FLIP ROBO

HOUSING PROJECT

Submitted by:

PROMISE AZOM

ACKNOWLEDGMENT

I would like to express my appreciation to Datatrained Education and Fliprobo technologies for their support in bringing me this far in my Data Science Journey. Your lectures and videos made the difference.

I also want to thank my wife and family for their patience.

INTRODUCTION

- ## Business Problem Framing

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 in the CSV file below.

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

Describe the business problem and how this problem can be related to the real world.

- ## Conceptual Background of the Domain Problem

This is critical stage in any machine learning process. It involves brainstorming and coming up with as many hypotheses as possible about what could affect the target variable. It facilitates in exploring the data at hand more efficiently and effectively. Domain Knowledge should be done before seeing the data or else we will end up with biased hypotheses. Below are some anticipated assertions on the problem statement.

- The general zoning classification of the sale is Commercial or Residential Medium Density
- Total Square feet: sum of Above grade (ground) living area square feet, basement area square feet and other place square feet
- Above grade (ground) living area square feet is important
- Size of garage in car capacity
- The house is located in high value Neighbourhood
- the overall material and finish of the house rating
- The pool quality is excellent
- The proximity to various condition is Normal
- The kitchens are above grade
- The overall condition of the house rating
- Original construction date
- Home functionality is Typical Functionality
- The exterior covering on house is Brick Face
- The condition of sale is normal

- **Review of Literature**

  The below key operations will be adopted:

  - Exploratory Data Analysis (EDA)
  - Data Preprocessing (Univariate,Bivariate,Multivariate)
  - Feature Selection
  - Metrics Measurement
  - Model Execution
  - Hyperparameter Tunning
  - Model Saving

- **Motivation for the Problem Undertaken**

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

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Mathematical Modelling:
  - Update of null values
  - Encoding

  Statistical Modelling:
  - Variance Inflation factor
  - SelectKBest: fscore, ANOVA

  Analytical Modelling:
  - Visualization Techniques: Density Plot, Scatter Plot, Count Plot, Boxplot

- **Data Sources and their formats**
  - Two datasets are being provided (test.csv, train.csv).
  - Train Data contains 1168 entries each having 81 variables (Target variable inclusive).
  - Test Data contains 292 entries each having 80 variables (Target variable exclusive).
  - Data contains Null values.
  - Data contains numerical as well as categorical variable.

See below table showing summary of the Variables:

S/N	Variable	Variable Type	Data Type	Data Form	Null	Variable Attributes
1	Id	Continous	int64	Numerical	0	Unique identifier
2	MSSubClass	Continous	int64	Numerical	0	The building class

3	MSZoning	Categorical	object	Non-Numerical	0	The general zoning classification
4	LotFrontage	Continous	float64	Numerical	214	Linear feet of street connected to property
5	LotArea	Continous	int64	Numerical	0	Lot size in square feet
6	Street	Categorical	object	Non-Numerical	0	Type of road access
7	Alley	Categorical	object	Non-Numerical	1091	Type of alley access
8	LotShape	Categorical	object	Non-Numerical	0	General shape of property
9	LandContour	Categorical	object	Non-Numerical	0	Flatness of the property
10	Utilities	Categorical	object	Non-Numerical	0	Type of utilities available
11	LotConfig	Categorical	object	Non-Numerical	0	Lot configuration
12	LandSlope	Categorical	object	Non-Numerical	0	Slope of property
13	Neighborhood	Categorical	object	Non-Numerical	0	Physical locations within Ames city limits
14	Condition1	Categorical	object	Non-Numerical	0	Proximity to main road or railroad
15	Condition2	Categorical	object	Non-Numerical	0	Proximity to main road or railroad (if a second is present)
16	BldgType	Categorical	object	Non-Numerical	0	Type of dwelling
17	HouseStyle	Categorical	object	Non-Numerical	0	Style of dwelling
18	OverallQual	Continous	int64	Numerical	0	Overall material and finish quality
19	OverallCond	Continous	int64	Numerical	0	Overall condition rating
20	YearBuilt	Continous	int64	Numerical	0	Original construction date
21	YearRemodAdd	Continous	int64	Numerical	0	Remodel date
22	RoofStyle	Categorical	object	Non-Numerical	0	Type of roof
23	RoofMatl	Categorical	object	Non-Numerical	0	Roof material
24	Exterior1st	Categorical	object	Non-Numerical	0	Exterior covering on house
25	Exterior2nd	Categorical	object	Non-Numerical	0	Exterior covering on house (if more than one material)
26	MasVnrType	Categorical	object	Non-Numerical	7	Masonry veneer type
27	MasVnrArea	Continous	float64	Numerical	7	Masonry veneer area in square feet
28	ExterQual	Categorical	object	Non-Numerical	0	Exterior material quality
29	ExterCond	Categorical	object	Non-Numerical	0	Present condition of the material on the exterior
30	Foundation	Categorical	object	Non-Numerical	0	Type of foundation

31	BsmtQual	Categorical	object	Non-Numerical	30	Height of the basement
32	BsmtCond	Categorical	object	Non-Numerical	30	General condition of the basement
33	BsmtExposure	Categorical	object	Non-Numerical	31	Walkout or garden level basement walls
34	BsmtFinType1	Categorical	object	Non-Numerical	30	Quality of basement finished area
35	BsmtFinSF1	Continous	int64	Numerical	0	Type 1 finished square feet
36	BsmtFinType2	Categorical	object	Non-Numerical	31	Quality of second finished area (if present)
37	BsmtFinSF2	Continous	int64	Numerical	0	Type 2 finished square feet
38	BsmtUnfSF	Continous	int64	Numerical	0	Unfinished square feet of basement area
39	TotalBsmtSF	Continous	int64	Numerical	0	Total square feet of basement area
40	Heating	Categorical	object	Non-Numerical	0	Type of heating
41	HeatingQC	Categorical	object	Non-Numerical	0	Heating quality and condition
42	CentralAir	Categorical	object	Non-Numerical	0	Central air conditioning
43	Electrical	Categorical	object	Non-Numerical	0	Electrical system
44	1stFlrSF	Continous	int64	Numerical	0	First Floor square feet
45	2ndFlrSF	Continous	int64	Numerical	0	Second floor square feet
46	LowQualFinSF	Continous	int64	Numerical	0	Low quality finished square feet (all floors)
47	GrLivArea	Continous	int64	Numerical	0	Above grade (ground) living area square feet
48	BsmtFullBath	Continous	int64	Numerical	0	full bathroom in the basement
49	BsmtHalfBath	Continous	int64	Numerical	0	Basement half bathrooms
50	FullBath	Continous	int64	Numerical	0	Full bathrooms above grade
51	HalfBath	Continous	int64	Numerical	0	Half baths above grade
52	BedroomAbvGr	Continous	int64	Numerical	0	Bedroom above Ground floor
53	KitchenAbvGr	Continous	int64	Numerical	0	Kitchen above ground floor
54	KitchenQual	Categorical	object	Non-Numerical	0	Kitchen quality
55	TotRmsAbvGrd	Continous	int64	Numerical	0	Total rooms above grade (does not include bathrooms)
56	Functional	Categorical	object	Non-Numerical	0	Home functionality rating
57	Fireplaces	Continous	int64	Numerical	0	Fireplace quality
58	FireplaceQu	Categorical	object	Non-Numerical	551	Fireplace quality
59	GarageType	Categorical	object	Non-Numerical	64	Garage location
60	GarageYrBlt	Continous	float64	Numerical	64	Year garage was built
61	GarageFinish	Categorical	object	Non-Numerical	64	Interior finish of the garage

62	GarageCars	Continous	int64	Numerical	0	Size of garage in car capacity
63	GarageArea	Continous	int64	Numerical	0	Size of garage in square feet
64	GarageQual	Categorical	object	Non-Numerical	64	Garage quality
65	GarageCond	Categorical	object	Non-Numerical	64	Garage condition
66	PavedDrive	Categorical	object	Non-Numerical	0	Paved driveway
67	WoodDeckSF	Continous	int64	Numerical	0	Wood deck area in square feet
68	OpenPorchSF	Continous	int64	Numerical	0	Open porch area in square feet
69	EnclosedPorch	Continous	int64	Numerical	0	Enclosed porch area in square feet
70	3SsnPorch	Continous	int64	Numerical	0	Three season porch area in square feet
71	ScreenPorch	Continous	int64	Numerical	0	Screen porch area in square feet
72	PoolArea	Continous	int64	Numerical	0	Pool area in square feet
73	PoolQC	Categorical	object	Non-Numerical	1161	Pool quality
74	Fence	Categorical	object	Non-Numerical	931	Fence quality
75	MiscFeature	Categorical	object	Non-Numerical	1124	Miscellaneous feature not covered in other categories
76	MiscVal	Continous	int64	Numerical	0	Value of miscellaneous feature
77	MoSold	Continous	int64	Numerical	0	Month Sold
78	YrSold	Continous	int64	Numerical	0	Year Sold
79	SaleType	Categorical	object	Non-Numerical	0	Type of sale
80	SaleCondition	Categorical	object	Non-Numerical	0	Condition of sale
81	SalePrice	Continous	int64	Numerical	0	The property's sale price in dollars (target variable)

NB: The SalePrice is the Dependent Variable (Target/Label) while the rest are the independent variables (Features).

