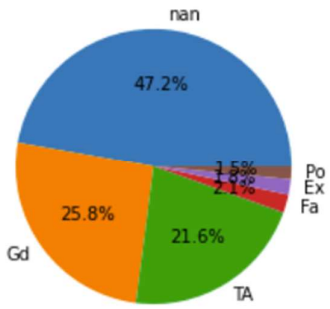
Out[42]:
```
TA     1041
Gd       56
Fa       39
NaN      30
Po        2
Name: BsmtCond, dtype: int64
```



more than 89% values are none so we can drop this column

In [43]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
df['FireplaceQu'].value_counts(dropna=False).plot.pie(autopct='%1.1f%%')
plt.ylabel('')
df['FireplaceQu'].value_counts(dropna=False)
```

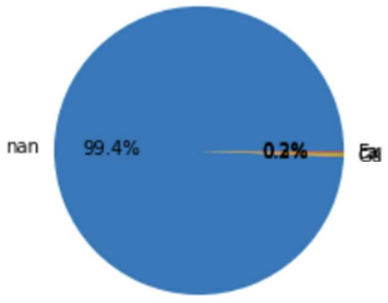
Out[43]:
```
NaN    551
Gd     301
TA     252
Fa      25
Ex      21
Po      18
Name: FireplaceQu, dtype: int64
```



more than 47% values are none so we can drop this column

In [44]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
df['PoolQC'].value_counts(dropna=False).plot.pie(autopct='%1.1f%%')
plt.ylabel('')
df['PoolQC'].value_counts(dropna=False)
```

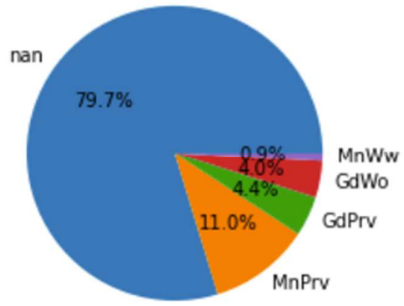
Out[44]:
```
NaN    1161
Gd        3
Ex        2
Fa        2
Name: PoolQC, dtype: int64
```



more than 99% values are none so we can drop this column

In [45]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
df['Fence'].value_counts(dropna=False).plot.pie(autopct='%1.1f%%')
plt.ylabel('')
df['Fence'].value_counts(dropna=False)
```

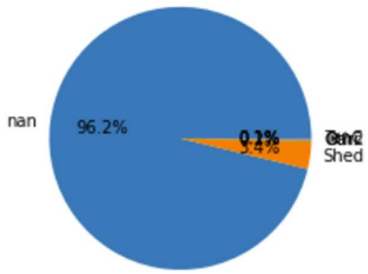
Out[45]:
```
NaN      931
MnPrv    129
GdPrv     51
GdWo      47
MnWw      10
Name: Fence, dtype: int64
```



more than 79% values are none so we can drop this column

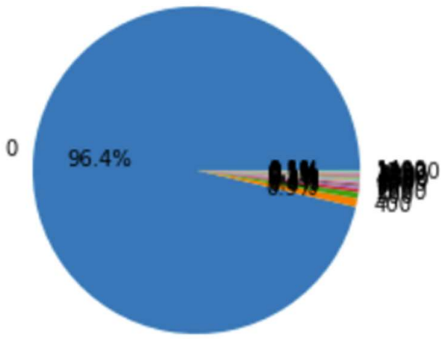
In [47]:
```
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
df['MiscFeature'].value_counts(dropna=False).plot.pie(autopct='%1.1f%%')
plt.ylabel('')
df['MiscFeature'].value_counts(dropna=False)
```

Out[47]:
```
NaN     1124
Shed      40
Gar2       2
Othr       1
TenC       1
Name: MiscFeature, dtype: int64
```



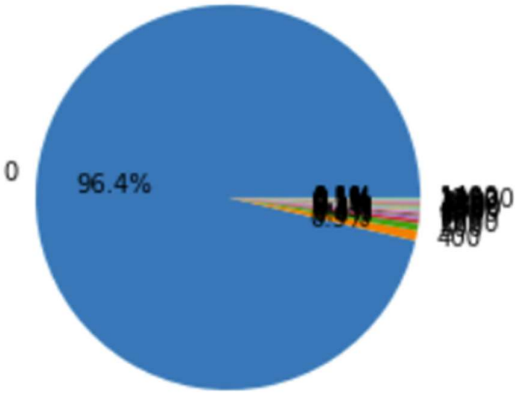
more than 96% values are none so we can drop this column

```
Name: MiscVal, dtype: int64
```



more than 96% values are none so we can drop this column

```
Name: MiscVal, dtype: int64
```



more than 96% values are none so we can drop this column

In [76]:
```
missing_count=df1.isnull().sum()
missing_count[missing_count>0].sort_values(ascending=False)
```

Out[76]:
```
Series([], dtype: int64)
```

```
In [50]: # replace empty values of the columns by mean,most frequent values
         import numpy as np
         from sklearn.impute import SimpleImputer

In [51]: for col in [ 'LotFrontage','TotalBsmtSF']:
             imp=SimpleImputer(missing_values=np.NaN,strategy="mean")
             df[col]=imp.fit_transform(df[col].values.reshape(-1,1))

In [52]: for col in [ 'BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2','GarageType','GarageFinish']:
             imp=SimpleImputer(missing_values=np.NaN,strategy="most_frequent")
             df[col]=imp.fit_transform(df[col].values.reshape(-1,1))
```

drop columns has null values and replace columns which has no garage

```
In [53]: df.drop(['Alley','MasVnrType','MasVnrArea','FireplaceQu','PoolQC','Fence','MiscFeature','MiscVal'], axis = 1, inplace = True)

In [54]: df['GarageQual'] = df['GarageQual'].replace(np.nan, 'No Garage')

In [55]: df['GarageCond'] = df['GarageCond'].replace(np.nan, 'No Garage')
```

Note:

MAKE TEST SET READY BY APPLYING ABOVE SAME PROCEDURE

Data Inputs- Logic- Output Relationships

1. If y represents the **dependent variable and x** the independent variable, this relationship is described as the regression of y on x. The relationship can be represented by a simple equation called the regression equation.
2. Linear regression **attempts to model the relationship between two variables by fitting a linear equation to observed data**. ... A linear regression line has an equation of the form Y = a + bX, where X is the explanatory variable and Y is the dependent variable.
3. The output variable (also called dependent variable, or regressand) is assumed to be a linear function of the input variables (also called **independent variables**, or regressors) and of an unobservable error term that adds noise to the linear relationship between inputs and outputs.
4. To do this a line is created that best fits a set of data pairs. The value of y is derived through the value of x, reflects their correlation
5. The RESPONSE of a system is linear **when the output is directly proportional to the input**, that is, any change in the input produces a proportional change in the output. When plotted on a graph, a straight line results.
6. **Correlation** describes the relationship between two sets of data.
7. Multiple linear regression is a regression model that estimates the relationship between a quantitative dependent variable and two or more independent variables using a straight line.
8. In multiple regression, there are multiple independent variables that enable us to estimate the dependent variable y. Multiple regression equation is derived by:

$$Y = a + b_{1*1} + b_{2*2} + b_{3*3} \ldots\ldots\ldots\ldots b_{k*k}$$

Here, y is an independent variables whereas b_1, b_2 and b_k

1. Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data.
2. In Multiple Linear Regression, a Residual is the Difference Between Estimated Dependent Variables and Actual Dependent Variables. Multiple linear regression assumes that the remaining variables' error is similar at each point of the linear model. This is known as homoscedasticity. When the data analysis is done, the standard residuals against the predicted values are plotted to determine if the points are properly distributed across independent variables' values. Larger residuals indicate that the regression line is a poor fit for the data, i.e. the actual data points do not fall close to the regression line. Smaller residuals indicate that the regression line fits the data better, i.e. the actual data points fall close to the regression line.

The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R^2 always increases as more predictors are added to the MLR model, even though the predictors may not be related to the outcome variable.

Multiple linear regression (MLR) is used to determine a mathematical relationship among several random variables.

In a multiple linear regression, the model calculates the line of best fit that minimizes the variances of each of the variables included as it relates to the dependent variable. Because it fits a line, it is a linear model.

The relationship can also be non-linear, and the dependent and independent variables will not follow a straight line. Linear and non-linear regression are used to track a response using two or more variables.

Assuming a linear relation in population, mean of Y for given X equals $\alpha+\beta X$ i.e. the "population regression line". If $Y = a + bX$ is the estimated line, then the fitted $\hat{Y}i = a + bXi$ is called the fitted (or predicted) value, and $Yi\ \hat{Y}i$ is called the residual.

The regression coefficient of y on x is represented by b yx and x on y as b xy. Both of the regression coefficients must have the same sign. If b yx is positive, bxy will also be positive and it is true for vice versa. If one regression coefficient is greater than unity, then others will be lesser than unity.

For a bivariate data (Xi, Yi), the relationship may be Y depends on X or X depends on Y. If Y depends on X then the regression line is Y on X. Y is dependent variable and X is independent variable. If X depends on Y, then regression line is X on Y and X is dependent variable and Y is independent variable.

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

DATA PREPROCESSING AND FEATURE ENGINEERING

drop column since no correlation with target

```
In [109]: df.drop(columns = ['Utilities'], axis=1, inplace = True)
```

```
In [110]: #converting string data type to int type using LabelEncoding
le=LabelEncoder()

list1=['MSZoning','Street','LotShape','LandContour','LotConfig','LandSlope','Neighborhood','Condition1','Condition2','BldgType',
for val in list1:
    df[val]=le.fit_transform(df[val].astype(str))
df
```

Out[110]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	Bl
0	127	120	3	70.98847	4928	1	0	3	4	0	13	2	2	
1	889	20	3	95.00000	15865	1	0	3	4	1	12	2	2	
2	793	60	3	92.00000	9920	1	0	3	1	0	15	2	2	
3	110	20	3	105.00000	11751	1	0	3	4	0	14	2	2	
4	422	20	3	70.98847	16635	1	0	3	2	0	14	2	2	
5	1197	60	3	58.00000	14054	1	0	3	4	0	8	2	2	
6	561	20	3	70.98847	11341	1	0	3	4	0	19	2	2	
7	1041	20	3	88.00000	13125	1	3	3	0	0	19	2	2	
8	503	20	3	70.00000	9170	1	3	3	0	0	7	1	2	
9	576	50	3	80.00000	8480	1	3	3	4	0	12	2	2	
10	449	50	4	50.00000	8600	1	3	0	4	0	9	2	2	

```
In [115]: #find correlation co.efficient of all variables in table
df.corr()
```

Out[115]:

orchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MoSold	SaleType	SaleCondition	IsRemodelled	BuiltOrRemodelAge	OldOrNewGarage	SalePrice
013642	0.004885	-0.021773	0.005169	0.065832	0.023479	0.024384	-0.014726	-0.003918	0.018025	0.004918	-0.023897
017468	-0.004252	-0.043210	-0.013291	0.009583	-0.016015	0.035050	-0.028981	-0.034023	-0.059092	0.000946	-0.060775
152694	0.111221	0.004409	0.030793	-0.001663	-0.051646	0.079854	0.004501	0.122313	0.174282	-0.201011	-0.133221
151328	0.020902	0.051084	0.030405	0.196001	0.022517	-0.035356	0.065091	-0.040283	-0.089055	0.096343	0.323779
093080	-0.007446	0.025794	0.025256	0.097107	0.015141	0.005421	0.034236	0.034951	-0.029495	-0.004516	0.249499
029649	0.021360	0.007338	0.016026	0.004505	-0.008860	0.025920	0.014176	0.027913	-0.059127	0.051709	0.044753
090534	0.078660	-0.016018	-0.057748	-0.022160	-0.050418	-0.015161	-0.054905	0.075393	0.156811	-0.078643	-0.248171
034995	-0.061782	-0.032990	0.016993	-0.014235	-0.023872	-0.041763	0.047715	-0.104171	-0.086335	0.098148	0.032836
052485	-0.069765	-0.049588	0.005699	-0.051021	0.019084	-0.002039	0.043692	0.008929	0.008653	0.036461	-0.060452
047393	0.001663	0.014520	0.030839	-0.017377	0.030526	0.056004	-0.061461	0.062724	0.048260	-0.103819	0.015485
149294	0.032618	-0.025838	-0.004306	-0.010331	0.023378	-0.023081	0.042340	-0.047031	-0.089821	0.074849	0.198942
098814	-0.070914	0.089272	-0.008405	0.009655	0.001801	-0.007101	0.059027	-0.071838	-0.157633	0.115009	0.105820
056692	-0.001633	-0.003002	-0.006555	-0.001843	0.013511	0.001565	0.054288	-0.023411	-0.040453	0.022641	0.033956
037433	-0.108686	-0.018450	-0.010995	-0.030991	-0.015765	-0.020145	-0.000308	-0.148499	-0.122156	0.130560	-0.066028
161262	-0.045623	-0.012872	-0.045850	0.086091	0.030316	0.056643	0.024668	-0.148816	-0.224113	0.099962	0.205502
341030	-0.098374	0.045919	0.059387	0.072247	0.090638	-0.049794	0.212932	-0.079552	-0.559110	0.571413	0.789185
024899	0.056074	0.040476	0.069463	-0.003603	0.005519	0.102515	0.023908	0.308380	-0.077124	-0.225313	-0.065642
020609	0.008710	0.040104	0.053627	0.023089	0.014650	-0.045064	0.064453	-0.054268	-0.002908	0.022594	0.192654

Correlation with Heatmap:

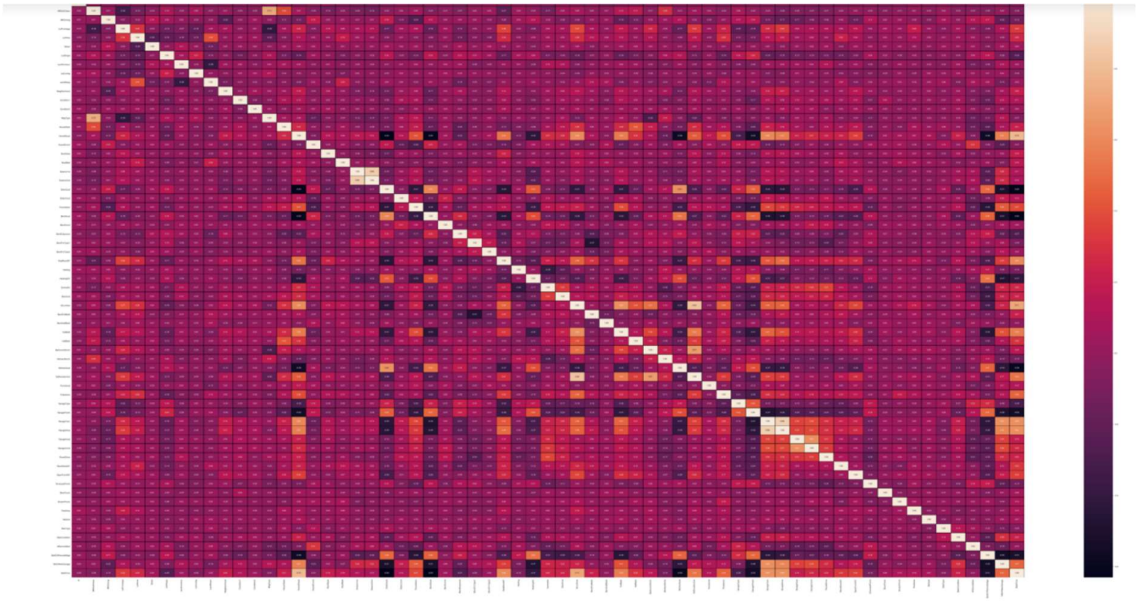
The correlation coefficient is a statistical measure of the strength of the relationship between the relative movements of two variables. The values range between -1.0 and 1.0. A calculated number greater than 1.0 or less than -1.0 means that there was an error in the correlation measurement. A correlation of -1.0 shows a perfect negative correlation, while a correlation of 1.0 shows a perfect positive correlation. A correlation of 0.0 shows no linear relationship between the movement of the two variables.Correlation statistics can be used in finance and investing. Pearson correlation is

the one most commonly used in statistics. This measures the strength and direction of a linear relationship between two variables.

It can also be defined as the measure of dependence between two different variables. If there are multiple variables and the goal is to find correlation between all of these variables and store them using appropriate data structure, the **matrix data structure** is used. Such matrix is called as **correlation matrix.**

Correlation heatmap is graphical representation of **correlation matrix** representing correlation between different variables.

For to do feature selection and make feature ready for the model building.we check correlation of variables using heatmap.And describe method for the census data set.



most positively correlated features are OverallQual,GrLivArea,GarageCars,GarageArea,TotalBsmtSF etc., most negatively correlated columns are BsmtQual,ExterQual,KitchenQual,GarageFinish,BuiltOrRemodelAge,HeatingQC,GarageType Id,IsRemodelled,BsmtHalfBath,street are the columns which has very less correlation with the target column

