FLIP ROBO

HOUSING PRICE PREDICTION PROJECT

SUBMITTED BY:

DEEPAK PAPNEJA

INTERNSHIP - 28

ACKNOWLEDGMENT

I WOULD LIKE TO THANK ALL OF MY MENTORS IN DATATRAINED EDUCATION AND FLIPROBO TECHNOLOGIES WHO GUIDED ME IN THIS ENTIRE JOURNEY OF MACHINE LEARNING PROGRAM SO THAT I HAVE GOT ABILITY TO COMPLETE THESE KIND OF PROJECTS. I WOULD ALSO LIKE TO EXPRESS MY GRATITUDE TOWARDS ANALYTICSVIDHYA.COM, GEEKSFORGEEKS.ORG, STACKOVERFLOW.COM, TOWARDSDATASCIENCE.COM, MEDIUM.COM, ETC. ONLINE SOURCES WHICH I HAVE ALWAYS FOLLOWED IN MY ENTIRE JOURNEY OF MACHINE LEARNING.

INTRODUCTION

It's always said that food, clothes and shelter are the basic necessities of each and every person on this planet. So, to have own house is everybody's dream. And in the context of buying own house, the price is the major factor. The price is the most effective feature to decide whether it is affordable for someone or not. And in this project, I am trying to make a model which predicts the price of a particular house based on its specification/ features/ factors.

- ## Business Problem Framing

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

- ## Conceptual Background of the Domain Problem

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

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

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

  So, in this project, I will use my data analysis and machine learning techniques to check the relationship between the sale price and all other features available. And then using machine learning techniques, I will develop a model which will try to predict the prices with the accuracy as much as possible.

- **Review of Literature**

On the internet, there are a lot of literatures regarding this housing price problem. One of which is by Pow, Janulewicz, & Liu, 2014 in which they have stated that ‘The relationship between house prices and the economy is an important motivating factor for predicting house prices and also there is no accurate system to calculate house prices.’

Another one is by (Khamis & Kamarudin, 2014) who states that Housing market is important for economic activities. Traditional housing price prediction is based on cost and sale price comparison. So, there is a need for building a model to efficiently predict the house price.

The best literature I can mention here is the description of the database provided to me for developing the model.

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

201-STORY 1946 & NEWER ALL STYLES
301-STORY 1945 & OLDER
401-STORY W/FINISHED ATTIC ALL AGES
451-1/2 STORY - UNFINISHED ALL AGES
501-1/2 STORY FINISHED ALL AGES
602-STORY 1946 & NEWER
702-STORY 1945 & OLDER
752-1/2 STORY ALL AGES
80SPLIT OR MULTI-LEVEL
85SPLIT FOYER
90DUPLEX - 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

    FinFinished
    RFn	Rough Finished
    Unf	Unfinished
    NANo 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
    NANo Garage

GarageCond: Garage condition

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

- **Motivation for the Problem Undertaken**

As stated in the above discussion on this valuable project, it is the importance of the prices in the finalization of any house and uncertainty of the prices associated with the houses or in other words, there is no mechanism to justify the price of houses, which led me to work on this project and inspired me to build a model so that my data science skills can be used in making a model that imparts some impact on the society.

I took it as an opportunity to work on this project and enjoyed developing its model as I applied a lot of permutations and combinations of approaches on this model and hence, finalized my model based on the bes

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

There are two datasets in this project which are provided to us. One is Train dataset and other is Test dataset. I developed my model using Train dataset and by using this model, I predicted the 'SalePrice' of Test dataset which is asked in this project. This is a regression problem.

There are a lot of features (81) in this dataset. Some of the features are having a lot of Nan values like ore than 90%. So, I deleted those features. Also, there are some other features which have one value for almost 90% of datapoints so I also dropped those features and hence, developed my model in this way. It will be shown with time to time in this project report with the screenshots attached.
I also did analysis part on this project by analyzing univariate and bivariate analysis and got some decisions based on these plots.

I also plotted some plots at the end showing the deviations between the predicted values of my model with the actual one. Also showed one plot of showing the difference between the actual and predicted values and hance, it was depicted that there was only 4.7% variation between actual and predicted ones.

- ## Data Sources and their formats

The data on which I worked was provided by FlipRoboTechnologies. It has 1460 entries each having 81 variables.
It's screenshot is attached below:

```
In [2]: df1=pd.read_csv('train.csv')
        pd.options.display.max_columns=None
        df1
```

Out[2]:

	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

Screenshot 1. Train dataset

```
In [19]: df2=pd.read_csv('test.csv')
         df2
```

Out[19]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	Corner	Gtl	StoneBr	Norm	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	StoneBr	Norm	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	Inside	Gtl	Crawfor	Norm	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	Somerst	Feedr	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm	
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm	
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Feedr	
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	BrDale	Norm	

292 rows × 80 columns

Screenshot 2. Test dataset

The test dataset has 80 columns rather than training dataset which has 81 columns as the target variable SalePrice is to be found here.
Regarding formats, some columns have integer values, some have float values and some other have text values. This is also shown in the following screenshot depicting the datatypes of this dataset.

Also, I found that the column names of both the datasets were having upper case and lower case characters used for their names. So, I renamed all the columns in both of the datasets to lower case and proceeded with these lowercase names then.

```
In [9]: df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 79 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   mssubclass     1168 non-null   int64
 1   mszoning       1168 non-null   object
 2   lotfrontage    954 non-null    float64
 3   lotarea        1168 non-null   int64
 4   street         1168 non-null   object
 5   alley          77 non-null     object
 6   lotshape       1168 non-null   object
 7   landcontour    1168 non-null   object
 8   lotconfig      1168 non-null   object
 9   landslope      1168 non-null   object
 10  neighborhood   1168 non-null   object
 11  condition1     1168 non-null   object
 12  condition2     1168 non-null   object
 13  bldgtype       1168 non-null   object
 14  housestyle     1168 non-null   object
 15  overallqual    1168 non-null   int64
 16  overallcond    1168 non-null   int64
 17  yearbuilt      1168 non-null   int64
 18  yearremodadd   1168 non-null   int64
 19  roofstyle      1168 non-null   object
 20  roofmatl       1168 non-null   object
 21  exterior1st    1168 non-null   object
 22  exterior2nd    1168 non-null   object
 23  masvnrtype     1161 non-null   object
 24  masvnrarea     1161 non-null   float64
 25  exterqual      1168 non-null   object
 26  extercond      1168 non-null   object
 27  foundation     1168 non-null   object
 28  bsmtqual       1138 non-null   object
 29  bsmtcond       1138 non-null   object
 30  bsmtexposure   1137 non-null   object
 31  bsmtfintype1   1138 non-null   object
 32  bsmtfinsf1     1168 non-null   int64
 33  bsmtfintype2   1137 non-null   object
 34  bsmtfinsf2     1168 non-null   int64
 35  bsmtunfsf      1168 non-null   int64
 36  totalbsmtsf    1168 non-null   int64
 37  heating        1168 non-null   object
 38  heatingqc      1168 non-null   object
 39  centralair     1168 non-null   object
 40  electrical     1168 non-null   object
 41  1stflrsf       1168 non-null   int64
 42  2ndflrsf       1168 non-null   int64
 43  lowqualfinsf   1168 non-null   int64
 44  grlivarea      1168 non-null   int64
 45  bsmtfullbath   1168 non-null   int64
 46  bsmthalfbath   1168 non-null   int64
 47  fullbath       1168 non-null   int64
 48  halfbath       1168 non-null   int64
 49  bedroomabvgr   1168 non-null   int64
 50  kitchenabvgr   1168 non-null   int64
 51  kitchenqual    1168 non-null   object
 52  totrmsabvgrd   1168 non-null   int64
 53  functional     1168 non-null   object
 54  fireplaces     1168 non-null   int64
 55  fireplacequ    617 non-null    object
 56  garagetype     1104 non-null   object
 57  garageyrblt    1104 non-null   float64
 58  garagefinish   1104 non-null   object
 59  garagecars     1168 non-null   int64
 60  garagearea     1168 non-null   int64
 61  garagequal     1104 non-null   object
 62  garagecond     1104 non-null   object
 63  paveddrive     1168 non-null   object
 64  wooddecksf     1168 non-null   int64
 65  openporchsf    1168 non-null   int64
 66  enclosedporch  1168 non-null   int64
 67  3ssnporch      1168 non-null   int64
 68  screenporch    1168 non-null   int64
 69  poolarea       1168 non-null   int64
 70  poolqc         7 non-null      object
 71  fence          237 non-null    object
 72  miscfeature    44 non-null     object
```

Screenshot 3. Datatypes of columns of Trian dataset.

