HOUSING PROJECT

Submitted by:

Leena Chatterjee

ACKNOWLEDGMENT

I wish to express my sincere gratitude to Datatrained Academy and FlipRobo Technologies who gave me the opportunity to do the House Price Prediction Project. It helped me to do a lot of research and I have grasped many new things. The data source is provided by FlipRobo Technologies. The training I have taken from Datatrained, Kaggle and GitHub helped me to complete this project (House Price Prediction Project).

INTRODUCTION

Business Problem Framing:

Houses are one among the required need of every and each person round the globe and thus housing and land market is one among the markets which is one among the main contributors within the world's economy. It's a really large market and there are various companies working within the domain. Data science comes as a really important tool to unravel problems within the domain to assist the businesses increase their overall revenue, profits, improving their marketing strategies and that specialize in changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are a number of the machine learning techniques used for achieving the business goals for housing companies. Our problem is said to at least one such housing company. A US-based housing company named Surprise Housing has decided to enter the Australian market. The corporate uses data analytics to get houses at a price below their actual values and flip them at a better price. For an equivalent purpose, the corporate has collected a knowledge set from the sale of homes in Australia. The corporate is watching prospective properties to shop for houses to enter the market.

The company wants to know:

• Which variables are important to predict the worth of variable?

• How do these variables describe the worth of the house?

In the real world the problem can be used to detect the worth of homes with the available independent variables. This model will then be employed by the management to know how precisely the prices vary with the variables. They will accordingly manipulate the strategy of the firm and consider areas which will yield high returns. Further, the models are going to be an honest way for the management to know the pricing dynamics of a replacement market.

Conceptual Background of the Domain Problem:

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.

Goals of the Study:

The main objectives of this study are as follows:

- To apply data preprocessing and preparation techniques in order to obtain clean data
- To build machine learning models able to predict house price based on house features
- To analyze and compare models performance in order to choose the best model

Review of Literature

Trends in housing prices indicate the current economic situation and also are a concern to the buyers and sellers. There are many factors that have an impact on house prices, such as the number of bedrooms and bathrooms. House price depends upon its location as well. A house with great accessibility to highways, schools, malls, employment opportunities, would have a greater price as compared to a house with no such accessibility. Predicting house prices manually are a difficult task and generally not very accurate, hence there are many systems developed for house price prediction. The variety of approaches either consider the entire data for modelling, or split the entire house data and model each partition independently for the choice of prediction model. There were may not sufficient training samples per partition for the latter approach. So, such modelling ignores the relatedness between partitions, and for all prediction scenarios.

Motivation for the Problem Undertaken:

Learning the theoretical background for data science or machine learning are often a frightening experience, because it involves multiple fields of mathematics and an extended list of online resources. By proper practical research and practice I can become better in this field. These suggestions are derived from my mentors/SME's and my own experience in the beginner projects.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

- In the project, we need to predict the price of the house using the provided features in the dataset.

- In the project we have used log transform using the NumPy library to remove the skewness.
- There are outliers in the dataset. So, we used Interquartile (IQR) to treat the outliers. IQR can be found from the difference of 75^{th} percentile and 25^{th} percentile.
- Describe method is used to display all the summary statistics of the dataset.
  - In Model building we split the dataset into train and test data and is scaled. The scaled data is passed into the model for training and results are predicted. The performance of the model is found by error metrics like r2 score, mae, mse.

Data Sources and their formats:

The Data source refers to Housing Project. The dataset is in a csv(comma separated values) format. The whole dataset contains 1168 rows and 80 columns. The following are the features with their description in detail:

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

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

MSZoning: Identifies the general zoning classification of the sale.

A Agriculture

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

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl Gravel
Pave Paved

Alley: Type of alley access to property

Grvl Gravel
Pave Paved
NA No alley access

LotShape: General shape of property

Reg Regular
IR1 Slightly irregular
IR2 Moderately Irregular
IR3 Irregular

LandContour: Flatness of the property

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

Utilities: Type of utilities available

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

NoSeWa Electricity and Gas Only
ELO Electricity only

LotConfig: Lot configuration

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

LandSlope: Slope of property

Gtl Gentle slope
Mod Moderate Slope
Sev Severe Slope

Neighborhood: Physical locations within Ames city limits

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

Somerst Somerset
StoneBr Stone Brook
Timber Timberland
Veenker Veenker

Condition1: Proximity to various conditions

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

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

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

BldgType: Type of dwelling

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

HouseStyle: Style of dwelling

1Story One story

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

OverallQual: Rates the overall material and finish of the house

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

OverallCond: Rates the overall condition of the house

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

YearBuilt: Original construction date

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

RoofStyle: Type of roof

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

RoofMatl: Roof material

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

Exterior1st: Exterior covering on house

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

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

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

MasVnrType: Masonry veneer type

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

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

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

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

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

Foundation: Type of foundation

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

BsmtQual: Evaluates the height of the basement

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

BsmtCond: Evaluates the general condition of the basement

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

BsmtExposure: Refers to walkout or garden level walls

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

NA No Basement

BsmtFinType1: Rating of basement finished area

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

BsmtFinSF1: Type 1 finished square feet
BsmtFinType2: Rating of basement finished area (if multiple types)

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

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

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

HeatingQC: Heating quality and condition

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

CentralAir: Central air conditioning

N No
Y Yes

Electrical: Electrical system

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

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

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

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

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

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

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

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

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

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

GarageType: Garage location

2Types More than one type of garage
Attchd Attached to home
Basment Basement Garage
BuiltIn Built-In (Garage part of house - typically has room above garage)
CarPort Car Port
Detchd Detached from home

NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished
RFn Rough Finished Unf
Unfinished
NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

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

GarageCond: Garage condition

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

PavedDrive: Paved driveway

Y Paved
P Partial Pavement
N Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

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

Fence: Fence quality

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

MiscFeature: Miscellaneous feature not covered in other categories

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

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

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

SaleCondition: Condition of sale

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

The below figure shows the loading of dataset using pandas.

```
# Loading the train dataset
df=pd.read_csv('train.csv')
df.head(5)
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Cond
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	

Data Pre-processing Done:

Data Pre-processing is defined as the process of preparing the data and making it suitable for a machine learning model. It’s the primary and crucial step while creating a machine learning model. When creating a machine learning project, it's not always a case that we encounter the clean and formatted data. And while

doing any operation with data, it's mandatory to wash it and put during a formatted way. So for this, we use data pre-processing task.

In simple terms, Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format. In the present project, data pre-processing steps are as follows:

- **Collection of Data**: The data collected is in the form of a csv file.
- **Importing the required Libraries**: Initially we import the libraries like NumPy, pandas and matplotlib. These libraries are well explained in the below section.

```
# Importing the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

- **Importing the Dataset**: This is also known as loading/reading the data which can be done by using Pandas. Dataset is read by giving the right path where it is located.

```
# Loading the train dataset
df=pd.read_csv('train.csv')
df.head(5)
```

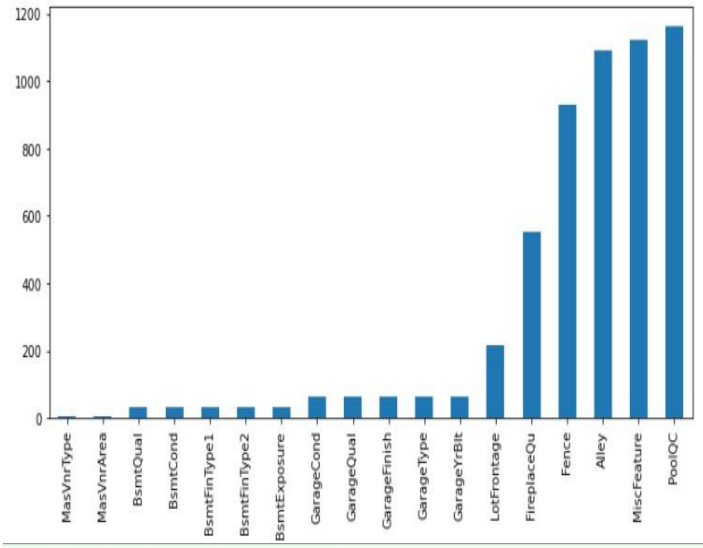
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Con
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	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	

- **Identifying the missing values and Handling them**: In the given dataset, we have more missing values in most of the columns. Some of the features are treated with the suitable values of their respective columns by analysing the dataset. Remaining numerical columns are imputed with mean and object columns with mode.

```
# Checking for null values in the dataset
df.isnull().sum().any()
```

