HOUSING PROJECT CASE STUDY PROJECT REPORT

SUBMITTED BY:
Wesley Sirra

BUSINESS PROBLEM

Housing and real estate markets are important contributors to a country's economy. It is a huge market with many firms operating in it. Data Science may assist countries improve their total income, profitability, and marketing strategies by solving challenges in this sector. Machine learning techniques may be utilized to help this housing company achieve its commercial objectives. Our problem is with a housing company based in the United States called Surprise Housing, which wants to enter the Australian market. The company intends to use Data Analytics to buy houses at a discount from their true value and resell them at a profit. The company has compiled a dataset based on house sales in Australia. The firm is looking at potential properties to purchase residences in order to enter the market. We will create a model utilizing Machine Learning to estimate the actual worth of potential properties, which will assist the firm in deciding whether or not to invest in .real estate

Analytical Problem Framing

Data sources and their formats

We are provided two CSV files comprising train and test datasets of house sales for this project.
The dataset used to train the Machine Learning model has 1168 rows and 81 columns. Using this dataset, we will train the Machine Learning models on 75% of the data and validate the models on 25% of the data.
Finally, we will forecast prices for the testing dataset, which has 292 rows and 80 columns.

Mathematical/Analytical modelling

Let's look at the data now. I've attached a snapshot to give you an idea of what I'm talking about.

```python
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Preprocessing,Standardizing
from sklearn.preprocessing import StandardScaler

#For Multicollinearity
from statsmodels.stats.outliers_influence import variance_inflation_factor

#Models
from sklearn.model_selection import train_test_split,RandomizedSearchCV, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression

#Metrics
from sklearn.metrics import r2_score

import warnings
warnings.filterwarnings('ignore')
```

```python
df=pd.read_csv('train.csv')
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mo
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	10
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	6
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	1
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	6

5 rows × 81 columns

```
df1=pd.read_csv('test.csv')
df1.head()
```

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

5 rows × 80 columns

EDA

```
df.shape
```

```
(1168, 81)
```

We have 1168 rows and 81 columns in our train dataset.

```
df1.shape
```

```
(292, 80)
```

We have 292 rows and 80 columns in our test dataset.

```
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)
```

```
1 df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292 entries, 0 to 291
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             292 non-null    int64
 1   MSSubClass     292 non-null    int64
 2   MSZoning       292 non-null    object
 3   LotFrontage    247 non-null    float64
 4   LotArea        292 non-null    int64
 5   Street         292 non-null    object
 6   Alley          14 non-null     object
 7   LotShape       292 non-null    object
 8   LandContour    292 non-null    object
 9   Utilities      292 non-null    object
 10  LotConfig      292 non-null    object
 11  LandSlope      292 non-null    object
 12  Neighborhood   292 non-null    object
 13  Condition1     292 non-null    object
 14  Condition2     292 non-null    object
 15  BldgType       292 non-null    object
 16  HouseStyle     292 non-null    object
 17  OverallQual    292 non-null    int64
 18  OverallCond    292 non-null    int64
 19  YearBuilt      292 non-null    int64
 20  YearRemodAdd   292 non-null    int64
 21  RoofStyle      292 non-null    object
 22  RoofMatl       292 non-null    object
 23  Exterior1st    292 non-null    object
 24  Exterior2nd    292 non-null    object
 25  MasVnrType     291 non-null    object
 26  MasVnrArea     291 non-null    float64
 27  ExterQual      292 non-null    object
 28  ExterCond      292 non-null    object
 29  Foundation     292 non-null    object
 30  BsmtQual       285 non-null    object
 31  BsmtCond       285 non-null    object
 32  BsmtExposure   285 non-null    object
 33  BsmtFinType1   285 non-null    object
 34  BsmtFinSF1     292 non-null    int64
 35  BsmtFinType2   285 non-null    object
 36  BsmtFinSF2     292 non-null    int64
 37  BsmtUnfSF      292 non-null    int64
 38  TotalBsmtSF    292 non-null    int64
 39  Heating        292 non-null    object
 40  HeatingQC      292 non-null    object
 41  CentralAir     292 non-null    object
 42  Electrical     291 non-null    object
 43  1stFlrSF       292 non-null    int64
 44  2ndFlrSF       292 non-null    int64
 45  LowQualFinSF   292 non-null    int64
 46  GrLivArea      292 non-null    int64
 47  BsmtFullBath   292 non-null    int64
 48  BsmtHalfBath   292 non-null    int64
 49  FullBath       292 non-null    int64
 50  HalfBath       292 non-null    int64
 51  BedroomAbvGr   292 non-null    int64
 52  KitchenAbvGr   292 non-null    int64
 53  KitchenQual    292 non-null    object
 54  TotRmsAbvGrd   292 non-null    int64
 55  Functional     292 non-null    object
 56  Fireplaces     292 non-null    int64
 57  FireplaceQu    153 non-null    object
 58  GarageType     275 non-null    object
 59  GarageYrBlt    275 non-null    float64
 60  GarageFinish   275 non-null    object
 61  GarageCars     292 non-null    int64
 62  GarageArea     292 non-null    int64
 63  GarageQual     275 non-null    object
 64  GarageCond     275 non-null    object
 65  PavedDrive     292 non-null    object
 66  WoodDeckSF     292 non-null    int64
 67  OpenPorchSF    292 non-null    int64
 68  EnclosedPorch  292 non-null    int64
 69  3SsnPorch      292 non-null    int64
 70  ScreenPorch    292 non-null    int64
 71  PoolArea       292 non-null    int64
 72  PoolQC         0 non-null      float64
 73  Fence          44 non-null     object
 74  MiscFeature    10 non-null     object
 75  MiscVal        292 non-null    int64
 76  MoSold         292 non-null    int64
 77  YrSold         292 non-null    int64
 78  SaleType       292 non-null    object
 79  SaleCondition  292 non-null    object
dtypes: float64(4), int64(34), object(42)
```

From the above tables, we can see that we have lot of null values in our train as well as test dataset. Also we have int, float and object datatypes. SalePrice is our target variable.

```
df.describe()
```

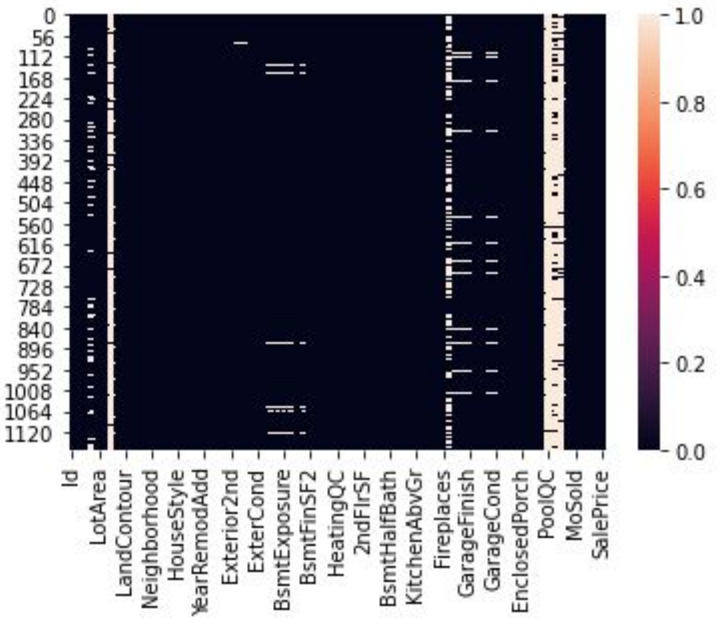
	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodD
count	1168.000000	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	...	1168.0
mean	724.136130	56.767979	70.98847	10484.749144	6.104452	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	...	126.15
min	1.000000	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	...	0.0000
25%	360.500000	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	...	0.0000
50%	714.500000	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	...	0.0000
75%	1079.500000	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	...	171.00
max	1460.000000	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.00

8 rows × 38 columns

As said above, we see some null values in the count row. Also we see some of the features have outliers present in them along with some skewness.

