HOUSING: PRICE PREDICTION

Submitted by:

ANKIT DADARWALA

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

- https://www.kaggle.com/harshjain123/feature-engineering-from-scratch using this website for Features selection, Missing value handling, Assumption, Statistical analysis.
- https://www.analyticssteps.com/blogs/introduction-statistical-data-analysis used this website for statistics theory concept and study about data set.
- https://www.codespeedy.com/p-value-in-machine-learning-python/ use for statistics tools like hypothesis test, p- values for features selection and features analysis.
- Use Some Googles, YouTube channels and Stack overflow for coding.

INTRODUCTION

• Business Problem Framing

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.

• Conceptual Background of the Domain Problem

House is very necessary thing on earth for each n every person for stay in and spend time with family or friends. On now today property prices is very high due real estate growth and as better arcatures are facilitate with like luxurious house or flats and also Hi-tech gadgets like automation of electrical equipment other facilities like extra spaces for vehicles, Storage room, 3BHK,4BKH, basement area, gardens, Bathrooms with attached rooms, more kitchen spaces with optimize with kitchen appliances and good-looking kitchens with latest hi-tech furniture. Also, facility with individual bedrooms like children's bedrooms, master bedrooms

In flats also proving penthouse with garden home looks beautiful, Duplex flats also available for big family, as well Fire safety facilities ,Other facilities like gas line provided by Govt, Electricity supplies, Water supplies for 24Hours,pool facility etc…

• Review of Literature

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.

• Motivation for the Problem Undertaken

Now days House buys are easy from the ecommerce websites like 99acres.com, housing.com, magicbricks.com etc. But for the person who not familiar with real estate and does not know how find exact value of the house, and they hired broker for that to find house according to their budgets and pay them fess too. So for that Make Machine Learning algorithm that find the price of house by given features(description of house) like

Lot area, frontage area, location of house, condition of house, construction year, Any modification done/not, foundation type, total square feet, usable square feet of house, No. of floors, No. of bedrooms ,total rooms, kitchen areas, pool facility ,water and gas facility etc….

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  We have Use two linear models for problems like Linear Regression and Logistic Regression, they apply according to target columns so if we have Binary or Class column, we use Logistic Regression and If we have Continuous feature target column use Linear Regression.

- ## Data Sources and their formats

In given problem two datasets provided for Training and Testing in csv format. Load data set into Pandas Data frame format, present here

Types of features as below given codes

```
# load dataset
train_df=pd.read_csv('Train.csv')
test_df=pd.read_csv('test.csv')
```

```
# train dataset
pd.set_option('display.max_columns',82)
train_df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Co
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill	Norm	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes	Norm	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge	Norm	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes	Norm	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	Edwards	Feedr	
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	NPkVill	Norm	
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	Inside	Gtl	IDOTRR	Feedr	
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	

1168 rows × 81 columns

train_df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
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)
memory usage: 739.2+ KB
```

- Data Preprocessing Done

So, in that we have check Missing values from the dataset and graphical represent it by code

From that we find insights that some features have 100% to 85% null values so we can't assume right values for that so decide to drop that columns are ["PoolQC,MiscFeature,Alley,Fence"] in not much correlated with any other features.

Now we have separate features of train and test dataset according to their types of format like Numerical, Categorical, Continues, Discrete features for better under standing of their corelation with target columns. We have 39- categorical features, 16- Continues features, 16- Discrete features, 38- Numerical features, 5- Time related features.

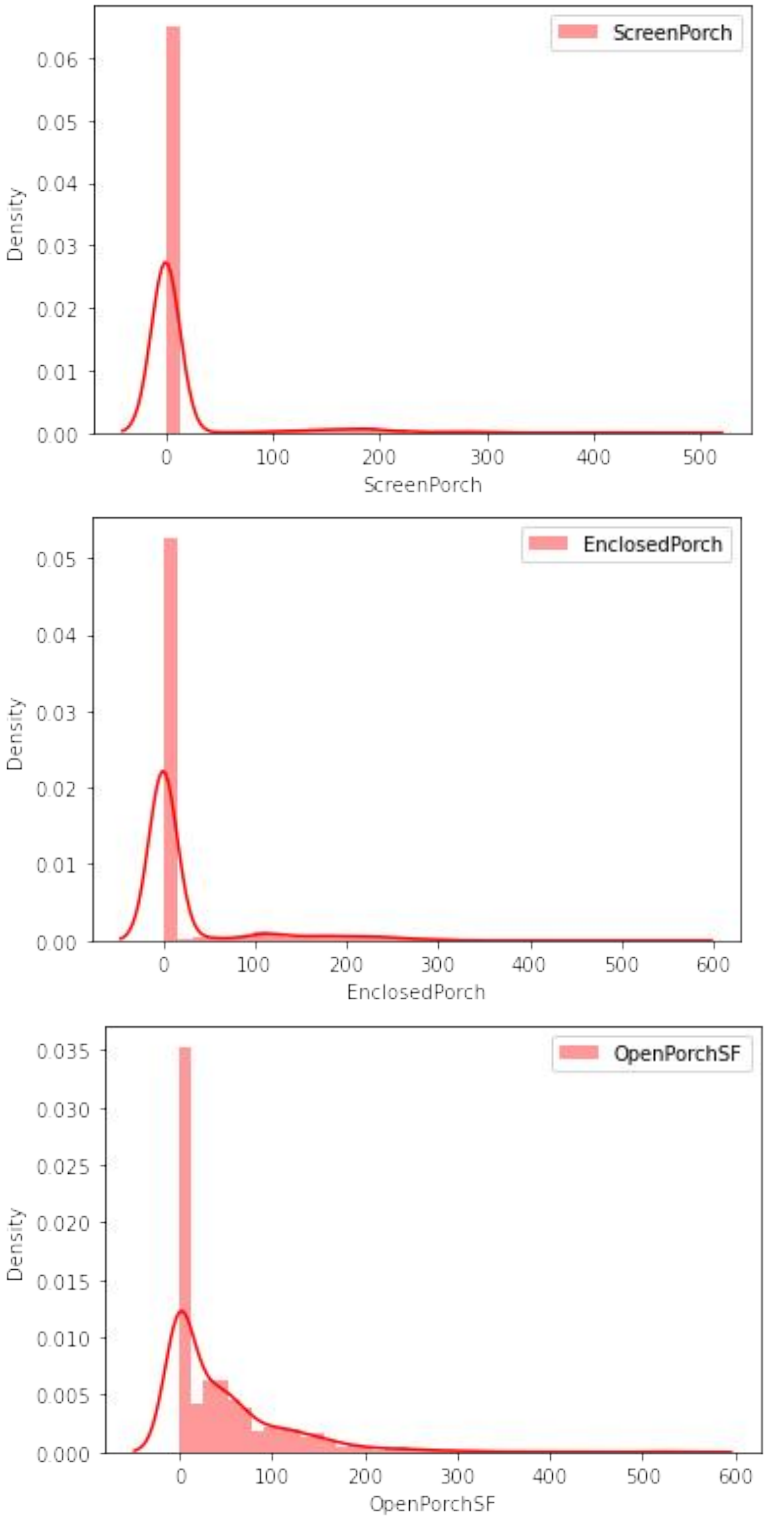
In statistical analysis we have also check outliers and skewness of dataset in that "Lotfrontage , MS subclass, Lotarea, MasVnrArea, SalePrice,etc… have outliers.

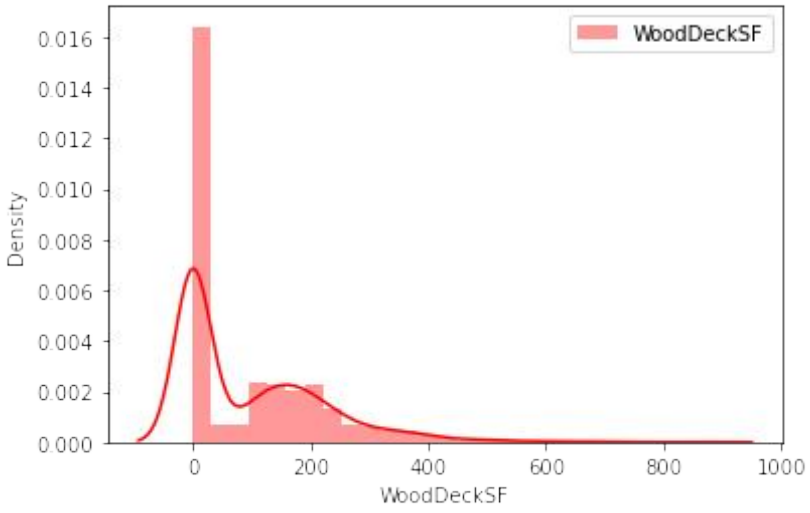
Have to check correlation with target columns by heatmap so get information about all features that how much correlated with target and also internal correlated or multicollinearity of features with each others.

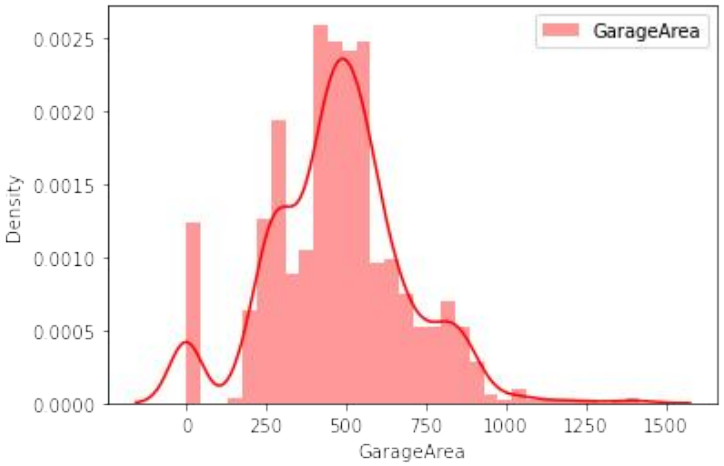
So, now we have mix features like category, numerical so in algorithm we have to fit only numerical format so convert categorical features into numerical by encoding them use of Ordinal encoder both train and test data set

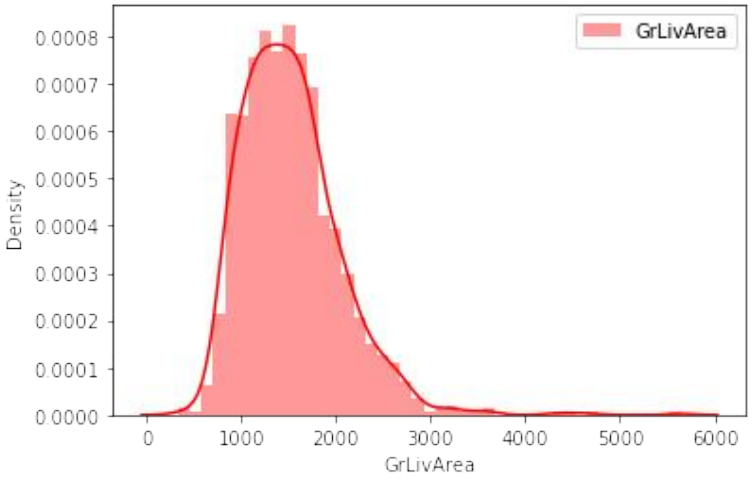
Skewness in data set show high percentage of that class or label in dataset so distribution is only one single high point that causes a machine learning algorithm to train only on that focus weights data points that causes model overfitting. Also linear model work on symmetric distribution so we have transform data in symmetric way so reduce RSME and improve score.

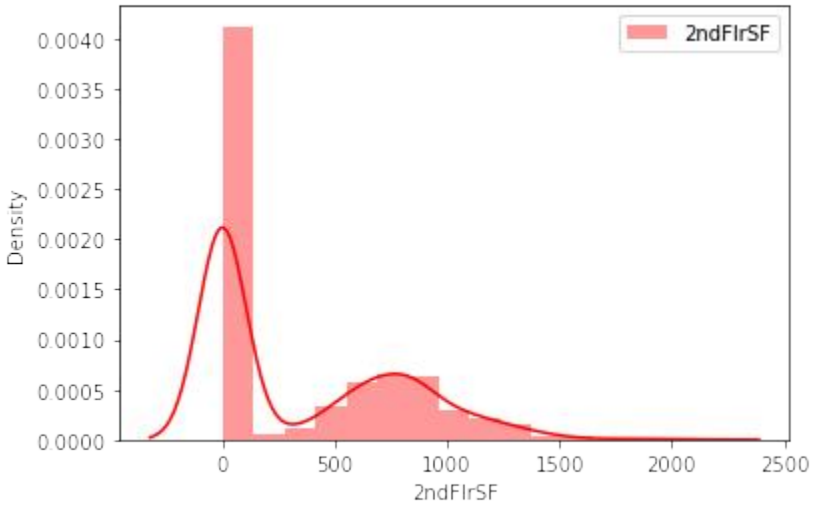
Distribution Plot of Skewed Features in dataset

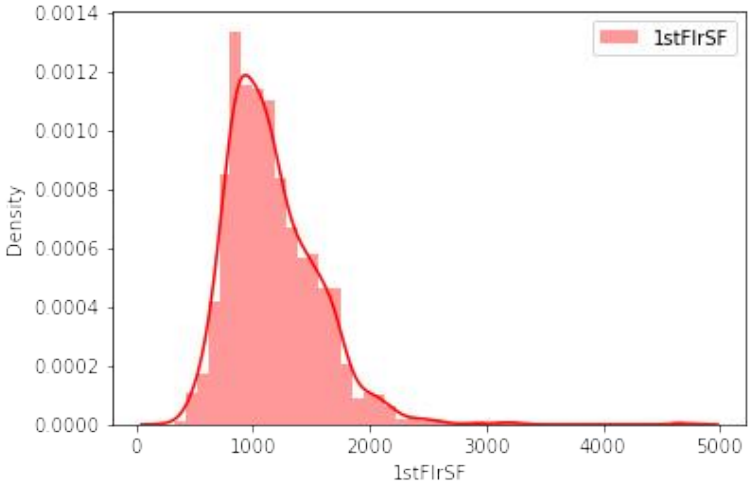


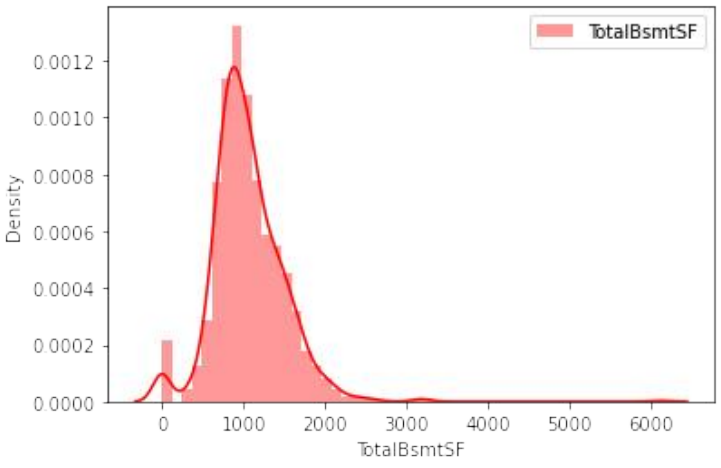

WoodDeckSF
Density
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
200
400
600
800
1000
WoodDeckSF

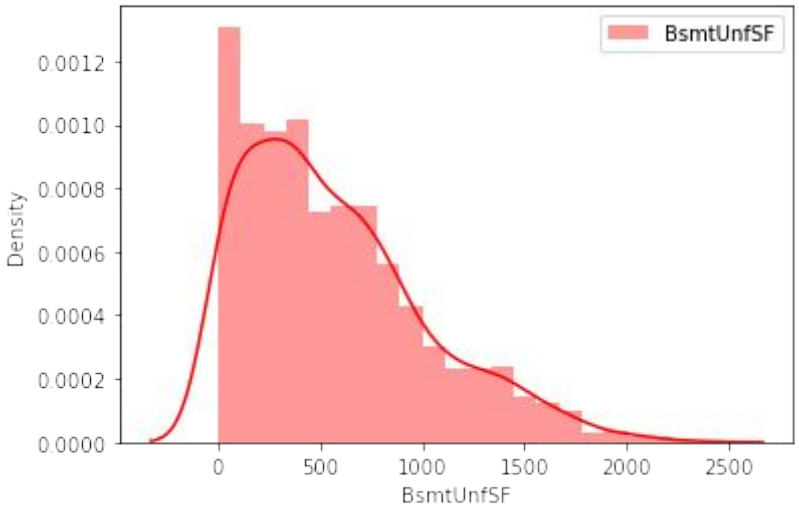

GarageArea
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
250
500
750
1000
1250
1500
GarageArea