```
In [117]: df.describe()
```

```
In [117]: df.describe()
```

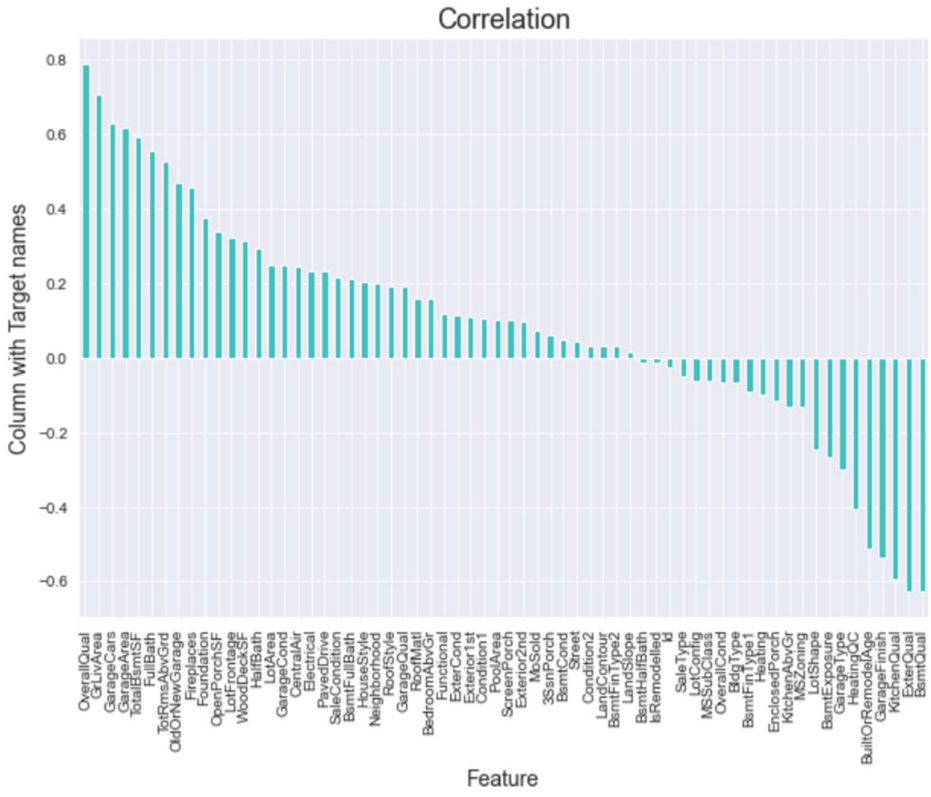
Out[117]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood
count	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000
mean	724.136130	56.767979	3.013699	70.988470	10484.749144	0.996575	1.938356	2.773973	3.004281	0.064212	12.145548
std	416.159877	41.940650	0.633120	22.437056	8957.442311	0.058445	1.412262	0.710027	1.642667	0.284088	6.010364
min	1.000000	20.000000	0.000000	21.000000	1300.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	360.500000	20.000000	3.000000	60.000000	7621.500000	1.000000	0.000000	3.000000	2.000000	0.000000	7.000000
50%	714.500000	50.000000	3.000000	70.988470	9522.500000	1.000000	3.000000	3.000000	4.000000	0.000000	12.000000
75%	1079.500000	70.000000	3.000000	79.250000	11515.500000	1.000000	3.000000	3.000000	4.000000	0.000000	17.000000
max	1460.000000	190.000000	4.000000	313.000000	164660.000000	1.000000	3.000000	3.000000	4.000000	2.000000	24.000000

slight skewness present in all continous area value numeric columns since due to outliers because of high diff between 75% and maximum value

Correlation model:

Graph depicts clearly the positive and negative correlation of each variables with target column, justifies the outcome outlined in Multivariate analysis, that higher the education higher the gain & vice-versa




```
In [120]: df.drop(['Id'], axis = 1, inplace = True)
```

Drop ID column since it makes not much correlation with the target column.

	Variables	VIF FACTOR
0	MSSubClass	12.723632
1	MSZoning	31.900828
2	LotFrontage	18.992696
3	LotArea	4.019869
4	Street	241.618376
5	LotShape	3.601681
6	LandContour	20.744895
7	LotConfig	4.988899
8	LandSlope	1.618043
9	Neighborhood	6.350953
10	Condition1	7.251098
11	Condition2	65.802814
12	BldgType	4.913823
13	HouseStyle	7.666446
14	OverallQual	77.538220
15	OverallCond	42.401801
16	RoofStyle	4.710279
17	RoofMatl	4.613815
18	Exterior1st	37.894550
19	Exterior2nd	36.441481
20	ExterQual	36.613959

drop the columns which has high vif values

In [126]:
```
df.drop(['Street'], axis = 1, inplace = True)
```

Drop street column since it has high vif

36	HalfBath	3.076468
37	BedroomAbvGr	36.232341
38	KitchenAbvGr	40.406216
39	KitchenQual	18.050904
40	TotRmsAbvGrd	90.400168
41	Functional	40.063789
42	Fireplaces	3.148432
43	GarageType	4.361123
44	GarageFinish	6.727267

```
In [128]: df.drop(['TotRmsAbvGrd'], axis = 1, inplace = True)
```

Drop TotRmsAbvGrd column since it has high vif

44	GarageCars	38.574046
45	GarageArea	33.678304
46	GarageQual	54.608548
47	GarageCond	73.143190
48	PavedDrive	20.989325
49	WoodDeckSF	2.031656
50	OpenPorchSF	2.008261
51	EnclosedPorch	1.368097
52	3SsnPorch	1.068552
53	ScreenPorch	1.234721
54	PoolArea	1.144336
55	MoSold	6.944996
56	SaleType	24.214339
57	SaleCondition	15.203433
58	IsRemodelled	3.056120
59	BuiltOrRemodelAge	5.987803
60	OldOrNewGarage	20.740842

```
In [130]: df.drop(['GarageCond'], axis = 1, inplace = True)
```

Drop GarageCond column since it has high vif value

MAKE TEST SET READY BY APPLYING ABOVE SAME PROCEDURE

Hardware and Software Requirements and Tools Used

HARDWARE & SOFTWARE TOOLS, LIBRARIES AND PACKAGES USED:

Hardware :Intel i7,RAM 16GB used.

Software: Jupyter Notebook (Anaconda 3)

Language: Python

Libraries:

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn
5. Sklean
6. Scipy
7. Statsmodels
8. Pip-Package install Manager

```
#import all libaries
import pandas as pd
import numpy as np
import statistics
import seaborn as sns
from matplotlib import pyplot as plt
import sklearn
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
import warnings
warnings.filterwarnings('ignore')
url="https://raw.githubusercontent.com/dsrscientist/dataset1/master/census_income.csv"
```

Category	Tool	Function
Data loading and analysis	Import pandas as pd	Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. Pandas is built on top of numpy. So, numpy gets some superpower with pandas. It offers data structures and operations for manipulating numerical tables and time series.
	Import numpy as np	NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It has Quantile method too for removing outliers. It is the fundamental package for scientific computing with Python
Data visualization	Import matplotlib.pyplot as plt	Matplotlib is a plotting library used for data visualization.
	Import seaborn as sns	Seaborn is also a plotting library. It is more advanced than matplotlib but works with matplotlib
Scikit Learn Preprocessing Libraries	Sklearn.preprocessing	Package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.Has power transformer to remove skewness. In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate. It has MinMaxScaler to scale the data.
	Sklearn.preprocessing import LabelEncoder	Label Encoding in Python can be implemented using the Sklearn Library. Sklearn furnishes a very effective method for encoding the categories of categorical features into numeric values. Label encoder encodes labels with credit between 0 and n-1 classes where n is the number of diverse labels.

Import statistics	Import statsmodels.api as sm	From scipy import stats This module provides functions for calculating mathematical statistics of numeric (Real-valued) data. This library provides a number of common functions and types useful in statistics. It focus on high performance, numerical robustness, and use of good algorithms

```
In [55]: #VIF calculation
         import statsmodels.api as sm
         from scipy import stats
         from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Variance Inflation Factors (VIFs) measure the correlation among independent variables in least squares regression models. Statisticians refer to this type of correlation as multicollinearity. Excessive multicollinearity can cause problems for regression models. The stats models package has VIF library and we can import this library.

```
In [75]: #train test split
         from sklearn.model_selection import train_test_split

         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=40)
```

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train_test_split() function. The function takes a loaded dataset as input and returns the dataset split into two subsets.train_test_split() will split arrays data into random subsets. The ideal split is said to be 80:20 for training and testing.

```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

Used to check the error and score of the model.The error tells how much is the difference between actual and predicted result.The score tells the accuracy of the model.

```
#perform gridsearchcv and cross val score on LinearRegression
from sklearn.model_selection import GridSearchCV
```

Grid search is used as an approach to hyper-parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid. GridSearchCV helps us combine an estimator with a grid search preamble to tune hyper-parameters.

```
from sklearn.model_selection import cross_val_score
```

Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.
The three steps involved in cross-validation are as follows :

1. Reserve some portion of sample data-set.
2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