```
sns.heatmap(df.isna())
```

<AxesSubplot:>

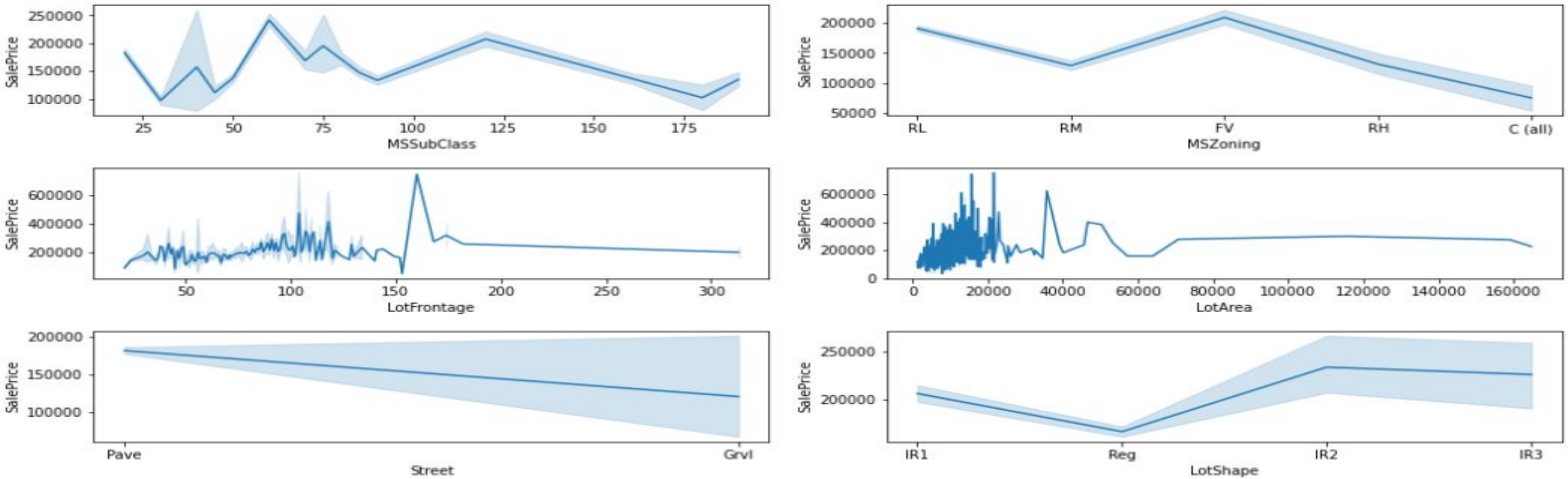


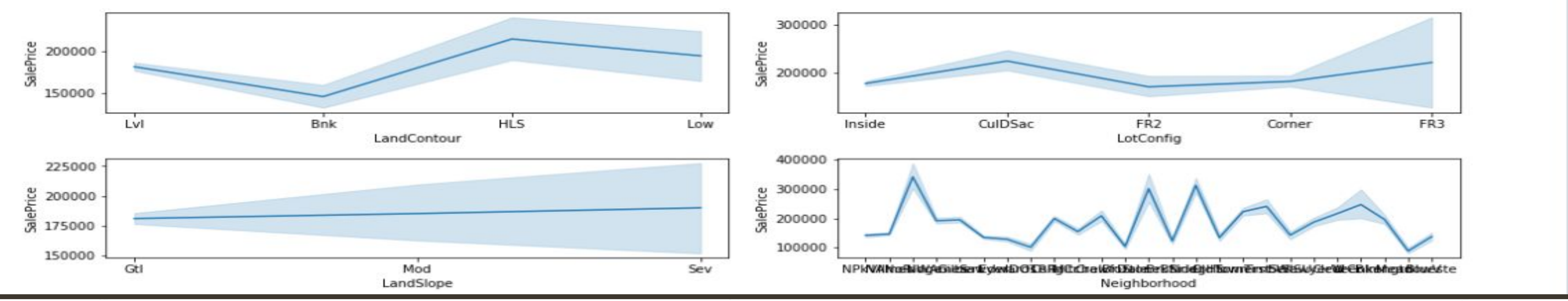
We see null values in a lot of features.

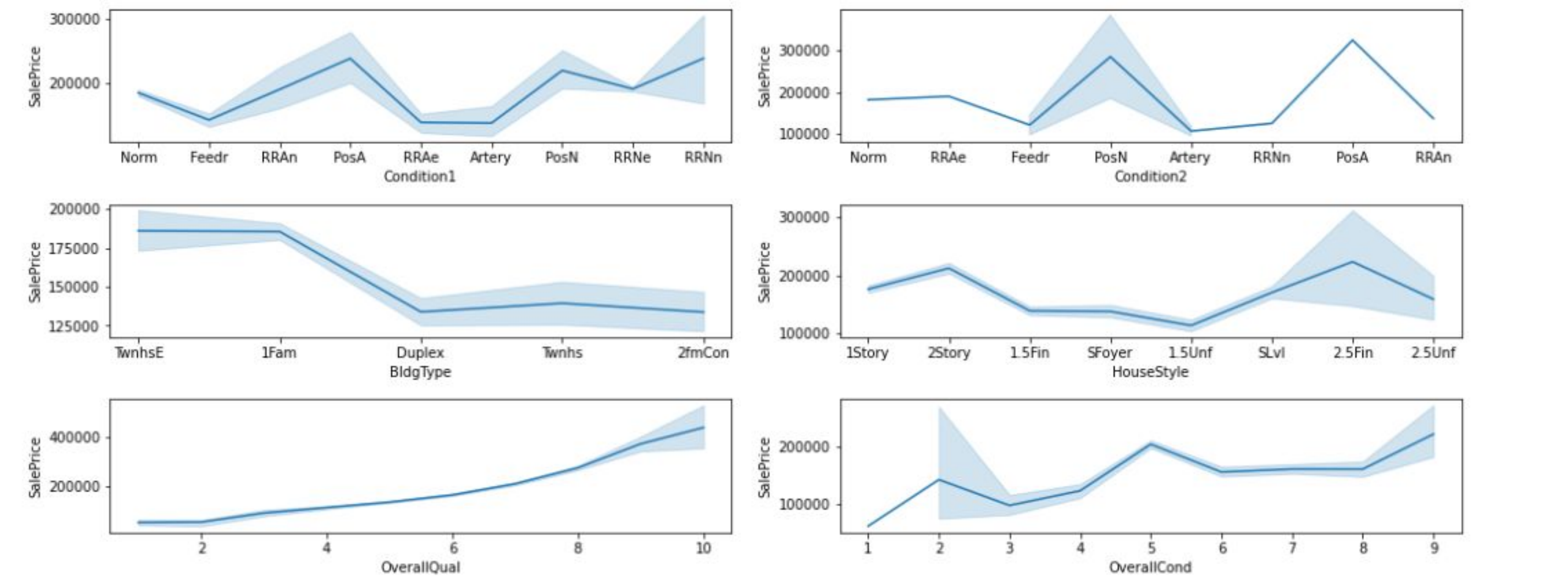
```python
y=df['SalePrice']
X=df.drop(columns=['SalePrice'])
```

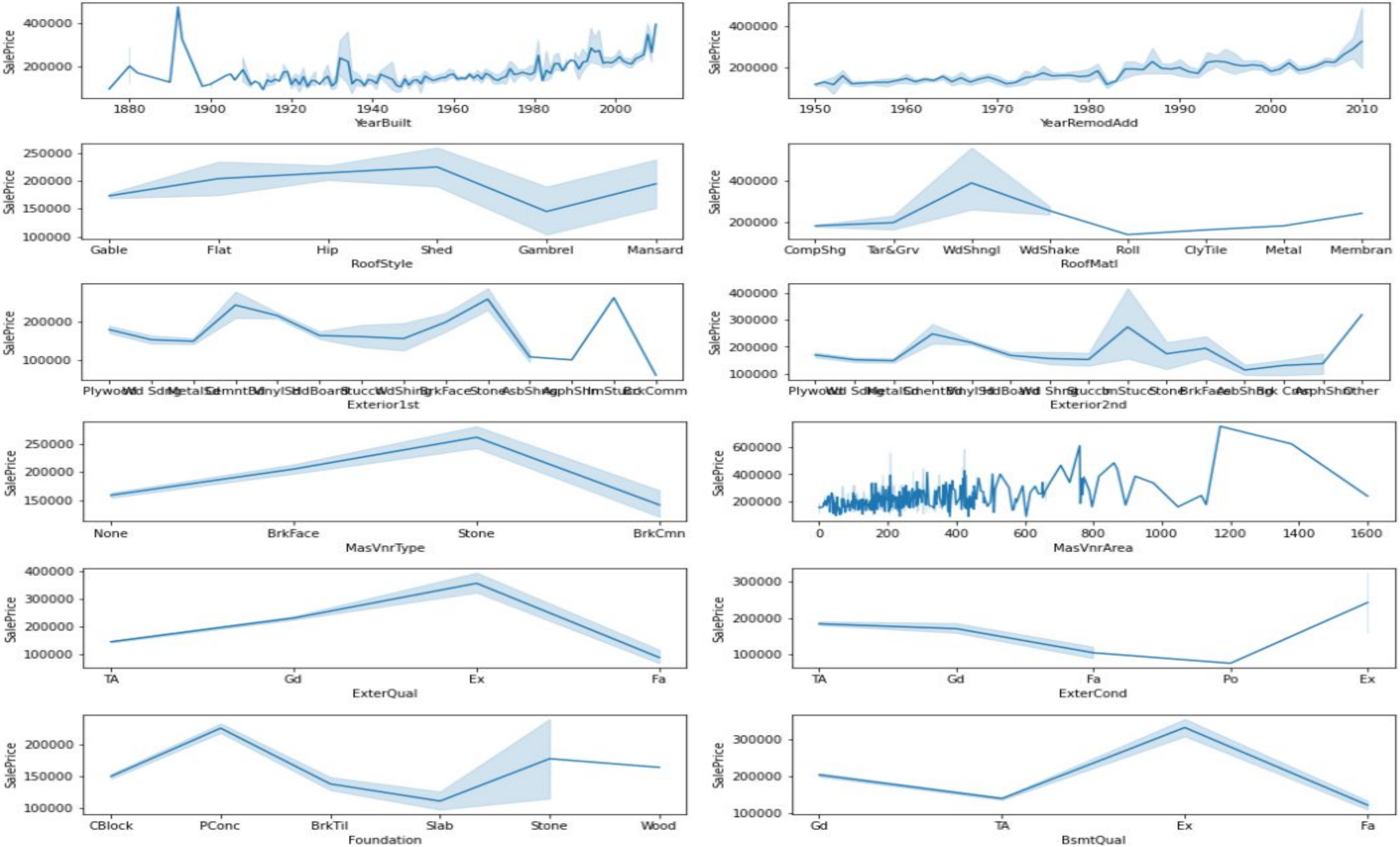
We have separated independent and dependent variables to visualize further with respect to our target variable.

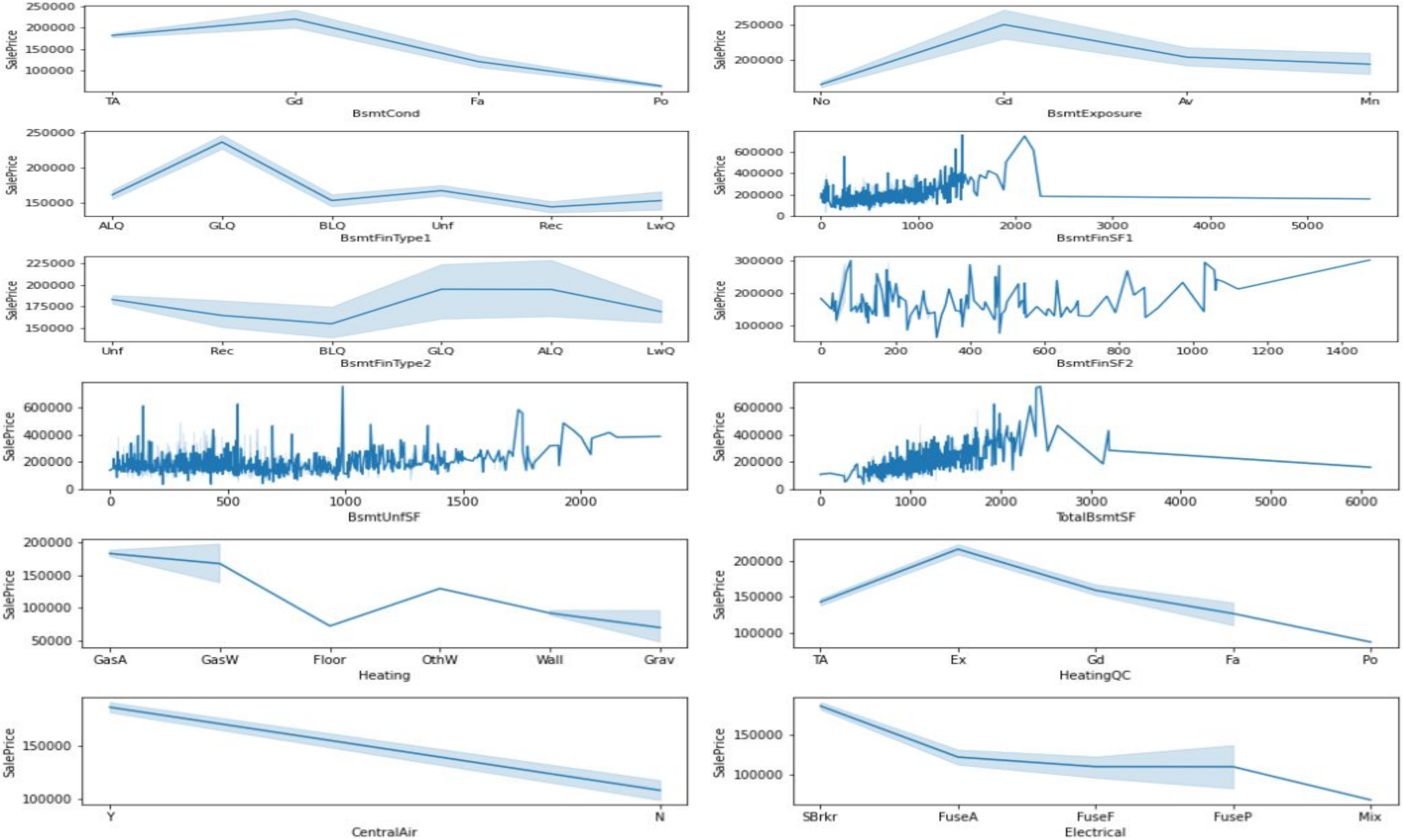
```python
plt.figure(figsize=(15,75),facecolor='white')
plotnumber=1

for column in X:
    if plotnumber<=76:
        ax=plt.subplot(38,2,plotnumber)
        sns.lineplot(X[column],y)
        plt.xlabel(column,fontsize=10)
        plt.ylabel('SalePrice',fontsize=10)