Also see snap shots of data…

Full view of Data

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

First Five rows

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

5 rows × 81 columns

Last Five rows

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

5 rows × 81 columns

Random Four samples

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
1041	14	20	RL	91.0	10652	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
470	1228	20	RL	72.0	8872	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
629	1427	60	RL	81.0	10944	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1102	1063	190	RM	85.0	13600	Pave	Grvl	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0

4 rows × 81 columns

Data Description(Continous Data)

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeck
count	1168.000000	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	...	1168.000
mean	724.136130	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	...	96.206
std	416.159877	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.664785	...	126.158
min	1.000000	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	...	0.000
25%	360.500000	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	...	0.000
50%	714.500000	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	...	0.000
75%	1079.500000	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	...	171.000
max	1460.000000	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.000

8 rows × 38 columns

Data Description(Categorical Data)

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageType	GarageFinish	GarageQua
count	1168	1168	77	1168	1168	1168	1168	1168	1168	1168	...	1104	1104	110
unique	5	2	2	4	4	1	5	3	25	9	...	6	3	
top	RL	Pave	Grvl	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	...	Attchd	Unf	T
freq	928	1164	41	740	1046	1168	842	1105	182	1005	...	691	487	105

4 rows × 43 columns

- **Data Pre-processing Done**

Observations and Assumptions on Data Cleaning

Observations

Below features with null values

- LotFrontage: 214
- Alley: 1091
- MasVnrType : 7
- MasVnrArea: 7
- BsmtQual: 30
- BsmtCond: 30
- BsmtExposure: 31
- BsmtFinType1: 30
- BsmtFinType2: 31
- FireplaceQu: 551

- GarageType: 64

- GarageYrBlt: 64

- GarageFinish: 64

- GarageQual: 64

- GarageCond: 64

- PoolQC : 1161

- Fence : 931

- MiscFeature: 1124

<u>Assumptions</u>

1. All features with null values greater than 900 will be dropped! This will prevent bias in our overall data

- Alley : 1091

- PoolQC : 1161

- MiscFeature: 1124

- Fence: 931

2. We shall apply *fillna* to only the following features while the above four will be dropped:

- LotFrontage: 214

- MasVnrType: 7

- MasVnrArea: 7

- BsmtQual: 30

- BsmtCond: 30
- BsmtExposure: 31
- BsmtFinType1: 30
- BsmtFinType2: 31
- FireplaceQu: 551
- GarageType: 64
- GarageYrBlt: 64
- GarageFinish: 64
- GarageQual: 64
- GarageCond: 64

3. To use the fillna method, we shall adopt the below based on the datatypes:

- All Object data will be filled(*fillna*) with 'mode'.
- All Integer data will be filled(*fillna*) with the 'absolute value Of mean.
- All floating data will be filled(*fillna*) with 'mean'.