- ## Data Preprocessing Done

Firstly, I checked the NaN values in the dataset and found that there were some features which were having almost 90% NAN values so, I decided to drop these features. Also, there were features id and utilities from which, id was having unique value for each and every datapoint, hence, I dropped that. Also, utilities feature was having one value for all datapoints so I deleted that. To check this, I also used nunique function to give me the unique values associated with each feature and decided based on this. I also plotted its count plot which is attached below:

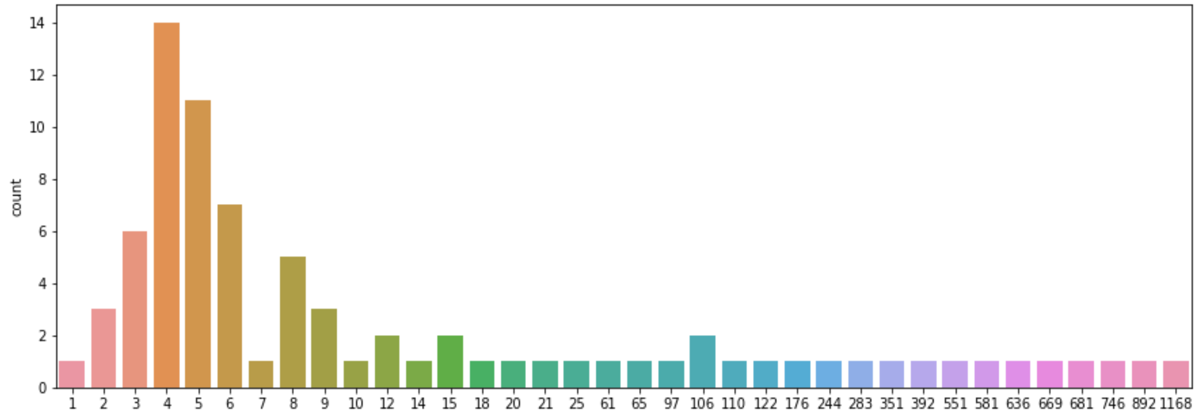

Fig. 1. Countplot of nunique values.

I formed a combined dataset where I joined train dataset and test dataset for further proceedings of imputation and encoding.

Then I divided all features into two categories- categorical and continuous features. For imputation of the features of both the categories, I used the SimpleImputer with stategy ‘mean’ for continuous features and ‘most_frequent’ for the categorical features and hence, imputation was done.

```
In [28]: from sklearn.impute import SimpleImputer
```

```
In [29]: si1=SimpleImputer(strategy='mean')
         si2=SimpleImputer(strategy='most_frequent')
```

```
In [30]: for i in df:
             if i =='saleprice':
                 pass
             else:
                 if i in cat_cols:
                     df[i]=si2.fit_transform(df[[i]])
                 else:
                     df[i]=si1.fit_transform(df[[i]])
         print(df.isna().sum().sum())
```

Screenshot. Simple Imputer.

I again deleted some more features based on the univariate analysis done on the features vs target which I have shown in the plot section of this report.

Then I encoded categorical features with LabelEncoder and hence, proceeded with that.

In [42]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [43]:
```
for i in cat_cols:
    df[i]=le.fit_transform(df[[i]])
```

Screenshot. Label Encoder.

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

There are two types of plots that I used in analyzing the relationship between the features and the target- one is Scatter plot between continuous features and target and second is Swarm plots between categorical features and target variable. There are 15 continuous features and 51 categorical features.

Scatter plots: These plots have been used to analyze the relationship between continuous features and the target variable.

I. The above plot shows maximum lotfrontage lies between 0 and 100.
II. Maximum lotarea lies between 0 and 20000.
III. Yearbuilt feature: it shows that the newly built is the house, more is the price.
IV. Yearremmodadd feature: most of the datapoints lie between 1990 and 2010.
V. Masvnrarea: a lot of datapoints having 0 value. And the next concentration is upto 200.
VI. Bsmtfinsf1: here, also most of the datapoints have values nearly zero.
VII. Bsmtunfsf: its values are scattered from 0 to 1500 mainly.
VIII. Totalbsmtsf: its maximum values lie in the range of 1000. More is its value, more is sale price.
IX. 1stflrsf: it’s scattered from 0 to 1000. More is its value, more is sale price.
X. 2ndflrsf: for a large range of datapoints, it has 0 value. For the rest, it varies from 500 to 1500.
XI. Grlivarea: more is the area, more is the sale price.
XII. Garageyrblt: the latest is the garage built, more is the sale price.
XIII. Garagearea: more is the area, more is the sale price.
XIV. Wooddecksf: it has 0 value for many datapoints.
XV. Openporchsf: it also has 0 value for many datapoints. For the rest, more is its value, more is the sale price.

Swarmplots: These plots have been plotted between categorical features and the target variable. As there are 51 categorical features, so I plotted there 3 plots- first two plots between 20 categorical features and the target variable and the third one is between 11 categorical features and target variable.

First swarmplot:

Based on this plot, I analyzed that the 3 features 'roofmatl','street' and 'condition2’ was having one particular value for most of the datapoints as shown in subplot 3,10 and 12 of the following figure and hence, I checked the value.counts() for these 3 features which showed that they were holding one particular value for almost 90% data, which made them irreluctant for my model and hence, I dropped these features as shown in screenshot.

saleprice vs first 20 categorical features

Fig. First Swarmplot.

```
In [34]: l1=['roofmatl','street','condition2']
         for i in l1:
             print(i,'\n',df1[i].value_counts(),'\n')

roofmatl
 CompShg    1144
Tar&Grv      10
WdShngl       6
WdShake       4
Roll          1
ClyTile       1
Metal         1
Membran       1
Name: roofmatl, dtype: int64

street
 Pave    1164
Grvl       4
Name: street, dtype: int64

condition2
 Norm     1154
Feedr       6
PosN        2
Artery      2
RRAe        1
RRNn        1
PosA        1
RRAn        1
Name: condition2, dtype: int64
```

```
In [35]: for i in l1:
             df=df.drop(i,axis=1)
             cat_cols.remove(i)
             del_cols.append(i)
```

Screenshot First Swarmplot decisions.

Second Swarmplot:

Then I plotted the second swarmplot between next 20 categorical features and the target variable and analyzed that there was one feature ‘heating’ which was having one particular value for almost 95% values as depicted from subplot 8 of this plot. So, dropped that particular feature.

Fig. Second Swarmplot

```
In [37]: df1['heating'].value_counts()

Out[37]: GasA     1143
         GasW       14
         Grav        5
         Wall        4
         Floor       1
         OthW        1
         Name: heating, dtype: int64
```

```
In [38]: df=df.drop('heating',axis=1)
         cat_cols.remove('heating')
         del_cols.append('heating')
```

Screenshot. Second Swarmplot decisions.

Third Swarmplot:

This last swarmplot showed that two features 'garagequal' and 'garagecond' were having one particular value for almost 90% of the dataset as shown in subplot 5 & 6 of this plot. So, after checking their count_values, I dropped these featurtes.

Fig. Third Swarmplot.

```
In [40]: l2=['garagequal','garagecond']
         for i in l2:
             print(i,'\n',df1[i].value_counts(),'\n')
```

```
garagequal
 TA    1050
Fa      39
Gd      11
Ex       2
Po       2
Name: garagequal, dtype: int64

garagecond
 TA    1061
Fa      28
Gd       8
Po       6
Ex       1
Name: garagecond, dtype: int64
```

```
In [41]: for i in l2:
             df=df.drop(i,axis=1)
             cat_cols.remove(i)
             del_cols.append(i)
```

Screenshot. Swarmplot 3 decisions.

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

  There were basic assumptions which I assumed in my model. These are:
  - Zscore of 3 is chosen for removing outliers from my train dataset.
  - Dataloss criteria in case of outliers removal has chosen to be less than 10%.
  - For skewness removal. I chose the -0.5 to +0.5 as the accepted range of skewness for my model. So, I chose that particular transformation technique after applying of which I got this range of skewness for my train dataset.
  - For removing features based on the correlation between the independent features, I chose -0.8 to +0.8 as the accepted range. If some feature had higher correlation with some other feature outside of this range, then the feature which had lower correlation with the target variable was dropped.
  - For removing multicollinearity, the accepted range of vif I chose to be <5.
  - For selecting the best features based on SelectKBest method, I chose 27 features from the total 54 features fed to it.