    plotnumber+=1
plt.tight_layout()
```

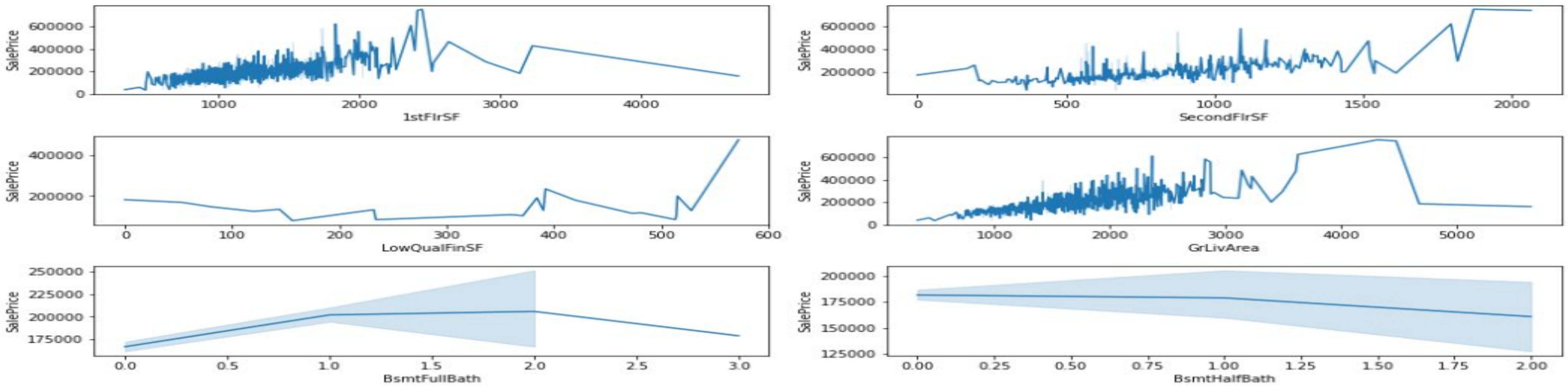


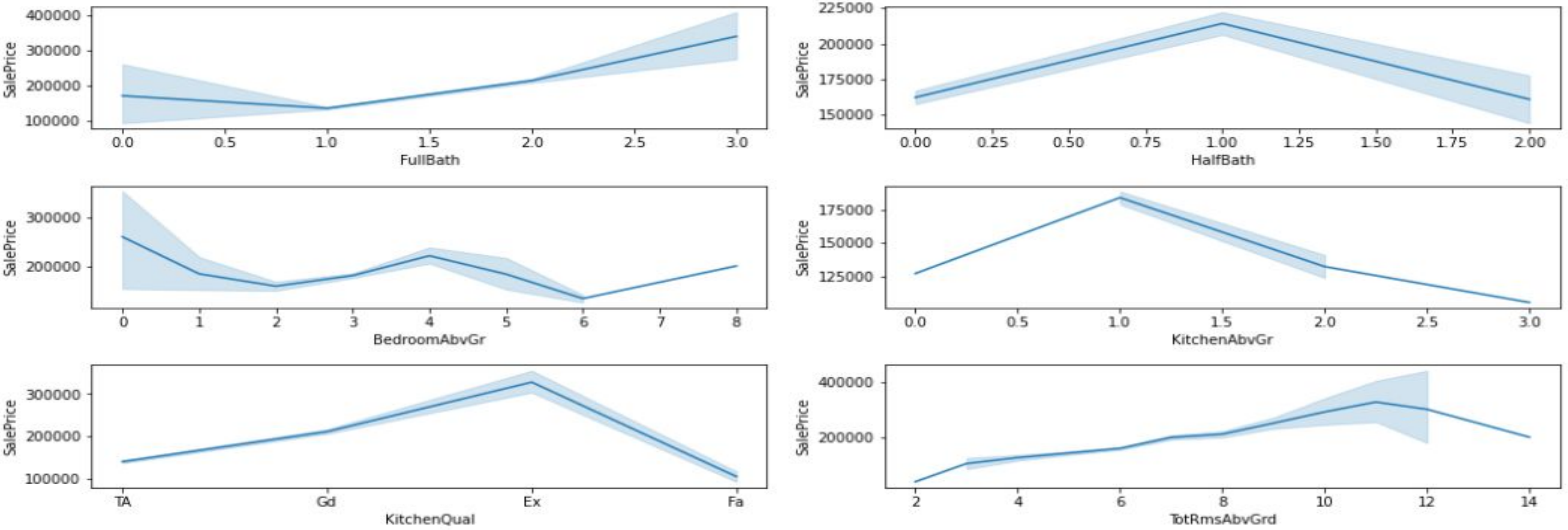

SalePrice
200000
150000
Lvl
Bnk
HLS
Low
LandContour
300000
200000
Inside
CulDSac
FR2
Corner
FR3
LotConfig
225000
200000
175000
150000
Gtl
Mod
Sev
LandSlope
400000
300000
200000
100000
Neighborhood

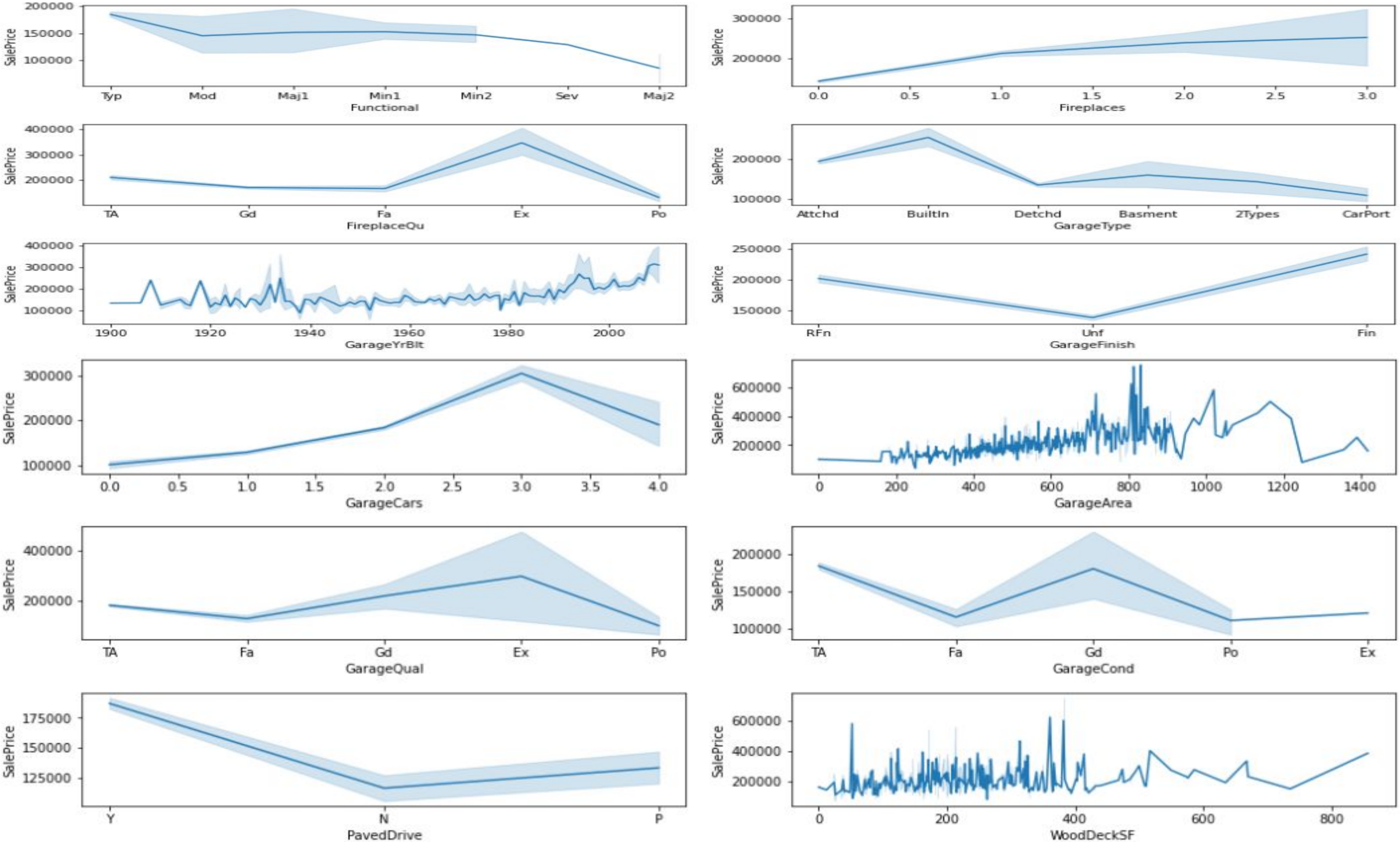

SalePrice
300000
200000
Norm
Feedr
RRAn
PosA
RRAe
Artery
PosN
RRNe
RRNn
Condition1
300000
200000
100000
Norm
RRAe
Feedr
PosN
Artery
RRNn
PosA
RRAn
Condition2
200000
175000
150000
125000
TwnhsE
1Fam
Duplex
Twnhs
2fmCon
BldgType
300000
200000
100000
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Fin
2.5Unf
HouseStyle
400000
200000
2
4
6
8
10
OverallQual
200000
100000
1
2
3
4
5
6
7
8
9
OverallCond


SalePrice
YearBuilt
1880
1900
1920
1940
1960
1980
2000
YearRemodAdd
1950
1960
1970
1980
1990
2000
2010
RoofStyle
Gable
Flat
Hip
Shed
Gambrel
Mansard
RoofMatl
CompShg
Tar&Grv
WdShngl
WdShake
Roll
ClyTile
Metal
Membran
Exterior1st
Exterior2nd
MasVnrType
None
BrkFace
Stone
BrkCmn
MasVnrArea
ExterQual
TA
Gd
Ex
Fa
ExterCond
Po
Foundation
CBlock
PConc
BrkTil
Slab
Wood
BsmtQual

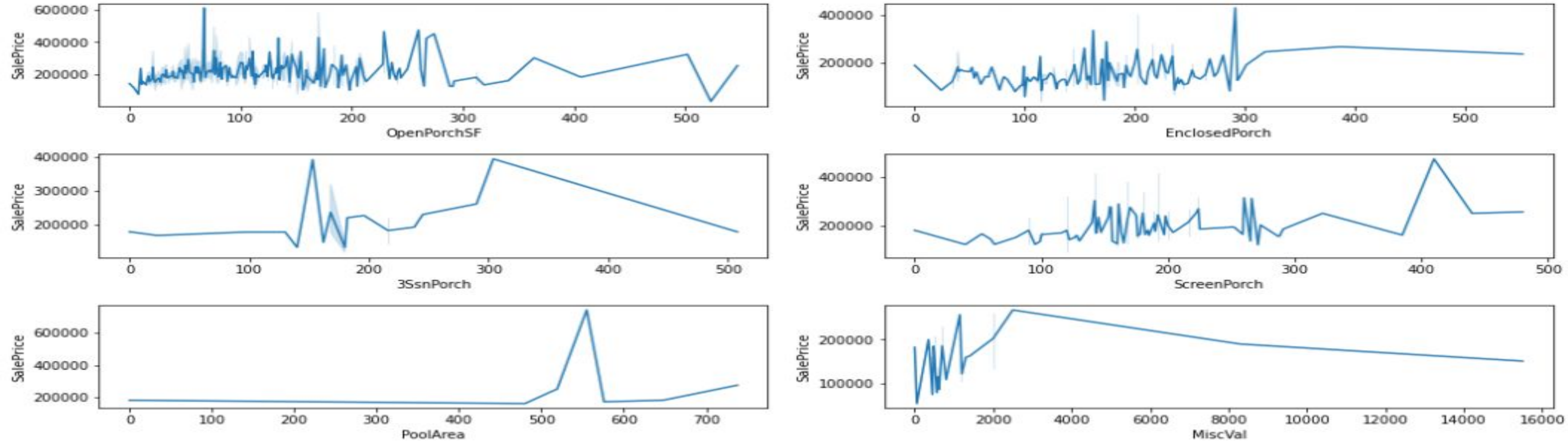

SalePrice
250000
200000
150000
100000
TA
Gd
Fa
Po
BsmtCond
SalePrice
250000
200000
No
Gd
Av
Mn
BsmtExposure
SalePrice
250000
200000
150000
ALQ
GLQ
BLQ
Unf
Rec
LwQ
BsmtFinType1
SalePrice
600000
400000
200000
0
0
1000
2000
3000
4000
5000
BsmtFinSF1
SalePrice
225000
200000
175000
150000
Unf
Rec
BLQ
GLQ
ALQ
LwQ
BsmtFinType2
SalePrice
300000
200000
100000
0
200
400
600
800
1000
1200
1400
BsmtFinSF2
SalePrice
600000
400000
200000
0
0
500
1000
1500
2000
BsmtUnfSF
SalePrice
600000
400000
200000
0
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF
SalePrice
200000
150000
100000
50000
GasA
GasW
Floor
OthW
Wall
Grav
Heating
SalePrice
200000
150000
100000
TA
Ex
Gd
Fa
Po
HeatingQC
SalePrice
150000
100000
Y
N
CentralAir
SalePrice
150000
100000
SBrkr
FuseA
FuseF
FuseP
Mix
Electrical

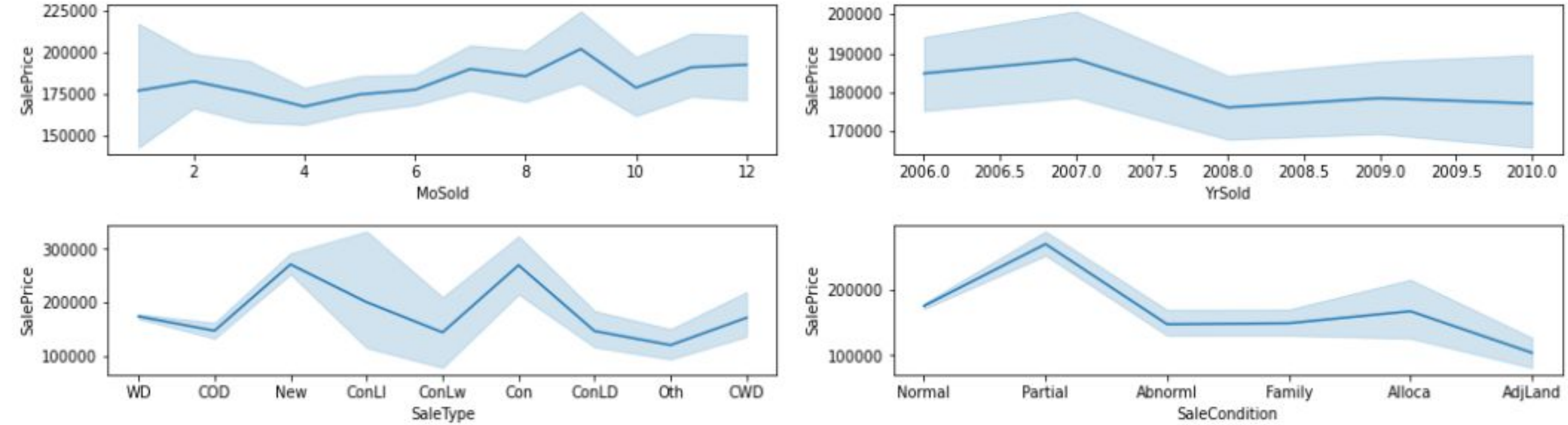

SalePrice
1stFlrSF
SalePrice
SecondFlrSF
SalePrice
LowQualFinSF
SalePrice
GrLivArea
SalePrice
BsmtFullBath
SalePrice
BsmtHalfBath


SalePrice
FullBath
SalePrice
HalfBath
SalePrice
BedroomAbvGr
SalePrice
KitchenAbvGr
SalePrice
TA
Gd
Ex
Fa
KitchenQual
SalePrice
TotRmsAbvGrd



SalePrice
200000
150000
100000
Typ
Mod
Maj1
Min1
Min2
Sev
Maj2
Functional
SalePrice
300000
200000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Fireplaces
SalePrice
400000
300000
200000
TA
Gd
Fa
Ex
Po
FireplaceQu
SalePrice
200000
100000
Attchd
BuiltIn
Detchd
Basment
2Types
CarPort
GarageType
SalePrice
400000
300000
200000
100000
1900
1920
1940
1960
1980
2000
GarageYrBlt
SalePrice
250000
200000
150000
RFn
Unf
Fin
GarageFinish
SalePrice
300000
200000
100000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
GarageCars
SalePrice
600000
400000
200000
0
200
400
600
800
1000
1200
1400
GarageArea
SalePrice
400000
200000
TA
Fa
Gd
Ex
Po
GarageQual
SalePrice
200000
150000
100000
TA
Fa
Gd
Po
Ex
GarageCond
SalePrice
175000
150000
125000
Y
N
P
PavedDrive
SalePrice
600000
400000
200000
0
200
400
600
800
WoodDeckSF


SalePrice
OpenPorchSF
0 100 200 300 400 500
200000 400000 600000
SalePrice
EnclosedPorch
0 100 200 300 400 500
200000 400000
SalePrice
3SsnPorch
0 100 200 300 400 500
200000 300000 400000
SalePrice
ScreenPorch
0 100 200 300 400 500
200000 400000
SalePrice
PoolArea
0 100 200 300 400 500 600 700
200000 400000 600000
SalePrice
MiscVal
0 2000 4000 6000 8000 10000 12000 14000 16000
100000 200000


SalePrice
MoSold
2 4 6 8 10 12
150000 175000 200000 225000
SalePrice
YrSold
2006.0 2006.5 2007.0 2007.5 2008.0 2008.5 2009.0 2009.5 2010.0
170000 180000 190000 200000
SalePrice
SaleType
WD COD New ConLI ConLw Con ConLD Oth CWD
100000 200000 300000
SalePrice
SaleCondition
Normal Partial Abnorml Family Alloca AdjLand
100000 200000

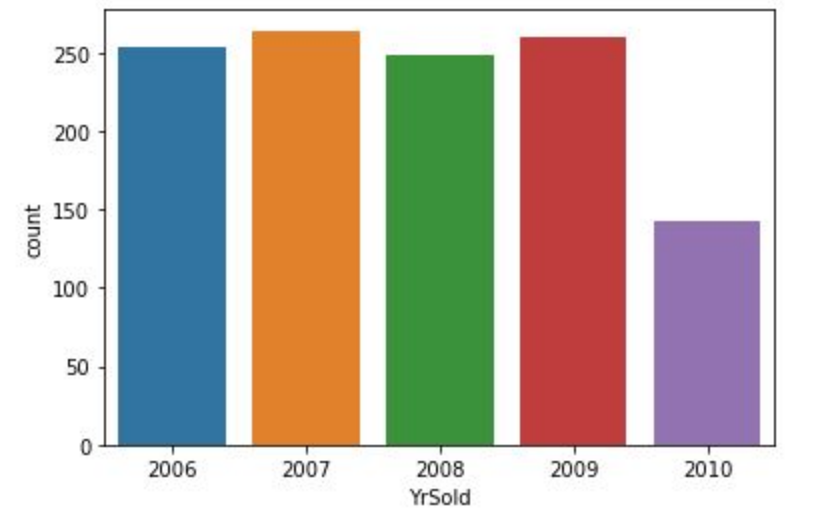
Observations:

Sale Price is highest for Floating Residential Village and lowest for Commercial zone.
Sale Price is highest for Paved Street and lowest for Gravel Street.
Sale Price is highest for Moderately Irregular shaped property and lowest for Regular shaped property.
Sale Price increases with Total square feet of basement area but it falls drastically after 2500 square feet area.
Sale Price is highest on the Hillside flatness whereas lowest in the banked flatness.
Sale Price doesn't have much of an impact on Type of Land Slope or Neighborhood.
Better the quality, higher the sale price.
Although not monotonic, but there's an increase in sale price with better overall condition.
The sale prices were highest during the late 19th century but fell sharply during the early 20th century and since then the prices have been increasing year by year.
Also the Sale price increases with every remodeling done.
Sale price is highest for houses with Shed roofs and lowest for houses with Gambrel roofs.
Sale price is highest for roofs made with Wood Shingles material and lowest for roofs made with Roll material.
Houses of Stone Masonry Veneer type have the highest sale price while houses of Brick common masonry veneer type have the lowest sale price.
Sale price increases with increase in Masonry veneer area but gradually declines after 1200 square feet area.
Sale price is highest for Poured Concrete foundation and lowest for Slab foundation.
Sale Price increases with increase in Total square feet of Basement area but gradually declines from 2500 sq. ft. area.

Sale prices are highest for houses with central air conditioning.
Prices increase with increase in first floor square feet but there is a gradual decline after 2500 sq. ft.
Prices increase with increase in Second floor sq. ft although not monotonic.
Prices increase with increase in Ground living area sq. ft although there is a sharp decline after 4500 sq. ft. area.
Prices decline with better quality of basement half bathrooms.

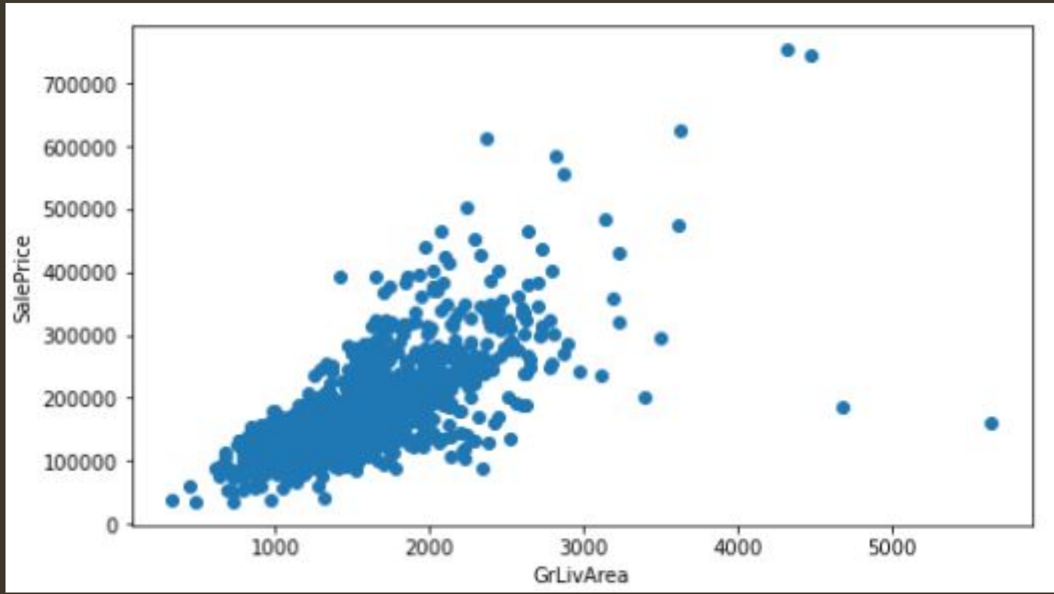
```
sns.countplot(df.YrSold)
```

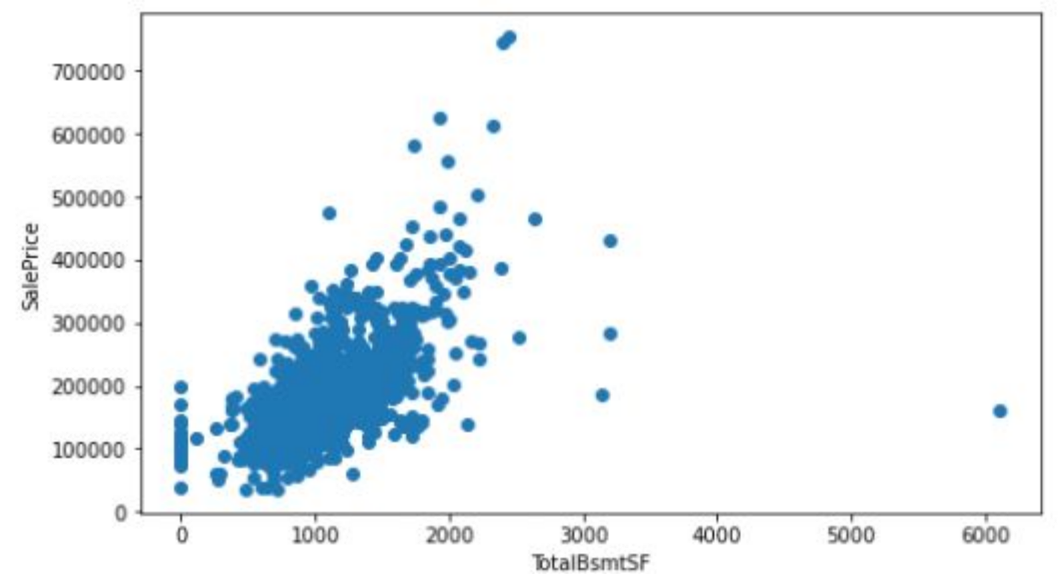
```
<AxesSubplot:xlabel='YrSold', ylabel='count'>
```

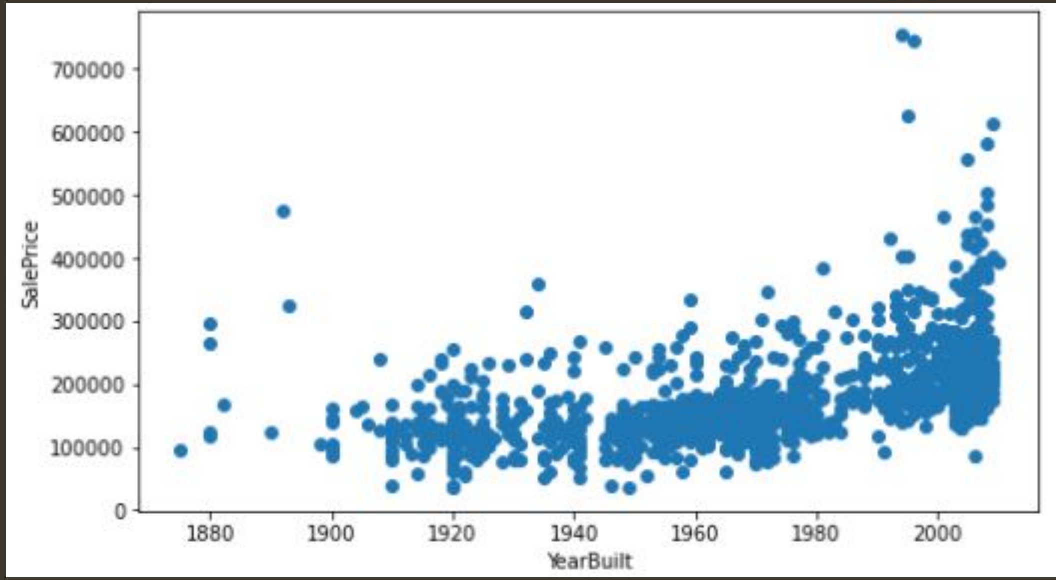


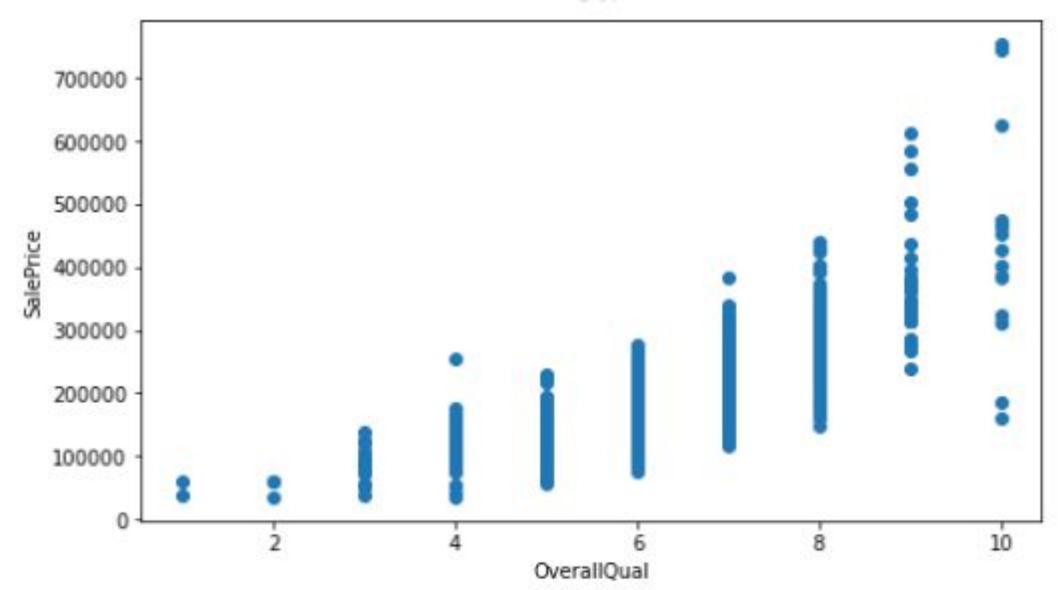