Dropping of features with null greater than 900

```
df=df.drop(columns=['Alley','PoolQC','MiscFeature','Fence'],axis=1)
df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	EnclosedPorch	3SsnPorch	ScreenPorch	Pc
0	127	120	RL	NaN	4928	Pave	IR1	Lvl	AllPub	Inside	...	0	0	0	
1	889	20	RL	95.0	15865	Pave	IR1	Lvl	AllPub	Inside	...	0	0	224	
2	793	60	RL	92.0	9920	Pave	IR1	Lvl	AllPub	CulDSac	...	0	0	0	
3	110	20	RL	105.0	11751	Pave	IR1	Lvl	AllPub	Inside	...	0	0	0	
4	422	20	RL	NaN	16635	Pave	IR1	Lvl	AllPub	FR2	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	RL	NaN	9819	Pave	IR1	Lvl	AllPub	Inside	...	0	0	0	
1164	554	20	RL	67.0	8777	Pave	Reg	Lvl	AllPub	Inside	...	0	0	0	
1165	196	160	RL	24.0	2280	Pave	Reg	Lvl	AllPub	FR2	...	0	0	0	
1166	31	70	C (all)	50.0	8500	Pave	Reg	Lvl	AllPub	Inside	...	172	0	0	
1167	617	60	RL	NaN	7861	Pave	IR1	Lvl	AllPub	Inside	...	0	0	0	

1168 rows × 77 columns

Adopting Assumptions on Fillna..

- LotFrontage(float64) - mean was used
- MasVnrType(object) - mode was used
- MasVnrArea(float64) - mean was used
- BsmtQual(object) - mode was used
- BsmtCond(object) - mode was used
- BsmtExposure(object) - mode was used
- BsmtFinType1(object) - mode was used
- BsmtFinType2(object) - mode was used
- FireplaceQu(object) - mode was used
- GarageType(object) - mode was used
- GarageYrBlt(float64) - mean was used
- GarageFinish(object) - mode was used
- GarageQual(object) - mode was used

- GarageCond(object) - mode was used

4. Encoding

- We are fully aware we cannot run an exhaustive EDA on non-numerical data.

- This makes it necessary for us to convert all non-numerical data into numerical ones.

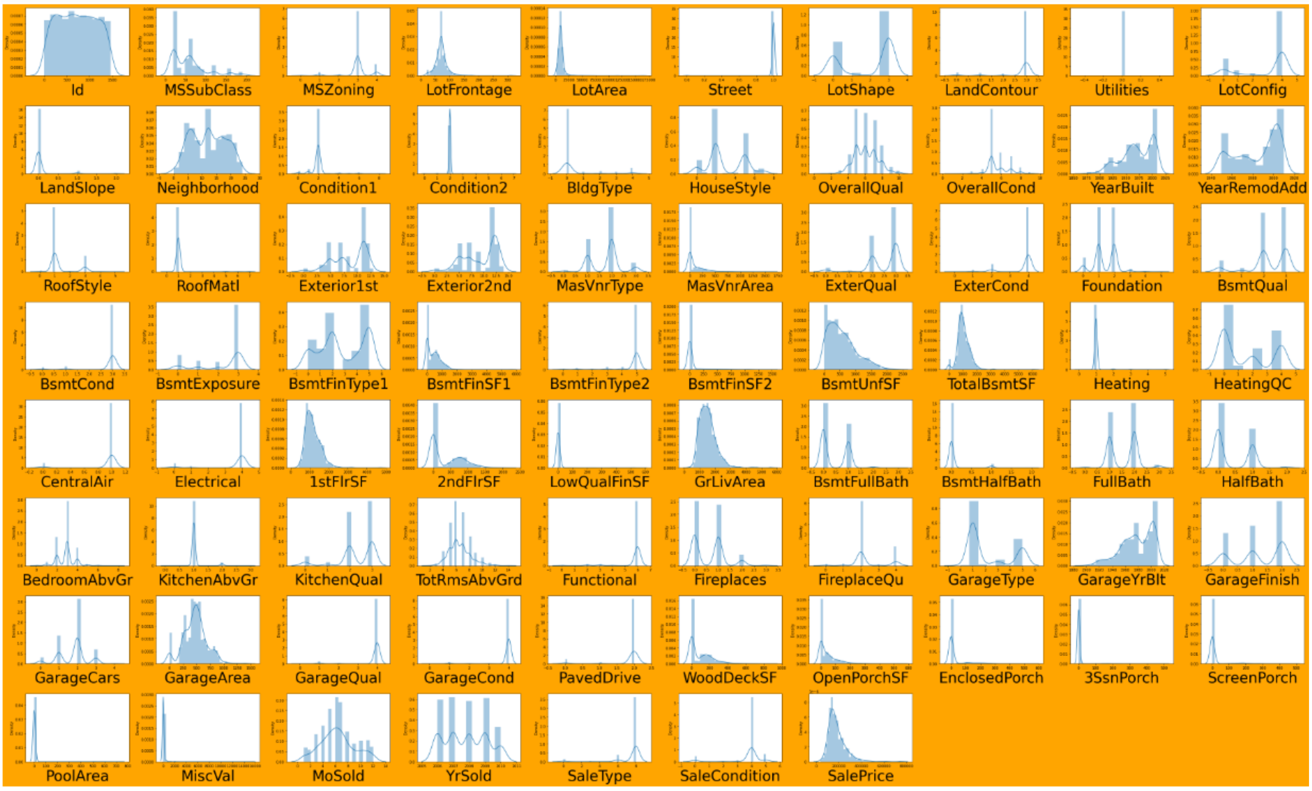
- Hence Label encoding was used on all the object columns.

<u>Encoded data</u>

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	EnclosedPorch	3SsnPorch	ScreenPorch	Pc
0	127	120	3	70.98847	4928	1	0	3	0	4	...	0	0	0	
1	889	20	3	95.00000	15865	1	0	3	0	4	...	0	0	224	
2	793	60	3	92.00000	9920	1	0	3	0	1	...	0	0	0	
3	110	20	3	105.00000	11751	1	0	3	0	4	...	0	0	0	
4	422	20	3	70.98847	16635	1	0	3	0	2	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1163	289	20	3	70.98847	9819	1	0	3	0	4	...	0	0	0	
1164	554	20	3	67.00000	8777	1	3	3	0	4	...	0	0	0	
1165	196	160	3	24.00000	2280	1	3	3	0	2	...	0	0	0	
1166	31	70	0	50.00000	8500	1	3	3	0	4	...	172	0	0	
1167	617	60	3	70.98847	7861	1	0	3	0	4	...	0	0	0	

1168 rows × 77 columns

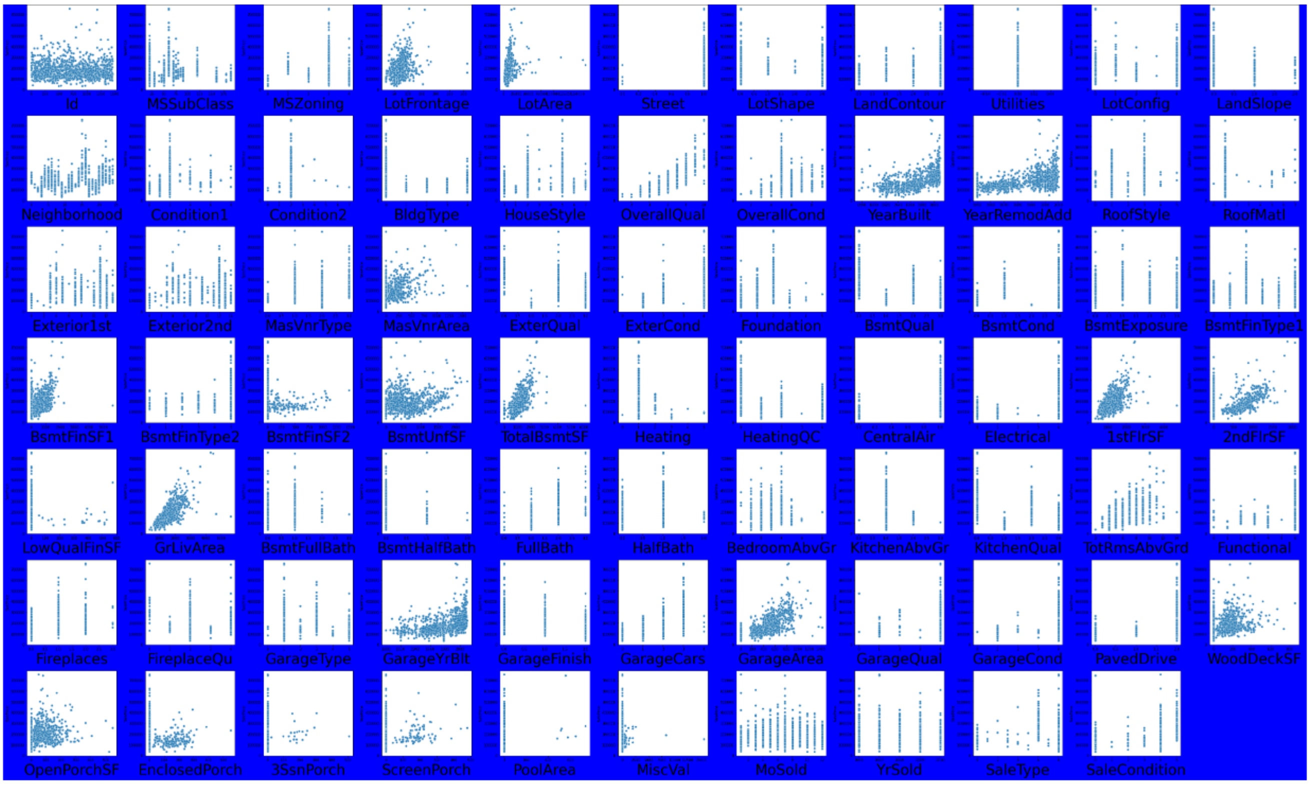
- **Data Inputs- Logic- Output Relationships**
  <u>Normal Distribution Check(Univariate Analysis)</u>
  From a density plot standpoint:



- All the features does not obey a normal distribution, the building blocks is not in tandem with a normalized curve.
- The normal distribution of the sales columns also has no contribution to our Model Building since its the Target variable

Scatter Plot Check(Bivariate Analysis)

From a Scatter plot standpoint:



From the above scatter plot we can see a strong relationship between some of the features and the Label (SalePrice):

1. LotFrontage
2. LotArea
3. Neighborhood
4. YearBuilt
5. YearRemodAdd
6. MasVnrArea
7. BsmtFinSF1
8. BsmtFinSF2
9. BsmtUnfSF
10. TotalBsmtSF
11. 1stFlrSF
12. 2ndFlrSF
13. GrLivArea
14. GarageYrBlt

15. GarageArea
16. WoodDeckSF
17. OpenPorchSF
18. EnclosedPorch
19. 3SsnPorch
20. ScreenPorch
21. PoolArea
22. MiscVal
23. SalePrice

Correlation Check (Collinearity and Multicollinearity)- Multivariate Analysis;

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	EnclosedPorch	3SsnPorc
Id	1.000000	0.004259	0.009307	-0.005969	-0.029212	0.003613	0.022978	-0.020245	NaN	0.053927	...	0.004885	-0.02177
MSSubClass	0.004259	1.000000	0.007478	-0.336681	-0.124151	-0.035981	0.104485	-0.021387	NaN	0.076880	...	-0.004252	-0.04321
MSZoning	0.009307	0.007478	1.000000	-0.069661	-0.023328	0.140215	0.053655	0.001175	NaN	-0.027246	...	0.111221	0.00440
LotFrontage	-0.005969	-0.336681	-0.069661	1.000000	0.299452	-0.035309	-0.144523	-0.073451	NaN	-0.192468	...	0.020902	0.05108
LotArea	-0.029212	-0.124151	-0.023328	0.299452	1.000000	-0.263973	-0.189201	-0.159038	NaN	-0.152063	...	-0.007446	0.02579
...	...	...	...	...	...	...	...	...	...	...	...	...	.
MoSold	0.023479	-0.016015	-0.051646	0.022517	0.015141	-0.008860	-0.050418	-0.023872	NaN	0.019084	...	-0.036523	0.02040
YrSold	-0.008853	-0.038595	-0.004964	-0.003885	-0.035399	-0.019635	0.021421	0.009499	NaN	-0.009817	...	-0.005767	0.01444
SaleType	0.024384	0.035050	0.079854	-0.035356	0.005421	0.025920	-0.015161	-0.041763	NaN	-0.002039	...	-0.008234	-0.01369
SaleCondition	-0.014726	-0.028981	0.004501	0.065091	0.034236	0.014176	-0.054905	0.047715	NaN	0.043692	...	-0.091563	0.00123
SalePrice	-0.023897	-0.060775	-0.133221	0.323779	0.249499	0.044753	-0.248171	0.032836	NaN	-0.060452	...	-0.115004	0.06011

77 rows × 77 columns

From the above correlation statistics;

Collinearity:

- From the above we can see that most of the features have correlation with the target

Multicollinearity:

- From the heatmap we can see that the some pairs of features have noticeable correlation between them

Skewness Check

We assumed a Skewness threshold is taken as +/-0.50. Meaning any value outside +/-0.50 contains skewness. Hence some of the features are having a skewness:

```
OverallQual     0.147539
MiscVal         7.935050
ExterQual      -0.702288
BsmtQual       -1.392097
KitchenQual    -1.487479
GarageCars     -0.363538
FullBath        0.076381
GarageArea     -0.094648
GarageFinish   -0.462565
YearBuilt      -0.499663
MasVnrArea      1.682511
Street          0.000000
LotArea         1.298758
YearRemodAdd   -0.457717
GarageYrBlt    -0.576434
Heating         0.000000
TotRmsAbvGrd    0.411674
MSZoning        1.613111
Fireplaces      0.515527
CentralAir      0.000000
Foundation     -0.250513
BsmtFinSF1      0.566009
OpenPorchSF     1.601716
LotShape       -0.570021
BsmtUnfSF       0.788119
HeatingQC       0.401310
SalePrice       0.968603
```

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

Assumption on Feature Selection

The selectKBest feature was used to pick out 30 features considering their fscores in descending order

	Feature_Name	Score
16	OverallQual	5.303071
71	MiscVal	3.564855
26	ExterQual	3.514221
45	GrLivArea	2.977506
29	BsmtQual	2.876879
52	KitchenQual	2.617125
60	GarageCars	2.578547
48	FullBath	2.435854
61	GarageArea	2.316328
59	GarageFinish	2.187163
18	YearBuilt	2.133300

```
42      1stFlrSF      2.036680
37   TotalBsmtSF      1.867714
25    MasVnrArea      1.852976
5         Street      1.835751
4        LotArea      1.826320
19  YearRemodAdd      1.813783
58   GarageYrBlt      1.725406
38       Heating      1.707885
53  TotRmsAbvGrd      1.656866
2       MSZoning      1.640044
55    Fireplaces      1.591973
40    CentralAir      1.557680
28    Foundation      1.528516
33    BsmtFinSF1      1.482500
43      2ndFlrSF      1.476305
66   OpenPorchSF      1.460290
6       LotShape      1.407526
36     BsmtUnfSF      1.401635
39     HeatingQC      1.358939
```

Assumption on Variance Inflation Factor

We used a variance inflation Factor of 10. Meaning any feature with Variance Inflation Factor greater than 10 is assumed to have a multicollinearity problem. It is not standard. The dataset demands.

In lieu of the above assumption, we will further drop the following:

- GrLivArea

- 1stFlrSF

- TotalBsmtSF

- 2ndFlrSF

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

  Hardware Requirments

  - Device name: DESKTOP
  - ProcessorIntel(R) Core(TM) i7-7500U CPU @ 2.70GHz, 2.90GHz
  - Installed RAM: 8.00 GB
  - System type: 64-bit operating system, x64-based processor

Software Requirments

- Jupyter Notebook
- Python Programming Language
- Libraries
  i. Pandas: Used loading the dataset
  ii. Numpy: Used for rounding up numbers
  iii. sklearn.linear_model: Used for initializing the Linear Regression Model
  iv. sklearn.neighbors: Used for initializing the KNeighbours Regression Model
  v. sklearn.tree: Used for initializing the DecisionTree Regression Model
  vi. sklearn.ensemble: Used for initializing the ensemble Techniques/Models - RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor,ExtraTreesRegressor
  vii. sklearn.preprocessing: Used Scaling, Power Transformation and Encoding of data
  viii. sklearn.feature_selection: Used for feature selection using SelectKBest
  ix. sklearn.model_selection: Used Data split into test and train. Also used for Initializing GridsearchCV during hyperparameter tunning
  x. sklearn.metrics: Used for metrics measurment
  xi. xgboost: Used for initializing the XGBoost Model
  xii. statsmodels: Used to solve for multicollinearity via Variance inflation Factor
  xiii. Scipy.stats: Used to remove outliers via zscore
  xiv. Seaborn: Used for visualization during variate analysis

xv. Matplotlib: Also used for visualization during variate analysis

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)
  The algorithm used were:
  - Standard Scaler
  - train_test_split
  - fit(x_train,y_train)

- Run and Evaluate selected models
  Eight Models were used:
  - Linear Regression

```
lm=LinearRegression()#Initializing...
lm.fit(x_train,y_train)#Training...
pred_test=lm.predict(x_test)#Prediciting using test data...
pred_train=lm.predict(x_train)#Prediciting using training data...
Test_Accuracy_lm= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_lm= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_lm= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_lm= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_lm= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_lm= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_lm= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_lm= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_lm)#printing mean absolute error
print("Test_MSE ", Test_mse_lm)#printing mean squared error
print("Test_RMSE ", Test_rmse_lm)#printing root mean squared error
```

```
Test_Accuracy  85.71
Test_MAE  17860.856664946663
Test_MSE  538567006.6400034
Test_RMSE  23207.046486789382
```

LinearRegression is producing average accuracy 85.71% which is very Good!. Now we will check Cross Validation score as well for overfitting(if exists).

▪ KNeighbors Regressor

```
knn=KNeighborsRegressor()#Initializing...
knn.fit(x_train,y_train)#Training...
pred_test=knn.predict(x_test)#Prediciting using test data...
pred_train=knn.predict(x_train)#Prediciting using training data...
Test_Accuracy_knn= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_knn= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_knn= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_knn= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_knn= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_knn= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_knn= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_knn= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_knn)#printing mean absolute error
print("Test_MSE ", Test_mse_knn)#printing mean squared error
print("Test_RMSE ", Test_rmse_knn)#printing root mean squared error
```

```
Test_Accuracy  83.77
Test_MAE  18145.913368983955
Test_MSE  611494944.726631
Test_RMSE  24728.423822124834
```

KNeighbors is producing average accuracy 83.77% which is very Good!. Now we will check Cross Validation score as well for overfitting(if exists).

▪ Decision Tree Regressor

```
dt=KNeighborsRegressor()#Initializing...
dt.fit(x_train,y_train)#Training...
pred_test=dt.predict(x_test)#Prediciting using test data...
pred_train=dt.predict(x_train)#Prediciting using training data...
Test_Accuracy_dt= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_dt= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_dt= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_dt= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_dt= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_dt= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_dt= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_dt= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_dt)#printing mean absolute error
print("Test_MSE ", Test_mse_dt)#printing mean squared error
print("Test_RMSE ", Test_rmse_dt)#printing root mean squared error
```

```
Test_Accuracy  77.13
Test_MAE  19928.03636363636
Test_MSE  796192310.8308021
Test_RMSE  28216.879891844917
```

Decision Tree is producing average accuracy 77.13% which is very Good!. Now we will check Cross Validation score as well for overfitting(if exists).

- ## RandomForest Regressor

```
rf=LinearRegression()#Initializing...
rf.fit(x_train,y_train)#Training...
pred_test=rf.predict(x_test)#Prediciting using test data...
pred_train=rf.predict(x_train)#Prediciting using training data...
Test_Accuracy_rf= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_rf= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_rf= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_rf= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_rf= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_rf= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_rf= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_rf= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_rf)#printing mean absolute error
print("Test_MSE ", Test_mse_rf)#printing mean squared error
print("Test_RMSE ", Test_rmse_rf)#printing root mean squared error
```