```
#Linear Regression
#model fitting
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.model_selection import train_test_split
```

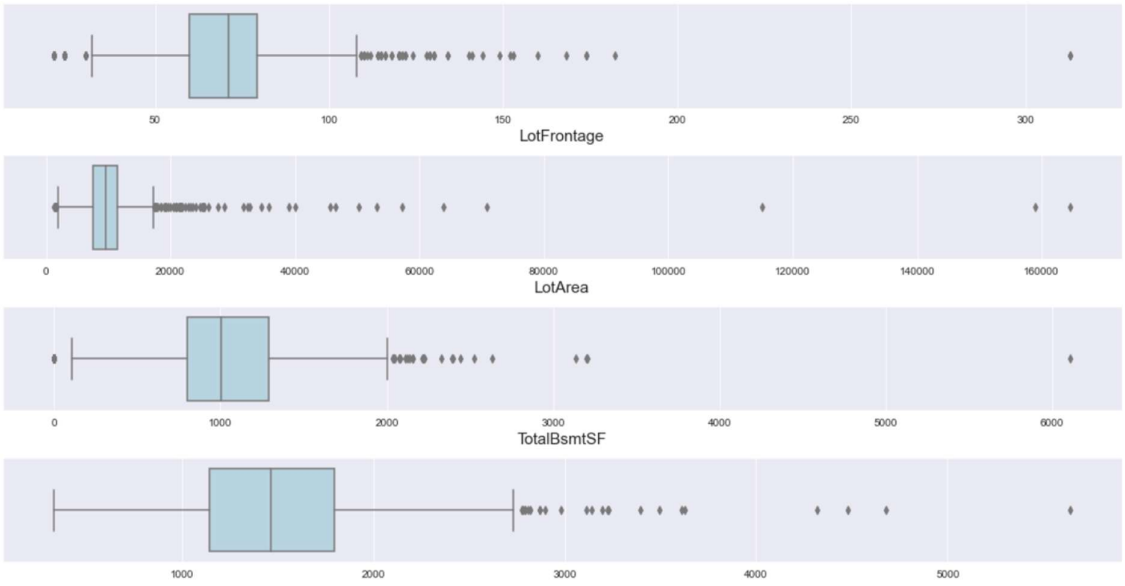
Various Models of Machine learning tools can be used in to do various iterations & select the optimal model for problem solving and predictions.
We can import the necessary evaluation metrics library and other libraries from sklearn.linear_model And sklearn.metrics.

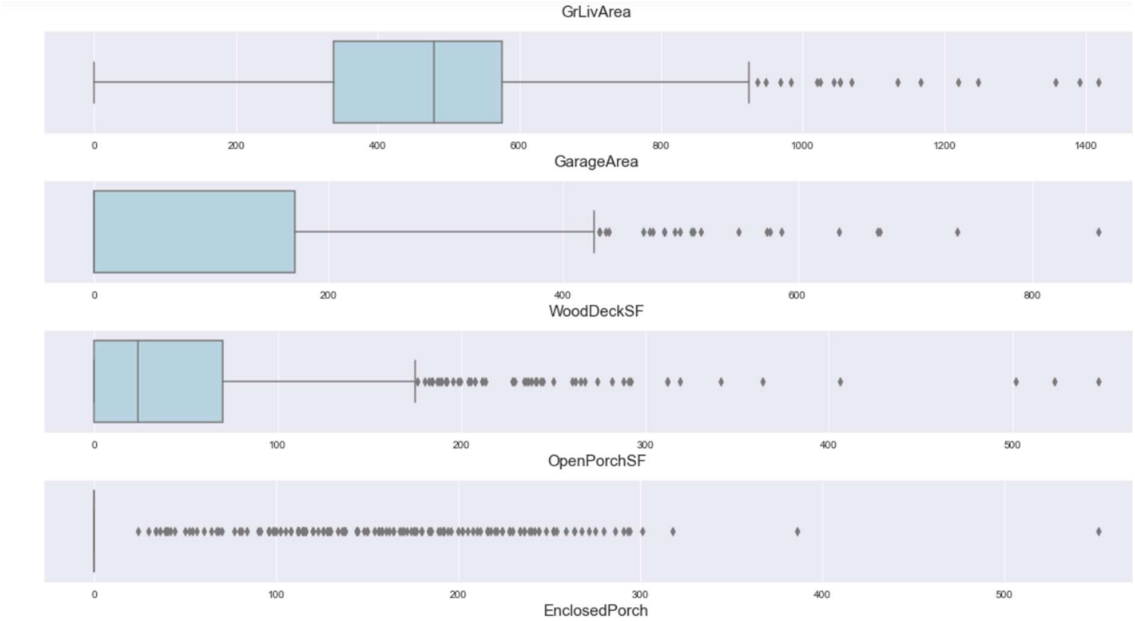
```
from sklearn.ensemble import GradientBoostingRegressor
```

Ensemble learning is a technique in machine learning which takes the help of several base models and combines their output to produce an optimized model. This type of machine learning algorithm helps in improving the overall performance of the model.

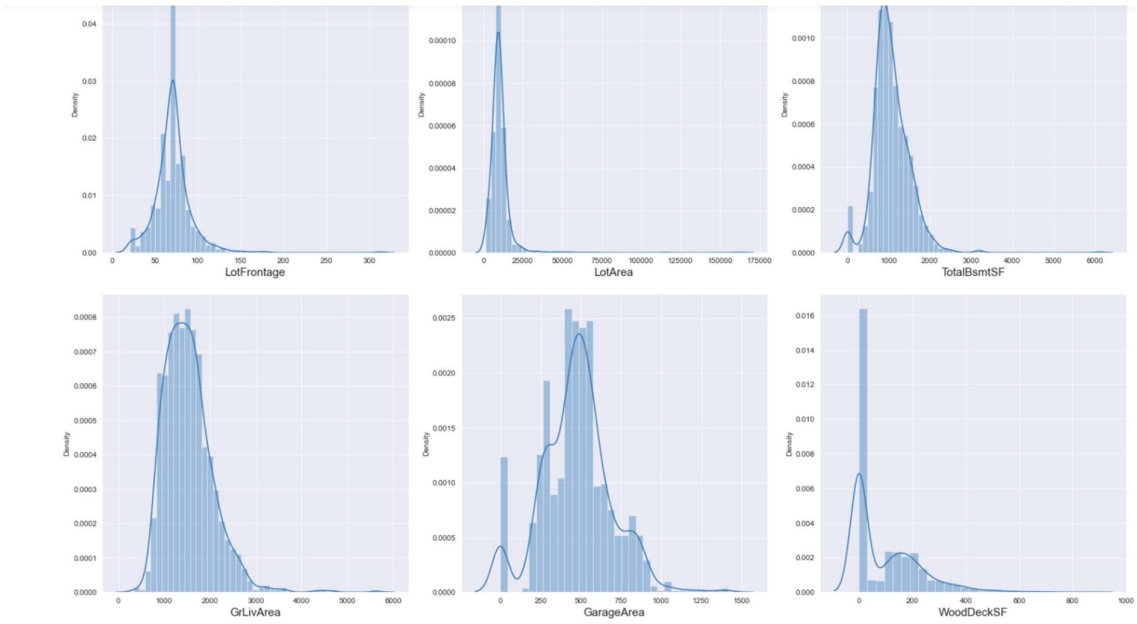
Model/s Development and Evaluation

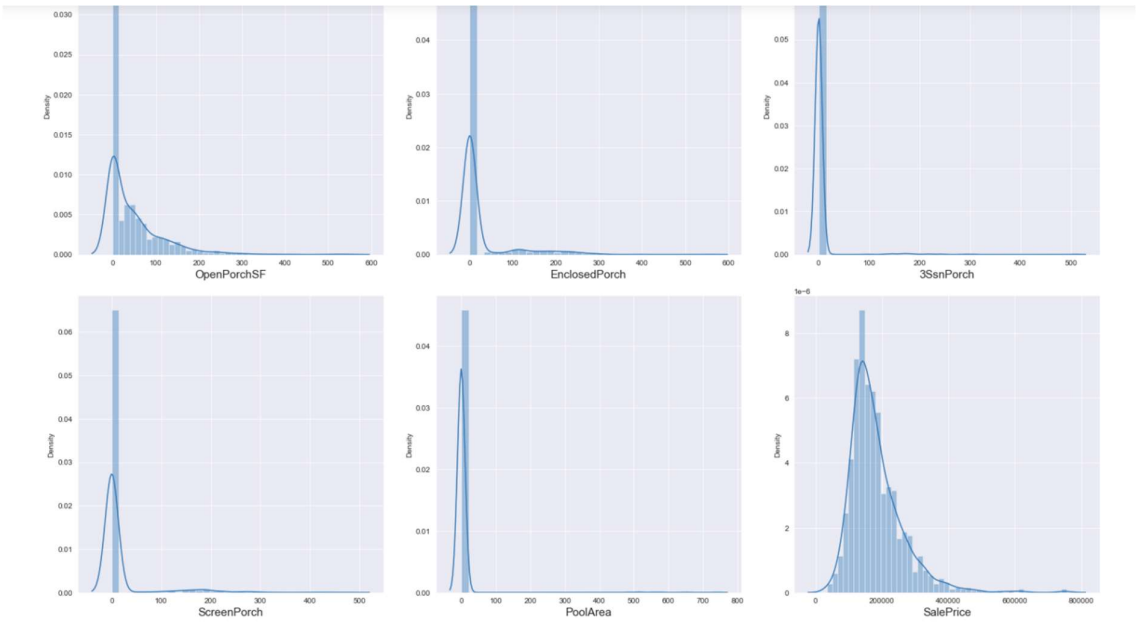
Identification of possible problem-solving approaches (methods)





'LotFrontage','LotArea','TotalBsmtSF','GrLivArea','GarageArea','WoodDeckSF','OpenPorchSF','EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','SalePrice' .These are the columns having outliers which is shown in box plot.It seems all area columns which is continuous having outliers.





'LotFrontage','LotArea','TotalBsmtSF','GrLivArea','GarageArea','WoodDeckSF','OpenPorchSF','EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','SalePrice'.These columns has skewness too because of outliers.

skewness present due to outliers

#checking Quantile to remove outliers

In [140]:
```
cols = ['LotFrontage','LotArea','TotalBsmtSF','GrLivArea','GarageArea','WoodDeckSF','OpenPorchSF','EnclosedPorch','3SsnPorch','Sc

Q1 = df[cols].quantile(0.25)
Q3 = df[cols].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[cols] < (Q1 - 1.5 * IQR)) |(df[cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [141]:
```
df.shape
```

Out[141]: (683, 61)

some outliers got removed

In [142]:
```
#finds data loss
loss_percent=(1168-685)/(1168*100)
print(loss_percent)
```

0.00413527397260274

data loss percentage is very low.data is cleaned also

In [144]:
```
#seggregate input data and output data
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [145]:
```
#removing skewness using power transform
from sklearn.preprocessing import power_transform
x=power_transform(x,method='yeo-johnson',standardize=False)
x
```

Out[145]:
```
array([[  2.45600779,  15.87630441, 317.04318195, ...,   0.46078334,
          3.14407245,   0.97423372],
       [  2.05806426,  15.87630441, 377.68614596, ...,  -0.        ,
          5.29304713,   0.97423372],
       [  2.05806426,  15.87630441, 225.03060806, ...,   0.46078334,
          2.98549333,   0.97423372],
       ...,
       [  2.45600779,  15.87630441, 225.03060806, ...,  -0.        ,
          3.14407245,   0.97423372],
       [  2.05806426,  15.87630441, 225.03060806, ...,  -0.        ,
          5.86223895,   0.97423372],
       [  2.45600779,  15.87630441, 225.03060806, ...,   0.46078334,
          1.61693995,   1.91445181]])
```

In [146]:
```
#scaling to get better model performance
from sklearn.preprocessing import MinMaxScaler
mmscaler = MinMaxScaler()
x= mmscaler.fit_transform(x)
x
```

Out[146]:
```
array([[0.55890157, 0.53727748, 0.76042942, ..., 1.        , 0.47417391,
        0.50888391],
       [0.        , 0.53727748, 0.95412225, ..., 0.        , 0.79827196,
        0.50888391],
       [0.        , 0.53727748, 0.46654248, ..., 1.        , 0.45025777,
        0.50888391],
       ...,
       [0.55890157, 0.53727748, 0.46654248, ..., 0.        , 0.47417391,
        0.50888391],
       [0.        , 0.53727748, 0.46654248, ..., 0.        , 0.88411474,
        0.50888391],
       [0.55890157, 0.53727748, 0.46654248, ..., 1.        , 0.24385912,
        1.        ]])
```

MAKE TEST SET READY BY APPLYING ABOVE SAME PROCEDURE

Testing of Identified Approaches (Algorithms)

These are all the Algorithms used for Model Building and Prediction.We did Hyper Parameter Tuning with these algorithms using the GridSearchCV.

Ridge Regressor

Lasso regression

LinearRegression

RandomForestRegressor

GradientBoostingRegressor

Support Vector Regressor

ElasticNet Regression

Decison Tree Regressor

These algorithms has been used for both Training and Testing purpose and got evaluated with r2 score.And also the predicted result got Evaluated with Key Metrics.

Run and Evaluate selected models

Linear Regression:

- Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.
- In linear regression, the observations (red) are assumed to be the result of random deviations (green) from an underlying relationship (blue) between a dependent variable (y) and an independent variable (x).

MODEL EVALUATION

REGRESSION :

Linear Regression

In [161]:
```
#Linear Regression
#model fitting
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.model_selection import train_test_split
```

In [162]:
```
#getting best accuracy with help of selecting random state
for i in range(0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.36,random_state=i)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    print(f"At random state {i},the training accuracy is:{r2_score(y_train,pred_train)*100}")
    print(f"At random state {i},the testing accuracy is:{r2_score(y_test,pred_test)*100}")
    print("\n")
```

In [163]:
```
#evaluation metrics

print("Accuracy:", r2_score(y_test,pred_test)*100)
```

```
Accuracy: 86.45666565561379
```

In [164]:
```
#cross validation at random state
Train_accuracy=(r2_score(y_train,pred_train)*100)
Test_accuracy=r2_score(y_test,pred_test)*100
from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(lr,x_test,y_test,cv=j)
    cv_mean=cv_score.mean()*100
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for Training is {Train_accuracy} and accuracy score fo
    print("\n")
```

```
At cross fold 2 the cv score is 73.28007545415997 and accuracy score for Training is 88.33089570426529 and accuracy score for T
esting is 86.45666565561379


At cross fold 3 the cv score is 82.08554035256786 and accuracy score for Training is 88.33089570426529 and accuracy score for T
esting is 86.45666565561379


At cross fold 4 the cv score is 81.04203271515935 and accuracy score for Training is 88.33089570426529 and accuracy score for T
esting is 86.45666565561379


At cross fold 5 the cv score is 81.22277206381595 and accuracy score for Training is 88.33089570426529 and accuracy score for T
esting is 86.45666565561379
```

Hyperparameter Tuning Done Using GridSearchCV:

parameter tuning

In [165]:
```
#perform gridsearchcv and cross val score on LinearRegression
from sklearn.model_selection import GridSearchCV

parameters={ 'fit_intercept':[True,False], 'normalize':[True,False], 'copy_X':[True,False], 'n_jobs':[None,1,-1], 'positive':[Tru
lr=LinearRegression()
clf=GridSearchCV(lr,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'copy_X': True, 'fit_intercept': False, 'n_jobs': None, 'normalize': True, 'positive': False}
```

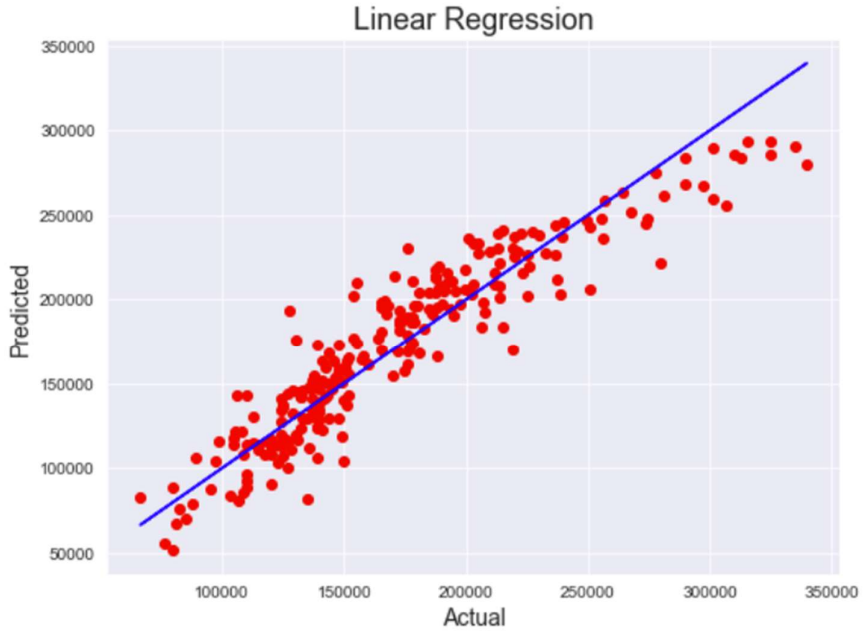
In [166]:
```
lr=LinearRegression(copy_X= True, fit_intercept= True, n_jobs= None, normalize= True, positive= False)
lr.fit(x_train,y_train)
pred_test_lr=lr.predict(x_test)
pred_train_lr=lr.predict(x_train)
lr_score = lr.score(x_train,y_train)
lr_acc_score=r2_score(y_test,pred_test)
print("Accuracy score is:",lr_acc_score*100)
print("score of model is:",lr_score*100)
```

