FLIP ROBO

HOUSING-PRICE PREDICTION PROJECT

Submitted by:

Durgadhar Pathak

Internship 15

ACKNOWLEDGMENT

The internship opportunity I have with Flip Robo Technologies is a great chance for learning and professional development. I am also grateful to our SME Mr. Sajid Choudhary for his valuable and constructive suggestions during the planning and development of this project. His quick support and references helped a lot in building this project. Also, I am very thankful to our SME & Flip Robo Team for understanding technical issue faced by me and provide quick resolution & providing enough time for submission.

Also, I am thankful to DT support Team for their continuous effort to resolve our queries during project building.

Research papers that helped me in this project was as follows: -

1- https://www.researchgate.net/publication/323135322_A_hybrid_regression_technique_for_house_prices_prediction
2- https://ieeexplore.ieee.org/document/8473231
3- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3565512

Articles that helped me in this project was as follows:

1- https://www.sciencedirect.com/science/article/pii/S1877050920316318
2- https://www.tandfonline.com/doi/full/10.1080/09599916.2020.1832558
3- https://link.springer.com/article/10.1023/A:1007751112669

References:

1- https://machinelearningmastery.com/
2- https://scikit-learn.org/stable/
3- https://www.geeksforgeeks.org/machine-learning/
4- https://pandas.pydata.org/
5- https://www.datacamp.com/
6- https://www.ibm.com/cloud/learn/machine-learning

INTRODUCTION

• Business Problem Framing

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. Housing price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. House price prediction is an important topic of   real estate. There are three factors that influence the price of a house which include physical conditions, concept and location.

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

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

• Conceptual Background of the Domain Problem

We 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?

Also, we are required to predict 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.

Our goal for this project is to build an end-to-end solution that would be capable of predicting the house prices better than individuals and also to understand the relationship between house features and how these variables are used to predict house price.

our target variable is "Sale Price" which is continuous in nature so we will use Regression algorithms to make our model.

• Review of Literature

The use of machine learning to create a housing price prediction model is not new but in the recent years it is growing rapidly. It is particularly the increased complexity of assessing housing prices that has increased the attention to machine learning. Machine learning algorithms enable the creation of a new model using existing anonymized historical data that would be used to train the model to make better predictions not only for housing prices, but also for other variables like Neighbourhood, Foundation, Sale Condition etc. With a good model, housing companies could predict the price easily. To mitigate the subjective part of the decision-making process, different scoring models are introduced to evaluate certain parameters that could affect the housing prices.

Models used: -

1- **Random Forest Regression**: - Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

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

3- **k-nearest neighbors**: - K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors.

Other models used are: - Linear Regression, Decision Tree, Ridge & Lasso regression.

Hyper Parameter tuning: - For Lasso regression, k-nearest neighbors & AdaBoost Regression algorithms, we used Grid search cross-validation technique to choose the best hyper-parameters & for implementing best model using Random Forest Regression algorithm, we have used Randomized Search CV to find best hyperparameters.

Evaluation Matrix: - Coefficient of determination (R2 score), Cross Validation Score, Mean Absolute Error, Mean Squared Error & Root Mean Squared Error.

• Motivation for the Problem Undertaken

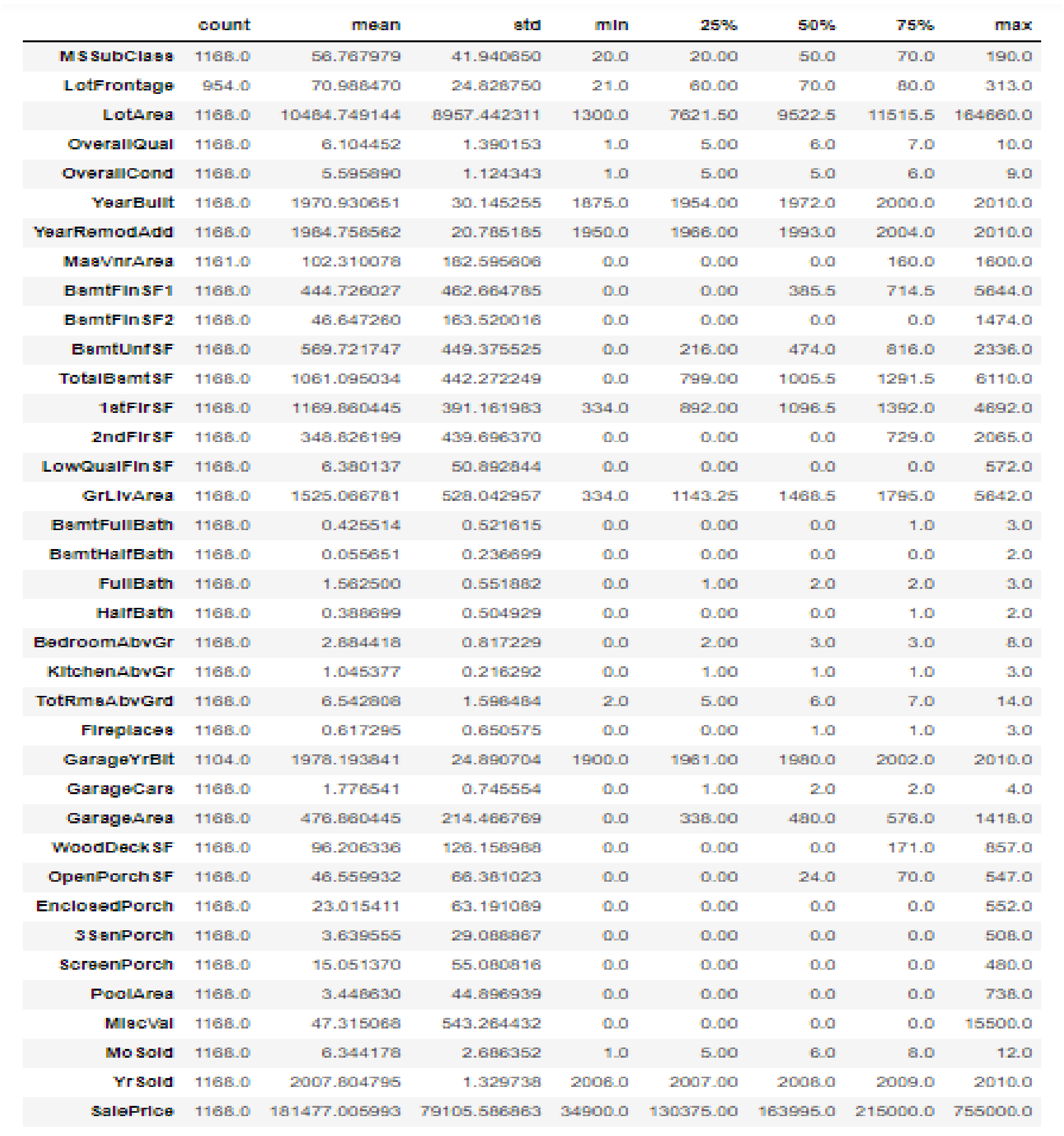
Houses are one of the necessary needs of each person around the globe and therefore housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. Housing and rental prices are continuing to rise everywhere. Machine Learning will definitely help the housing companies to increase their profit & market value. Also, it will help customers to purchase house as per their requirements and budget. So, we have worked on this model to complete these goals.

Analytical Problem Framing

• Mathematical/ Analytical Modelling of the Problem

In this project, input data is provided to the model along with the output data so it is a type of supervised learning. Also output Variable “Sale Price" is continuous in nature so it is a Regression based problem and We have to predict the price of houses with available independent variables. We have performed regression tasks and it models a target prediction value based on independent variables and is mostly used for finding out the relationship between variables and forecasting. Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. We have checked statistical summary, correlation matrix, skewness, missing values & outliers in dataset and try to handle them very carefully.

Statistical Summary: summary statistics is used to summarize set of observations, in order to communicate the largest amount of information as simply as possible. It includes central Tendency, dispersion, skewness, variance, range, deviation etc.



	count	mean	std	min	25%	50%	75%	max
MSSubClass	1168.0	56.767979	41.940650	20.0	20.00	50.0	70.0	190.0
LotFrontage	954.0	70.988470	24.828750	21.0	60.00	70.0	80.0	313.0
LotArea	1168.0	10484.749144	8957.442311	1300.0	7621.50	9522.5	11515.5	164660.0
OverallQual	1168.0	6.104452	1.390153	1.0	5.00	6.0	7.0	10.0
OverallCond	1168.0	5.595890	1.124343	1.0	5.00	5.0	6.0	9.0
YearBuilt	1168.0	1970.930651	30.145255	1875.0	1954.00	1972.0	2000.0	2010.0
YearRemodAdd	1168.0	1984.758562	20.785185	1950.0	1966.00	1993.0	2004.0	2010.0
MasVnrArea	1161.0	102.310078	182.595606	0.0	0.00	0.0	160.0	1600.0
BsmtFinSF1	1168.0	444.726027	462.664785	0.0	0.00	385.5	714.5	5644.0
BsmtFinSF2	1168.0	46.647260	163.520016	0.0	0.00	0.0	0.0	1474.0
BsmtUnfSF	1168.0	569.721747	449.375525	0.0	216.00	474.0	816.0	2336.0
TotalBsmtSF	1168.0	1061.095034	442.272249	0.0	799.00	1005.5	1291.5	6110.0
1stFlrSF	1168.0	1169.860445	391.161983	334.0	892.00	1096.5	1392.0	4692.0
2ndFlrSF	1168.0	348.826199	439.696370	0.0	0.00	0.0	729.0	2065.0
LowQualFinSF	1168.0	6.380137	50.892844	0.0	0.00	0.0	0.0	572.0
GrLivArea	1168.0	1525.066781	528.042957	334.0	1143.25	1468.5	1795.0	5642.0
BsmtFullBath	1168.0	0.425514	0.521615	0.0	0.00	0.0	1.0	3.0
BsmtHalfBath	1168.0	0.055651	0.236699	0.0	0.00	0.0	0.0	2.0
FullBath	1168.0	1.562500	0.551882	0.0	1.00	2.0	2.0	3.0
HalfBath	1168.0	0.388699	0.504929	0.0	0.00	0.0	1.0	2.0
BedroomAbvGr	1168.0	2.884418	0.817229	0.0	2.00	3.0	3.0	8.0
KitchenAbvGr	1168.0	1.045377	0.216292	0.0	1.00	1.0	1.0	3.0
TotRmsAbvGrd	1168.0	6.542808	1.598484	2.0	5.00	6.0	7.0	14.0
Fireplaces	1168.0	0.617295	0.650575	0.0	0.00	1.0	1.0	3.0
GarageYrBlt	1104.0	1978.193841	24.890704	1900.0	1961.00	1980.0	2002.0	2010.0
GarageCars	1168.0	1.776541	0.745554	0.0	1.00	2.0	2.0	4.0
GarageArea	1168.0	476.860445	214.466769	0.0	338.00	480.0	576.0	1418.0
WoodDeckSF	1168.0	96.206336	126.158988	0.0	0.00	0.0	171.0	857.0
OpenPorchSF	1168.0	46.559932	66.381023	0.0	0.00	24.0	70.0	547.0
EnclosedPorch	1168.0	23.015411	63.191089	0.0	0.00	0.0	0.0	552.0
3SsnPorch	1168.0	3.639555	29.088867	0.0	0.00	0.0	0.0	508.0
ScreenPorch	1168.0	15.051370	55.080816	0.0	0.00	0.0	0.0	480.0
PoolArea	1168.0	3.448630	44.896939	0.0	0.00	0.0	0.0	738.0
MiscVal	1168.0	47.315068	543.264432	0.0	0.00	0.0	0.0	15500.0
MoSold	1168.0	6.344178	2.686352	1.0	5.00	6.0	8.0	12.0
YrSold	1168.0	2007.804795	1.329738	2006.0	2007.00	2008.0	2009.0	2010.0
SalePrice	1168.0	181477.005993	79105.586863	34900.0	130375.00	163995.0	215000.0	755000.0

Observations on basis of Summary Statistics: -

1-Maximum Sales Price is 755000 and minimum sales price is 34900.

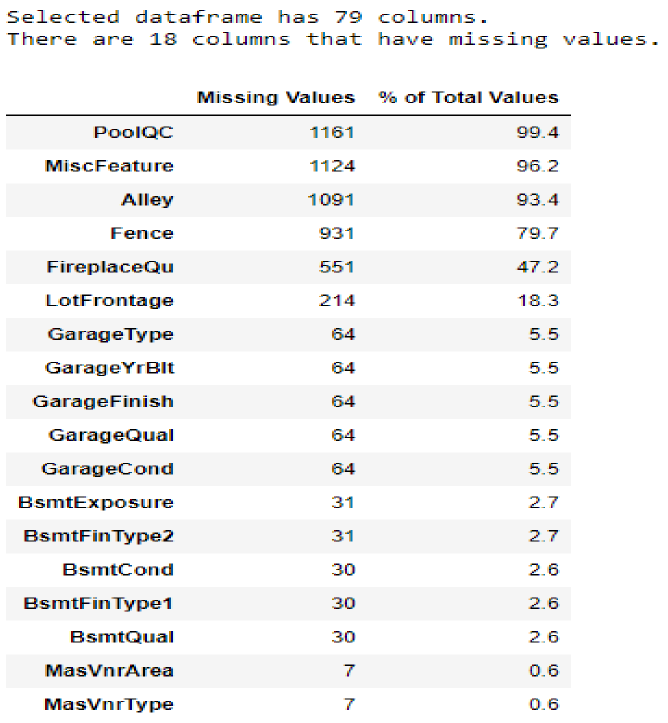
2-For input features, Lot Area has highest standard deviation of 8957.44.

3-In MSSubclass, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfsF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, HalfBath, TotRmsAbvGrd, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, Miscval & salePrice columns, the value of mean is considerably greater than median so there are strong chances of positive skewness.

4-In Full Hd, Bedroom, Fireplace, Garage Car, Garage Area & YrS old columns, value of median is greater than mean so the columns are negatively skewed.

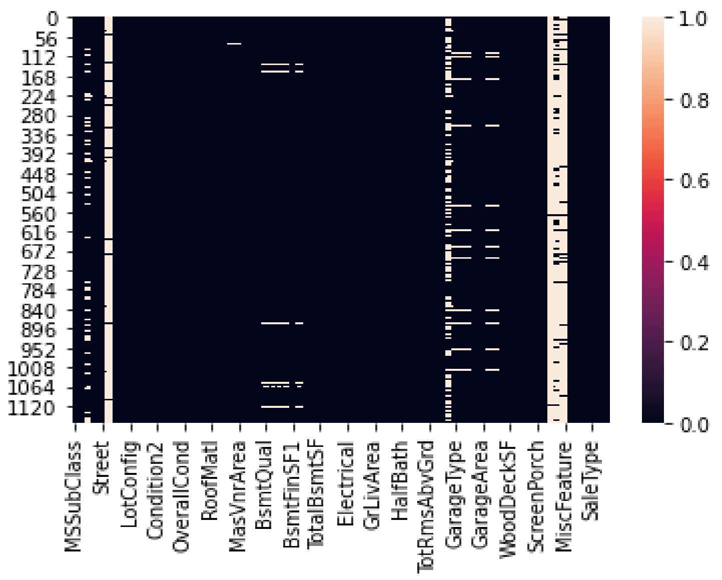
5- In MSSubClass, LotFrontage, LotArea, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtHalfBath, BedroomAbvGr, ToRmsAbvGrd, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal & SalePrice columns, due to considerable difference between the 75th percentile and maximum value so there are chances of presence of outliers.

Missing Data: - Missing data are values that are not recorded in a dataset. They can be a single value missing in a single cell or missing of an entire observation. Missing data can occur both in a continuous variable or categorical variable. The problem of missing data is relatively common in almost all research and can have a significant effect on the conclusions that can be drawn from the data. Missing data present various problems. First, the absence of data reduces statistical power, which refers to the probability that the test will reject the null hypothesis when it is false. Second, the lost data can cause bias in the estimation of parameters. Third, it can reduce the representativeness of the samples. Fourth, it may complicate the analysis of the study. Each of these distortions may threaten the validity of the trials and can lead to invalid conclusions. We had checked missing values in our dataset and also calculate percentage of missing values in our dataset. The missing data analysis is as: -



```
Selected dataframe has 79 columns.
There are 18 columns that have missing values.
```

	Missing Values	% of Total Values
PoolQC	1161	99.4
MiscFeature	1124	96.2
Alley	1091	93.4
Fence	931	79.7
FireplaceQu	551	47.2
LotFrontage	214	18.3
GarageType	64	5.5
GarageYrBlt	64	5.5
GarageFinish	64	5.5
GarageQual	64	5.5
GarageCond	64	5.5
BsmtExposure	31	2.7
BsmtFinType2	31	2.7
BsmtCond	30	2.6
BsmtFinType1	30	2.6
BsmtQual	30	2.6
MasVnrArea	7	0.6
MasVnrType	7	0.6



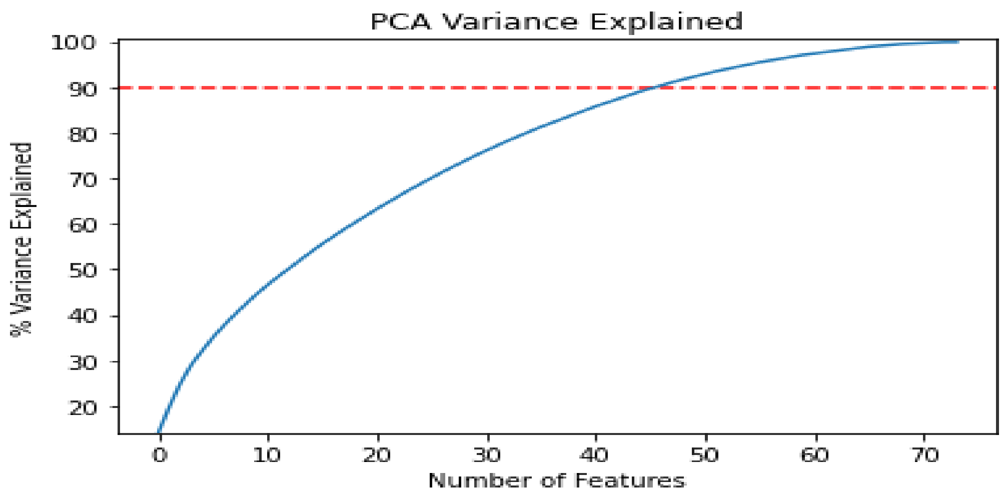
Observation:

Majority of data is missing in PoolQC, MiscFeature, Alley, Fence & FireplaceQu columns.

We have tried to treat these missing values in our dataset very carefully and also look for patterns of missingness and try to fill values in the place of missing data using imputation methods. We have handled missing values of category & continuous type data separately.

Removing Outliers: - As we have seen outliers in some columns so we were trying to remove them using Z scores but data loss was nearly 60.86%. Due to high amount of data loss, we eliminated the Z score implementation process.

PCA: - Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest. Multicollinearity was present between various columns so we did PCA on our dataset for dimensionality reduction and for making predictive models. Plot of PCA components with variance is shown below: -



As per Principal component analysis & variance analysis we need 45 features for higher percentage of information and less data loss.

Correlation: After seeing many correlated values we can say that many columns have correlation values and dropping some of these will be better for our dataset.

Skewness: If the skewness is between -0.5 and 0.5, then dataset is fairly symmetrical and symmetrical distribution will have a skewness of zero. So accordingly, we are removing skewness using NumPy mathematical functions log & square transform.

• Data Sources and their formats

1- **Dataset**: - The sample data is provided from client database in *.csv format
2- There are 18 columns that have missing values.
3- We have received 2 datasets, one for training and other for testing. Train Dataset have 1168 rows and 81 columns while test Dataset have 292 Rows and 80 columns.
4- Data contains numerical as well as categorical variable.
5- Dataset contains 3 types of data: - float data type, int data type and object data type.
6- We have used label encoder for Encoding categorical data in to numerical format for better processing.
7- We have used standard Scaler for pre-processing to bring features to common scale.

Features used in our Dataset and their format: -

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

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

MSZoning: Identifies the general zoning classification of the sale.