True

```
#lets visualize the null values using barplot
plt.figure(figsize=(11,5))
null_values.sort_values(inplace=True)
null_values.plot.bar();
```




```
# Displaying the columns having null values
null_values=df.isnull().sum()
null_values=null_values[null_values>0]
# Displaying the columns having null values in percentage format
null_percent = null_values * 100 / df.shape[0]
# Concatenating the number and perecentage of missing values into one dataframe and sorting it in Descending order
pd.concat([null_values,null_percent], axis=1,keys=['Null Values', 'Null Percentage']).sort_values(by="Null Values",
                                                                                                  ascending=False)
```

	Null Values	Null Percentage
PoolQC	1161	99.400685
MiscFeature	1124	96.232877
Alley	1091	93.407534
Fence	931	79.708904
FireplaceQu	551	47.174658
LotFrontage	214	18.321918
GarageType	64	5.479452
GarageYrBlt	64	5.479452
GarageFinish	64	5.479452
GarageQual	64	5.479452
GarageCond	64	5.479452
BsmtExposure	31	2.654110
BsmtFinType2	31	2.654110
BsmtCond	30	2.568493
BsmtFinType1	30	2.568493
BsmtQual	30	2.568493
MasVnrArea	7	0.599315
MasVnrType	7	0.599315

```
df["PoolQC"].fillna("No Pool", inplace=True)
df['MiscFeature'].fillna('No feature', inplace=True)
df['Alley'].fillna('No Alley', inplace=True)
df['Fence'].fillna('No Fence', inplace=True)
df['FireplaceQu'].fillna('No Fireplace', inplace=True)
df['LotFrontage'].fillna(0, inplace=True)
```

- **Encoding the categorical data**: This section is of converting Categorical columns to Numerical columns since, model takes only numerical values. There are varieties of encoding techniques. Here, we use Label Encoding to convert the categorical columns. Before using Label Encoding, we import label encoder from sklearn.preprocessing library. In the given dataset, we have nearly half of the features as categorical (object type) columns.

```
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in df.columns:
    if df[column].dtypes == 'object':
        label_encoders[column] = LabelEncoder()
        df[column] = label_encoders[column].fit_transform(df[column])
```

```
df.head(3)
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Conditi
127	120	3	0.0	4928	1	1	0	3	0	4	0	13	2	
889	20	3	95.0	15865	1	1	0	3	0	4	1	12	2	
793	60	3	92.0	9920	1	1	0	3	0	1	0	15	2	

- **Handling the Outliers:** Outliers may affect the model. In the present project, there are many outliers in most of the features. So, by removing all the outliers may result in loss of valuable data. We imputed the outliers of upper bridge and lower bridge values with null values and then with median values instead of losing the data.

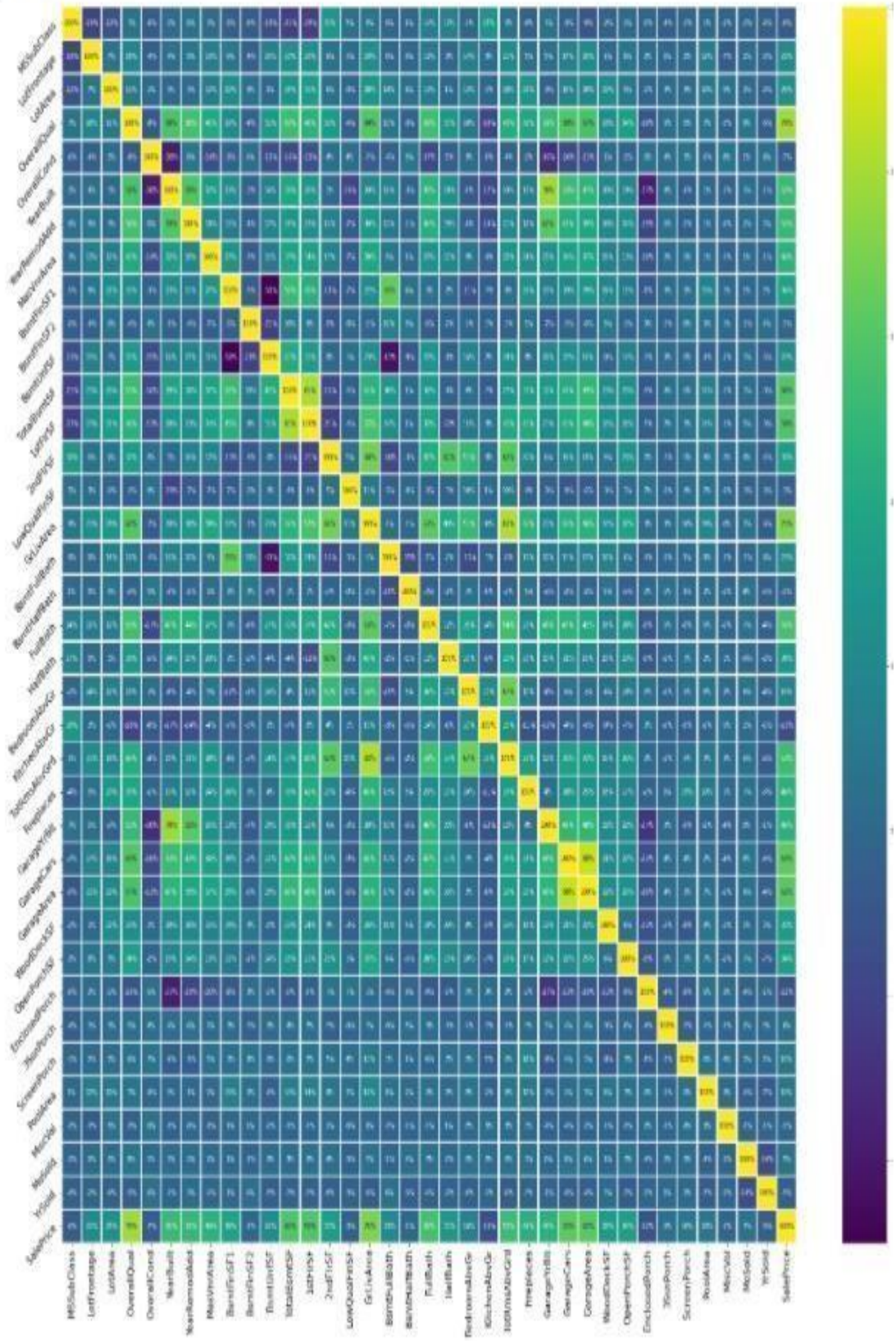
  The above-mentioned steps treat the raw data and set to useful data which is used to build the model and predictions.

```
for col in df.columns:
    IQR = df[col].quantile(0.75)-df[col].quantile(0.25)
    upper_bridge = df[col].quantile(0.75)+(IQR*1.5)
    lower_bridge = df[col].quantile(0.25)-(IQR*1.5)
    df.loc[df[col]>upper_bridge, col]=np.nan
    df.loc[df[col]<lower_bridge, col]=np.nan
# Treating the outliers by imputing the median values
    df[col].fillna(df[col].median(),inplace=True)
```

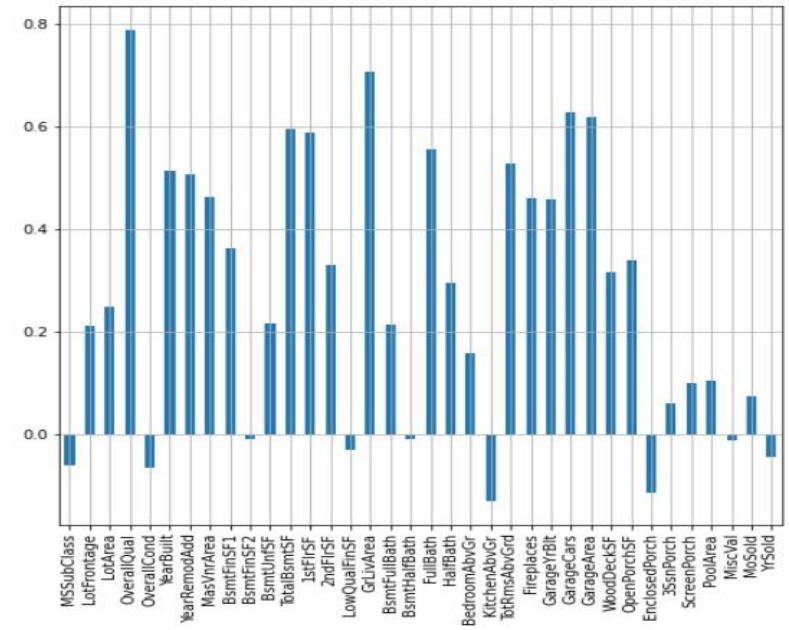
Data Inputs- Logic- Output Relationships:

In the given dataset, label is the output feature and all the remaining features can be taken as input. The relationship between the input and output variables are found by correlation. In general, the correlation can be found using heatmap in which all the relationship between the feature variables can be observed clearly. We can also use different plots by giving the target variable against the remaining columns.