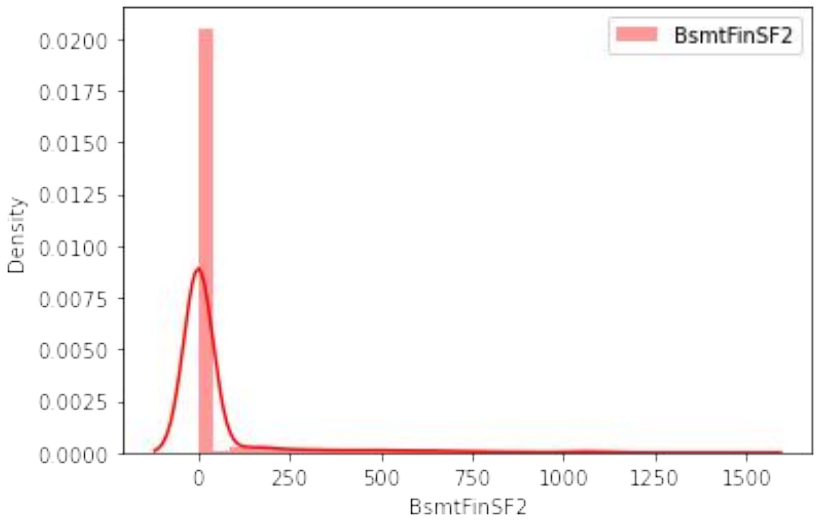

GrLivArea
Density
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
0
1000
2000
3000
4000
5000
6000
GrLivArea

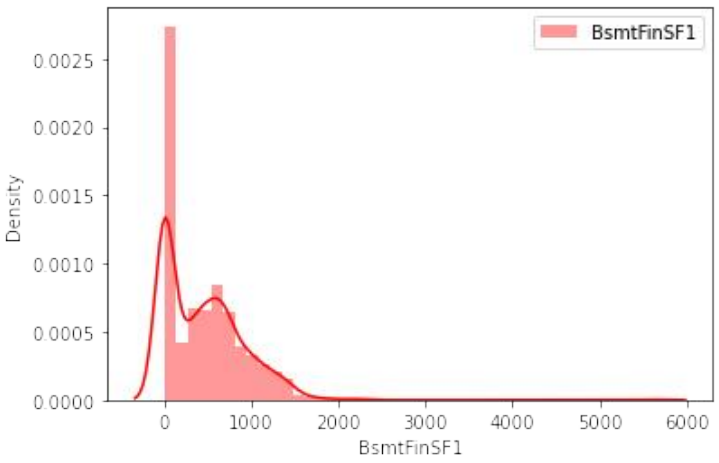

2ndFlrSF
Density
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
500
1000
1500
2000
2500
2ndFlrSF

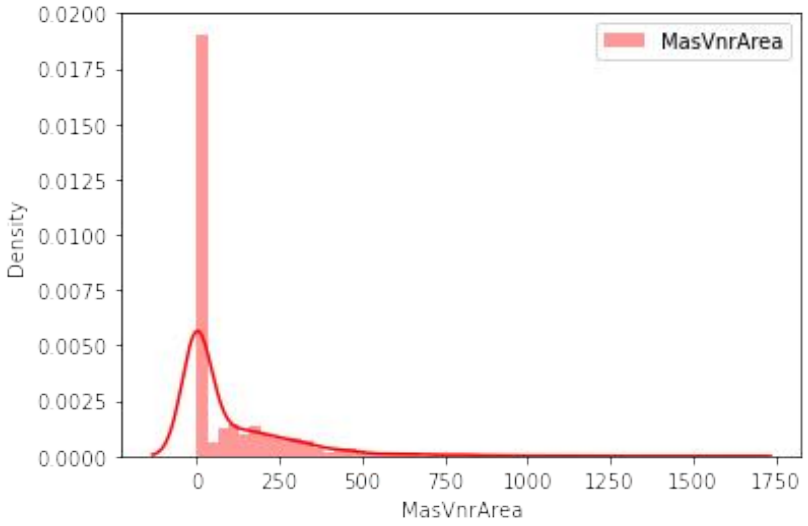

1stFlrSF
Density
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
5000
1stFlrSF

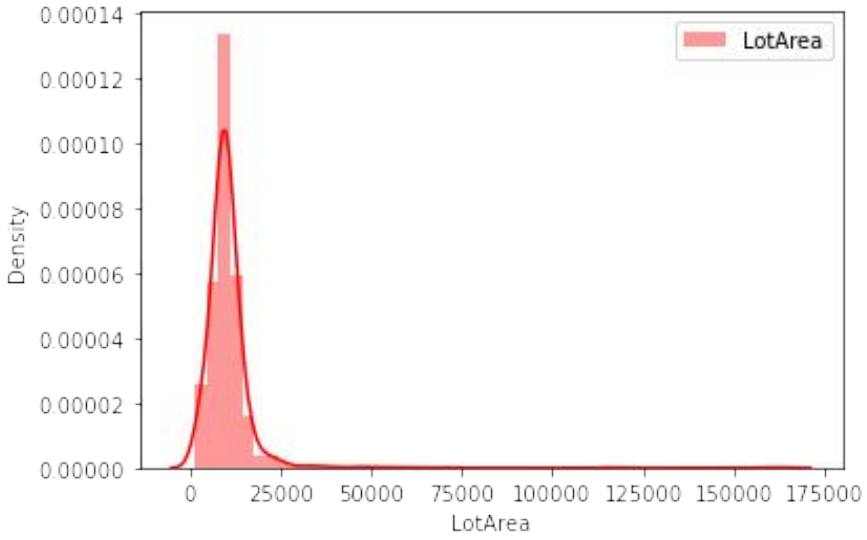

TotalBsmtSF
Density
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

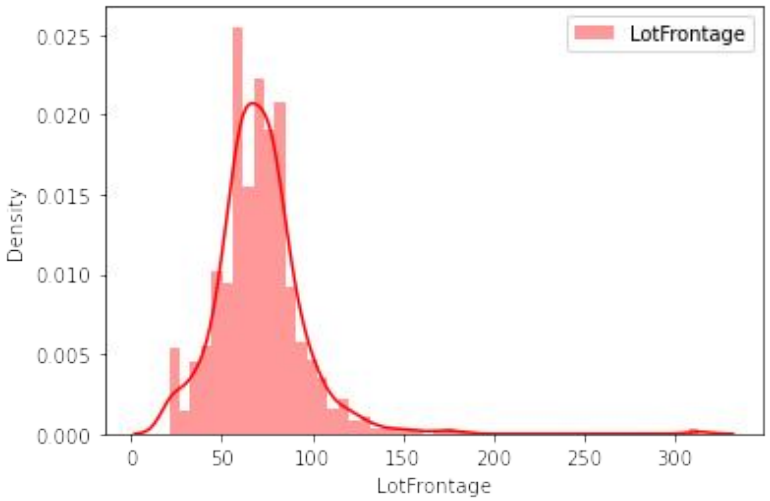

BsmtUnfSF
Density
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
500
1000
1500
2000
2500
BsmtUnfSF

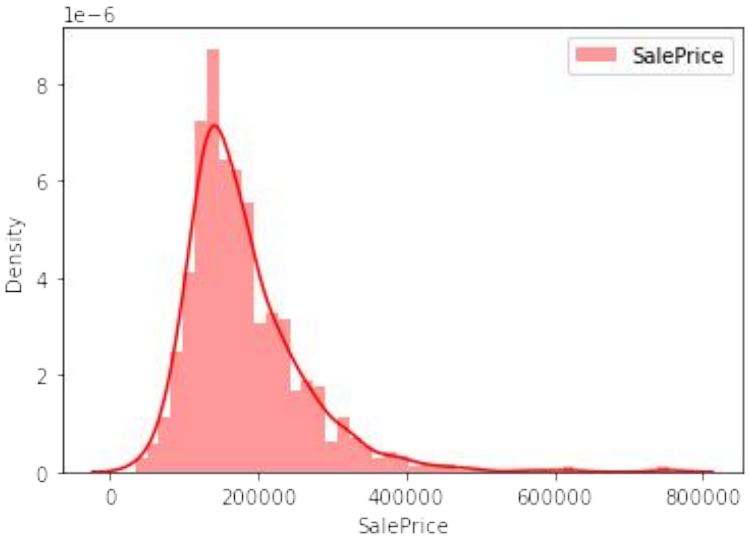

BsmtFinSF2
Density
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
250
500
750
1000
1250
1500
BsmtFinSF2


BsmtFinSF1
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
1000
2000
3000
4000
5000
6000
BsmtFinSF1


MasVnrArea
Density
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
250
500
750
1000
1250
1500
1750
MasVnrArea


LotArea
Density
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
25000
50000
75000
100000
125000
150000
175000
LotArea


LotFrontage
Density
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
300
LotFrontage



- In above graph all features have left skewed data in that mean is grater than median values.

```
# Remove skewness from train dataset
for i in countinus_features:
    train_df[i]=np.log1p(train_df[i])
```

```
# Remove skewness from test dataset
for i in countinus_test:
    test_df[i]=np.log1p(test_df[i])
```

```
train_df[countinus_features].skew()
```

```
LotFrontage     -0.793980
LotArea         -0.169107
MasVnrArea       0.518966
BsmtFinSF1      -0.606859
BsmtFinSF2       2.502785
BsmtUnfSF       -2.129882
TotalBsmtSF     -5.146574
1stFlrSF         0.105795
2ndFlrSF         0.289335
GrLivArea        0.001448
GarageArea      -3.508379
WoodDeckSF       0.131075
OpenPorchSF     -0.004279
EnclosedPorch    2.079610
ScreenPorch      3.104665
SalePrice        0.073610
dtype: float64
```

- Data Inputs- Logic- Output Relationships

To check relation with target column used a heatmap for visualization and correlation method to which particular feature affect positively to target and which feature affect target negatively, and also know how to individual feature correlated to each other so check multi-collinearity by Variance inflation factor.

Now from below code " Lotfrontage ,Lotarea, OverallQual, Yearbuilt, YearRemodAdd, MasVnrArea, BsmtFinSF1, TotalBsmtSF, 1stFlrSF, GrLivArea, FullBath, TotRmsAbvGrd, Fireplaces, GarageYrBlt, GarageCars, GarageArea, WoodDeckSF " are highly Positive correlated with target columns.

"LotShape,ExterQual,BsmtQual,HeatingQC,KitchenQual,GarageType,GarageFinish" are negatively correlation with House price.

- **CODE: -** train_df.corr()['SalePrice']

```
Id              -0.023897
MSSubClass      -0.060775
LotFrontage      0.323851
LotArea          0.249499
OverallQual      0.789185
OverallCond     -0.065642
YearBuilt        0.514408
YearRemodAdd     0.507831
MasVnrArea       0.460535
BsmtFinSF1       0.362874
BsmtFinSF2      -0.010151
BsmtUnfSF        0.215724
TotalBsmtSF      0.595042
1stFlrSF         0.587642
2ndFlrSF         0.330386
LowQualFinSF    -0.032381
GrLivArea        0.707300
BsmtFullBath     0.212924
BsmtHalfBath    -0.011109
FullBath         0.554988
HalfBath         0.295592
BedroomAbvGr     0.158281
KitchenAbvGr    -0.132108
TotRmsAbvGrd     0.528363
Fireplaces       0.459611
GarageYrBlt      0.453840
GarageCars       0.628329
GarageArea       0.619000
WoodDeckSF       0.315444
OpenPorchSF      0.339500
EnclosedPorch   -0.115004
3SsnPorch        0.060119
ScreenPorch      0.100284
PoolArea         0.103280
MiscVal         -0.013071
MoSold           0.072764
YrSold          -0.045508
SalePrice        1.000000
Name: SalePrice, dtype: float64
```

For the Multicollinearity we check VIF score of features

ter House Price predict Last Checkpoint: a day ago (autosaved)

lit View Insert Cell Kernel Widgets Help

Run Code

17]:
```
# Checking multi-colinearity
pd.set_option('display.max_rows',77)
vif=pd.DataFrame()
vif['vif']=[variance_inflation_factor(x1,i) for i in range(x1.shape[1])]
vif['columns']=x.columns
vif
```

17]:

	vif	columns
0	4.353366	Id
1	9.758535	MSSubClass
2	32.169303	MSZoning
3	34.381115	LotFrontage
4	46.322329	LotArea
5	255.901808	Street
6	3.715310	LotShape
7	21.145447	LandContour
8	NaN	Utilities
9	5.007110	LotConfig
10	1.532412	LandSlope
11	6.518410	Neighborhood
12	7.299729	Condition1
13	70.290759	Condition2
14	6.024842	BldgType
15	8.571691	HouseStyle
16	60.165988	OverallQual

18	20.921446	YearBuilt
19	6.783015	YearRemodAdd
20	5.011000	RoofStyle
21	4.645135	RoofMatl
22	38.418817	Exterior1st
23	36.427384	Exterior2nd
24	13.943194	MasVnrType
25	3.255569	MasVnrArea
26	37.583488	ExterQual
27	31.187896	ExterCond
28	10.716824	Foundation
29	18.922274	BsmtQual
30	22.017395	BsmtCond
31	7.845865	BsmtExposure
32	10.957589	BsmtFinType1
33	13.110164	BsmtFinSF1
34	98.546508	BsmtFinType2
35	4.673239	BsmtFinSF2
36	25.656929	BsmtUnfSF
37	93.807943	TotalBsmtSF
38	16.471119	Heating
39	3.064977	HeatingQC
40	25.927080	CentralAir
41	18.558703	Electrical

42	300.640728	1stFlrSF
43	34.717357	2ndFlrSF
44	1.595861	LowQualFinSF
45	637.110450	GrLivArea
46	3.497118	BsmtFullBath
47	1.302092	BsmtHalfBath
48	29.681373	FullBath
49	3.929972	HalfBath
50	37.734044	BedroomAbvGr
51	45.792194	KitchenAbvGr
52	18.630796	KitchenQual
53	47.042270	TotRmsAbvGrd
54	41.757643	Functional
55	3.624622	Fireplaces
56	11.478016	FireplaceQu
57	4.895830	GarageType
58	9.996215	GarageYrBlt
59	7.078315	GarageFinish
60	28.319738	GarageCars
61	57.828600	GarageArea
62	69.821337	GarageQual
63	94.536507	GarageCond
64	22.718761	PavedDrive
65	2.629083	WoodDeckSF

65	2.629083	WoodDeckSF
66	3.428053	OpenPorchSF
67	1.641491	EnclosedPorch
68	1.079503	3SsnPorch
69	1.241820	ScreenPorch
70	1.127046	PoolArea
71	1.096505	MiscVal
72	5.373364	MoSold
73	3.172602	YrSold
74	24.867167	SaleType
75	15.132985	SaleCondition

From that table pic we show Vif score of individual features and with referance with heatmap we chech which is more related with taget columns and drop other which less correlated.

MSSubclass and Bldgtype are highly collinear with each other and have equal values so choose one from that.

OverallQual and Yearbult negatively with each other because year spend house quality decrease

OverallQual and YearRemodAdd are positive correlated with each other.

Extrior1st and Exterior2nd are highly correlated with each other have Vif score 38.4 and 36.4 accordingly equally correlated with target so we assume one material use in house drop Exterior2nd

GarageCars and Garagearea are high correlated with each other and also positive with target column so we assume NO.of cars parked in garage are necessary so drop other one.

Consider that multicollinear columns that drop that column "MSSubClass','Condition1','TotRmsAbvGrd','Utilities','YearBuilt','2ndFlr SF','Exterior2nd','BsmtUnfSF','GarageArea','OverallCond"

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

- Data is missing completely at random (MCAR)
- The missing observations, most likely look like the majority of the observations in the variable (aka, the mean/median)
- If data is missing completely at random, then it is fair to assume that the missing values are most likely very close to the value of the mean or the median of the distribution, as these represent the most frequent/average observation.
- Frequency Count Imputation (Mode)
- Frequent category imputation—or mode imputation—consists of replacing all occurrences of missing values (NA) within a variable with the mode, or the most frequent value.
- This method is suitable for numerical and categorical variables, but in practice, we use this technique with categorical variables.
- You can use this method when data is missing completely at random, and no more than 5% of the variable contains missing data.
- **Most common causes of outliers on a data set:**
  Box Plot
  Scatter Plot
  Z-Score
  IQR Score
- **Algorithms that are NOT sentitive to outliers**
- Naive Bayes
- SVM
- Decision Trees
- Random Forest
- XGBoost, GBM
- KNN
- **Algorithms that are sentitive to outliers**

- Linear Regression
- Logistic Regression
- K-Means Clustering
- Hierarchical Clustering
- PCA
- Neural Networks

Feature transformation is the process of modifying your data but keeping the information. These modifications will make Machine Learning algorithms understanding easier, which will deliver better results.