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

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl	Gravel
Pave	Paved

Alley: Type of alley access to property

Grvl	Gravel
Pave	Paved
NA	No alley access

LotShape: General shape of property

Reg	Regular
IR1	Slightly irregular
IR2	Moderately Irregular
IR3	Irregular

LandContour: Flatness of the property

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

Utilities: Type of utilities available

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

LotConfig: Lot configuration

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

LandSlope: Slope of property

Gtl	Gentle slope
Mod	Moderate Slope
Sev	Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn	Bloomington Heights

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

Condition1: Proximity to various conditions

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

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

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

BldgType: Type of dwelling

1Fam	Single-family Detached
2FmCon	Two-family Conversion; originally built as one-family dwelling

Duplx	Duplex
TwnhsE	Townhouse End Unit
TwnhsI	Townhouse Inside Unit

HouseStyle: Style of dwelling

1Story	One story
1.5Fin	One and one-half story: 2nd level finished
1.5Unf	One and one-half story: 2nd level unfinished
2Story	Two story
2.5Fin	Two and one-half story: 2nd level finished
2.5Unf	Two and one-half story: 2nd level unfinished
SFoyer	Split Foyer
SLvl	Split Level

OverallQual: Rates the overall material and finish of the house

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

OverallCond: Rates the overall condition of the house

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

YearBuilt: Original construction date

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

RoofStyle: Type of roof

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

RoofMatl: Roof material

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

Exterior1st: Exterior covering on house

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

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

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

MasVnrType: Masonry veneer type

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

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

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

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

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

Foundation: Type of foundation

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

BsmtQual: Evaluates the height of the basement

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

BsmtCond: Evaluates the general condition of the basement

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

BsmtExposure: Refers to walkout or garden level walls

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

BsmtFinType1: Rating of basement finished area

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

BsmtFinSF1: Type 1 finished square feet

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

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

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

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

HeatingQC: Heating quality and condition

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

CentralAir: Central air conditioning

N	No
Y	Yes

Electrical: Electrical system

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

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

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

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

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

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

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

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

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

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

GarageType: Garage location

2Types	More than one type of garage
Attchd	Attached to home
Basment	Basement Garage
BuiltIn	Built-In (Garage part of house - typically has room above garage)
CarPort	Car Port
Detchd	Detached from home
NA	No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin	Finished
RFn	Rough Finished
Unf	Unfinished
NA	No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

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

GarageCond: Garage condition

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

PavedDrive: Paved driveway

Y	Paved
P	Partial Pavement
N	Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

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

Fence: Fence quality

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

MiscFeature: Miscellaneous feature not covered in other categories

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

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

Sale Type: Type of sale

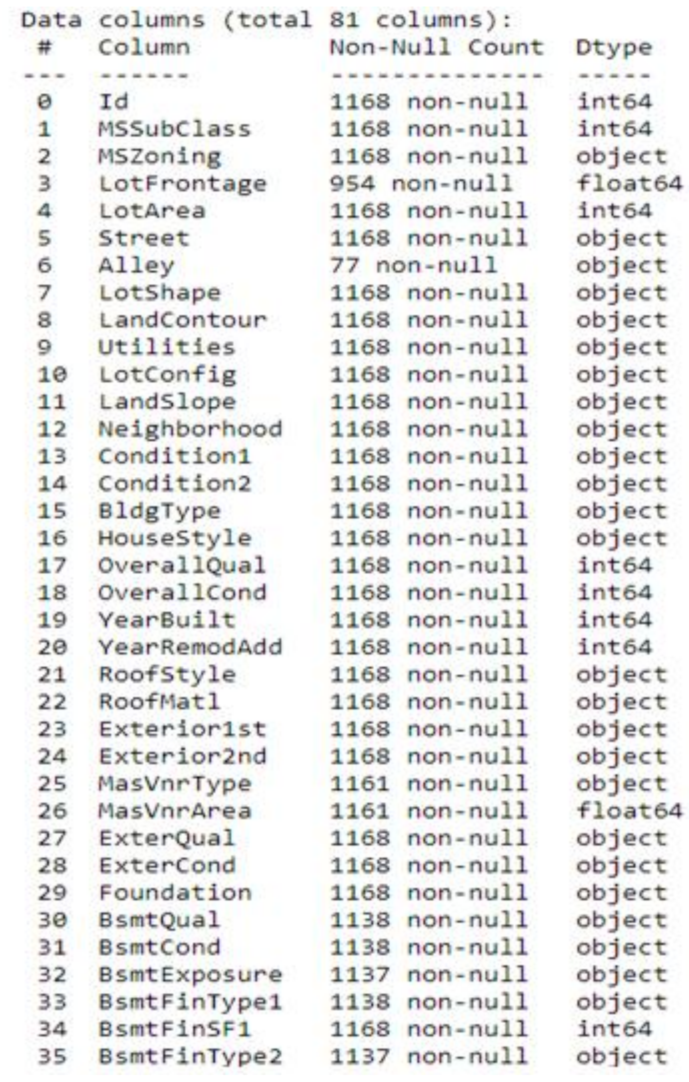
WD	Warranty Deed - Conventional
CWD	Warranty Deed - Cash

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

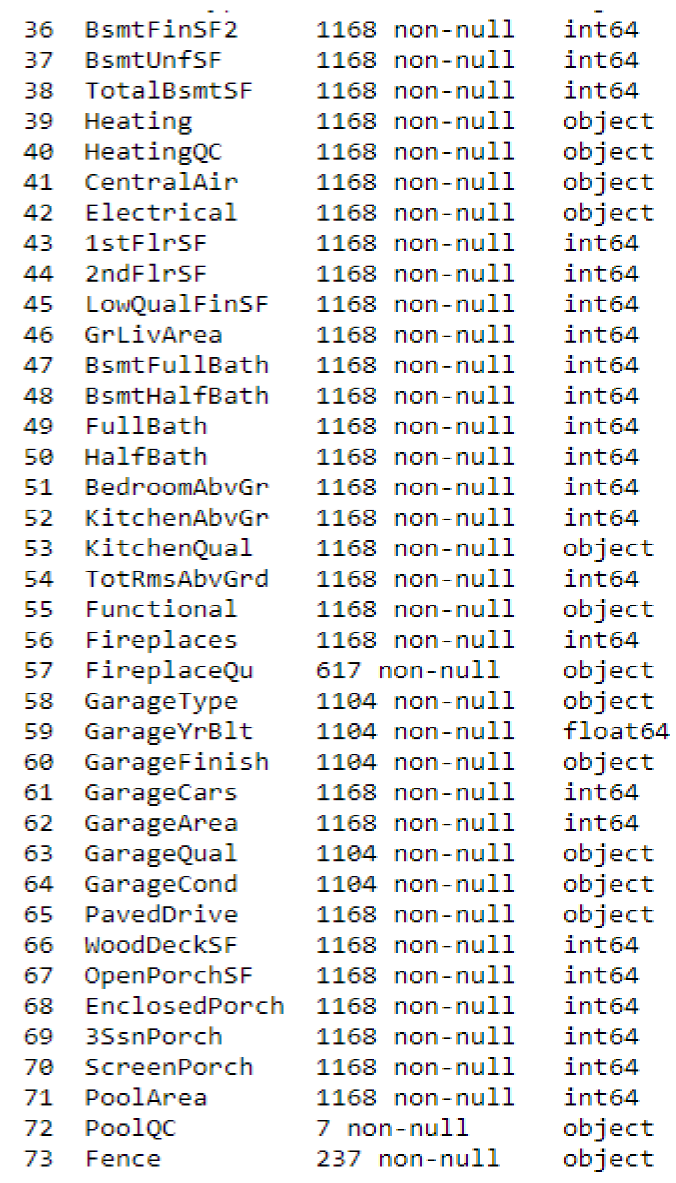
Sale Condition: Condition of sale

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)

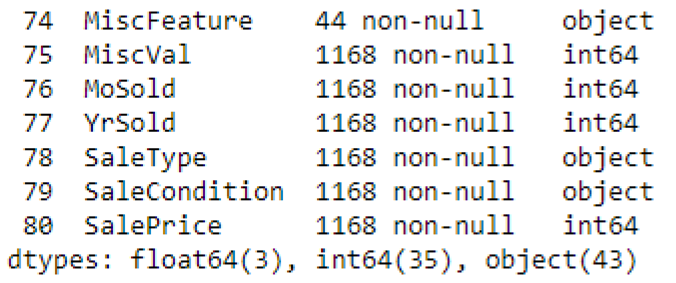
Concise Summary of our Dataset: -



```
Data columns (total 81 columns):
 #   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
```

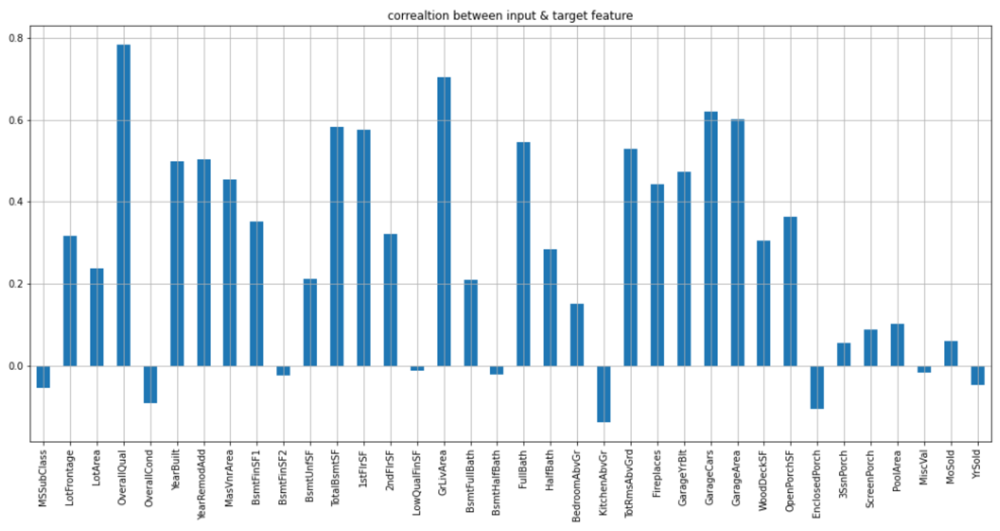


```
 74  MiscFeature    44 non-null     object
 75  MiscVal        1168 non-null   int64
 76  MoSold         1168 non-null   int64
 77  YrSold         1168 non-null   int64
 78  SaleType       1168 non-null   object
 79  SaleCondition  1168 non-null   object
 80  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
```

- ## Data Pre-processing Done

The data pipeline starts with collecting the data and ends with communicating the results & Data pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format. It involves 4 steps: - Cleaning, Formatting, Scaling, and Normalization. While building a machine learning model, if we haven't done any pre-processing like correcting outliers, handling missing values, normalization and scaling of data, or feature engineering, we might end up considering those 1% of results that are false. Some steps performed in this project are: -

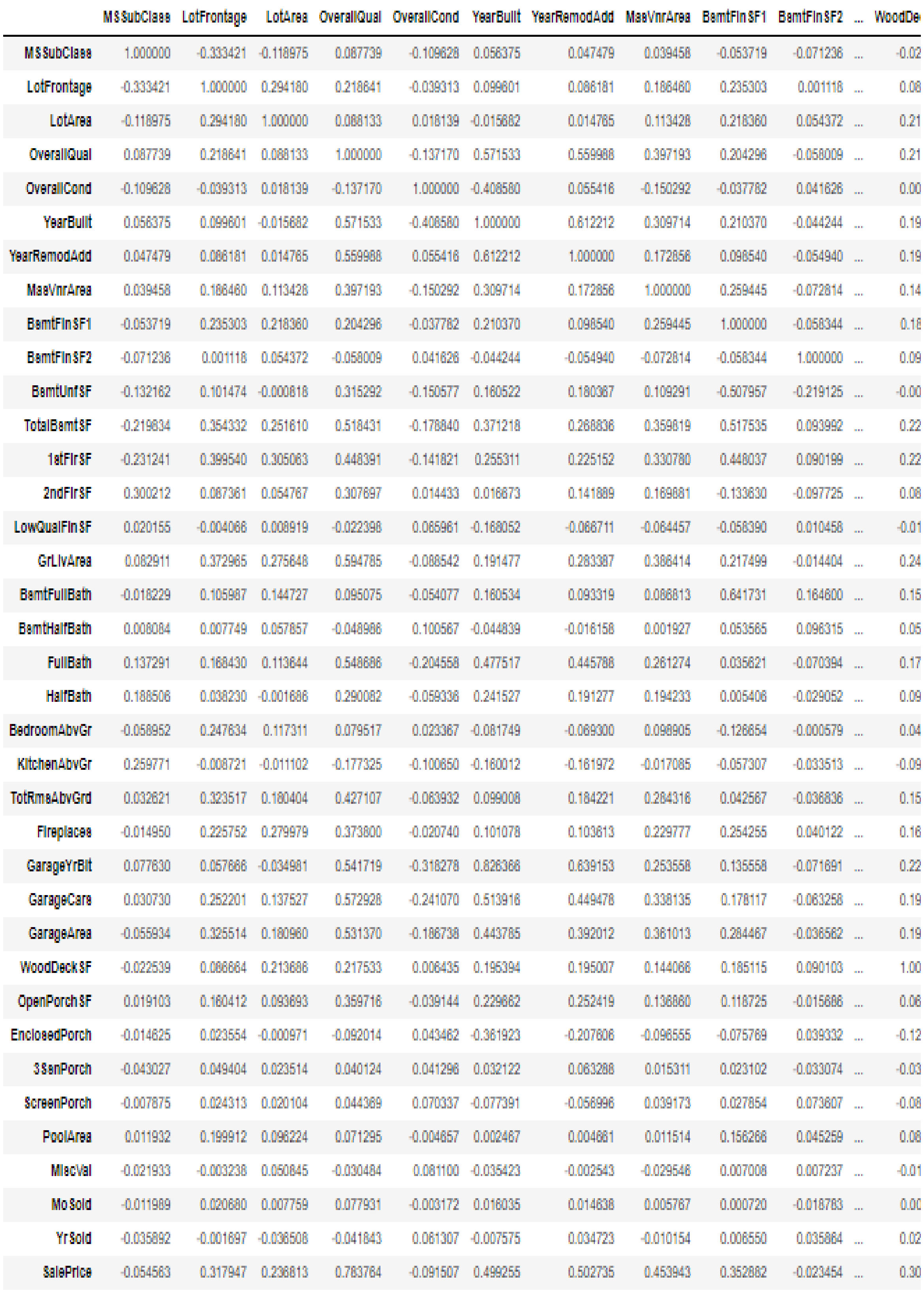
1- Column wise Empty cell analysis done & found missing values in 18 columns.
2- Checked unique values in each column to explore dataset more deeply.
3- Multiple columns had missing values so treated accordingly to fulfil those values.
4- We had seen outliers in some columns so we were trying to remove them using Z scores but data loss was nearly 60.86%. Due to high amount of data loss, we eliminated the Z score implementation process.
5- Skewness was present in dataset. If the skewness is between -0.5 and 0.5, then dataset is fairly symmetrical and symmetrical distribution will have a skewness of zero. So accordingly, we removed skewness using NumPy mathematical functions log, cube root & square root transform.
6- **Feature Encoding and Normalization**: - Features came in a variety of format, e.g., integers, floating numbers, string values, etc. So, we Checked Concise Summary of our Data Frame and we have noticed that 43 columns have object (str or mix str) data type. This was a challenge for us as these features cannot be directly used for training. To prevent regression biases towards certain features, we dropped some of the irrelevant features and make sure that we haven't lose important data. In the end, we also normalize the feature values so that all features are evaluated on the same scale. Also, we did PCA on our dataset for dimensionality reduction.
7- **Data Cleaning**: - We have performed data cleaning separately for category & continuous type variables and tried to avoid data loss. Some steps taken are as: -
a- GarageYrBlt column is continuous type feature and it have information about year of manufacture so we dropped null values from this column.
b- Columns Pool QC, Misc Feature, Alley & Fence have not much effect on our dataset so we have removed these columns.
c- Column ID have identification numbers and have no effect on our dataset so we have removed it from dataset.
d- Column Utilities have only one unique value and have not much effect on our dataset so we have removed it from dataset.
e- After Data cleaning, our dataset has 1104 rows and 75 columns.
8- **Correlation:** - We checked the linear relationship between two variables. After seeing these correlated values, we had found that many columns had multicollinearity is also present between various columns. We have checked correlation between input variables and output variable "Sale Price”: -



Observations:

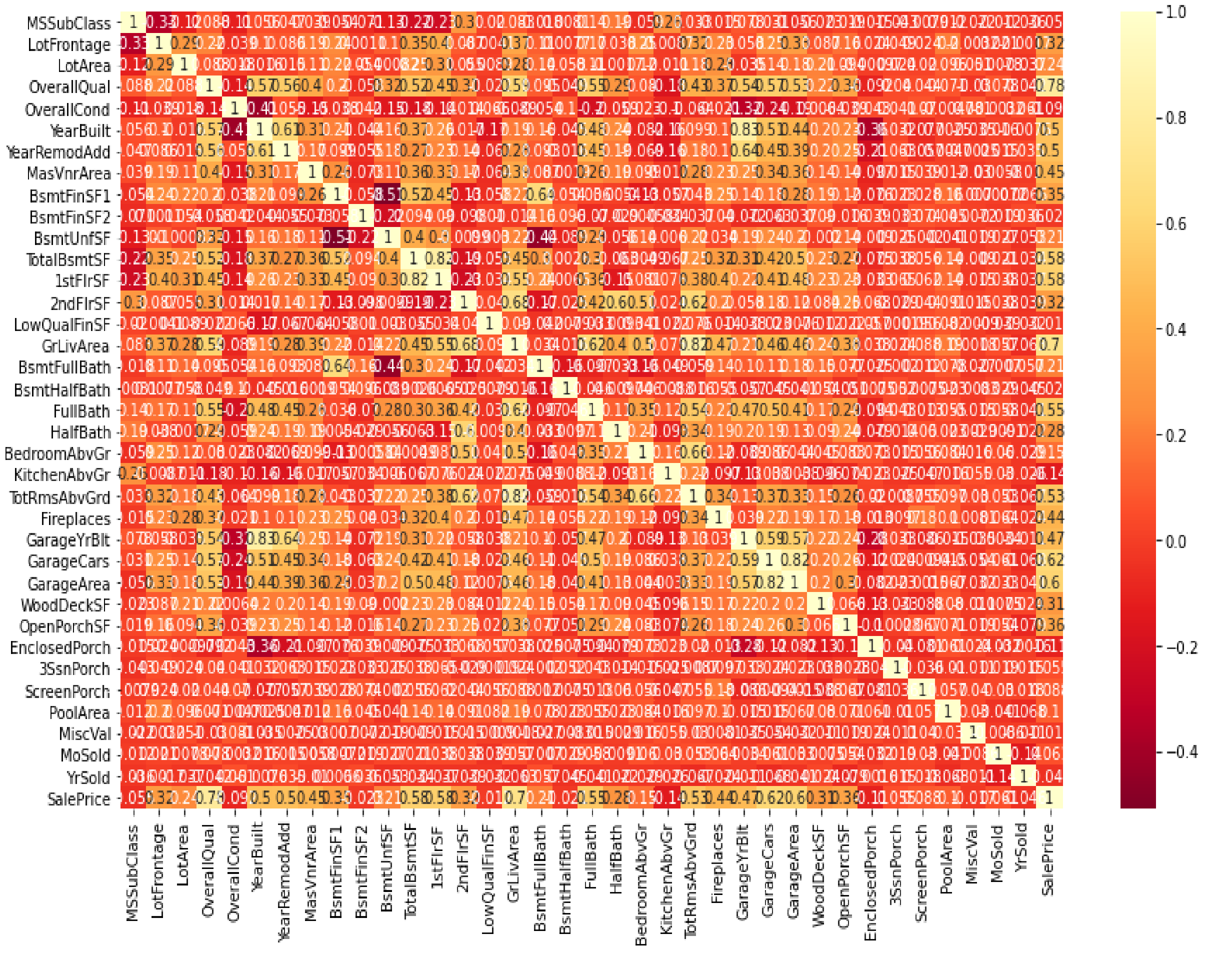
1. OverallQual column is most positively correlated with Sale Price column.
2. KitchenAbvGrd column is most negatively correlated with Sale Price column.

Correlation Matrix: - A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses. A correlation matrix consists of rows and columns that show the variables.



	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDe
MSSubClass	1.000000	-0.333421	-0.118975	0.087739	-0.109628	0.056375	0.047479	0.039458	-0.053719	-0.071236	...	-0.02
LotFrontage	-0.333421	1.000000	0.294180	0.218641	-0.039313	0.099601	0.086181	0.186460	0.235303	0.001118	...	0.08
LotArea	-0.118975	0.294180	1.000000	0.088133	0.018139	-0.015682	0.014765	0.113428	0.218360	0.054372	...	0.21
OverallQual	0.087739	0.218641	0.088133	1.000000	-0.137170	0.571533	0.559988	0.397193	0.204296	-0.058009	...	0.21
OverallCond	-0.109628	-0.039313	0.018139	-0.137170	1.000000	-0.408580	0.055416	-0.150292	-0.037782	0.041626	...	0.00
YearBuilt	0.056375	0.099601	-0.015682	0.571533	-0.408580	1.000000	0.612212	0.309714	0.210370	-0.044244	...	0.19
YearRemodAdd	0.047479	0.086181	0.014765	0.559988	0.055416	0.612212	1.000000	0.172856	0.098540	-0.054940	...	0.19
MasVnrArea	0.039458	0.186460	0.113428	0.397193	-0.150292	0.309714	0.172856	1.000000	0.259445	-0.072814	...	0.14
BsmtFinSF1	-0.053719	0.235303	0.218360	0.204296	-0.037782	0.210370	0.098540	0.259445	1.000000	-0.058344	...	0.18
BsmtFinSF2	-0.071236	0.001118	0.054372	-0.058009	0.041626	-0.044244	-0.054940	-0.072814	-0.058344	1.000000	...	0.09
BsmtUnfSF	-0.132162	0.101474	-0.000818	0.315292	-0.150577	0.160522	0.180367	0.109291	-0.507957	-0.219125	...	-0.00
TotalBsmtSF	-0.219834	0.354332	0.251610	0.518431	-0.178840	0.371218	0.268836	0.359819	0.517535	0.093992	...	0.22
1stFlrSF	-0.231241	0.399540	0.305063	0.448391	-0.141821	0.255311	0.225152	0.330780	0.448037	0.090199	...	0.22
2ndFlrSF	0.300212	0.087361	0.054767	0.307697	0.014433	0.016873	0.141889	0.169881	-0.133630	-0.097725	...	0.08
LowQualFinSF	0.020155	-0.004066	0.008919	-0.022398	0.065961	-0.168052	-0.066711	-0.064457	-0.058390	0.010458	...	-0.01
GrLivArea	0.082911	0.372965	0.275648	0.594785	-0.088542	0.191477	0.283387	0.388414	0.217499	-0.014404	...	0.24
BsmtFullBath	-0.018229	0.105987	0.144727	0.095075	-0.054077	0.160534	0.093319	0.086813	0.641731	0.164600	...	0.15
BsmtHalfBath	0.008084	0.007749	0.057857	-0.048986	0.100567	-0.044839	-0.016158	0.001927	0.053565	0.096315	...	0.05
FullBath	0.137291	0.168430	0.113644	0.548686	-0.204558	0.477517	0.445788	0.261274	0.035621	-0.070394	...	0.17
HalfBath	0.188506	0.038230	-0.001686	0.290082	-0.059336	0.241527	0.191277	0.194233	0.005406	-0.029052	...	0.09
BedroomAbvGr	-0.058952	0.247634	0.117311	0.079517	0.023367	-0.081749	-0.069300	0.098905	-0.126654	-0.000579	...	0.04
KitchenAbvGr	0.259771	-0.008721	-0.011102	-0.177325	-0.100650	-0.160012	-0.161972	-0.017085	-0.057307	-0.033513	...	-0.09
TotRmsAbvGrd	0.032621	0.323517	0.180404	0.427107	-0.063932	0.099008	0.184221	0.284316	0.042567	-0.036836	...	0.15
Fireplaces	-0.014950	0.225752	0.279979	0.373800	-0.020740	0.101078	0.103613	0.229777	0.254255	0.040122	...	0.16
GarageYrBlt	0.077630	0.057666	-0.034981	0.541719	-0.318278	0.826368	0.639153	0.253558	0.135558	-0.071691	...	0.22
GarageCars	0.030730	0.252201	0.137527	0.572928	-0.241070	0.513916	0.449478	0.338135	0.178117	-0.063258	...	0.19
GarageArea	-0.055934	0.325514	0.180960	0.531370	-0.186738	0.443785	0.392012	0.361013	0.284467	-0.036562	...	0.19
WoodDeckSF	-0.022539	0.086664	0.213686	0.217533	0.006435	0.195394	0.195007	0.144066	0.185115	0.090103	...	1.00
OpenPorchSF	0.019103	0.160412	0.093693	0.359716	-0.039144	0.229662	0.252419	0.136860	0.118725	-0.015686	...	0.06
EnclosedPorch	-0.014625	0.023554	-0.000971	-0.092014	0.043462	-0.361923	-0.207606	-0.096555	-0.075769	0.039332	...	-0.12
3SsnPorch	-0.043027	0.049404	0.023514	0.040124	0.041296	0.032122	0.063288	0.015311	0.023102	-0.033074	...	-0.03
ScreenPorch	-0.007875	0.024313	0.020104	0.044369	0.070337	-0.077391	-0.056996	0.039173	0.027854	0.073607	...	-0.08
PoolArea	0.011932	0.199912	0.096224	0.071295	-0.004657	0.002467	0.004681	0.011514	0.156268	0.045259	...	0.08
MiscVal	-0.021933	-0.003236	0.050845	-0.030484	0.081100	-0.035423	-0.002543	-0.029546	0.007008	0.007237	...	-0.01
MoSold	-0.011989	0.020680	0.007759	0.077931	-0.003172	0.016035	0.014638	0.005767	0.000720	-0.018783	...	0.00
YrSold	-0.035892	-0.001697	-0.036508	-0.041843	0.061307	-0.007575	0.034723	-0.010154	0.006550	0.035864	...	0.02
SalePrice	-0.054563	0.317947	0.236813	0.783764	-0.091507	0.499255	0.502735	0.453943	0.352882	-0.023454	...	0.30

Checking correlation using Heatmap with annotations: -



Observations on basis of correlations in our dataset**: -**

a- SalePrice is highly positively correlated with the columns OverallQual, YearBuilt, YearRemodAdd, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, GarageCars & GarageArea.