```
Accuracy score is: 86.45666565561379
score of model is: 88.33682682116039
```

In [167]:
```
cv_score_lr=cross_val_score(lr,x_test,y_test,cv=5)
cv_mean_lr=cv_score_lr.mean()
print("cv_mean is:",cv_mean_lr*100)
```

```
cv_mean is: 81.14955064705399
```

```
In [168]: plt.figure(figsize=(8,6))
          plt.scatter(x=y_test,y=pred_test,color='r')
          plt.plot(y_test,y_test,color='b')
          plt.xlabel('Actual',fontsize=14)
          plt.ylabel('Predicted',fontsize=14)
          plt.title('Linear Regression',fontsize=18)
          plt.show()
```




```
In [169]: #evaluation metrics
          print("mean absolute error",mean_absolute_error(y_test,pred_test))
          print("mean squared error",mean_squared_error(y_test,pred_test))
          print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test)))
          print("r2_score is",r2_score(y_test,pred_test)*100)

          mean absolute error 15967.84535676521
          mean squared error 412496438.925007
          Root mean squared error 20310.008343794616
          r2_score is 86.45666565561379
```

REGULARIZATION TECHNIQUES:

LASSO REGRESSION

The full form of LASSO is the Least Absolute Shrinkage and Selection Operation. As the name suggests, LASSO uses the “shrinkage” technique in which coefficients are determined, which get shrunk towards the central point as the mean.

The LASSO regression in regularization is based on simple models that posses fewer parameters. We get a better interpretation of the models due

to the shrinkage process. The shrinkage process also enables the identification of variables strongly associated with variables corresponding to the target.

parameter tuning

In [170]:
```
#perform gridsearchcv and cross val score on lasso regression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import Lasso
parameters={'alpha':[0.0001,0.001,0.01,0.1,1,10],'random_state':list(range(0,10)),'fit_intercept':[True,False],'normalize':[True,
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'alpha': 10, 'fit_intercept': True, 'normalize': True, 'random_state': 0, 'tol': 0.001}
```

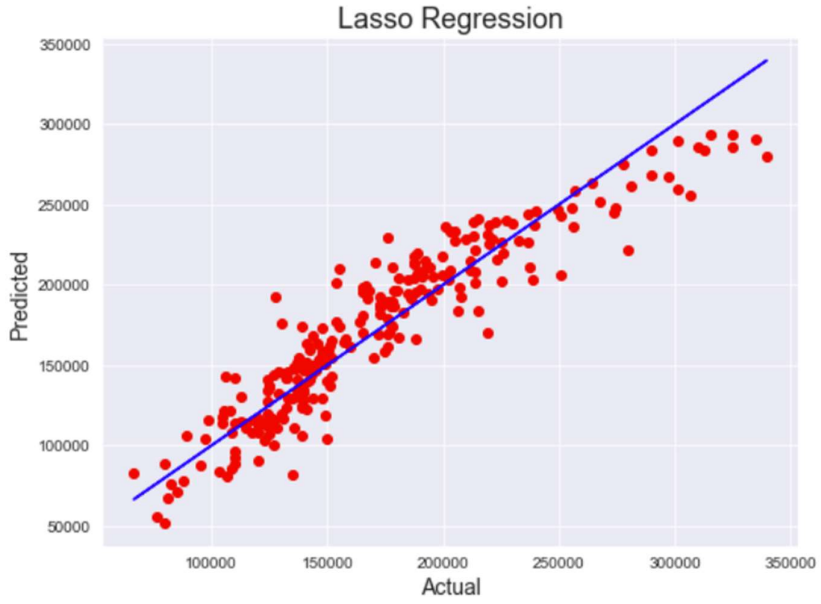
In [171]:
```
ls=Lasso(alpha=1,random_state=0,fit_intercept= True, normalize= False,tol= 0.001)
ls.fit(x_train,y_train)
pred_test_ls=ls.predict(x_test)
pred_train_ls=ls.predict(x_train)
ls_score = ls.score(x_train,y_train)
ls_acc_score=r2_score(y_test,pred_test_ls)
print("Accuracy score is:",ls_acc_score*100)
```

```
Accuracy score is: 86.47084602153473
```

In [172]:
```
#checks cv score
for u in range(2,10):
    cv_score_ls=cross_val_score(ls,x,y,cv=5)
    cv_mean_ls=cv_score_ls.mean()
    print("At cv :-",j )
    print("cv_score is:",cv_mean_ls*100)
    print("accuracy score---r2_score is",ls_acc_score*100)
```

```
At cv :- 9
cv_score is: 84.37101233501139
accuracy score---r2_score is 86.47084602153473
At cv :- 9
cv_score is: 84.37101233501139
accuracy score---r2_score is 86.47084602153473
At cv :- 9
cv_score is: 84.37101233501139
accuracy score---r2_score is 86.47084602153473
At cv :- 9
```

In [173]:
```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_test_ls,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual',fontsize=14)
plt.ylabel('Predicted',fontsize=14)
plt.title('Lasso Regression',fontsize=18)
plt.show()
```



In [174]:
```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_ls))
print("mean squared error",mean_squared_error(y_test,pred_test_ls))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_ls)))
print("r2_score is",r2_score(y_test,pred_test_ls)*100)
```

```
mean absolute error 15972.999155960168
mean squared error 412064540.0811698
Root mean squared error 20299.372898717087
r2_score is 86.47084602153473
```

RANDOM FOREST REGRESSION:

- Random Forest is a Supervised learning algorithm that is based on the ensemble learning method and many Decision Trees. Random Forest is a Bagging technique, so all calculations are run in parallel and there is no interaction between the Decision Trees when building them.

In [175]:
```
#perform gridsearchcv and cross val score on RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
parameters={'criterion':['squared_error', 'mse','absolute_error'],'max_features':["auto","sqrt"],'bootstrap': [True, False],'min_
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'bootstrap': False, 'criterion': 'absolute_error', 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}
```

In [176]:
```
rf=RandomForestRegressor(criterion="mae",max_features="log2",bootstrap= True,min_samples_leaf= 2, min_samples_split= 2)
rf.fit(x_train,y_train)
pred_test_rf=rf.predict(x_test)
pred_train_rf=rf.predict(x_train)
rf_score = rf.score(x_train,y_train)
rf_acc_score=r2_score(y_test,pred_test)
print("Accuracy score is:",rf_acc_score*100)
print("score of model is:",rf_score*100)
```

```
Accuracy score is: 86.45666565561379
score of model is: 92.4200230189798
```

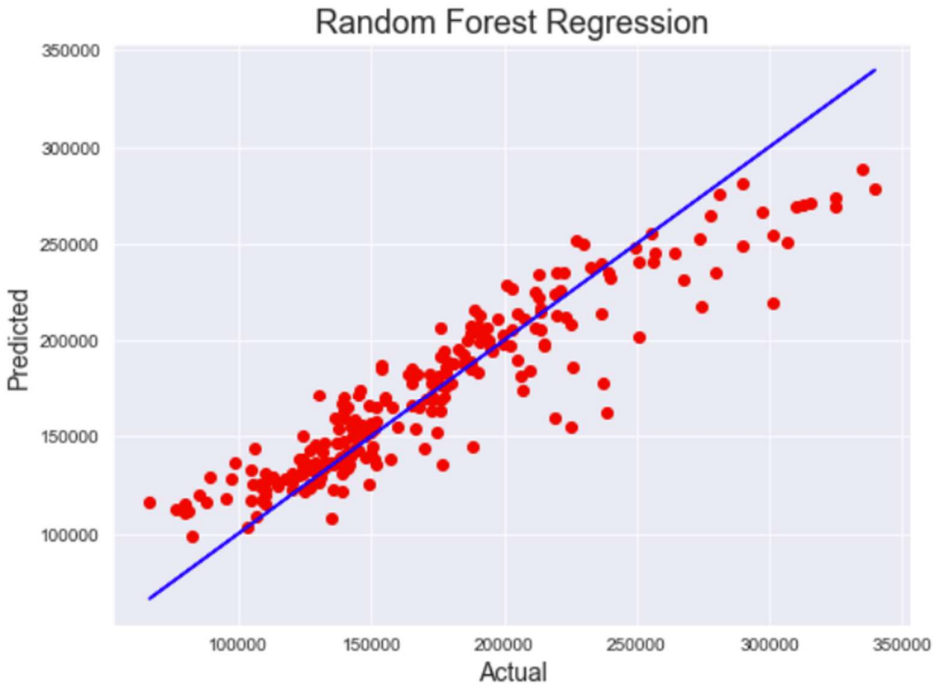
In [177]:
```
cv_score_rf=cross_val_score(rf,x,y,cv=5)
cv_mean_rf=cv_score_rf.mean()
print("cv_mean is:",cv_mean_rf*100)
```

```
cv_mean is: 83.34250892773419
```

In [178]:
```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_rf))
print("mean squared error",mean_squared_error(y_test,pred_test_rf))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_rf)))
print("r2_score is",r2_score(y_test,pred_test_rf)*100)
```

```
mean absolute error 15864.544491869918
mean squared error 480564045.0346275
Root mean squared error 21921.771028697192
r2_score is 84.22182855019051
```

```
In [179]: plt.figure(figsize=(8,6))
          plt.scatter(x=y_test,y=pred_test_rf,color='r')
          plt.plot(y_test,y_test,color='b')
          plt.xlabel('Actual',fontsize=14)
          plt.ylabel('Predicted',fontsize=14)
          plt.title('Random Forest Regression',fontsize=18)
          plt.show()
```



Ridge Regression:

- The Ridge regression is a technique which is specialized to analyze multiple regression data which is multicollinearity in nature.
- Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

  The biggest benefit of ridge regression is its ability to produce a lower test mean squared error (MSE) compared to least squares regression when multicollinearity is present.

RIDGE REGRESSION

In [180]:
```
#perform gridsearchcv and cross val score on Ridge Regressor
from sklearn.linear_model import Ridge
parameters={'alpha':[0.0001,0.001,0.01,0.1,1],'fit_intercept':[True,False],'normalize':[True,False],'tol':[0.001],'random_state':
rd=Ridge()
clf=GridSearchCV(rd,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'alpha': 0.1, 'fit_intercept': True, 'normalize': True, 'random_state': 0, 'tol': 0.001}
```

In [181]:
```
rd=Ridge(alpha=0.001,fit_intercept=True,normalize=True,random_state=0,tol=0.001)
rd.fit(x_train,y_train)
pred_test_rd=rd.predict(x_test)
pred_train_rd=rd.predict(x_train)
rd_score = rd.score(x_train,y_train)
rd_acc_score=r2_score(y_test,pred_test_rd)
print("Accuracy score is:",rd_acc_score*100)
print("score of model is:",rd_score*100)
```

```
Accuracy score is: 86.5336156323644
score of model is: 88.33678321749832
```

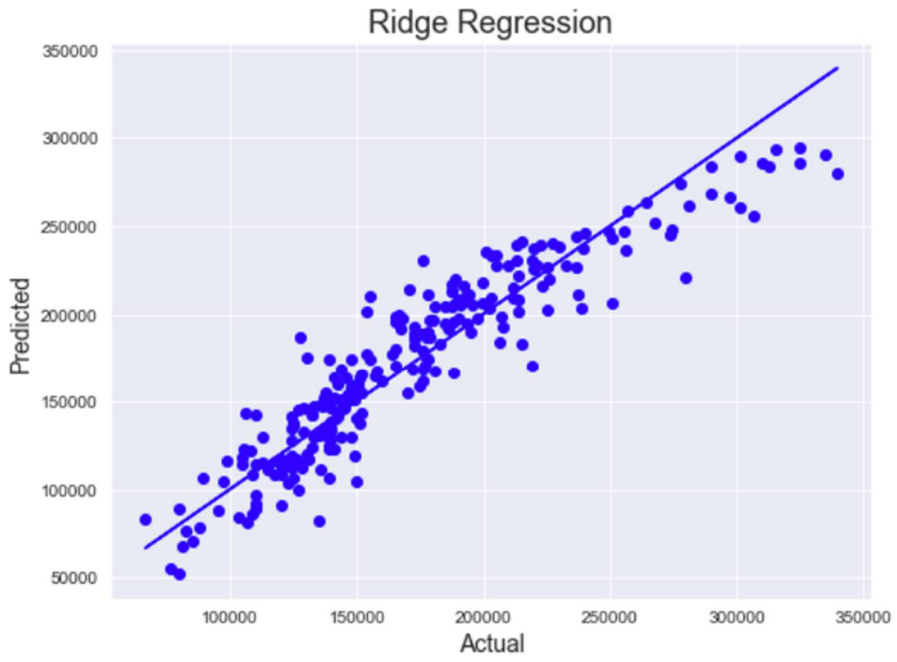
In [182]:
```
cv_score_rd=cross_val_score(rd,x,y,cv=5)
cv_mean_rd=cv_score_rd.mean()
print("cv_mean is:",cv_mean_rd*100)
```

```
cv_mean is: 84.4025604201422
```

In [183]:
```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_rd))
print("mean squared error",mean_squared_error(y_test,pred_test_rd))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_rd)))
print("r2_score is",r2_score(y_test,pred_test_rd)*100)
```

```
mean absolute error 15968.957945495158
mean squared error 410152733.11535543
Root mean squared error 20252.227855605306
r2_score is 86.5336156323644
```