- **Min-Max Scaler**
- The MinMax scaler is one of the simplest scalers to understand. It just scales all the data between 0 and 1.
- FORMULA: x_scaled = (x – x_min)/(x_max – x_min)

- **Feature Selection Methods:**

1. Univariate Selection
2. Feature Importance
3. Correlation Matrix with Heatmap

We use Correlation Matrix with Heatmap

Correlation states how the features are related to each other or the target variable. Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable).Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

- Device and Library Use
- Hardware and Software Requirements and Tools Used
  - Device name: HP Pavilion
  - Processor: AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
  - RAM: 8.00 GB
  - System type: 64-bit operating system, x64-based processor
  - Edition: Windows 10 Home Single Language
  - Version: 21H1
  - OS build: 19043.1288
  - Jupyter NoteBooks Version : 6.4.3
  - Python3 vesrion : 3.8.8

- Required Libraries Used For Dataset
  - import pandas as pd
  - import numpy as np
  - import matplotlib.pyplot as plt
  - %matplotlib inline
  - import seaborn as sns
  - import pandas_profiling as ProfileReport
  - import IPython.display as display
  - import missingno as msno
  - import warnings
  - warnings.filterwarnings('ignore')
  - from sklearn.impute import SimpleImputer
  - from sklearn.preprocessing import OrdinalEncoder,MinMaxScaler
  - from statsmodels.stats.outliers_influence import variance_inflation_factor
  - from sklearn.linear_model import LinearRegression,SGDRegressor,Lasso,Ridge,BayesianRidge,ElasticNet
  - from sklearn.tree import DecisionTreeRegressor
  - from sklearn.svm import SVR
  - from sklearn.neighbors import KNeighborsRegressor
  - from xgboost import XGBRegressor,XGBRFRegressor
  - from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor,BaggingRegressor
  - from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV,KFold
  - from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
  - from sklearn.feature_selection import SelectFromModel
  - import joblib

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)
  - LinearRegression()
  - DecisionTreeRegressor()
  - SVR(kernel="linear",C=0.1)
  - KNeighborsRegressor(n_neighbors=10)
  - Lasso(alpha=0.001,fit_intercept=True,normalize=False,precompute=True)
  - Ridge(alpha=1,solver="saga")
  - ElasticNet(alpha=0.001,max_iter=10)
  - BayesianRidge()
  - SGDRegressor()
  - XGBRegressor()
  - XGBRFRegressor()
  - RandomForestRegressor()
  - AdaBoostRegressor()
  - GradientBoostingRegressor()
  - BaggingRegressor()
- Run and Evaluate selected models:

  First we have select best random state by this code for avoid Over and under fitting

```
for i in range(1,500):
    x_train,x_test,y_train,y_test=train_test_split(a,y,test_size=0.30,random_state=i)
    lr.fit(x_train,y_train)
    pred_tr=lr.predict(x_train)
    pred_te=lr.predict(x_test)
    if round(r2_score(y_train,pred_tr)*100,1)==round(r2_score(y_test,pred_te)*100,1):
        print('Random state',i)
        print('Train score',r2_score(y_train,pred_tr)*100)
        print('Test score',r2_score(y_test,pred_te)*100)
```

```
Random state 18
Train score 88.34598422121698
Test score 88.31647186948834
Random state 144
Train score 88.14106709511054
Test score 88.11891400218718
Random state 171
Train score 88.16944900283956
Test score 88.1799847941863
Random state 182
Train score 88.01441980752627
Test score 87.96596345995053
Random state 206
Train score 88.12863419749637
Test score 88.09160896665342
Random state 330
Train score 88.21480383786091
Test score 88.20992558669238
Random state 416
Train score 88.07703651538866
Test score 88.12160567686298
Random state 428
Train score 88.41947387947519
```

```
x_train,x_test,y_train,y_test=train_test_split(a,y,test_size=0.30,random_state=428)
lr.fit(x_train,y_train)
pred_lr=lr.predict(x_test)
print('train score',lr.score(x_train,y_train)*100)
print('R2 score',r2_score(y_test,pred_lr)*100)
print('RMSE',np.sqrt(mean_squared_error(y_test,pred_lr)))
print('Absolute error',mean_absolute_error(y_test,pred_lr))
```

```
train score 88.41947387947519
R2 score 88.41019891488016
RMSE 0.1298902532695707
Absolute error 0.0974747373569047
```

```
# Regularization
def regul(f):
    f.fit(x_train,y_train)
    print(f,'\n\t',f.score(x_train,y_train)*100)
    pred=f.predict(x_test)
    print('R2_score :',r2_score(y_test,pred)*100)
    print('error1:\n:',mean_absolute_error(y_test,pred))
    print('RSME:\n:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
regul(l1)
```

```
Lasso(alpha=0.001, precompute=True)
         87.5504116572219
R2_score : 88.06625292541139
error1:
: 0.09848292455455741
RSME:
: 0.13180351337816715
```

```
regul(l2)
```

```
Ridge(alpha=1, solver='saga')
         88.28989179112739
R2_score : 88.23966071294048
error1:
: 0.0985149444164677
RSME:
: 0.130842398951749
```

```
regul(l3)
```

```
ElasticNet(alpha=0.001, max_iter=10)
         87.74225946552806
R2_score : 88.57155357409037
error1:
: 0.09656336362401148
RSME:
: 0.12898290999421322
```

```
regul(l4)
```

```
BayesianRidge()
         88.29949608173077
R2_score : 88.25237609710194
error1:
: 0.09842303577380243
```

```python
# Other Regressors
def regressor(f):
    f.fit(x_train,y_train)
    print(f,'\n',f.score(x_train,y_train)*100)
    pred=f.predict(x_test)
    print('error1:\n',mean_absolute_error(y_test,pred))
    print('RSME:\n',np.sqrt(mean_squared_error(y_test,pred)))
    print('r2 score:\n',r2_score(y_test,pred)*100)
```

```python
regressor(dtc)
```

```
DecisionTreeRegressor()
 100.0
error1:
 0.16892358090239679
RSME:
 0.23335603319068043
r2 score:
 62.59230726101026
```

```python
regressor(svr)
```

```
SVR(C=0.1, kernel='linear')
 87.43585163923788
error1:
 0.0985001033540667
RSME:
 0.13087817935664825
r2 score:
 88.23322782513374
```

```python
regressor(kn)
```

```
KNeighborsRegressor(n_neighbors=10)
 75.68275890990849
error1:
 0.15518401134229967
RSME:
 0.20944565895799844
r2 score:
 69.86538953499044
```

```python
regressor(sgd)
```

```
SGDRegressor()
 6.247295214512927
error1:
 0.29383417131682193
RSME:
 0.39771061593287027
r2 score:
 -8.656877940338536
```

```
: regressor(xgb)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
 99.99118203541902
error1:
 0.10632605034605996
RSME:
 0.1451112080343192
r2 score:
 85.53478909290165
```

```
: regressor(xgbr)
```

```
XGBRFRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
               colsample_bytree=1, gamma=0, gpu_id=-1, importance_type='gain',
               interaction_constraints='', max_delta_step=0, max_depth=6,
               min_child_weight=1, missing=nan, monotone_constraints='()',
               n_estimators=100, n_jobs=8, num_parallel_tree=100,
               objective='reg:squarederror', random_state=0, reg_alpha=0,
               scale_pos_weight=1, tree_method='exact', validate_parameters=1,
               verbosity=None)
 94.43194088390064
error1:
 0.10792937549116083
RSME:
 0.1481063221804689
r2 score:
 84.93149903997168
```

```
score:  BaggingRegressor() :
 96.88354207009546
 R2 score: 83.0095056496773
error1:
 0.11188380771560742
RSME:
 0.15726844042534321
```

```
for i in ensemble:
    i.fit(x_train,y_train)
    print('\n \nscore: ',i,':\n',i.score(x_train,y_train)*100)
    pred=i.predict(x_test)
    print(' R2 score:',r2_score(y_test,pred)*100)
    print('error1:\n',mean_absolute_error(y_test,pred))
    print('RSME:\n',np.sqrt(mean_squared_error(y_test,pred)))
```

```
score:  RandomForestRegressor() :
 97.91743119843316
 R2 score: 85.97496535116736
error1:
 0.10135131668986914
RSME:
 0.14288628469137704


score:  AdaBoostRegressor() :
 85.90087836265121
 R2 score: 80.13237615750224
error1:
 0.13029367490982421
RSME:
 0.17006366605129875


score:  GradientBoostingRegressor() :
 96.43205251608936
 R2 score: 88.54539932969499
error1:
 0.09592428081258705
RSME:
 0.12913041572583414
```

```
# Check Cross validation for Over/Under fitting
scores=cross_val_score(gd,a,y,cv=KFold(shuffle=True,random_state=428),scoring='r2')
score=np.mean(scores)
std=np.std(scores)
print('CV mean',score)
print('std:',std)
```

```
CV mean 0.8838225504433733
std: 0.01297130616090728
```

Select Best Model base Less RSME and Test score and cross validation score Gradient Boosting Algorithm working best for dataset

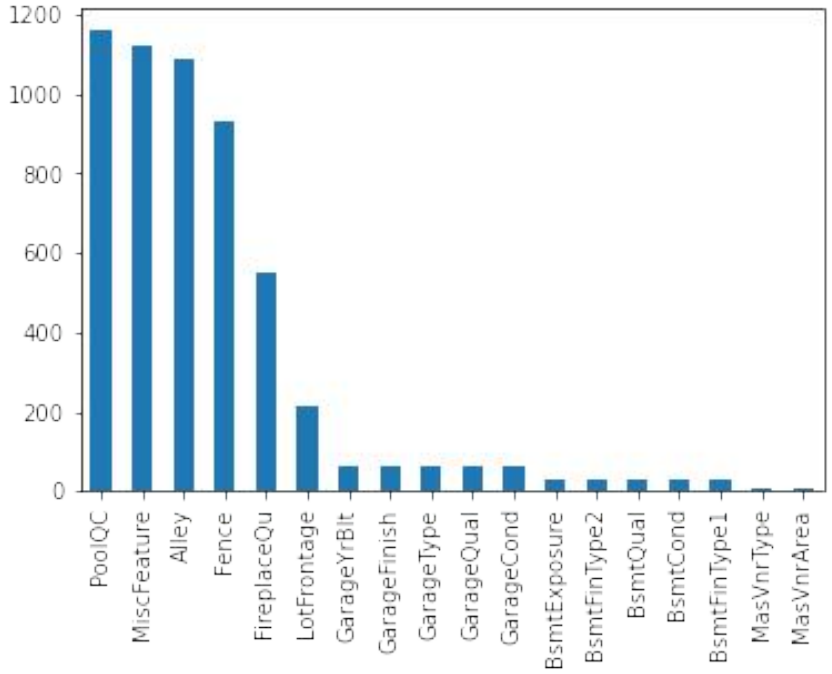
```
# Hypertuning best model for imrove efficiency
pr={'learning_rate':[0.1,0.01,1],
    'n_estimators':[1000,500,100],
    'criterion':['friedman_mse','squared_error','mse','mae'],
    'max_features':['auto','sqrt','log2']}
grid = GridSearchCV(GradientBoostingRegressor(),pr,cv=5,scoring='r2')
grid.fit(a,y)
print(grid.best_params_)
```

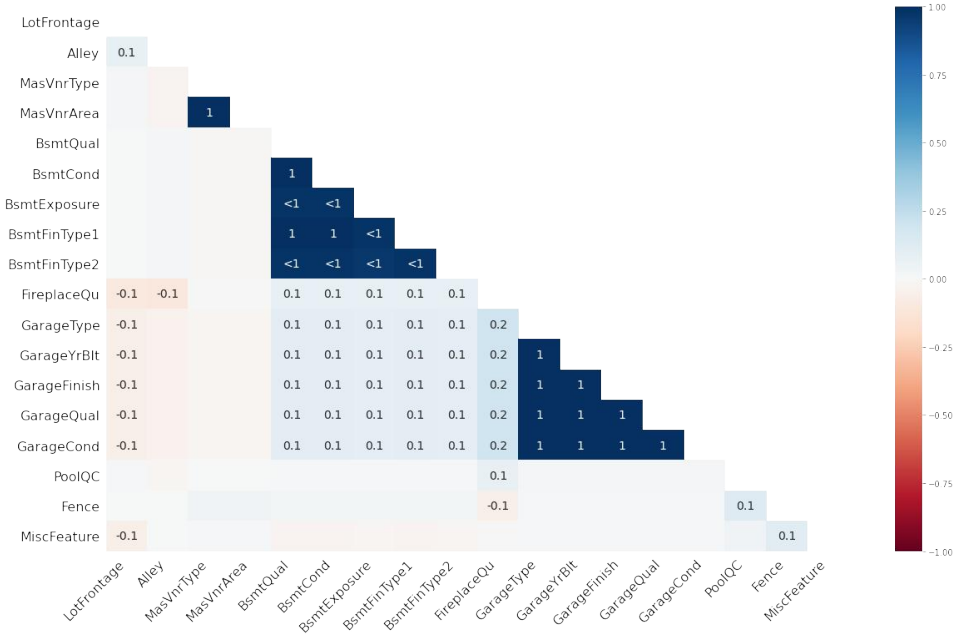
```
{'criterion': 'friedman_mse', 'learning_rate': 0.01, 'max_features': 'log2', 'n_estimators': 1000}
```

```
# Final model for prediction
gd=GradientBoostingRegressor(criterion='friedman_mse',learning_rate=0.01,max_features='log2',n_estimators=1000)
gd.fit(x_train,y_train)
pred_gd=gd.predict(x_test)
print("score",gd.score(x_train,y_train)*100)
print('R2_score :',r2_score(y_test,pred_gd)*100)
print('error1:\n:',mean_absolute_error(y_test,pred_gd))
print('RSME:\n:',np.sqrt(mean_squared_error(y_test,pred_gd)))
```

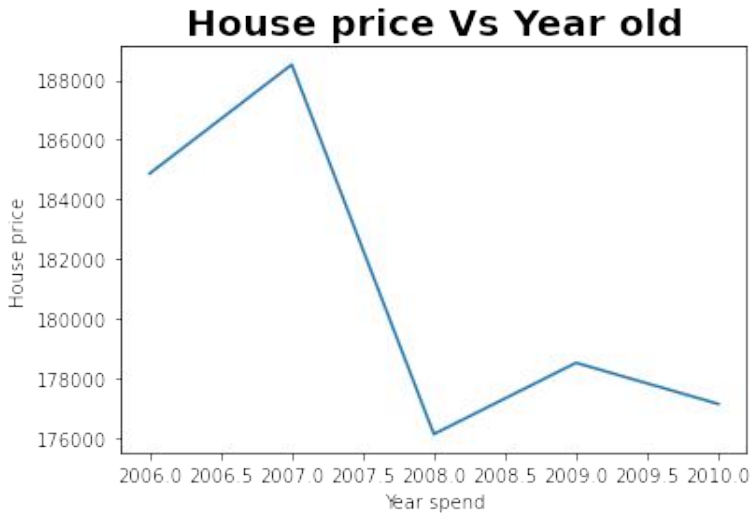
```
score 95.47459077793525
R2_score : 89.45220882899599
error1:
: 0.09102748295892088
RSME:
: 0.12391370255316776
```

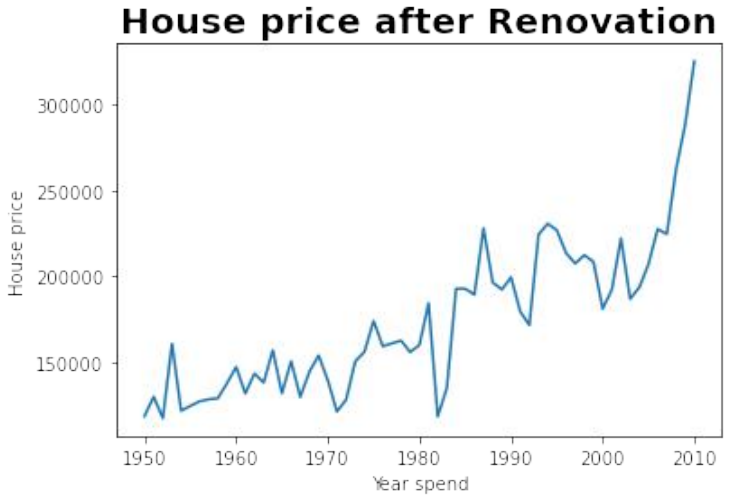
- Key Metrics for success in solving problem under consideration
  - Here We used Root mean squared error metrics
  - Absolute mean error metrics
- Visualizations
  - Count plot
  - Bar plot
  - Distribution plot
  - Scatter plot
  - Box plot
  - Correlation matrix Heatmap
  - Line plot





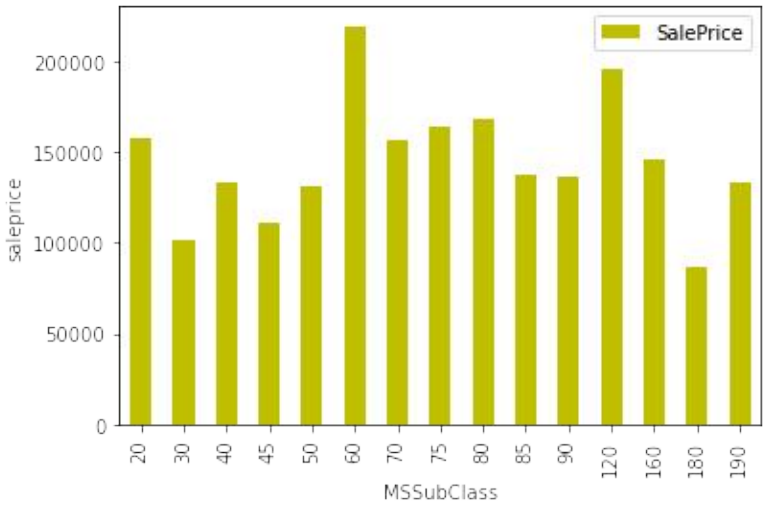
- Above graph represent the Missing values and correlation with missing values with each other's.
- PoolQC, MiscFeature, Alley, Fence have more 50% to 98% Null values so drop that column also they columns have not correlation with others.