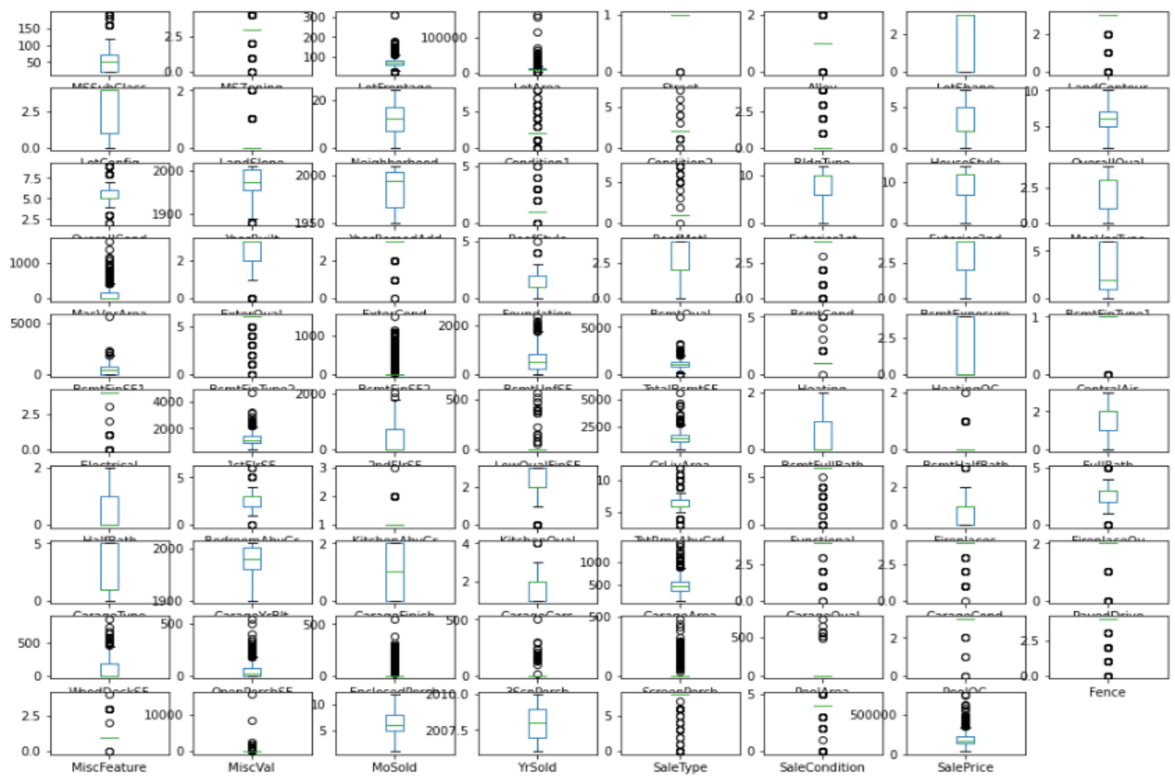
b- SalePrice is negatively correlated with OverallCond, KitchenAbvGr, Encloseporch & YrSold columns.

c- Multicollinearity is present between various columns so using Principal Component Analysis (PCA) will be a great choice.

9- **Plotting Outliers: -** An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Outliers should be investigated carefully. So, we will use graphical techniques Box Plot for identifying outliers.

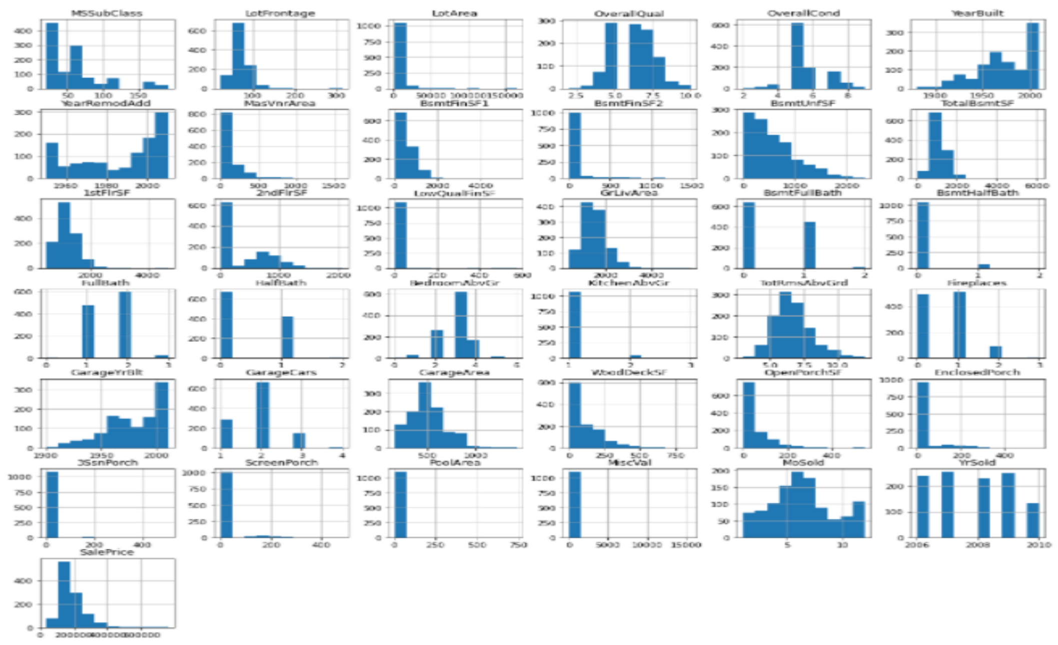
Box Plot- The box plot is a useful graphical display for describing the behaviour of the data in the middle as well as at the ends of the distributions. The box plot uses the median and the lower and upper quartiles (defined as the 25th and 75th percentiles). A box plot is constructed by drawing a box between the upper and lower quartiles with a solid line drawn across the box to locate the median.

Univariate Analysis: - Univariate involves the analysis of a single variable. First, we are going to do univariate analysis using Box Plot method.



Observation: Outliers are present in various columns so; we were trying to remove them using Z scores but data loss was nearly 60.86%. Due to high amount of data loss, we eliminated the Z score implementation process.

10- **Histogram:** - we are creating histograms to get broader idea of the distribution.

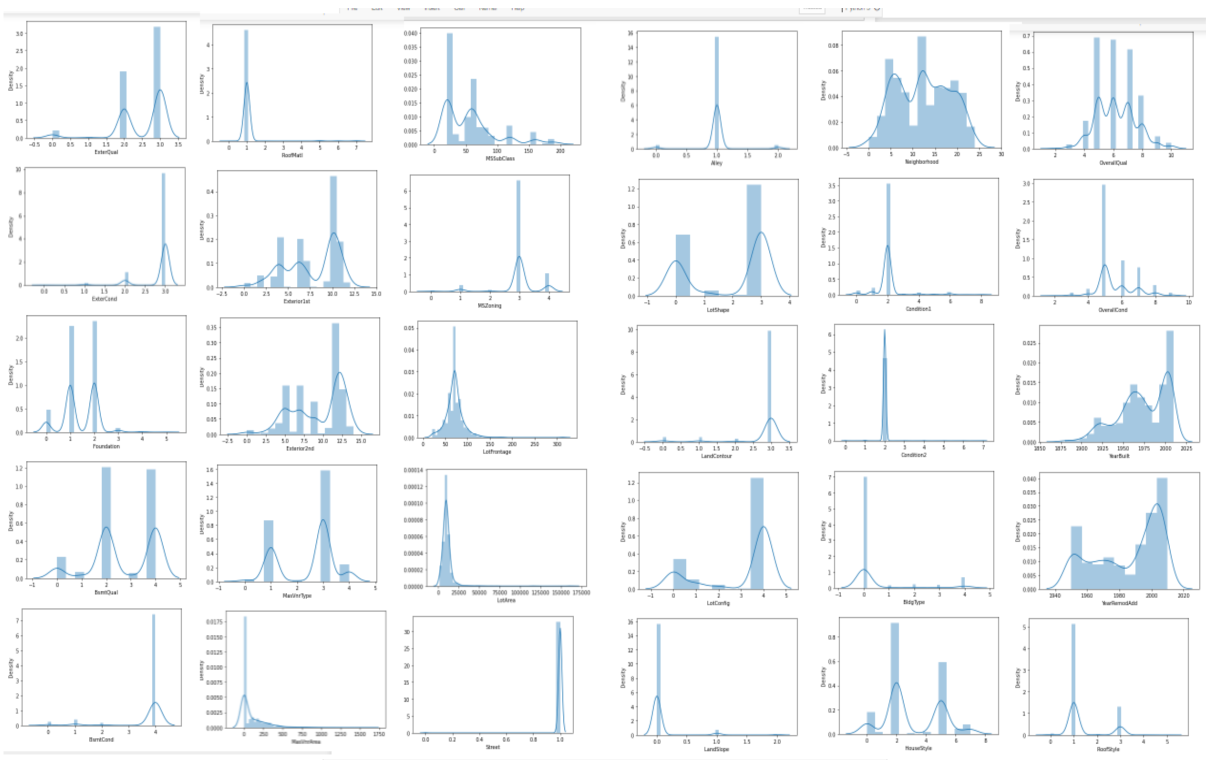


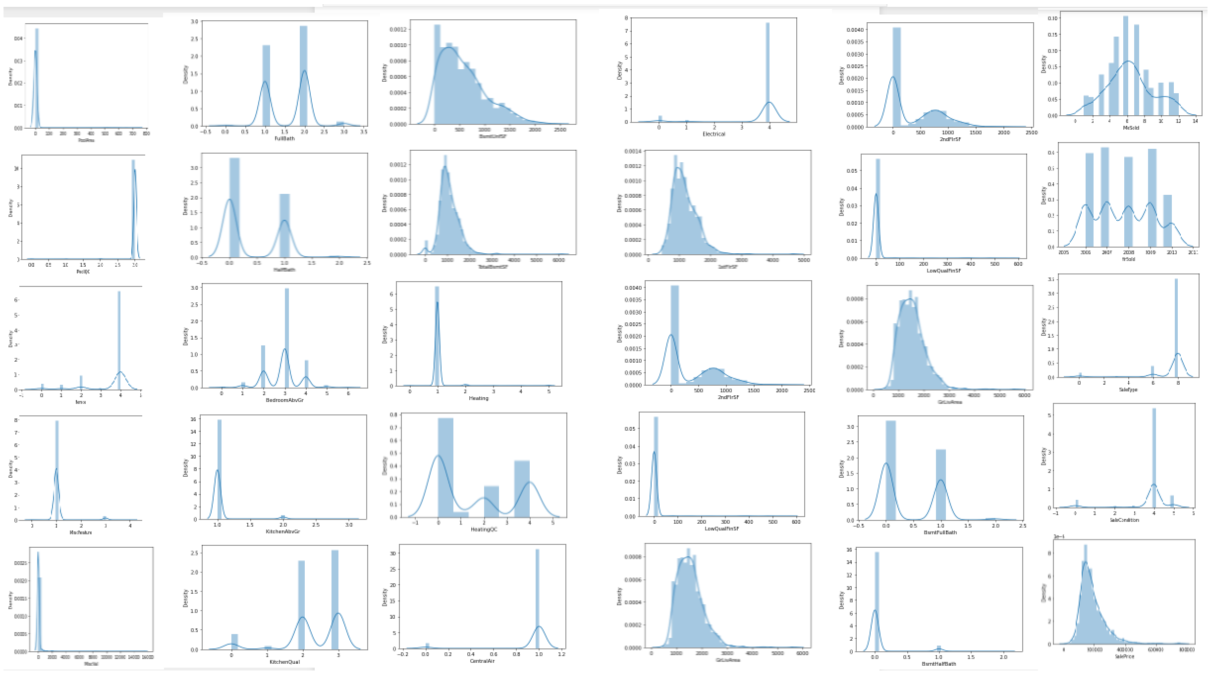
Observation:

Presence of unusual values in above histograms & also distribution is not normal in some columns and these things denote the possibility of potential outliers.

11- Skewness: - Skewness is a measure of asymmetry or distortion of symmetric distribution. It measures the deviation of the given distribution of a random variable from a symmetric distribution, such as normal distribution. A normal distribution is without any skewness, as it is symmetrical on both sides. Hence, a curve is regarded as skewed if it is shifted towards the right or the left. Skewness is of 2 types: - 1- Positive Skewness 2- Negative Skewness.

Distplot to check Distribution of Skewness: - Distplot plots a univariate distribution of observations. The distplot () function combines the matplotlib hist function with the seaborn kde plot () and rug plot () functions. For individual columns we are using Distplot.

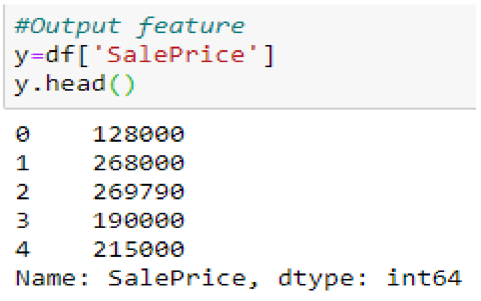




Observation: Skewness is present in various columns. So, we have removed most of skewness using NumPy mathematical functions cube root, log & square root transform.

• Data Inputs- Logic- Output Relationships

Output Feature: - In this project Target Variable = dependent variable = y and shape of our dependent variable is (1104,1). The head of output feature is as: -



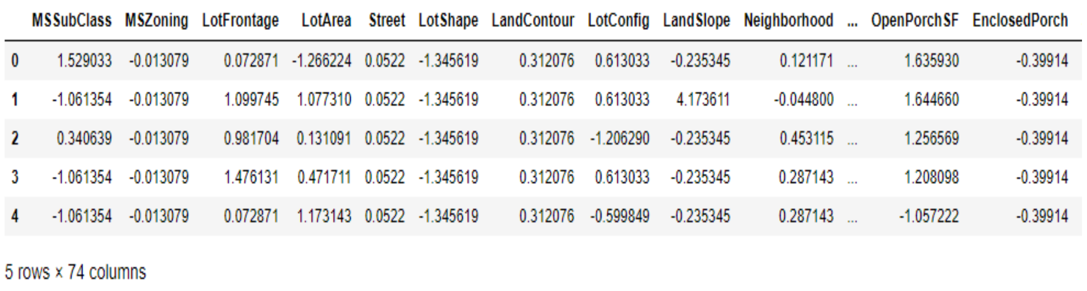
```
#Output feature
y=df['SalePrice']
y.head()
```

```
0     128000
1     268000
2     269790
3     190000
4     215000
Name: SalePrice, dtype: int64
```

Output Variable “Sale Price" is continuous in nature so it is a Regression based problem and We have to predict the price of houses with available independent variables.

Input Feature: - The Independent variable in this project = feature vector = x and shape of Independent   Variable is 1104 rows × 74 columns. After certain transformations and a nalysis input feature is as: -

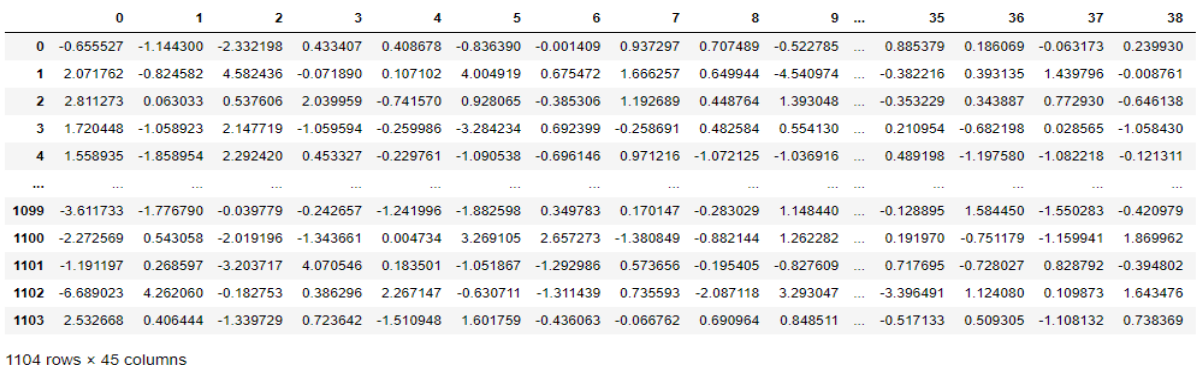
1- Data Pre-processing: - Using Standard Scaler’s fit _ transform method we have tried to bring input features(x) to common scale and modified the input feature as x1. The shap e was remained same and head of input feature is as: -



	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	Neighborhood	...	OpenPorchSF	EnclosedPorch
0	1.529033	-0.013079	0.072871	-1.266224	0.0522	-1.345619	0.312076	0.613033	-0.235345	0.121171	...	1.635930	-0.39914
1	-1.061354	-0.013079	1.099745	1.077310	0.0522	-1.345619	0.312076	0.613033	4.173611	-0.044800	...	1.644660	-0.39914
2	0.340639	-0.013079	0.981704	0.131091	0.0522	-1.345619	0.312076	-1.206290	-0.235345	0.453115	...	1.256569	-0.39914
3	-1.061354	-0.013079	1.476131	0.471711	0.0522	-1.345619	0.312076	0.613033	-0.235345	0.287143	...	1.208098	-0.39914
4	-1.061354	-0.013079	0.072871	1.173143	0.0522	-1.345619	0.312076	-0.599849	-0.235345	0.287143	...	-1.057222	-0.39914

5 rows × 74 columns

2- PCA: - As per Principal component analysis & variance analysis we need 45 features for higher percentage of information and less data loss. So, the new shape of our input feature is 1104 rows and 45 columns. After PCA, input feature is as: -



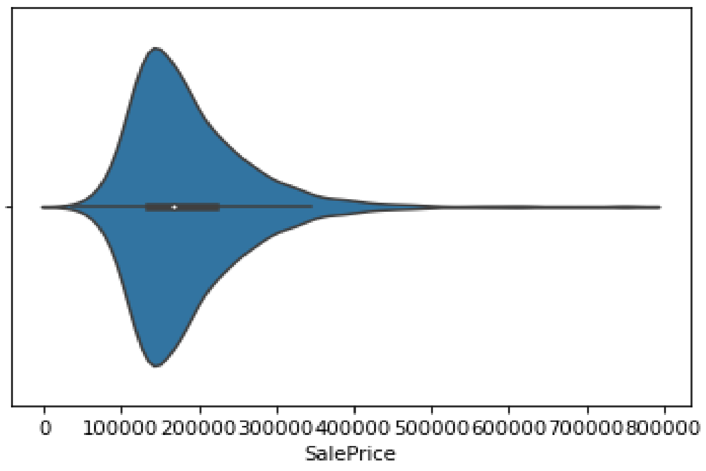
	0	1	2	3	4	5	6	7	8	9	...	35	36	37	38
0	-0.655527	-1.144300	-2.332198	0.433407	0.408678	-0.836390	-0.001409	0.937297	0.707489	-0.522785	...	0.885379	0.186069	-0.063173	0.239930
1	2.071762	-0.824582	4.582436	-0.071890	0.107102	4.004919	0.675472	1.666257	0.649944	-4.540974	...	-0.382216	0.393135	1.439796	-0.008761
2	2.811273	0.063033	0.537606	2.039959	-0.741570	0.928065	-0.385306	1.192689	0.448764	1.393048	...	-0.353229	0.343887	0.772930	-0.646138
3	1.720448	-1.058923	2.147719	-1.059594	-0.259986	-3.284234	0.692399	-0.258691	0.482584	0.554130	...	0.210954	-0.682198	0.028565	-1.058430
4	1.558935	-1.858954	2.292420	0.453327	-0.229761	-1.090538	-0.696146	0.971216	-1.072125	-1.036916	...	0.489198	-1.197580	-1.082218	-0.121311
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1099	-3.611733	-1.776790	-0.039779	-0.242657	-1.241996	-1.882598	0.349783	0.170147	-0.283029	1.148440	...	-0.128895	1.584450	-1.550283	-0.420979
1100	-2.272569	0.543058	-2.019196	-1.343661	0.004734	3.269105	2.657273	-1.380849	-0.882144	1.262282	...	0.191970	-0.751179	-1.159941	1.869962
1101	-1.191197	0.268597	-3.203717	4.070546	0.183501	-1.051867	-1.292986	0.573656	-0.195405	-0.827609	...	0.717695	-0.728027	0.828792	-0.394802
1102	-6.689023	4.262060	-0.182753	0.386296	2.267147	-0.630711	-1.311439	0.735593	-2.087118	3.293047	...	-3.396491	1.124080	0.109873	1.643476
1103	2.532668	0.406444	-1.339729	0.723642	-1.510948	1.601759	-0.436063	-0.066762	0.690964	0.848511	...	-0.517133	0.509305	-1.108132	0.738369

1104 rows × 45 columns

Correlation between input and output features: -In above steps, we have checked already about correlation between input variables and output variable "Sale Price". As per observations “OverallQual” column is most positively correlated with output and “Kitchen AbvGrd” column is most negatively correlated with output feature.

