Housing Price Prediction

Submitted by:

Matta Akhil Reddy

ACKNOWLEDGMENT

I would like to thank **FlipRobo Technologies** for providing me with this opportunity for writing the blog, and also in providing the guidance and constant support in providing necessary information regarding the project, by providing their support in completing the project

I also want to thank my **SME Ms.Khushboo garg** for their instant response in solving the problems and providing the valuable information by addressing out our query in right time.

Housing Price Prediction

INTRODUCTION

Business Problem Framing:

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy.

It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.

Our problem is related to one such housing company. A US-based housing company named **Surprise Housing** has decided to enter the Australian market. The company uses

data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia.

The company is looking at prospective properties to buy houses to enter the market.

Business Goal:

You are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the

prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

We will have two types of data,

- Training Data - This data will contain the information related to the Year Sold and Sale Price of House.
- Test Data - It will contain all the information about a house. And, based on all the given information, Logistic Regression Algorithm will predict the selling price of a house.

Conceptual Background of the Domain Problem

Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies.

Hedonic Characteristics of Housing Price: A Hedonic approach is preferred for predicting the sale prices in the housing market because the market displays resilience, flexibility and spatial fixity.

Housing Attributes: Studying the structural, locational, and economic attributes of housing properties is crucial in understanding their mutually inclusive relationships with their pricing

Motivation for the Problem Undertaken

There is a steady rise in house demand with every passing year, and consequently the house prices are rising every year. The problem arises when there are numerous variables such as location and property demand that influence the pricing. Therefore, buyers, sellers,

developers and the real estate industry are keen to know the most important factors influencing the house price to help investors make sound decisions and help house builders set the optimal house price.

There are many benefits that home buyers, property investors, and house builders can reap from the house-price model. This model aims to serve as a repository of such information and gainful insights to home buyers, property investors and house builders, that will help them determine best house prices.

This model can be useful for potential buyers in deciding the characteristics of a house they want that best fits their budget and will be of tremendous benefit, especially to housing developers and researchers, to ascertain the most significant attributes to determine house prices and to acknowledge the best machine learning model to be used to conduct a study in this field.

Required Libraries

Imported basic libraries in the basic level

```
']: # Importing Libraries
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    warnings.filterwarnings('ignore')
```

Train Data & Test Data

Housing Price Prediction

```
In [138]: train_df = pd.read_csv('train.csv')
          train_df.head()
```

Out[138]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows × 81 columns

```
In [139]: test_df = pd.read_csv('test.csv')
          test_df.head()
```

Out[139]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeatur
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	Nal
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	Nal
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	Nal
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	Nal
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	Nal

5 rows × 80 columns

Finding the shape of Train and test Data

Statistical Information of Dataset

```
In [140]: train_df.shape
Out[140]: (1168, 81)
```

- we have 1168 rows and 81 columns in train data

```
In [141]: test_df.shape
Out[141]: (292, 80)
```

- we have 292 rows and 80 columns in test data

Information of the Dataframe

```
#   Column        Non-Null Count  Dtype
---  ------       --------------  -----
 0   Id           1168 non-null   int64
 1   MSSubClass   1168 non-null   int64
 2   MSZoning     1168 non-null   object
 3   LotFrontage  954 non-null    float64
 4   LotArea      1168 non-null   int64
 5   Street       1168 non-null   object
 6   Alley        77 non-null     object
 7   LotShape     1168 non-null   object
 8   LandContour  1168 non-null   object
 9   Utilities    1168 non-null   object
 10  LotConfig    1168 non-null   object
 11  LandSlope    1168 non-null   object
 12  Neighborhood 1168 non-null   object
 13  Condition1   1168 non-null   object
```

14	Condition2	1168 non-null	object
15	BldgType	1168 non-null	object
16	HouseStyle	1168 non-null	object
17	OverallQual	1168 non-null	int64
18	OverallCond	1168 non-null	int64
19	YearBuilt	1168 non-null	int64
20	YearRemodAdd	1168 non-null	int64
21	RoofStyle	1168 non-null	object
22	RoofMatl	1168 non-null	object
23	Exterior1st	1168 non-null	object
24	Exterior2nd	1168 non-null	object
25	MasVnrType	1161 non-null	object
26	MasVnrArea	1161 non-null	float64
27	ExterQual	1168 non-null	object
28	ExterCond	1168 non-null	object
29	Foundation	1168 non-null	object
30	BsmtQual	1138 non-null	object
31	BsmtCond	1138 non-null	object
32	BsmtExposure	1137 non-null	object
33	BsmtFinType1	1138 non-null	object
34	BsmtFinSF1	1168 non-null	int64
35	BsmtFinType2	1137 non-null	object
36	BsmtFinSF2	1168 non-null	int64
37	BsmtUnfSF	1168 non-null	int64
38	TotalBsmtSF	1168 non-null	int64
39	Heating	1168 non-null	object
40	HeatingQC	1168 non-null	object
41	CentralAir	1168 non-null	object
42	Electrical	1168 non-null	object
43	1stFlrSF	1168 non-null	int64
44	2ndFlrSF	1168 non-null	int64
45	LowQualFinSF	1168 non-null	int64
46	GrLivArea	1168 non-null	int64
47	BsmtFullBath	1168 non-null	int64
48	BsmtHalfBath	1168 non-null	int64
49	FullBath	1168 non-null	int64
50	HalfBath	1168 non-null	int64
51	BedroomAbvGr	1168 non-null	int64
52	KitchenAbvGr	1168 non-null	int64
53	KitchenQual	1168 non-null	object
54	TotRmsAbvGrd	1168 non-null	int64
55	Functional	1168 non-null	object
56	Fireplaces	1168 non-null	int64
57	FireplaceQu	617 non-null	object
58	GarageType	1104 non-null	object
59	GarageYrBlt	1104 non-null	float64
60	GarageFinish	1104 non-null	object
61	GarageCars	1168 non-null	int64
62	GarageArea	1168 non-null	int64
63	GarageQual	1104 non-null	object
64	GarageCond	1104 non-null	object
65	PavedDrive	1168 non-null	object
66	WoodDeckSF	1168 non-null	int64
67	OpenPorchSF	1168 non-null	int64
68	EnclosedPorch	1168 non-null	int64
69	3SsnPorch	1168 non-null	int64
70	ScreenPorch	1168 non-null	int64

71	PoolArea	1168 non-null	int64
72	PoolQC	7 non-null	object
73	Fence	237 non-null	object
74	MiscFeature	44 non-null	object
75	MiscVal	1168 non-null	int64
76	MoSold	1168 non-null	int64
77	YrSold	1168 non-null	int64
78	SaleType	1168 non-null	object
79	SaleCondition	1168 non-null	object
80	SalePrice	1168 non-null	int64

Features Description

- **Id** : Unique id of the housing prediction dataset
- **MSSubClass** : The Building class
- **MSZoning** : zone of the house(RL, RM, FV, RH, C(all))
- **LotFrontage** : Linear feet of street connected to property
- **LotArea** : Lot size in square feet
- **Street** : This talks about the type of road(Pave/gravel)
- **Alley** : Type of alley access (pave,grvl) note: we have nan values in this feature
- **LotShape** : general shape of the property('IR1', 'Reg', 'IR2', 'IR3')
- **LandContour** : Flatness of the property ['Lvl', 'Bnk', 'HLS', 'Low']
- **Utilities** : Type of utilities available['AllPub'], as we have only one variable we can drop
- **LotConfig** : Lot configuration ['Inside', 'CulDSac', 'FR2', 'Corner', 'FR3']
- **LandSlope** : Slope of property ['Gtl', 'Mod', 'Sev']
- **Neighborhood** : Physical locations within ames city limits ['NPkVill', 'NAmes', 'NoRidge', 'NWAmes', 'Gilbert', 'Sawyer','Edwards', 'IDOTRR', 'CollgCr', 'Mitchel', 'Crawfor', 'BrDale','StoneBr', 'BrkSide', 'NridgHt', 'OldTown', 'Somerst', 'Timber','SWISU', 'SawyerW', 'ClearCr', 'Veenker', 'Blmngtn', 'MeadowV','Blueste']
- **Condition1** : Proximity to main road or rail road ['Norm', 'Feedr', 'RRAn', 'PosA', 'RRAe', 'Artery', 'PosN', 'RRNe',
- 'RRNn']

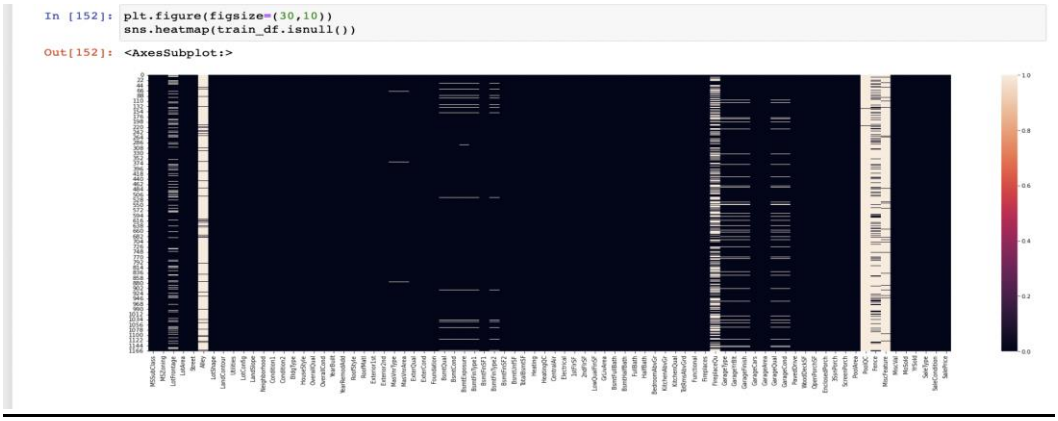
- **Condition2** : Proximity to main road or rail road if a second is present ['Norm', 'RRAe', 'Feedr', 'PosN', 'Artery', 'RRNn', 'PosA', 'RRAn']
- **BldgType** : Type of dwelling ['TwnhsE', '1Fam', 'Duplex', 'Twnhs', '2fmCon']
- **HouseStyle** : style of dwelling ['1Story', '2Story', '1.5Fin', 'SFoyer', '1.5Unf', 'SLvl', '2.5Fin','2.5Unf']
- **OverallQual** : Overall quality [6, 8, 7, 5, 9, 1, 2, 4, 3, 10]
- **OverallCond** : Overall Condition [5, 6, 7, 4, 8, 2, 3, 9, 1]
- **YearBuilt** : Construction Date (min : 1875, max :2010)
- **YearRemodAdd** : remodel date (max:2010,min 1950)
- **RoofStyle** : Style of the roof ['Gable', 'Flat', 'Hip', 'Shed', 'Gambrel', 'Mansard']
- **RoofMatl** : roof material ['CompShg', 'Tar&Grv', 'WdShngl', 'WdShake', 'Roll', 'ClyTile','Metal', 'Membran']
- **Exterior1st** : Exterior covering of house ['Plywood', 'Wd Sdng', 'MetalSd', 'CemntBd', 'VinylSd', 'HdBoard','Stucco', 'WdShing', 'BrkFace', 'Stone', 'AsbShng', 'AsphShn','ImStucc', 'BrkComm']
- **Exterior2nd** : Exterior covering of house if more than one material ['Plywood', 'Wd Sdng', 'MetalSd', 'CmentBd', 'VinylSd', 'HdBoard','Wd Shng', 'Stucco', 'ImStucc', 'Stone', 'BrkFace', 'AsbShng','Brk Cmn', 'AsphShn', 'Other']
- **MasVnrType** : Masonry veneer type ['None', 'BrkFace', 'Stone', 'BrkCmn', nan] we have nan and None type values here
- **MasVnrArea** : Masonry veneer area in square feet