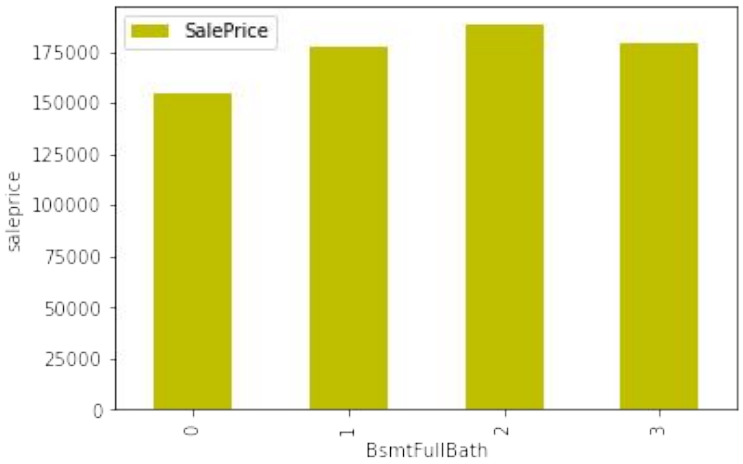


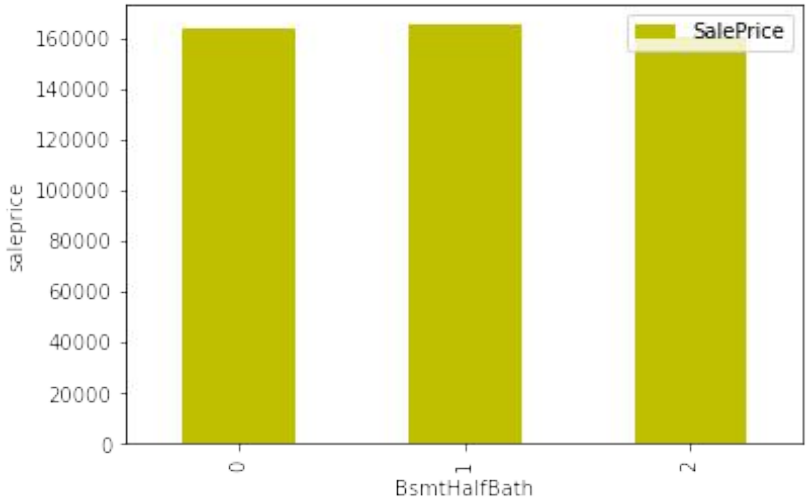


- Above graph show the House price with respect to year olds In 1st graph saw that price continues decrease with year's old.
- 2nd graph we saw price increasing with Renovation done in year.
- 3rd graph most of 4 to 10 months period price is continuously increasing.

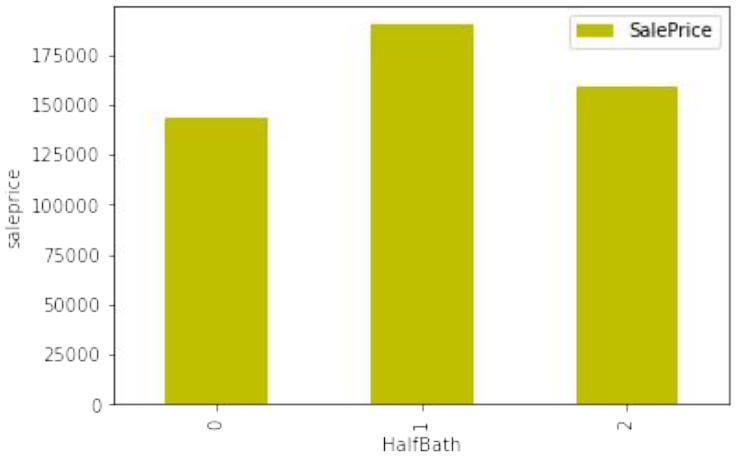
Discrete Features Affect Sale Price



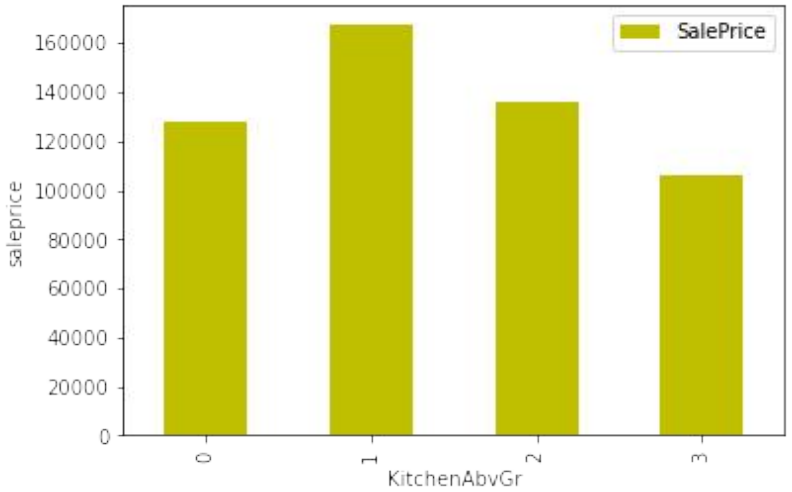

SalePrice
saleprice
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
BsmtFullBath

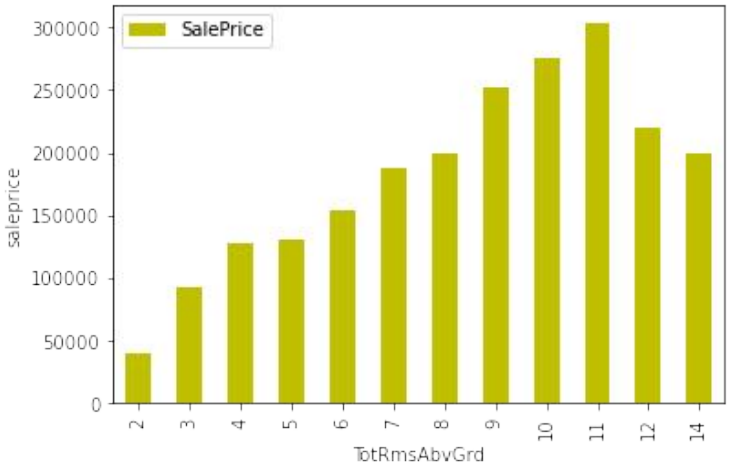

SalePrice
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
BsmtHalfBath

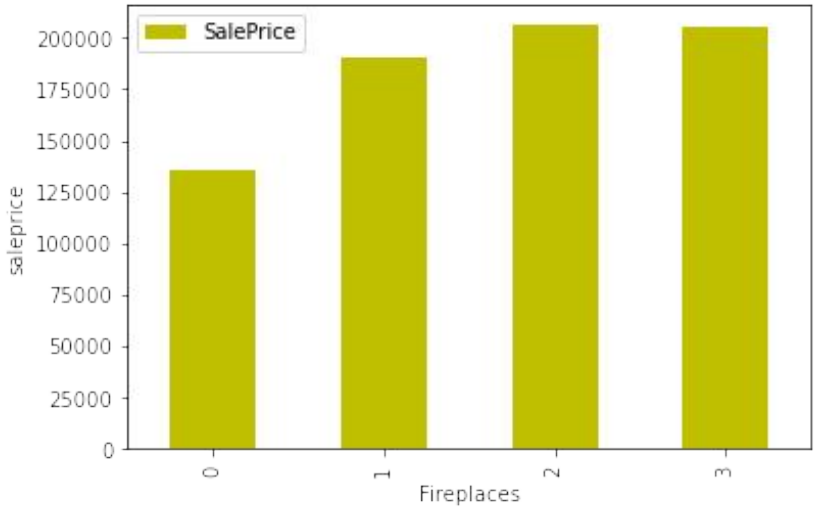

SalePrice
saleprice
300000
250000
200000
150000
100000
50000
0
0
1
2
3
FullBath

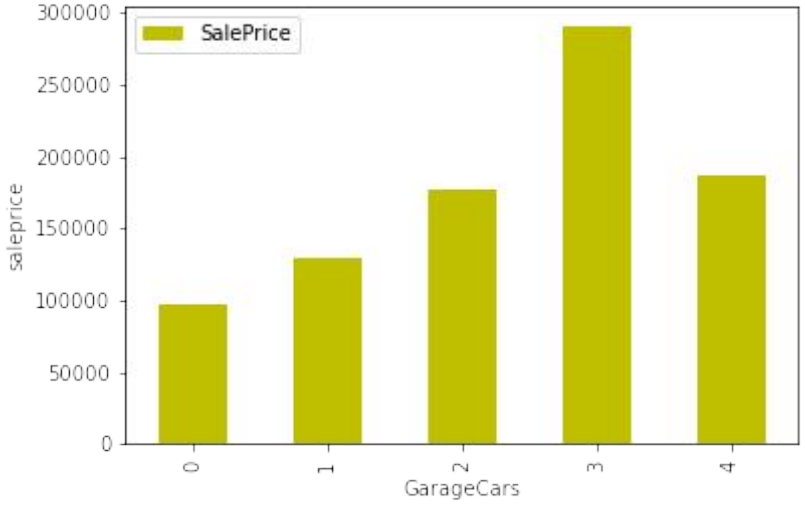

SalePrice
saleprice
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
HalfBath

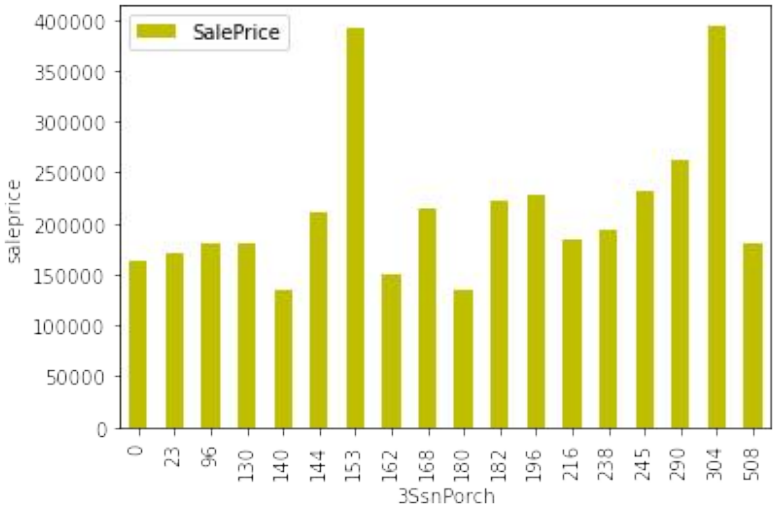

SalePrice
saleprice
250000
200000
150000
100000
50000
0
0
1
2
3
4
5
6
8
BedroomAbvGr

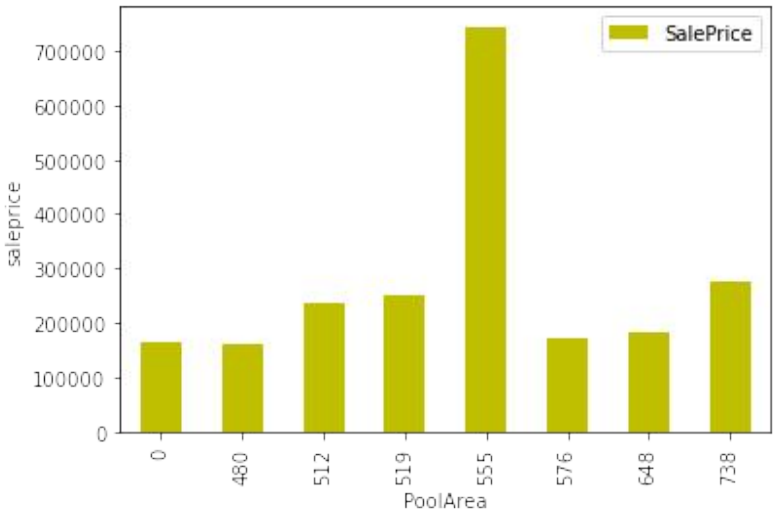

SalePrice
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
2
3
KitchenAbvGr


SalePrice
saleprice
300000
250000
200000
150000
100000
50000
0
2
3
4
5
6
7
8
9
10
11
12
14
TotRmsAbvGrd


SalePrice
saleprice
200000
175000
150000
125000
100000
75000
50000
25000
0
0
1
2
3
Fireplaces


SalePrice
saleprice
300000
250000
200000
150000
100000
50000
0
0
1
2
3
4
GarageCars

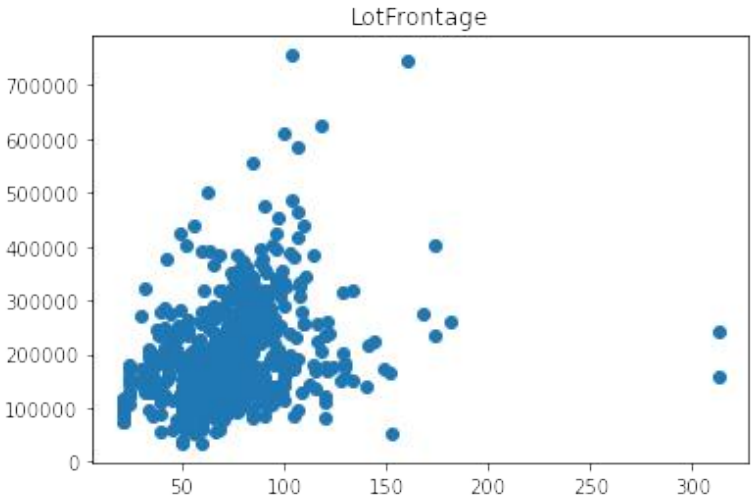


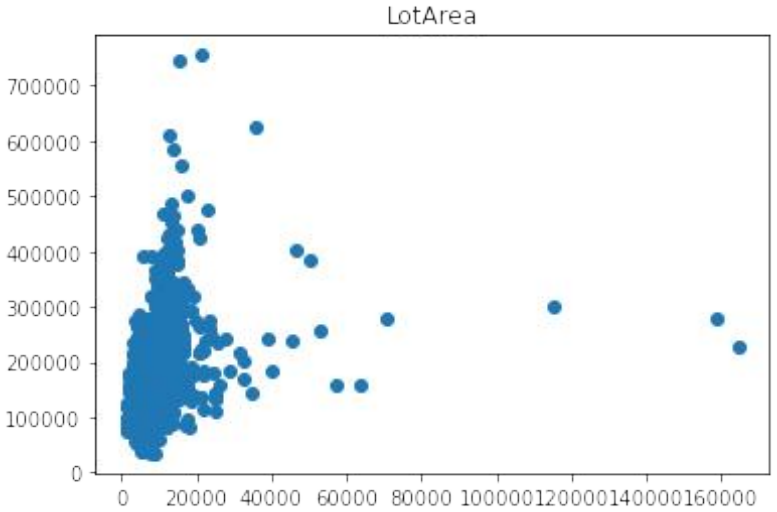


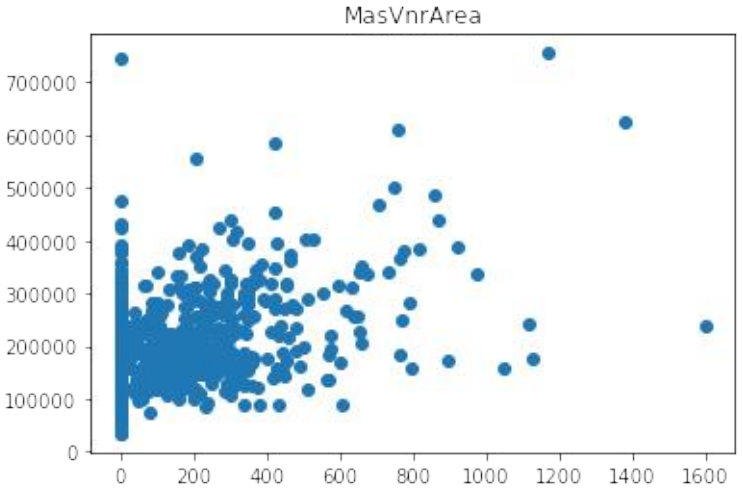
- From Above graph show house sales price with discrete features of data set
- 2-STORY 1946 & NEWER and 1-STORY PUD (Planned Unit Development) - 1946 & NEWER have max high house price up 200000/-
- 2-STORY 1945 & OLDER,2-1/2 STORY ALL AGES, SPLIT OR MULTI-LEVEL, SPLIT FOYER, DUPLEX - ALL STYLES AND AGES have Average price around 150000/-
- Over all Quality increase affect price positively that increase price as well
- Over all condition between 5 to 10 high prices above 125000/-
- Bsmtfullbath and Bsmthalfbath doesn’t affect price much have same
- 3 Fullbath has max price