```
# Visualising the correlation among each feature using heatmap
plt.figure(figsize=(30,30))
p = sns.heatmap(df.corr(), annot=True, lw=1.5, fmt='.0%', cmap='viridis')
xlabel = p.set_xticklabels(p.get_xticklabels(),fontdict={'fontsize':15}, rotation=90)
ylabel = p.set_yticklabels(p.get_yticklabels(),fontdict={'fontsize':15}, rotation=45)
```



```
# Visualizing the correlation
plt.figure(figsize=(8,8))
df.drop('SalePrice',axis = 1).corrwith(df['SalePrice']).plot(kind='bar',grid=True);
```



❾ From the Graph, we infer 'OverallQual' is highly correlated and 'KitchenAbvgr' is negatively correlated with the 'SalePrice'(Target Variable).

Set of assumptions (if any) related to the problem under consideration:

In the given dataset, 1161 out of 1168 houses have 0 pool area which means no pools and the PoolQC column have same 1161 missing values. So, we can replace the null values in "PoolQC" column with "No Pool".

We can see that Misc Val column has 1126 entries with a value of 0. Misc Feature has 1124 missing values. Then, as with Pool QC, we can say that each house without a "miscellaneous feature" has a missing value in Misc Feature column and a value of 0 in Misc Val column. So let's fill the missing values in Misc Feature column with "No Feature"

Hardware and Software Requirements and Tools Used:

Hardware:HP Laptop, Intel Core i5 Processor, 8th Generation

Software: The complete project is done using Jupyter Notebook.

Libraries and Packages used:

Initially we load the basic libraries such as pandas, seaborn and matplotlib **NumPy:** NumPy is the library used for scientific computing in python. It is also termed as numerical python.

NumPy is imported as “import numpy”.

Pandas: Pandas is mainly used for data analysis. It allows importing data from various file formats.

The Pandas library is imported as “import pandas”

Seaborn: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Matplotlib: Matplotlib is a visualization library in Python for 2D plots of arrays.

sklearn.**preprocessing**:The 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. In general, learning algorithms benefit from standardization of the data set.

LabelEncoder: LabelEncoder library is used to convert categorical columns to numerical columns. It is imported from sklearn.preprocessing package.LabelEncoder is imported as

from sklearn.preprocessing import LabelEncoder

StandardScaler: StandardScaler is used to standardize the data to a single scale and it is imported as

from sklearn.preprocessing import StandardScaler

SciPy: SciPylibrary includes modules for linear algebra, integration, optimization, and statistics. Skew is used to treat the skewness in the dataset. It is imported from scipy.stats library in the following way:

from scipy.stats import skew

StandardScaler: It is used to scale all the values of each feature such that its distribution will have a mean value 0 and standard deviation of 1.It is imported from sklearn.preprocessing library.

from sklearn.preprocessing import StandardScaler train_test_split: Used in breaking our input and target variable into train and test data. In the project we

have split train and test into 80:20 respectively. It is imported from sklearn.model_selectionlibrary. **from sklearn.model_selection import train_test_split**

All the used Algorithms are imported as shown in figure below.

from sklearn import linear_model
from sklearn.linear_model import LinearRegression,Lasso,Ridge from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

Performance metrics:

All the Regression model's performance is found by mae, rmse, r2_score.

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

As some of these libraries are frequently used, we write these in short form as following:

import numpy as np import
pandas as pd import
matplotlib.pyplot as plt import
seaborn as sns

All the machine learning algorithms libraries/packages are explained detail in further sections.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods) The whole problem-solving approach includes the following steps:

- **Problem Framing:** It includes understanding the problem that is whether the problem is of regression or classification. The present project is of Classification type.

- **Data Understanding:** Data understanding means having an intimate grasp of both the distributions of variables and the relationships between variables. It also includes summary statistics and data visualization.

- **Data Cleaning:** The process of identifying and repairing issues with the data is termed as data cleaning. Statistical methods are used for data cleaning. Some of them are outlier detection and imputation. There are many outliers in the present dataset, so we have imputed outliers with the upper bridge values. We can also remove outliers which may lead to loss of valuable data.

- **Data Selection:** The process of reducing the scope of data to those elements that are most useful for making predictions is called data selection.

- **Data Preparation:** It includes the data to identify the features to be selected and removed. In the present dataset, we have set "Id" column as index and included only 44 features. Before passing the data into the model, we should convert all the categorical data into numerical. So, we Label encoder and convert. In addition to this, the skewness is mitigated by "log transformation" method.

- **Model Evaluation:** Model evaluation consists of identifying input and output variables and splitting the dataset into train and test datasets. The output variable is "label" and the remaining features are input variables. In this project, we have split into 80:20 train and test respectively.
- **Model Configuration:** Hyperparameter tuning the models will get the best fit parameters of each and every model. In this project we use GridSearchCV for knowing the best fit parameters. This can be seen detail in the further sections with a snapshot of it.

- **Model Selection:** The process of selecting one method as the solution is called model selection. It includes the regression model performance metrics.

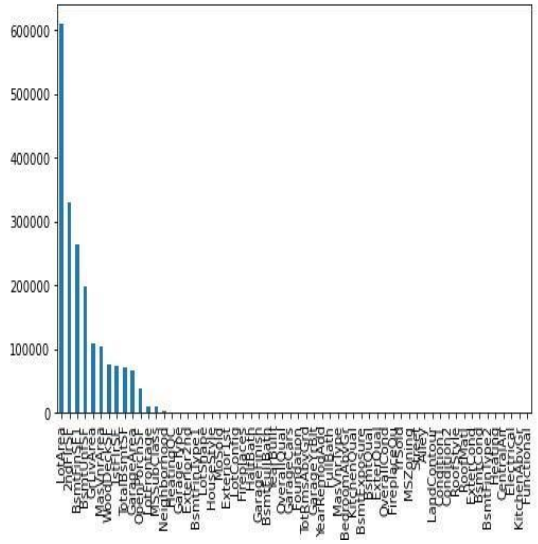
Feature Selection:

Before passing the features into the model, we select some features which is of high importance. The feature selection is shown in below figure.

```
# Filtering out the selected features for model.
filtered_features = [i[0] for i in sorted([[list(df.columns)[x], i] for x, i in enumerate(list(bestfeatures.scores_)) if i > 0.05
print(filtered_features)
```

```
['LotArea', '2ndFlrSF', 'BsmtFinSF1', 'BsmtUnfSF', 'GrLivArea', 'MasVnrArea', 'WoodDeckSF', '1stFlrSF', 'TotalBsmtSF', 'GarageA
rea', 'OpenPorchSF', 'LotFrontage', 'MSSubClass', 'Neighborhood', 'HeatingQC', 'GarageType', 'Exterior2nd', 'BsmtFinType1', 'Lo
tShape', 'HouseStyle', 'MoSold', 'Exterior1st', 'LotConfig', 'Fireplaces', 'HalfBath', 'GarageFinish', 'BsmtFullBath', 'YearBui
lt', 'OverallQual', 'GarageCars', 'Foundation', 'TotRmsAbvGrd', 'GarageYrBlt', 'YearRemodAdd', 'FullBath', 'MasVnrType', 'Bedro
omAbvGr', 'KitchenQual', 'BsmtExposure', 'BsmtQual', 'ExterQual', 'OverallCond', 'FireplaceQu', 'YrSold']
```

```
plt.figure(figsize=(8,5))
feat_importances = pd.Series(bestfeatures.scores_, index=x.columns)
feat_importances.nlargest(60).plot(kind='bar')
plt.show()
```



Testing of Identified Approaches (Algorithms):

The Machine Learning Algorithms used in this project for training and testing to predict the prices of the houses are namely:

- Linear Regression
- Lasso Regression
- Ridge Regression
- Decision Tree Regressor

In addition to the above algorithms, ensembling techniques are also use. They are:

- Random Forest Regressor
- Gradient Boosting Regressor

Linear Regression:

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More

specifically, that y can be calculated from a linear combination of the input variables (x). When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, it refers to the method as multiple linear regression.

Lasso Regression:
Lasso regression is a type linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models .This particular type of regression is well-suited for models showing high levels of multicollinearity.

Ridge Regression:
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.

Decision Tree Regressor:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Random Forest Regressor:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Gradient Boosting Regressor:

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Run and Evaluate selected models:

The above-mentioned algorithms have been run in the jupyter notebook and the performance metrics are found as shown in further below figures:

To check whether the model is overfitting/underfitting GridSearchCV is used and cross validated the models as shown in below figures.