```
Test_Accuracy  86.9
Test_MAE  17227.728259494368
Test_MSE  462143929.540206
Test_RMSE  21497.533103596004
```

RandomForest is producing fair accuracy = 86.9%!. Now we will check Cross Validation score as well for overfitting(if exists).

- ## AdaBoost Regressor

```
ada=AdaBoostRegressor()#Initializing...
ada.fit(x_train,y_train)#Training...
pred_test=ada.predict(x_test)#Prediciting using test data...
pred_train=ada.predict(x_train)#Prediciting using training data...
Test_Accuracy_ada= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_ada= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_ada= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_ada= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_ada= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_ada= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_ada= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_ada= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_ada)#printing mean absolute error
print("Test_MSE ", Test_mse_ada)#printing mean squared error
print("Test_RMSE ", Test_rmse_ada)#printing root mean squared error
```

```
Test_Accuracy  83.94
Test_MAE  18745.848989496997
Test_MSE  566470827.1510409
Test_RMSE  23800.64762041237
```

AdaBoost is producing good accuracy = 83.94!. Now we will check Cross Validation score as well for overfitting(if exists).

GradientBoosting Regressor

```
gb=GradientBoostingRegressor()#Initializing...
gb.fit(x_train,y_train)#Training...
pred_test=gb.predict(x_test)#Prediciting using test data...
pred_train=gb.predict(x_train)#Prediciting using training data...
Test_Accuracy_gb= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_gb= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_gb= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_gb= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_gb= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_gb= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_gb= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_gb= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_gb)#printing mean absolute error
print("Test_MSE ", Test_mse_gb)#printing mean squared error
print("Test_RMSE ", Test_rmse_gb)#printing root mean squared error
```

```
Test_Accuracy  87.7
Test_MAE  15879.461846050717
Test_MSE  463448560.83700955
Test_RMSE  21527.855463027652
```

GradientBoosting is producing good accuracy = 87.7%. Now we will check Cross Validation score as well for overfitting(if exists).

XGBoost Regressor

```
xgb=XGBRegressor()#Initializing...
xgb.fit(x_train,y_train)#Training...
pred_test=xgb.predict(x_test)#Prediciting using test data...
pred_train=xgb.predict(x_train)#Prediciting using training data...
Test_Accuracy_xgb=(metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_xgb= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_xgb= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_xgb= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_xgb= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_xgb= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_xgb= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_xgb= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_xgb)#printing mean absolute error
print("Test_MSE ", Test_mse_xgb)#printing mean squared error
print("Test_RMSE ", Test_rmse_xgb)#printing root mean squared error
```

```
Test_Accuracy  86.94
Test_MAE  16246.7420203877
Test_MSE  446360918.4544198
Test_RMSE  21127.255345984242
```

XGBoost is producing good accuracy = 86.94%. Now we will check Cross Validation score as well for overfitting(if exists).

- ExtraTrees Regressor

```
ex=ExtraTreesRegressor()#Initializing...
ex.fit(x_train,y_train)#Training...
pred_test=ex.predict(x_test)#Prediciting using test data...
pred_train=ex.predict(x_train)#Prediciting using training data...
Test_Accuracy_ex= (metrics.r2_score(y_test,pred_test))#Calculating r2 score for test data
Train_Accuracy_ex= (metrics.r2_score(y_train,pred_train))#Calculating r2 score for training data
Test_mae_ex= mean_absolute_error(y_test,pred_test)#Calculating mean absolute error for test data
Train_mae_ex= mean_absolute_error(y_train,pred_train)#Calculating mean absolute error for training data
Test_mse_ex= mean_squared_error(y_test,pred_test)#Calculating mean squared error for test data
Train_mse_ex= mean_squared_error(y_train,pred_train)#Calculating mean squared error for training data
Test_rmse_ex= np.sqrt(mean_squared_error(y_test,pred_test))#Calculating root mean squared error for test data
Train_rmse_ex= np.sqrt(mean_squared_error(y_train,pred_train))#Calculating root mean squared error for training data
print("Test_Accuracy ",round(metrics.r2_score(y_test,pred_test)*100,2))#printing testing accuracy
print("Test_MAE ", Test_mae_ex)#printing mean absolute error
print("Test_MSE ", Test_mse_ex)#printing mean squared error
print("Test_RMSE ", Test_rmse_ex)#printing root mean squared error
```