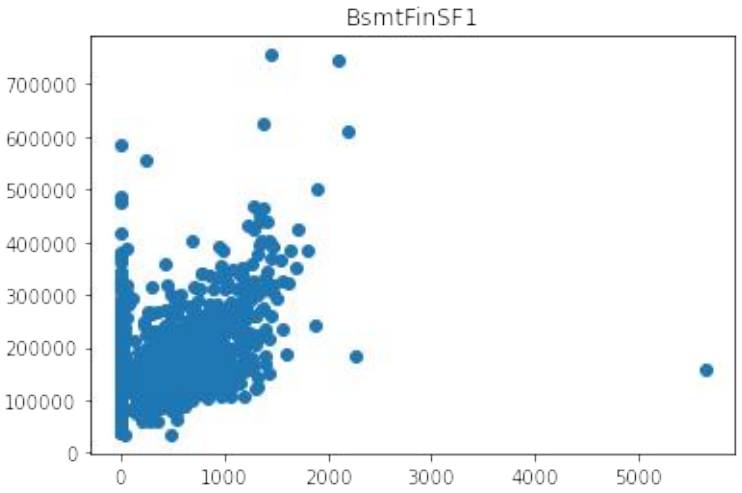
- Total rooms increase price also increase at 10 rooms than price decrease by increase rooms
- Most of 3 cars park in garage space have high price
- Poolarea 555 area have max price

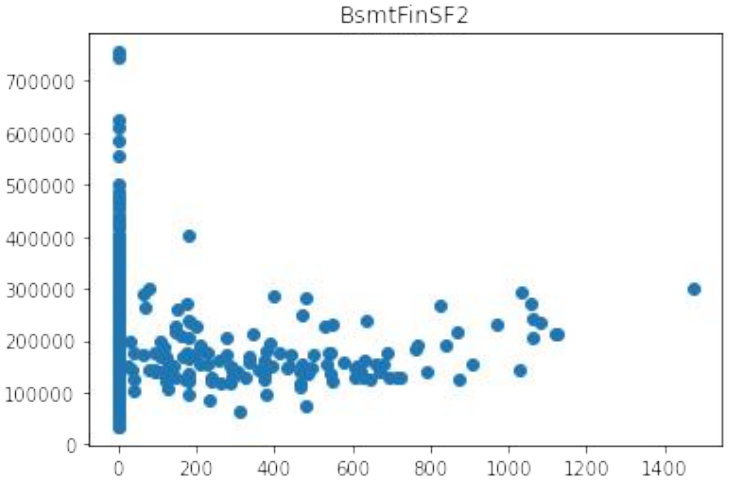
Continuous Features Vs Sale Price

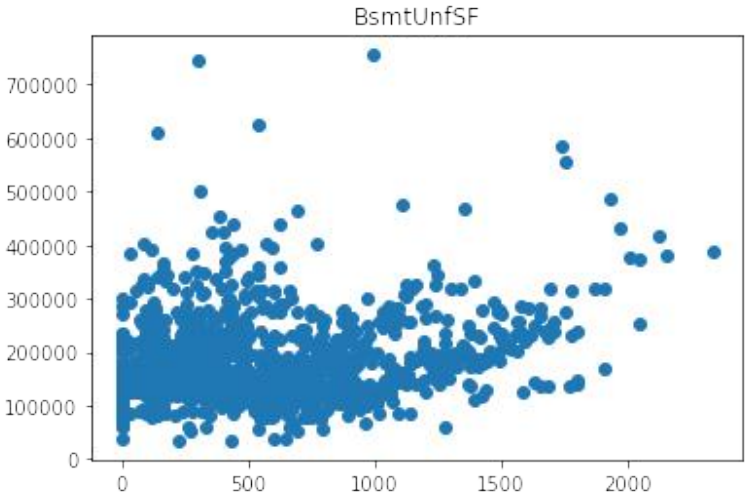


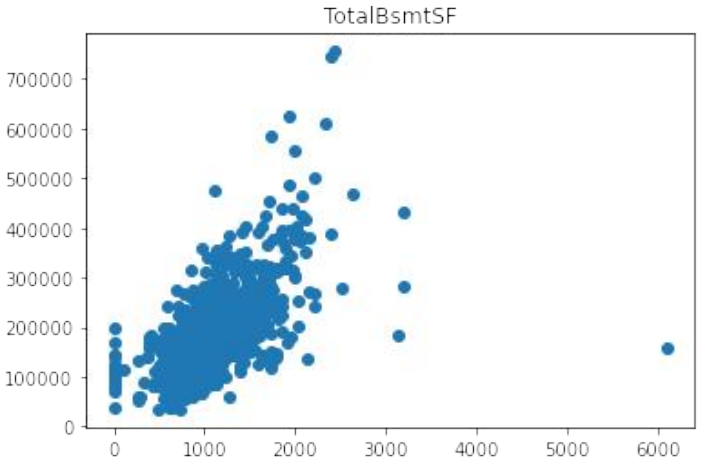


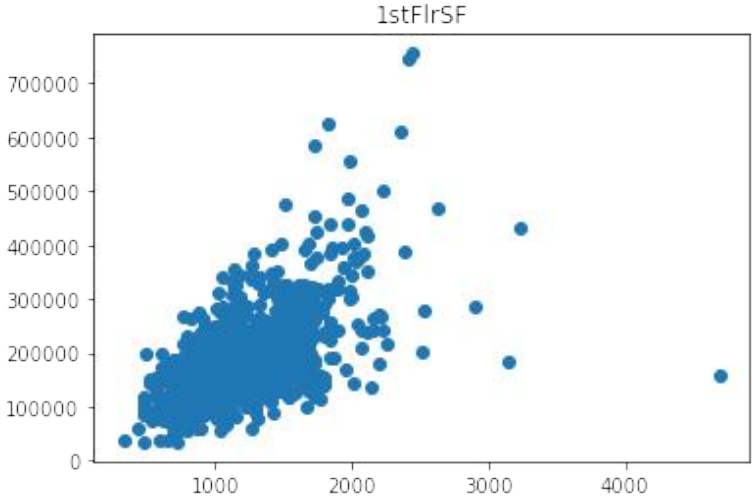


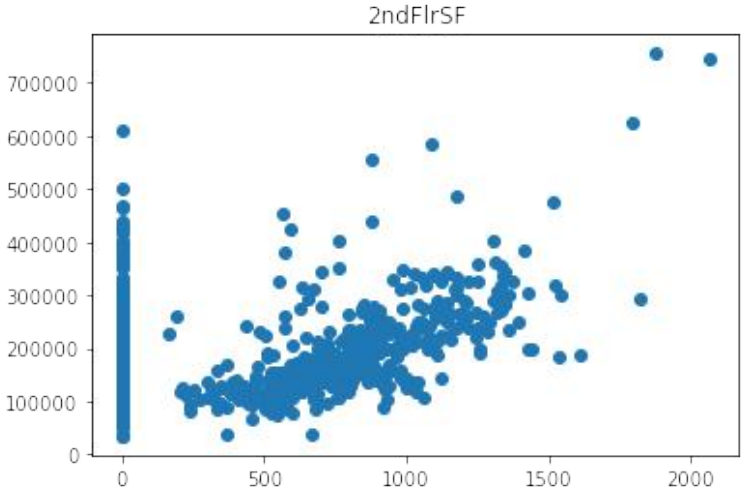

BsmtFinSF1
700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000

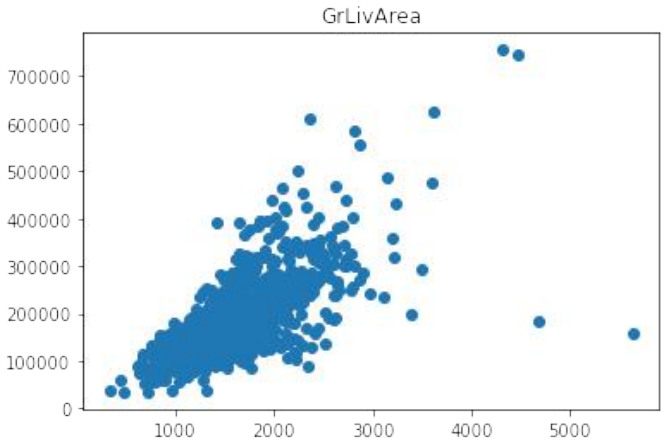

BsmtFinSF2
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800
1000
1200
1400

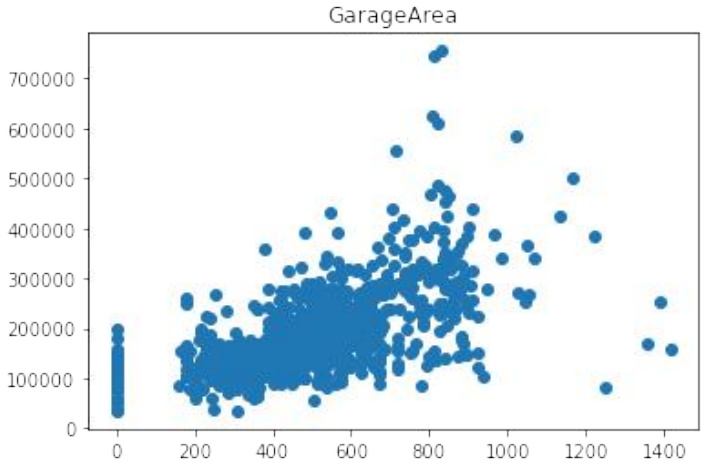

BsmtUnfSF
700000
600000
500000
400000
300000
200000
100000
0
0
500
1000
1500
2000

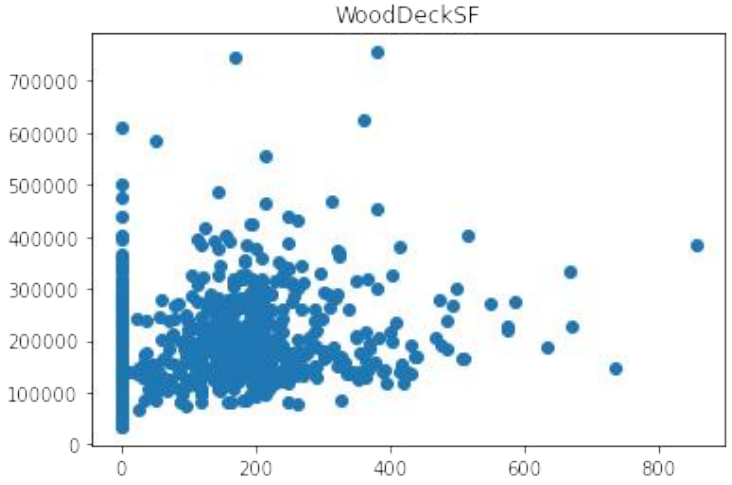

TotalBsmtSF
700000
600000
500000
400000
300000
200000
100000
0
0
1000
2000
3000
4000
5000
6000

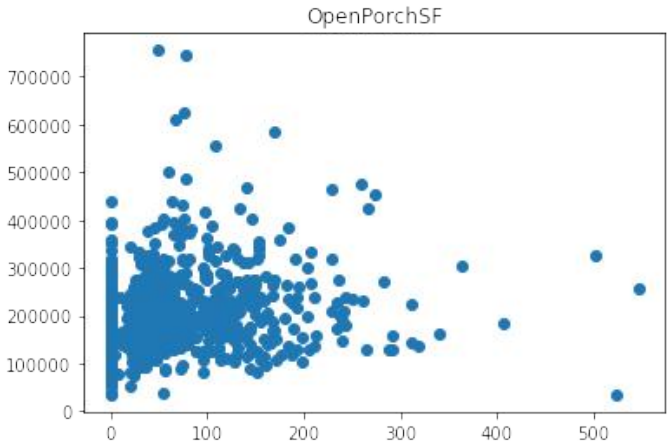

1stFlrSF
700000
600000
500000
400000
300000
200000
100000
0
1000
2000
3000
4000

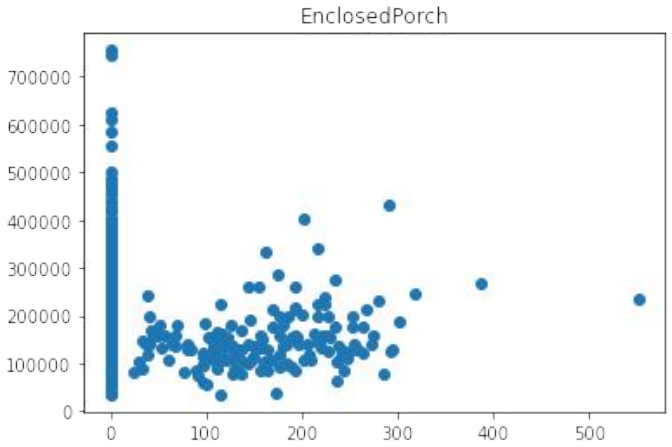

2ndFlrSF
700000
600000
500000
400000
300000
200000
100000
0
0
500
1000
1500
2000

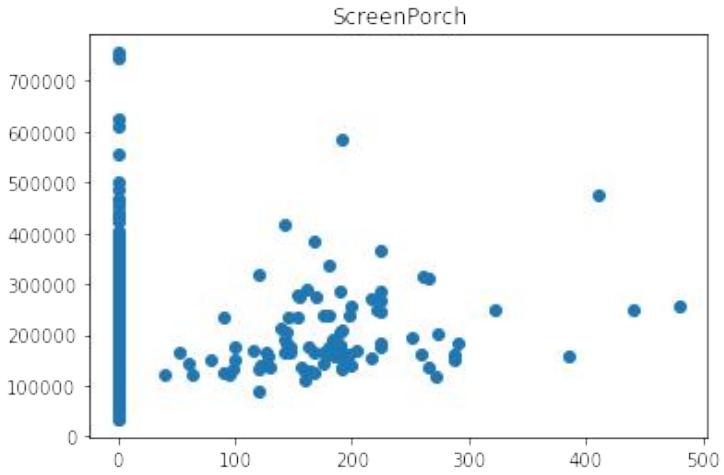

GrLivArea
700000
600000
500000
400000
300000
200000
100000
0
1000
2000
3000
4000
5000


GarageArea
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800
1000
1200
1400