- **ExterQual** : Exterior material quality ['TA', 'Gd', 'Ex', 'Fa']
- **ExterCond** : present condition of the material on the exterior ['TA', 'Gd', 'Fa', 'Po', 'Ex']
- **Foundation** : Type of foundation ['CBlock', 'PConc', 'BrkTil', 'Slab', 'Stone', 'Wood']
- **BsmtQual** : Height of the basement ['Gd', 'TA', 'Ex', nan, 'Fa'] we have NaN values
- **BsmtCond** : General condition of the basement ['TA', 'Gd', 'Fa', nan, 'Po'] we have NaN values
- **BsmtExposure** : walk out or garden level basement walls ['No', 'Gd', 'Av', 'Mn', nan] we have Nan values
- **BsmtFinType1** : quality of basement finished area ['ALQ', 'GLQ', 'BLQ', 'Unf', 'Rec', 'LwQ', nan]
- **BsmtFinSF1** : type 1 finished square feet.
- **BsmtFinType2** : quality of seconf finished area if present ['Unf', 'Rec', 'BLQ', 'GLQ', nan, 'ALQ', 'LwQ'] we have nan values
- **BsmtFinSF2** : Type 2 finished square feet
- **BsmtUnfSF** : Unfinished squarefeet of basement area
- **TotalBsmtSF** : Total sqaure feet of basement area
- **Heating** : type of heating ['GasA', 'GasW', 'Floor', 'OthW', 'Wall', 'Grav']
- **HeatingQC** : Heating quality and condition ['TA', 'Ex', 'Gd', 'Fa', 'Po']
- **CentralAir** : Central Air condition
- **Electrical** : electrical system ['SBrkr', 'FuseA', 'FuseF', 'FuseP', 'Mix']
- **1stFlrSF** : first floor square feet
- **2ndFlrSF** : second floor square feet
- **LowQualFinSF** : low quality finished square feet all floors
- **GrLivArea** : above grade living area square feet
- **BsmtFullBath** : basement full bathrooms
- **BsmtHalfBath** : basement half bathrooms
- **FullBath** : full bath rooms above grade
- **HalfBath** : Half baths above grade
- **BedroomAbvGr** : no of bedrroms above basement level
- **KitchenAbvGr** : no of kitchens
- **KitchenQual** : Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **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
- **Fence**: Fence quality
- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: Value of miscellaneous feature

- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

Data Preprocessing Done

Checking For the Null Values in the Dataset



we have Null values in

- LotFrontage
- Alley
- MasVnrType
- MasVnrArea
- BsmtQual
- BsmtCond
- BsmtExposure
- BsmtFinType1

Housing Price Prediction

- BsmtFinType2
- FireplaceQu
- GarageType
- GarageYrBlt
- GarageFinish
- GarageQual
- GarageCond
- PoolQC
- Fence
- MiscFeature

The ID columns from test and train datasets were also dropped

since they don't contribute to building a good model for predicting

the target variable values

Dropping all features which have heavy amount of NaN values in the dataset features as they dont contribute much in targeet variable

```
In [168]: train_df.drop(columns=['Alley','PoolQC','Fence','MiscFeature','FireplaceQu','MasVnrType'],axis=1,inplace=True)

In [169]: # Test Data also have same amount of Nan value percentages so dropping them
          test_df.drop(columns=['Alley','PoolQC','Fence','MiscFeature','FireplaceQu','MasVnrType'],axis=1,inplace=True)
```

Data Inputs- Logic- Output Relationships

```
In [143]: train_df.describe()
```

Out[143]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeck
count	1168.000000	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	...	1168.0000
mean	724.136130	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	...	96.2063
std	416.159877	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.664785	...	126.1589
min	1.000000	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	...	0.0000
25%	360.500000	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	...	0.0000
50%	714.500000	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	...	0.0000
75%	1079.500000	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	...	171.0000
max	1460.000000	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.0000

8 rows × 38 columns

- There is lot of difference b/w mean and std
- There is also lot of difference b/w 75% and max
- we can expect both outliers and skewness

Based on the above results i have observed.

- Huge Difference b/w max value and 75% in SalePrice, MSSubClass, LotFrontage,LotArea, BsmtFinSF1,BsmtFinSF2 etc which are indicating presence of outliers.
- There is the presence of higher standard deviation which indicates the presence of skewness.

Housing Price Prediction

Hardware and Software Requirements and Tools Used

Hardware Used:

- Processor Apple M1
- Physical Memory: 8 GB

Softwares Used:

- MAC Operating System
- Anaconda Package and Environment Manager: Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes datascience packages suitable for Windows and provides a host of tools and environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects.
- Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.
- Python3: It is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used for Data Analytics And Data science projects/application. Python provides numerous libraries to deal with mathematics, statistics and scientific function.
- Python Libraries
- Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc
- Numpy: For performing a variety of operations on the datasets.
- matplotlib.pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns

- Scipy: For performing operations on the datasets
- Statsmodels: For performing statistical analysis
- sklearn for Modelling Machine learning algorithms, Data Encoding, Evaluation metrics, Data Transformation,Data Scaling, Component analysis,Feature selection etc

Visualisation For Object Type Features

Univariate Analaysis

Housing Price Prediction

Observations

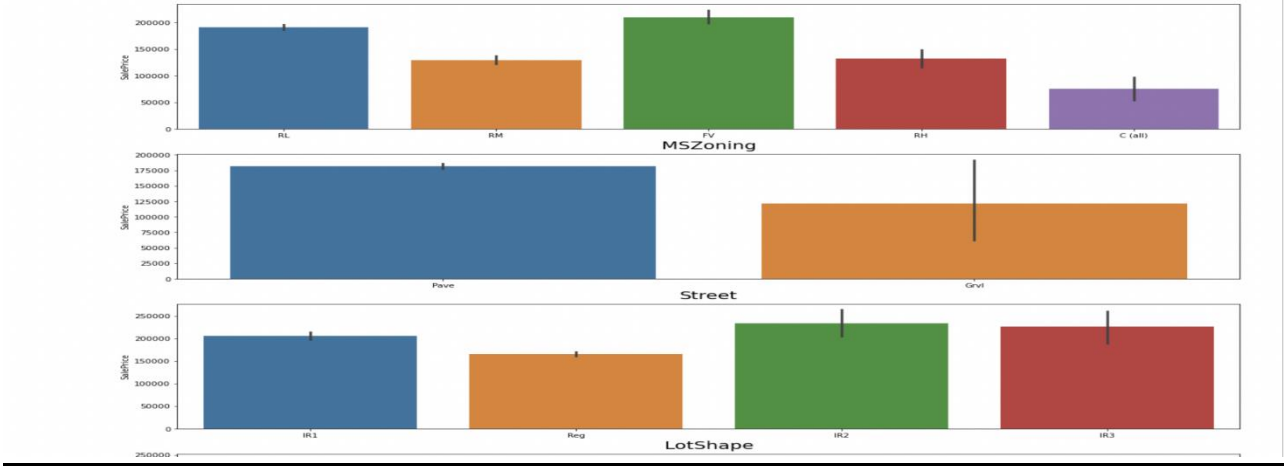
- MSZoning has the more data related to RL
- Most common street type isi pave
- Most of the Lotshapes has the regular structures
- Most of the properties has lv1
- utilities has only one value(ALLPUB)
- Most common LotConfig is Inside type
- slope of property land is mostly gentle
- Most of the properties are situated in neighborhoods of Neighborhoods of North AMerica,followed by college creek Edwards and old town
- Most of the housing properties with feature condition1 and condition2 are in normal conditions
- Mostly all the propertie sare 1 storey and 2 storey

Housing Price Prediction

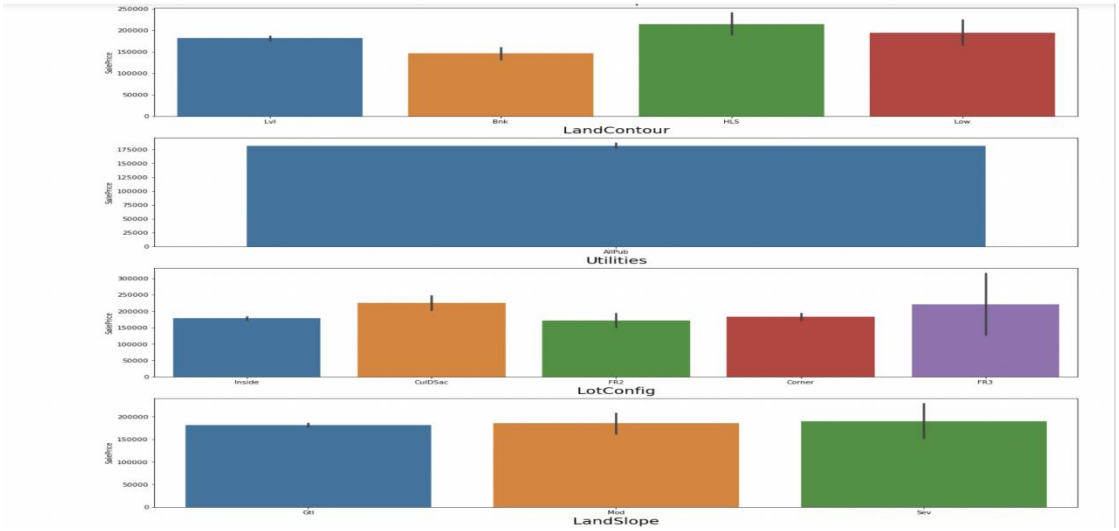
- Most of the properties with feature BldgtYPE HAS IFam type
- Most of properties have the Gable type roofstyle
- Most of the properties used roofMatl has Compshg
- most of the properties has Exteriors has vinylsd type
- most of properties dont have masonry veneer type while some has brick face
- Most of the properties has the typical/average exteriors
- Most of the properties has the typical/average exterior conditions
- most of the properties has foundation type as cblock.pconc
- The height of the basement is usually either typical (80-89 inches) or good (90-99 inches)
- The general condition of the basement is commonly Typical with slight dampness
- Basements most commonly have no exposure
- Most of the houses basements are unfinished followed by houses with basements having good living quarters
- most houses have gas forced warm air furnance heating arrangment
- most properties have good good quality and condition
-
- most properties have air conditioning
- most properties have standarad circuit breakers and romex electrical system
- most properties have good kitchen quality
- most properties has garage attached to home
- most of the properties has unfinished garage
- garage condition is usually typical/average
- conventional is most common type of sale

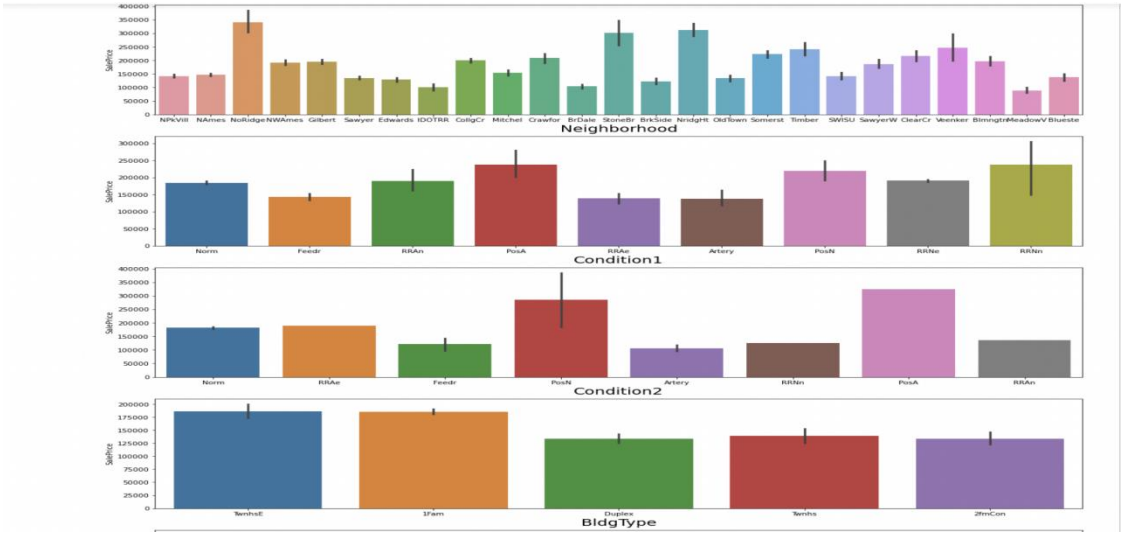
Bivariate Analaysis

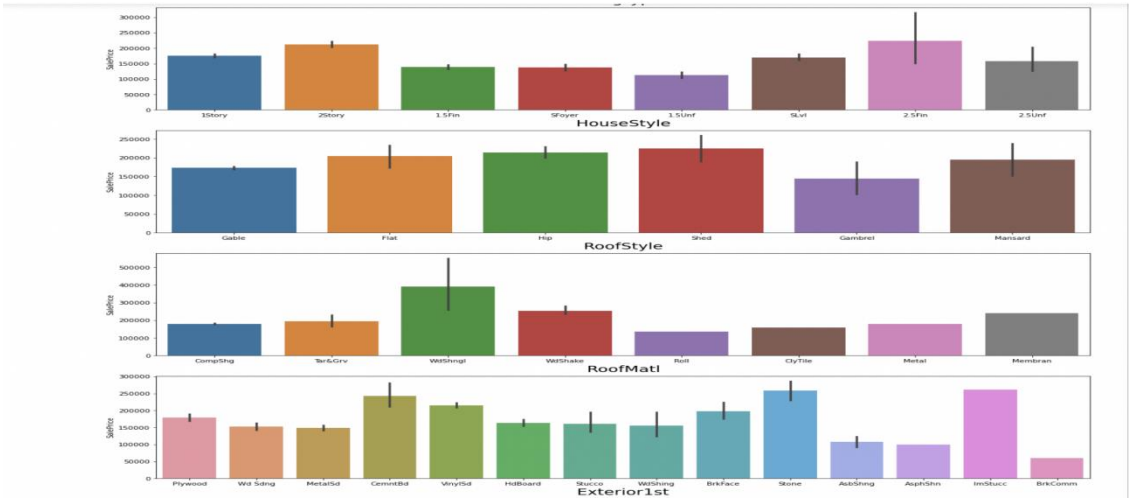
```
In [186]:
plt.figure(figsize=(20,200),facecolor ='white')
plotnum = 1
for col in train_df[train_df.columns[train_df.dtypes == 'object']]:
    if plotnum<=40:
        plt.subplot(40,1,plotnum)
        sns.barplot(x=train_df[col],y=train_df.SalePrice)
        plt.xlabel(col,fontsize=20)
    plotnum+=1
plt.show()
```