```
Test_Accuracy  87.33
Test_MAE  15801.065561497326
Test_MSE  446908284.1821636
Test_RMSE  21140.205395931316
```

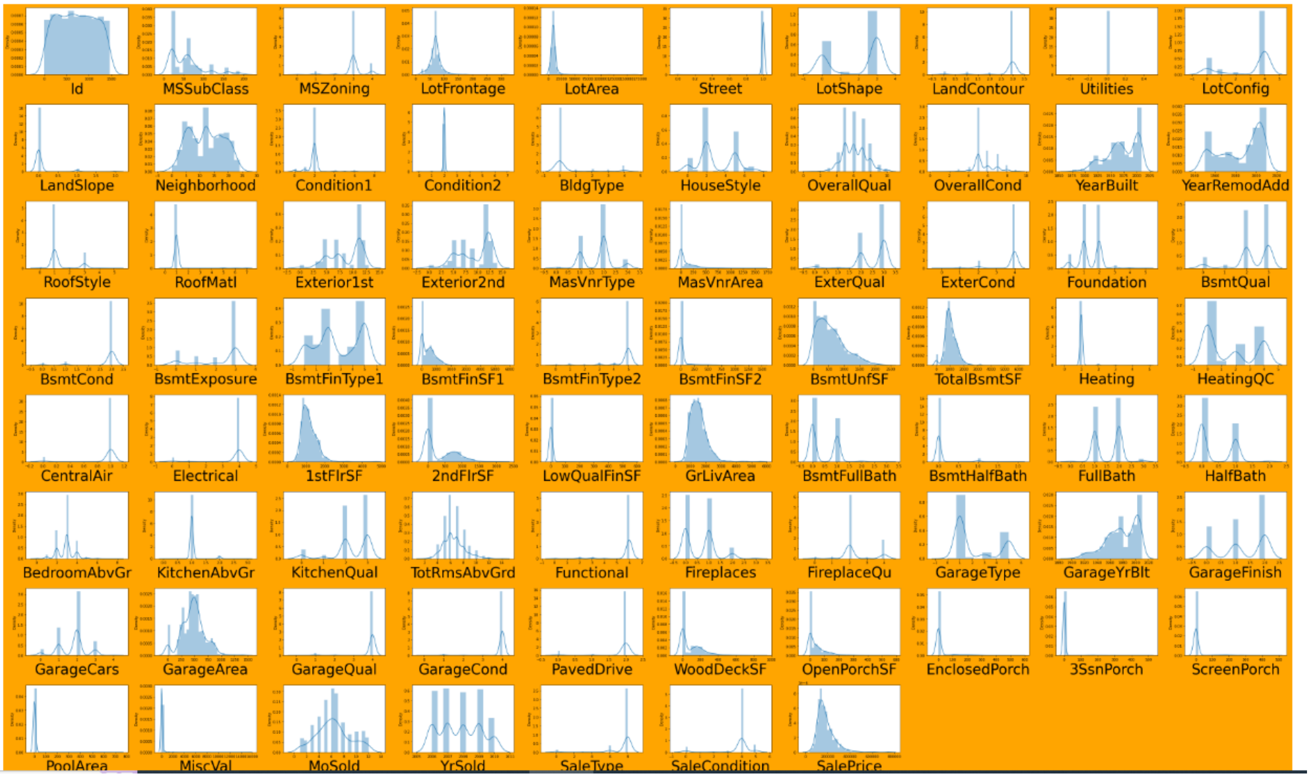
ExtraTress is producing good accuracy = 87.33%. Now we will check Cross Validation score as well for overfitting(if exists).

- **Key Metrics for success in solving problem under consideration**
- R2score
- Cross validation
- Root mean square.

The above metrics were used since it’s a Regression Problem

- **Visualizations**

<u>Normal Distribution Check(Univariate Analysis)</u>



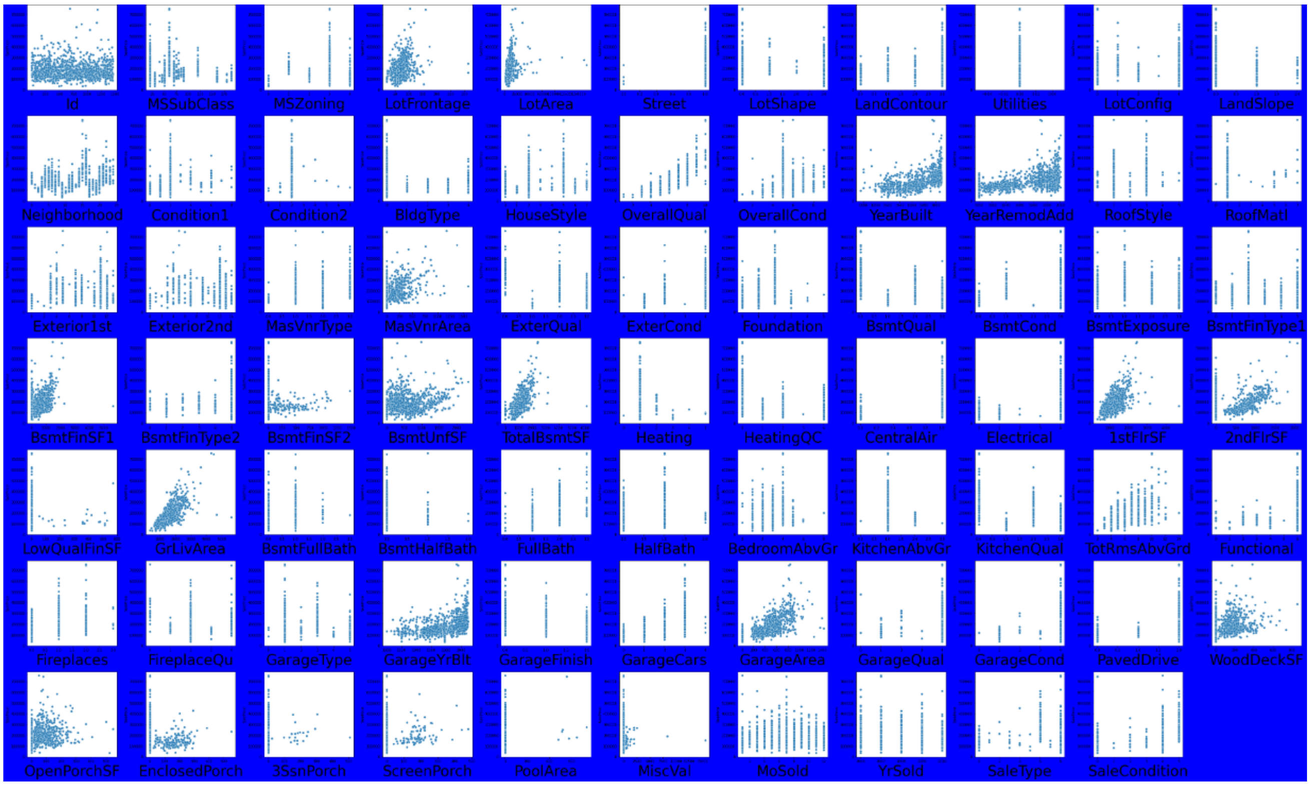
<u>Observations on Normal Distribution Check</u>

From the above density plot

- We observed all the features does not obey a normal distribution, the building blocks is not in tandem with a normalized curve.

- The normal distribution of the sales columns also has no contribution to our Model Building since its the Target variable

Scatter Plot Check(Bivariate Analysis)

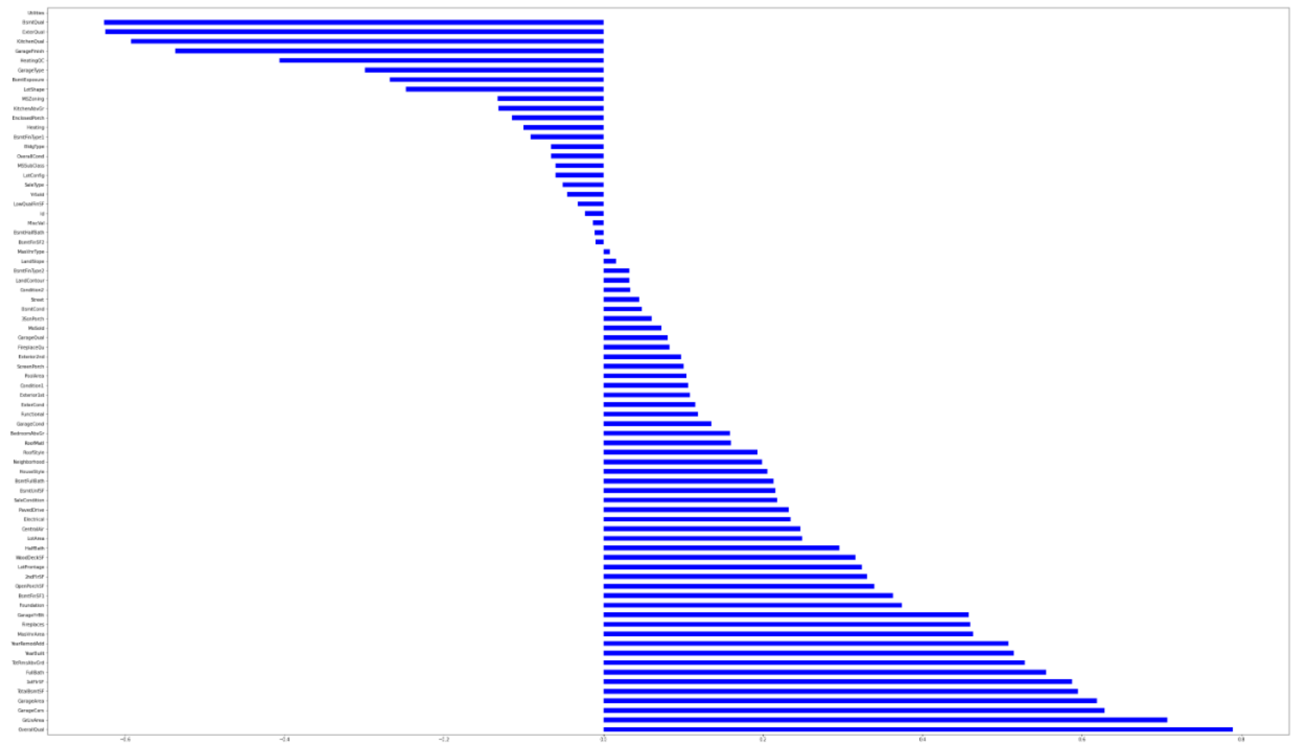


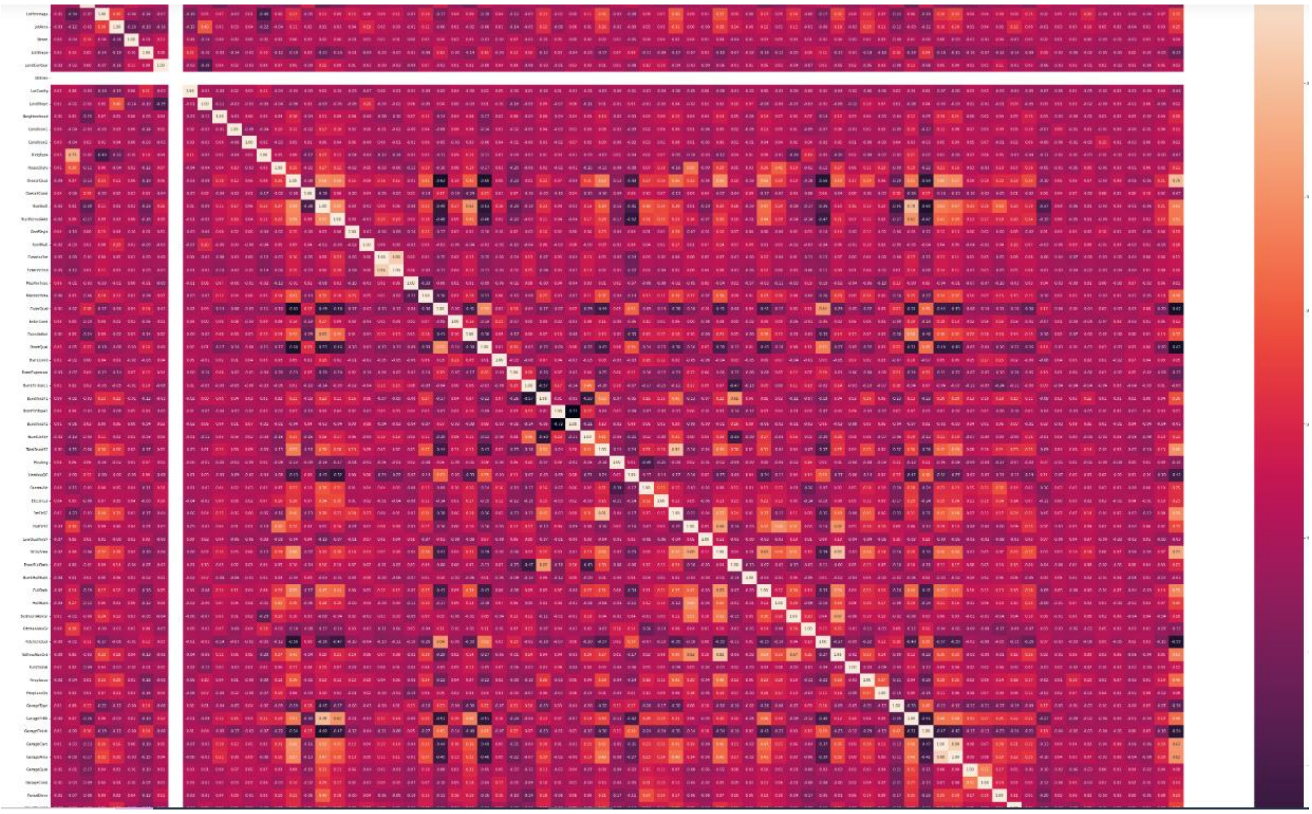
From the above scatter plot we can see a strong relationship between some of the features and the Label (SalePrice):

24. LotFrontage
25. LotArea
26. Neighborhood
27. YearBuilt
28. YearRemodAdd
29. MasVnrArea
30. BsmtFinSF1
31. BsmtFinSF2
32. BsmtUnfSF
33. TotalBsmtSF
34. 1stFlrSF
35. 2ndFlrSF
36. GrLivArea
37. GarageYrBlt
38. GarageArea

39. WoodDeckSF
40. OpenPorchSF
41. EnclosedPorch
42. 3SsnPorch
43. ScreenPorch
44. PoolArea
45. MiscVal
46. SalePrice

Correlation Check(Collinearity and Multicollinearity)- Multivariate Analysis





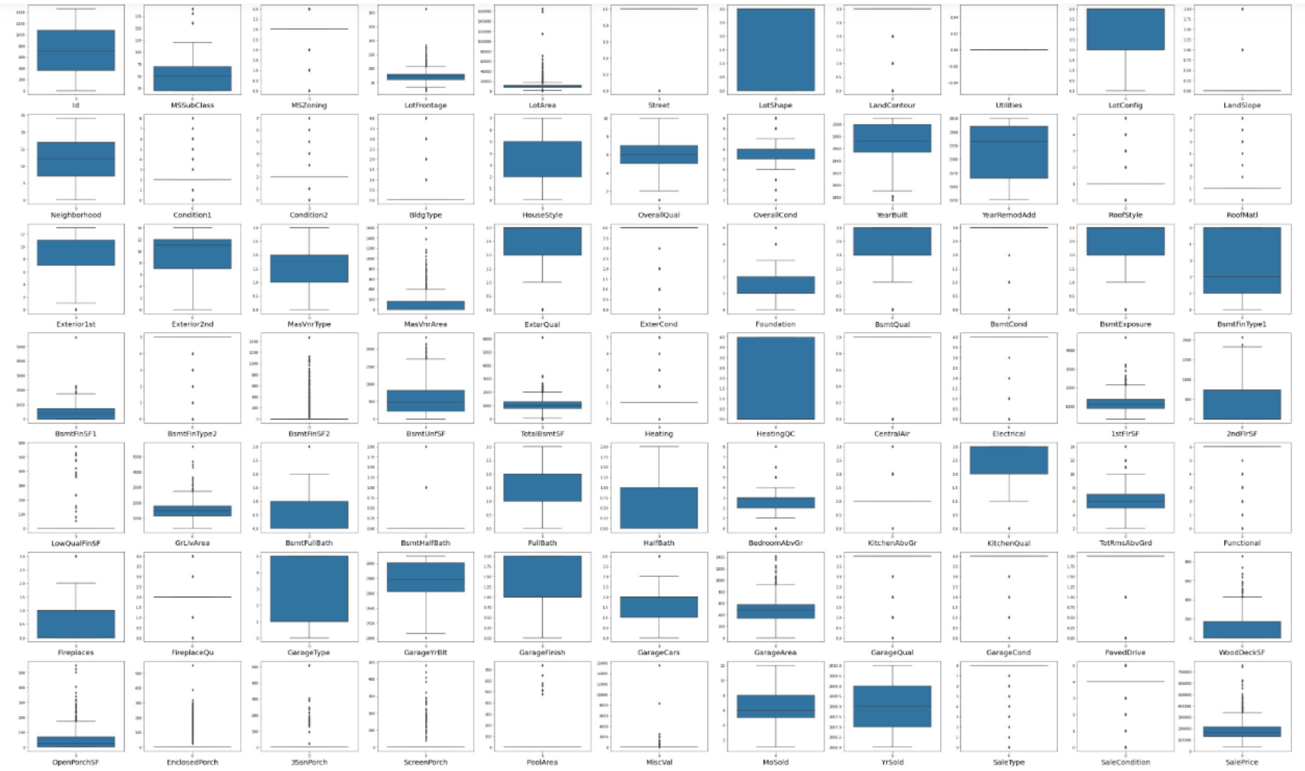
<u>Observations</u>

From the above correlation statistics;