```
#Importing Models library
from sklearn import linear_model
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.tree import DecisionTreeRegressor

#Importing Boosting models
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

#Importing Error Metrics
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.model_selection import GridSearchCV,cross_val_score
```

```
# random state value with which we get a maximum r2_scre
from sklearn.model_selection import train_test_split
model_r = [LinearRegression(),Lasso(),Ridge(),DecisionTreeRegressor(),RandomForestRegressor(),GradientBoostingRegressor()]
for m in model_r:
    max_r_score=0
    for r_state in range (42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_size=0.20)
        m.fit(x_train,y_train)
        y_pred=m.predict(x_test)
        r2_scr=r2_score(y_test,y_pred)
        if r2_scr>max_r_score:
            max_r_score=r2_scr
            final_r_state=r_state
    print('max r2 scoring corresponding to',m,final_r_state,'is',max_r_score)
```

```
max r2 scoring corresponding to LinearRegression() 56 is 0.782175275940563
max r2 scoring corresponding to Lasso() 56 is 0.7821590631267854
max r2 scoring corresponding to Ridge() 56 is 0.7821343567783109
max r2 scoring corresponding to DecisionTreeRegressor() 62 is 0.6732647602430295
max r2 scoring corresponding to RandomForestRegressor() 86 is 0.8109213135167754
max r2 scoring corresponding to GradientBoostingRegressor() 72 is 0.83799452811612
```

We have chosen the best R2 score by running the random state of 42 to 100.

Gradient Boosting Regressor gave the best R2 score with random state of 72.

```
model_params = {

    'LinearRegression':{
        'model':LinearRegression(),
        'params': {
            'fit_intercept':['True','False'],
            'n_jobs':[1,2,3,4]
        }
    },

    'Lasso':{
        'model':Lasso(),
        'params': {
            'alpha':[1.0,2.0,3.0,4.0,5.0],
            'selection':['cyclic','random']
        }
    },

     'Ridge' : {
        'model': Ridge(),
        'params': {
            'max_iter': [1,2,3,4],
            'solver':['auto','svd','sag','cholesky']
        }
    },

        'Decision Tree Regressor': {
        'model':DecisionTreeRegressor(),
        'params':{
           'criterion':['mse','mae','poisson'],
            'splitter':['best','random']
       }
    },

        'random Forest Regressor': {
        'model': RandomForestRegressor(),
        'params' : {
            'criterion':['mse','mae'],
            'n_estimators': [50,70,100,120]
        }
    },


        'Gradient Boosting Regressor':{
        'model':GradientBoostingRegressor(),
        'params':{
            'criterion':['mse','mae','friedman_mse'],
            'n_estimators': [50,70,100,120]
        }
    }
}
```

```
scores = []
from sklearn.model_selection import GridSearchCV,cross_val_score
for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5,scoring="r2", return_train_score=False)
    clf.fit(x, y)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
```

```
# Making a dataframe best which contains model,best_score & best_params
best= pd.DataFrame(scores,columns=['model','best_score','best_params'])
# Viewing the dataframe
best
```

	model	best_score	best_params
0	LinearRegression	0.702125	{'fit_intercept': 'True', 'n_jobs': 1}
1	Lasso	0.702183	{'alpha': 5.0, 'selection': 'cyclic'}
2	Ridge	0.702223	{'max_iter': 1, 'solver': 'auto'}
3	Decision Tree Regressor	0.462377	{'criterion': 'mse', 'splitter': 'best'}
4	random Forest Regressor	0.751208	{'criterion': 'mae', 'n_estimators': 70}
5	Gradient Boosting Regressor	0.750110	{'criterion': 'mae', 'n_estimators': 100}

Passing the best fit parameters into the models.

```
# Passing the obtained best fit parameters into the model
lnr = LinearRegression(fit_intercept=True,n_jobs=1)
ls = Lasso(alpha=5.0,selection='cyclic')
rdg = Ridge(max_iter= 1, solver='auto')
dtr = DecisionTreeRegressor(criterion='mse',splitter='best')
rfr = RandomForestRegressor(criterion='mae',n_estimators=70)
gbr = GradientBoostingRegressor(criterion='mae',n_estimators=100)
```

```
#Using algorithms via loop
model = [lnr,ls,rdg,dtr,rfr,gbr]
for m in model:
    m.fit(x_train,y_train)
    print('score of',m,'is:',m.score(x_train,y_train))
    predm=m.predict(x_test)
    print('Error:')

    Mean_absolute_error=mean_absolute_error(y_test,predm)
    print('Mean absolute error:', Mean_absolute_error)
    meansquarederror=mean_squared_error(y_test, predm)
    print('Mean squared error:', meansquarederror)
    root_mean_squared_error=np.sqrt(mean_squared_error(y_test, predm))
    print('Root Mean Squared Error:',root_mean_squared_error)
    R2_score=r2_score(y_test, predm)
    print('r2_score:',R2_score)
    Cross_Validation_Score=cross_val_score(m,x,y,cv=4,scoring='r2').mean()
    print('Cross Validation Score:',Cross_Validation_Score)
    print('=============================================================================================')
    print('\n')
```

```
score of LinearRegression(n_jobs=1) is: 0.7311673479940296
Error:
Mean absolute error: 19508.350440231086
Mean squared error: 809079663.6002935
Root Mean Squared Error: 28444.325683698207
r2_score: 0.7287643045636902
Cross Validation Score: 0.6961236977384624
=============================================================================================


score of Lasso(alpha=5.0) is: 0.7311666258777711
Error:
Mean absolute error: 19504.804514897052
Mean squared error: 809160177.761015
Root Mean Squared Error: 28445.740942380373
r2_score: 0.7287373130134657
Cross Validation Score: 0.6963018643741454
=============================================================================================


score of Ridge(max_iter=1) is: 0.7311666203855764
Error:
Mean absolute error: 19504.61403842565
Mean squared error: 809022550.480748
Root Mean Squared Error: 28443.321720234224
r2_score: 0.7287834511538165
Cross Validation Score: 0.6963175740762484
=============================================================================================


score of DecisionTreeRegressor() is: 1.0
Error:
Mean absolute error: 27887.615384615383
Mean squared error: 1775633292.974359
Root Mean Squared Error: 42138.26400048249
r2_score: 0.40473707012157034
Cross Validation Score: 0.47657961013452355
=============================================================================================


score of RandomForestRegressor(criterion='mae', n_estimators=70) is: 0.9587655911919614
Error:
Mean absolute error: 16542.954151404152
Mean squared error: 559637352.6044453
Root Mean Squared Error: 23656.655566762714
r2_score: 0.8123872921853688
Cross Validation Score: 0.7441784762652831
=============================================================================================


score of GradientBoostingRegressor(criterion='mae') is: 0.9022410016756991
Error:
Mean absolute error: 14793.313280245146
Mean squared error: 447997945.68185973
Root Mean Squared Error: 21165.96195975651
r2_score: 0.8498132633684785
Cross Validation Score: 0.7317747909764654
=============================================================================================
```

From the figure, we can say that Gradient Boosting Regressor works best.

Key Metrics for success in solving problem under consideration:

Evaluating a model is a major part of building an effective machine learning model. The performance metrics of a Regression model are R2 score, Mean absolute error, mean squared error, Root Mean Squared Error. Among these R2 score should be in between 0 & 1.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=72,test_size=0.20)

gbr.fit(x_train,y_train)
print('score of',gbr,'is:',gbr.score(x_train,y_train))
predgbr=gbr.predict(x_test)
print('Mean absolute error:', mean_absolute_error(y_test,predgbr))
print('Mean squared error:', mean_squared_error(y_test, predgbr))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(y_test, predgbr)))
print('r2_score:',r2_score(y_test, predgbr))
```

```
score of GradientBoostingRegressor(criterion='mae') is: 0.8993446197668933
Mean absolute error: 14777.321228619807
Mean squared error: 459507008.40417814
Root Mean Squared Error: 21436.114582735794
r2_score: 0.8459549676137476
```

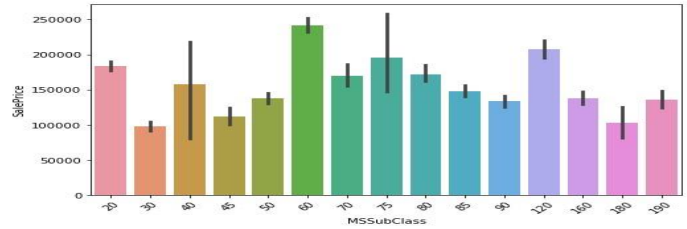
From the figure, we can say that Gradient Boosting Regressor works best. The performance metrics obtained are good.
All the data pre-processing steps are done in test dataset which is similar to train dataset and is executed in a single cell.
The predictions made on the test dataset are shown in below figure:

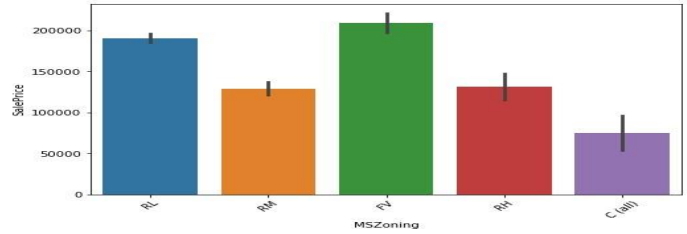
```
# Making predictions on test dataset
gbr.predict(df_test)
```

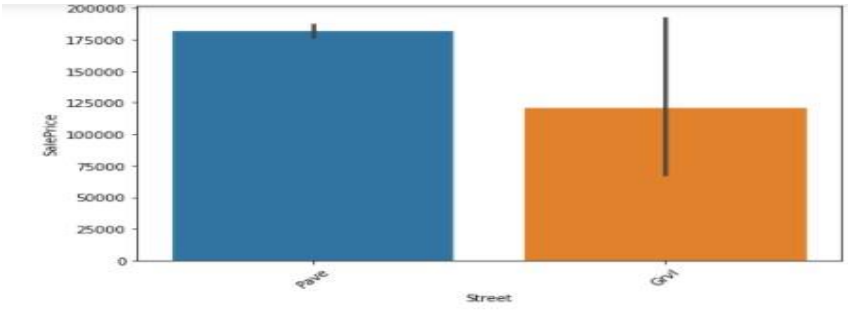
```
array([127398.27264607, 129950.65492034, 165510.07834999, 196135.05814569,
       161101.88325682, 197000.4415838 , 160605.73638052, 156245.24241358,
       119887.49714699, 158352.40479977, 175669.5093013 , 128012.9493068 ,
       203346.62084739, 192169.92344862, 161749.14721307, 175929.81939997,
       182839.27011926, 172606.59976741, 168214.66486563, 142173.16819052,
       196135.05814569, 163686.85282661, 144369.28363243, 252951.63255373,
       169700.0347323 , 141652.46761371, 198857.15417583, 156277.12351622,
       168916.1255953 , 202515.33709584, 223948.9834209 , 158352.40479977,
       130343.61110495, 194412.90336519, 175669.5093013 , 188987.72345202,
       136265.89624782, 182005.20759048, 124581.46858764, 166310.5269203 ,
       170531.31848386, 168214.66486563, 144152.72586254, 170668.89415387,
       170836.5927869 , 197241.48000766, 139577.18633016, 204453.04270937,
       127398.27264607, 127398.27264607, 175524.85826766, 165671.14142361,
       222293.05217595, 204453.04270937, 183819.24552104, 138012.3447461 ,
       219467.35405386, 191561.28456416, 143905.79919446, 190837.95492418,
       144426.49977127, 117271.27518561, 168593.61287032, 129335.9782596 ,
       127398.27264607, 165738.16037896, 170668.89415387, 139714.76200017,
       172911.87407045, 158214.82912975, 168916.1255953 , 142262.12124624,
       220573.77591584, 187419.16308713, 129335.9782596 , 117285.59897596,
       194231.34672538, 143783.52973449, 151101.32048876, 115558.88428268,
       161081.65396231, 195433.59741602, 158520.1034328 , 158214.82912975,
       163209.04564301, 174040.43177001, 204453.04270937, 135564.43551816,
       196135.05814569, 149341.43308802, 201408.91523386, 139577.18633016,
       220573.77591584, 174563.08743932, 150416.93807232, 158352.40479977,
       125322.99136252, 146280.11754323, 135564.43551816, 170836.5927869 ,
       146280.11754323, 192775.66053772, 203651.89515043, 181691.85384034,
       161081.65396231, 147776.45182594, 142173.16819052, 192607.96190469,
       213902.68969626, 219467.35405386, 170974.16845692, 169221.39989834,
       199963.57603781, 151988.99878235, 198857.15417583, 220573.77591584,
       168561.73176768, 167832.80222936, 139577.18633016, 202858.88544015,
       172606.59976741, 161749.14721307, 171724.46202977, 197082.91514638,
       195433.59741602, 136265.89624782, 168593.61287032, 170668.89415387,
       212165.9426498 , 230484.65956942, 128012.9493068 , 115178.43658976,
       139714.76200017, 135545.60391724, 173280.4695311 , 127398.27264607,
       127398.27264607, 144152.72586254, 168377.52357399, 194107.62906215,
       170846.94445108, 135564.43551816, 204745.36071443, 183819.24552104,
       177771.47332937, 173021.50945485, 221718.26400896, 188839.18853401,
       202858.88544015, 163209.04564301, 193688.67624486, 139577.18633016,
       134877.29857884, 158623.47860006, 223788.73512847, 142173.16819052,
       148382.27052951, 196581.57908734, 172606.59976741, 149548.06751031,
       161749.14721307, 141652.46761371, 139714.76200017, 158214.82912975,
       210179.73399444, 127398.27264607, 134877.29857884, 163209.04564301,
       170499.43738121, 198857.15417583, 163373.0245422 , 130343.61110495,
       132281.31671848, 190837.95492418, 148382.27052951, 194231.34672538,
       219467.35405386, 170531.31848386, 143981.05078216, 142173.16819052,
```

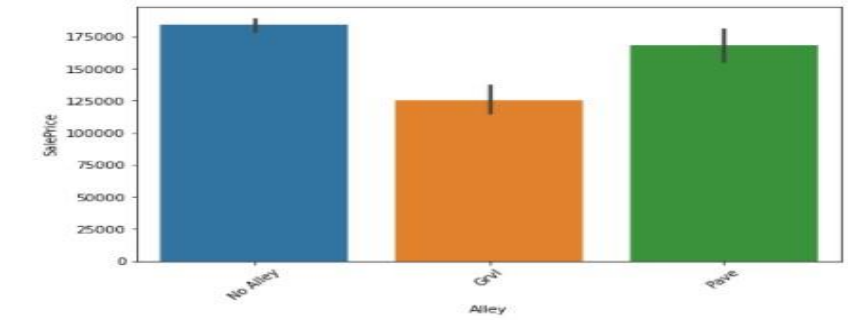
Visualizations:

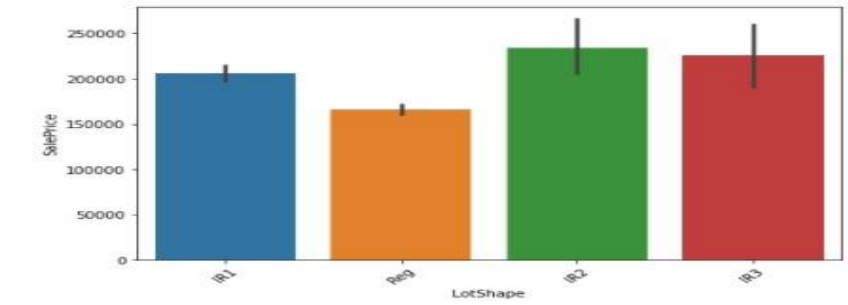
```
for col in Input.columns:
    plt.figure(figsize=(8,5))
    sns.barplot(Input[col],Target)
    plt.xticks(rotation=45)
    plt.show()
```