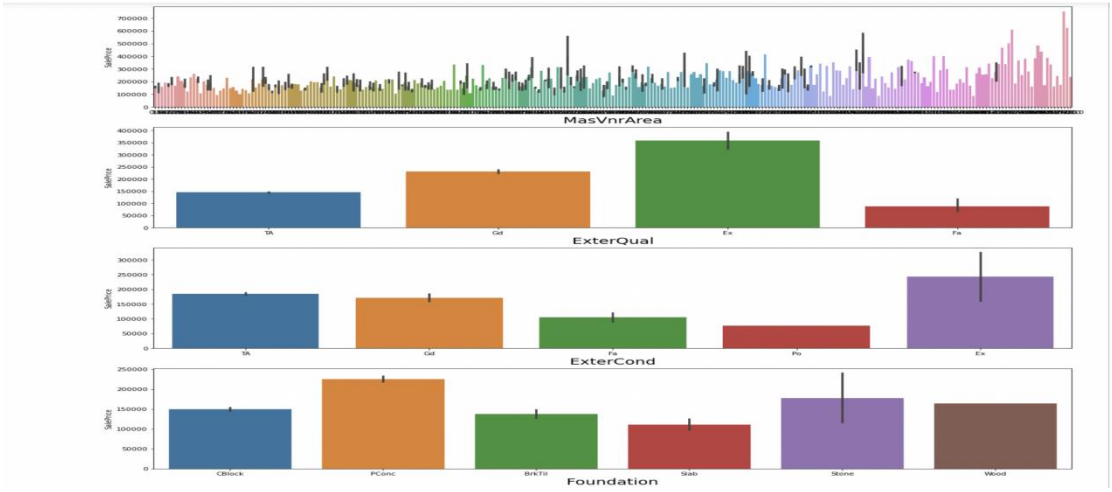
Housing Price Prediction

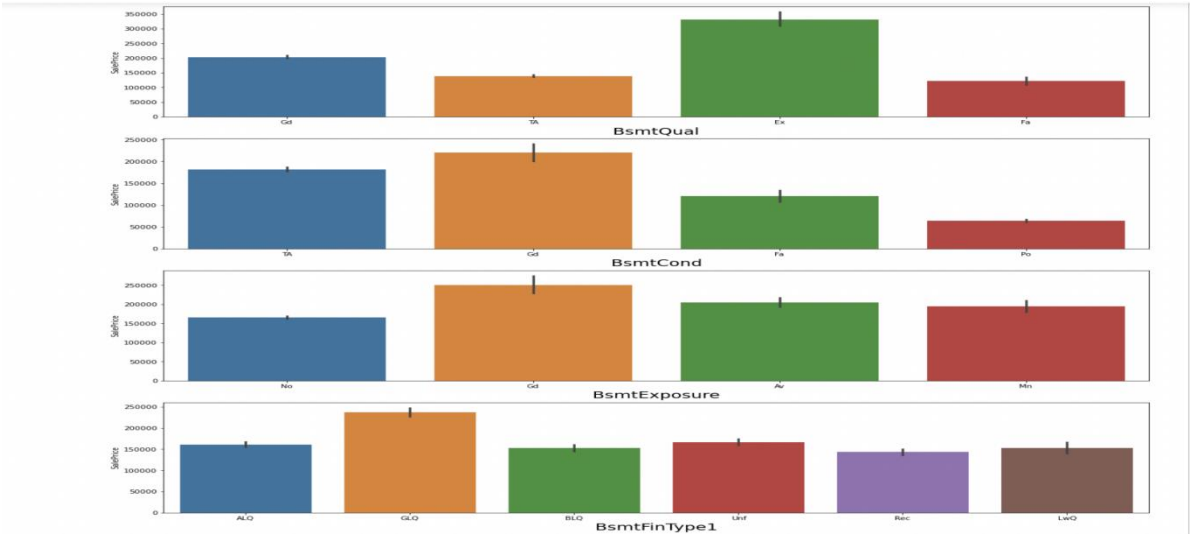


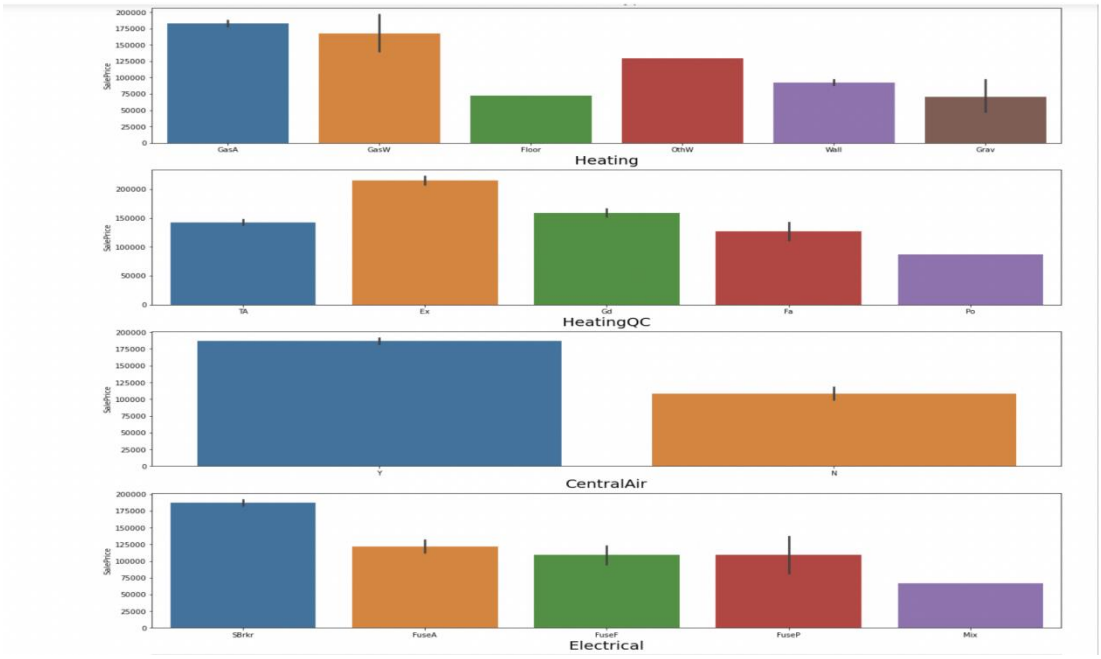




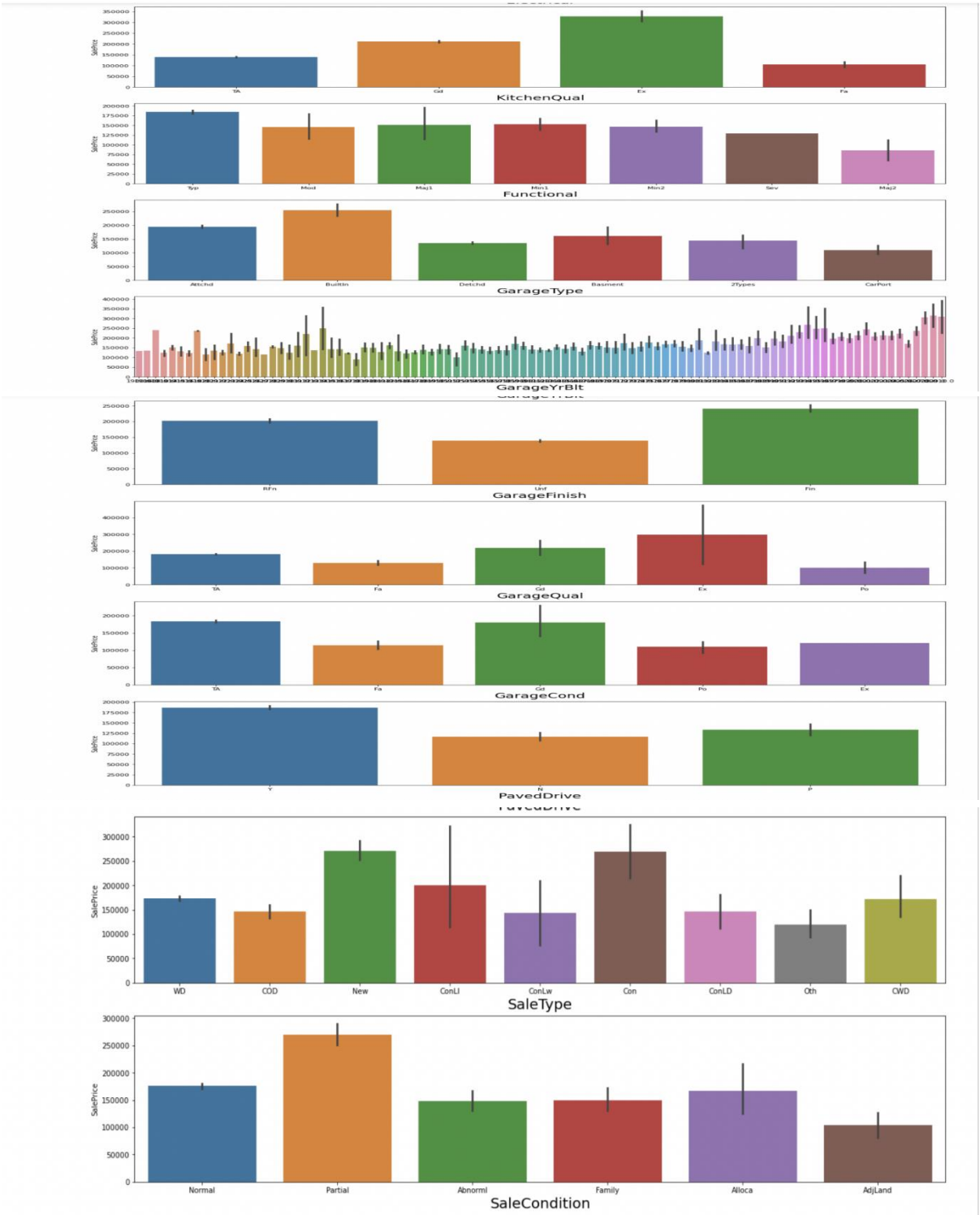
Housing Price Prediction







Housing Price Prediction



Observations:

- In MSZoning most of the sale price is contributed by FV

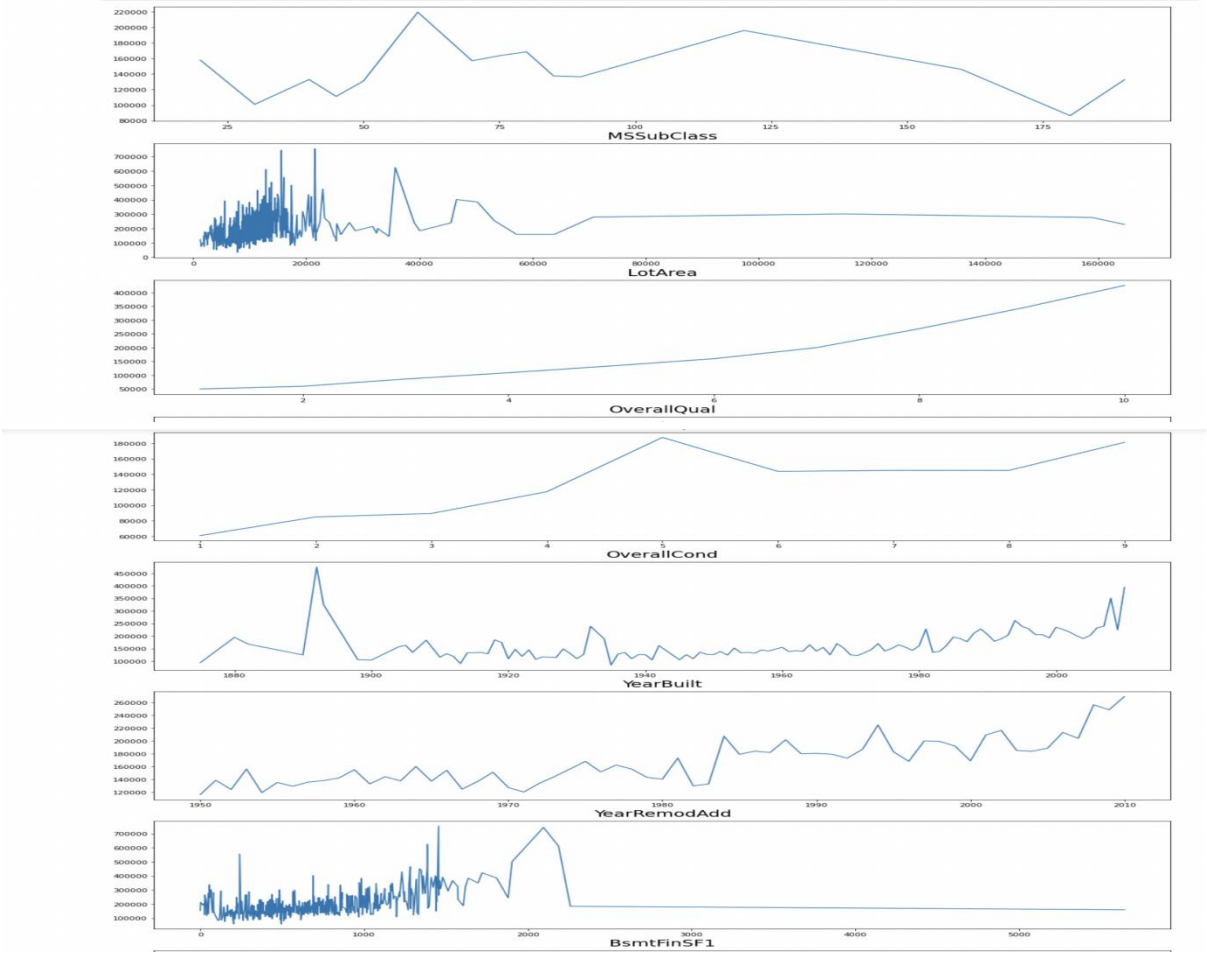
Housing Price Prediction

- Most of streetw which are contributed in high sale price is Pvae type streets
- High sale price is also contributed by having the LotShape of the house as IR2 followed by IR3 and IR1
- High sale price is contributed bu having LandContour as HLS followed by Low and Lvl type
- LotConfig feature has high sale price when the config type is either FR3 or CulDSac
- Landslope feature is contributing equally in sale price
- Neighbourhood feature of saleprice is highly contributing with variable NoRidge
- In Condition1 and Condition2 feature the sale price is high when the variable is POSA followed by PosN
- BldgType feature with variable TwnhsE,1Fam contributing more for the saleprice
- HouseStyle with 2.5Fin, 2Story are contrbuting more in house sale price
- Roofstyle with Shed type followed by Hip,Flat, Mansard are contributing more for the saleprice
- The house with RoofMatl WidShngl is contributing the high cost in the saleprice
- Exterior1st feature with Stone,CemntBd type are contributing more for the house sale price
- Exterior2nd feature with Other,ImStucc type are contributing more for the house sale price
- ExterQual feature with Ex type is contributing more for the saleprice
- ExterCond feature with Ex type is contributing more for the saleprice and variable Po is contributing less with saleprice
- Foundation feature with Pconc is contributing more with the saleprice
- BsmtQual with ex type has the high sale price
- BsmtCond with Gd TYPE HAS THE high sale price
- BsmtExposure with Gd type has the high sale price
- BsmtFinType1 feature with GLQ has the high sale price
- BsmtFinType1 feature with GLQ,ALQ has the high sale price
- Heating Feature with GasA type is having teh high sale price
- HeatingQC feature with Ex type is contributing more for saleprice
- High sale price for houses will happen when there is central air conditioning system
- KitchenQual with Ex type is contributing more for the sale price
- Functional feature with Typ type is contributing more for sale price
- Builtin Garage type is having high saleprice whereas Detchd type has less contribution for sale price
- GarageFinish with Fin type has high contribution towards saleprice
- GarageQual with Gd type has high saleprice
- GarageCond with Gd type is contributing more for saleprice
- Yes paved drive is contributing more for sale price
- saletype with New,Con type has high saleprices
- Partial condition is having high sale price

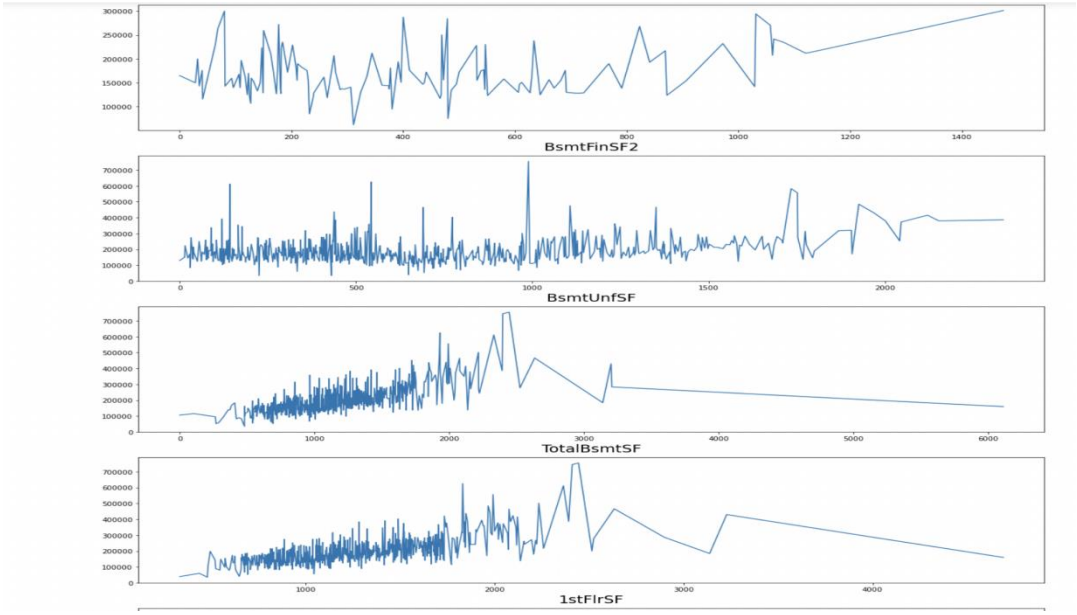
Housing Price Prediction

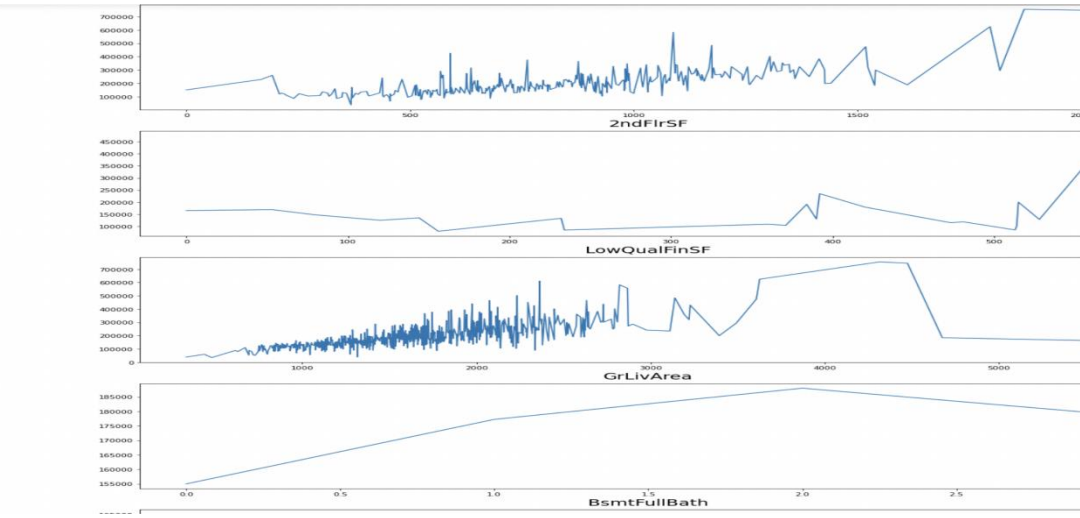
Visualisation For Numerical Type Features

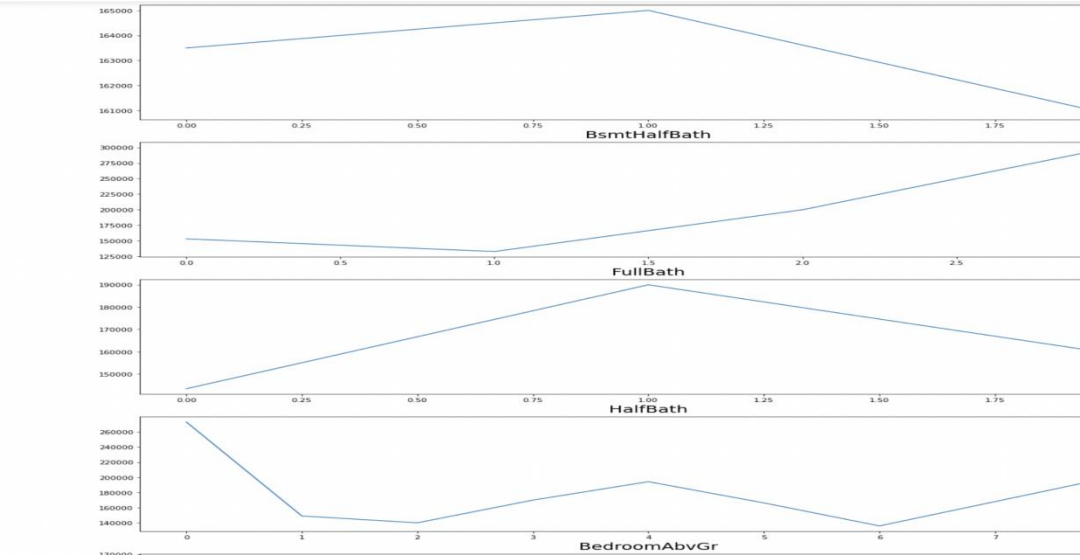
In [187]:
```
plt.figure(figsize=(20,200),facecolor ='white')
plotnum = 1
for col in train_df[train_df.columns[train_df.dtypes == 'int64']]:
    if plotnum<=40:
        plt.subplot(40,1,plotnum)
        a = train_df.groupby(col).median()
        try:
            a['SalePrice'].plot()
        except:
            pass
        plt.xlabel(col,fontsize=20)
    plotnum+=1
plt.show()
```