```
plt.figure(figsize=(15,155),facecolor='white')
plotnumber=1

for column in X:
    if plotnumber<=76:
        ax=plt.subplot(38,2,plotnumber)
        plt.scatter(X[column],y)
        plt.xlabel(column,fontsize=10)
        plt.ylabel('SalePrice',fontsize=10)

    plotnumber+=1
plt.tight_layout()
```

Most number of houses were sold in 2009 and the least in 2010.


SalePrice
GrLivArea


SalePrice
TotalBsmtSF


SalePrice
YearBuilt


SalePrice
OverallQual

:Observation

•'GrLivArea' and 'TotalBsmtSF' appear to be correlated with 'SalePrice' in a linear fashion. Both correlations are positive, which implies that when one variable increases, so does the other. In the instance of 'TotalBsmtSF,' the slope of the linear connection is extremely steep.

•'OverallQual' and 'YearBuilt' appear to be connected to 'SalePrice' as well. The relationship appears to be greater in the case of 'OverallQual,' where the scatter plot demonstrates how sales prices increase as overall quality improves.

Data Preprocessing

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

```
df1.drop(columns=['Id','Alley','Utilities','PoolQC','Fence','MiscFeature'],axis=1,inplace=True)
```

As we have seen earlier, there were number of missing values in our train and test datasets. So we will deal with them now. But we also have features that have more than 80% null values. So it's better we drop them along with unwanted columns.

Next, we will deal with the remaining features that has very less percentage of null values. The numerical columns will be replaced with the mean and the categorical columns will be replaced with the mode of respective features. We will do this in train as well as test dataset.

```python
df['LotFrontage']=df['LotFrontage'].fillna(df['LotFrontage'].mean())
df['MasVnrType']=df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])
df['MasVnrArea']=df['MasVnrArea'].fillna(df['MasVnrArea'].mean())
df['BsmtQual']=df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['BsmtCond']=df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtExposure']=df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtFinType1']=df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0])
df['BsmtFinType2']=df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0])
df['FireplaceQu']=df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0])
df['GarageType']=df['GarageType'].fillna(df['GarageType'].mode()[0])
df['GarageYrBlt']=df['GarageYrBlt'].fillna(df['GarageYrBlt'].mean())
df['GarageFinish']=df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual']=df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond']=df['GarageCond'].fillna(df['GarageCond'].mode()[0])
```

```python
df1['LotFrontage']=df1['LotFrontage'].fillna(df1['LotFrontage'].mean())
df1['MasVnrType']=df1['MasVnrType'].fillna(df1['MasVnrType'].mode()[0])
df1['MasVnrArea']=df1['MasVnrArea'].fillna(df1['MasVnrArea'].mean())
df1['BsmtQual']=df1['BsmtQual'].fillna(df1['BsmtQual'].mode()[0])
df1['BsmtCond']=df1['BsmtCond'].fillna(df1['BsmtCond'].mode()[0])
df1['BsmtExposure']=df1['BsmtExposure'].fillna(df1['BsmtExposure'].mode()[0])
df1['BsmtFinType1']=df1['BsmtFinType1'].fillna(df1['BsmtFinType1'].mode()[0])
df1['BsmtFinType2']=df1['BsmtFinType2'].fillna(df1['BsmtFinType2'].mode()[0])
df1['Electrical']=df1['Electrical'].fillna(df1['Electrical'].mode()[0])
df1['FireplaceQu']=df1['FireplaceQu'].fillna(df1['FireplaceQu'].mode()[0])
df1['GarageType']=df1['GarageType'].fillna(df1['GarageType'].mode()[0])
df1['GarageYrBlt']=df1['GarageYrBlt'].fillna(df1['GarageYrBlt'].mean())
df1['GarageFinish']=df1['GarageFinish'].fillna(df1['GarageFinish'].mode()[0])
df1['GarageQual']=df1['GarageQual'].fillna(df1['GarageQual'].mode()[0])
df1['GarageCond']=df1['GarageCond'].fillna(df1['GarageCond'].mode()[0])
```

Now, we will convert all our categorical data into numerical data so that our ML model can understand our data. We will do this in our train as well as test data too.