WoodDeckSF
700000
600000
500000
400000
300000
200000
100000
0
0
200
400
600
800

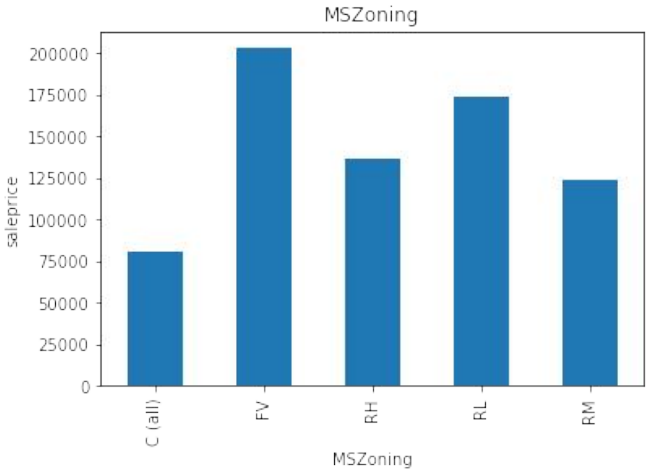


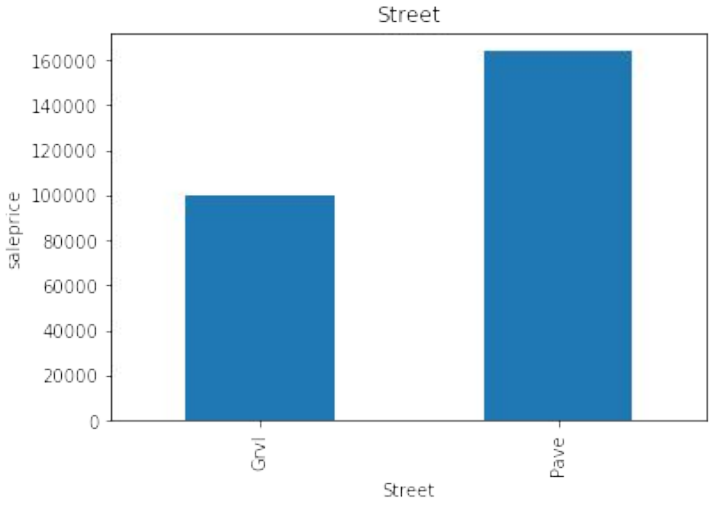


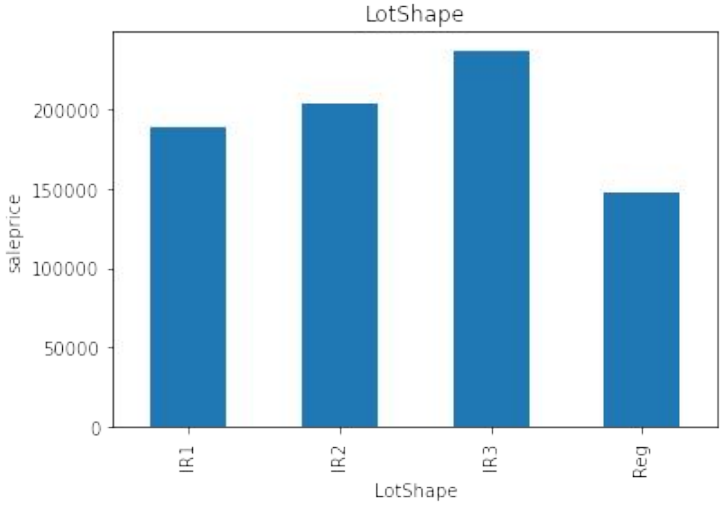


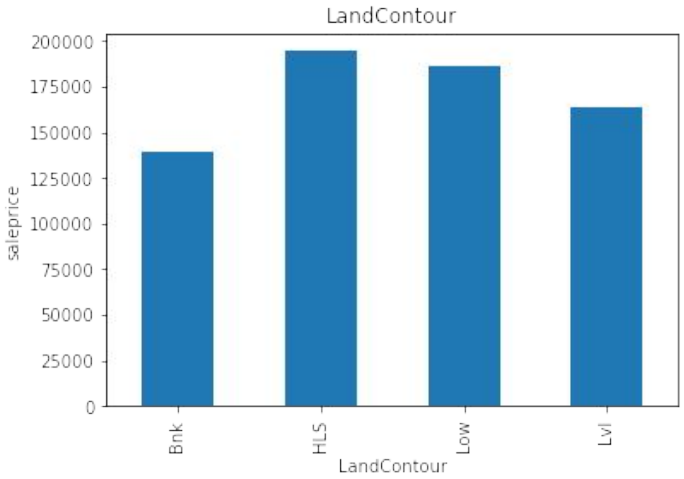
- From the scatter plot we saw that price linearly increase with some features like Garage area, Grlivearea, 1stflrSF, 2ndfltSF, TotBsmtSF, LotFrontage.

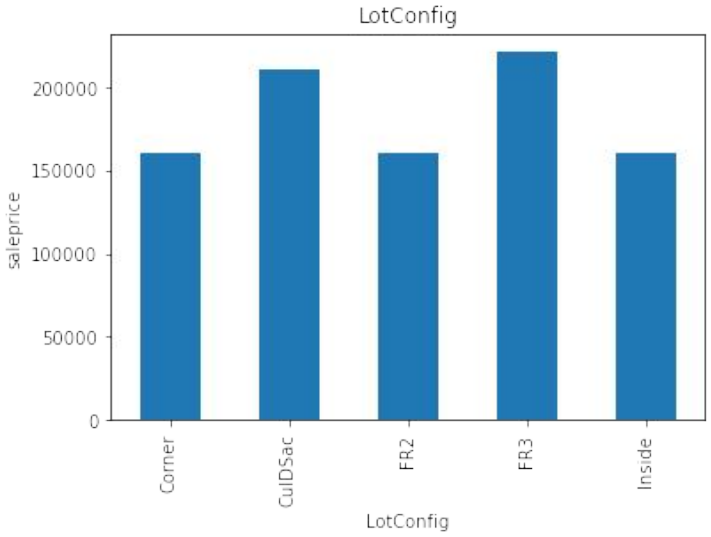
Categorical Features affect Sale Price

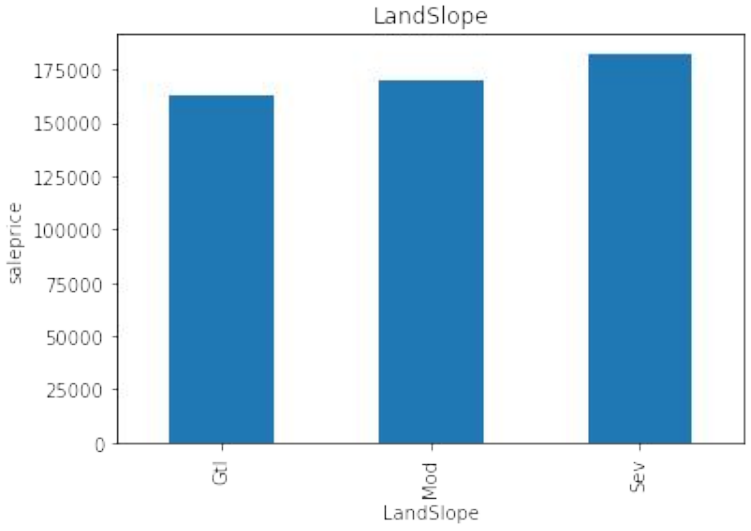

MSZoning
saleprice
200000
175000
150000
125000
100000
75000
50000
25000
0
C (all)
FV
RH
RL
RM
MSZoning

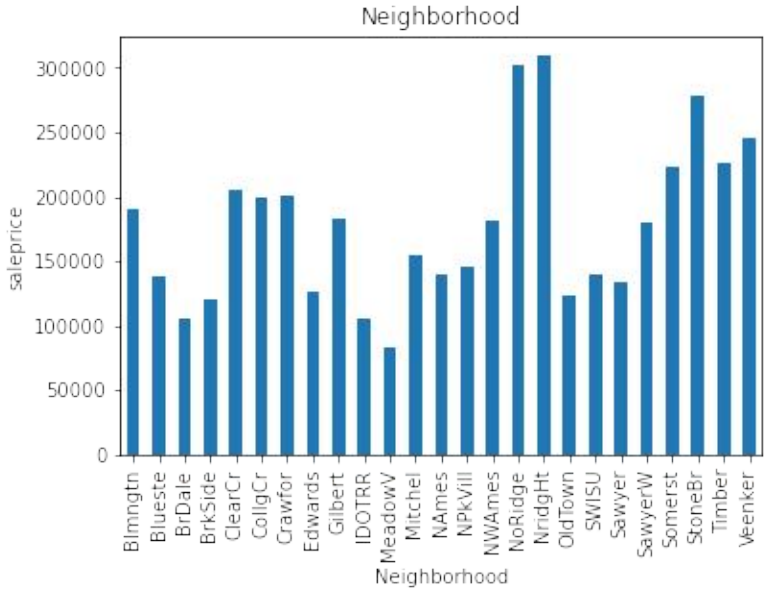

Street
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
Grvl
Pave
Street

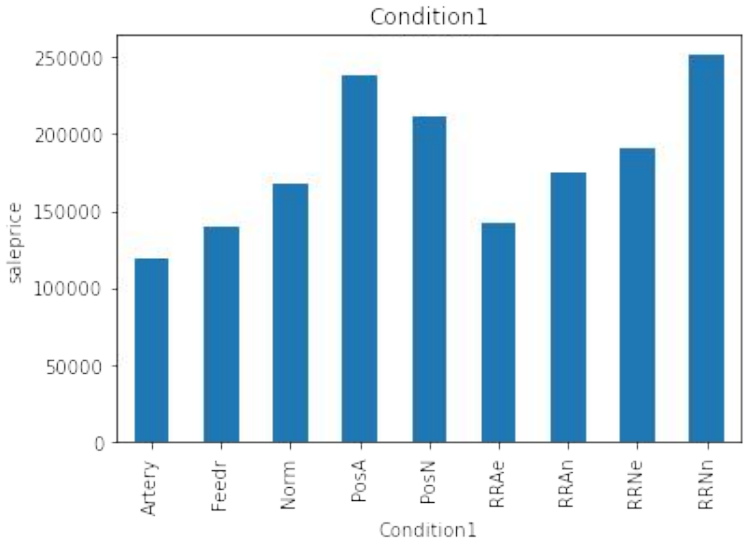

LotShape
saleprice
200000
150000
100000
50000
0
IR1
IR2
IR3
Reg
LotShape

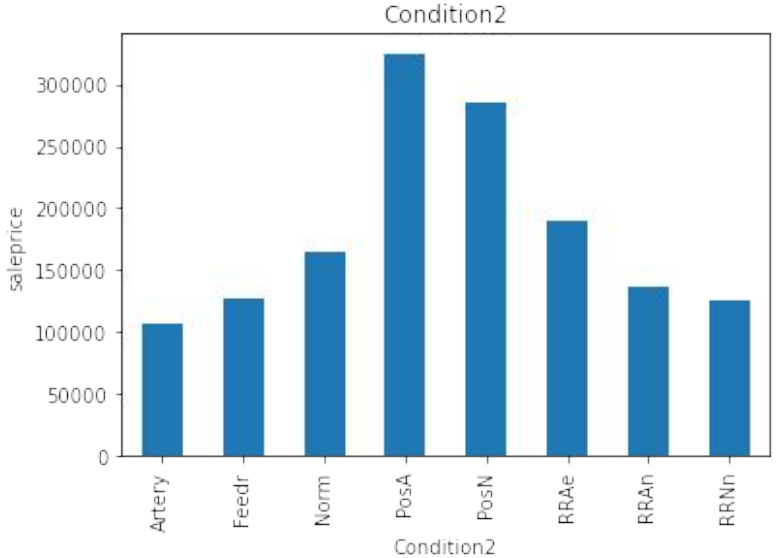

LandContour
saleprice
200000
175000
150000
125000
100000
75000
50000
25000
0
Bnk
HLS
Low
Lvl
LandContour

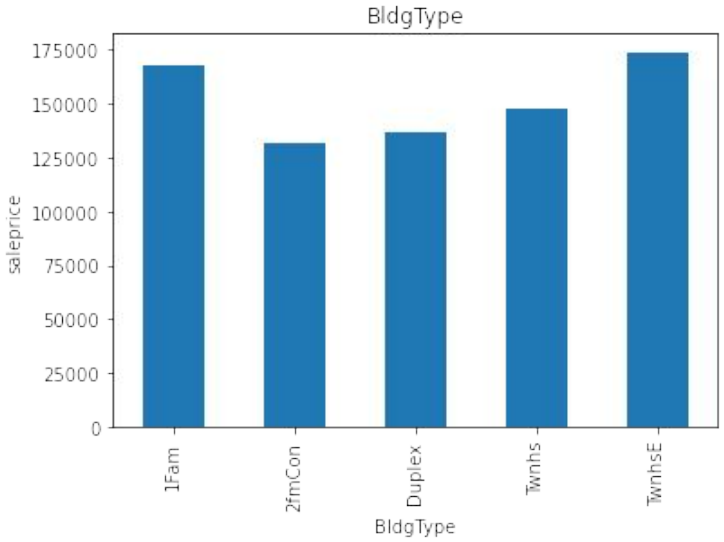

LotConfig
saleprice
200000
150000
100000
50000
0
Corner
CulDSac
FR2
FR3
Inside
LotConfig

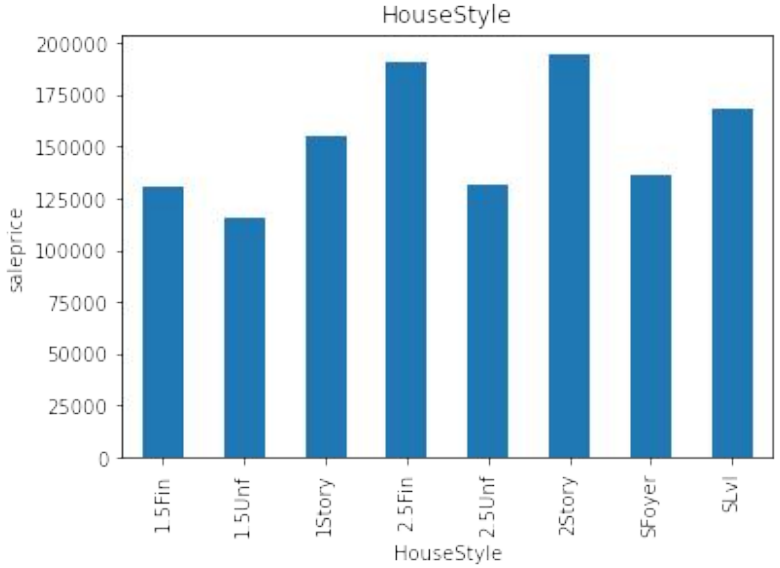

LandSlope
saleprice
175000
150000
125000
100000
75000
50000
25000
0
Gtl
Mod
Sev
LandSlope

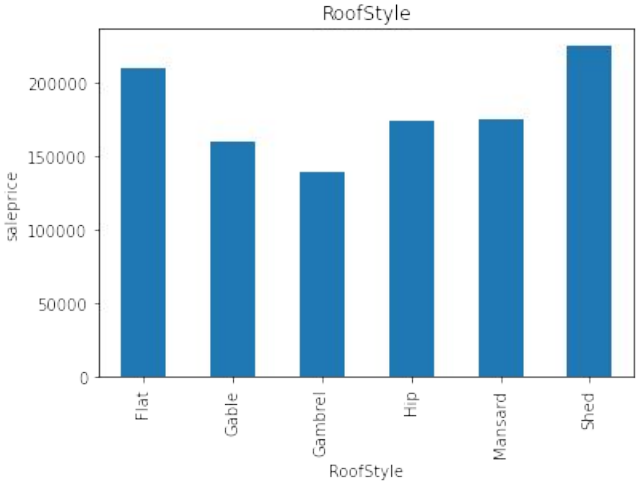

Neighborhood
saleprice
300000
250000
200000
150000
100000
50000
0
Blmngtn
Blueste
BrDale
BrkSide
ClearCr
CollgCr
Crawfor
Edwards
Gilbert
IDOTRR
MeadowV
Mitchel
NAmes
NPkVill
NWAmes
NoRidge
NridgHt
OldTown
SWISU
Sawyer
SawyerW
Somerst
StoneBr
Timber
Veenker
Neighborhood

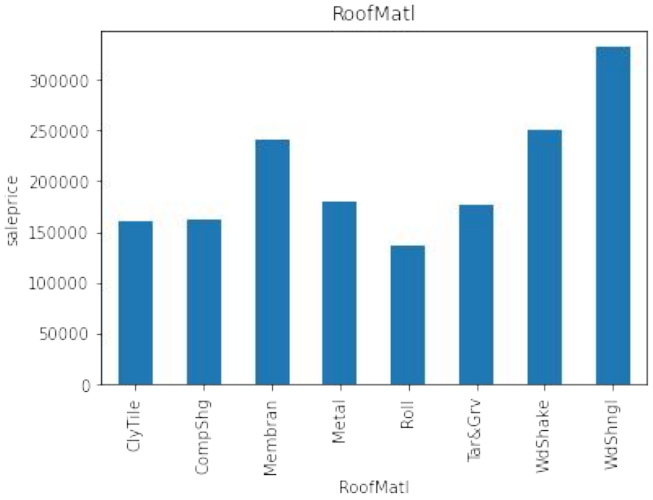

Condition1
saleprice
250000
200000
150000
100000
50000
0
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNe
RRNn
Condition1