Data Visualization: Data visualization is the graphical representation of information and data. Different visualizations for our input and output features are as: -

Exploring Output Feature: - Violin plot for our target variable & value counts are as: -

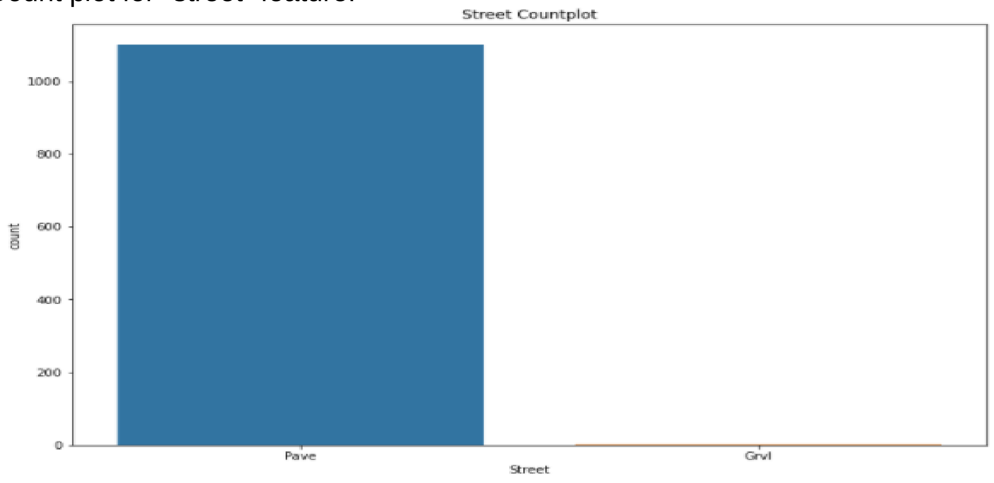


```
140000    18
135000    15
155000    12
160000    10
139000    10
          ..
126175     1
204000     1
186000     1
369900     1
164900     1
Name: SalePrice
```

Observation: Sale Price value is maximum between 140000 and 230000.

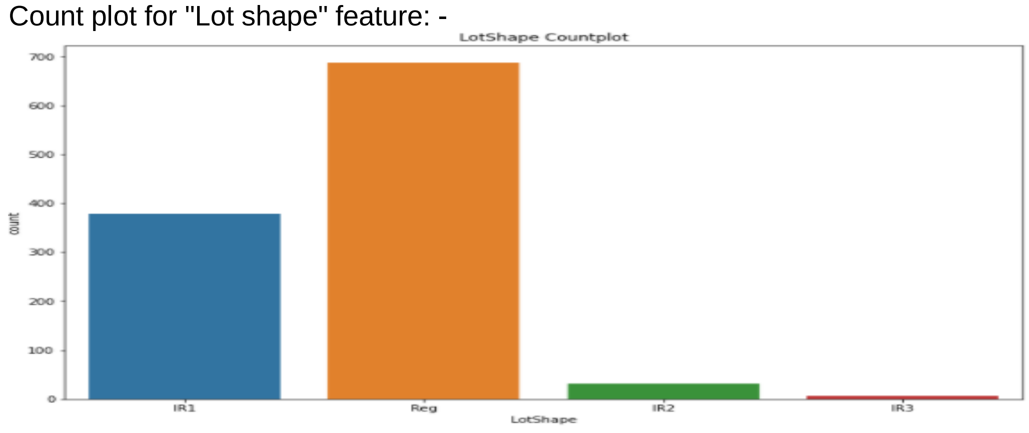
Exploring Input Features: - Plots for different input variables are as: -

1- Count plot for "street" feature: -



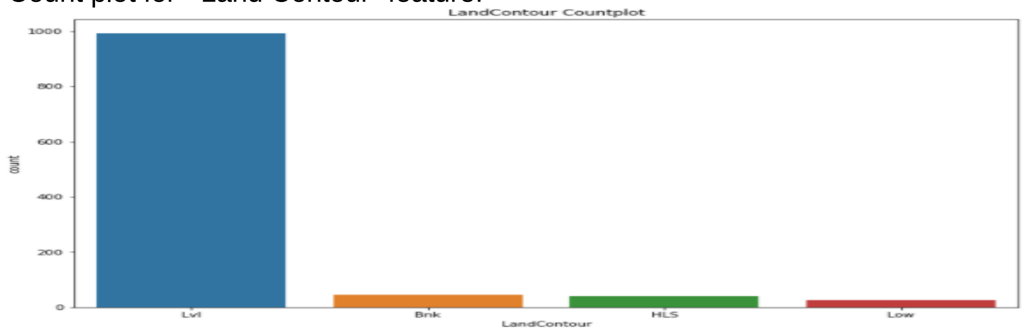
Observation: Pave has 1101 & Grvl has only 3 values in street column.

2- Count plot for "Lot shape" feature: -



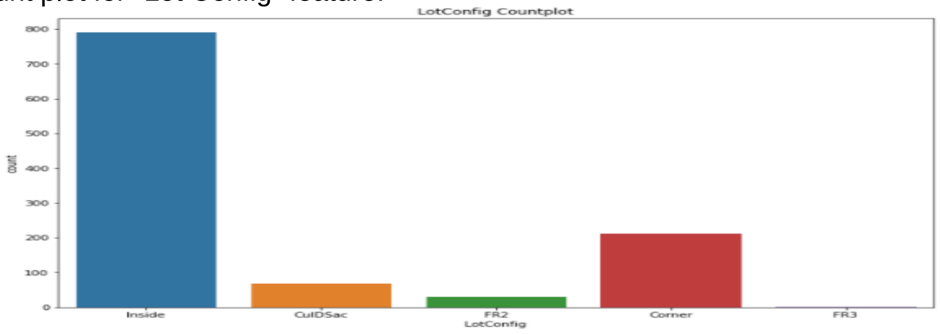
Observation: In Lot shape column, Reg has highest value.

3- Count plot for " Land Contour" feature: -



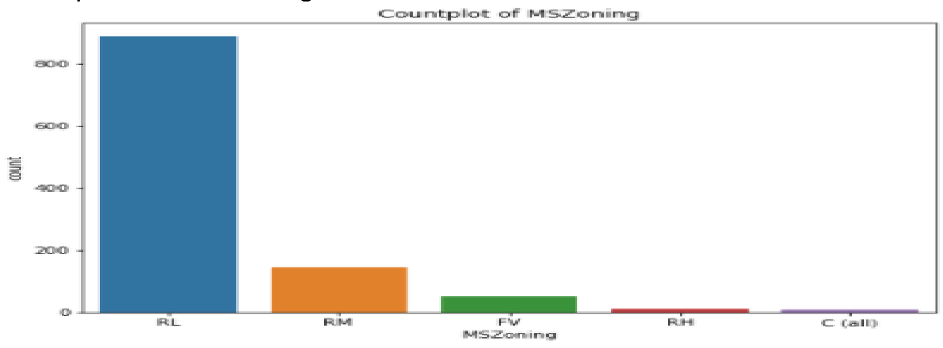
Observation: In Land Contour column, Lvl has highest value.

4- Count plot for "Lot Config" feature: -



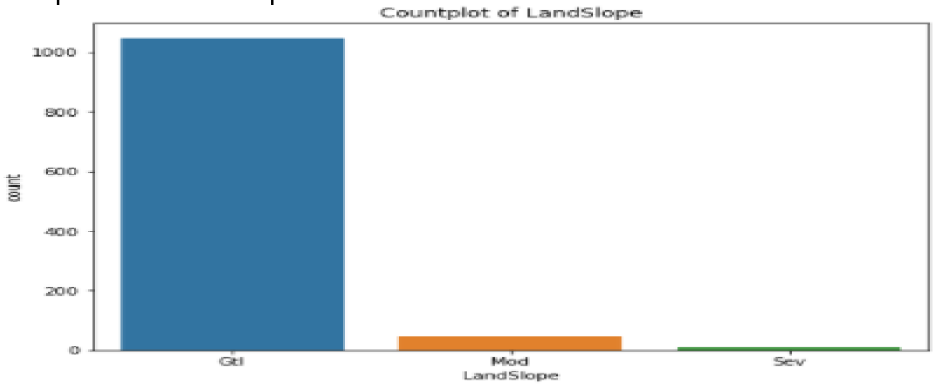
Observation: In Lot Config column, Inside has highest value.

5- Count plot for "MS Zoning" feature: -



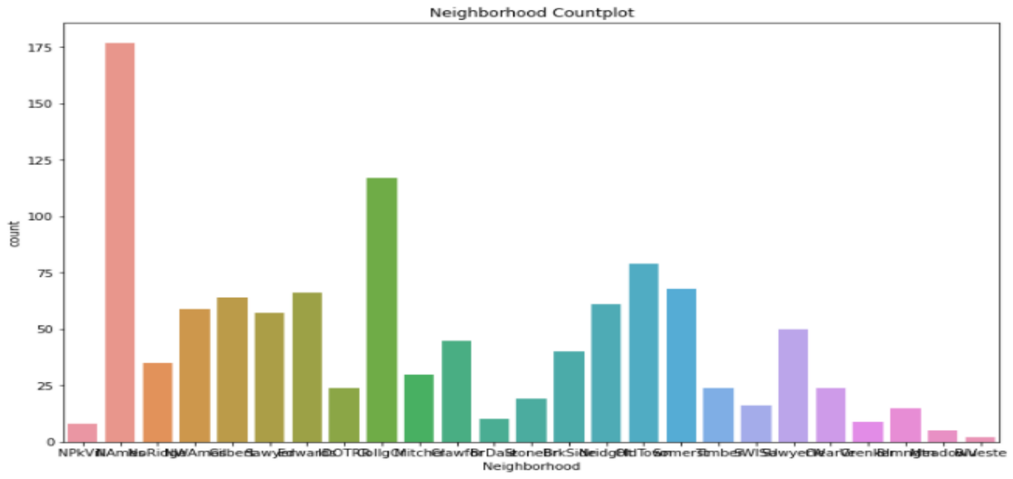
Observation: In MS Zoning column, RL has highest value.

6- Count plot for "Land Slope" feature: -



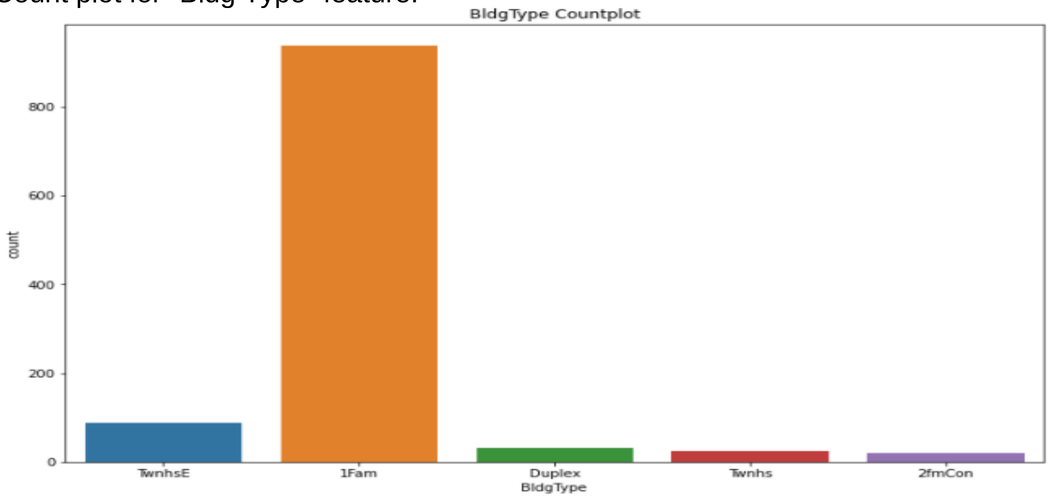
Observation: In Land Slope column, Gtl has highest value.

7- Count plot for "Neighborhood" feature: -



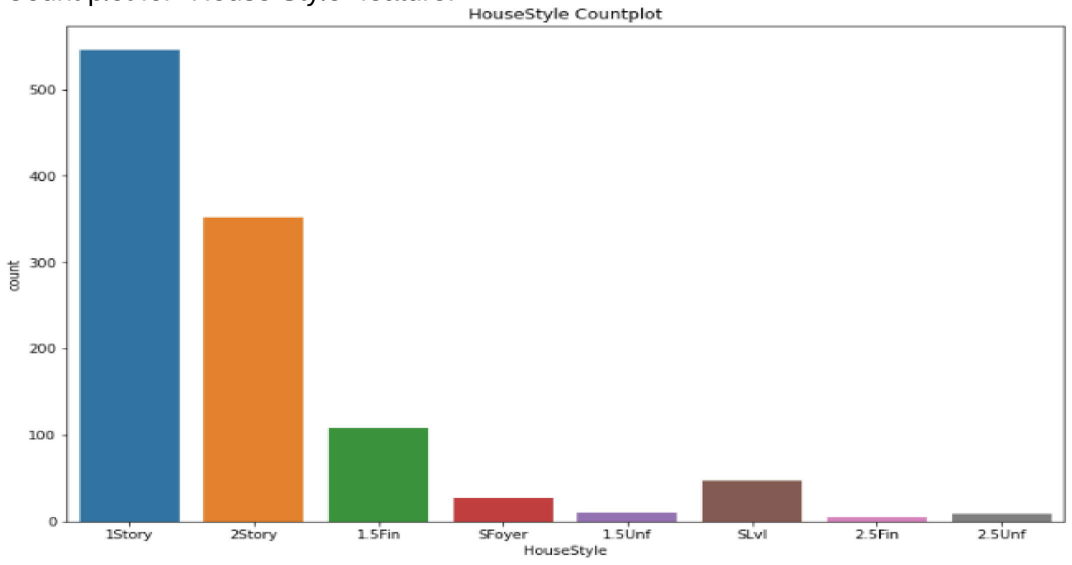
Observation: In Neighborhood column, “NAmes” has highest value.

8- Count plot for "Bldg Type" feature: -



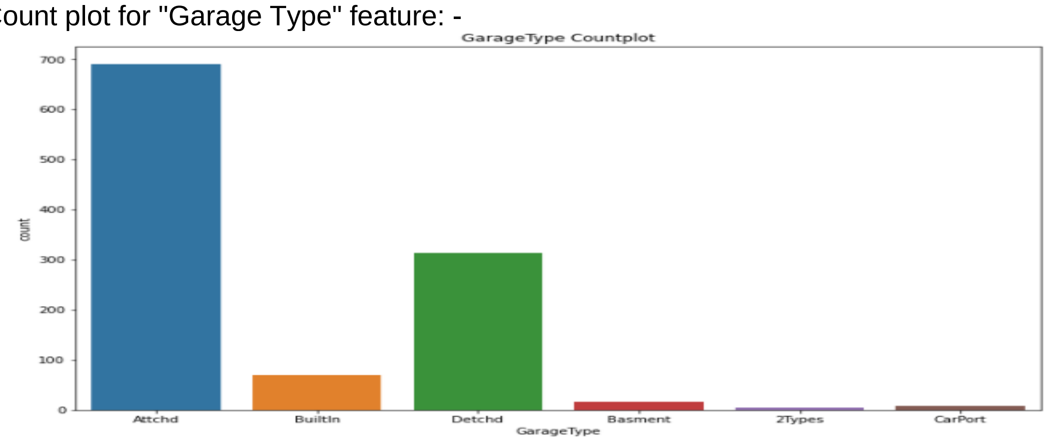
Observation: In Bldg Type column, 1Fam has highest value.

9– Count plot for "House Style" feature: -



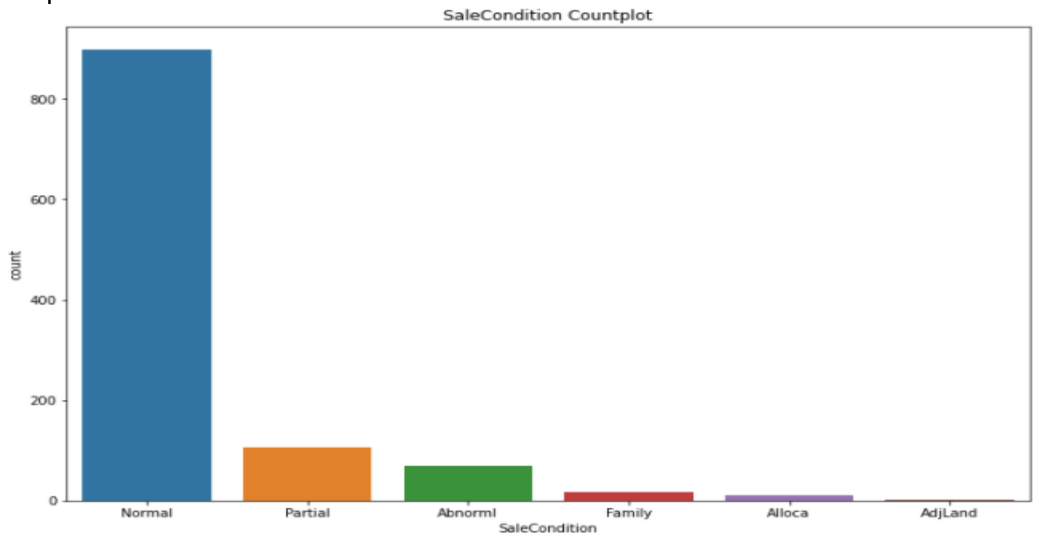
Observation: In House Style column, 1Story has highest value.

10- Count plot for "Garage Type" feature: -



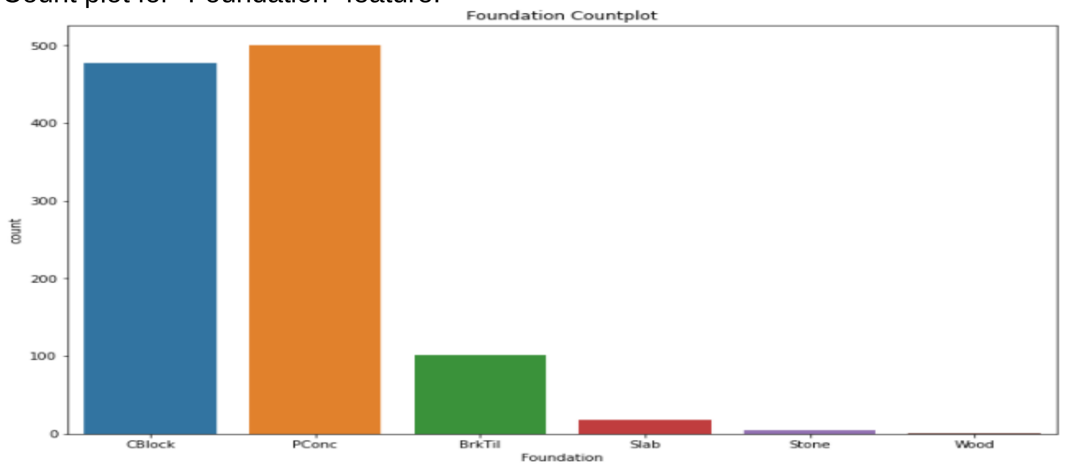
Observation: In Garage Type column, Attchd has highest value.

11- Count plot for " Sale Condition " feature: -



Observation: In Sale Condition column, Normal has highest value.

12- Count plot for "Foundation" feature: -

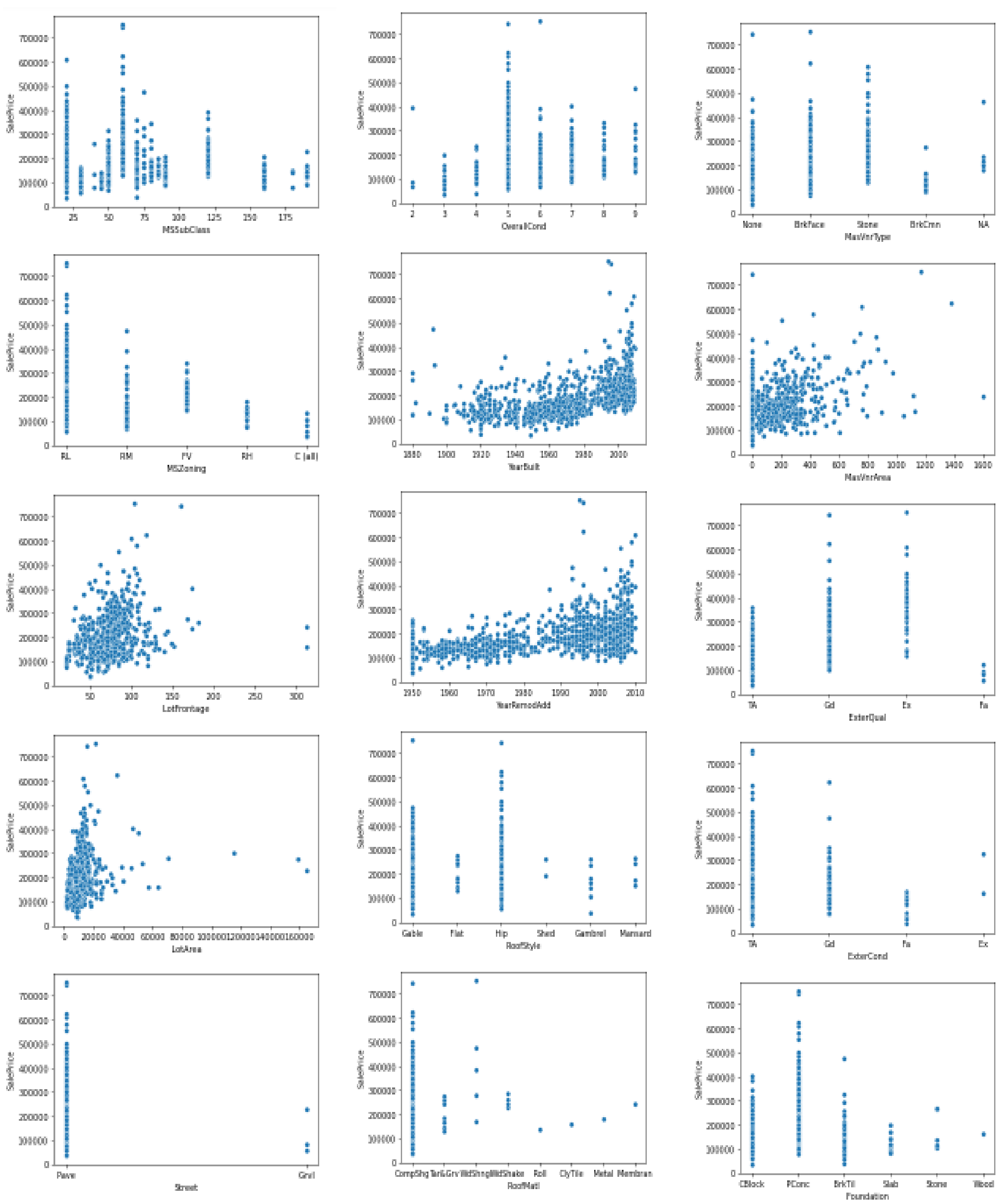


Observation: In Foundation column, PConc has highest value.