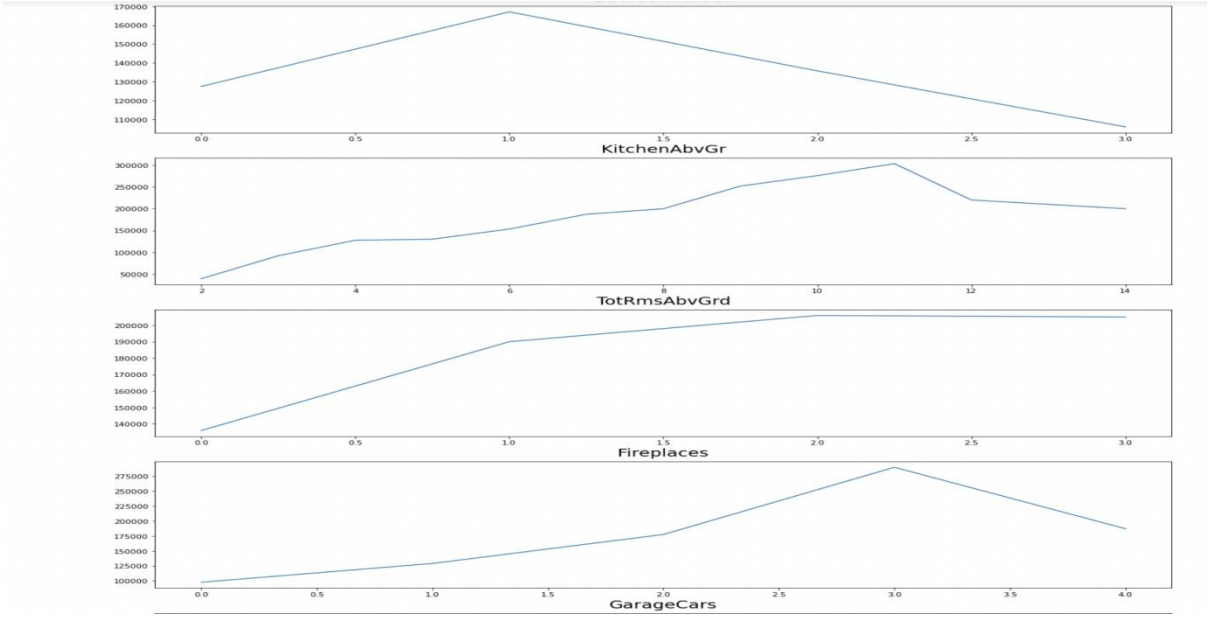
Housing Price Prediction

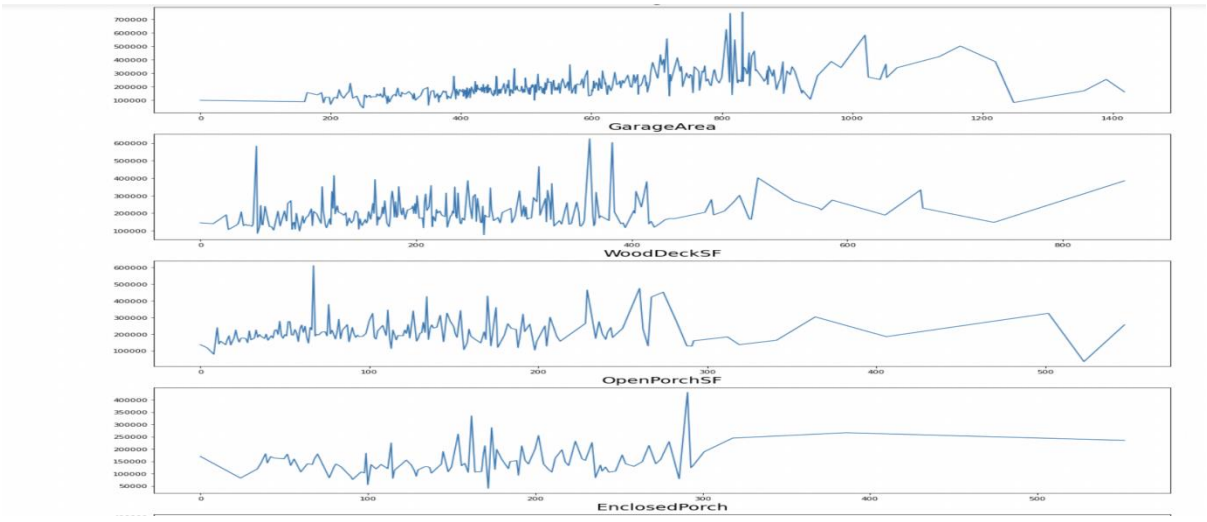


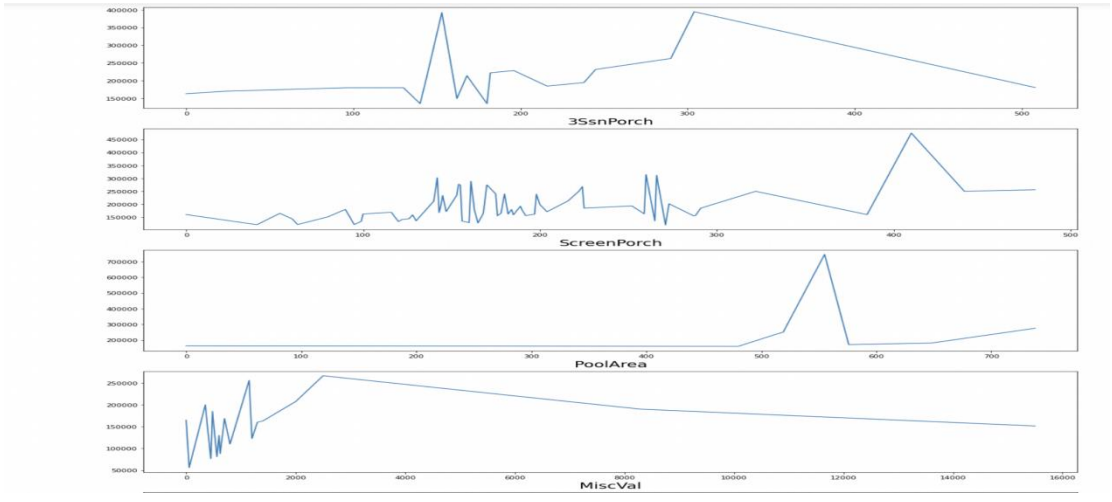




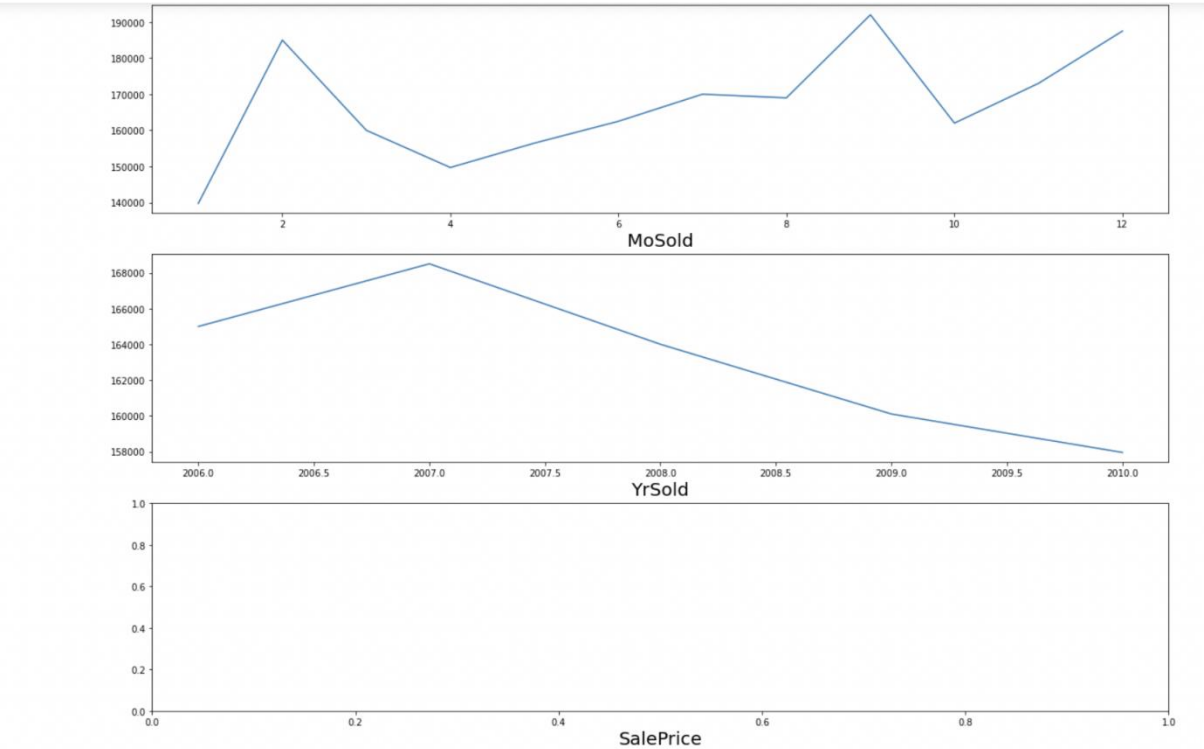
Housing Price Prediction







Housing Price Prediction



Observations:

- 1 story and 2 and 2.5 story houses built in 1946 and newer fetch the highest amount in sales.
- Houses with LotFrontage between 100 ft and 200 ft are sold for the highest amount.
- Houses wtih Lot area upto 25000 sqft fetch the highest amount.
- There is a Linear positive relation between Overall Quality and SalesPriceo There is a Linear positive relation between Overall Condition and SalesPrice
- There is a Linear positive relation between Masonry veneer area and SalesPrice
- Most Sales were done for Type 1 Finished basement with area upto 2500 sqft
- There is a Linear positive relation between Type 2 Finished basement area and SalesPrice
- There is a Linear positive relation between Total Basement area and SalesPrice
- There is a Linear positive relation between Total 1st area and 2nd floor area and SalesPrice
- There is a Linear positive relation between low Quality finished square feet and SalesPrice
- There is a Linear positive relation between Above grade living area square feet and SalesPrice
- There is a Linear negative relation between basment half bath and SalesPrice
- There is a Linear positive relation between Full Bathroom and SalesPrice
- There is a Linear positive relation between Total rooms above grade and SalesPrice
- There is a Linear positive relation between Fireplaces and SalesPrice

Housing Price Prediction

- There is a Linear positive relation between Garage Car capacity and SalesPrice
- There is a Linear positive relation between Garage area and SalesPrice
- Sales Prices peaked between 0-400 square feet area for Wooden Deck
- Sales Prices peaked between 0-300 square feet area for Open Porch
- Sales Price and Enclosed Porch area have a positive relation
- Sales Price and 3 season Porch area have a positive relation
- Sales Price and screen Porch area have a positive relation
- Sales Price and Month Sold have a positive relation
- Sales Price and Month Sold have a positive relation
- Sales Price and house age have a negative relationo Sales Price and remodelling age have a negative relation
- Sales Price and Garage age have a negative relation.

As we have some of the date related variables is some features in string format modifying it into Datetime format and extract useful insights from it.

Extracting the ages, remodeled house, garage age of the houses so it gives good insights to target variable

In [188]:
```
train_df['House_age'] = train_df['YearBuilt'].apply(lambda y: 2021 -y)
train_df['Remod_Age'] = train_df['YearRemodAdd'].apply(lambda y: 2021 -y)
train_df['Garage_age'] = train_df['GarageYrBlt'].apply(lambda y: 2021 -y)
test_df['House_age'] = test_df['YearBuilt'].apply(lambda y: 2021 -y)
test_df['Remod_Age'] = test_df['YearRemodAdd'].apply(lambda y: 2021 -y)
test_df['Garage_age'] = test_df['GarageYrBlt'].apply(lambda y: 2021 -y)
```

Dropping the columns after ages extraction

In [189]:
```
train_df.drop(columns=['YearBuilt','YearRemodAdd','GarageYrBlt'],axis=1,inplace=True)
test_df.drop(columns=['YearBuilt','YearRemodAdd','GarageYrBlt'],axis=1,inplace=True)
```

As we have some object related data(Categeorical), to create a Ml model we need all the data in Numerical format so Label encoding the values.

Housing Price Prediction

Label Encoding Object related columns

In [190]:
```
from sklearn.preprocessing import LabelEncoder
encodr = LabelEncoder()
for col in train_df[train_df.columns[train_df.dtypes == 'object']]:
    train_df[col] = encodr.fit_transform(train_df[col])
for col in test_df[test_df.columns[test_df.dtypes == 'object']]:
    test_df[col] = encodr.fit_transform(test_df[col])
```

In [191]:
```
#mapping yrsold and utilities to custom labels as they are numeric type
train_df['YrSold'] = train_df.YrSold.map({2007:2,2009:4,2006:1, 2008: 3, 2010: 5}) # encoding years in YrSold Column
train_df['Utilities'] = train_df.Utilities.map({0:1})
test_df['YrSold'] = test_df.YrSold.map({2007:2,2009:4,2006:1, 2008: 3, 2010: 5}) # encoding years in YrSold Column
test_df['Utilities'] = test_df.Utilities.map({0:1})
```