```
In [184]: plt.figure(figsize=(8,6))
          plt.scatter(x=y_test,y=pred_test_rd,color='b')
          plt.plot(y_test,y_test,color='b')
          plt.xlabel('Actual',fontsize=14)
          plt.ylabel('Predicted',fontsize=14)
          plt.title('Ridge Regression',fontsize=18)
          plt.show()
```



DECISION TREE REGRESSOR

A 1D regression with decision tree.

The decision trees is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

We can see that if the maximum depth of the tree (controlled by the `max_depth` parameter) is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

Parameter tuning using GridsearchCV

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
parameters={'criterion':['mse','friedman_mse','mae','poisson'],'splitter':['best','random'],'max_features':['auto', 'sqrt', 'log2
dt=DecisionTreeRegressor()
clf=GridSearchCV(dt,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'criterion': 'mse', 'max_features': 'auto', 'min_weight_fraction_leaf': 0.1, 'random_state': 0, 'splitter': 'best'}
```

In [186]:
```
dt=DecisionTreeRegressor(criterion='mse',max_features= 'auto', min_weight_fraction_leaf= 0.1, random_state= 1, splitter= 'random'
dt.fit(x_train,y_train)
pred_test_dt=dt.predict(x_test)
pred_train_dt=dt.predict(x_train)
dt_score = dt.score(x_train,y_train)
dt_acc_score=r2_score(y_test,pred_test_dt)
print("Accuracy score is:",dt_acc_score*100)
print("score of model is:",dt_score*100)
```

```
Accuracy score is: 56.739912249500414
score of model is: 59.56587171045029
```

In [187]:
```
cv_score_dt=cross_val_score(dt,x,y,cv=5)
cv_mean_dt=cv_score_dt.mean()
print("cv_mean is:",cv_mean_dt*100)
```

```
cv_mean is: 54.77017402545449
```

In [188]:
```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_dt))
print("mean squared error",mean_squared_error(y_test,pred_test_dt))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_dt)))
print("r2_score is",r2_score(y_test,pred_test_dt)*100)
```

```
mean absolute error 27542.39758460575
mean squared error 1317595186.7466893
Root mean squared error 36298.69400883025
r2_score is 56.739912249500414
```

```
In [189]: plt.figure(figsize=(8,6))
          plt.scatter(x=y_test,y=pred_test_dt,color='b')
          plt.plot(y_test,y_test,color='b')
          plt.xlabel('Actual',fontsize=14)
          plt.ylabel('Predicted',fontsize=14)
          plt.title('Decision Tree Regression',fontsize=18)
          plt.show()
```

GRADIENT BOOSTING REGRESSOR

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

Gradient Boosting algorithm is used to generate an ensemble model by combining the weak learners or weak predictive models. Gradient boosting algorithm can be used to train models for both regression and classification problem. Gradient Boosting Regression algorithm is used to fit the model which predicts the continuous value.

```
from sklearn.datasets import make_regression
from sklearn.ensemble import GradientBoostingRegressor
parameters={'loss':['ls','lad','huber','quantile'],'n_estimators':[50,100],'criterion':['friedman_mse','mse'],'learning_rate':[0.
gbr=GradientBoostingRegressor()
clf=GridSearchCV(gbr,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

```
{'criterion': 'friedman_mse', 'learning_rate': 0.05, 'loss': 'ls', 'max_depth': 9, 'min_weight_fraction_leaf': 0.1, 'n_estimato
rs': 100}
```

n [191]:

```
gbr=GradientBoostingRegressor(criterion='friedman_mse',loss='huber',n_estimators=100,learning_rate= 0.05,max_depth= 5, min_weight
gbr.fit(x_train,y_train)
pred_test_gbr=gbr.predict(x_test)
pred_train_gbr=gbr.predict(x_train)
gbr_score = gbr.score(x_train,y_train)
gbr_acc_score=r2_score(y_test,pred_test)
print("Accuracy score is:",gbr_acc_score*100)
print("score of model is:",gbr_score*100)
```

```
Accuracy score is: 86.45666565561379
score of model is: 91.42770499004375
```

n [192]:

```
cv_score_gbr=cross_val_score(gbr,x,y,cv=5)
cv_mean_gbr=cv_score_gbr.mean()
print("cv_mean is:",cv_mean_gbr*100)
```

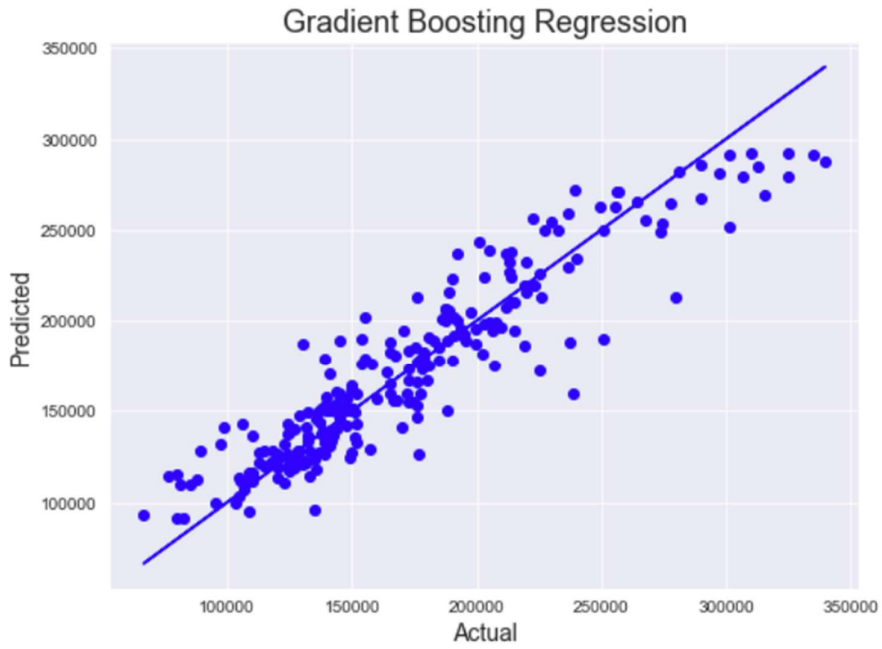
```
cv_mean is: 85.68376275819702
```

In [193]:

```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_gbr))
print("mean squared error",mean_squared_error(y_test,pred_test_gbr))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_gbr)))
print("r2_score is",r2_score(y_test,pred_test_gbr)*100)
```

```
mean absolute error 15056.858210198094
mean squared error 423034655.31872797
Root mean squared error 20567.806283576476
r2_score is 86.11066851589156
```

In [194]:
```
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=pred_test_gbr,color='b')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual',fontsize=14)
plt.ylabel('Predicted',fontsize=14)
plt.title('Gradient Boosting Regression',fontsize=18)
plt.show()
```



SUPPORT VECTOR REGRESSOR

Support Vector Machine is a discriminative algorithm that tries to find the optimal hyperplane that distinctly classifies the data points in N-dimensional space(N - the number of features). In a two-dimensional space, a hyperplane is a line that optimally divides the data points into two different classes. In a higher-dimensional space, the hyperplane would have a different shape rather than a line.

Support Vector Regression (SVR) is quite different than other Regression models. It uses the Support Vector Machine (SVM, a classification algorithm) algorithm to predict a continuous variable.

In [196]:
```
sv=SVR(kernel='poly',gamma='scale',degree= 5, max_iter= -1, shrinking= False, tol= 0.0001)
sv.fit(x_train,y_train)
pred_test_sv=sv.predict(x_test)
pred_train_sv=sv.predict(x_train)
sv_score = sv.score(x_train,y_train)
sv_acc_score=r2_score(y_test,pred_test)
print("Accuracy score is:",sv_acc_score*100)
print("score of model is:",sv_score*100)
```

```
Accuracy score is: 86.45666565561379
score of model is: 11.726083440170997
```

In [197]:
```
cv_score_sv=cross_val_score(sv,x,y,cv=5)
cv_mean_sv=cv_score_sv.mean()
print("cv_mean is:",cv_mean_sv*100)
```

```
cv_mean is: 12.97159352216371
```

In [198]:
```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_sv))
print("mean squared error",mean_squared_error(y_test,pred_test_sv))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_sv)))
print("r2_score is",r2_score(y_test,pred_test_sv)*100)
```

```
mean absolute error 39821.19206088683
mean squared error 2676268099.0014367
Root mean squared error 51732.659887168345
r2_score is 12.131135593679875
```

ELASTIC NET REGRESSION(COMBINATION OF L1 AND L2)

Elastic net is a combination of the two most popular regularized variants of linear regression: ridge and lasso. Ridge utilizes an L2 penalty and lasso uses an L1 penalty. With elastic net, you don't have to choose between these two models, because elastic net uses both the L2 and the L1 penalty!

In [201]:
```
enr=ElasticNet(alpha=0.0001,fit_intercept= True,normalize= False, random_state= 0, tol= 0.001)
enr.fit(x_train,y_train)
pred_test_enr=enr.predict(x_test)
pred_train_enr=enr.predict(x_train)
enr_score = enr.score(x_train,y_train)
enr_acc_score=r2_score(y_test,pred_test)
print("Accuracy score is:",enr_acc_score*100)
print("score of model is:",enr_score*100)
```

```
Accuracy score is: 86.45666565561379
score of model is: 88.32896018871593
```

In [202]:
```
cv_score_enr=cross_val_score(enr,x,y,cv=5)
cv_mean_enr=cv_score_enr.mean()
print("cv_mean is:",cv_mean_enr*100)
```

```
cv_mean is: 84.42384788777652
```

In [203]:
```
#evaluation metrics
print("mean absolute error",mean_absolute_error(y_test,pred_test_enr))
print("mean squared error",mean_squared_error(y_test,pred_test_enr))
print("Root mean squared error",np.sqrt(mean_squared_error(y_test,pred_test_enr)))
print("r2_score is",r2_score(y_test,pred_test_enr)*100)
```

```
mean absolute error 15949.423689244759
mean squared error 407958492.81254786
Root mean squared error 20197.982394599414
r2_score is 86.60565826655125
```

These are all the algorithms used and it described here with the snapshot of their code and the results observed over different evaluation metrics are also mentioned.
The evaluation metrics used here is r2 score.

R2 score:

It is the proportion of the variance in the dependent variable that is predictable from the independent variable(s).” Another definition is “(total variance explained by model) / total variance.” So if it is 100%, the two variables are perfectly correlated, i.e., with no variance at all.

Key Metrics for success in solving problem under consideration

Error addresses exactly this and summarizes on average how close predictions were to their expected values. There are three error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are: Mean Squared Error (MSE). Root Mean Squared Error (RMSE) and Mean Absolute Error(MAE).

We got minimum errors in Ridge Regression Model of our project.

Because it handles multicollinearity with features well and gives good accuracy with less errors.so we used this metrics.

Visualizations

VISUALIZATION

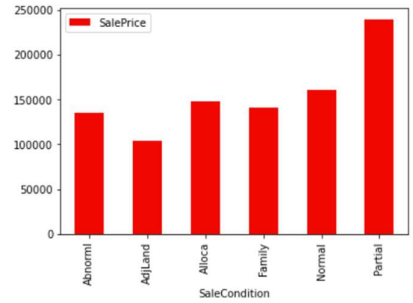
In [87]:
```
sp_pivot1 = df.pivot_table(index='OverallCond', values ='SalePrice', aggfunc=np.median)
sp_pivot1.plot(kind='bar', color='red')
```

Out[87]: <AxesSubplot:xlabel='OverallCond'>

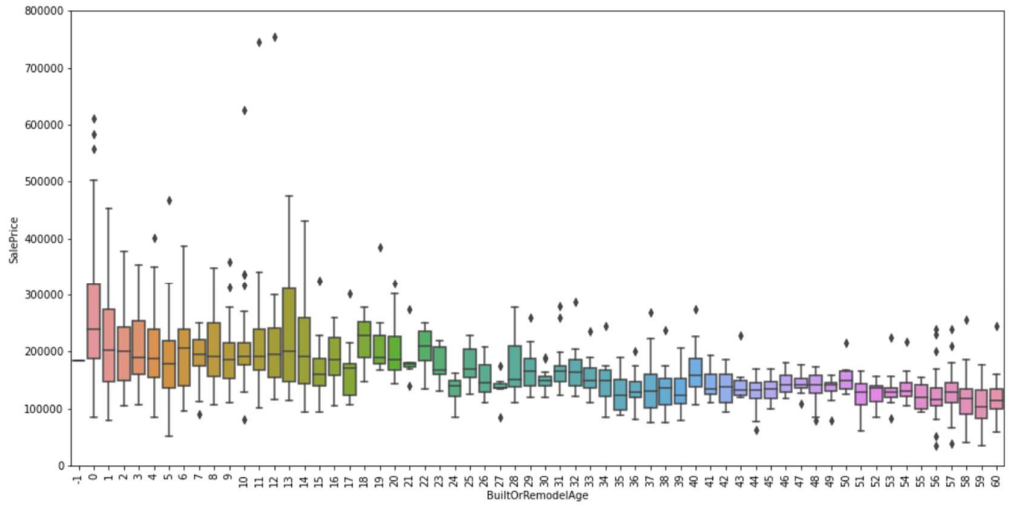
when overall condition increases from average to excellent the sale price increases.we can buy average condition home in less price and we can sell in higher prices with good returns

In [88]:
```
sp_pivot = df.pivot_table(index='SaleCondition', values ='SalePrice', aggfunc=np.median)
sp_pivot.plot(kind='bar', color='red')
```

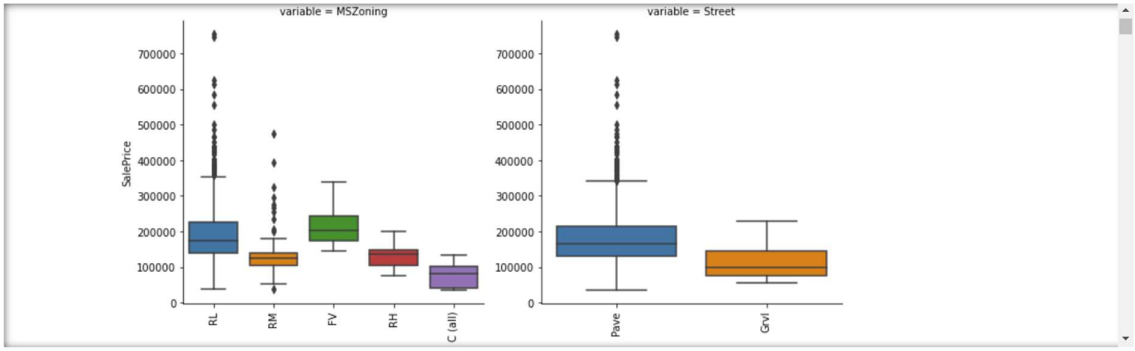
Out[88]: <AxesSubplot:xlabel='SaleCondition'>



when sale condition is partial the sale price increases so we can buy land from other category sale condition land and we can sell in higher sale prices.so that we can get good returns



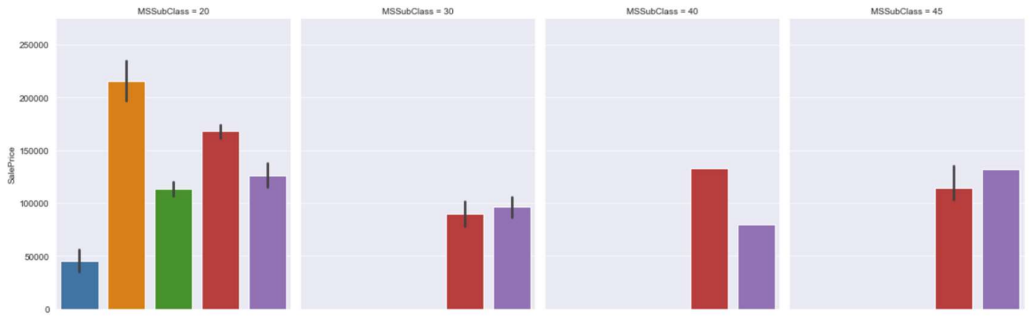
when age of the building is less then sale price increases.we can see when age is 1 and 2 years the sale price is very high



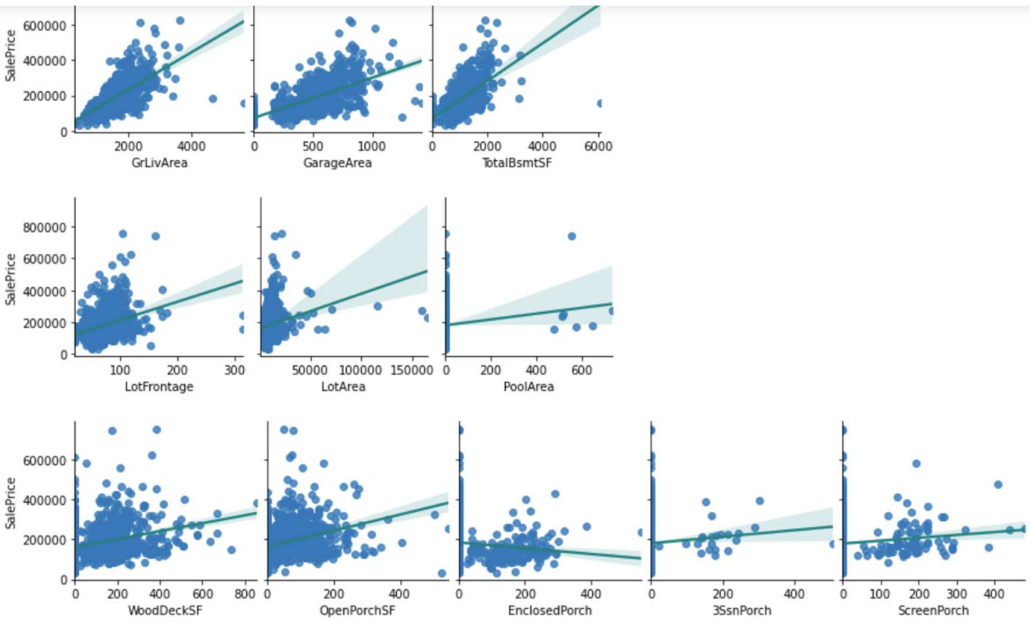
1.sale price is high when ms zone is RL and good when ms zone is FV 2.sale price is high when street is paved type and low when street is grvl type 3.sale price is high when lotshape is IR1 4.sale price is high when landcontour is Lvl 5.sale price is high when neighbourhood is stoneBr 6.sale price is high when landslope is Gtl 7.sale price is high when house style is 1story and 2story 8.sale price is high when bldgtype 1farm 9.sale price is high when electrical is SBrkr 10.sale price is high when garage quality is excellent and garage type is TA 11.sale price is high when sale type is WD and new 12.sale price is high when sale condition is normal and partial