- **Hardware and Software Requirements and Tools Used**

  Hardware:

  Processor:

  - core i5 or above
  - RAM: 8 GB or above

▪ ROM/SSD: 250 GB or above

Software:

Anaconda 3- language used Python 3 and worked on Jupyter Notebook.

Libraries Imported:

- Numpy
- Pandas
- Matplotlib
- Seaborn

Model/s Development and Evaluation

Identification of possible problem-solving approaches

▪ **Outliers Removal**: I removed the outlier from train dataset using zscore method. Firstly, I analyzed outliers using boxplots and then found that there were 97 datapoints which I needed to delete in order to remove outliers from my train dataset. As the data loss was less than 10% (it was almost 8%), so, I deleted those datapoints and hence, left with 1071 records from 1168 records that was available earlier.

The boxplot and hence, screenshot showing the outliers removal technique has been attached:

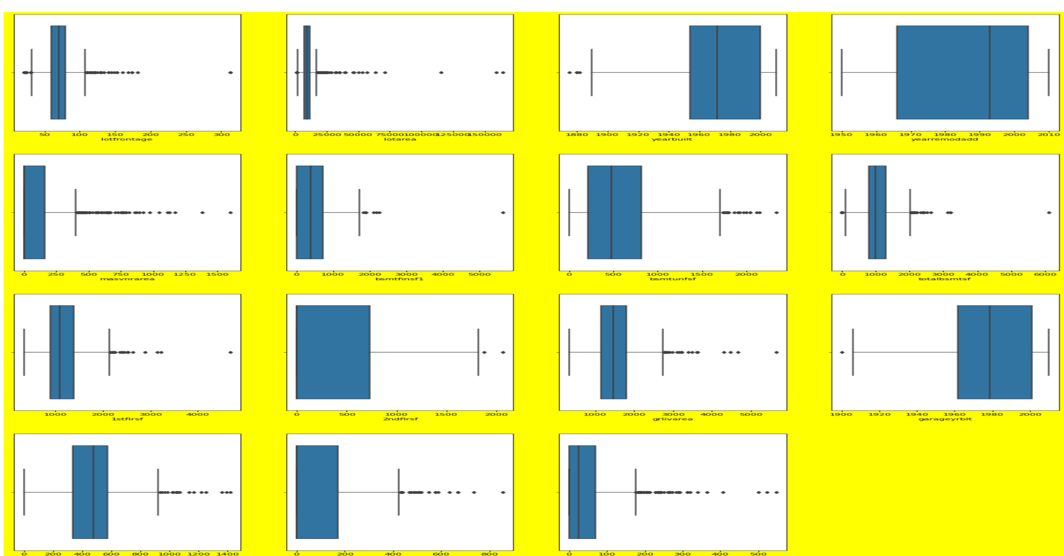

Fig. Boxplot showing outliers.

```
In [53]: ((np.abs(zscore(df_trcont)))>3).any()
```

```
Out[53]: lotfrontage      True
         lotarea          True
         yearbuilt        True
         yearremodadd    False
         masvnrarea       True
         bsmtfinsf1       True
         bsmtunfsf        True
         totalbsmtsf      True
         1stflrsf         True
         2ndflrsf         True
         grlivarea        True
         garageyrblt      True
         garagearea       True
         wooddecksf       True
         openporchsf      True
         dtype: bool
```

```
In [54]: ind1=np.where((np.abs(zscore(df_trcont)))>3)
         ind1
```

```
Out[54]: (array([  23,   40,   51,   68,  103,  103,  113,  119,  119,  140,  141,
                  141,  141,  141,  141,  141,  142,  142,  151,  152,  191,  192,
                  192,  192,  195,  232,  232,  232,  241,  241,  241,  243,  245,
                  245,  273,  299,  303,  305,  305,  305,  305,  309,  310,  325,
                  338,  352,  355,  356,  361,  361,  361,  361,  361,  361,  361,
                  381,  394,  403,  434,  449,  452,  490,  500,  504,  504,  504,
                  523,  525,  561,  561,  574,  581,  592,  592,  592,  592,  592,
                  592,  592,  592,  592,  600,  600,  608,  614,  626,  626,  639,
                  655,  681,  683,  689,  691,  691,  691,  691,  691,  695,  697,
                  697,  707,  711,  713,  720,  736,  746,  757,  757,  758,  762,
                  762,  762,  762,  772,  772,  800,  821,  821,  830,  833,  839,
                  839,  839,  858,  861,  863,  864,  870,  897,  897,  914,  914,
                  914,  956,  980, 1017, 1017, 1038, 1046, 1047, 1053, 1073, 1082,
                 1094, 1104, 1120, 1120, 1121, 1123, 1123, 1134, 1142, 1150],
                dtype=int64),
          array([14,  0, 14,  8, 10, 14,  1,  1, 13,  4,  0,  5,  7,  8,  9, 10,  4,
                 13, 14, 13,  0,  2,  9, 10,  0,  4,  6, 10,  4,  5,  8,  7,  1, 13,
                 10,  2,  0,  6,  7,  8, 10, 13, 13, 13,  4,  4,  4,  1,  1,  4,  5,
                  7,  8, 10, 14, 13, 14, 11,  4, 14, 10, 13, 13,  4,  7,  8,  4,  4,
                 12, 14,  6, 14,  0,  1,  4,  5,  7,  8, 10, 12, 14,  1, 13,  4, 10,
                  0, 14, 14,  2,  4, 14,  1,  4,  7,  8,  9, 10, 13,  4,  5, 13, 14,
                 14,  4,  4, 14,  4,  6,  0,  4,  6,  7,  8,  6, 12,  0,  7,  8,  2,
                  1,  4,  9, 10,  4,  4,  6,  0,  4,  4, 12,  4,  5, 12, 14, 12, 13,
```

Screenshot Outlier Removal (1)

```
In [55]: ind1=list(set(ind1[0]))
         len(ind1)
```

```
Out[55]: 97
```

```
In [56]: df_tr1=df_tr.drop(df_tr.index[ind1])
         df_tr1
```

Out[56]:

	mssubclass	mszoning	lotfrontage	lotarea	lotshape	landcontour	lotconfig	landslope	neighborhood	condition1	bldgtype	housestyle	overallqual	ov
0	11	3	70.049958	4928.0	0	3	4	0	13	2	4	2	5	
1	0	3	95.000000	15865.0	0	3	4	1	12	2	0	2	7	
2	5	3	92.000000	9920.0	0	3	1	0	15	2	0	5	6	
3	0	3	105.000000	11751.0	0	3	4	0	14	2	0	2	5	
4	0	3	70.049958	16635.0	0	3	2	0	14	2	0	2	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	0	3	70.049958	9819.0	0	3	4	0	19	2	0	2	4	
1164	0	3	67.000000	8777.0	3	3	4	0	7	1	0	2	3	
1165	12	3	24.000000	2280.0	3	3	2	0	13	2	3	5	5	
1166	6	0	50.000000	8500.0	3	3	4	0	9	1	0	5	3	
1167	5	3	70.049958	7861.0	0	3	4	0	8	2	0	5	5	

1071 rows × 61 columns

```
In [57]: df_tr1.reset_index(drop=True,inplace=True)
```

Screenshot Outlier Removal (2)

- **Skewness Removal:**

  Then I plotted distribution plot for checking skewness in the train dataset and found that there was a lot of skewness associated with some features. Hence, after quantifying them, I used PowerTransformer for removing the skewness and got the skewness in the range -0.5 to +0.5.
  The distplots and screenshot of applying PowerTransformer are attached below:

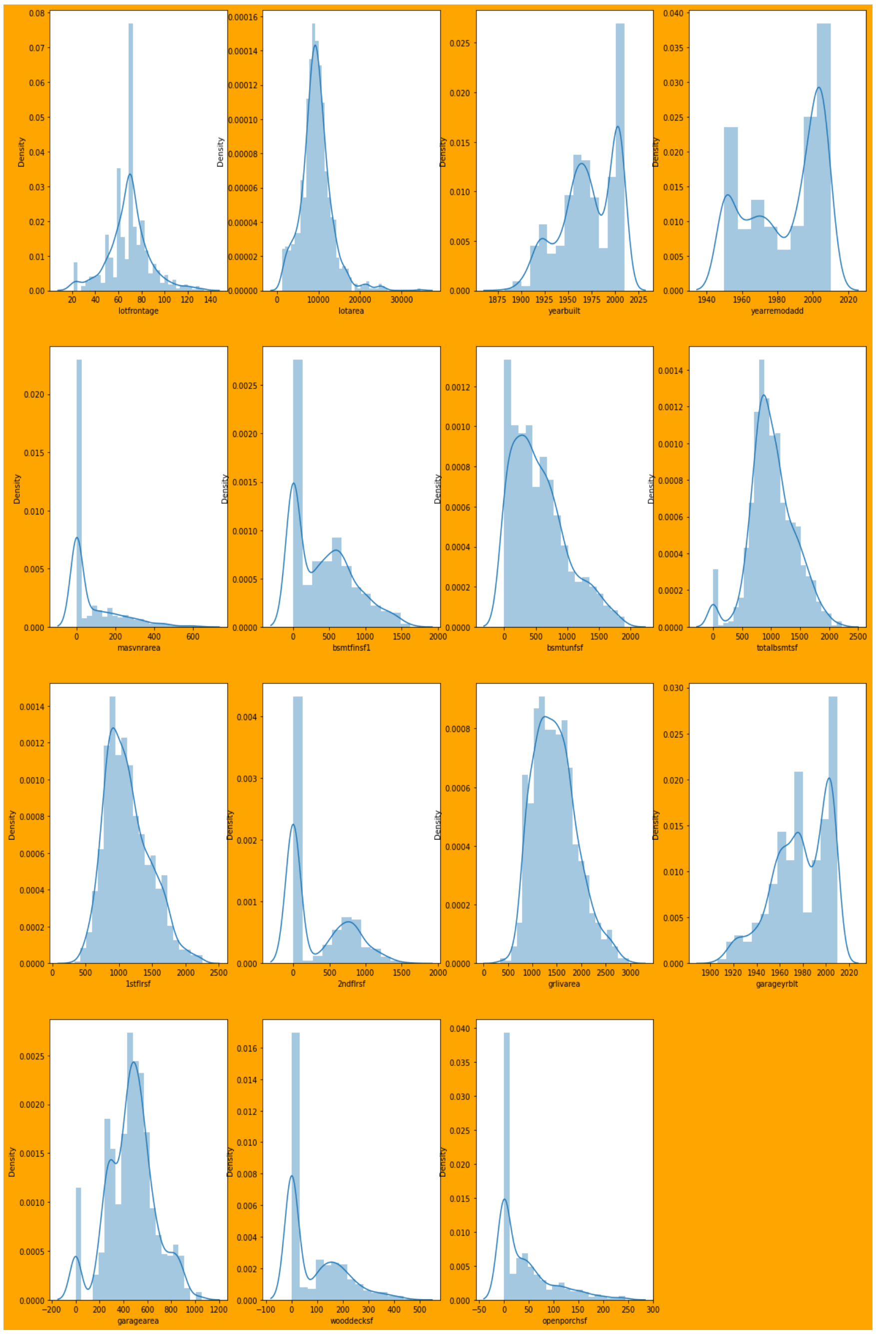

Fig. Distplot.

```
In [63]: from sklearn.preprocessing import PowerTransformer
         pt=PowerTransformer()

In [64]: for i in cont_cols:
             df_tr1[i]=pt.fit_transform(df_tr1[[i]])
             df_ts[i]=pt.fit_transform(df_ts[[i]])

In [65]: for i in cont_cols:
             print(i,'\t',df_tr1[i].skew())

         lotfrontage     0.10966436396848639
         lotarea         0.1180440722342076
         yearbuilt       -0.1118301277715297
         yearremodadd    -0.20778716194569236
         masvnrarea      0.4477641424251305
         bsmtfinsf1      -0.41206105327534553
         bsmtunfsf       -0.31501661307605106
         totalbsmtsf     -0.2077574538602659
         1stflrsf        -0.0028961020145377263
         2ndflrsf        0.32902766905791253
         grlivarea       -0.005844975469505966
         garageyrblt     -0.1240210034601354
         garagearea      -0.42972218035122983
         wooddecksf      0.13511077272552297
         openporchsf     0.037423816457054344

In [66]: # hence, skewness removed.
```

Screenshot of applying Power Transformer.

- **Features deletion based on correlation between the features:**

  I plotted heatmap of correlation matrix and deleted some of the features based on the correlation value between them. I am pasting here the heatmap of correlation matrix before features deletion and after features deletion. I deleted garagearea, exterior1st, garageyrblt and totrmsabvgrd features based on this correlation matrix and hence, I got the heatmap with the correlation values between the features in the range -0.8 to +0.8.

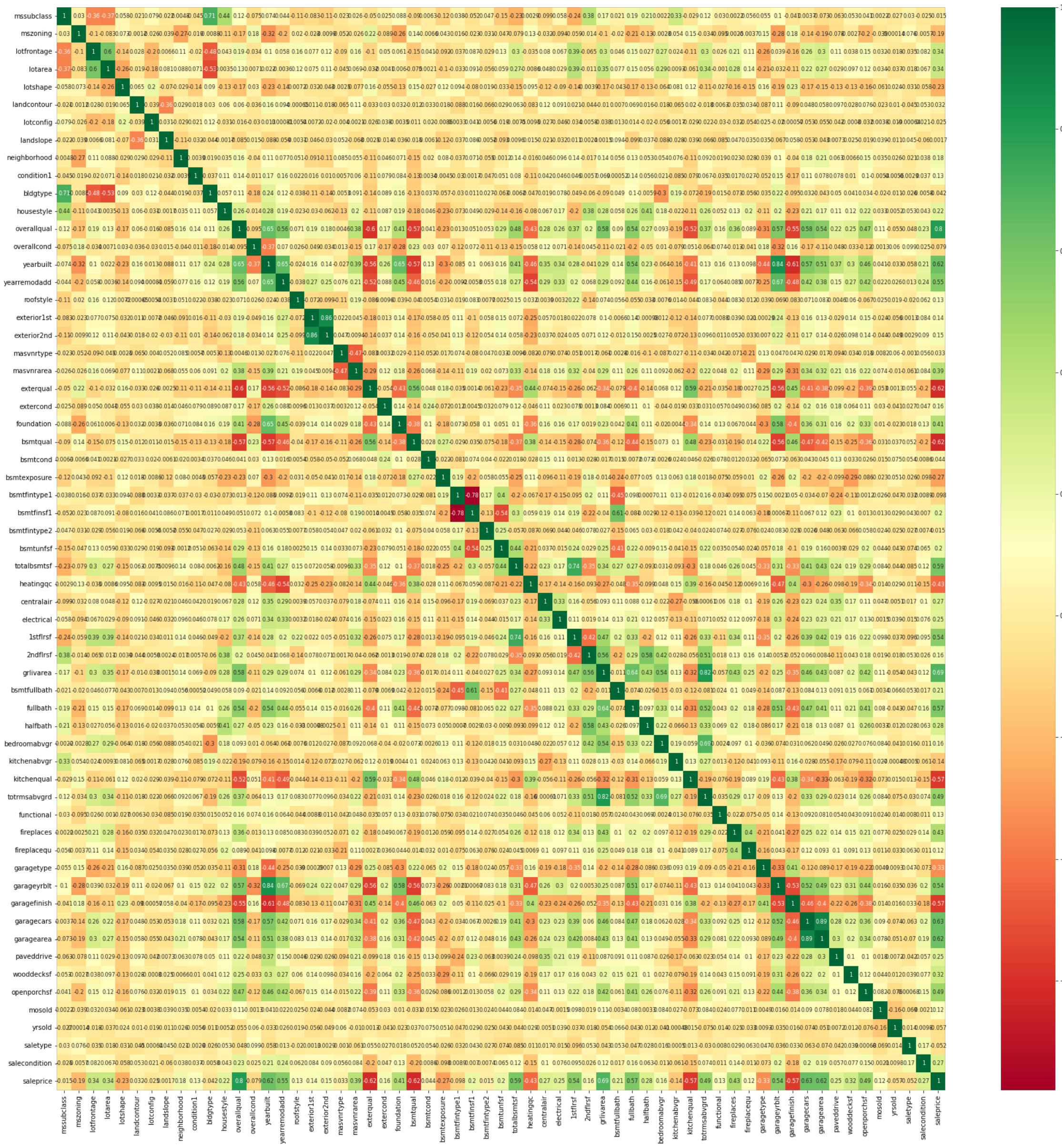

Fig. Correlation Heatmap before feature deletion.