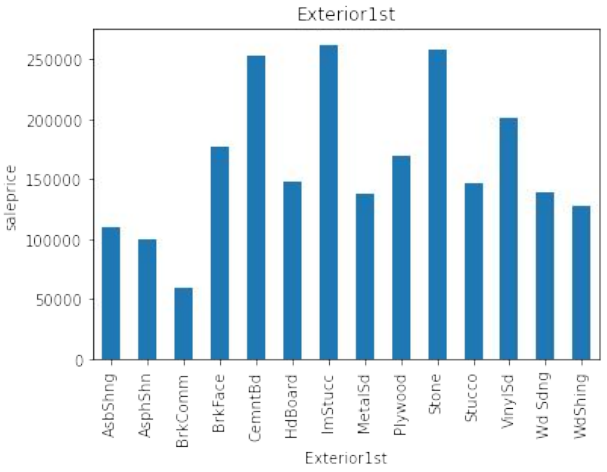

Condition2
saleprice
300000
250000
200000
150000
100000
50000
0
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNn
Condition2

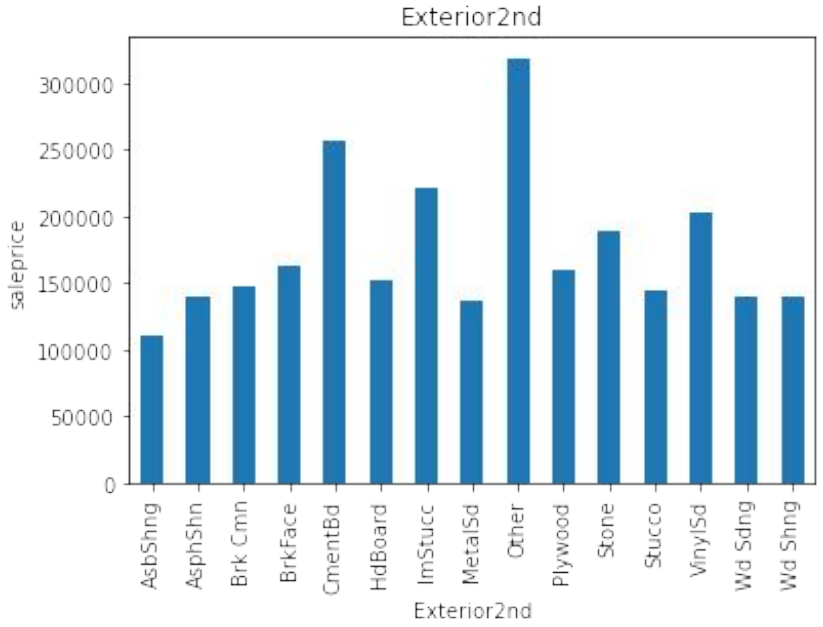

BldgType
saleprice
175000
150000
125000
100000
75000
50000
25000
0
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
BldgType

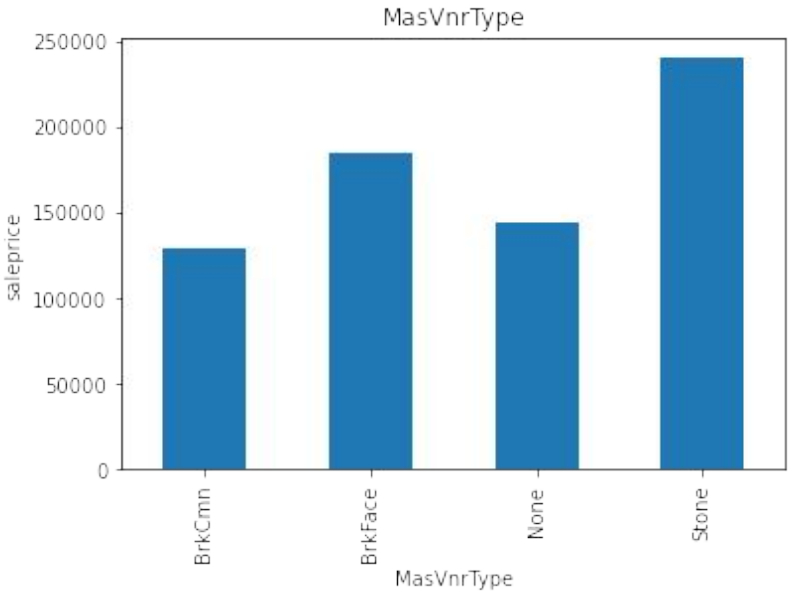

HouseStyle
saleprice
200000
175000
150000
125000
100000
75000
50000
25000
0
1.5Fin
1.5Unf
1Story
2.5Fin
2.5Unf
2Story
SFoyer
SLvl
HouseStyle

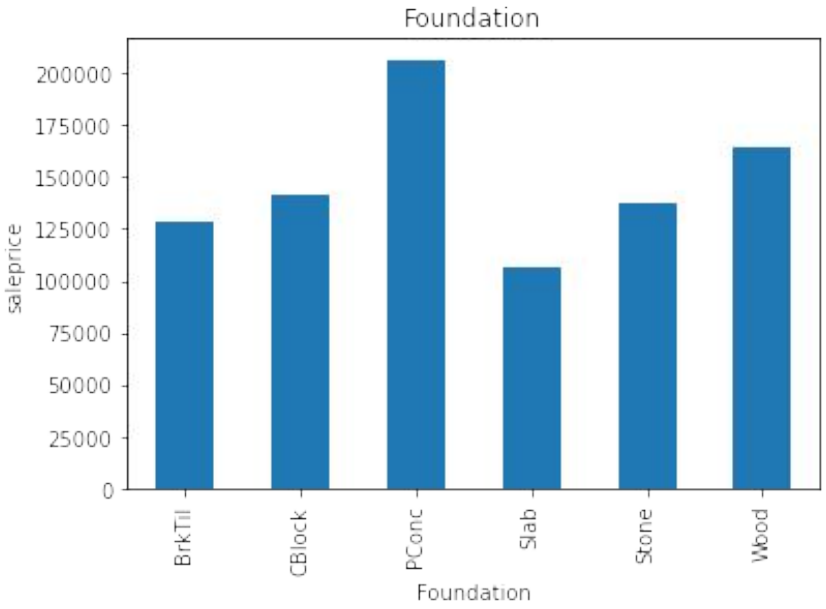

RoofStyle
saleprice
200000
150000
100000
50000
0
Flat
Gable
Gambrel
Hip
Mansard
Shed
RoofStyle

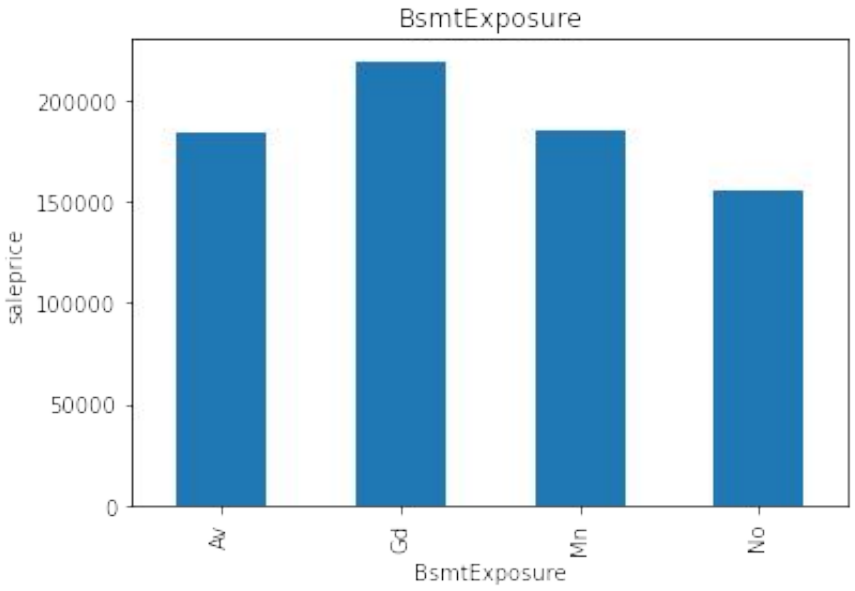

RoofMatl
saleprice
300000
250000
200000
150000
100000
50000
0
ClyTile
CompShg
Membran
Metal
Roll
Tar&Grv
WdShake
WdShngl
RoofMatl

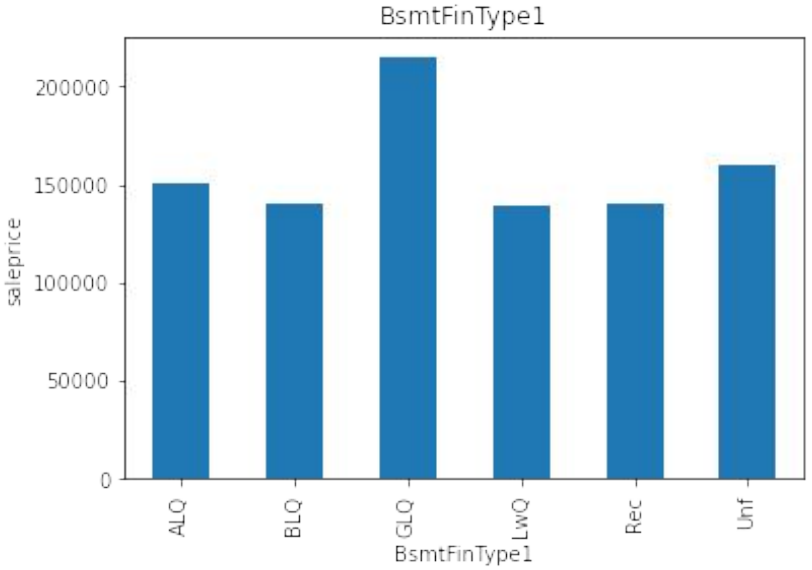

Exterior1st
saleprice
250000
200000
150000
100000
50000
0
AsbShng
AsphShn
BrkComm
BrkFace
CemntBd
HdBoard
ImStucc
MetalSd
Plywood
Stone
Stucco
VinylSd
Wd Sdng
WdShing
Exterior1st

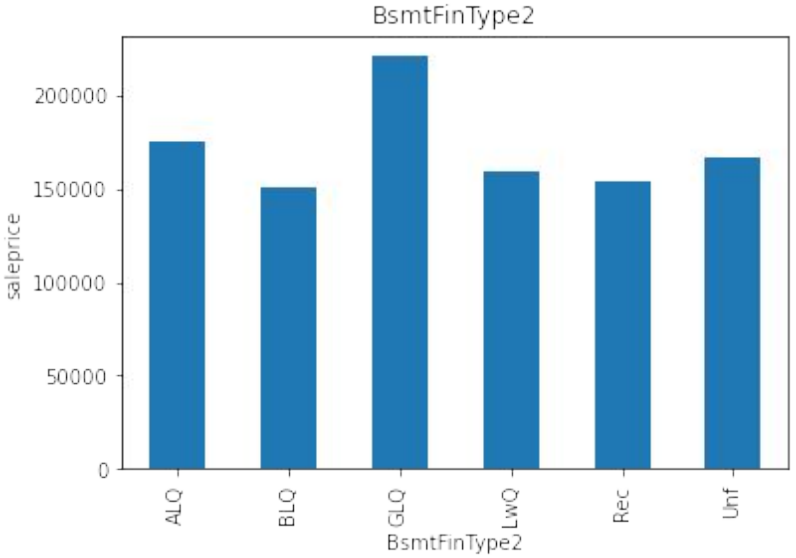

Exterior2nd
saleprice
300000
250000
200000
150000
100000
50000
0
AsbShng
AsphShn
Brk Cmn
BrkFace
CmentBd
HdBoard
ImStucc
MetalSd
Other
Plywood
Stone
Stucco
VinylSd
Wd Sdng
Wd Shng
Exterior2nd

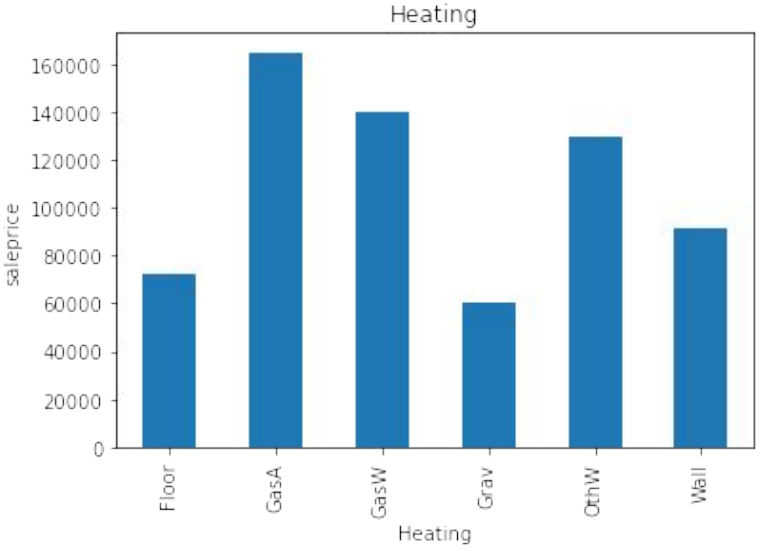

MasVnrType
saleprice
250000
200000
150000
100000
50000
0
BrkCmn
BrkFace
None
Stone
MasVnrType

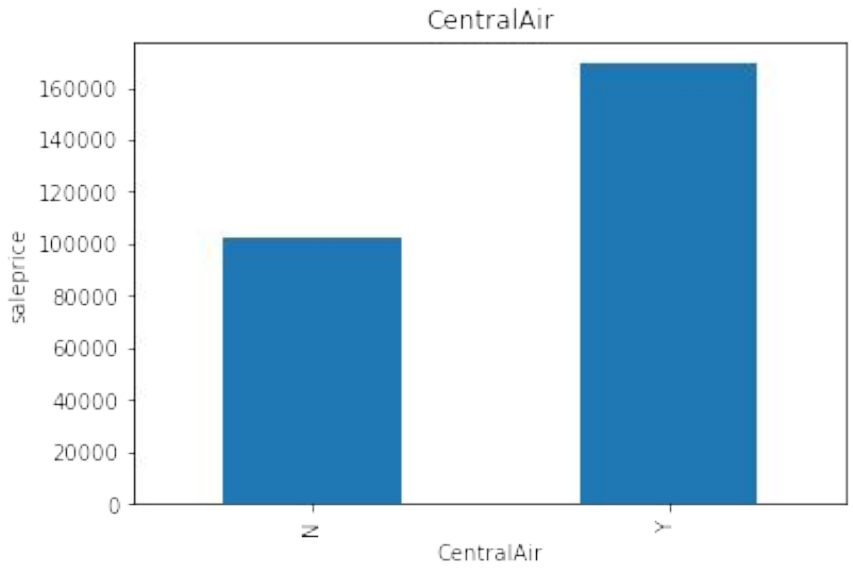

Foundation
saleprice
200000
175000
150000
125000
100000
75000
50000
25000
0
BrkTil
CBlock
PConc
Slab
Stone
Wood
Foundation