```
In [290]: sns.factorplot(x='MSZoning', y='SalePrice', col='MSSubClass', data=df, kind='bar', col_wrap=4, aspect=0.8)
```

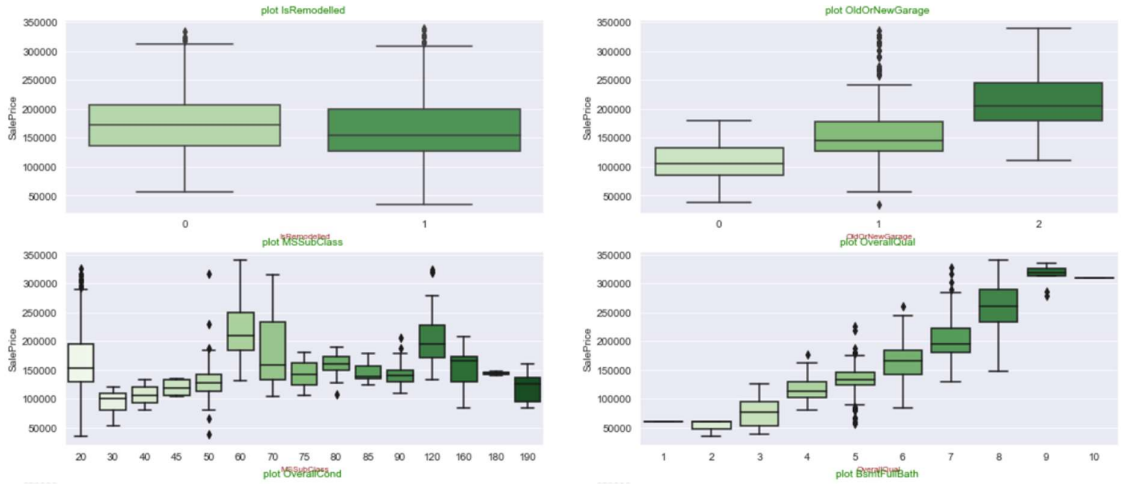
```
Out[290]: <seaborn.axisgrid.FacetGrid at 0x222fa0e4490>
```



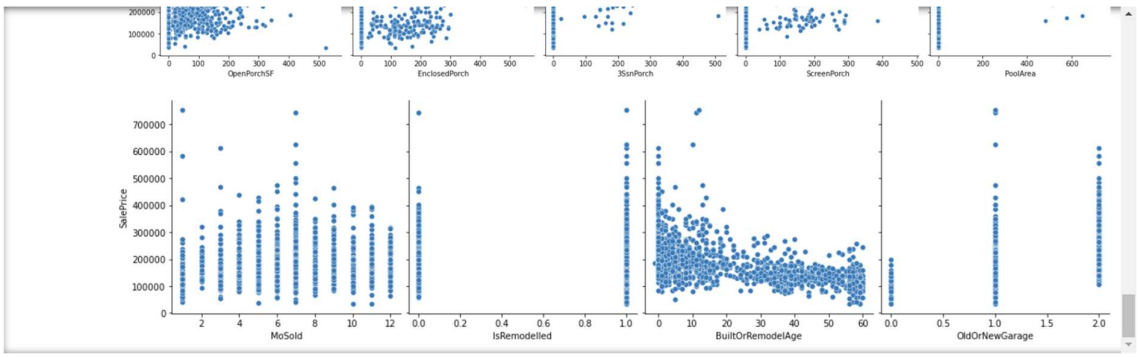
sale price is higher when ms class is 20,60,70,120,160 mostly and with higher number of houses



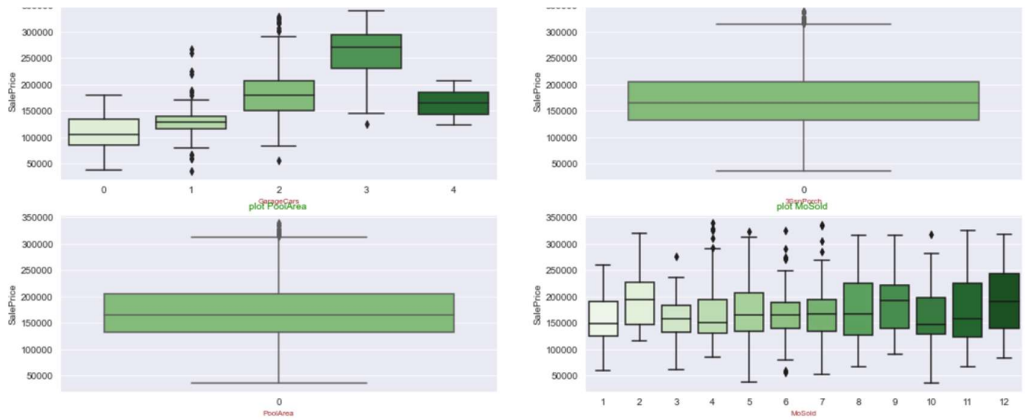
the above all are results of area vs price.it clearly shows when the area increases the price will also increase for the house



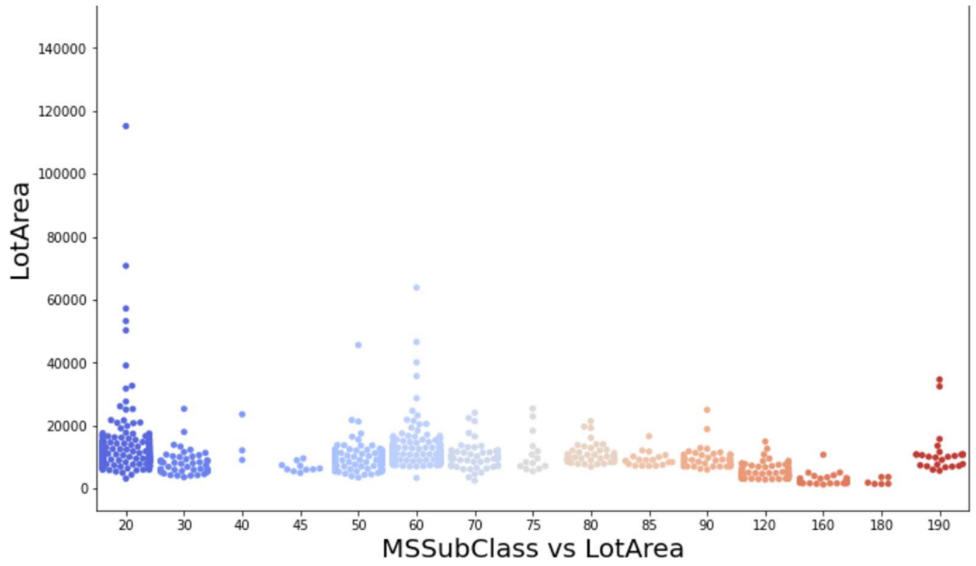
1.when garage is new price increases 2.when remodelled then price increases 3.when bedroom abv grnd increases the price increases 4.fireplace 2,3,4 the price increases



1.the higher count of selling is there when mssubclass is 20 and 60 2.the higher count of selling is there when bsmtfullbath is 1.0 3.the higher count of selling is there when fullbath is 2 4.the higher count of selling is there when firplaces is 1 5.the higher count of selling is there when totalroomsabvground is 6 to 9 category 6.the higher count of selling is there when kitchenaboveground is 1 7.the higher count of selling is there when bedroomaboveground is 2 to 4 8.the higher count of selling is there when garage cars can be 2 to 3 9.the higher count of selling is there when house got remodelled 10.the higher count of selling is there when new garage 11.month of selling and pool area has no higher impact on prices 12.otherwise generally for any area of house features increases then price will increase too

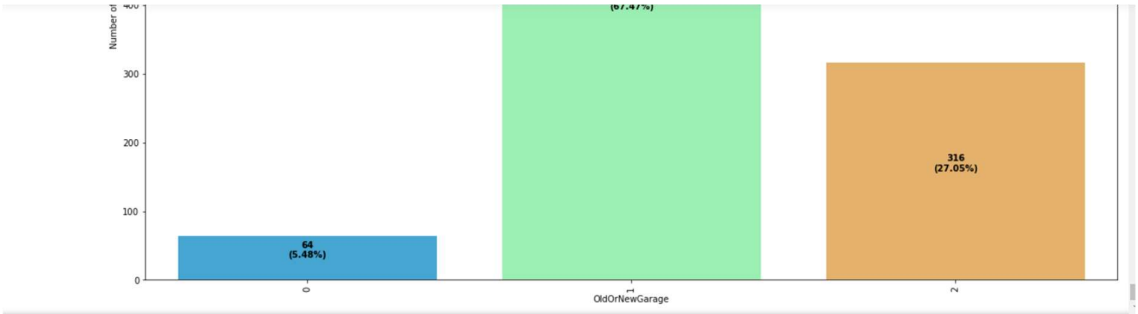


1.when garage is new price increases 2.when remodelled then price increases 3.when bedroom abv grnd increases the price increases 4.fireplace 2,3,4 the price increases

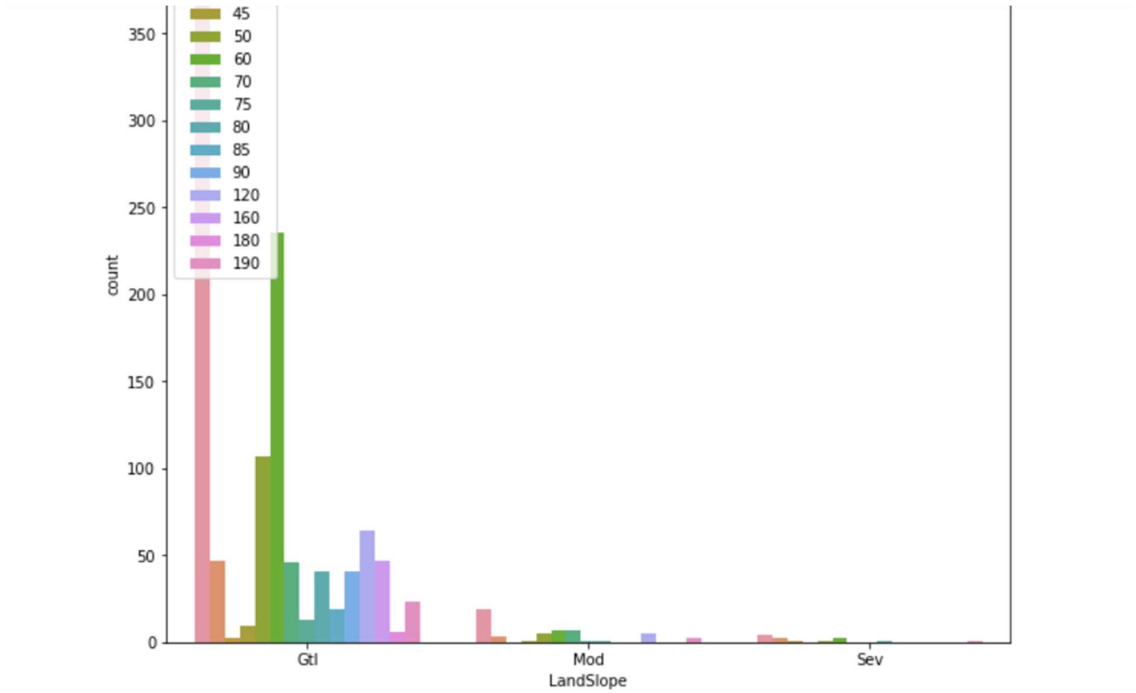


the mssubclass 20 has higher lot areas and higher in selling numbers too

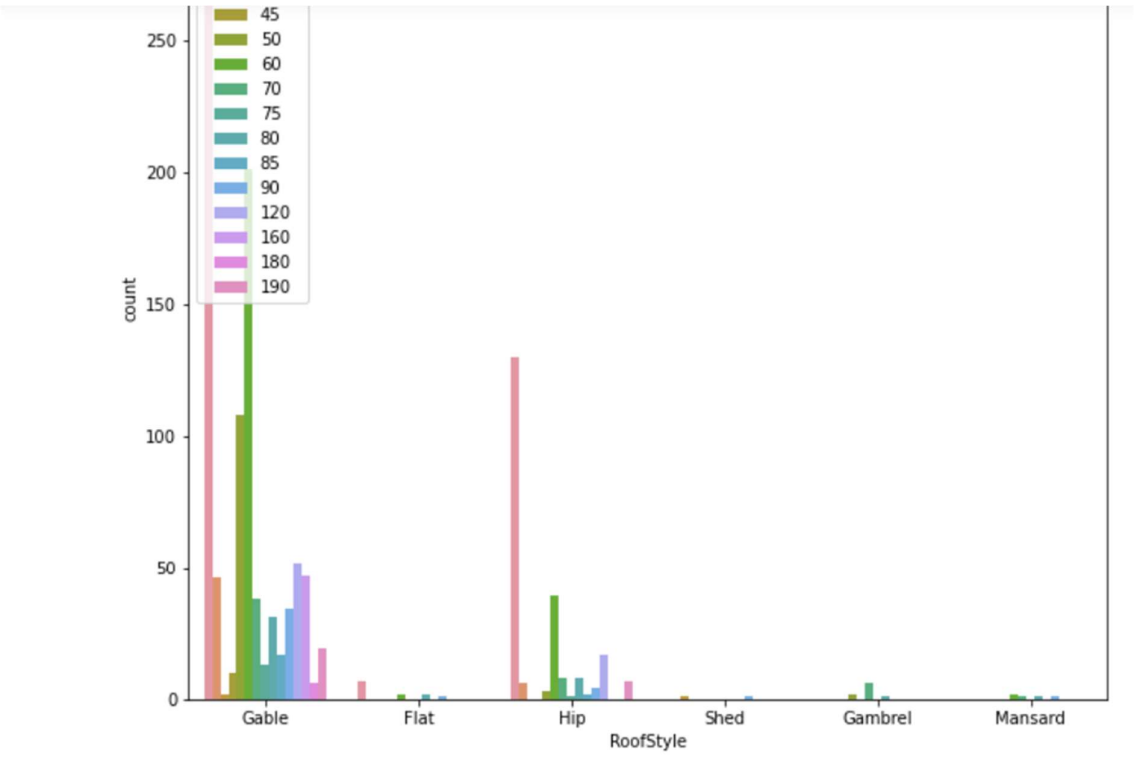
neighbourhood NAmes,Edwards,Sawyers,Crawfor and Mitchel has higher number of houses



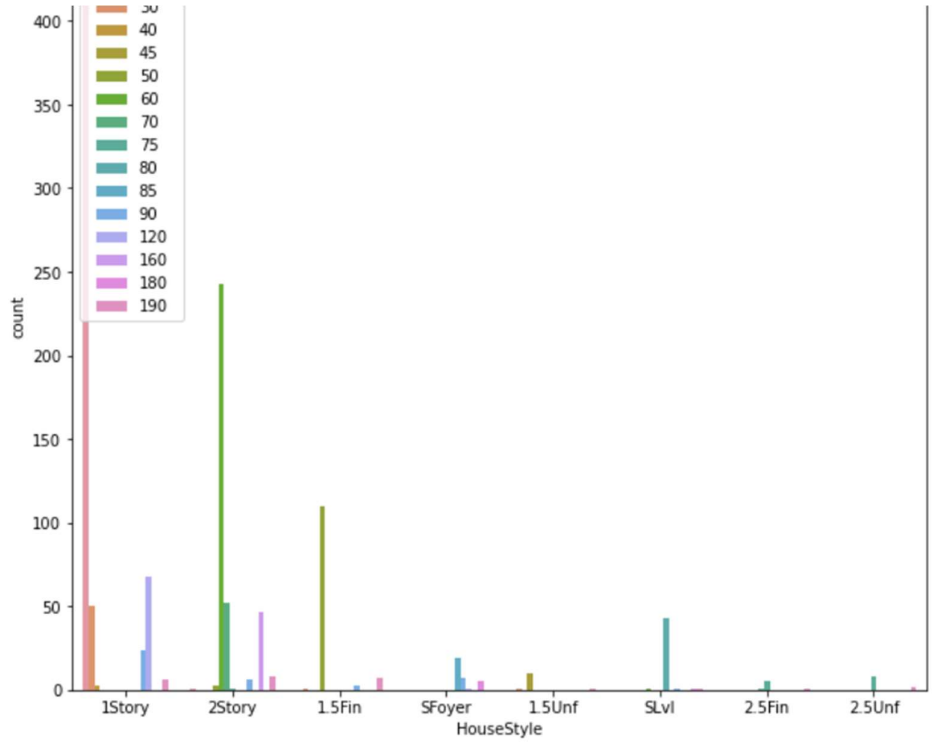
1.mssubclass 20,50,and 60 are higher in numbers.And 20 is the most among all.we can invest in 70 and 120. 2.mszoning RL are higher in numbers and we can invest in RM 3.most of the way are paved way only 4.lotshape Reg are higher in numbers and we can invest in IR1 5.landcontour LVl are higher in numbers. 6.lotconfig inside are higher in numbers and we can invest in corner 5.Neighbourhood NAmes,CollgCr,OldTown,Edwards has higher number of houses we can ivest in these areas 6.all houses selling conditions are mostly in normal only 7.bldtype 1 farm has higher number of houses 8.1story and 2 story are higher in number of type of houses and we can invest in this 9.most of houses has overall quality of 5,6,7 10.most of houses overall condition is 5. 11.most of roofstyle is Gable 12.most of roofmaterial is compshg 13.most of the exterior 1 and 2 is Vnyshld only 14.most of exterior quality and exterior condition is TA 15.most of foundation is CBlock and PConc 16.most of bsmt quality and bsmt condition is TA 17.most of bsmt exposure is NO 18.most of bsmtfintype1 is GLQ and ALQ 19.most of bsmtfintype2 is Unf 20.most of heating is GasA 21.most of heatingQC is TA and EX 22.most of times centralAir is Y 23.most of times Electrical is Sbrkr 24.most of times bsmtfullbath is 0 and 1 25.most of times bsmthalfbath is 0 26.most of times bedroomabvgrnd is 2,3,4 27.most of times kitchenabvgrnd is 1 28.most of times kitchenquality is TA and Gd 29.most of times fireplaces is 0 and 1 30.most of times garagecars is 1,2,3 31.most of times salecondition is normal 32.most of times sale type is WD 33.most of times garage is old 34.50% of garage is remodelled 35.month sold has no impact on sale price