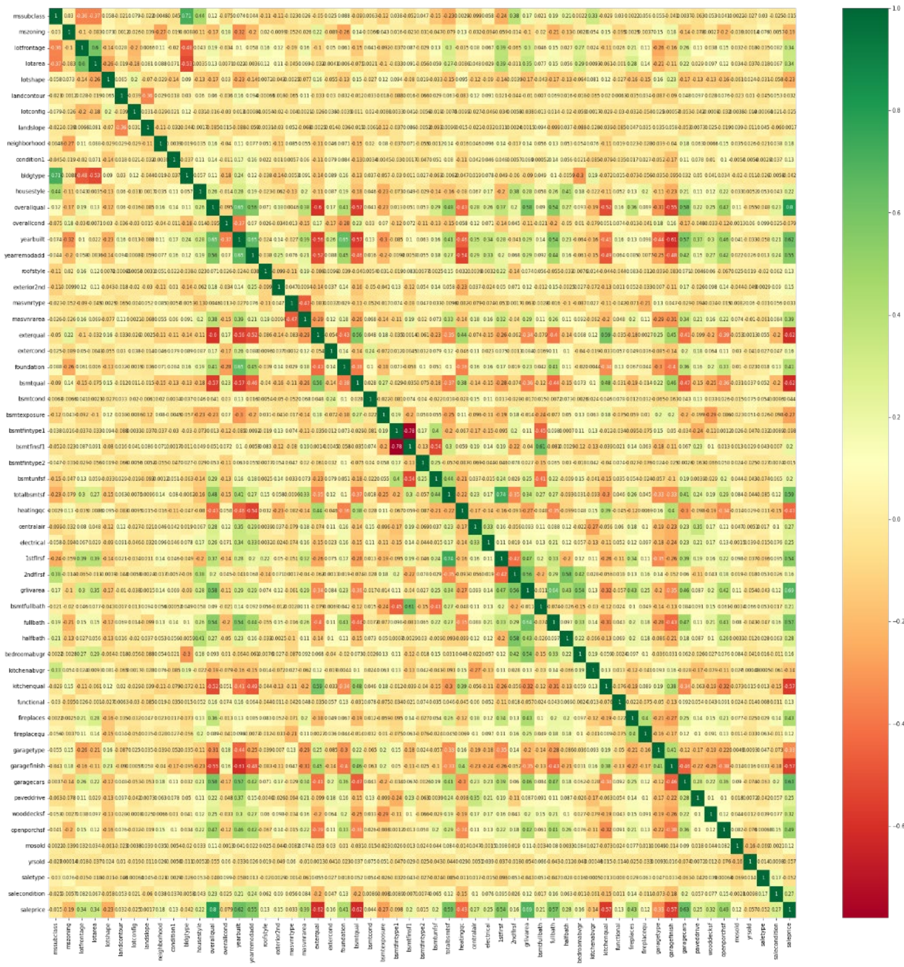

Fig. Correlation Heatmap after feature Deletion.

- **Mullticollinearity Removal:**

  I removed multicollinearity based on vif values. The features grlivarea and totalbsmtsf were deleted based on these vif values and at the end, I got vif values in the range of <5.

```
In [80]: vif_check(df_tr1)

          vif      features
10  13.863697     grlivarea
9   12.535964      2ndflrsf
8   11.346200      1stflrsf
7    5.260209   totalbsmtsf
6    3.626535     bsmtunfsf
5    3.090623    bsmtfinsf1
2    2.375288     yearbuilt
3    1.873615  yearremodadd
1    1.751938       lotarea
0    1.668585   lotfrontage
12   1.489043   openporchsf
4    1.326629    masvnrarea
11   1.176145    wooddecksf
```

Screenshot. vif values before feature deletion.

```
In [84]: vif_check(df_tr1)

         vif       features
2   2.219249      yearbuilt
7   2.088079       1stflrsf
3   1.870047   yearremodadd
6   1.812809      bsmtunfsf
5   1.766543     bsmtfinsf1
1   1.737991        lotarea
0   1.656039    lotfrontage
8   1.487432       2ndflrsf
10  1.470558    openporchsf
4   1.312646     masvnrarea
9   1.157389     wooddecksf
```

Screenshot vif values after feature deletion.

- **Scaling:**

  I used StandardScaler for applying scaling on the continuous features of my dataset.

- **Feature Selection based on SelectKBest method:**

  I used SelectKBest method with f_regression score_func to find the best features and selected first 27 features out of 54 features and developed my model with that. I also plotted a graph showing scores of each and every feature of the dataset found from SelectKBest method.

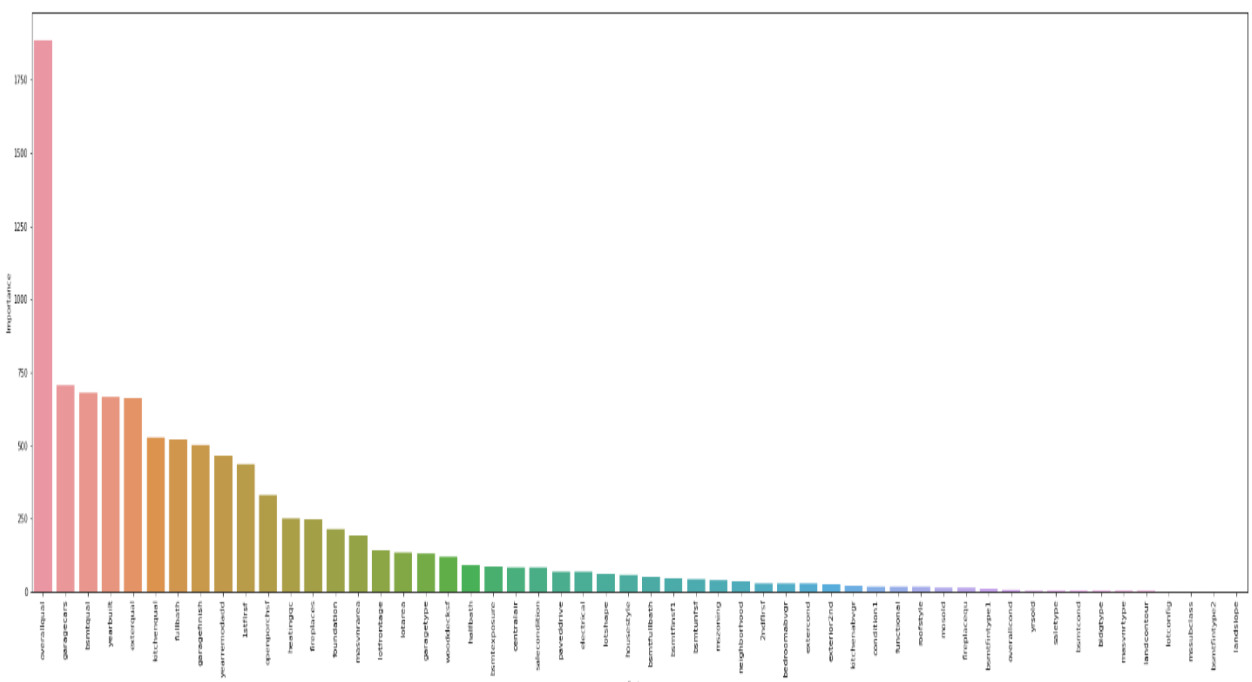

Fig. Features' scores based on SelectKBest.