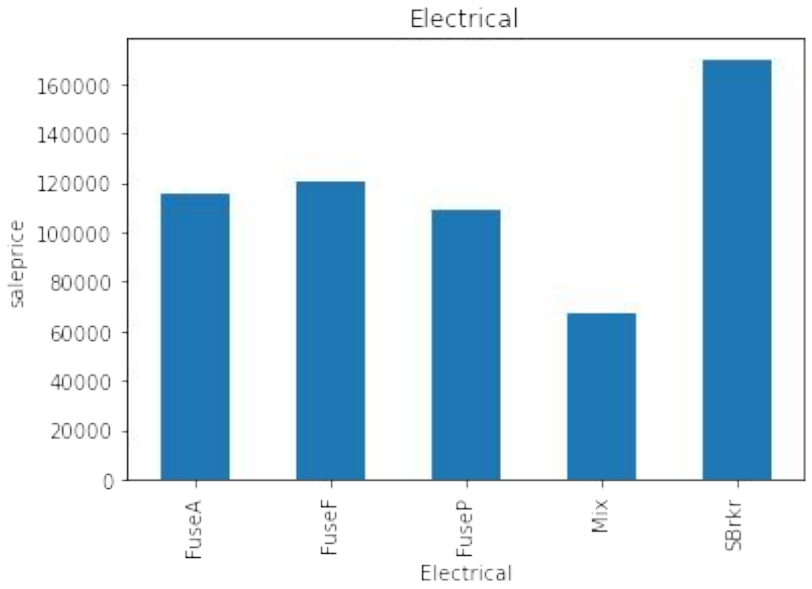

BsmtExposure
saleprice
200000
150000
100000
50000
0
Av
Gd
Mn
No
BsmtExposure

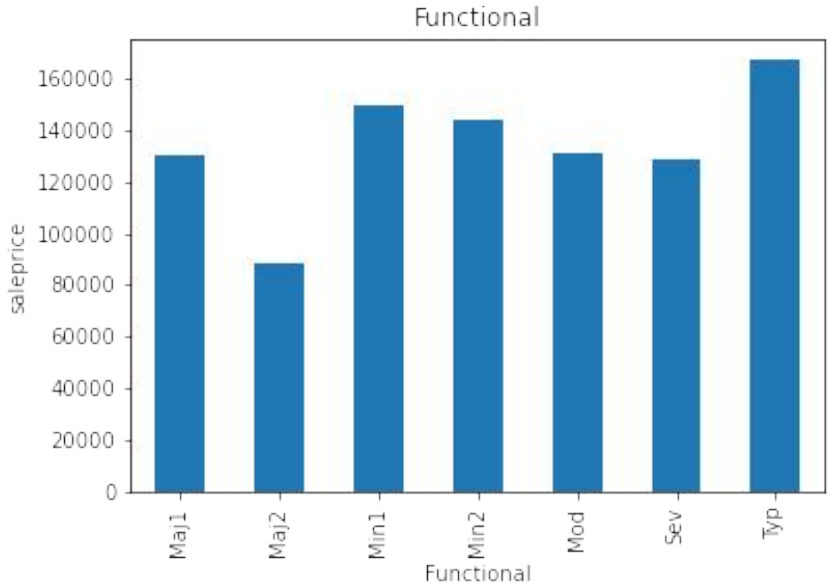

BsmtFinType1
saleprice
200000
150000
100000
50000
0
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType1

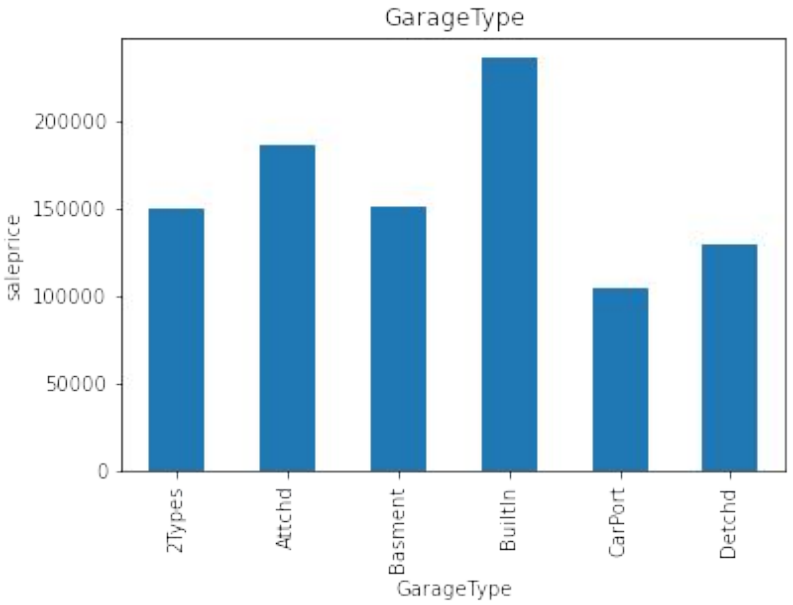

BsmtFinType2
saleprice
200000
150000
100000
50000
0
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType2

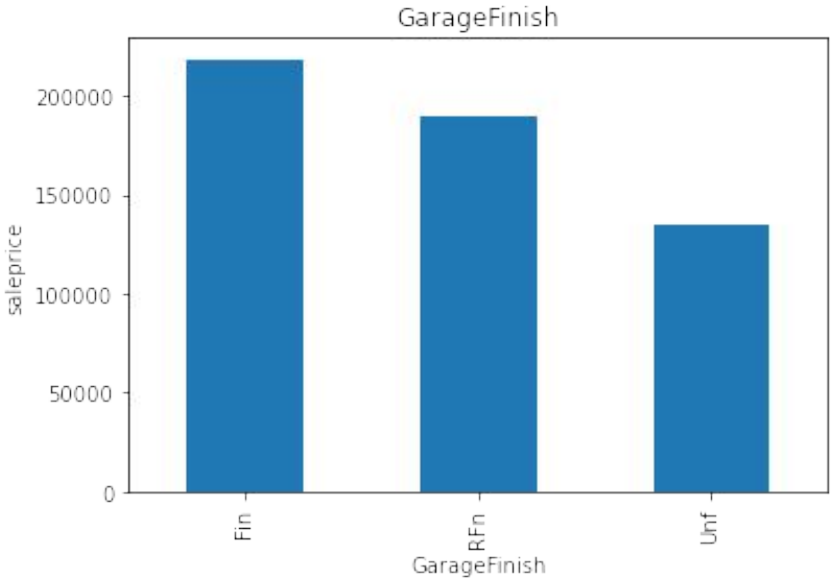

Heating
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
Floor
GasA
GasW
Grav
OthW
Wall
Heating

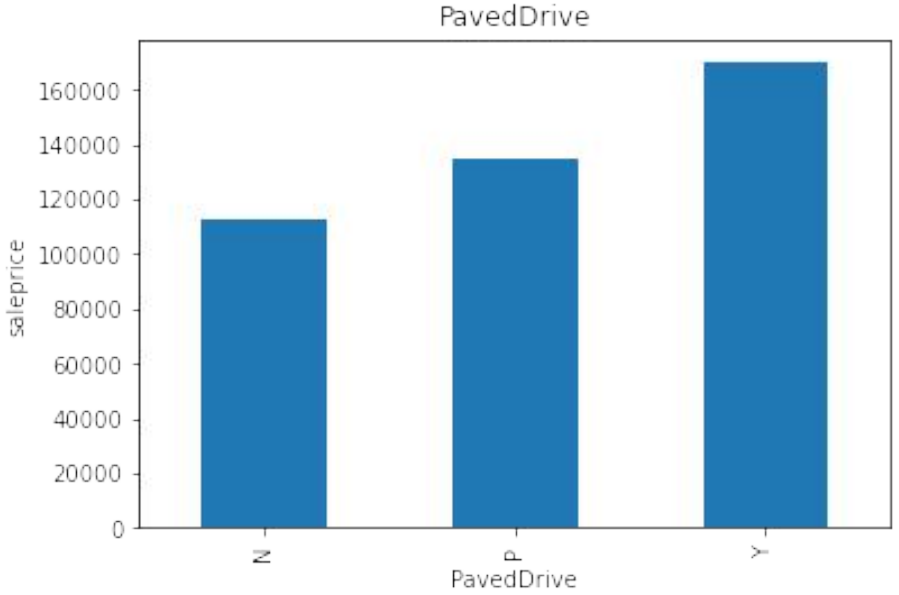

CentralAir
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
N
Y
CentralAir

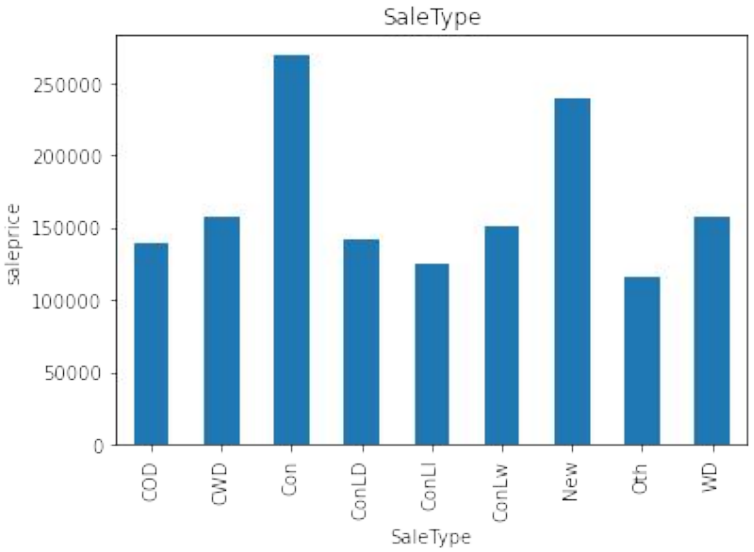

Electrical
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
FuseA
FuseF
FuseP
Mix
SBrkr
Electrical

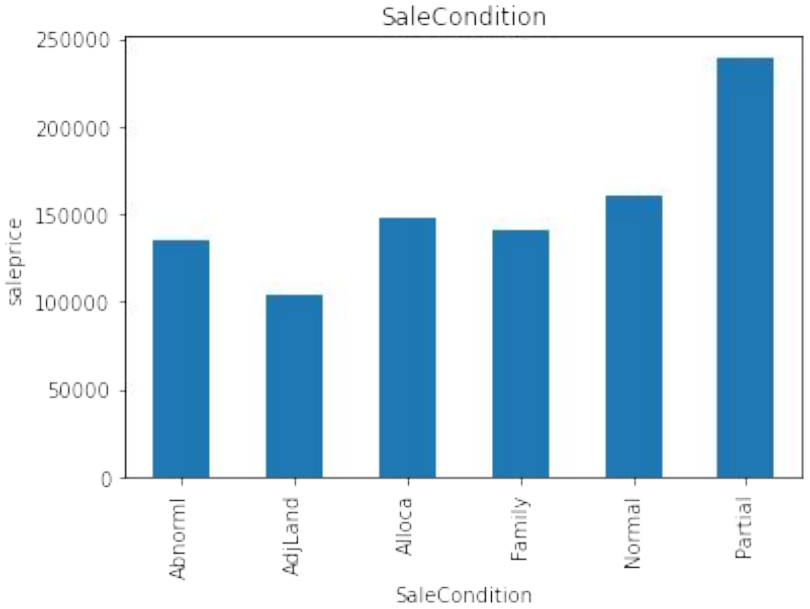

Functional
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
Maj1
Maj2
Min1
Min2
Mod
Sev
Typ
Functional


GarageType
saleprice
200000
150000
100000
50000
0
2Types
Attchd
Basment
BuiltIn
CarPort
Detchd
GarageType


GarageFinish
saleprice
200000
150000
100000
50000
0
Fin
RFn
Unf
GarageFinish


PavedDrive
saleprice
160000
140000
120000
100000
80000
60000
40000
20000
0
N
P
Y
PavedDrive





- In that Floating Village Residential and Residential Low Density has max price of house
- On Pave Street house price is high
- Moderately Irregular, Hillside - Significant slope from side to side, Depression max price
- Corner and Frontage 3 side property has maximum price of house.
- Northridge , NorthPark Villa, Stone brook near property has price more than 250000.
- Clear creek, College creek, Crawford, Gilbert, North Ames, sawyer west, Somerest, Timberland, Veenker place have

Average house prices up to 200000, Remaining Places have price between 50000 to 100000

- Flat and Shed type roof price is more also used materials of wood shakers and Wood shingles prices high
- Exterior covering of by Cement board, Imitation Stucco, Stone, Vinyl Siding has more price.
- Masonry veneer type in Stone the house price more
- Poured concrete type foundation has more price of house cost of concrete is more than other.
- Gas forced warm air furnace, Gas hot water or steam heat, Hot water or steam heat other than gas facilities have more price than others.
- Central air conditioning facility have more price of house.
- Standard Circuit Breakers & Romex electric facility has high price of house.
- Built in garage facility house price is more.
- Contract 15% Down payment regular terms, Home just constructed and sold both have max price.
- Home was not completed when last assessed (associated with New Homes) has high price

CONCLUSION

- Key Findings and Conclusions of the Study

```
# features selection for model
feature_selection=SelectFromModel(Ridge(alpha=1,solver="saga",random_state=428))
feature_selection.fit(a,y)
```

```
SelectFromModel(estimator=Ridge(alpha=1, random_state=428, solver='saga'))
```

```
feature_selection.get_support()
```

```
array([False, False, False,  True,  True, False, False, False, False,
       False,  True, False, False,  True,  True, False,  True, False,
       False, False, False, False,  True, False, False, False, False,
        True, False, False,  True, False, False,  True, False,  True,
       False,  True,  True, False,  True, False,  True,  True, False,
        True,  True, False, False, False, False,  True, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False,  True])
```

```
print('Total features',x2.shape[1])
print('Select features',len(select_feat))
```

```
Total features 66
Select features 21
```

```
# features that affect most in house price
select_feat=x2.columns[(feature_selection.get_support())]
select_feat
```

```
Index(['LotArea', 'Street', 'Condition2', 'OverallQual', 'YearRemodAdd',
       'RoofMatl', 'Foundation', 'BsmtFinSF1', 'TotalBsmtSF', 'CentralAir',
       '1stFlrSF', 'GrLivArea', 'BsmtFullBath', 'FullBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'Functional', 'Fireplaces', 'GarageCars', 'PoolArea',
       'SaleCondition'],
      dtype='object')
```

- From the above code find features that most affect the house price or say key features of datasets by Lasso model because its equal weight the all features.

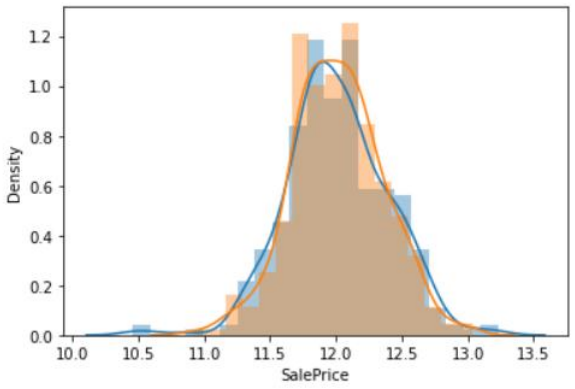
```
pd.DataFrame({'Actual':np.exp(y_test),'Prediction':np.exp(gd.predict(x_test))})
```

	Actual	Prediction
997	262001.0	285517.687892
604	280001.0	271371.772154
1145	139001.0	132536.090884
1118	200001.0	186625.427826
791	392501.0	266884.570308
...	...	...
933	150751.0	140659.733593
26	112001.0	150292.442400
309	147501.0	150027.572485
291	307001.0	284671.059664
380	201801.0	206315.782051

351 rows × 2 columns

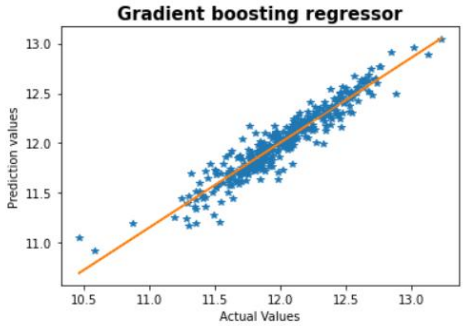
```
# Distribution plot of Prediction
sns.distplot(y_test,)
sns.distplot(gd.predict(x_test))
```

```
<AxesSubplot:xlabel='SalePrice', ylabel='Density'>
```




```
x=np.array(y_test)
y=np.array(gd.predict(x_test))
plt.plot(x,y,'*')
m,b=np.polyfit(x,y,1)
plt.plot(x,m*x+b,)
plt.xlabel('Actual Values')
plt.ylabel('Prediction values')
plt.title('Gradient boosting regressor',{'fontweight':'bold','fontsize':15})
```

```
Text(0.5, 1.0, 'Gradient boosting regressor')
```



➢ Test Data set Prediction result Below

```
house_price=pd.DataFrame({'ID':X_test['Id'],'Prediction':np.exp(gd.predict(X_test1))})
```

```
house_price.head(10)
```

	ID	Prediction
0	337.0	172537.229585
1	1018.0	124728.362422
2	929.0	155916.447043
3	1148.0	101851.326551
4	1227.0	142393.641501
5	650.0	69200.510582
6	1453.0	104318.307390
7	152.0	166767.672200
8	427.0	138337.116055
9	776.0	114456.069470

```
house_price.to_csv('Datafile.csv',index=False)
```

- Learning Outcomes of the Study in respect of Data Science
  - After data cleaning, features selection, transform the data **Gradient Boosting Regressor** performed well with **Parameters of (“criterion='friedman_mse',learning_rate=0.01,max_features='log2',n_estimators=1000”)**
- Limitations of this work and Scope for Future Work
  - In this dataset we Find information regarding to Australian real-estate business so our model boundary is limited for that only.
  - In Dataset 50 to 100% Missing values in some features so we don’t have information regarding them and other missing values we assumed that values at random and fill-up this also affect the model performance .
  - If we have Geological data like Latitude and Longitude so that also affect more in performance of model and Visualization as well.