In [192]:
```
train_df.sample()
```

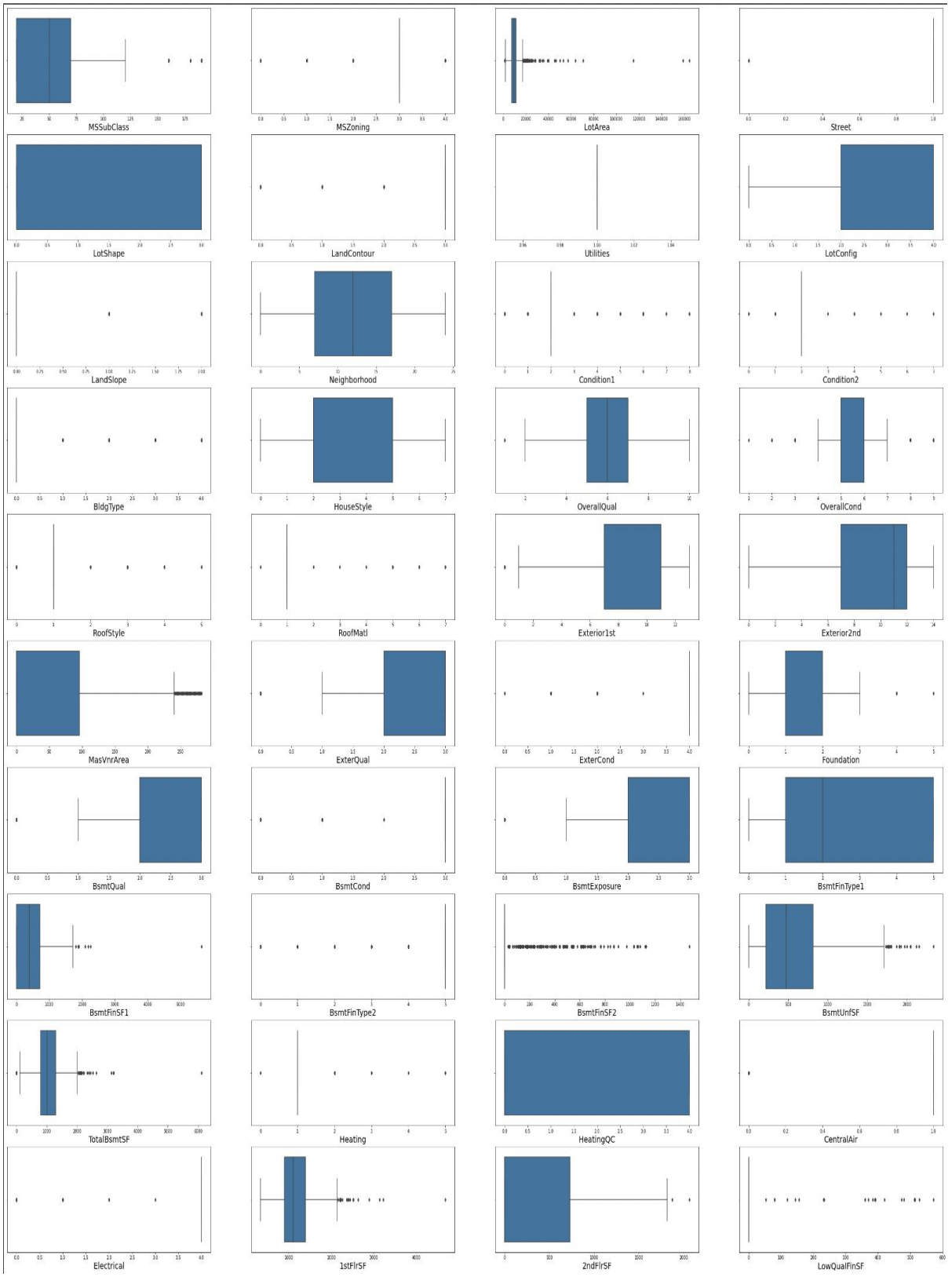
Out[192]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	...	PoolArea	MiscVal	MoSold	YrSold	Sa
831	180	4	21.0	1491	1	3	3	1	4	0	...	0	0	5	5	

1 rows x 74 columns

Checking For Outliers in the Dataset

Housing Price Prediction



We have some outliers in the dataset.

Housing Price Prediction

Lets Check the Amount of outliers present in Dataset Using Z-score method

Remving Outliers

Using z-score method

```
In [195]: zscore_df = train_df[['LotFrontage','MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'House_age',
                'Remod_Age', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
                '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
                'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
                'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
                'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
                'MiscVal', 'MoSold', 'YrSold', 'SalePrice']]

In [196]: # Applying z-score method
          from scipy.stats import zscore
          zscore_abs = np.abs(zscore(zscore_df))
          #THreshold is taken as +3/-3
          zdf = zscore_df[(zscore_abs < 3).all(axis=1)]

In [197]: train_df.shape

Out[197]: (1168, 74)

In [198]: zdf.shape

Out[198]: (814, 35)

In [199]: # Checking data loss after using z-score
          print((1168-814)/1168 * 100)

          30.30821917808219
```

- we have 30% of data loss after performing z-score which is not in considerable range

After Removing the outliers using z-score we have data loss of 30% which is not in considerable range

Lets Check the Amount of outliers present in Dataset Using IQR method

Using IQR method

```
In [200]:
          # Creating a function to remove outlier
          """Takes the dataframe and removed the outliers and will return back"""
          outlier_index = np.array([])
          for column in zscore_df:
              q1 = zscore_df[column].quantile(0.25)
              q3 = zscore_df[column].quantile(0.75)
              IQR = q3-q1
              col = zscore_df[column]
              indx = col[(col > q3+1.5*IQR) | (col < q1-1.5*IQR)].index
              outlier_index = np.append(outlier_index,indx)
          outlier_index

Out[200]: array([  12.,   19.,   35., ..., 1032., 1064., 1150.])

In [201]: print(zscore_df.shape)
          new_df = zscore_df.drop(outlier_index)
          print(new_df.shape)

          (1168, 35)
          (471, 35)

In [202]: # Checking data loss after using IQR
          print((1168-471)/1168 * 100)

          59.67465753424658
```

- Not at all in considerable range

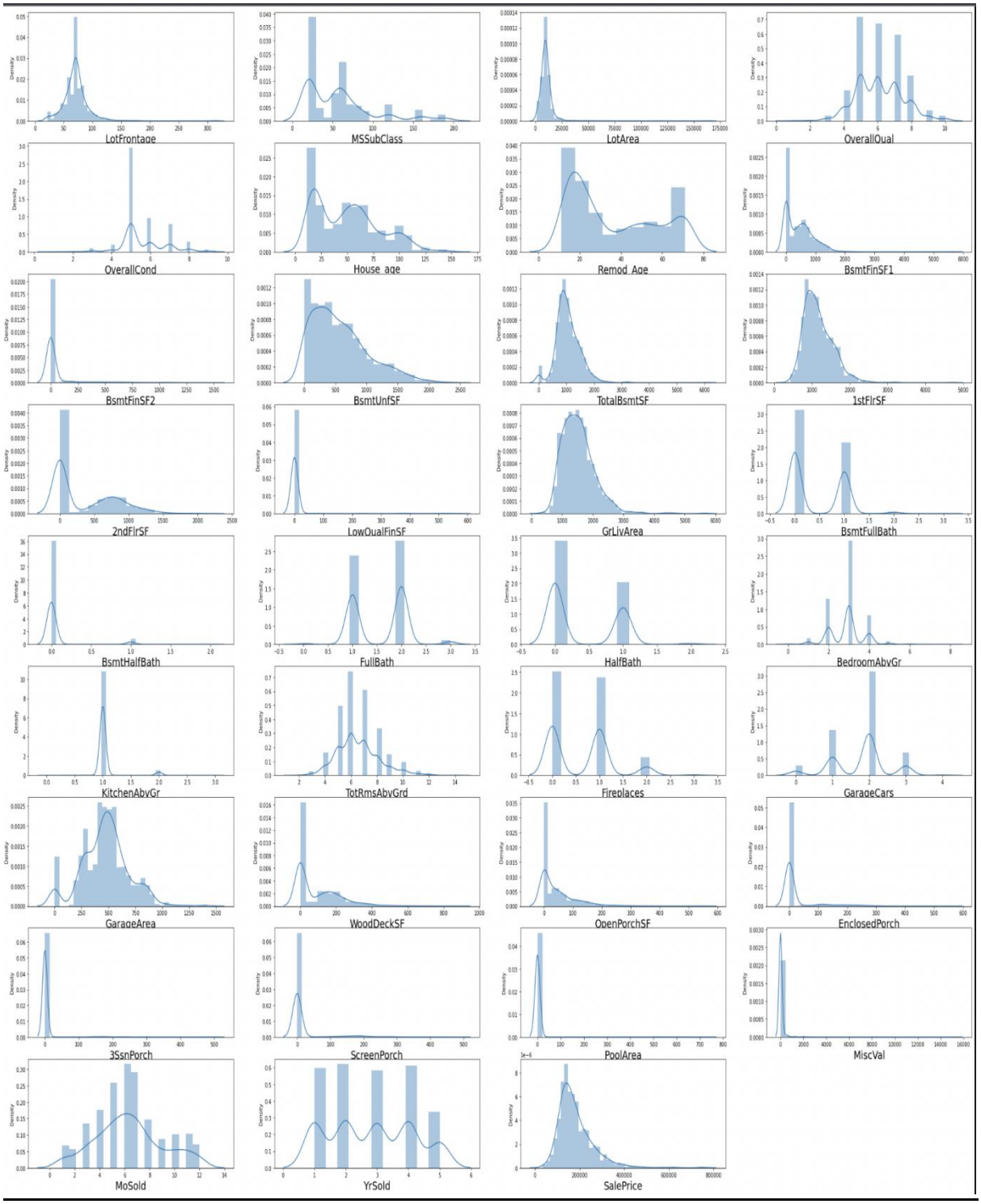
- Leaving outliers as it is

After Removing the outliers using IQR we have data loss of 59% which is not in considerable range

So leaving the outliers as it is because i dont want to lose the valuable data.

Checking for the skewness in the Dataset

Housing Price Prediction



Lot of the data are right skewed in the dataframe

Some of the columns are multinodal.

Housing Price Prediction

Skewness Values

```
In [204]: zscore_df.skew()

Out[204]: LotFrontage       2.710383
          MSSubClass        1.422019
          LotArea          10.659285
          OverallQual       0.175082
          OverallCond       0.580714
          House_age         0.579204
          Remod_Age         0.495864
          BsmtFinSF1        1.871606
          BsmtFinSF2        4.365829
          BsmtUnfSF         0.909057
          TotalBsmtSF       1.744591
          1stFlrSF          1.513707
          2ndFlrSF          0.823479
          LowQualFinSF      8.666142
          GrLivArea         1.449952
          BsmtFullBath      0.627106
          BsmtHalfBath      4.264403
          FullBath          0.057809
          HalfBath          0.656492
          BedroomAbvGr      0.243855
          KitchenAbvGr      4.365259
          TotRmsAbvGrd      0.644657
          Fireplaces        0.671966
          GarageCars       -0.358556
          GarageArea        0.189665
          WoodDeckSF        1.504929
          OpenPorchSF       2.410840
          EnclosedPorch     3.043610
          3SsnPorch         9.770611
          ScreenPorch       4.105741
          PoolArea         13.243711
          MiscVal          23.065943
          MoSold            0.220979
          YrSold            0.115765
          SalePrice         1.953878
          dtype: float64
```

Handling Skewness Using PowerTransformer

Skewness removal using power transformer method

```
In [205]: from sklearn.preprocessing import PowerTransformer
          power_transform = PowerTransformer(method = 'yeo-johnson', standardize=True)
          nor_df = zscore_df.copy()
          transformed= power_transform.fit_transform(nor_df)
          transformed = pd.DataFrame(transformed, columns=nor_df.columns)
          transformed
```

5	-0.552490	0.490047	0.855555	0.656375	-0.496528	-1.398048	-1.200508	-1.347679	-0.363019	0.783582	...	0.852307	0.27
6	0.093658	-1.167999	0.424957	-0.785224	0.407009	0.632651	-0.306007	1.207441	-0.363019	-1.139284	...	-0.959773	-1.06
7	0.837233	-1.167999	0.717859	-0.785224	-1.498082	0.632651	-0.607149	0.033227	2.756025	-0.419477	...	-0.959773	-1.06
8	0.047197	-1.167999	0.001967	-0.785224	1.234321	0.413301	1.020922	0.790917	2.753010	-0.072115	...	-0.959773	-1.06
9	0.499839	0.237618	-0.152859	-0.785224	-0.496528	0.879341	1.389258	0.520753	-0.363019	-0.142851	...	1.002061	-1.06
10	-0.996296	0.237618	-0.125083	-0.052799	0.407009	1.101890	1.389258	-1.347679	-0.363019	0.624472	...	-0.959773	-1.06
11	-1.358798	0.490047	0.082104	0.656375	0.407009	-1.182964	-0.877119	0.571047	-0.363019	0.013711	...	1.064638	0.64
12	2.320316	-1.167999	0.007580	0.656375	-0.496528	-1.182964	-0.877119	-1.347679	-0.363019	1.683414	...	0.852307	1.26
13	0.499839	-1.167999	-0.059020	-0.785224	-0.496528	0.552965	1.154844	-1.347679	-0.363019	1.058400	...	0.751582	-1.06
14	-0.499737	0.237618	1.175905	-0.785224	1.234321	0.684216	1.277023	-1.347679	-0.363019	0.683517	...	0.915200	-1.06
15	-0.714204	0.696557	0.288554	-0.052799	1.234321	1.381821	-0.306007	-1.347679	-0.363019	0.308442	...	0.838749	-1.06
16	0.093658	-0.517772	-0.867869	-0.052799	2.000294	1.245955	-1.323923	0.633563	-0.363019	-0.436673	...	-0.959773	-1.06

```
In [206]: transformed.index = train_df.index
          train_df[['LotFrontage','MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'House_age',
                 'Remod_Age', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
                 '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
                 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
                 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
                 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
                 'MiscVal', 'MoSold', 'YrSold', 'SalePrice']] = transformed[['LotFrontage','MSSubClass', 'LotArea', 'OverallQual'
                 'Remod_Age', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
                 '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
                 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
                 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
                 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
                 'MiscVal', 'MoSold', 'YrSold', 'SalePrice']]
          train_df.skew()
```