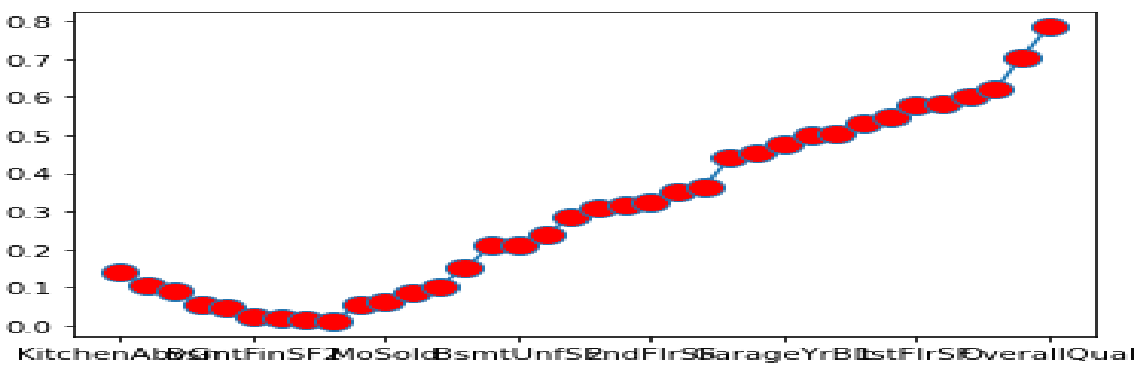
Bivariate Analysis: - It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. We are using scatterplot for this purpose.

Scatter Plot- Scatter plot reveals relationships or association between two variables.

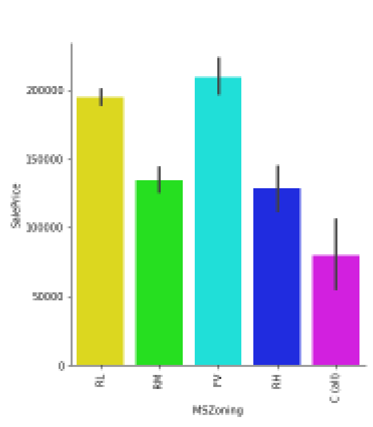
Scatter Plot between Output and Input Features: -

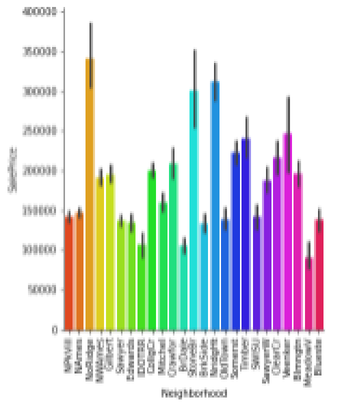


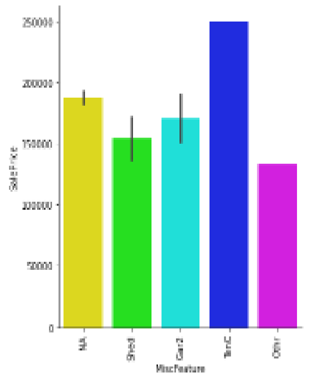
Using Marker Plot for checking highly correlated values with Output variable Sale Price: -



Cat Plot between Output and Input Features (Categorical Type): -



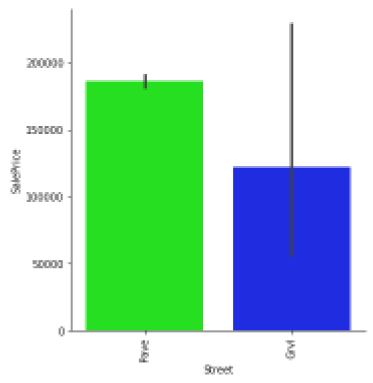


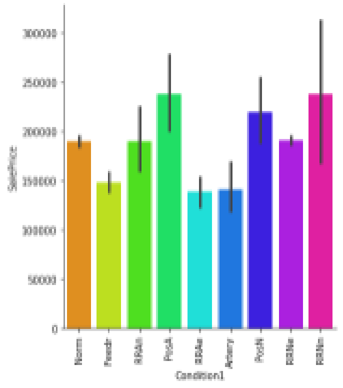


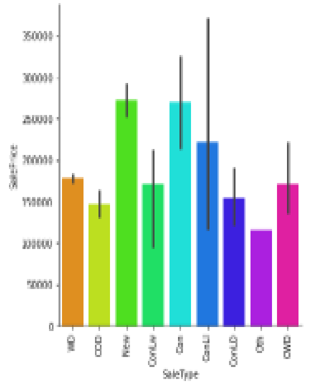
```
<Figure size 1512x1008 with 0 Axes>
```

```
<Figure size 1512x1008 with 0 Axes>
```

```
<Figure size 1512x1008 with 0 Axes>
```



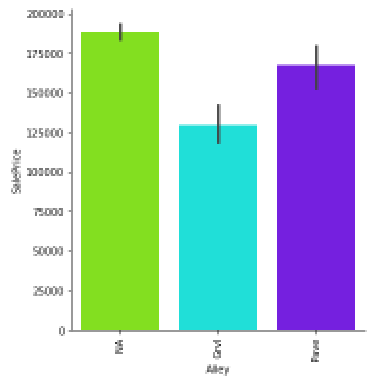


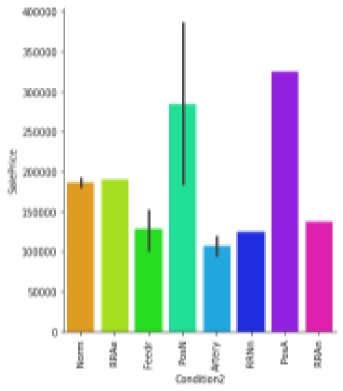


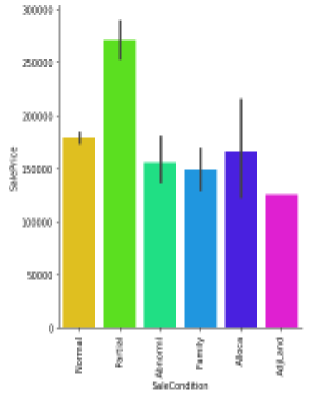
```
<Figure size 1512x1008 with 0 Axes>
```

```
<Figure size 1512x1008 with 0 Axes>
```

```
<Figure size 1512x1008 with 0 Axes>
```

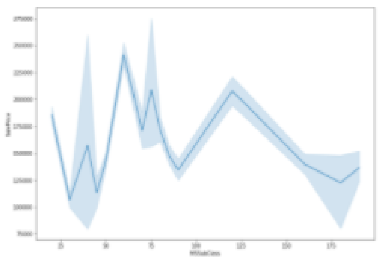


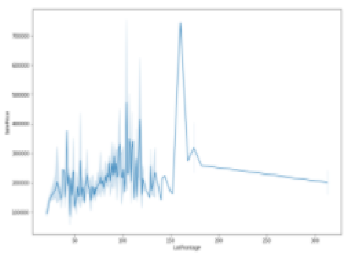


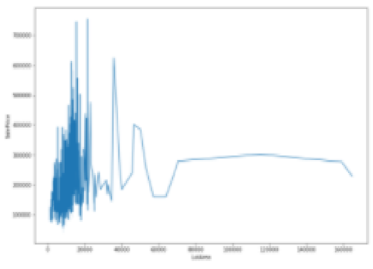


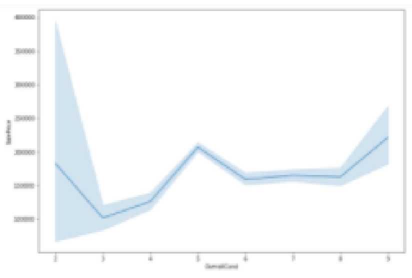
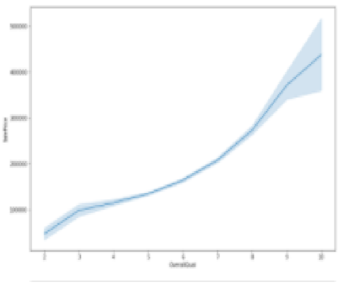
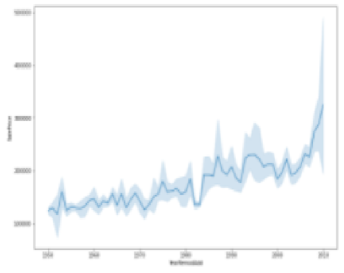
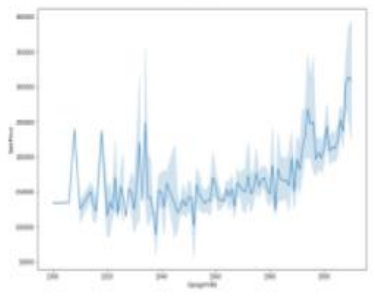
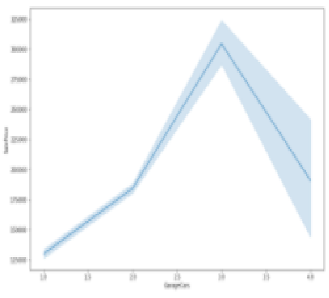
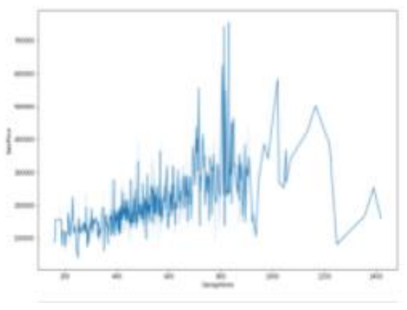
Observation: Sale Price is maximum with FV MS ZOning, Pave Street, IR2 Lot Shape, HLS LandContour, CulDsac Lot Config, Sev Land slope, No Ridge Neighborhood,1Farm BldgType,2.5Fin House Style, Shed Roof Style, Wdshake RoofMatl, Stone MasVnrType,Ex ExterQual, Ex ExterCond, Pconc Foundation, Built in Garage Type & Ex PoolQC values.

Line Plot between Output and Input Features (Continuous Type): -







Observations:

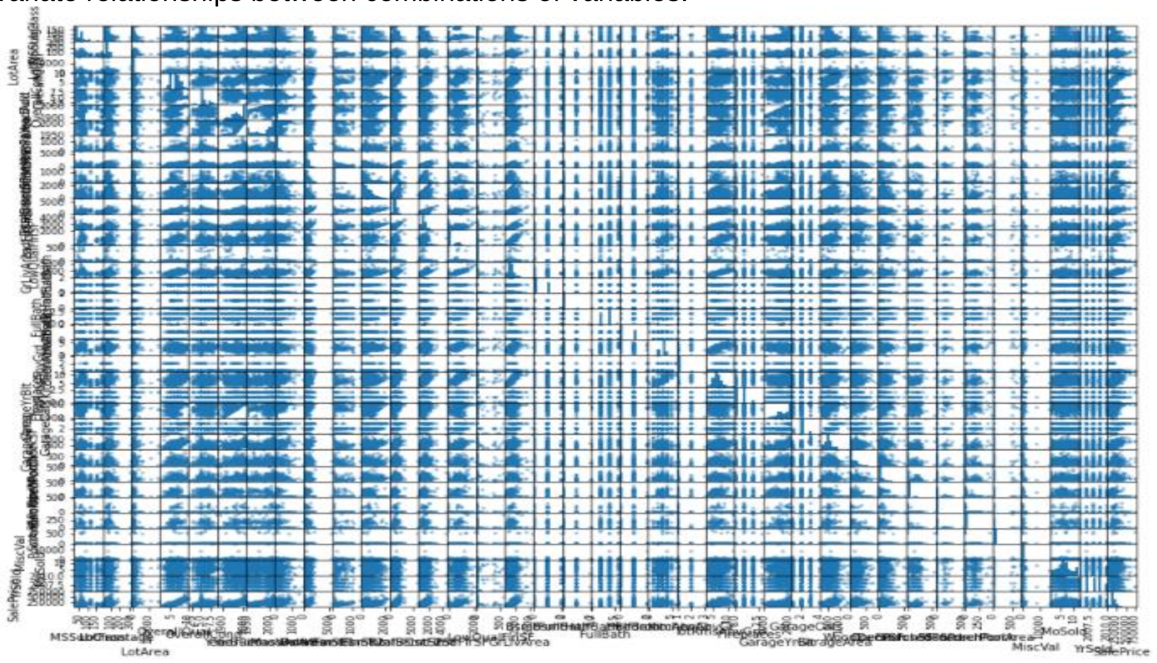
1- If OverallQual is high then Sale price will be maximum.

2- Price of the houses increases with the overall material and finish of the house.

3- Extra amenities such as garage of 3 car capacity, Pool Area, enclosed Porch etc. increase the house price.

Multivariate Analysis: - Multivariate analysis is used to study more complex sets of data. It is a statistical method that measures relationships between two or more response variables.

Scatter plot matrix: -Scatter plot matrix is a grid (or matrix) of scatter plots used to visualize bivariate relationships between combinations of variables.



Observation: Using multivariate analysis, we can look at interactions between variables. Scatter plots of all pair of attributes helps us to spot structured relationship between input variables.

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

1- Multicollinearity is present between various columns so using Principal Component Analysis (PCA) will be a great choice.
2- We had found outliers in dataset so tried to remove them using Z scores but data loss was nearly 60.86%. Due to high amount of data loss, we eliminated the Z score implementation process.
3- By looking into the target variable "Sale Price", we assume that this project is a Regression based problem as target variable is continuous in nature.
4- We have removed irrelevant column which does not have more impact on dataset.

Hardware and Software Requirements and Tools Used

Hardware:4GB RAM, Intel I3 Processor.
System Software: 64Bit O/S Windows 10(x64-based processor)

Software Tools: Software Tools used in this project are as: -

1- Anaconda3 (64-bit)
2- Jupyter Notebook 6.1.4
3- Python 3.8
4- MS-Office 2019 (Excel, Word, Power point)
5- Notepad
6- Google Chrome Web Browser

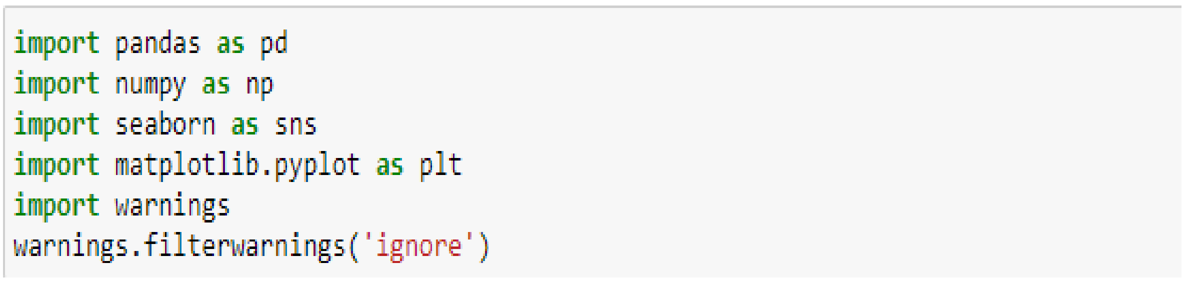
Libraries & Packages used:

We have used Python and Jupyter Notebook to compute the majority of this project. For analysis, visualization, statistics, machine learning & evaluation, we have used these: -

1- Pandas (data analysis)
2- NumPy (matrix computation)
3- Matplotlib (Visualization)
4- Seaborn (visualization)
5- Scikit-Learn (Machine Learning)
6- SciPy (Z-score)
7- Job lib (saving final model)
8- Warnings (filter warnings) & etc.

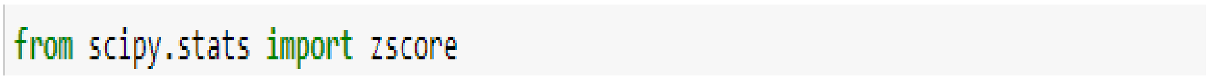
Packages and libraries used in this project are: -

1- For Data Analysis & Visualization: -



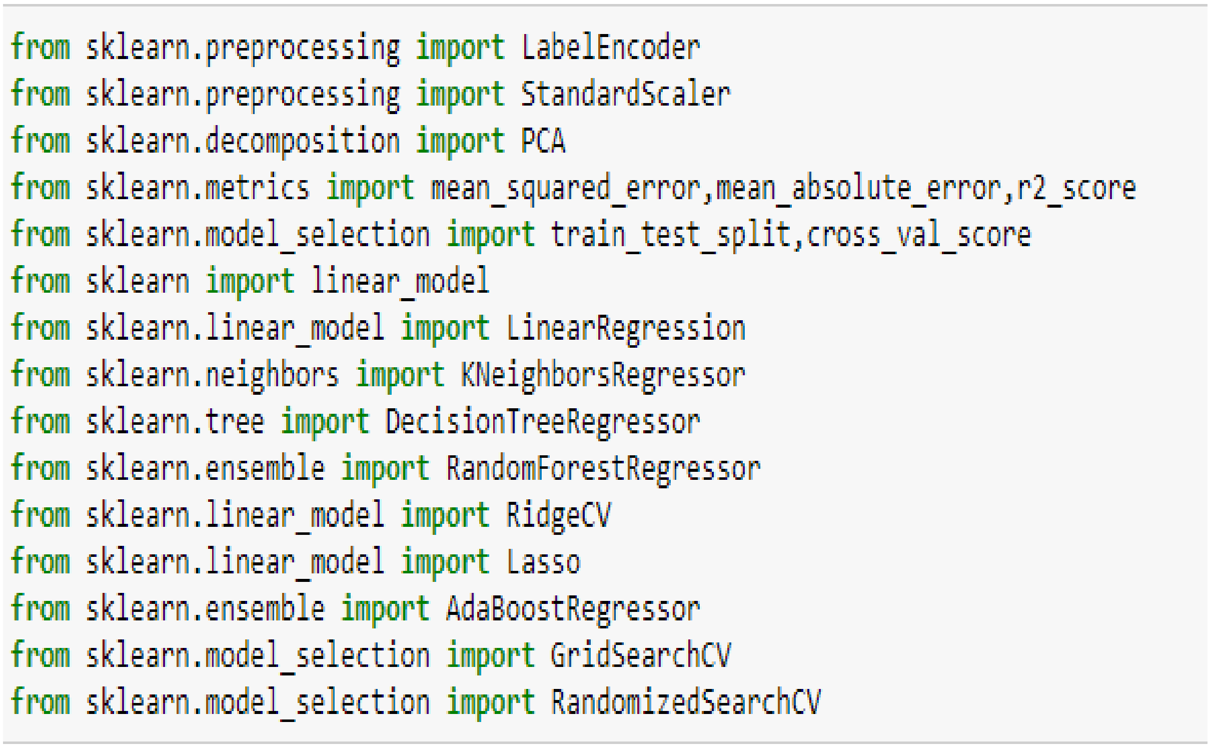
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

2- For Z score: -



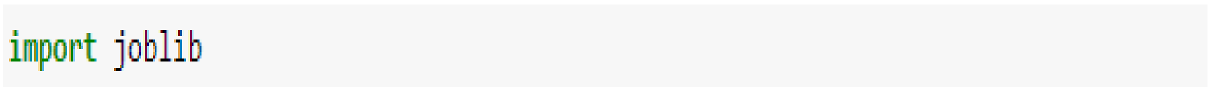
```
from scipy.stats import zscore
```

3- From Scikit-Learn Library: -



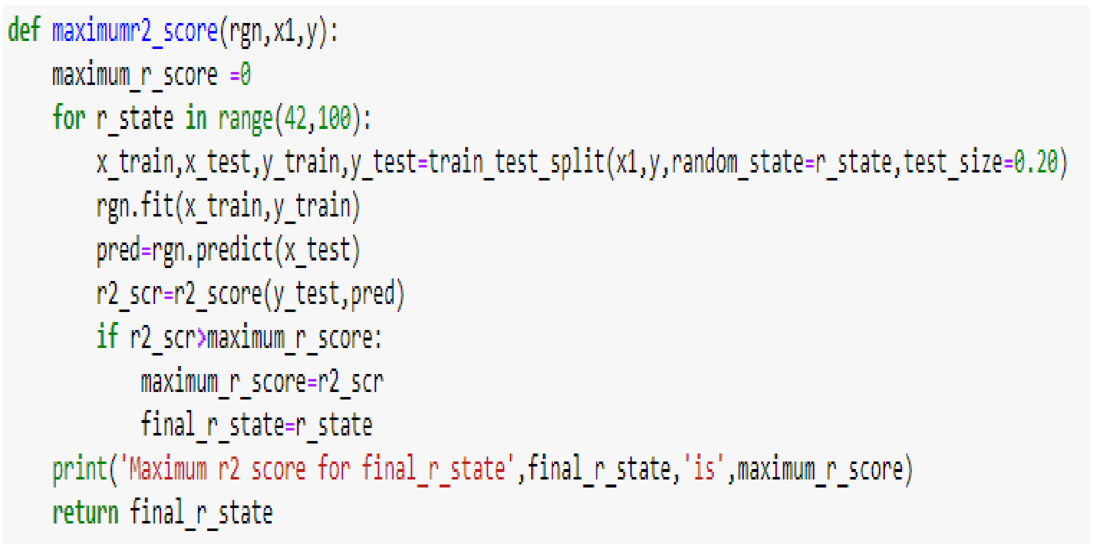
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import Lasso
from sklearn.ensemble import AdaBoostRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