Collinearity: From the above we can see that most of the features have correlation with the target

 Multicollinearity: From the heatmap we can see that the some pairs of features have noticeable correlation between them

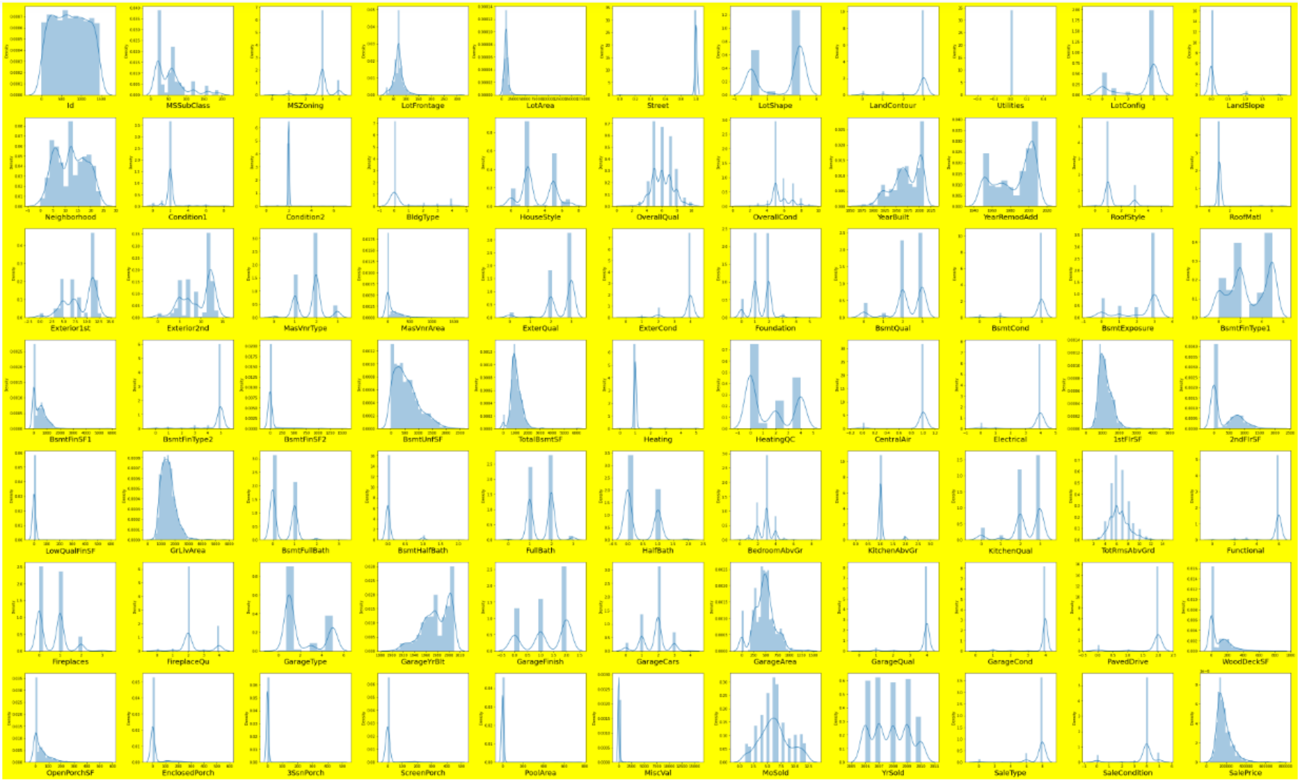
Outlier Check



Observations on Outlier Check

From the above visualization plot its evident the most features possess outliers,

Skewness Check



Observations on Skewness Check:

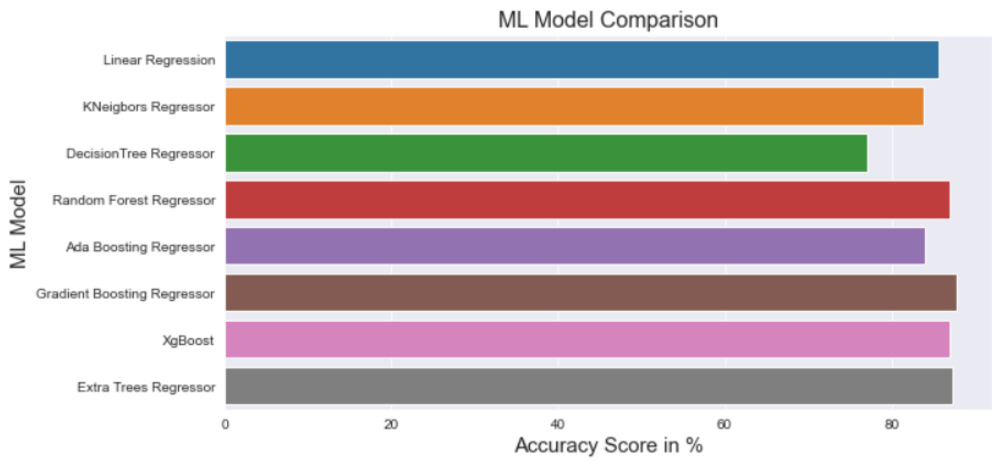
We assumed a Skewness threshold of +/-0.50. Meaning any value outside +/-0.50 contains skewness. Hence some of the features are having a skewness.

CONCLUSION

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

  COMPARING ALL EIGHT MACHINE LEARNING MODELS

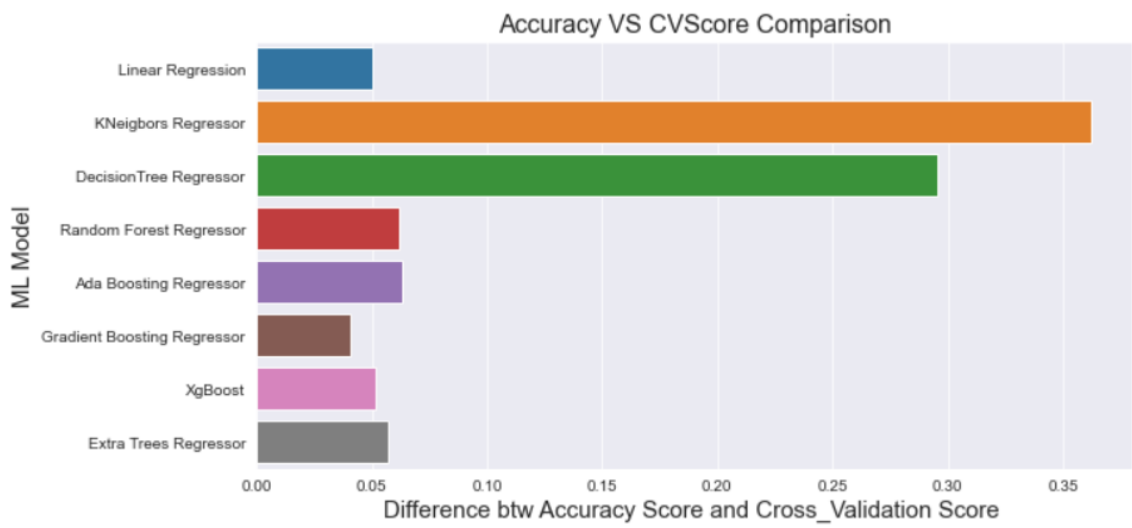
	ML_Model	Accuracy_Score	Cross_Validation_Score	Accuracy_VS_CVScore	MAE	MSE	RMSE
5	Gradient Boosting Regressor	87.70	83.61	0.040880	15879.461846	4.634486e+08	21527.855463
7	Extra Trees Regressor	87.33	81.61	0.057158	15801.065561	4.469083e+08	21140.205396
6	XgBoost	86.94	81.75	0.051824	16246.742020	4.463609e+08	21127.255346
3	Random Forest Regressor	86.90	80.69	0.062118	17227.728259	4.621439e+08	21497.533104
0	Linear Regression	85.71	80.69	0.050224	17860.856665	5.385670e+08	23207.046487
4	Ada Boosting Regressor	83.94	77.63	0.063071	18745.848989	5.664708e+08	23800.647620
1	KNeigbors Regressor	83.77	47.59	0.361805	18145.913369	6.114949e+08	24728.423822
2	DecisionTree Regressor	77.13	47.59	0.295419	19928.036364	7.961923e+08	28216.879892



Now from the above diagram it seems that Gradient Boosting Regrerssor(87.70%) has the highest Accuracy, However, our aim is to find the BEST MODEL, by considering the least difference Between Accuracy_Score and Cross_Validation_Score....