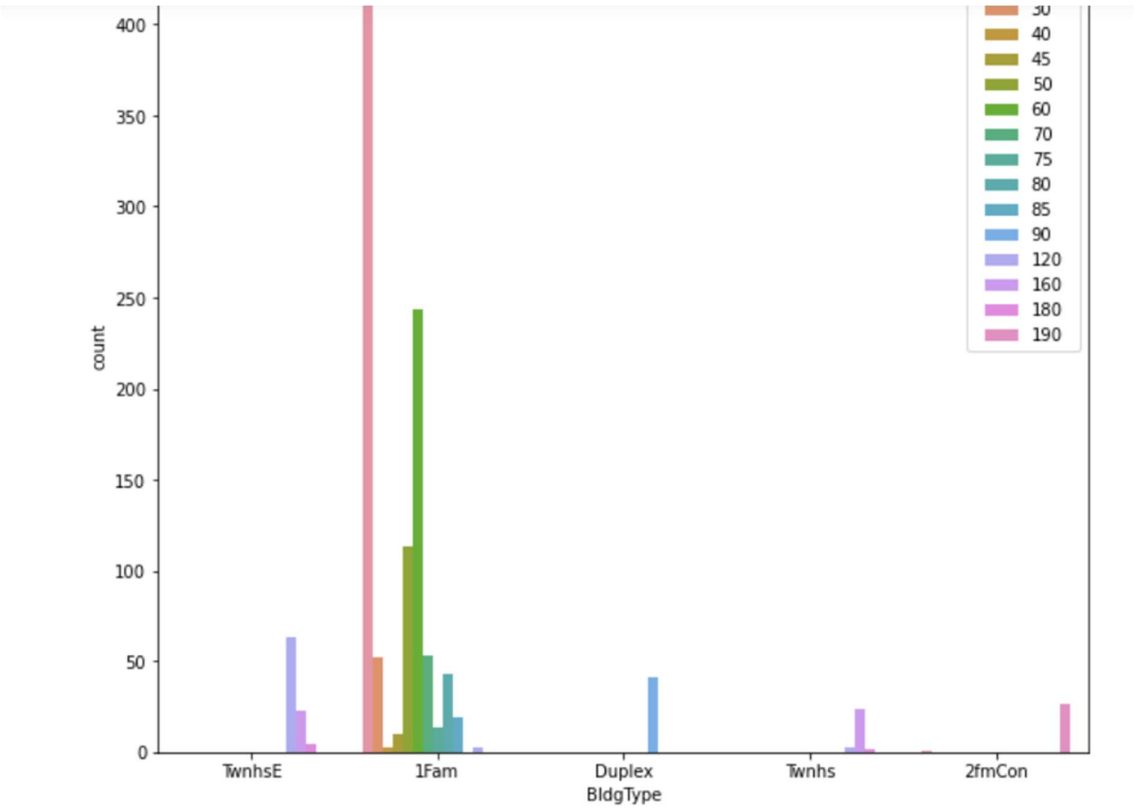
most of landslopes falls under mssubclass 20 and 6o category since this category has higher no of houses



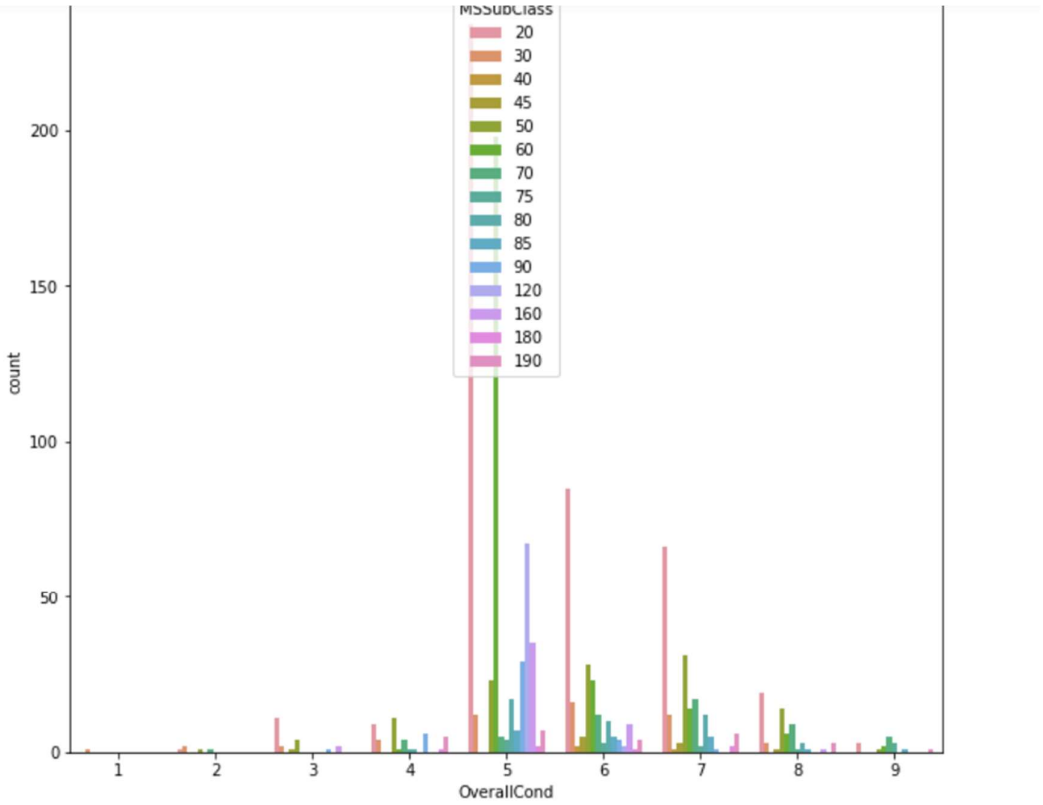
most number of houses are in mssubclass 20 and 60 has mostly gable type roofstyle only



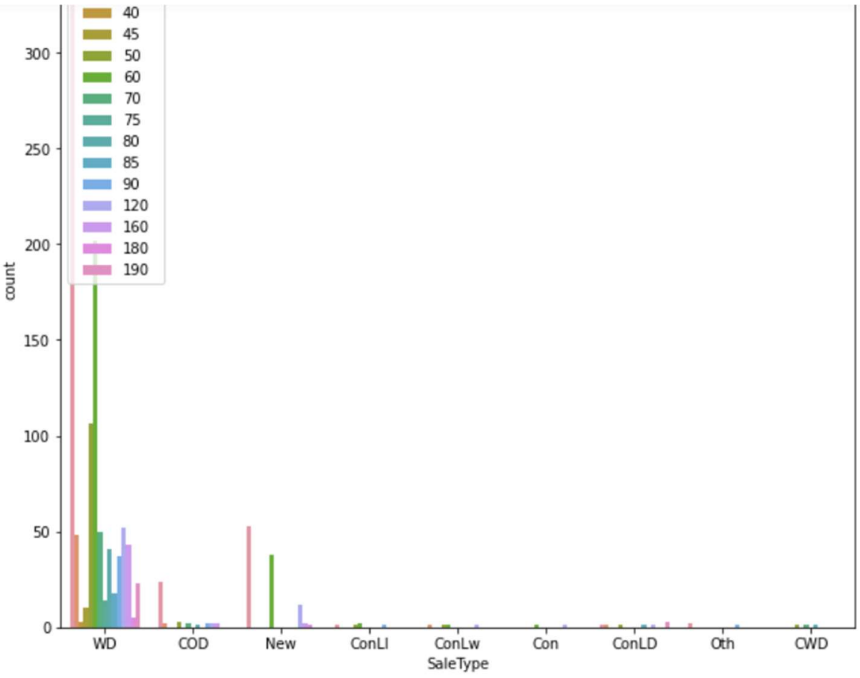
most of houses are mssubclass 20 and 60 and has 1 story and 2 story houstyle mostly in this category only



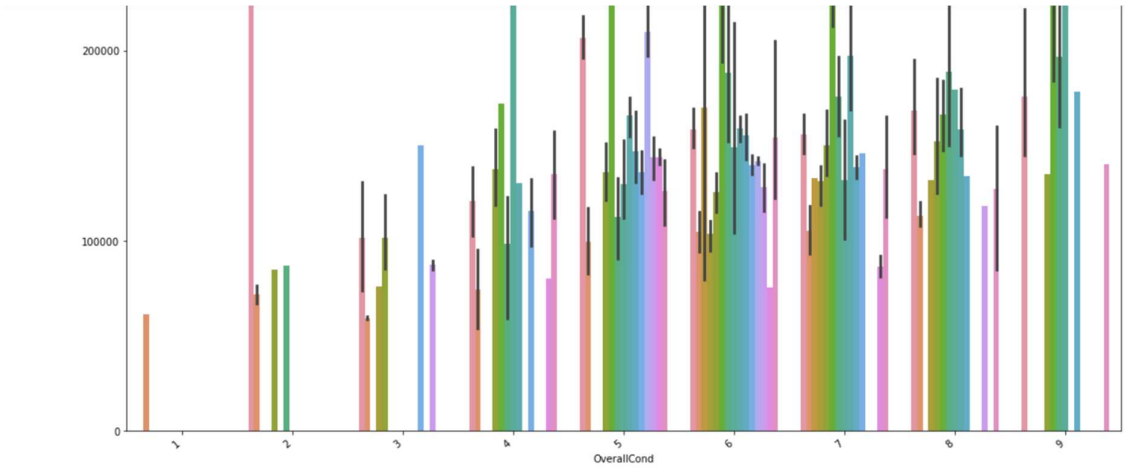
most of the bldgtype is in mssubclass 20 and 60 and has 1 farm category mostly



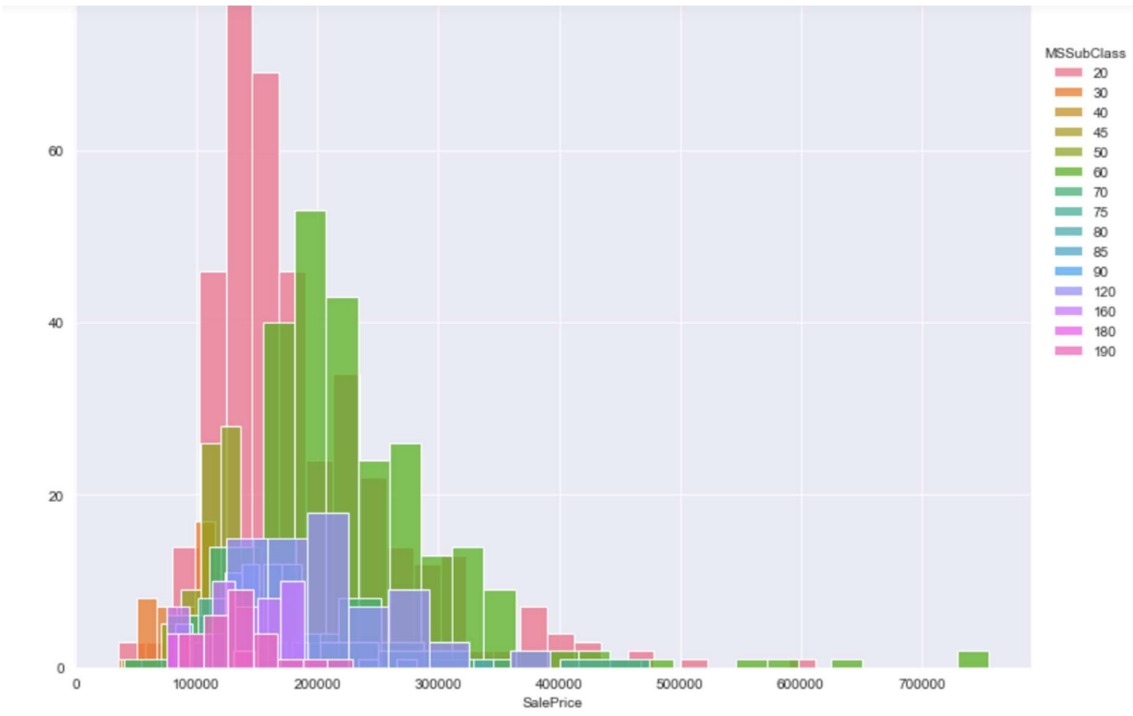
most of houses are in mssubclass 20,60 and has overall condition is 5,6 and 7 in this cateogory only



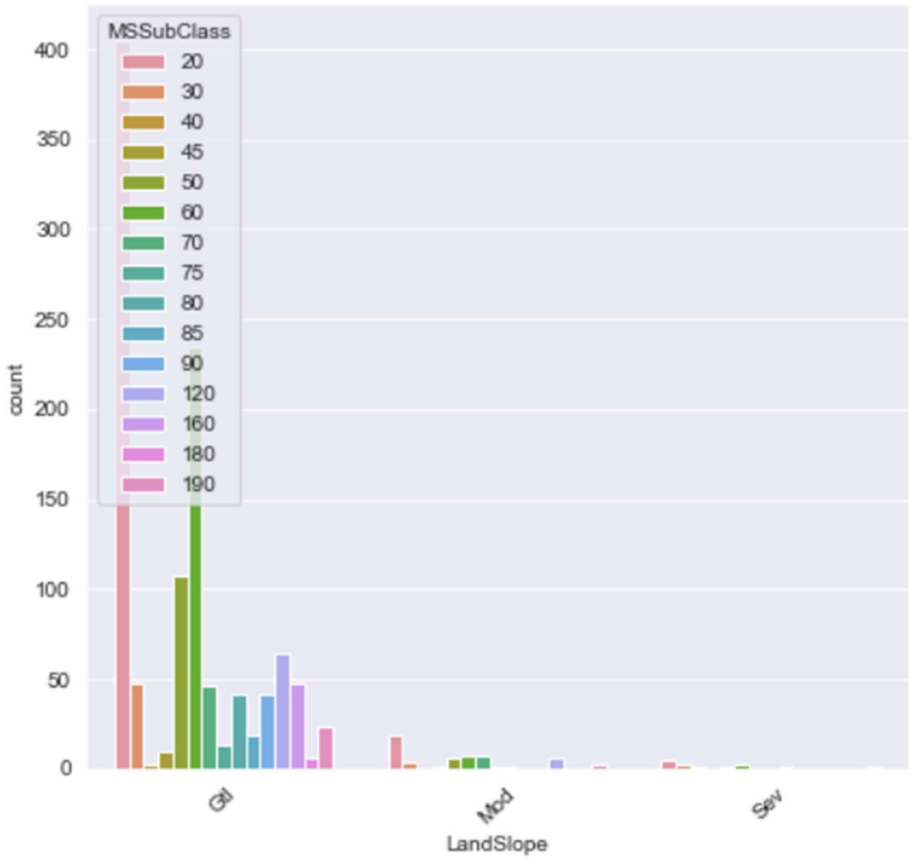
most of houses are in mssubclass 20 and 60 and has WD category of sale type mostly in this category



when overall condition increases in all categories of mssubclass types then their sale value increases for all irrespective of the class types



most number of houses are in mssubclass 20 and 60 category their total sale value is higher in this category among all



all types of mssubclass has landslopes Gtl type only

Interpretation of the Results

In [316]:
```
import pandas as pd
Model_scores=pd.DataFrame({})
Model_scores['Nos']=Nos
Model_scores['Model Names']=models
Model_scores['Scores']=scores
Model_scores.sort_values(by='Scores', ascending=False).style.hide_index()
```

Out[316]:

Nos	Model Names	Scores
4	Ridge Regressor	86.533616
2	Lasso regression	86.470846
1	Linear Regression	86.456666
3	RandomForestRegressor	86.456666
6	GradientBoostingRegressor	86.456666
7	Support Vector Regressor	86.456666
8	ElasticNet Regression	86.456666
5	Decison Tree Regressor	56.739912

CONCLUSION

OUTPUT

In [311]:
```
import numpy as np
a=np.array(y_test)
predicted=np.array(rd.predict(x_test))
df_con=pd.DataFrame({"Original":a,"Predicted":predicted},index=range(len(a)))
df_con
```

Out[311]:

	Original	Predicted
0	145000	148571.397100
1	178400	211333.507861
2	188000	166696.733465
3	251000	242855.053860
4	144000	150721.663737
5	186500	191905.658076
6	219210	230812.765539
7	117500	108290.089995
8	301000	260219.634870
9	146000	163631.681074
10	142500	159343.880455

MAKE PREDICTION BY USING OUR SEPERATE NEW TEST DATA SET

Predict the output for new unseen Test data

In [318]:
```
pred_test_new_raw_data=lr.predict(test_data)
```

In [323]:
```
newdf = df_con.join(df_ouput)
```

In [324]:
```
newdf
```

Out[324]:

	Original	Predicted	Predicted Test Set Result
0	145000	148571.397100	290838
1	178400	211333.507861	197967
2	188000	166696.733465	232452
3	251000	242855.053860	174621
4	144000	150721.663737	214608
5	186500	191905.658076	92637
6	219210	230812.765539	142365
7	117500	108290.089995	5701072
8	301000	260219.634870	180172
9	146000	163631.681074	166200
10	142500	159343.880455	126816

The predicted result for the new unseen test set is most similar to the actual results.so thus our model performs well and we can save this model for prediction

MODEL SAVING:

In [312]:
```
import pickle
filename='HOUSING_SALES_PRICE _PREDICTION__rd.pkl'
pickle.dump(rd,open(filename,'wb'))
```

Inferences:

shows houses with zone,condition,age,sale type and price details