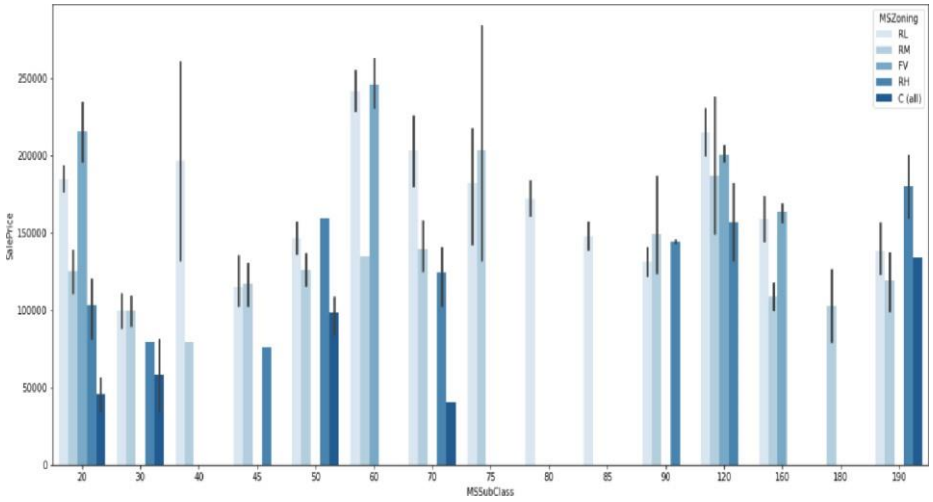


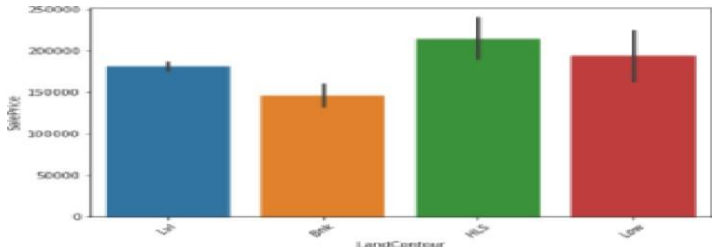


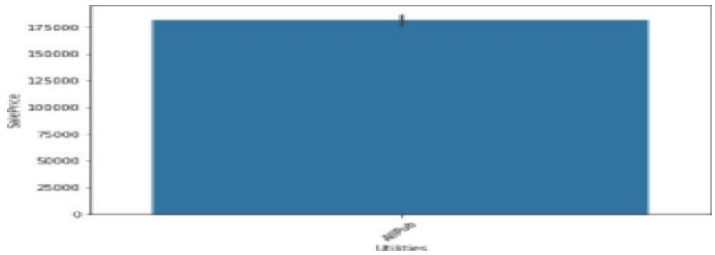


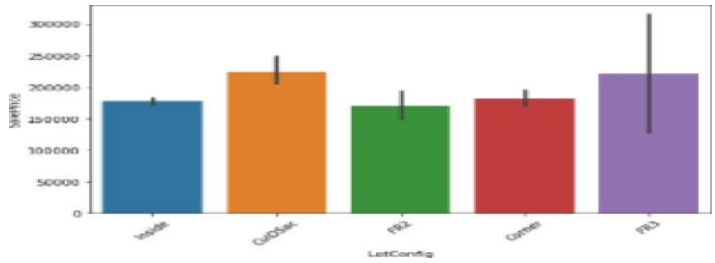
pit, figure(I-igsize=(28,8))
sns. harplot(data=df,x= ' ' , hue= cloning ' , palette= ' Blues ')

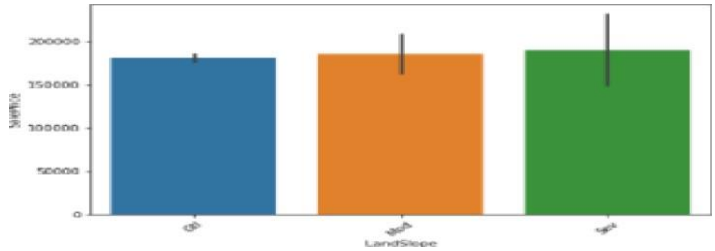
pit ' 1)

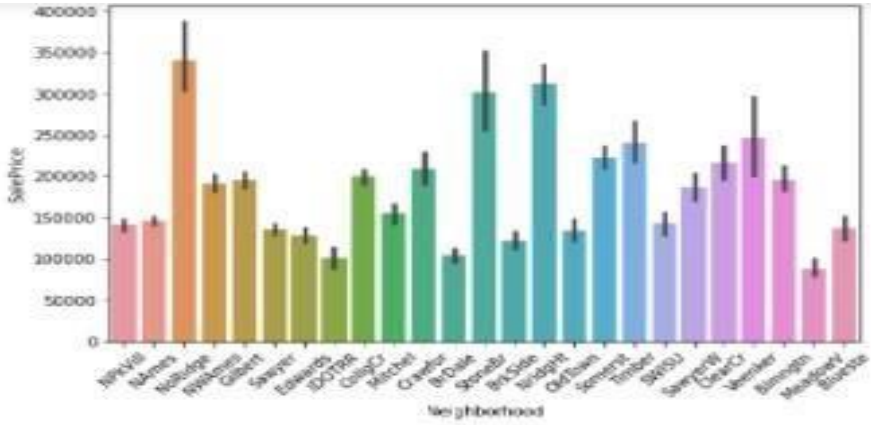

MSZoning
RL
RM
FV
RH
C (all)
SalePrice
MSSubClass

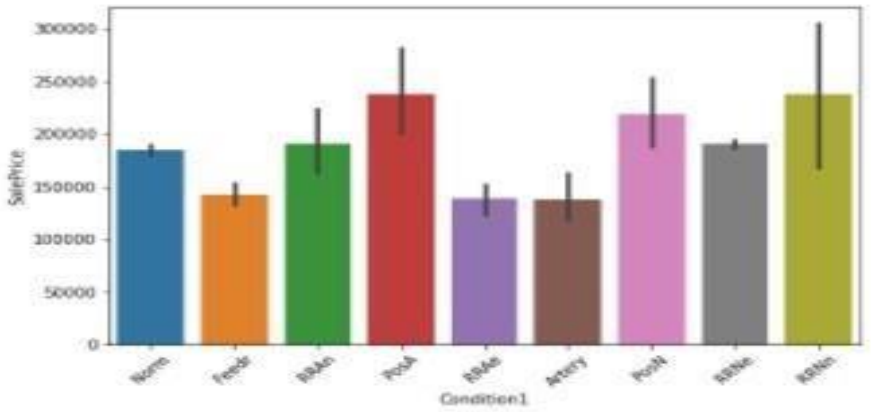

SalePrice
Lvl
Bnk
HLS
Low
LandContour

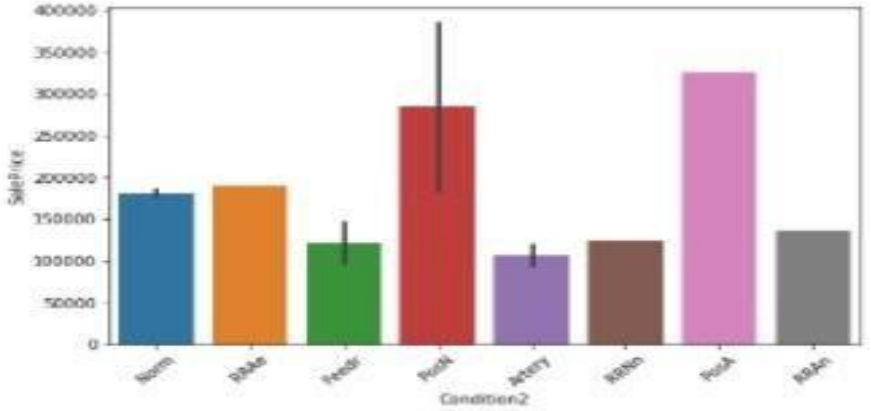

SalePrice
AllPub
Utilities

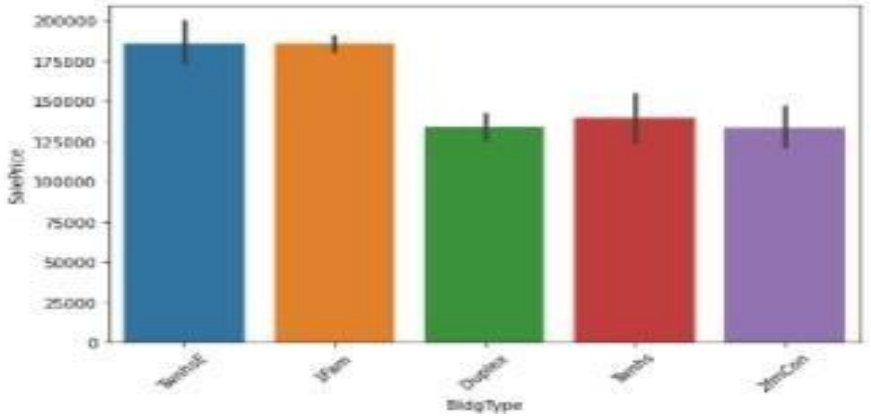

SalePrice
Inside
CulDSac
FR2
Corner
FR3
LotConfig

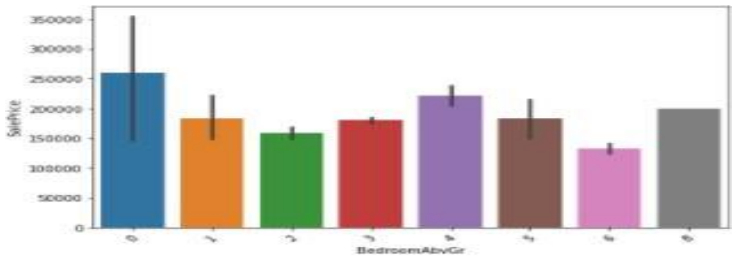

SalePrice
Gtl
Mod
Sev
LandSlope

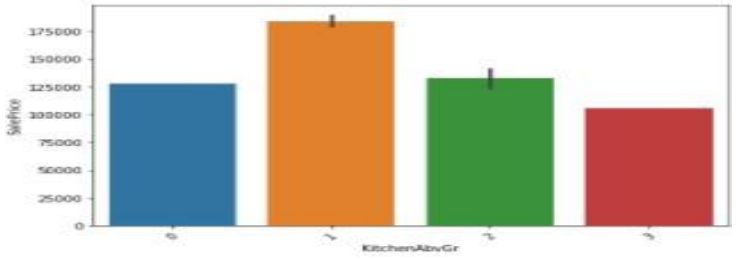

SalePrice
400000
350000
300000
250000
200000
150000
100000
50000
0
NPkVill
NAmes
NoRidge
NridgHt
Gilbert
Sawyer
Edwards
SWISU
BrkSide
Crawfor
Mitchel
NWAmes
OldTown
BrDale
Somerst
BrkSide
NridgHt
OldTown
Somerst
Timber
SWISU
SawyerW
ClearCr
Veenker
Blmngtn
Meadowv
Blueste
Neighborhood