Housing Price Prediction

After handling the skewness the values are:

MSSubClass	0.064007
MSZoning	-1.796785
LotFrontage	0.161368
LotArea	0.032509
Street	-17.021969
LotShape	-0.603775
LandContour	-3.125982
Utilities	0.000000
LotConfig	-1.118821
LandSlope	4.812568
Neighborhood	0.043735
Condition1	3.008289
Condition2	11.514458
BldgType	2.318657
HouseStyle	0.285680
OverallQual	0.021658
OverallCond	0.048063
RoofStyle	1.498560
RoofMatl	7.577352
Exterior1st	-0.612816
Exterior2nd	-0.592349
MasVnrArea	1.355502
ExterQual	-1.810843
ExterCond	-2.516219
Foundation	-0.002761
BsmtQual	-1.343781
BsmtCond	-3.293554
BsmtExposure	-1.166987
BsmtFinType1	-0.068901
BsmtFinSF1	-0.404528
BsmtFinType2	-3.615783
BsmtFinSF2	2.394737
BsmtUnfSF	-0.284390
TotalBsmtSF	0.286779
Heating	10.103609
HeatingQC	0.449933
CentralAir	-3.475188
Electrical	-3.104209
1stFlrSF	-0.002391
2ndFlrSF	0.280208
LowQualFinSF	6.922843
GrLivArea	-0.000054
BsmtFullBath	0.365488
BsmtHalfBath	3.954345
FullBath	-0.045944
HalfBath	0.498003
BedroomAbvGr	0.116498
KitchenAbvGr	-2.370593
KitchenQual	-1.408106
TotRmsAbvGrd	0.002332
Functional	-3.999663
Fireplaces	0.084950
GarageType	0.831142
GarageFinish	-0.450190
GarageCars	-0.022970

GarageArea	-0.320370
GarageQual	-4.582386
GarageCond	-5.422472
PavedDrive	-3.274035
WoodDeckSF	0.113026
OpenPorchSF	-0.002749
EnclosedPorch	2.022616
3SsnPorch	7.087955
ScreenPorch	3.067153
PoolArea	12.817372
MiscVal	4.991071
MoSold	-0.035838
YrSold	-0.083577
SaleType	-3.660513
SaleCondition	-2.671829
SalePrice	-0.006010
House_age	-0.063679
Remod_Age	0.039294
Garage_age	0.708074

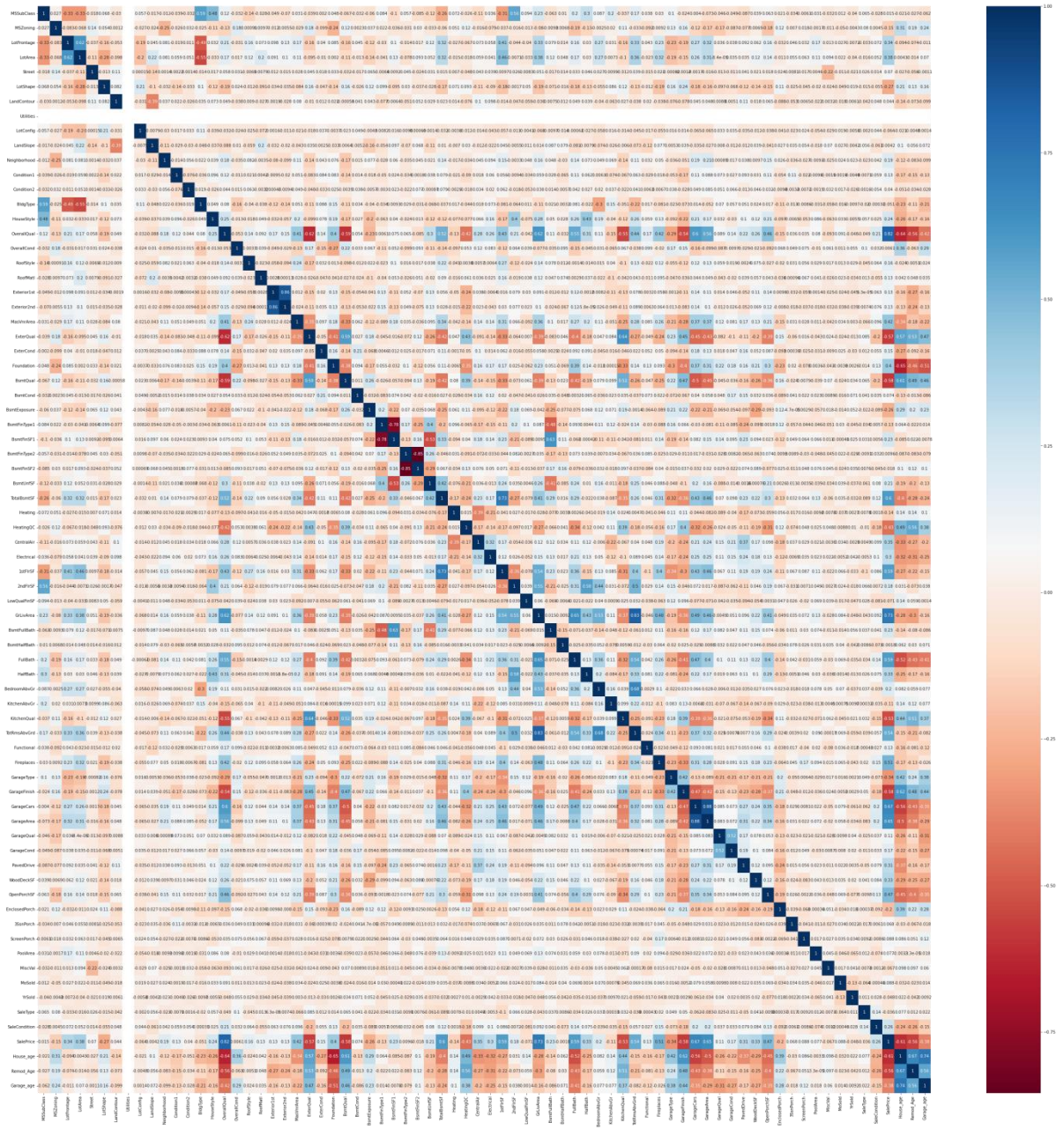
most of the skewness is removed from the data

Handled the same for test data also:

In [207]:
```
zscore_test_df = test_df[['LotFrontage','MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'House_age',
       'Remod_Age', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
       '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
       'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
       'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold']]
from sklearn.preprocessing import PowerTransformer
power_transform = PowerTransformer(method = 'yeo-johnson', standardize=True)
nor_df = zscore_test_df.copy()
transformed= power_transform.fit_transform(nor_df)
transformed = pd.DataFrame(transformed, columns=nor_df.columns)
transformed.index = test_df.index
test_df[['LotFrontage','MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'House_age',
       'Remod_Age', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
       '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
       'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
       'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold']] = transformed[['LotFrontage','MSSubClass', 'LotArea', 'OverallQual', 'OverallCon
       'Remod_Age', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
       '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
       'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr',
       'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold']]
test_df.skew()
```

Housing Price Prediction

Analayzing Correlation in train Data



- Max Number of features are positively correlated with SalePrice
- Yearbuild is negatively correlated with EnclosedPorch
- BsmtUnfsF is negatively correlated with BsmtFullBath

Housing Price Prediction

- ExternalQual,BsmtQual,KitchenQual, GarageFinish is negatively correlated with saleprice

Dropping Utilities feature as it is not showing scope on sale price

Dropping utilities folde as it is not much useful for model creation

In [209]:
```
train_df.drop('Utilities',axis=1,inplace=True)
test_df.drop('Utilities',axis=1,inplace=True)
```

Feature Selection

Feature selection

In [210]:
```
x = train_df.drop(columns=['SalePrice'])
y = train_df['SalePrice']
```

Standardization

STANDARDIZATION

In [211]:
```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
scaled_x = scalar.fit_transform(x)
```

Checking for multicollinearity using variation inflation factor

Checking for multicollinearity using variation inflation factor

In [212]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif["Features"] = x.columns
vif['vif'] = [variance_inflation_factor(scaled_x,i) for i in range(scaled_x.shape[1])]
```

In [213]:
```
vif
```

Out[213]:

	Features	vif
0	MSSubClass	6.044518
1	MSZoning	1.398651
2	LotFrontage	2.174025
3	LotArea	2.877675
4	Street	1.178426
5	LotShape	1.291252
6	LandContour	1.327374
7	LotConfig	1.156893
8	LandSlope	1.490095
9	Neighborhood	1.268704
10	Condition1	1.160713
11	Condition2	1.091204

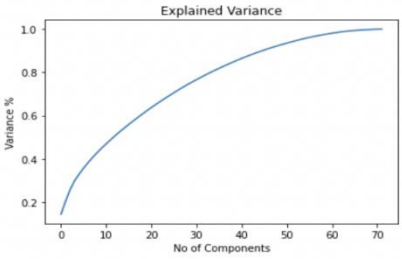
- MSSubClass, BsmtFinSF1, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, GrLivArea, House_age has high multi collinearity

MSSubClass, BsmtFinSF1, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, GrLivArea, House_age has high multi collinearity.

Principle Component Analysis

Principle Component Analaysis

In [214]:
```
from sklearn.decomposition import PCA
pca = PCA()
principlecomponents = pca.fit_transform(scaled_x)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('No of Components')
plt.ylabel('Variance %')
plt.title('Explained Variance')
plt.show()
```



- 70 components speak about 95% variance in data

70 components speak about 95% variance in data.

Selecting K-best Features and dropping un-useful columns

Housing Price Prediction

Selecting k-best features

```
In [216]: from sklearn.feature_selection import SelectKBest, f_classif
          bestfeat = SelectKBest(score_func=f_classif,k='all')
          fit = bestfeat.fit(x,y)
          dfscores = pd.DataFrame(fit.scores_)
          dfcolumns = pd.DataFrame(x.columns)
          featurescores = pd.concat([dfcolumns,dfscores],axis=1)
          featurescores.columns=['Feature','Score']
          print(featurescores.nlargest(75,'Score'))
```

```
60  EnclosedPorch  0.971408
22      ExterCond  0.971359
64        MiscVal  0.965799
65         MoSold  0.965082
39   LowQualFinSF  0.941239
13     HouseStyle  0.934468
62    ScreenPorch  0.913819
61      3SsnPorch  0.904359
0      MSSubClass  0.895056
56     GarageCond  0.893599
42   BsmtHalfBath  0.853796
29   BsmtFinType2  0.823425
10     Condition1  0.803848
49     Functional  0.797839
30     BsmtFinSF2  0.793233
11     Condition2  0.782965
67       SaleType  0.778011
63       PoolArea  0.698724
46   KitchenAbvGr  0.655580
```

```
In [218]: x.shape
Out[218]: (1168, 72)
```

Selecting best features based on their scores

```
In [219]: x_best = x.drop(columns=['SaleType','Condition2','BsmtFinSF2']).copy()
          scaled_x_best = scalar.fit_transform(x_best)
```

Importing Model Libraries and Finding Random state

Model Building

Finding best Random State

```
In [220]: from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
          from sklearn.model_selection import GridSearchCV
          from sklearn.linear_model import Ridge
          from sklearn.tree import DecisionTreeRegressor
          from xgboost import XGBRegressor
          from sklearn.ensemble import RandomForestRegressor
          from sklearn.neighbors import KNeighborsRegressor
          from sklearn.ensemble import AdaBoostRegressor
          from sklearn.svm import SVR
          from sklearn.metrics import r2_score,mean_squared_error
          import joblib
          from sklearn.model_selection import cross_val_score,train_test_split,GridSearchCV
```

```
In [222]: maxAcc = 0
          maxRS=0
          for i in range(1,100):
              x_train,x_test,y_train,y_test = train_test_split(scaled_x_best,y,test_size = .33, random_state = i)
              modRF =  RandomForestRegressor()
              modRF.fit(x_train,y_train)
              pred = modRF.predict(x_test)
              acc  = r2_score(y_test,pred)
              if acc>maxAcc:
                  maxAcc=acc
                  maxRS=i
          print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")
```