```
#Converting strings into numberical format

from sklearn.preprocessing import LabelEncoder

lab_enc=LabelEncoder()

df1=lab_enc.fit_transform(df['MSZoning'])
df2=lab_enc.fit_transform(df['Street'])
df3=lab_enc.fit_transform(df['LotShape'])
df4=lab_enc.fit_transform(df['LandContour'])
df6=lab_enc.fit_transform(df['LotConfig'])
df7=lab_enc.fit_transform(df['LandSlope'])
df8=lab_enc.fit_transform(df['Neighborhood'])

df9=lab_enc.fit_transform(df['Condition1'])
df10=lab_enc.fit_transform(df['Condition2'])
df11=lab_enc.fit_transform(df['BldgType'])
df12=lab_enc.fit_transform(df['HouseStyle'])
df13=lab_enc.fit_transform(df['RoofStyle'])
df14=lab_enc.fit_transform(df['RoofMatl'])
df15=lab_enc.fit_transform(df['Exterior1st'])
df16=lab_enc.fit_transform(df['Exterior2nd'])

df17=lab_enc.fit_transform(df['MasVnrType'])
df18=lab_enc.fit_transform(df['ExterQual'])
df19=lab_enc.fit_transform(df['ExterCond'])
df20=lab_enc.fit_transform(df['Foundation'])
df21=lab_enc.fit_transform(df['BsmtQual'])
df22=lab_enc.fit_transform(df['BsmtCond'])
df23=lab_enc.fit_transform(df['BsmtExposure'])
df24=lab_enc.fit_transform(df['BsmtFinType1'])
```

```
df25=lab_enc.fit_transform(df['BsmtFinType2'])
df26=lab_enc.fit_transform(df['Heating'])
df27=lab_enc.fit_transform(df['HeatingQC'])
df28=lab_enc.fit_transform(df['CentralAir'])
df29=lab_enc.fit_transform(df['Electrical'])
df30=lab_enc.fit_transform(df['KitchenQual'])
df31=lab_enc.fit_transform(df['Functional'])
df32=lab_enc.fit_transform(df['FireplaceQu'])

df33=lab_enc.fit_transform(df['GarageType'])
df34=lab_enc.fit_transform(df['GarageFinish'])
df35=lab_enc.fit_transform(df['GarageQual'])
df36=lab_enc.fit_transform(df['GarageCond'])
df37=lab_enc.fit_transform(df['PavedDrive'])
df38=lab_enc.fit_transform(df['SaleType'])
df39=lab_enc.fit_transform(df['SaleCondition'])

df['MSZoning']=df1
df['Street']=df2
df['LotShape']=df3
df['LandContour']=df4
df['LotConfig']=df6
df['LandSlope']=df7
df['Neighborhood']=df8

df['Condition1']=df9
df['Condition2']=df10
df['BldgType']=df11
df['HouseStyle']=df12
df['RoofStyle']=df13
df['RoofMatl']=df14
df['Exterior1st']=df15
df['Exterior2nd']=df16
```

```
df['MasVnrType']=df17
df['ExterQual']=df18
df['ExterCond']=df19
df['Foundation']=df20
df['BsmtQual']=df21
df['BsmtCond']=df22
df['BsmtExposure']=df23
df['BsmtFinType1']=df24

df['BsmtFinType2']=df25
df['Heating']=df26
df['HeatingQC']=df27
df['CentralAir']=df28
df['Electrical']=df29
df['KitchenQual']=df30
df['Functional']=df31
df['FireplaceQu']=df32

df['GarageType']=df33
df['GarageFinish']=df34
df['GarageQual']=df35
df['GarageCond']=df36
df['PavedDrive']=df37
df['SaleType']=df38
df['SaleCondition']=df39
```

```
#Converting strings into numberical format

from sklearn.preprocessing import LabelEncoder

lab_enc=LabelEncoder()

df111=lab_enc.fit_transform(df1['MSZoning'])
df2=lab_enc.fit_transform(df1['Street'])
df3=lab_enc.fit_transform(df1['LotShape'])
df4=lab_enc.fit_transform(df1['LandContour'])
df6=lab_enc.fit_transform(df1['LotConfig'])
df7=lab_enc.fit_transform(df1['LandSlope'])
df8=lab_enc.fit_transform(df1['Neighborhood'])

df9=lab_enc.fit_transform(df1['Condition1'])
df10=lab_enc.fit_transform(df1['Condition2'])
df11=lab_enc.fit_transform(df1['BldgType'])
df12=lab_enc.fit_transform(df1['HouseStyle'])
df13=lab_enc.fit_transform(df1['RoofStyle'])
df14=lab_enc.fit_transform(df1['RoofMatl'])
df15=lab_enc.fit_transform(df1['Exterior1st'])
df16=lab_enc.fit_transform(df1['Exterior2nd'])

df17=lab_enc.fit_transform(df1['MasVnrType'])
df18=lab_enc.fit_transform(df1['ExterQual'])
df19=lab_enc.fit_transform(df1['ExterCond'])
df20=lab_enc.fit_transform(df1['Foundation'])
df21=lab_enc.fit_transform(df1['BsmtQual'])
df22=lab_enc.fit_transform(df1['BsmtCond'])
df23=lab_enc.fit_transform(df1['BsmtExposure'])
df24=lab_enc.fit_transform(df1['BsmtFinType1'])
```

```
df25=lab_enc.fit_transform(df1['BsmtFinType2'])
df26=lab_enc.fit_transform(df1['Heating'])
df27=lab_enc.fit_transform(df1['HeatingQC'])
df28=lab_enc.fit_transform(df1['CentralAir'])
df29=lab_enc.fit_transform(df1['Electrical'])
df30=lab_enc.fit_transform(df1['KitchenQual'])
df31=lab_enc.fit_transform(df1['Functional'])
df32=lab_enc.fit_transform(df1['FireplaceQu'])

df33=lab_enc.fit_transform(df1['GarageType'])
df34=lab_enc.fit_transform(df1['GarageFinish'])
df35=lab_enc.fit_transform(df1['GarageQual'])
df36=lab_enc.fit_transform(df1['GarageCond'])
df37=lab_enc.fit_transform(df1['PavedDrive'])
df38=lab_enc.fit_transform(df1['SaleType'])
df39=lab_enc.fit_transform(df1['SaleCondition'])

df1['MSZoning']=df111
df1['Street']=df2
df1['LotShape']=df3
df1['LandContour']=df4
df1['LotConfig']=df6
df1['LandSlope']=df7
df1['Neighborhood']=df8

df1['Condition1']=df9
df1['Condition2']=df10
df1['BldgType']=df11
df1['HouseStyle']=df12
df1['RoofStyle']=df13
df1['RoofMatl']=df14
df1['Exterior1st']=df15
df1['Exterior2nd']=df16
```

```
df1['MasVnrType']=df17
df1['ExterQual']=df18
df1['ExterCond']=df19
df1['Foundation']=df20
df1['BsmtQual']=df21
df1['BsmtCond']=df22
df1['BsmtExposure']=df23
df1['BsmtFinType1']=df24

df1['BsmtFinType2']=df25
df1['Heating']=df26
df1['HeatingQC']=df27
df1['CentralAir']=df28
df1['Electrical']=df29
df1['KitchenQual']=df30
df1['Functional']=df31
df1['FireplaceQu']=df32

df1['GarageType']=df33
df1['GarageFinish']=df34
df1['GarageQual']=df35
df1['GarageCond']=df36
df1['PavedDrive']=df37
df1['SaleType']=df38
df1['SaleCondition']=df39
```

Finding Correlation:

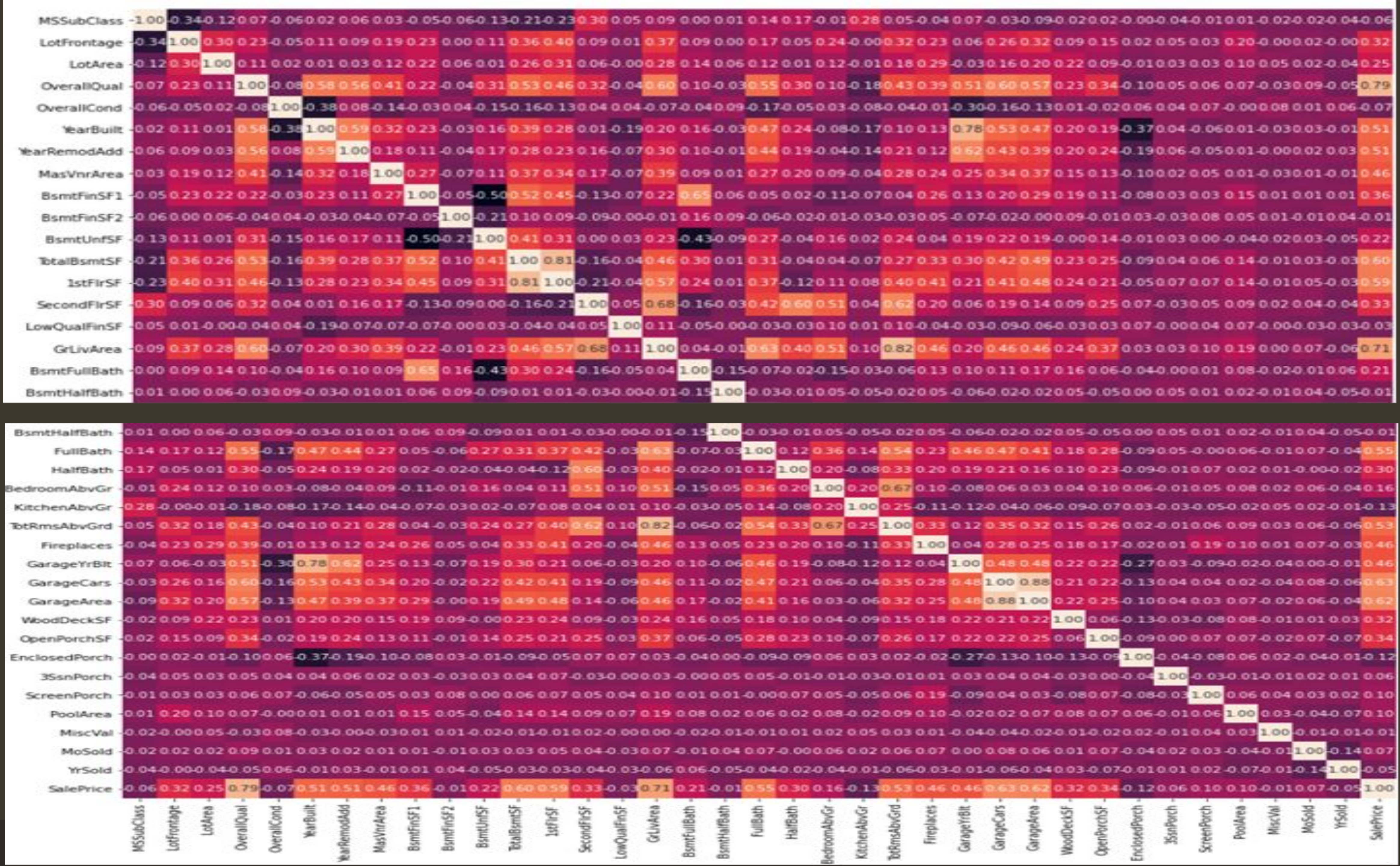
```
df_corrnumericals=df[['MSSubClass','LotFrontage','LotArea','OverallQual','OverallCond','YearBuilt','YearRemodAdd','M
1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','1stFlrSF','SecondFlrSF','LowQualFinSF','GrLivArea','BsmtFullBath','BsmtHa
alfBath','BedroomAbvGr','KitchenAbvGr','TotRmsAbvGrd','Fireplaces','GarageYrBlt','GarageCars','GarageArea','WoodDeck
closedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal','MoSold','YrSold','SalePrice']]
```

```
df_corrcategorical=df[['MSZoning','Street','LotShape','LandContour','LotConfig','LandSlope','Neighborhood','Conditio
gType','HouseStyle','RoofStyle','RoofMatl','Exterior1st','Exterior2nd','MasVnrType','ExterQual','ExterCond','Foundat
Cond','BsmtExposure','BsmtFinType1','BsmtFinType2','Heating','HeatingQC','CentralAir','Electrical','KitchenQual','Fu
u','GarageType','GarageFinish','GarageQual','GarageCond','PavedDrive','SaleType','SaleCondition','SalePrice']]
```

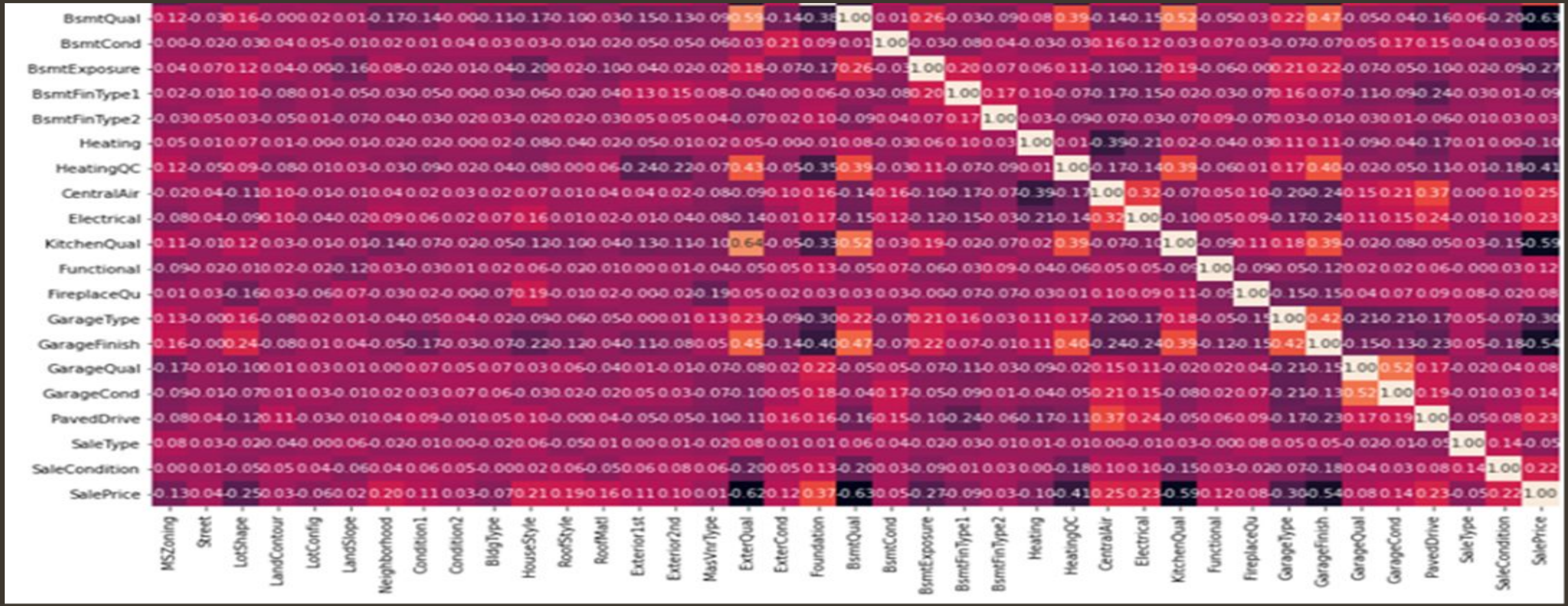
```
corr_1=df_corrnumericals.corr()
print(corr_1.shape)
corr_2=df_corrcategorical.corr()
print(corr_2.shape)
```