4- For saving the final model: -



```
import joblib
```

Function to calculate maximum R2 score at best random state: -



```
def maximumr2_score(rgn,x1,y):
    maximum_r_score =0
    for r_state in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=r_state,test_size=0.20)
        rgn.fit(x_train,y_train)
        pred=rgn.predict(x_test)
        r2_scr=r2_score(y_test,pred)
        if r2_scr>maximum_r_score:
            maximum_r_score=r2_scr
            final_r_state=r_state
    print('Maximum r2 score for final_r_state',final_r_state,'is',maximum_r_score)
    return final_r_state
```

Model/s Development and Evaluation

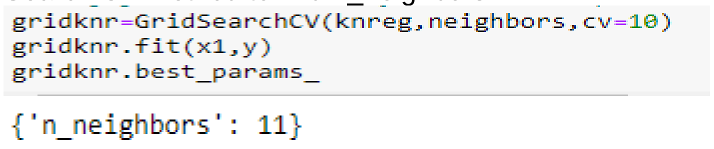
• Identification of possible problem-solving approaches (methods)

1- As per visualizations, Target Variable “Sale Price” is maximum between 140000 and 230000. Also, it is continuous in nature so we will use different regression algorithms to try and find the features that have the best explanation of the target variable.
2- Explored input features and their values using count plot.
3- Checking missing values and handling them properly.
4- Checking Summary Statistics to summarize set of observations as- central Tendency, dispersion, skewness, variance, range, deviation etc.
5- Checking Correlation between target variable and input features using Correlation Matrix & correlation Heatmap using Seaborn.
6- To check distribution & spread of data we used Histogram.
7- Checked Scatter Plots between input & output feature for bivariate analysis.
8- To check highly correlated values with Output variable Sale Price used Marker Plot.
9- Divided input features into category type & continuous type features.
10- Checked Cat plot for category type & Line plot for continuous type features.
11- Used Scatter matrix for multivariate analysis.
12- Removed irrelevant columns which does not have more impact on dataset.
13- Used Label Encoding to encode categorical data in to numerical format.
14- Used Boxplot for summarizing variations & check outliers.
15- Tried to remove outliers using Z scores but data loss was nearly 60.86%. Due to high amount of data loss, we eliminated the Z score implementation process.
16- Divided dataset into input and output sets to explore more briefly.
17- Used Distplot to check distribution of skewness and removed skewness using NumPy mathematical functions log, cube root & square root transformations.
18- Standardization is useful to speed up the learning algorithm and it rescales the features so that they will have the properties of the standard normal distribution with $\mu = 0$ and $\sigma = 1$. We have used Standard Scaler to standardize the data.
19- Multicollinearity is present between various columns so used Principal Component Analysis (PCA) for dimensionality reduction.
20- We will use Coefficient of determination(R2) score as our metric.
21- After Splitting data in to Training & Test Sets, checked scores at best random state after applying different regression algorithms.
22- Used Cross validation to check how accurately a predictive model will perform in practice.
23- To check error of forecasting model, we used Error Metric (MAE, MSE, RMSE).
24- To choose set of optimal hyperparameters for learning algorithm, we did Hyperparameter Tuning. We used Grid Search CV to find estimators/neighbors/alpha for learning algorithms and Randomized search CV to find best parameters for final model.
25- AdaBoost is used as ensemble method to combine several machine learning techniques into one predictive model in order to decrease variance, bias & improve predictions.
26- Compared all algorithms on basis of scores, plots and errors & finalized the best model.
27- Implementing the best model, calculating scores/errors & also checking predicted values.
28- Checked scatterplot between Predicted values and Test values.
29- Saved final model using job lib.
30- Test Dataset sheet checked, handled missing values, removed irrelevant columns, did feature engineering, standardization, principal component & variance analysis.
31- Loading saved model to predict values for Test Dataset.
32- In last, we saved predicted values of test dataset in a CSV file.

• Testing of Identified Approaches (Algorithms)

1- **Linear Regression**: - In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables. Linear Regression fits a linear model with coefficients w = (w1, …) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

2- **k-nearest neighbors**: - K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. We have used Grid Search CV method to find n_neighbors.



```
gridknr=GridSearchCV(knreg,neighbors,cv=10)
gridknr.fit(x1,y)
gridknr.best_params_
```

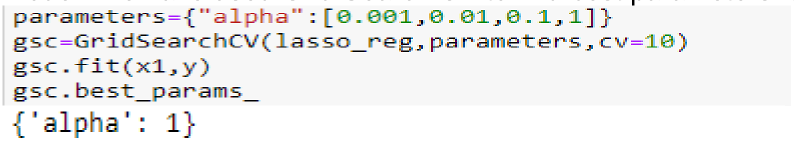
```
{'n_neighbors': 11}
```

3- **Decision Tree Regression**: - Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output which means that the output is not discrete.

4- **Random Forest Regression**: - Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

5- **Ridge Regression**: - Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where independent variables are highly correlated. This method performs L2 regularization. It reduces the model complexity by coefficient shrinkage.

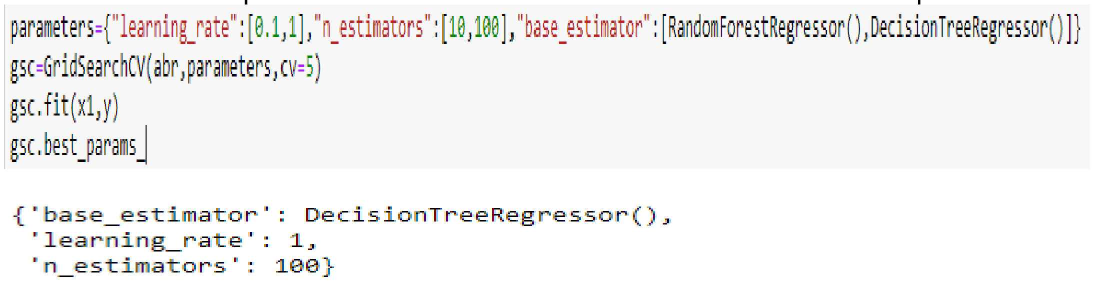
6- **Lasso Regression**: - Lasso (Least Absolute Shrinkage and Selection Operator) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. We have used Grid Search CV to find best parameters for Lasso regression.



```
parameters={"alpha":[0.001,0.01,0.1,1]}
gsc=GridSearchCV(lasso_reg,parameters,cv=10)
gsc.fit(x1,y)
gsc.best_params_
```

```
{'alpha': 1}
```

7- **AdaBoost Regression**: - An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. We have used Grid Search CV to find best parameters.



```
parameters={"learning_rate":[0.1,1],"n_estimators":[10,100],"base_estimator":[RandomForestRegressor(),DecisionTreeRegressor()]}
gsc=GridSearchCV(abr,parameters,cv=5)
gsc.fit(x1,y)
gsc.best_params_
```

```
{'base_estimator': DecisionTreeRegressor(),
 'learning_rate': 1,
 'n_estimators': 100}
```

- ## Run and evaluate selected models

Selected Models: -

1- LinearRegression()

2- KNeighborsRegressor(n_neighbors=11)

3- DecisionTreeRegressor()

4- RandomForestRegressor()

5- RidgeCV()

6- Lasso(alpha=1)

7- AdaBoostRegressor(base_estimator=DecisionTreeRegressor(), learning_rate=1,n_estimators=100)

Linear Regression: -

```
from sklearn.linear_model import LinearRegression
lg=LinearRegression()
r_state=maximumr2_score(lg,x1,y)
print('Mean r2 score for linear Regression is:',cross_val_score(lg,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for linear Regression is',cross_val_score(lg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 42 is 0.8744814651377527
Mean r2 score for linear Regression is: 0.7755191172909399
	Best possible r2score is 1.0
Standard deviation in r2 score for linear Regression is 0.04961727592470395
```

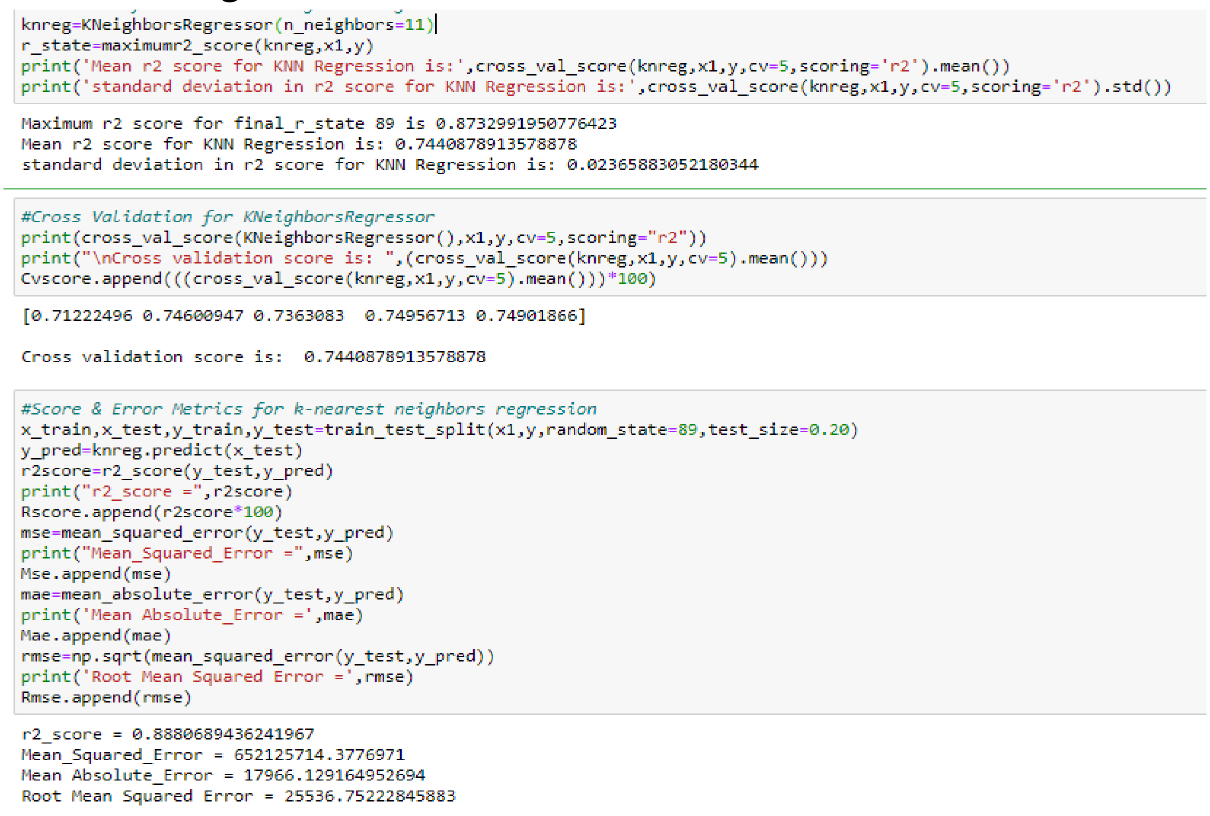
```
#Cross Validation for Linear Regression
from sklearn import linear_model
print(cross_val_score(linear_model.LinearRegression(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(lg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(lg,x1,y,cv=5).mean()))*100)
```

```
[0.80137293 0.77424551 0.68145084 0.82397796 0.79654834]

Cross validation score is:  0.7755191172909399
```

```
#Score & Error Metrics for Linear Regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=42,test_size=0.20)
y_pred=lg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.8817904233889523
Mean_Squared_Error = 862387383.9284116
Mean Absolute_Error = 20733.41598109653
Root Mean Squared Error = 29366.432945259312
```

k-nearest neighbors: -



```
knreg=KNeighborsRegressor(n_neighbors=11)
r_state=maximumr2_score(knreg,x1,y)
print('Mean r2 score for KNN Regression is:',cross_val_score(knreg,x1,y,cv=5,scoring='r2').mean())
print('standard deviation in r2 score for KNN Regression is:',cross_val_score(knreg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 89 is 0.8732991950776423
Mean r2 score for KNN Regression is: 0.7440878913578878
standard deviation in r2 score for KNN Regression is: 0.02365883052180344
```

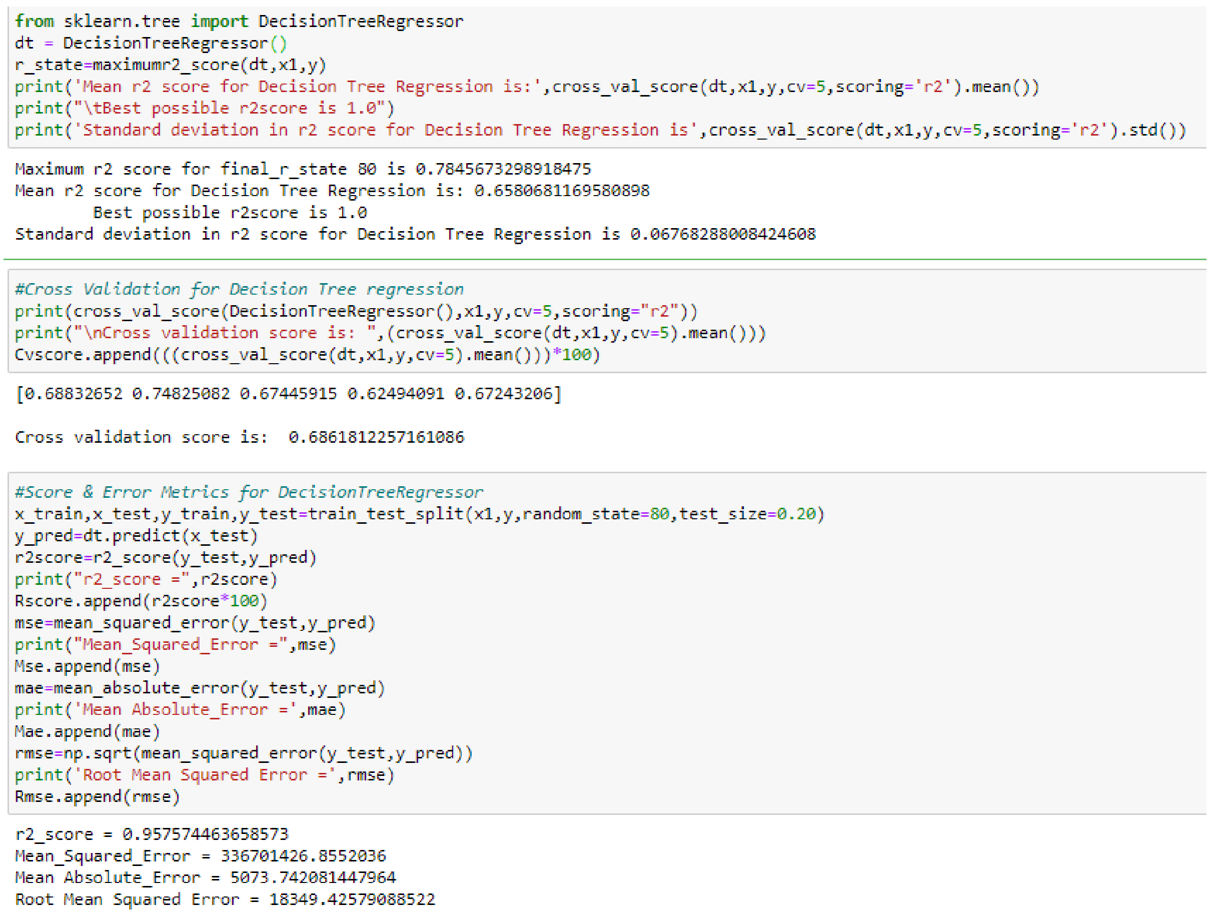
```
#Cross Validation for KNeighborsRegressor
print(cross_val_score(KNeighborsRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(knreg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(knreg,x1,y,cv=5).mean()))*100)
```

```
[0.71222496 0.74600947 0.7363083  0.74956713 0.74901866]

Cross validation score is:  0.7440878913578878
```

```
#Score & Error Metrics for k-nearest neighbors regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=89,test_size=0.20)
y_pred=knreg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.8880689436241967
Mean_Squared_Error = 652125714.3776971
Mean Absolute_Error = 17966.129164952694
Root Mean Squared Error = 25536.75222845883
```

Decision Tree Regression: -



```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
r_state=maximumr2_score(dt,x1,y)
print('Mean r2 score for Decision Tree Regression is:',cross_val_score(dt,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Decision Tree Regression is',cross_val_score(dt,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 80 is 0.7845673298918475
Mean r2 score for Decision Tree Regression is: 0.6580681169580898
        Best possible r2score is 1.0
Standard deviation in r2 score for Decision Tree Regression is 0.06768288008424608
```

```
#Cross Validation for Decision Tree regression
print(cross_val_score(DecisionTreeRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(dt,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(dt,x1,y,cv=5).mean()))*100)
```

```
[0.68832652 0.74825082 0.67445915 0.62494091 0.67243206]

Cross validation score is:  0.6861812257161086
```

```
#Score & Error Metrics for DecisionTreeRegressor
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=80,test_size=0.20)
y_pred=dt.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.957574463658573
Mean_Squared_Error = 336701426.8552036
Mean Absolute_Error = 5073.742081447964
Root Mean Squared Error = 18349.42579088522
```

Random Forest Regression: -

```
: from sklearn.ensemble import RandomForestRegressor
  rf=RandomForestRegressor()
  r_state=maximumr2_score(rf,x1,y)
  print('Mean r2 score for Random Forest Regression is:',cross_val_score(rf,x1,y,cv=5,scoring='r2').mean())
  print("\tBest possible r2score is 1.0")
  print('Standard deviation in r2 score for Random Forest Regression is',cross_val_score(rf,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 60 is 0.89358989662262
Mean r2 score for Random Forest Regression is: 0.8095929219719024
        Best possible r2score is 1.0
Standard deviation in r2 score for Random Forest Regression is 0.03605988591822964
```

```
: #Cross Validation for Random Forest Regression
  print(cross_val_score(RandomForestRegressor(),x1,y,cv=5,scoring="r2"))
  print("\nCross validation score is: ",(cross_val_score(rf,x1,y,cv=5).mean()))
  Cvscore.append(((cross_val_score(rf,x1,y,cv=5).mean()))*100)
```

```
[0.80917722 0.81169123 0.7411683  0.83710728 0.83341146]

Cross validation score is:  0.8177345495666029
```

```
: #Score & Error Metrics for RandomForestRegressor
  x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=60,test_size=0.20)
  y_pred=rf.predict(x_test)
  r2score=r2_score(y_test,y_pred)
  print("r2_score =",r2score)
  Rscore.append(r2score*100)
  mse=mean_squared_error(y_test,y_pred)
  print("Mean_Squared_Error =",mse)
  Mse.append(mse)
  mae=mean_absolute_error(y_test,y_pred)
  print('Mean Absolute_Error =',mae)
  Mae.append(mae)
  rmse=np.sqrt(mean_squared_error(y_test,y_pred))
  print('Root Mean Squared Error =',rmse)
  Rmse.append(rmse)
```

```
r2_score = 0.967810956214687
Mean_Squared_Error = 172545037.51599544
Mean Absolute_Error = 8493.470316742081
Root Mean Squared Error = 13135.639973598372
```

Ridge Regression: -

```
from sklearn.linear_model import RidgeCV
rg=RidgeCV()
r_state=maximumr2_score(rg,x1,y)
print('Mean r2 score for Ridge Regression is:',cross_val_score(rg,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Ridge Regression is',cross_val_score(rg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 42 is 0.8741460084475092
Mean r2 score for Ridge Regression is: 0.7759940601763123
        Best possible r2score is 1.0
Standard deviation in r2 score for Ridge Regression is 0.04991724336570281
```

```
#Cross Validation for Ridge Regression
print(cross_val_score(linear_model.RidgeCV(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(rg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(rg,x1,y,cv=5).mean()))*100)
```

```
[0.80135603 0.77469479 0.68129183 0.82454739 0.79808026]

Cross validation score is:  0.7759940601763123
```

```
#Score & Error Metrics for Ridge Regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=42,test_size=0.20)
y_pred=rg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.8814240435595796
Mean_Squared_Error = 865060275.174918
Mean Absolute_Error = 20730.41375320355
Root Mean Squared Error = 29411.907030570426
```

Lasso regression: -

```
lasso_reg=Lasso(alpha=1)
r_state=maximumr2_score(lasso_reg,x1,y)
print('Mean r2 score for Lasso Regression is',cross_val_score(lasso_reg,x1,y,cv=5,scoring='r2').mean())
print('standard deviation in r2 score for Lasso Regrssion is',cross_val_score(lasso_reg,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 42 is 0.8744746876710778
Mean r2 score for Lasso Regression is 0.7755261349189034
standard deviation in r2 score for Lasso Regrssion is 0.049625518108642866
```

```
#Cross Validation for Lasso regression
print(cross_val_score(linear_model.Lasso(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(lasso_reg,x1,y,cv=5).mean()))
Cvscore.append(((cross_val_score(lasso_reg,x1,y,cv=5).mean()))*100)
```

```
[0.8013671  0.77425451 0.68144055 0.82399074 0.79657778]

Cross validation score is:  0.7755261349189034
```

```
#Score & Error Metrics for Lasso regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=42,test_size=0.20)
y_pred=lasso_reg.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
```

```
r2_score = 0.8817820575645211
Mean_Squared_Error = 862448416.0516328
Mean Absolute_Error = 20733.943336873428
Root Mean Squared Error = 29367.472074586753
```

ADA Boost Regression: -

```
abr=AdaBoostRegressor(base_estimator=DecisionTreeRegressor(),learning_rate=1,n_estimators=100)
maxr2_score(abr,x1,y)
print('Mean r2 score for ADABoost Regression is:',cross_val_score(abr,x1,y,cv=5,scoring='r2').mean())
print('Standard Deviation in r2 score for ADABoost Regression is:',cross_val_score(abr,x1,y,cv=5,scoring='r2').std())
```

```
Maximum r2 score for final_r_state 80 is 0.8793129241645792
Mean r2 score for ADABoost Regression is: 0.8156839063438648
Standard Deviation in r2 score for ADABoost Regression is: 0.03525724891919767
```

```
#Cross Validation for ADABoost regression
print(cross_val_score(AdaBoostRegressor(),x1,y,cv=5,scoring="r2"))
print("\nCross validation score is: ",(cross_val_score(abr,x1,y,cv=5,scoring="r2").mean()))
Cvscore.append(((cross_val_score(abr,x1,y,cv=5,scoring="r2").mean()))*100)
```