the mssubclass type 20,60,50 which are 1farm category only most available
the building type 1farm and housestyle 1story and 2 story are most available
the housestyle 1story and 2story which is in RL zone are most available
the housestyle 1story and 2story with condition normal are most available
when overallcondition is 9 which is is excellent then sale price will be increased
when sale condition is partial then sale price also increase
when overallquality very excellent then sale price also increases and will be high
mssubclass 20 are higher in numbers and has higher sum amount
mssubclass 20 are higher in numbers and has higher amount of selling price too

In [14]:
```
most_corr = pd.DataFrame(cols)
most_corr.columns = ['Most Correlated Features']
most_corr
```

Out[14]:

	Most Correlated Features
0	SalePrice
1	OverallQual
2	GrLivArea
3	GarageCars
4	GarageArea
5	TotalBsmtSF
6	1stFlrSF
7	FullBath
8	TotRmsAbvGrd
9	YearBuilt
10	YearRemodAdd
11	GarageYrBlt
12	MasVnrArea
13	Fireplaces
14	BsmtFinSF1

These are the most correlated features with the target column

These are the most correlated features.The price of the house mostly depends on these features.

CONCLUSION

Key Findings and Conclusions of the Study

Saleprice of the house mostly depends on overall condition,salecondition,house zones and types,garage area,lot

area,totalroom above ground,age of the house,remodelled or not,fireplace,Full Bath etc.,

These are the variables are important to predict the price of variable. These variables describes the price of the house because these depends on environment,people's living style,and needs of Australia.
All Houses are fixed with price depends on the above all prescribed variables only.

Learning Outcomes of the Study in respect of Data Science

From the above models,Ridge Regressor performs well,Because Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.

This is our Best Fit Model.So we save this model for our analysis

We faced multicollinearity because too many features in the dataset.Eventhough the number of features after reduced from 81 to 61.we still face multicollinearity issue.So we used Ridge Regression Model because it handles the problem of multicollinearity very well.

Because to get out of this issue of without completely removing some predictor variables from the model is to use a method known as ridge regression, which instead seeks to minimize the following: where j ranges from 1 to p and $\lambda \geq 0$. This second term in the equation is known as a shrinkage penalty.

To get out of this issue without completely removing some predictor variables from the model is to use a method known as ridge regression, which instead seeks to minimize the following: where j ranges from 1 to p and $\lambda \geq 0$. This second term in the equation is known as a shrinkage penalty.

Thus this Ridge Regression Model performs well.so we saved this model.

Limitations of this work and Scope for Future Work

The biggest drawback of ridge regression is its inability to perform variable selection since it includes all predictor variables in the final model.

Since some predictors will get shrunken very close to zero, this can make it hard to interpret the results of the model.

Handling of multicollinearity with more accuracy can improve the model performance to great extent and can predict any kind of similar datasets with more accurate results.

For Implementation Code check my [srividya89/Housing-project (github.com)]