```
(37, 37)
(39, 39)
```

```
plt.figure(figsize=(20,20))

sns.heatmap(corr_1,cbar=True,square=True,cbar_kws={'shrink':0.82},fmt='.2f',annot=True,annot_kws={'size':10})
plt.show()
```


MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
SecondFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice

```python
plt.figure(figsize=(20,20))

sns.heatmap(corr_2,cbar=True,square=True,cbar_kws={'shrink':0.82},fmt='.2f',annot=True,annot_kws={'size':10})
plt.show()
```

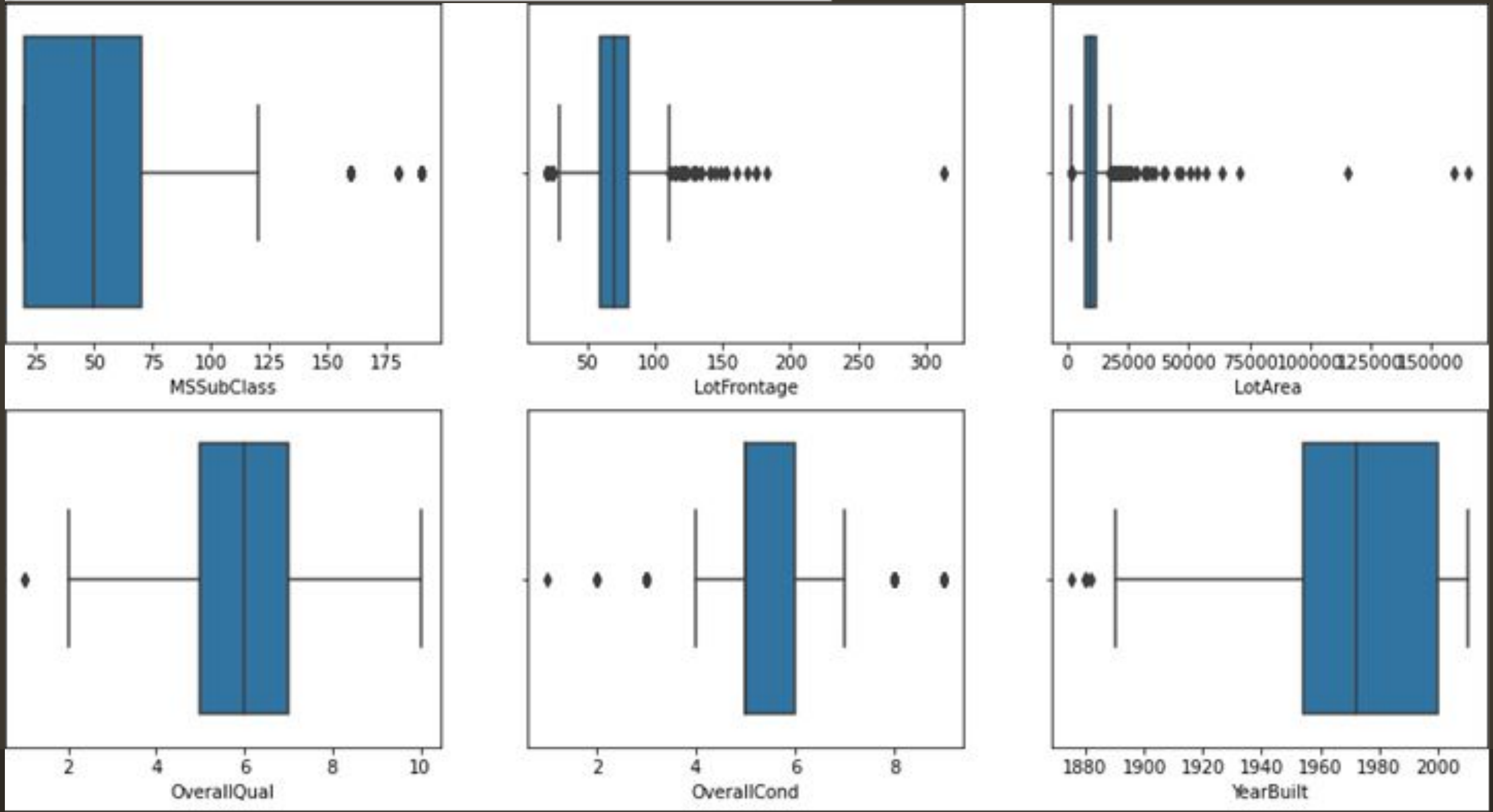


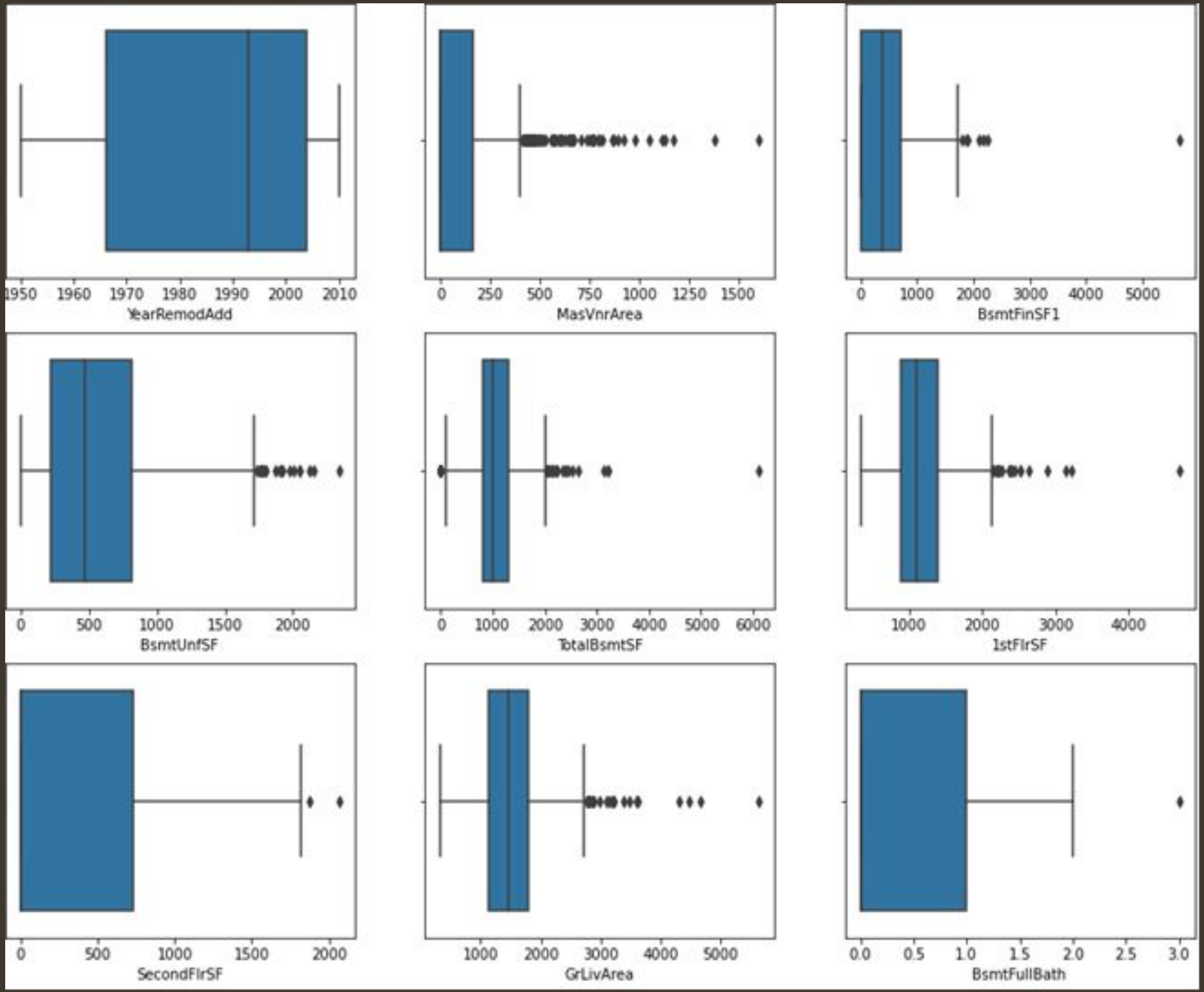
Observations:

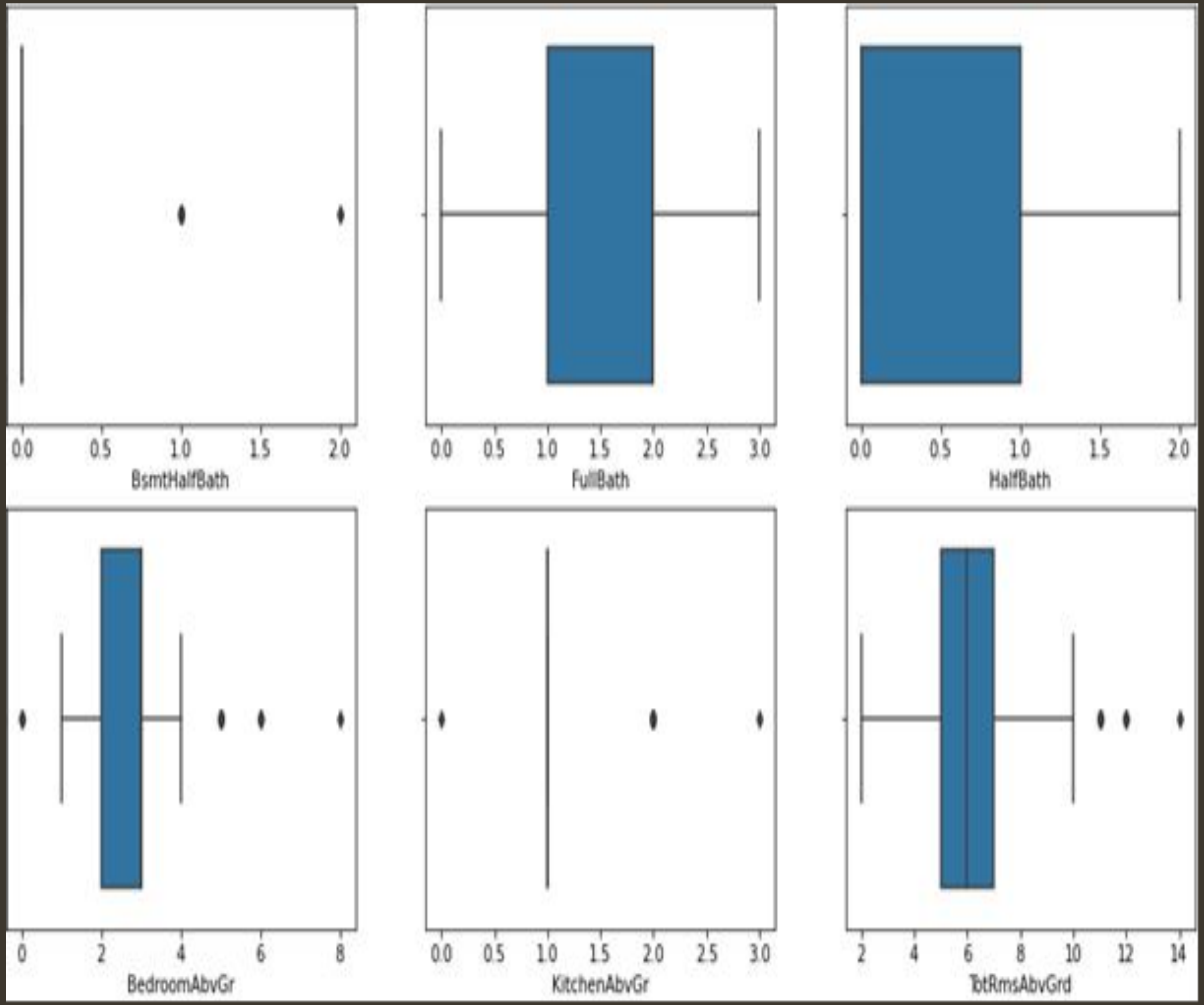
- OverallQual has the highest positive correlation with SalePrice followed closely by GrLivarea while OverallCond, KitchenAbvGr, EnclosedPorch have the lowest correlations with SalePrice. (Numerical values)
- GarageCars and GarageArea are also some of the most strongly correlated variables.
- TotalBsmtSF and 1stFlrSF are also strongly correlated variables.
- TotRmsAbvGrd and GrLivArea are also strongly correlated.

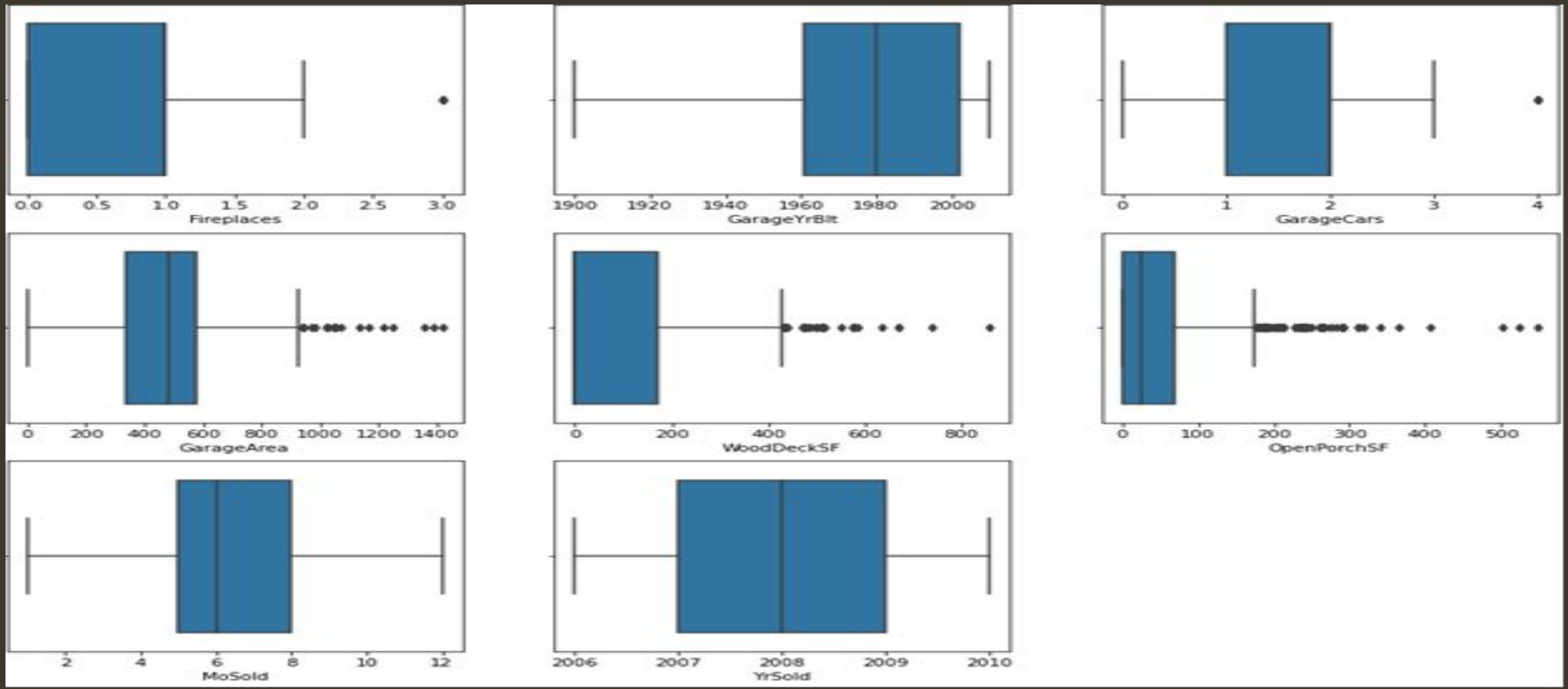
Finding Outliers (Train Dataset):