- ## Testing of Identified Approaches (Algorithms)

I used following algorithms for my model:

- LinearRegression.
- DecisionTreeRegressor.
- KNeighborsRegressor.
- AdaBoostRegressor.
- RandomForestRegressor.
- XGBoostRegressor
- Regularization Techniques/ algos: Lasso and Ridge.
- SVR

- ## Run and Evaluate selected models

I used the below mentioned function to find out the result for each algorithm I tried.

```
In [94]: def algo_check (x,y,algo):
    min_diff=1
    max_i=0
    for i in range(100):
        x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state = i)
        algo.fit(x_train,y_train)
        y_pred1 =algo.predict(x_train)
        acc1= r2_score(y_train,y_pred1)
        y_pred2 =algo.predict(x_test)
        acc2= r2_score(y_test,y_pred2)
        acc=acc1-acc2
        if acc< min_diff:
            min_diff=acc
            max_i = i
            i+=1
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state = max_i)
    algo.fit(x_train,y_train)
    y_pred1 =algo.predict(x_train)
    acc1= r2_score(y_train,y_pred1)
    y_pred2 =algo.predict(x_test)
    acc2= r2_score(y_test,y_pred2)
    cvs=cross_val_score(algo,x_train,y_train,cv=5,scoring='r2')
    ac=cvs.mean()
    mae=mean_absolute_error(y_pred2,y_test)
    mse=mean_squared_error(y_pred2,y_test)
    rmse=np.sqrt(mse)
    print(f'''for algo {algo}, \nthe training accuracy is {acc1}, \ntesing accuracy is {acc2} \nat random_state {max_i}
and hence, mean square error is {mse} \nand mean_absolute_error is {mae} \nand hence, rmse is {rmse},
also cross_validation_score is {ac}''')
```

Screenshot of function used for getting result for each algo.

Output of linear Regression:

```
In [96]: algo_check(x_skb,y_tr,lr)

for algo LinearRegression(),
the training accuracy is 0.8221825890900674,
tesing accuracy is 0.8619189235509143
at random_state 18
and hence, mean square error is 626092840.2499946
and mean_absolute_error is 19656.686453870203
and hence, rmse is 25021.847258945425,
also cross_validation_score is 0.8076500849618661
```

Output of DecisionTreeRegressor:

```
In [98]: algo_check(x_skb,y_tr,dtr)

for algo DecisionTreeRegressor(),
the training accuracy is 1.0,
tesing accuracy is 0.6323387495131446
at random_state 72
and hence, mean square error is 1679715259.9402986
and mean_absolute_error is 26300.014925373136
and hence, rmse is 40984.32944358488,
also cross_validation_score is 0.567057415010826
```

Output for KNeighborsRegressor:

```
In [97]: algo_check(x_skb,y_tr,knr)

for algo KNeighborsRegressor(),
the training accuracy is 0.822044021795754,
tesing accuracy is 0.8213598074010174
at random_state 66
and hence, mean square error is 839850039.3322387
and mean_absolute_error is 20060.571641791044
and hence, rmse is 28980.166309602824,
also cross_validation_score is 0.7253037630886694
```

Output for AdaBoostRegressor:

```
In [99]: algo_check(x_skb,y_tr,abr)

for algo AdaBoostRegressor(),
the training accuracy is 0.8371929886738481,
tesing accuracy is 0.8149190969103381
at random_state 19
and hence, mean square error is 856943505.7443556
and mean_absolute_error is 22710.151281811428
and hence, rmse is 29273.597417200974,
also cross_validation_score is 0.7499280570847586
```

Output for RandomForestRegressor:

```
In [100]: algo_check(x_skb,y_tr,rfr)

for algo RandomForestRegressor(),
the training accuracy is 0.9753314880630268,
tesing accuracy is 0.8800664474326366
at random_state 18
and hence, mean square error is 543807598.399337
and mean_absolute_error is 17872.35895771144
and hence, rmse is 23319.682639335748,
also cross_validation_score is 0.7999813078226655
```

Output for SVR:

```
In [101]: algo_check(x_skb,y_tr,svr)

for algo SVR(),
the training accuracy is -0.04949296190498176,
tesing accuracy is -0.006062726074020697
at random_state 68
and hence, mean square error is 3266431049.176049
and mean_absolute_error is 44740.86015644706
and hence, rmse is 57152.69940410557,
also cross_validation_score is -0.04720669182574642
```

Output for XGBRegressor:

```
In [102]: algo_check(x_skb,y_tr,xgb)

          for algo XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
                       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                       early_stopping_rounds=None, enable_categorical=False,
                       eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
                       importance_type=None, interaction_constraints='',
                       learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
                       max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
                       missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
                       num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
                       reg_lambda=1, ...),
          the training accuracy is 0.9997583364153075,
          tesing accuracy is 0.8777879012968108
          at random_state 18
          and hence, mean square error is 554139075.0832244
          and mean_absolute_error is 18226.82372318097
          and hence, rmse is 23540.158773534735,
          also cross_validation_score is 0.7914314190023903
```

Output for Regularization Lasso:

```
In [103]: lasscv = LassoCV(alphas=[0.001,0.005,0.01,0.08,0.1,0.5,1,5,10],max_iter = 100, normalize =True)
          algo_check(x_skb,y_tr,lasscv)
          alp=lasscv.alpha_

          for algo LassoCV(alphas=[0.001, 0.005, 0.01, 0.08, 0.1, 0.5, 1, 5, 10], max_iter=100,
                  normalize=True),
          the training accuracy is 0.8215338528126973,
          tesing accuracy is 0.863216298424227
          at random_state 18
          and hence, mean square error is 620210230.2632389
          and mean_absolute_error is 19468.14097647041
          and hence, rmse is 24904.020363452142,
          also cross_validation_score is 0.8082733415947395

In [104]: lasso_reg =Lasso(alp)
          algo_check(x_skb,y_tr,lasso_reg)

          for algo Lasso(alpha=10.0),
          the training accuracy is 0.8221807683690006,
          tesing accuracy is 0.8620066369921925
          at random_state 18
          and hence, mean square error is 625695126.3923812
          and mean_absolute_error is 19642.643891483727
          and hence, rmse is 25013.898664390188,
          also cross_validation_score is 0.8076960192724766
```

Output for Regularization Ridge:

In [107]:
```
ridgecv = RidgeCV(alphas=np.arange(0.001,0.1,0.01), normalize =True)
algo_check(x_skb,y_tr,ridgecv)
alp2=ridgecv.alpha_
```

```
for algo RidgeCV(alphas=array([0.001, 0.011, 0.021, 0.031, 0.041, 0.051, 0.061, 0.071, 0.081,
       0.091]),
        normalize=True),
the training accuracy is 0.8217399774344464,
tesing accuracy is 0.8625871975574291
at random_state 18
and hence, mean square error is 623062725.0338925
and mean_absolute_error is 19477.116734552157
and hence, rmse is 24961.224429780934,
also cross_validation_score is 0.8086549805823985
```

In [108]:
```
ridge_reg =Ridge(alp2)
algo_check(x_skb,y_tr,ridge_reg)
```

```
for algo Ridge(alpha=0.040999999999999995),
the training accuracy is 0.8221825852544293,
tesing accuracy is 0.8619230079456011
at random_state 18
and hence, mean square error is 626074320.6212668
and mean_absolute_error is 19655.962506313863
and hence, rmse is 25021.477187034077,
also cross_validation_score is 0.8076543556884802
```

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

Based on the above results and taken r2_score metrics for evaluation, I concluded that LinearRegression algorithm gave the best results and also best cross_val_score. So, I finalized LinearRegression algorithm for my model and developed my model with this algorithm.
In order to improve it further, I used GridSearchCV for hypertuning its parameters and got the generalized results as shown in below attached screenshot.

```
In [113]: # Let's do hypertuning on this.
```

```
In [114]: params={'fit_intercept':[True,False],'normalize':[True,False],'copy_X':[True,False],'positive':[True,False]}
          grid=GridSearchCV(lr,param_grid=params,cv=9)
          grid.fit(x_skb,y_tr)
          grid.best_params_

Out[114]: {'copy_X': True, 'fit_intercept': True, 'normalize': False, 'positive': False}
```

```
In [115]: lr1=LinearRegression(copy_X=True,fit_intercept=True,normalize=False,positive=False)
          lr1.fit(x_train,y_train)
          y_pred1 =lr1.predict(x_train)
          acc1= r2_score(y_train,y_pred1)
          y_pred2 =lr1.predict(x_test)
          acc2= r2_score(y_test,y_pred2)
          cvs=cross_val_score(lr1,x_train,y_train,cv=5,scoring='r2')
          ac=cvs.mean()
          mae=mean_absolute_error(y_pred2,y_test)
          mse=mean_squared_error(y_pred2,y_test)
          rmse=np.sqrt(mse)
          print(f'''for algo Linear Regression with hypertuned parameters, \nthe training accuracy is {acc1}, \ntesing accuracy is {acc2}
          and hence, mean square error is {mse} \nand mean_absolute_error is {mae} \nand hence, rmse is {rmse},
          also cross_validation_score is {ac}''')

          for algo Linear Regression with hypertuned parameters,
          the training accuracy is 0.8221825890900674,
          tesing accuracy is 0.8619189235509143
          and hence, mean square error is 626092840.2499946
          and mean_absolute_error is 19656.686453870203
          and hence, rmse is 25021.847258945425,
          also cross_validation_score is 0.8076500849618661
```

Screenshot of hypertuning effect on LinearRegression.