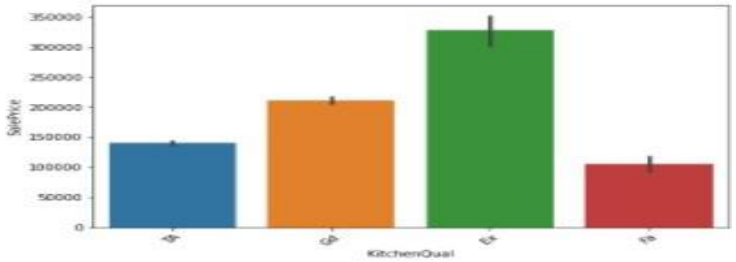

SalePrice
300000
250000
200000
150000
100000
50000
0
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1

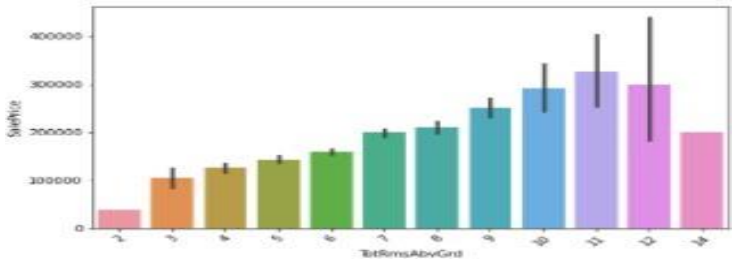

SalePrice
400000
350000
300000
250000
200000
150000
100000
50000
0
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
Condition2


SalePrice
200000
175000
150000
125000
100000
75000
50000
25000
0
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType

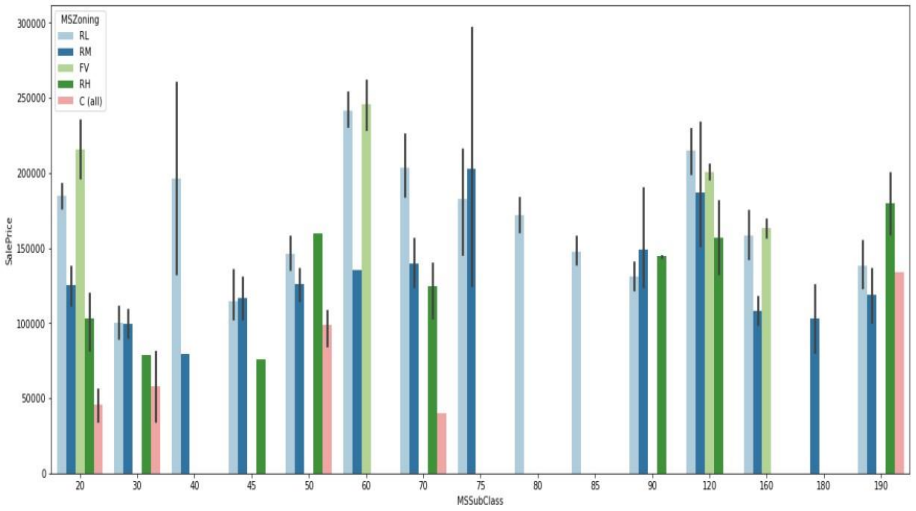

SalePrice
BedroomAbvGr


SalePrice
KitchenAbvGr


SalePrice
KitchenQual

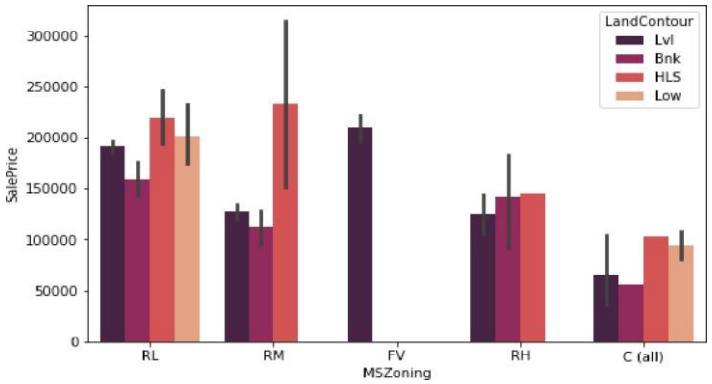

SalePrice
TotRmsAbvGrd

```
plt.figure(figsize=(20,8))
sns.barplot(data=df,x='MSSubClass',y='SalePrice',hue='MSZoning',palette='Paired')
plt.show()
```

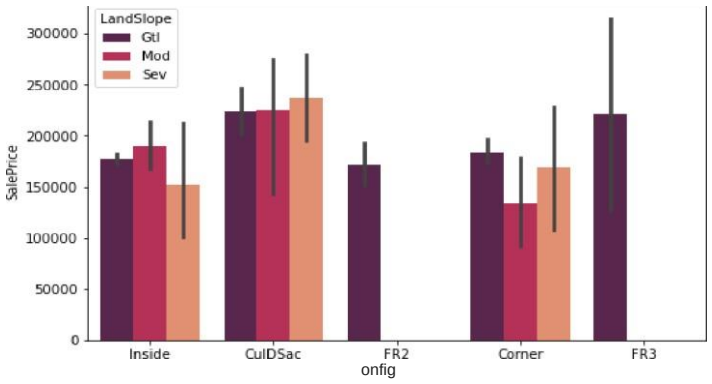


pit . F-1gur e(f1g s1 z e=(8 , 5))
s ns . bar'p1 at (d a ta=dI, x=' l'.SZo I r z ', y= ' S a e P'° ce, ' hue= ' a 4cC a r *cc t - ', pa 1ette= ' roe k e° ')

```
plt.show()
```

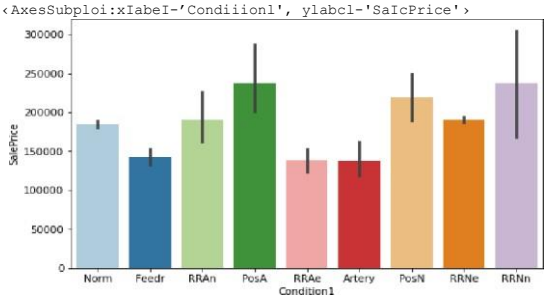


pit . 7lgur (f1g s1 z e=(8 , 5))
s ns . bar'p1 at (d a ta=dI, x=' L a- cor fig , ' y= ' S a ePi i ze ' , ' L ar JS ope,' pad e the= ' roe k a - ') ;



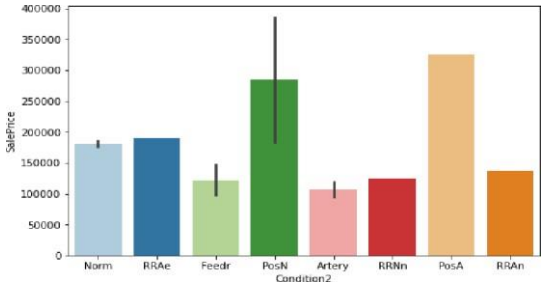

```
plt.figure(figsize=(8 B))
```

‹AxesSubploi:xIabeI-’Condiiion1', ylabcl-'SaIcPrice'›

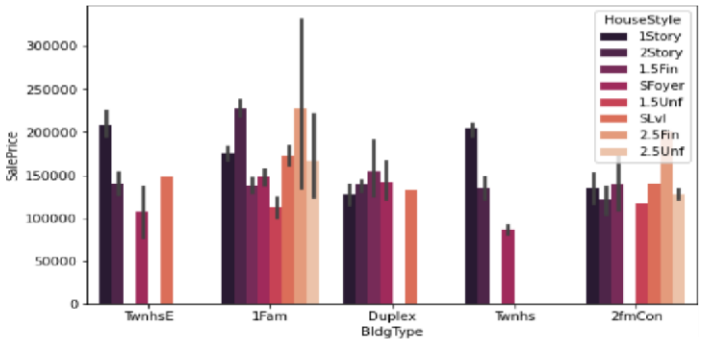



```
plt.figure(figsize=(8 s))
```

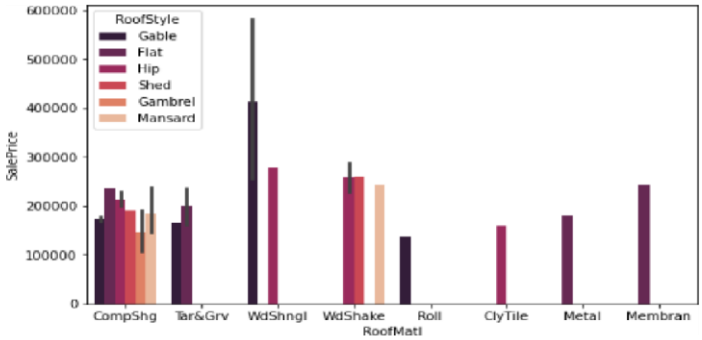
‹AxesSubplol:xIabeI=’Condilion?', ylabcl='SaIcPrice'›