```
plt.figure(figsize=(15,50))
graph=1

for column in df_numericals:
    if(graph<=36):
        ax=plt.subplot(12,3,graph)
        sns.boxplot(df_numericals[column],orient='v')
        plt.xlabel(column,fontsize=10)
    graph+=1
plt.show()
```




1950 1960 1970 1980 1990 2000 2010
YearRemodAdd
0 250 500 750 1000 1250 1500
MasVnrArea
0 1000 2000 3000 4000 5000
BsmtFinSF1
0 500 1000 1500 2000
BsmtUnfSF
0 1000 2000 3000 4000 5000 6000
TotalBsmtSF
1000 2000 3000 4000
1stFlrSF
0 500 1000 1500 2000
SecondFlrSF
1000 2000 3000 4000 5000
GrLivArea
0.0 0.5 1.0 1.5 2.0 2.5 3.0
BsmtFullBath


0.0 0.5 1.0 1.5 2.0
BsmtHalfBath
0.0 0.5 1.0 1.5 2.0 2.5 3.0
FullBath
0.0 0.5 1.0 1.5 2.0
HalfBath
0 2 4 6 8
BedroomAbvGr
0.0 0.5 1.0 1.5 2.0 2.5 3.0
KitchenAbvGr
2 4 6 8 10 12 14
TotRmsAbvGrd



Removing Ouliers:

```
#Find the IQR (inter quantile range) to identify outliers

q1=df.quantile(0.25) #1st quantile
q3=df.quantile(0.75) #3rd quantile

#IQR
iqr=q3-q1
iqr
```

```
MSSubClass          50.00
MSZoning             0.00
LotFrontage         19.25
LotArea           3894.00
Street               0.00
                    ...
MoSold               3.00
YrSold               2.00
SaleType             0.00
SaleCondition        0.00
SalePrice        84625.00
Length: 75, dtype: float64
```

```python
index=np.where(df['BsmtFinSF1']>(q3.BsmtFinSF1)+(1.5*iqr.BsmtFinSF1))
df=df.drop(df.index[index])
print('Shape:',df.shape)
df.reset_index()
```

Shape: (1161, 75)

```python
index=np.where(df['TotalBsmtSF']<(q1.TotalBsmtSF)-(1.5*iqr.TotalBsmtSF))
df=df.drop(df.index[index])
print('Shape:',df.shape)
df.reset_index()
```

Shape: (1131, 75)

```python
index=np.where(df['LotArea']<(q1.LotArea)-(1.5*iqr.LotArea))
df=df.drop(df.index[index])
print('Shape:',df.shape)
df.reset_index()
```

Shape: (1120, 75)

```python
index=np.where(df['YearBuilt']<(q1.YearBuilt)-(1.5*iqr.YearBuilt))
df=df.drop(df.index[index])
print('Shape:',df.shape)
df.reset_index()
```

Shape: (1114, 75)

```python
index=np.where(df['GarageYrBlt']<(q1.GarageYrBlt)-(1.5*iqr.GarageYrBlt))
df=df.drop(df.index[index])
print('Shape:',df.shape)
df.reset_index()
```

Shape: (1113, 75)

```python
index=np.where(df['SecondFlrSF']>(q3.SecondFlrSF)+(1.5*iqr.SecondFlrSF))
df=df.drop(df.index[index])
print('Shape:',df.shape)
df.reset_index()
```

Shape: (1112, 75)

```python
(1168-1112)/1168*100
```

4.794520547945205

We did not remove all the outliers as that would mean we will have to lose more than 30% of the data. Right now, we have lost around 4.8% data which is affordable. The same is done for test data as well.

Removing Skewness:

```
df_numericals.skew()
```

```
MSSubClass        1.422019
LotFrontage       2.450241
LotArea          10.659285
OverallQual       0.175082
OverallCond       0.580714
YearBuilt        -0.579204
YearRemodAdd     -0.495864
MasVnrArea        2.826173
BsmtFinSF1        1.871606
BsmtUnfSF         0.909057
TotalBsmtSF       1.744591
1stFlrSF          1.513707
SecondFlrSF       0.823479
GrLivArea         1.449952
BsmtFullBath      0.627106
BsmtHalfBath      4.264403
FullBath          0.057809
HalfBath          0.656492
BedroomAbvGr      0.243855
KitchenAbvGr      4.365259
TotRmsAbvGrd      0.644657
Fireplaces        0.671966
GarageYrBlt      -0.644564
GarageCars       -0.358556
GarageArea        0.189665
WoodDeckSF        1.504929
OpenPorchSF       2.410840
MoSold            0.220979
YrSold            0.115765
dtype: float64
```

```
df['LotFrontage']=np.sqrt(df['LotFrontage'])
df['LotArea']=np.sqrt(df['LotArea'])
df['MasVnrArea']=np.sqrt(df['MasVnrArea'])
df['BsmtFinSF1']=np.sqrt(df['BsmtFinSF1'])
df['BsmtFinSF2']=np.sqrt(df['BsmtFinSF2'])
df['BsmtUnfSF']=np.sqrt(df['BsmtUnfSF'])
df['TotalBsmtSF']=np.sqrt(df['TotalBsmtSF'])
df['1stFlrSF']=np.sqrt(df['1stFlrSF'])
df['SecondFlrSF']=np.sqrt(df['SecondFlrSF'])
df['LowQualFinSF']=np.sqrt(df['LowQualFinSF'])
df['GrLivArea']=np.sqrt(df['GrLivArea'])
df['WoodDeckSF']=np.sqrt(df['WoodDeckSF'])
df['OpenPorchSF']=np.sqrt(df['OpenPorchSF'])
df['EnclosedPorch']=np.sqrt(df['EnclosedPorch'])
df['3SsnPorch']=np.sqrt(df['3SsnPorch'])
df['ScreenPorch']=np.sqrt(df['ScreenPorch'])
df['PoolArea']=np.sqrt(df['PoolArea'])
df['MiscVal']=np.sqrt(df['MiscVal'])
```

We have removed the skewness for all those variables that had skewness greater than -0.5 and 0.5.

Selecting Best Features:

```
from sklearn.feature_selection import SelectKBest, f_classif
```

```
y=df['SalePrice']
X=df.drop(columns=['SalePrice'])
```

```
best_features=SelectKBest(score_func=f_classif,k=40)
fit=best_features.fit(X,y)
df_scores=pd.DataFrame(fit.scores_)
df_columns=pd.DataFrame(X.columns)

feature_scores=pd.concat([df_columns,df_scores],axis=1)
feature_scores.columns=['Feature_Name','Score'] #name output columns
print(feature_scores.nlargest(40,'Score')) #print 40 best features
```

	Feature_Name	Score
14	OverallQual	5.149582
24	ExterQual	3.572308
43	GrLivArea	3.126663
27	BsmtQual	2.807488
50	KitchenQual	2.738808
58	GarageCars	2.468362
46	FullBath	2.432211
59	GarageArea	2.373555
40	1stFlrSF	2.268067
57	GarageFinish	2.190499
16	YearBuilt	2.179121
35	TotalBsmtSF	2.123556
17	YearRemodAdd	1.821069
4	Street	1.816640
56	GarageYrBlt	1.751381
1	MSZoning	1.687061
51	TotRmsAbvGrd	1.629845
26	Foundation	1.585004
53	Fireplaces	1.584844
23	MasVnrArea	1.580853
64	OpenPorchSF	1.570705
3	LotArea	1.543094
69	MiscVal	1.490801
38	CentralAir	1.451007
31	BsmtFinSF1	1.449826
47	HalfBath	1.351555
5	LotShape	1.343748
37	HeatingQC	1.318367
36	Heating	1.288921
28	BsmtCond	1.268416
9	Neighborhood	1.258020
63	WoodDeckSF	1.251856
2	LotFrontage	1.236566
54	FireplaceQu	1.203226
55	GarageType	1.194020
41	SecondFlrSF	1.189878
29	BsmtExposure	1.183483
22	MasVnrType	1.171093
39	Electrical	1.141787
12	BldgType	1.116274

Here we have selected 40 best features with respect to SalePrice among 74 features.

Next, we will do feature scaling and look for correlated features and remove the multicollinearity.

```
#Feature Scaling
scaler=StandardScaler()
X_scaler=scaler.fit_transform(X)

vif=pd.DataFrame()
vif['score']=[variance_inflation_factor(X_scaler,i) for i in range(X_scaler.shape[1])]
vif['Features']=X.columns

vif
```

	score	Features
0	3.411386	OverallQual
1	2.424910	ExterQual
2	43.818682	GrLivArea
3	2.041251	BsmtQual
4	2.046020	KitchenQual
5	5.528327	GarageCars
6	3.070291	FullBath
7	5.342874	GarageArea
8	28.276521	1stFlrSF
9	1.948804	GarageFinish
10	6.024791	YearBuilt
11	4.947137	TotalBsmtSF
12	2.444004	YearRemodAdd
13	1.145724	Street
14	3.557388	GarageYrBlt
15	1.310950	MSZoning
16	3.769092	TotRmsAbvGrd
17	2.220164	Foundation
18	1.757329	Fireplaces
19	1.839734	MasVnrArea
20	1.423990	OpenPorchSF
21	1.775042	LotArea
22	1.058512	MiscVal
23	1.530886	CentralAir
24	1.306240	BsmtFinSF1
25	2.336808	HalfBath
26	1.178640	LotShape
27	1.607399	HeatingQC
28	1.244246	Heating
29	1.082713	BsmtCond
30	1.195169	Neighborhood
31	1.235987	WoodDeckSF
32	1.699123	LotFrontage
33	1.409648	FireplaceQu
34	1.699595	GarageType
35	33.779819	SecondFlrSF
36	1.305831	BsmtExposure
37	1.418914	MasVnrType
38	1.281845	Electrical
39	1.651884	BldgType

We had seen earlier from the heatmap that SecondFlrSF was correlated with GrLivArea. So now will be dropping SecondFlrSF and that would remove the multicollinearity issue from all other variables.

```
X.drop(columns=['SecondFlrSF'],axis=1,inplace=True)
```

Identification of possible problem-solving approaches (methods)

Given that this is a regression problem with an output of 'Sale Price,' we will employ regression models such as Linear Regression, KNeighbors Regressor, Decision Tree Regressor, Gradient Boosting Regressor, Random Forest Regressor, Ada Boost Regressor, and so on. We will use these algorithms to train our training data, and then we will test on the test data set for final house price prediction. The algorithm with the highest accuracy and the least difference between Cross validation and r2 scores will be chosen as the final model.

Testing of Identified Approaches (Algorithms)

KNeighborsRegressor, Decision Tree Regressor, Gradient Boosting Regressor, Lasso, Random Forest Regressor, Ada Boost Regressor, and Linear Regression methods will be used here.

Run and Evaluate selected models

We will first find the best random state and then predict our test data with the respective algorithms.

```
maxAccu=0
maxRs=0
for i in range(1,200):
    X_train,x_test,Y_train,y_test=train_test_split(X_scaler,y,test_size=0.25,random_state=i)
    mod=DecisionTreeRegressor()
    mod.fit(X_train,Y_train)
    pred=mod.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRs=i
print("Best accuracy is:",maxAccu,"on Random State",maxRs)
```

```
Best accuracy is: 0.80209554248774 on Random State 185
```

Our best random state is 185 which gives the accuracy of 80.2%. We will use this random state for all the models.