- ## Visualization

All the plots which I have plotted, I have explained them in their respective area in this report. Here, I am going to explore the inferences from the plots of the predicted values got from the algorithm chosen LinearRegression.

Plot 1. Actual vs Predicted:

This plot was drawn between actual values fed to LinearRegression algorithm of the testing pat=rt of the train dataset, y_test and its predicted values y_pred2. Y_test values are here shown by + sign in orange color and the predicted ones are shown by dot in blue color.

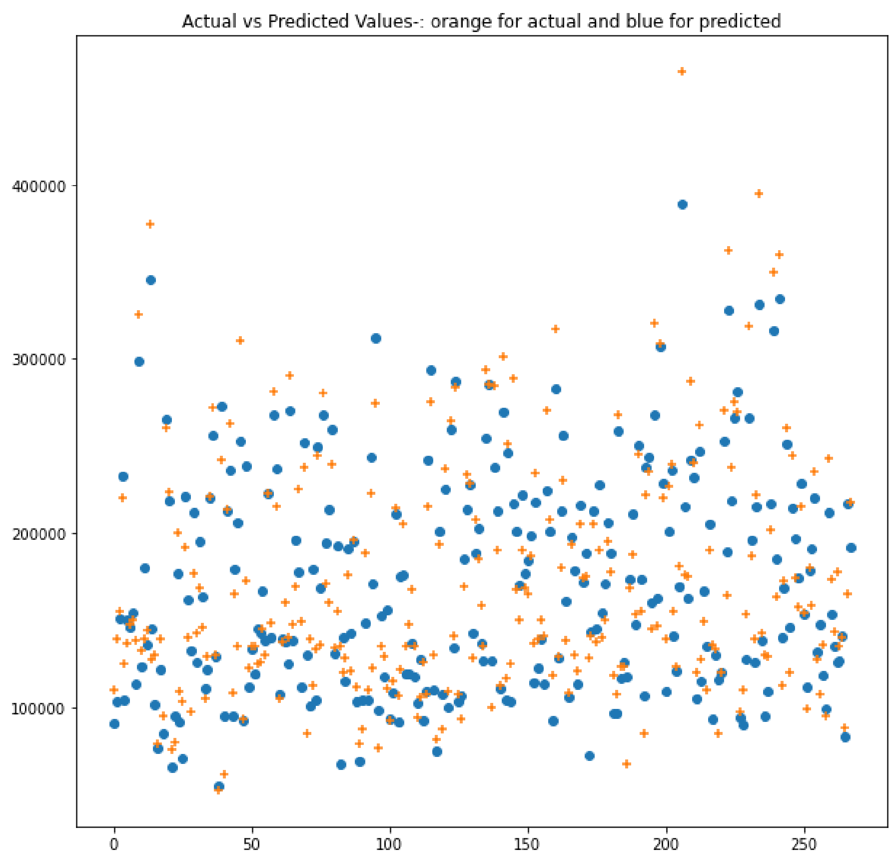

Fig. plot actual vs predicted values.

Plot 2. Actual-Predicted values plot:

This plot was drawn on the behavior/ frequency of difference between actual and predicted values. It shows that the maximum difference between actual and predicted values lie in the range of -2000 to +2000.

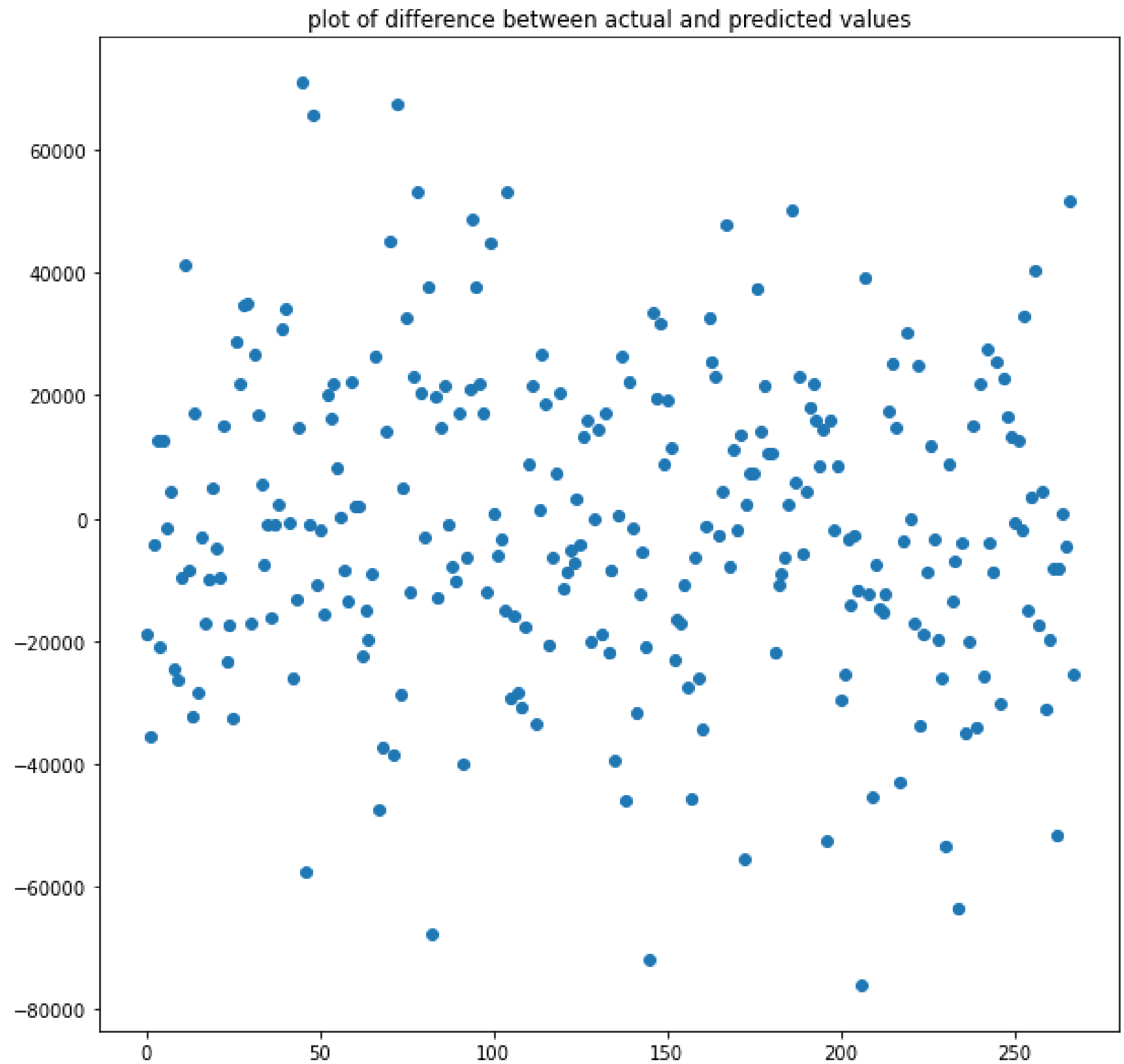

Fig. Plot of difference between actual and predicted values.

Plot 3. Regression plot between actual and predicted values.

This plot shows that the maximum predicted values lie near the best fit line.