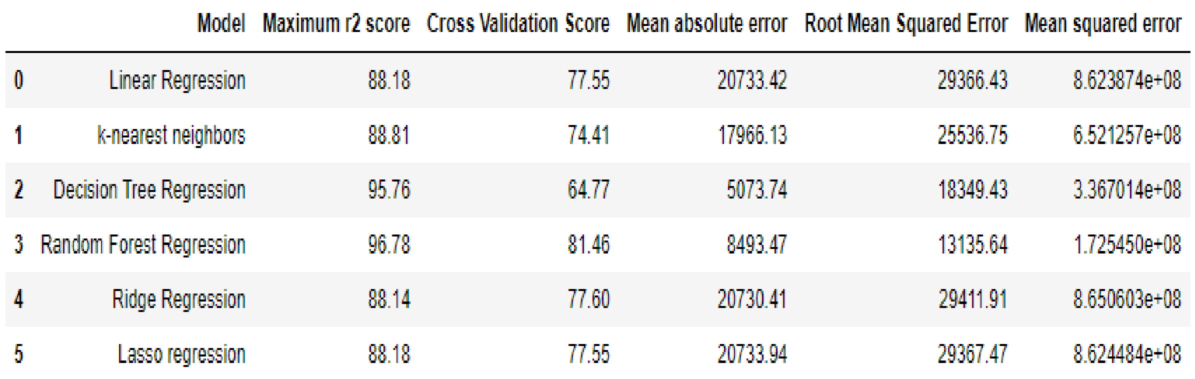
```
[0.78033864 0.77370056 0.70897708 0.79957852 0.69956477]

Cross validation score is:  0.8168332015654742
```

```
#Score & Error Metrics for ADABoost regression
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=80,test_size=0.20)
y_pred=abr.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score)
Rscore.append(r2score*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
Mse.append(mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
Mae.append(mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
Rmse.append(rmse)
model.append('ADABoost Regression')
```

```
r2_score = 0.8793129241645792
Mean_Squared_Error = 957807823.800905
Mean Absolute_Error = 19126.072398190045
Root Mean Squared Error = 30948.470459796637
```

- **Key Metrics for success in solving problem under consideration**: -

Before using ensemble methods, all score and calculations are as: -



	Model	Maximum r2 score	Cross Validation Score	Mean absolute error	Root Mean Squared Error	Mean squared error
0	Linear Regression	88.18	77.55	20733.42	29366.43	8.623874e+08
1	k-nearest neighbors	88.81	74.41	17966.13	25536.75	6.521257e+08
2	Decision Tree Regression	95.76	64.77	5073.74	18349.43	3.367014e+08
3	Random Forest Regression	96.78	81.46	8493.47	13135.64	1.725450e+08
4	Ridge Regression	88.14	77.60	20730.41	29411.91	8.650603e+08
5	Lasso regression	88.18	77.55	20733.94	29367.47	8.624484e+08

Evaluation Metrics used: -

1- **r2 score**: - Coefficient of determination, denoted R2 or r2, is the proportion variation in the dependent variable that is predictable from the independent variables. It is a regression score function and best possible score is 1.0. We have calculated maximum & mean r2 score for models and will select model with best r2 score.

2-**Cross validation score**: - It is a score evaluated by cross-validation. When we want to estimate how accurately a predictive model will perform in practice and our goal is prediction then we use cross validation. cross validation score returns score of test fold where cross validation predicts returns predicted y values for the test fold. We will select model with best cross validation score.

Error Metrics used: - We will select model with minimum errors.

1-Mean Absolute Error (MAE)- MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. MAE score is calculated as the average of the absolute error values. MAE can be calculated as follows:

MAE = 1 / N * sum for i to N abs(y_i – yhat_i)

2-Mean Squared Error (MSE): - MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset. MSE can be calculated as follows:

MSE = 1 / N * sum for i to N (y_i – yhat_i)^2

3-Root Mean Squared Error (RMSE): - RMSE is an extension of the mean squared error. It's the square root of the average of squared differences between prediction and actual observation. RMSE can be calculated as follows:

RMSE = sqrt(1 / N * sum for i to N (y_i – yhat_i)^2)

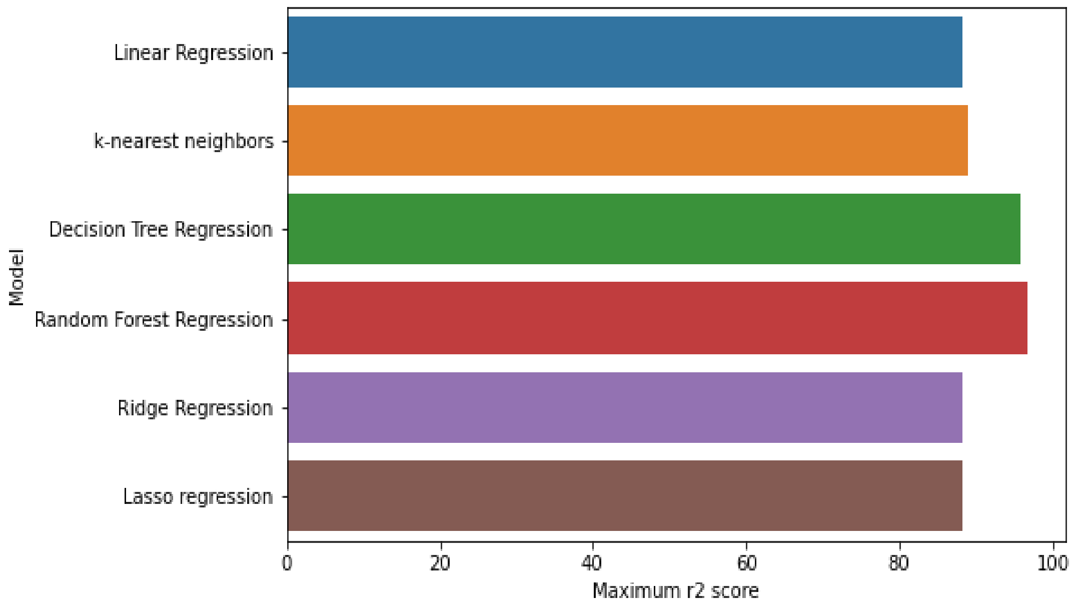
Hyperparameter Tuning: - Hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. We used Grid Search CV to find estimators /neighbors/alpha for learning algorithms and Randomized search CV to find best parameters for implementation of final selected model.

• Visualizations (machine learning): -

All visualizations done before applying machine learning algorithms are shown above. We have done visualizations for exploring output variable, input features, relationship & correlation between input and output features, to check outliers, skewness & missing values, PCA analysis, etc. Here we are visualizing different regression model's performances, scores after applying ensemble methods & Hyperparameter Tuning, Final model selection, relation between true & predicted values & test dataset.

1- **Before using ensemble methods**: - After applying different regression algorithms on model and using cross validation we have obtained different coefficient of determination & cross validation scores and after using Error metrics we have calculated MAE, MSE & RMSE for all models. All scores & errors are shown above. Now we are seeing visualization using Bar Plot of r2 scores: -

```
<AxesSubplot:xlabel='Maximum r2 score', ylabel='Model'>
```

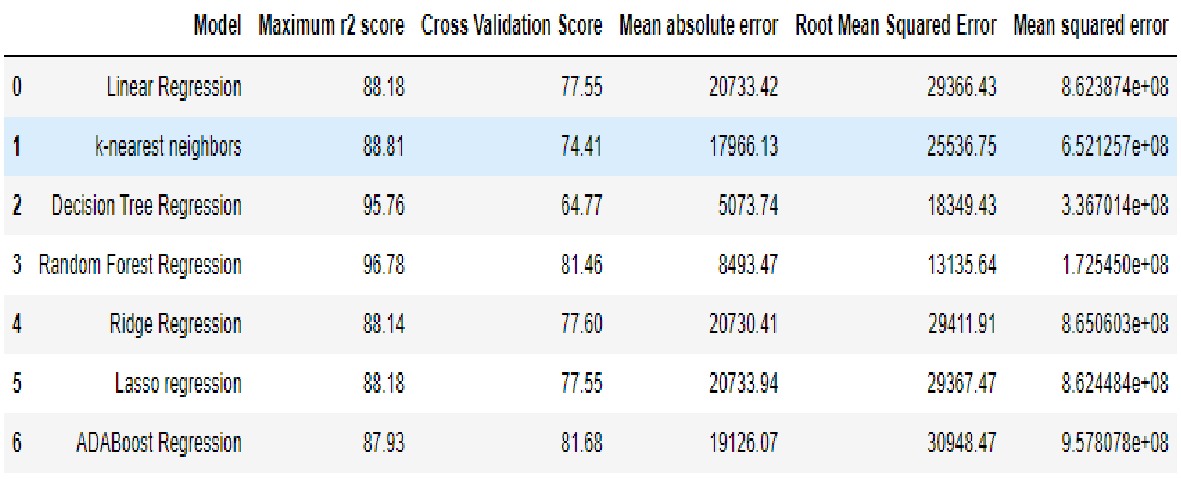


Observations: After comparing above 6 models, these 2 models are good: -

1- Random Forest Regression (R2 & Cross validation scores are highest & RMSE is minimum & other errors are less)

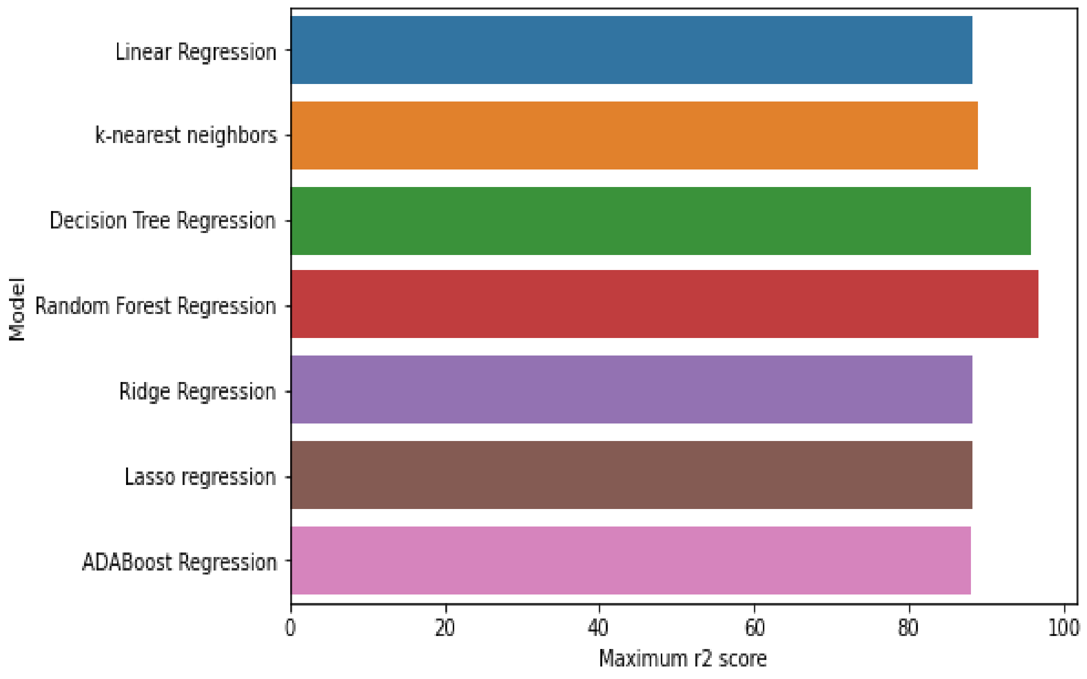
2- Decision Tree Regression (R2 score is good & MAE is minimum)

2- **After using ensemble methods**: - AdaBoost is used as ensemble method to combine several machine learning techniques into one predictive model in order to decrease variance, bias & improve predictions. After applying ensemble methods on model and using cross validation we have obtained different coefficient of determination & cross validation scores and after using Error metrics we have calculated MAE, MSE & RMSE for all models. Now we are seeing visualization using Bar Plot of r2 scores: -



	Model	Maximum r2 score	Cross Validation Score	Mean absolute error	Root Mean Squared Error	Mean squared error
0	Linear Regression	88.18	77.55	20733.42	29366.43	8.623874e+08
1	k-nearest neighbors	88.81	74.41	17966.13	25536.75	6.521257e+08
2	Decision Tree Regression	95.76	64.77	5073.74	18349.43	3.367014e+08
3	Random Forest Regression	96.78	81.46	8493.47	13135.64	1.725450e+08
4	Ridge Regression	88.14	77.60	20730.41	29411.91	8.650603e+08
5	Lasso regression	88.18	77.55	20733.94	29367.47	8.624484e+08
6	ADABoost Regression	87.93	81.68	19126.07	30948.47	9.578078e+08

```
<AxesSubplot:xlabel='Maximum r2 score', ylabel='Model'>
```



Observations:

1- After comparing above 7 models on basis of scores and errors, & also after using ensemble methods still these 2 models Random Forest regression & Decision Tree Regression are giving good performance.

2- Ada boost Regression is giving good scores but errors are high so we will not select this option.

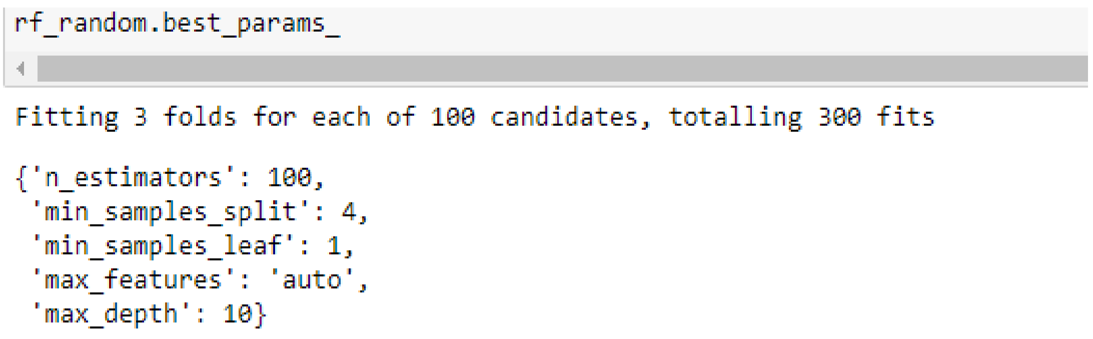
3- But when we see all the parameters very carefully and making a final decision about selection then **Random Forest regression** is the best option because all scores are maximum, RMSE is minimum and also other errors are less.

4- Now we are going to do Hyperparameter Tuning for Random Forest regression models using Randomized Search CV approach for best model selection so that we will get best results after model implementation.

• Interpretation of the Results

After comparing above 7 models on basis of scores and errors, & after using ensemble methods we have selected **Random Forest regression** for this project. To get best scores, now are using randomized search cv for hyperparameter tuning.

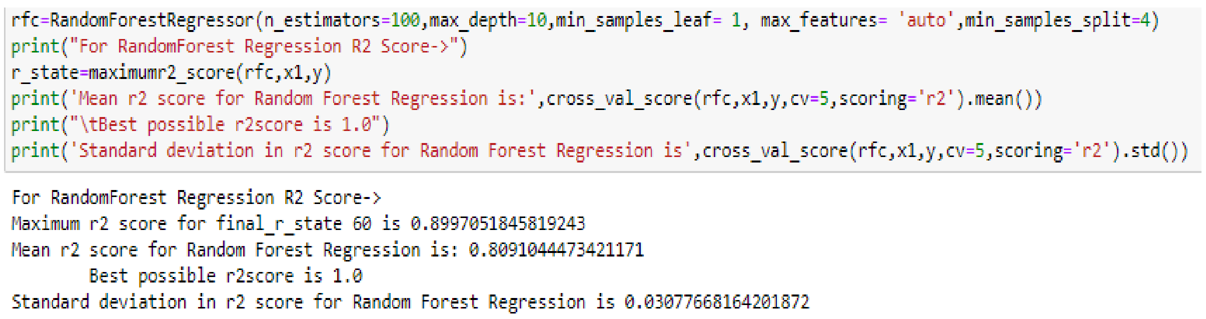
Hyperparameter tuning using randomized search cv – Using Scikit-Learn 's Randomized Search CV method, we can define a grid of hyperparameter ranges, and randomly sample from the grid, performing K-Fold CV with each combination of values.



```
rf_random.best_params_
```

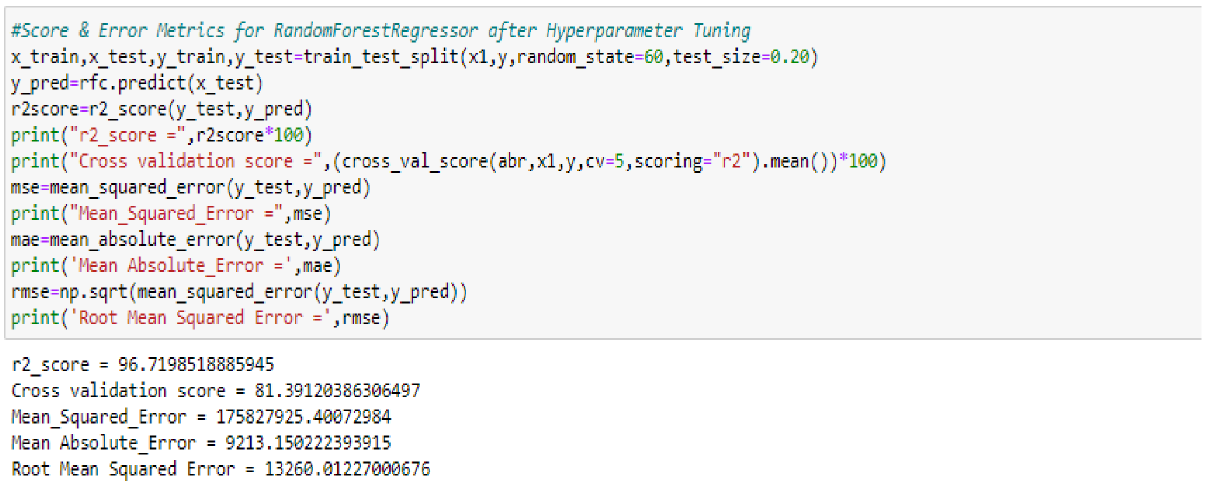
```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

{'n_estimators': 100,
 'min_samples_split': 4,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 10}
```

Evaluation & error metrics for final model: -



```
rfc=RandomForestRegressor(n_estimators=100,max_depth=10,min_samples_leaf= 1, max_features= 'auto',min_samples_split=4)
print("For RandomForest Regression R2 Score->")
r_state=maximumr2_score(rfc,x1,y)
print('Mean r2 score for Random Forest Regression is:',cross_val_score(rfc,x1,y,cv=5,scoring='r2').mean())
print("\tBest possible r2score is 1.0")
print('Standard deviation in r2 score for Random Forest Regression is',cross_val_score(rfc,x1,y,cv=5,scoring='r2').std())
```

```
For RandomForest Regression R2 Score->
Maximum r2 score for final_r_state 60 is 0.8997051845819243
Mean r2 score for Random Forest Regression is: 0.8091044473421171
	Best possible r2score is 1.0
Standard deviation in r2 score for Random Forest Regression is 0.03077668164201872
```



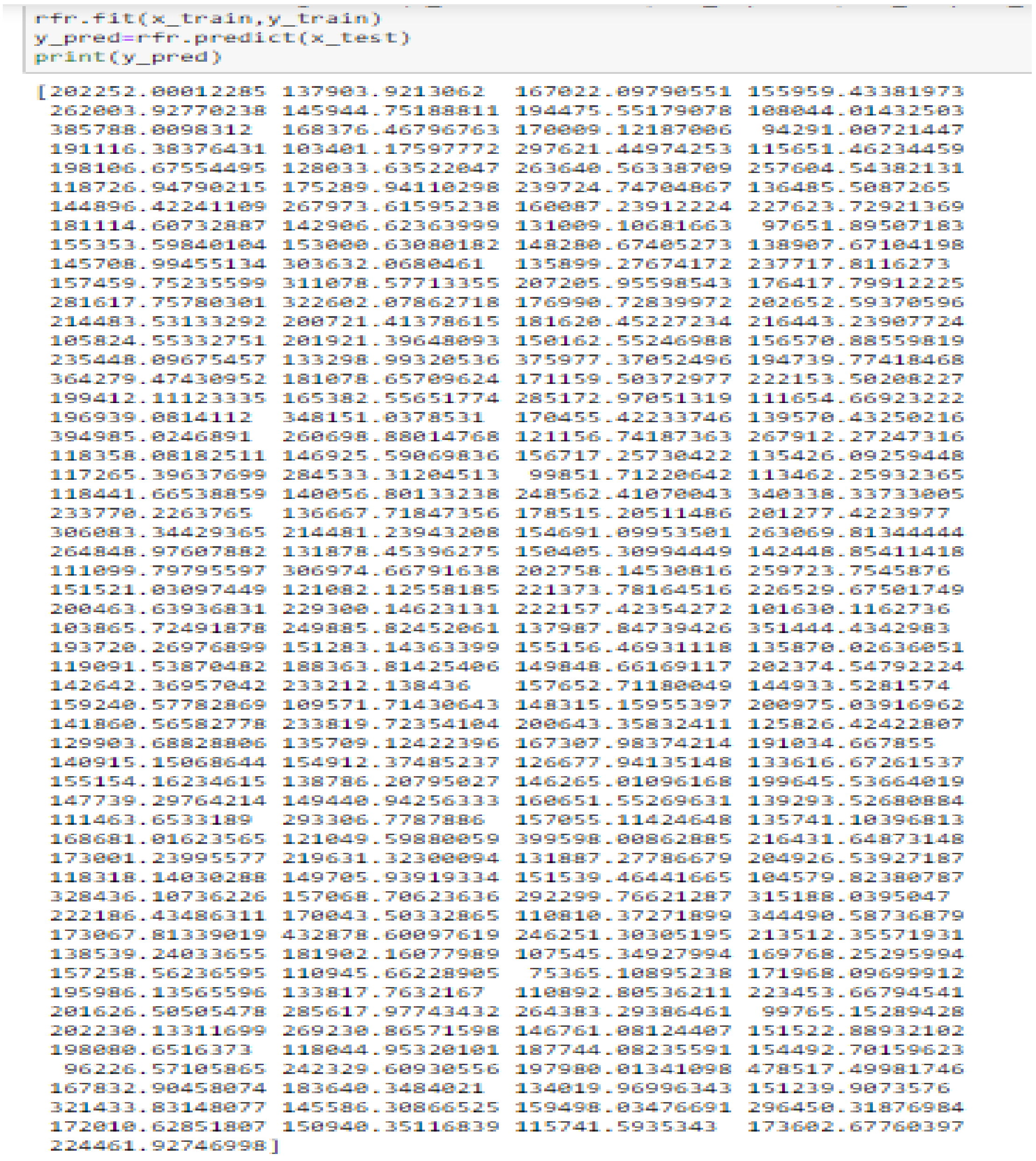
```
#Score & Error Metrics for RandomForestRegressor after Hyperparameter Tuning
x_train,x_test,y_train,y_test=train_test_split(x1,y,random_state=60,test_size=0.20)
y_pred=rfc.predict(x_test)
r2score=r2_score(y_test,y_pred)
print("r2_score =",r2score*100)
print("Cross validation score =",(cross_val_score(abr,x1,y,cv=5,scoring="r2").mean())*100)
mse=mean_squared_error(y_test,y_pred)
print("Mean_Squared_Error =",mse)
mae=mean_absolute_error(y_test,y_pred)
print('Mean Absolute_Error =',mae)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error =',rmse)
```

```
r2_score = 96.7198518885945
Cross validation score = 81.39120386306497
Mean_Squared_Error = 175827925.40072984
Mean Absolute_Error = 9213.150222393915
Root Mean Squared Error = 13260.01227000676
```

Observation: Random Forest Regression gives best performance before Hyper Parameter Tuning & also after Hyper Parameter Tuning as compared to other models so implementation of Random Forest Regression model in our project is good choice.