```
Best Accuracy is: 0.8924397101311748 on random_state: 24
```

Best Accuracy is: 0.8924397101311748 on random_state: 24

Training Data with Different Model Libraries

Housing Price Prediction

```
In [223]: models= [RandomForestRegressor(),DecisionTreeRegressor(),XGBRegressor(),Ridge(),SVR(),AdaBoostRegressor(),KNeighborsReg
          x_train,x_test,y_train,y_test = train_test_split(scaled_x_best,y,test_size=.25,random_state=59)

          def compare_models():
              for model in models:

                  model.fit(x_train,y_train)
                  model_predict = model.predict(x_test)
                  score_r2 = r2_score(y_test, model_predict)
                  mean_error = mean_squared_error(y_test, model_predict)
                  crossval_score = cross_val_score(model,scaled_x_best,y,cv=4).mean()
                  print("R2 score of the model : ",model,"=",round(score_r2*100,1),"%")
                  print("mean squared error for model : ",model,"=",mean_error)
                  print("cross val score for model : ",model, "=",crossval_score)
                  print('_____')
          compare_models()
```

Results for Model Training

R2 score of the model : RandomForestRegressor() = 87.1 %
mean squared error for model : RandomForestRegressor() = 0.12404986872351596
cross val score for model : RandomForestRegressor() = 0.8543177964627404

_____

R2 score of the model : DecisionTreeRegressor() = 71.2 %
mean squared error for model : DecisionTreeRegressor() = 0.27647808913929195
cross val score for model : DecisionTreeRegressor() = 0.7029503817522236

_____

R2 score of the model : 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, ...) = 87.5 %
mean squared error for model : 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, ...) = 0.12040357496536425
cross val score for model : 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, ...) = 0.8526377678389339

R2 score of the model : Ridge() = 82.3 %
mean squared error for model : Ridge() = 0.16960554508358183
cross val score for model : Ridge() = 0.8549982920253969

R2 score of the model : SVR() = 82.1 %
mean squared error for model : SVR() = 0.17225632516358153
cross val score for model : SVR() = 0.8024883218752944

R2 score of the model : AdaBoostRegressor() = 80.4 %
mean squared error for model : AdaBoostRegressor() = 0.1881442455910854
cross val score for model : AdaBoostRegressor() = 0.7988978681272337

R2 score of the model : KNeighborsRegressor() = 78.3 %
mean squared error for model : KNeighborsRegressor() = 0.20842969143319431
cross val score for model : KNeighborsRegressor() = 0.7651416544528145

Random Forest Regression is giving good accuracy results comparing with other models

Performing HyperParameter Tuning for Random Forest Regression

Hyper parameter tuning with model RandomForestRegressorm

```
In [227]: import warnings
warnings.filterwarnings('ignore')
random_forest_params = {'n_estimators':[30,60,80],
                   'max_depth':[10,20,40],
                   'min_samples_leaf':[1,2,5,10,20,30],
                   'min_samples_split':[5,10,20],
                   'max_features':['auto','sqrt','log2'],
                   'criterion':['mse','mae']
    }
grid_classifier = GridSearchCV(RandomForestRegressor(),random_forest_params,cv=5,n_jobs=-1,verbose=1)
grid_classifier.fit(x_train,y_train)
print(f"The best parameters are {grid_classifier.best_params_}")
print(f"The best Score is {grid_classifier.best_score_}")
```

```
/Users/akhilmatta/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py:407: FutureWarning: Criterion
'mae' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='absolute_error'` which is equivalen
t.
  warn(
/Users/akhilmatta/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py:407: FutureWarning: Criterion
'mae' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='absolute_error'` which is equivalen
t.
  warn(
/Users/akhilmatta/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py:407: FutureWarning: Criterion
'mae' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='absolute_error'` which is equivalen
t.
  warn(
/Users/akhilmatta/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py:407: FutureWarning: Criterion
'mae' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='absolute_error'` which is equivalen
t.
  warn(

The best parameters are {'criterion': 'mse', 'max_depth': 40, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_sam
ples_split': 5, 'n_estimators': 60}
The best Score is 0.8525915175039765
```

The best parameters are {
'criterion': 'mse',
'max_depth': 40,
'max_features': 'sqrt',
'min_samples_leaf': 1,
'min_samples_split': 5,
'n_estimators': 60}

The best Score is 0.8525915175039765

Training the model with good Parameters

Housing Price Prediction

```
In [228]: Best_mod1 = RandomForestRegressor(n_estimators=60,criterion='mse',max_depth=40,max_features='sqrt',min_samples_leaf=1,m
          Best_mod1.fit(x_train,y_train)
          rfrpred = Best_mod1.predict(x_test)
          acc = r2_score(y_test,rfrpred)
          print(acc*100)

          85.67670571543046
```

The Random Forest Regressor model got accuracy of

85.6%

Saving the Model

Saving the model

```
In [229]: import joblib
          joblib.dump(Best_mod1,"Housing.pkl")

Out[229]: ['Housing.pkl']
```

Loading the model

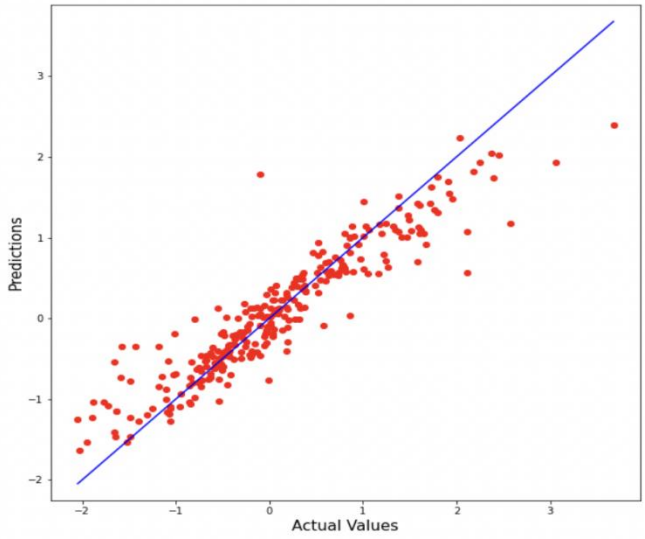
Loading the model

```
In [230]: mod = joblib.load("Housing.pkl")
          print(mod.predict(x_test))

[ 1.51530174e+00  8.12447687e-01 -1.01016992e-01 -2.05854784e-01
  3.17146065e-01  1.27535700e-01  6.27792997e-01  6.14869294e-01
  5.48534873e-01  5.72972118e-01 -1.72002359e-01  4.90603508e-01
 -3.50812007e-01  8.26314924e-01 -3.32780203e-01 -1.18538173e+00
  7.77506346e-03  4.87059062e-01 -9.38102996e-01  1.00032409e+00
  2.63188138e-01 -6.30411738e-01 -6.00447522e-01  5.62138494e-01
  1.02142690e+00 -5.56595195e-04  1.17936066e+00  7.51882576e-01
 -7.98523369e-01  6.71278326e-01  8.16024835e-01  1.14634178e+00
  1.65553040e-01 -6.00572742e-01  2.23832833e+00  2.48747566e-02
 -1.47242298e+00  1.78016669e+00 -1.07842196e+00 -4.68348268e-01
 -1.00499555e+00 -4.89701896e-01 -6.13924120e-01 -1.53787506e+00
  1.00660140e+00  5.40411813e-01 -2.49990178e-01  5.36342722e-01
  5.79707269e-01  2.93238905e-01 -1.03793953e+00  1.22725731e-01
  1.60807889e-01 -1.70030627e-01 -1.09902231e+00  1.40288961e+00
 -1.09416275e+00 -8.45833258e-01 -4.64514226e-01 -5.61846000e-01
  9.06108405e-01  5.05833264e-01 -1.16197847e+00  9.92096181e-01
 -5.38090575e-01  1.33937795e-01  5.39823729e-01 -2.93465571e-01
 -5.85097446e-01  5.69113898e-01 -1.67691285e-02  1.44297164e+00
```

Housing Price Prediction

Visualising Predicted and actual values

```
In [231]: plt.figure(figsize=(10,10))
          plt.scatter(y_test, rfrpred, c='red')
          plt.yscale('linear')
          plt.xscale('linear')

          p1 = max(max(rfrpred), max(y_test))
          p2 = min(min(rfrpred), min(y_test))
          plt.plot([p1, p2], [p1, p2], 'b-')
          plt.xlabel('Actual Values', fontsize=15)
          plt.ylabel('Predictions', fontsize=15)
          plt.axis('equal')
          plt.show()
```



Testing Model with Test Data

Testdata

```
In [233]: x_test_best = test_df.drop(columns=['SaleType','Condition2','BsmtFinSF2']).copy()
          scaled_x_test_best = scalar.fit_transform(x_test_best)

In [234]: mod = joblib.load("Housing.pkl")
          print(mod.predict(scaled_x_test_best))
```

```
  4.56903315e-01 -1.15038566e+00 -2.48966947e-01  5.26084440e-01
 -4.67533456e-01 -2.32459085e-02  1.20201963e+00 -8.46311085e-02
  1.86235359e-02 -6.19123582e-02 -5.29892936e-01  7.94750249e-01
  1.49766193e+00  3.63049535e-01  1.25567026e+00 -3.65700780e-01
  3.27105636e-01 -2.91321518e-01 -3.15312119e-01 -1.09379360e-01
  2.23821756e-01  6.67991977e-01 -9.26111000e-01  1.96543981e+00
 -2.10886158e-01  2.39867707e-01  9.27803112e-01 -5.36460967e-01
 -4.81966143e-01 -7.27310333e-01  3.34206959e-01 -6.15665304e-02
  8.51220293e-01  2.22503051e-01  1.74167491e+00 -6.56363552e-01
  9.76457364e-01 -1.39409626e+00 -1.00212822e+00 -9.65757687e-02
  3.95113822e-01 -4.99078270e-01  1.06145410e+00 -2.03703312e-01
  3.49453412e-01  3.46085416e-01  4.41653426e-01  5.30285944e-02
  2.39332503e-01  8.67106178e-01 -7.60176167e-01 -1.13498424e+00
 -5.49170845e-01  3.32937052e-01 -4.27696402e-01 -1.01287497e+00
 -1.29008027e+00  4.30623335e-01  7.85767489e-01 -3.89251198e-01
 -4.27859466e-01  4.78095031e-01 -8.38695556e-01  1.42750736e-01
 -1.49759167e+00 -1.01212774e+00 -3.97877511e-01  6.52709786e-01
 -4.33051157e-01 -1.55022138e-01 -2.00670662e-03  1.48672433e+00
  5.03765114e-01 -6.86179315e-01  1.24445410e+00 -9.76515334e-01
```

```
In [235]: Prediction_accuracy = pd.DataFrame({'Predictions': mod.predict(scaled_x_test_best), 'Actual Values': y[0:292]})
          Prediction_accuracy
```

Out[235]:

	Predictions	Actual Values
0	1.646611	-0.671353
1	0.385230	1.181174
2	1.118156	1.197587
3	0.135784	0.326442
4	0.782864	0.635122
5	-1.526581	0.683393
6	-0.389004	-0.804316
7	1.395865	-0.185682
8	0.815575	-0.443445
9	0.004957	-0.868211
10	-1.915331	-0.846727

Random Forest Regressor Performed the best out of all the models that were tested.

CONCLUSION

Key Findings and Conclusions of the Study and Learning Outcomes with respect to Data Science

Based on the in-depth analysis of the Housing Project, The Exploratory analysis of the datasets, and the analysis of the Outputs of the models the following observations are made:

• Structural attributes of the house Structural attributes of the

house like lot size, lot shape, quality and condition of the house, garage capacity, rooms, Lot frontage, number of bedrooms, bathrooms, overall finishing of the house etc play a big role in influencing the house price.

• Neighbourhood qualities can be included in deciding house price.

• Various plots like Barplots,Countplots and Lineplots helped in visualising the Feature-label relationships which corroborated the importance of structural and locational attributes for estimating Sale Prices.

• Due to the Training dataset being very small, the outliers had to be retained for proper training of the models.

• Therefore, Random Forest Regressor, being robust to outliers and being indifferent to non linear features, performed well despite having to work on small dataset.

Limitations of this work and Scope for Future Work

While features that focus on structural and locational attributes of housing properties are crucial for estimating the Sale Price of Housing properties, they aren't the only factors that influence the value in the housing market. Data on Demographics(Age,Income,Regional preferences of buyers, purpose of buying a property) is very important for understanding the Housing market. Interest Rates too impact the price and demand of houses. Economic cycles also influence Real Estate prices. Government Policies, Regulations, Legalizations are also important factors that may influence the sales of houses. The availability of data on above features would help build a predictive model that would more accurately understand the relationship between the features and target variable and yield more accurate predictions.

Thank You