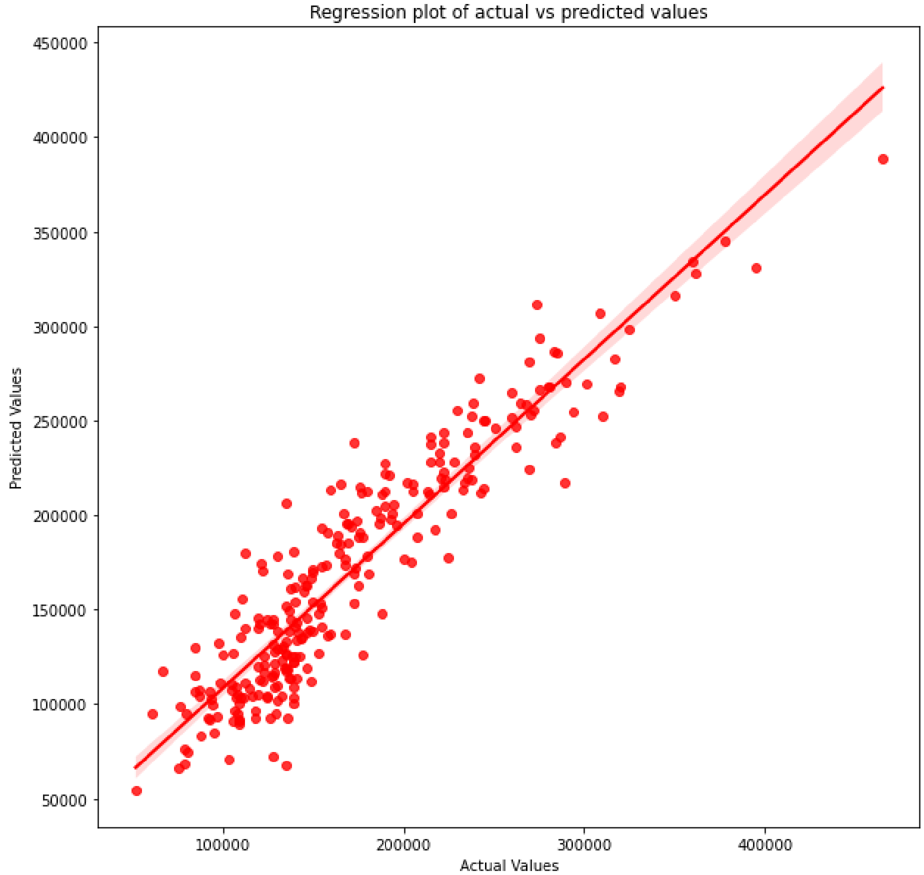

Fig. Regression plot b/w actual and predicted values.

- ## Interpretation of the Results

Based on the above results, it was interpreted that the predicted values lie near the best fit line which shows that the above model is acceptable.
Also, the above plot shows that the maximum deviation of the predicted one from the actual values lie in the range of -2000 to +2000 which is acceptable as the minimum value of actual y is 52,000 and its maximum value is 465000.
Also, as the mean_absolute_error comes out to be 19,656, so, (mae/(y_test.max()-y_test.min()))*100 comes out to be 4.75% which is acceptable range.

CONCLUSION

- **Key Findings and Conclusions of the Study**

  ✓ I found that LinearRegression algo worked well with my model. So, I finalized my model with this algo and hence, found the SalePrice for my test dataset which comes out to be.

```
In [132]: y_reqd

Out[132]: array([302314.62057734, 203916.9984838 , 271909.62645952, 181708.45790143,
                 224272.7151716 ,  60263.73653028, 127493.19495709, 280405.04353211,
                 237966.32393806, 175077.38960918,  78643.55070339, 140804.62559381,
                 122951.65051149, 192464.96326044, 276574.31256464, 113104.90509172,
                  92717.66056655, 110017.58658579, 186519.94420372, 194024.19580864,
                 151940.52204016, 173094.01647424, 138144.91924259,  81421.09837039,
                 113867.71422401,  98068.81219603, 176536.13846592, 132658.49926278,
                 163220.2048829 ,  82087.36103166, 149370.22460108, 201848.1745273 ,
                 230793.67874138, 178655.08859102, 132783.59716371, 194683.74808867,
                 197530.8282485 , 104303.29311848, 147536.83153551, 137233.29358834,
                 100300.43897933, 257613.71522401, 225402.39321417, 195644.37599827,
                 143677.63419871, 124586.74389014, 112030.46597751, 107192.49322173,
                 204503.27952546, 331438.29976869, 135328.77961501, 193603.47810576,
                  77341.2922627 ,  95430.84715572, 241158.99461674, 100650.22810132,
                 138578.75499082, 207351.92398731, 139845.27564314, 245634.14083787,
                  61294.51359222, 173817.08362806, 120757.64857737, 161356.87937661,
                 224280.1903228 ,  67461.75804603, 157259.59539913, 227670.85101853,
                 125110.26292959, 168967.45816627, 271059.66211413, 160828.6870546 ,
                 173565.4072332 , 141968.47774799, 148759.67776834, 224725.76954802,
                 281494.86180692, 216483.52019356, 258594.91272921, 145322.18407017,
                 250803.62852521, 128049.54204967, 157069.09654867, 143211.77173194,
                 199671.13700762, 237823.80960106, 119069.60702198, 286868.29037284,
                 143679.27783218, 181114.66901479, 229991.33731417, 149037.08814411,
                 126590.96928107, 145613.94702282, 187611.68948802, 169780.7529849 ,
                 260229.20176801, 204950.18966915, 289470.71083895, 111164.42557327,
                 251081.33656198, 110592.14769086, 132774.69733041, 167414.80563792,
                 199421.35996634, 127270.34044629, 233644.54256367, 155863.87534454,
                 186883.54216491, 202042.28503433, 189784.48180109, 155991.49584213,
                 197104.43821514, 249238.10804487, 135821.90533335,  85828.0491059 ,
                 120705.06778536, 180950.91481921, 130785.9584509 , 116595.12553899,
                  75852.90455136, 212635.02072944, 207058.22914818, 155699.17224269,
                 131686.31568663, 198963.27999309, 100680.4231338 , 165935.46730685,
                  99272.54313998,  80783.92133494, 139677.05909223, 240999.25096974,
                 123062.41947713, 155740.56779619, 167347.14826889, 281218.63783382,
                 219064.97753029, 122168.17932315, 256021.72273866, 115648.84419387,
                 134612.78572179, 350514.24522491,  90352.14806722, 360974.5688183 ,
                 162525.00816559, 210495.7952443 , 164644.61740595, 117881.84047683,
                 114137.06390599, 213746.10234063, 179609.64879113, 137073.64069992,
                 215787.89177194,  84193.44802891,  98400.09573792, 178036.40781584,
                 208249.91652726, 204109.94362003, 140919.67614974, 177707.09994297,
                 220013.68039324, 139285.77135794, 190987.27283646,  97499.45986218,
                 102270.4810626 , 263775.54933714, 183564.62850095, 215406.45393396,
                 103697.10217162, 214881.36549286, 164044.23557196, 112836.05270098,
                 141086.79131051, 276399.05139919, 116309.57039541, 345685.69542205,
```

Also, it was plotted as follows:

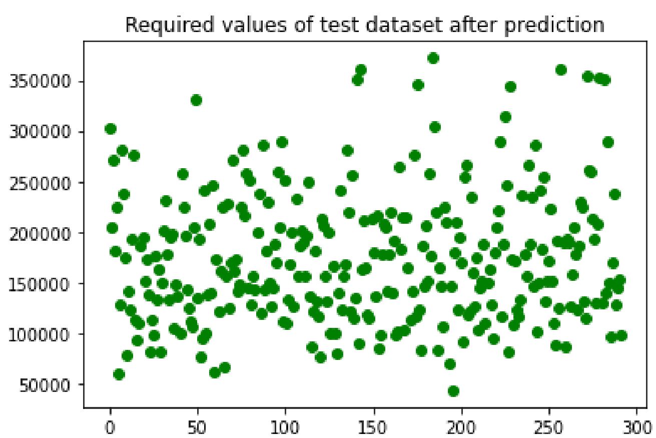

Fig. Predicted SalePrice values for Test dataset.

• Learning Outcomes of the Study in respect of Data Science

While working on this project I learned more things about the housing market and how the machine learning models have helped to predict the price of house which indeed helps the sellers and buyers to understand the future price of the house. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This graphical representation helped me to understand which features are important and how these features describe the sale price. Data cleaning was one of the important and crucial things in this project where I replaced all the null values with imputation methods and dealt with features having zero values and time variables.

Finally, our aim is achieved by predicting the house price for the test data, I hope this will be further helps for sellers and buyers to understand the house marketing. The machine learning models and data analytic techniques will have an important role to play in this type of problems. It helps the customers to know the future price of the houses.

• Limitations of this work and Scope for Future Work

- One limitation is the small amount of dataset. Its size should be more so that the accuracy may be increased.
- Secondly, it has included data from one particular area. More areas should be included so that the model may work more effectively.
- next more features need to be added in this dataset so that a more precise model may be formed.