Comparing Differences between Accuracy and Cross_Validation Scores...

	ML_Model	Accuracy_Score	Cross_Validation_Score	Accuracy_VS_CVScore	MAE	MSE	RMSE
5	Gradient Boosting Regressor	87.70	83.61	0.040880	15879.461846	4.634486e+08	21527.855463
0	Linear Regression	85.71	80.69	0.050224	17860.856665	5.385670e+08	23207.046487
6	XgBoost	86.94	81.75	0.051824	16246.742020	4.463609e+08	21127.255346
7	Extra Trees Regressor	87.33	81.61	0.057158	15801.065561	4.469083e+08	21140.205396
3	Random Forest Regressor	86.90	80.69	0.062118	17227.728259	4.621439e+08	21497.533104
4	Ada Boosting Regressor	83.94	77.63	0.063071	18745.848989	5.664708e+08	23800.647620
2	DecisionTree Regressor	77.13	47.59	0.295419	19928.036364	7.961923e+08	28216.879892
1	KNeigbors Regressor	83.77	47.59	0.361805	18145.913369	6.114949e+08	24728.423822



From the above we can see the Model with least difference is STILL GRADIENT BOOSTING REGRESSOR!

Conclusion on Best Choice of Model

From the above we can see:

- The Model with least difference (0.04) between Accuracy Score (r2 score) and Cross Validation Score is GRADIENT BOOSTING REGRESSOR!

- Accuracy is 87%

- We also went further to engage in **Hyperparamter Tunning** using GridSearchCV and arrived at a Final Accuracy of 88%

- Learning Outcomes of the Study in respect of Data Science
  - The SVM and MLP were not suitable models as they produced negative accuracy.
  - Dropping of features with more than 50% null was very key in feature selection
  - SelectKBest was also instrumental in feature selection since we had a lot of features.
- **Limitations of this work and Scope for Future Work**

  The limitation was basically Time