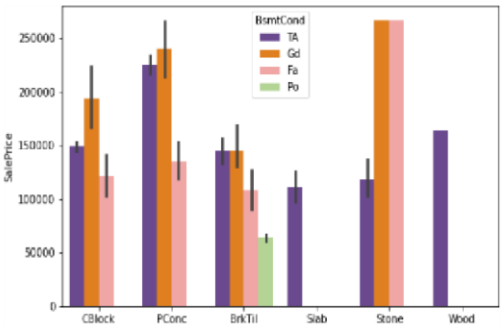

```
pit.figure(figsize=(8 s))
    sns. barplot(data=df.x='bldgType'.y='salePrice',hue='HouseStyle',palette='rocket');
```



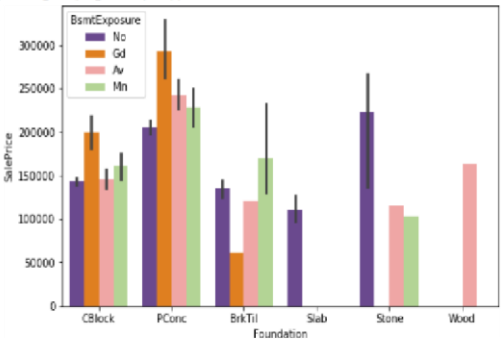

```
plt.figure(figsize=(8,5))
sns. bar'p1ot ( d ata-d-f. hue= 'Rao-f-          '. y= ' S    ePrd  ce  ', x-  'Rao-Ft 'et  l  ', pale l l e= ' no c ke l  ' ) ;
```



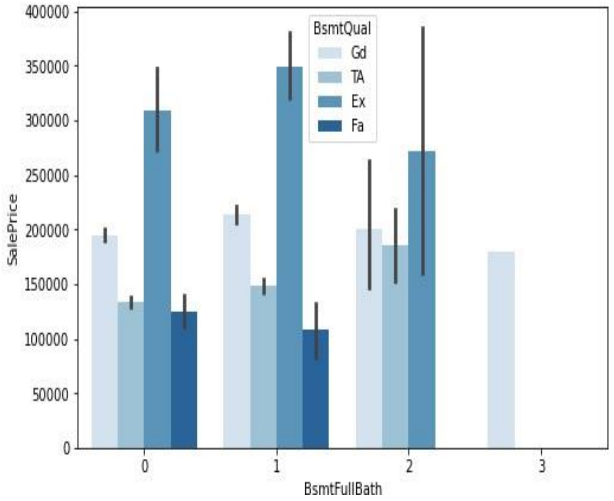

```
plt.figure(figsize=(8,5))
sns.barplot(data=df,x='Foundation',y='SalePrice',hue='BsmtCond',palette='Paired_r');
```



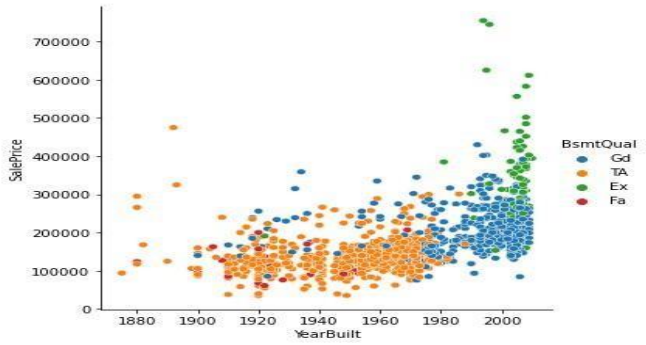

```
sns.barp  at(data=df,x='Foendation',y='Sa1ePrice',hue='BsrtE xpos u  e',oa1ett e='Pa          r ' ) ;
plt.figure(figsize=(8,5))
```



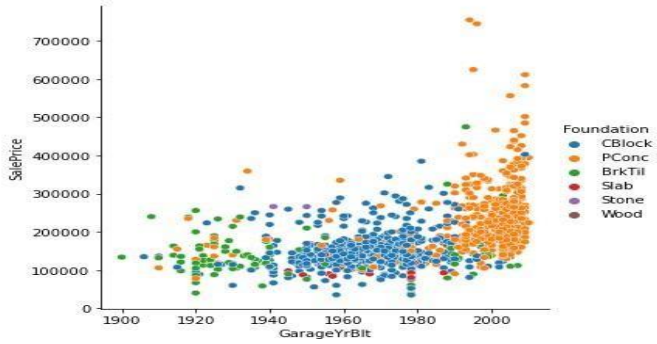
```
plt.figure(figsize=(8,5))
sns.barplot(data=df,x='BsmtFullBath',y='SalePrice',hue='BsmtQual',palette='Blues');
```




```
sns.relplot(data=df,x='YearBuilt',y='SalePrice',hue='BsmtQual');
```




```
sns.relplot(data=df,x='GarageYrBlt',y='SalePrice',hue='Foundation');
```



Interpretation of the Results:

Observations from Visualization:

- There is only one value for the feature Utilities.
- Residential Low Density,Floating Village Residential have high values at MSSubClass of 60.
- There are more Paved houses when compared to Gravel houses

- Hillside landcounter has high SalePrice at MSZoning of Residential Medium Density.
- At condition1 and condition2 Adjacent to postive off-site feature has high SalePrice.
- Wood Shingles of rooftype Gable has high Sale Price

Observations from pre-processing:

- There are some missing values in the dataset. So, we treated with imputation.
- The outliers are treated by imputing with the nan values and then with median of their respective columns.

Observations from modelling:

- Target variable is selected as “SalePrice”
- Split the dataset to 80:20 for train and test respectively.
- Used standard scaler to scale the values
- All the data obtained from above steps given to model and made the predictions.
- Observed the best fit parameters from GridSearchCV
- Gradient Boosting Regressor has the better performance metrics.

CONCLUSION:

Key Findings and Conclusions of the Study:

The goal is to achieve the system which will reduce the human effort to find a house having reasonable price. The proposed system House Price Prediction model approximately tries to achieve the same one. We have managed out how to prepare a model that gives users for a best approach with future lodging value predictions. Proposed system focused on predict the house price according to the area and machine learning methods are used. The experimental results showed that this technique that is used while developing system will give best prediction of house price.

Learning Outcomes of the Study in respect of Data Science:

Data Visualization made the project problem easy to understand easy and every feature. While doing the research of the problem, we have clearly identified and

imputed the columns with missing values. All the Data pre-processing steps made the problem easier to clean the data. New treatments of outliers are learnt. I have always dropped the outliers but, in this project imputed with other values without losing data.

Five Algorithms are used, in which “Gradient Boosting Regressor” has the best performance metrics with R2 score.

Limitations of this work and Scope for Future Work:

- **Limitations:** We couldn’t reach predicting the house price with R2 score of 1 in practical.

Scope for Future Work:

There are quite a few things that can be polished or add in the future work. Though, we were able to identify most of the residential areas. There may be some more places that have housing complexes or multi-storey apartments which are located in commercial areas. These can be counted in future to give a more accurate result. With more and more demand for housing in metropolitan cities, there is a definite increase in the number private builders to attract more customers. There are several other models available that can be implemented for prediction. Data given as input to such model should be compatible with the tool used and the operators involved in the process. Also, more number of data sets can be used to increase the accuracy of the model. The main objective of using a different model should be to reduce the calculation time and carry out the whole process in ease. The usability of system will be further increased by making real time use of Google maps instead of using satellite images of Google map. The use of online database will increase the capacity of system to store any location available on Google map.