After implementation of Random Forest regression in our model we are going to check predicted values, true values and relationship between them using visualizations and also Checking Scores and errors after model fitting.

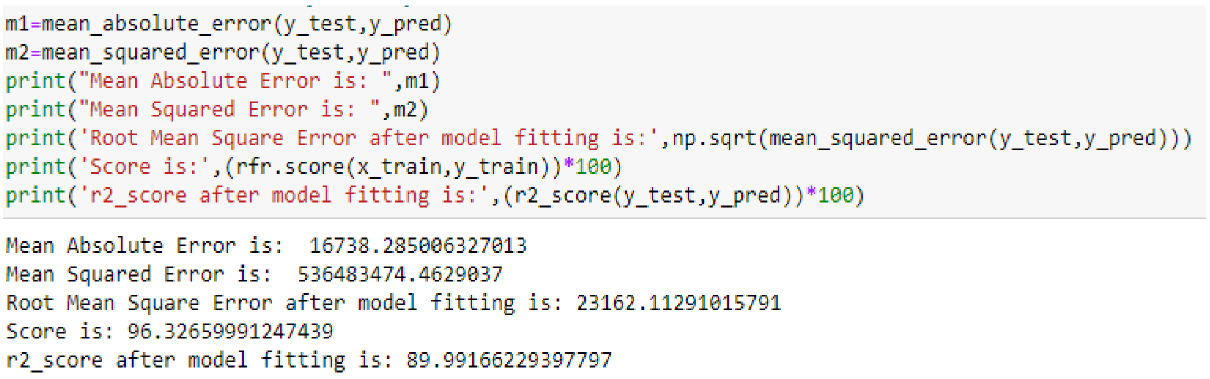
Predictions after Model Fitting: -



```
rfr.fit(x_train,y_train)
y_pred=rfr.predict(x_test)
print(y_pred)
```

```
[202252.00012285 137903.9213062  167022.09790551 155959.43381973
 262003.92770238 145944.75188811 194475.55179078 108044.01432503
 385788.0098312  168376.46796763 170009.12187006  94291.00721447
 191116.38376431 103401.17597772 297621.44974253 115651.46234459
 198106.67554495 128033.63522047 263640.56338709 257604.54382131
 118726.94790215 175289.94110298 239724.74704867 136485.5087265
 144896.42241109 267973.61595238 160087.23912224 227623.72921369
 181114.60732887 142906.62363999 131009.10681663  97651.89507183
 155353.59840104 153000.63080182 148280.67405273 138907.67104198
 145708.99455134 303632.0680461  135899.27674172 237717.8116273
 157459.75235599 311078.57713355 207205.95598543 176417.79912225
 281617.75780301 322602.07862718 176990.72839972 202652.59370596
 214483.53133292 200721.41378615 181620.45227234 216443.23907724
 105824.55332751 201921.39648093 150162.55246988 156570.88559819
 235448.09675457 133298.99320536 375977.37052496 194739.77418468
 364279.47430952 181078.65709624 171159.50372977 222153.50208227
 199412.11123335 165382.55651774 285172.97051319 111654.66923222
 196939.0814112  348151.0378531  170455.42233746 139570.43250216
 394985.0246891  260698.88014768 121156.74187363 267912.27247316
 118358.08182511 146925.59069836 156717.25730422 135426.09259448
 117265.39637699 284533.31204513  99851.71220642 113462.25932365
 118441.66538859 140056.80133238 248562.41070043 340338.33733005
 233770.2263765  136667.71847356 178515.20511486 201277.4223977
 306083.34429365 214481.23943208 154691.09953501 263069.81344444
 264848.97607882 131878.45396275 150405.30994449 142448.85411418
 111099.79795597 306974.66791638 202758.14530816 259723.7545876
 151521.03097449 121082.12558185 221373.78164516 226529.67501749
 200463.63936831 229300.14623131 222157.42354272 101630.1162736
 103865.72491878 249885.82452061 137987.84739426 351444.4342983
 193720.26976899 151283.14363399 155156.46931118 135870.02636051
 119091.53870482 188363.81425406 149848.66169117 202374.54792224
 142642.36957042 233212.138436   157652.71180049 144933.5281574
 159240.57782869 109571.71430643 148315.15955397 200975.03916962
 141860.56582778 233819.72354104 200643.35832411 125826.42422807
 129903.68828806 135709.12422396 167307.98374214 191034.667855
 140915.15068644 154912.37485237 126677.94135148 133616.67261537
 155154.16234615 138786.20795027 146265.01096168 199645.53664019
 147739.29764214 149440.94256333 160651.55269631 139293.52680884
 111463.6533189  293306.7787886  157055.11424648 135741.10396813
 168681.01623565 121049.59880059 399598.00862885 216431.64873148
 173001.23995577 219631.32300094 131887.27786679 204926.53927187
 118318.14030288 149705.93919334 151539.46441665 104579.82380787
 328436.10736226 157068.70623636 292299.76621287 315188.0395047
 222186.43486311 170043.50332865 110810.37271899 344490.58736879
 173067.81339019 432878.60097619 246251.30305195 213512.35571931
 138539.24033655 181902.16077989 107545.34927994 169768.25295994
 157258.56236595 110945.66228905  75365.10895238 171968.09699912
 195986.13565596 133817.7632167  110892.80536211 223453.66794541
 201626.50505478 285617.97743432 264383.29386461  99765.15289428
 202230.13311699 269230.86571598 146761.08124407 151522.88932102
 198080.6516373  118044.95320101 187744.08235591 154492.70159623
  96226.57105865 242329.60930556 197980.01341098 478517.49981746
 167832.90458074 183640.3484021  134019.96996343 151239.9073576
 321433.83148077 145586.30866525 159498.03476691 296450.31876984
 172010.62851807 150940.35116839 115741.5935343  173602.67760397
 224461.92746998]
```

Error Metrics & R2Score after Model Fitting: -

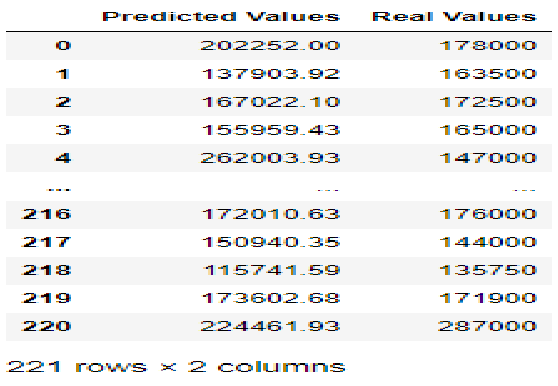


```
m1=mean_absolute_error(y_test,y_pred)
m2=mean_squared_error(y_test,y_pred)
print("Mean Absolute Error is: ",m1)
print("Mean Squared Error is: ",m2)
print('Root Mean Square Error after model fitting is:',np.sqrt(mean_squared_error(y_test,y_pred)))
print('Score is:',(rfr.score(x_train,y_train))*100)
print('r2_score after model fitting is:',(r2_score(y_test,y_pred))*100)
```

```
Mean Absolute Error is:  16738.285006327013
Mean Squared Error is:  536483474.4629037
Root Mean Square Error after model fitting is: 23162.11291015791
Score is: 96.32659991247439
r2_score after model fitting is: 89.99166229397797
```

Saving Final model: - We have saved final model using job lib in *.pkl format.

Evaluate Predictions: - Predicted and real values are as: -

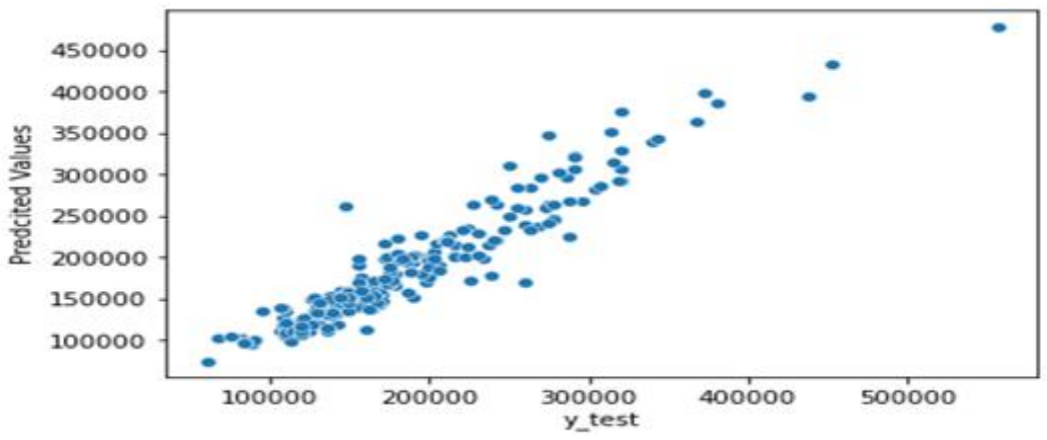


	Predicted Values	Real Values
0	202252.00	178000
1	137903.92	163500
2	167022.10	172500
3	155959.43	165000
4	262003.93	147000
...	...	...
216	172010.63	176000
217	150940.35	144000
218	115741.59	135750
219	173602.68	171900
220	224461.93	287000

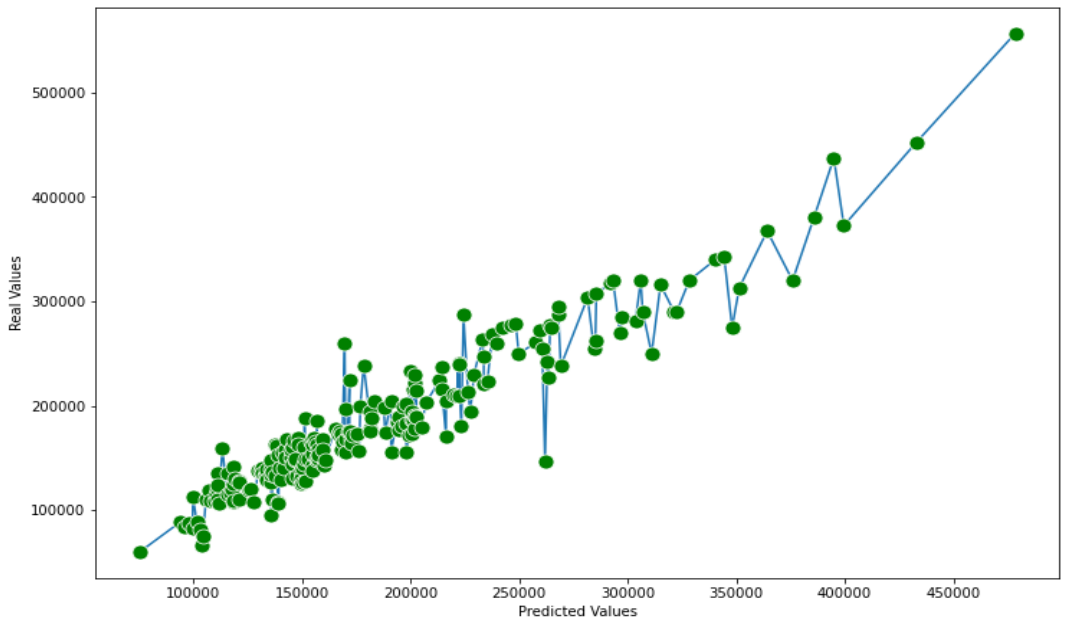
221 rows × 2 columns

Visualizations to find relation between real and Predicted values: -

1- Scatter Plot: -



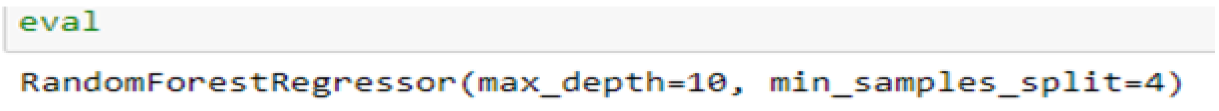
2- Line Plot: -



Observations: 1- Both scatter & Line Plot shows Predicted values are nearly close to real values.

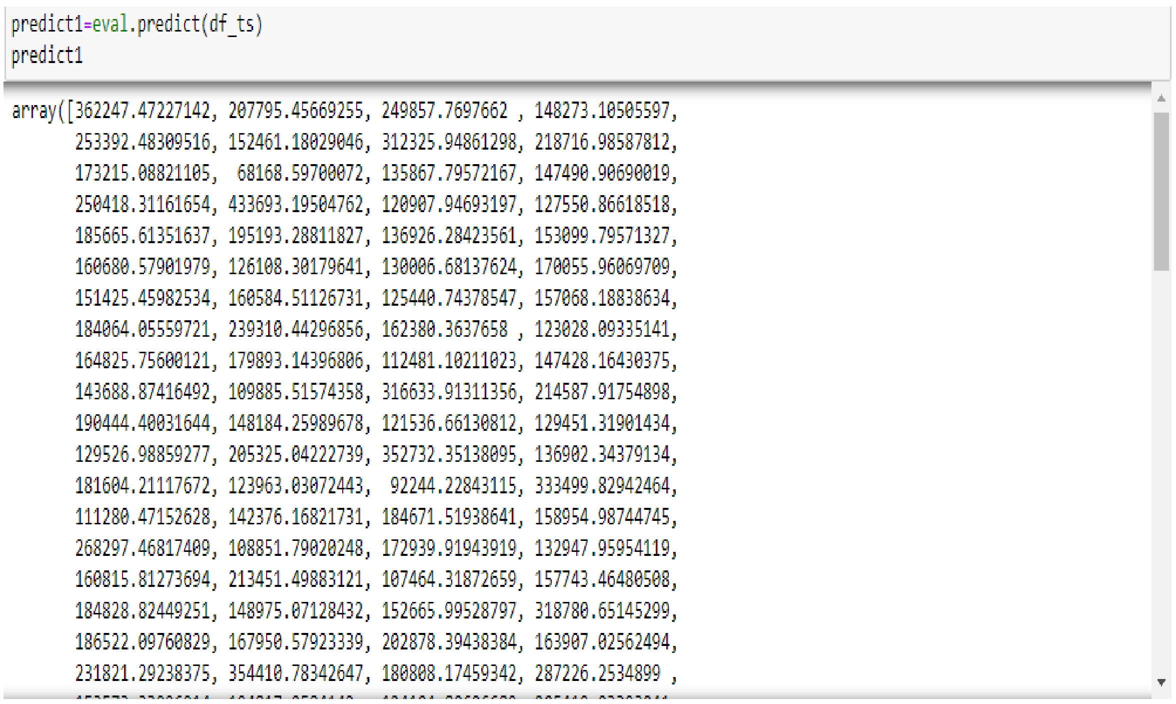
2- Graph is linear except few deviations & it shows good relation between predicted & real values.

Test Dataset: - We have imported test dataset, checked its shape, missing values in it, type of data in it, removed irrelevant columns, handled missing values very clearly & separately for category type & continues type features, did feature engineering using label encoder, did pre-processing using standard scaler, did PCA for dimensionality reduction and in last applied our saved random forest regression model on Test Dataset and predict Sale price.



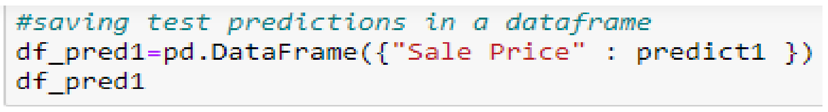
```
eval
```

```
RandomForestRegressor(max_depth=10, min_samples_split=4)
```

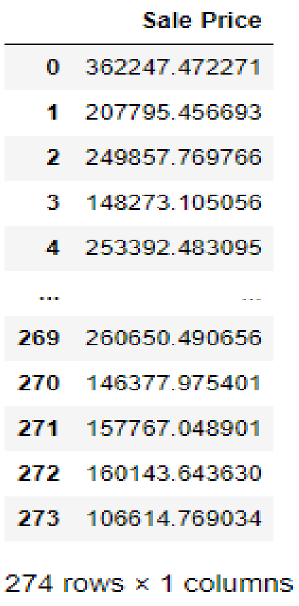


```
predict1=eval.predict(df_ts)
predict1
```

```
array([362247.47227142, 207795.45669255, 249857.7697662 , 148273.10505597,
       253392.48309516, 152461.18029046, 312325.94861298, 218716.98587812,
       173215.08821105,  68168.59700072, 135867.79572167, 147490.90690019,
       250418.31161654, 433693.19504762, 120907.94693197, 127550.86618518,
       185665.61351637, 195193.28811827, 136926.28423561, 153099.79571327,
       160680.57901979, 126108.30179641, 130006.68137624, 170055.96069709,
       151425.45982534, 160584.51126731, 125440.74378547, 157068.18838634,
       184064.05559721, 239310.44296856, 162380.3637658 , 123028.09335141,
       164825.75600121, 179893.14396806, 112481.10211023, 147428.16430375,
       143688.87416492, 109885.51574358, 316633.91311356, 214587.91754898,
       190444.40031644, 148184.25989678, 121536.66130812, 129451.31901434,
       129526.98859277, 205325.04222739, 352732.35138095, 136902.34379134,
       181604.21117672, 123963.03072443,  92244.22843115, 333499.82942464,
       111280.47152628, 142376.16821731, 184671.51938641, 158954.98744745,
       268297.46817409, 108851.79020248, 172939.91943919, 132947.95954119,
       160815.81273694, 213451.49883121, 107464.31872659, 157743.46480508,
       184828.82449251, 148975.07128432, 152665.99528797, 318780.65145299,
       186522.09760829, 167950.57923339, 202878.39438384, 163907.02562494,
       231821.29238375, 354410.78342647, 180808.17459342, 287226.2534899 ,
```



```
#saving test predictions in a dataframe
df_pred1=pd.DataFrame({"Sale Price" : predict1 })
df_pred1
```



	Sale Price
0	362247.472271
1	207795.456693
2	249857.769766
3	148273.105056
4	253392.483095
...	...
269	260650.490656
270	146377.975401
271	157767.048901
272	160143.643630
273	106614.769034

274 rows × 1 columns

Saving Test Predictions: - We have saved test predictions using pandas in *.csv format.

CONCLUSION

• Key Findings and Conclusions of the Study

In this project, we demonstrated the use of machine learning algorithms on a very challenging dataset to predict housing prices. To achieve the best performance, we showed that data pre-processing, a careful selection of techniques of balancing dataset, handling missing values, performed data cleaning, removing skewness, performed PCA and regression algorithms are all very important. Random Forest & Decision Tree regressors work quite well on our dataset, and the use of AdaBoost Regression is also effective. In the future, we want to continue exploring more sophisticated learning algorithms and dimension reduction techniques to further improve model performance on this important prediction task.

Also, we have tested these machine learning algorithms on 2 different PCs of different technical configurations and have also compared their performance using evaluation & error metrics & and then chose the best performing model.

• Learning Outcomes of the Study in respect of Data Science

Learning outcomes are as: -

1- Data cleaning is quite tedious task and also very time taking and while working on this project we had encountered multiple issues but after research, study and guidance we neutralized these issues and also cleaned data properly.
2- Using Data visualization, we can easily identify outliers, skewness, missing values & correlation etc. Also, we can identify the relation between target & other features using it. In this project we have used Matplotlib & Seaborn library for data visualization.
3- Training & Test both Datasets have huge missing values so handled them very carefully.
4- Random Forest Regression have worked best in terms of r2 score & cross validation and RMSE is minimum and other errors are also less. Even it gave best parameters during hyperparameter tuning and we have used these parameters in our final model.

• Limitations of this work and Scope for Future Work

Following limitations and scope of future work are as: -

1- Selection of best random state to calculate the maximum accuracy of model is very time taking especially in case of Random Forest and AdaBoost Algorithms.
2- Hyperparameter tuning using Grid Search CV is very time consuming specially in prediction of best parameters for AdaBoost, Lasso & k-nearest neighbors. So, we used randomized search cv to get best parameters for random forest regression.
3- Dataset have many missing values so it took huge time & steps to handle this issue.
4- Size of dataset is huge so sometimes it was difficult to handle this dataset but after completion of this project we got enough confidence to handle big datasets.
5- There are some irrelevant columns in this dataset so we have tried our best to check and remove them and also tried not to lose important data.
6- We were trying to remove outliers using Z score but data loss was nearly 60.86%.so, we eliminated this process. In future we will try to use more statistical features.
7- We have handled skewness using NumPy methods and also performed PCA.
8- In future we will work on more algorithms to make more efficient model.

||Thank you||