```
X_train,x_test,Y_train,y_test=train_test_split(X_scaler,y,test_size=0.25,random_state=185)
```

```
DTC=DecisionTreeRegressor()
DTC.fit(X_train,Y_train)
pred=DTC.predict(x_test)
print(r2_score(y_test,pred))
```

0.7365631694095114

```
lr=LinearRegression()
lr.fit(X_train,Y_train)
pred=lr.predict(x_test)
print(r2_score(y_test,pred))
```

0.8838209898296076

```
RFR=RandomForestRegressor()
RFR.fit(X_train,Y_train)
pred=RFR.predict(x_test)
print(r2_score(y_test,pred))
```

0.8866947818540951

```
from sklearn.ensemble import AdaBoostRegressor
ada=AdaBoostRegressor()
ada.fit(X_train,Y_train)
pred=ada.predict(x_test)
print(r2_score(y_test,pred))
```

0.85029516538742

```
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor()
gbr.fit(X_train,Y_train)
pred=gbr.predict(x_test)
print(r2_score(y_test,pred))
```

0.9130577965562511

```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
knn.fit(X_train,Y_train)
pred=knn.predict(x_test)
print(r2_score(y_test,pred))
```

0.8391263401691001

```
import xgboost as xgb
xgb=xgb.XGBRegressor()
xgb.fit(X_train,Y_train)
pred=xgb.predict(x_test)
print(r2_score(y_test,pred))
```

0.9008225747150398

We have different accuracy for different models:

- Decision Tree Regressor - 74%
- Linear Regression - 88%
- Random Forest Regressor - 89%
- Ada Boost Regressor - 85%
- Gradient Boosting Regressor - 91%
- KNeighbors Regressor - 84%
- XGB Regressor - 90%

We will be regularizing our model by Lasso just in case if we have an over fitting model.

```
from sklearn.linear_model import Lasso
```

```
parameters={'alpha':[.0001,.001,.01,.1,1,10],'random_state':list(range(0,10))}
ls=Lasso()
clf=GridSearchCV(ls,parameters)
clf.fit(X_train,Y_train)
print(clf.best_params_)
```

```
{'alpha': 10, 'random_state': 0}
```

```
ls=Lasso(alpha=10,random_state=0)
ls.fit(X_train,Y_train)
ls.score(X_train,Y_train)
pred_ls=ls.predict(x_test)

lss=r2_score(y_test,pred_ls)
lss
```

```
0.8838455350351817
```

We will now be performing cross validation method.

```
from sklearn.model_selection import cross_val_score
```

```
print(cross_val_score(DTC,X_scaler,y,cv=5).mean())
```

```
0.709551159819362
```

```
print(cross_val_score(lr,X_scaler,y,cv=5).mean())
```

```
0.8536637395737239
```

```
print(cross_val_score(RFR,X_scaler,y,cv=5).mean())
```

```
0.8614677527483913
```

```
print(cross_val_score(ada,X_scaler,y,cv=5).mean())
```

```
0.812295753936262
```

```
print(cross_val_score(gbr,X_scaler,y,cv=5).mean())
```

```
0.8773549402659715
```

```
print(cross_val_score(knn,X_scaler,y,cv=5).mean())
```

```
0.7973955361663048
```

```
print(cross_val_score(xgb,X_scaler,y,cv=5).mean())
```

```
0.8591662058575767
```

From the above technique, we can see that Gradient Boosting Regressor has the least difference between cross validation score and r2 score. Therefore, GBR is our best model.

Now we will be performing hyperparameter tuning to our best model just to see if it can increase the accuracy.

```
parameters={'criterion':['friedman_mse','squared_error','mse','mae'],'max_features':['auto','sqrt','log2'],'n_estimators':[40,47,49,5
0],'learning_rate':[0.30,0.40,0.45],'loss':['squared_error','ls','absolute_error','lad','huber','quantile']}
GBR=GradientBoostingRegressor()
clf=RandomizedSearchCV(GBR, cv=5, param_distributions=parameters)
clf.fit(X_train,Y_train)

print(clf.best_params_)
```

```
{'n_estimators': 47, 'max_features': 'auto', 'loss': 'ls', 'learning_rate': 0.3, 'criterion': 'mse'}
```

```
Final_model=GradientBoostingRegressor(max_features='auto',criterion='mse',n_estimators=47,loss='ls',learning_rate=0.3)
Final_model.fit(X_train,Y_train)
pred=Final_model.predict(x_test)
acc=r2_score(y_test,pred)
print('Accuracy:',acc*100)
```

```
Accuracy: 90.51786813222704
```

After hyperparameter also we see the accuracy is around 91% only. So we can conclude that there is no over fitting issue with our model.
We will now save our model and then predict our test dataset.

```
import pickle
filename='FinalisedModel_Housing_Final.pkl'
pickle.dump(gbr,open(filename,'wb'))
```

We will be using the same 40 features used in train dataset to predict our test dataset.

```
df1=df1[['OverallQual','ExterQual','GrLivArea','BsmtQual','KitchenQual','GarageCars','FullBath','GarageArea','1stFlrSF','GarageFinish','Y
earBuilt','TotalBsmtSF','YearRemodAdd','Street','GarageYrBlt','MSZoning','TotRmsAbvGrd','Foundation','Fireplaces','MasVnrArea','OpenPorch
SF','LotArea','MiscVal','CentralAir','BsmtFinSF1','HalfBath','LotShape','HeatingQC','Heating','BsmtCond','Neighborhood','WoodDeckSF','Lot
Frontage','FireplaceQu','GarageType','SecondFlrSF','BsmtExposure','MasVnrType','Electrical','BldgType']]
```

Now we will be performing the same steps that we did for train dataset.

```
#Feature Scaling
scaler=StandardScaler()
X_scaled=scaler.fit_transform(df1)

vif=pd.DataFrame()
vif['score']=[variance_inflation_factor(X_scaler,i) for i in range(X_scaled.shape[1])]
vif['Features']=df1.columns

vif
```

	score	Features
0	3.411386	OverallQual
1	2.424910	ExterQual
2	43.818682	GrLivArea
3	2.041251	BsmtQual
4	2.046020	KitchenQual
5	5.528327	GarageCars
6	3.070291	FullBath
7	5.342874	GarageArea
8	28.276521	1stFlrSF
9	1.948804	GarageFinish
10	6.024791	YearBuilt
11	4.947137	TotalBsmtSF
12	2.444004	YearRemodAdd
13	1.145724	Street
14	3.557388	GarageYrBlt
15	1.310950	MSZoning
16	3.769092	TotRmsAbvGrd
17	2.220164	Foundation
18	1.757329	Fireplaces
19	1.839734	MasVnrArea
20	1.423990	OpenPorchSF
21	1.775042	LotArea
22	1.058512	MiscVal
23	1.530886	CentralAir
24	1.306240	BsmtFinSF1
25	2.336808	HalfBath
26	1.178640	LotShape
27	1.607399	HeatingQC
28	1.244246	Heating
29	1.082713	BsmtCond
30	1.195169	Neighborhood
31	1.235987	WoodDeckSF
32	1.699123	LotFrontage
33	1.409648	FireplaceQu
34	1.699595	GarageType
35	33.779819	SecondFlrSF
36	1.305831	BsmtExposure
37	1.418914	MasVnrType
38	1.281845	Electrical
39	1.651884	BldgType

```python
df1.drop(columns=['SecondFlrSF'], inplace=True)
```

```python
y_pred=gbr.predict(X_scaled)
```

```python
y_pred
```

```
array([383758.40803782, 231557.87362096, 247082.86042353, 184402.15237363,
       226862.74576426,  78557.47047957, 138070.26071821, 368695.88050689,
       229370.63393585, 163182.08564235,  82345.18541002, 142031.13420669,
       138084.79680097, 308889.12586795, 103752.58616021, 120004.26461187,
       125197.64109978, 177762.37198688, 200216.11442486, 153513.37954415,
       141866.11948935, 152752.63872363,  84862.48762455,  98848.17413772,
       128796.81751758, 179783.20105104, 141221.55867706, 173073.94298595,
        89159.49457403, 156042.2998346 , 192958.27907928, 222782.70957811,
       173662.47034021, 118086.02408583, 173079.22834807, 206227.37528479,
       122286.51162894, 164286.64705234, 144697.13848417, 104409.05947711,
       333818.33905954, 209207.31640699, 193085.0412121 , 139973.32163591,
       127098.37122421, 130661.30906766, 105217.62577411, 236513.34390489,
       361953.7391358 , 148851.65606111, 193770.49798553, 100373.29946484,
       105582.64695247, 286675.25098156, 130514.09647305, 151297.90162487,
       192553.51020931, 109291.77190731, 251196.29874042, 108694.52598139,
       182530.87037475, 131914.29563989, 140831.57000582, 202071.91726372,
       107405.72065733, 150943.88042772, 206993.98729889, 148072.88883143,
       154532.77562103, 167447.29083572, 195601.91304222, 176883.66555919,
       140402.44835703, 257797.86085456, 317317.64044921, 198916.1639511 ,
       299810.45197256, 141545.19947471, 226917.37420583, 148653.84293245,
       157694.18329296, 163827.75374673, 202723.64476519, 123809.0855903 ,
       156815.44305583, 179783.20105104, 238952.88870026, 146681.60508593,
       137221.89104153, 135443.11360286, 191913.63030858, 159305.77461663,
       234234.53289139, 171623.23113299, 125596.90162723, 293606.20866003,
        98841.14545362, 123429.22159439, 156195.70204782, 213030.50473526,
       138483.62728984, 280637.48934341, 158111.95532761, 191502.69943788,
       209109.30618029, 218580.97385375, 179134.80669884, 233428.84345863,
       239912.29397233, 137468.73566405,  85620.72773386, 146096.59311672,
       197023.55548148, 152171.83799457, 110190.54612759, 106397.02147164,
       193377.00882873, 265799.37617321, 130249.09494917, 149778.38926279,
       195197.16678502, 121228.15543278, 180355.2313192 ,  97411.47899128,
```

```
121733.62072721, 141038.87947824, 236759.37606523, 130939.22875268,
150902.16652568, 195427.28866893, 325836.14149558, 200089.44930307,
115948.91483575, 305495.76693718, 118409.28746018, 143280.60077654,
 96536.49920751, 203757.29040816, 255153.0394448 , 169343.69290199,
118999.92986156, 115538.5008439 , 201097.66158779, 169851.08579319,
146142.05737475, 198433.95081686, 115329.74637619,  93081.17745538,
176475.66047802, 187333.224998   , 136727.12766582, 166459.81362032,
206331.07110725, 137729.05900198, 203823.14969595, 120141.59474131,
102911.82851857, 258509.69381642, 188498.49414536, 203558.66934299,
126015.0285423 , 238468.96757435, 160687.28734951, 127961.36271347,
139876.8702592 , 273783.9989699 , 136748.85840934, 438786.51599164,
139229.5451122 , 116263.56862946, 153153.52695155, 164488.99654425,
213571.77678557, 161507.96053462, 259488.0391947 , 175392.15355026,
397597.02569184, 239586.78435773,  92368.98889832, 167896.78035439,
151424.04833714, 113085.08521754, 208355.94346526, 197467.28919857,
 95318.54775105, 150928.75398698,  69760.96850396, 162167.16919692,
199636.4202282 , 149527.86639572, 191213.48651822, 149225.8379538 ,
108051.55650792, 298812.22988223, 346132.62627036, 133371.90954135,
123630.49060797, 265870.40142091, 151265.2698544 , 139473.55701306,
167033.28036958,  97716.99543965, 162349.62585492, 149682.38916514,
179536.58580427,  79498.38289628, 141456.34567729, 146897.31457397,
126186.868702   , 175854.6622627 , 213435.04186693, 213178.33065036,
264587.92794254, 140094.68886049, 196033.46343267, 359203.5399148 ,
227575.61729466, 107461.34264754, 390913.18893392, 180551.12460063,
110937.74367547, 151932.33923318, 101103.13229247, 137696.67021438,
119368.1103506 , 218190.9099679 , 176930.2654538 , 163506.64732092,
245561.84489514, 178191.88195015, 220630.47120693, 139609.67332661,
102796.20769476, 144258.82306709, 259758.20722607, 206864.77609293,
272190.15314902, 148203.08524477, 153600.41659699, 178441.45873773,
267304.30929483, 163781.56406917, 118938.55882194, 113365.98791761,
210124.71290769, 137528.41795792, 393085.47228527, 191950.88977736,
124069.73300841, 196152.227229   , 182484.46729341, 102967.68516744,
149097.67950707, 215822.23606744, 166207.79151961,  95147.31174199,
178189.14093928, 264908.71534707, 270002.19080601, 166326.42152069,
127760.63069575, 356973.46658465, 263674.79267434, 318460.65250079,
225699.66074802, 178078.78908363, 145959.99167323, 187443.90270853,
127602.28927559, 339982.11733349, 123585.47126877, 313763.59047613,
136516.0882395 , 131454.72129999, 167578.88383783, 242950.24142664,
146753.66361206, 153597.9880393 , 148764.19748938, 110179.06342117])
```

CONCLUSION

Key Findings and Conclusions of the Study

In this we found that Variables like OverallQual (overall material and finish of the house), Year Built, TotRmsAbvGrd (Total rooms above grade (does not include bathrooms), GarageCars (Size of garage in car capacity), GarageArea (Size of garage in square feet), GrLivArea (Above grade (ground) living area square feet), FullBath (Full bathrooms above grade) have positive relationship with the sales Price and they affect the sales price hence these factors should be considered.

Learning Outcomes of the Study in respect of Data Science

The objective is to create a system that will minimize the amount of time it takes to find a property at a fair price. The House Price Prediction model makes an attempt to attain the same result. Using several machine learning approaches, the system focuses on estimating the property price based on the neighborhood and other key factors. The experimental results demonstrate that the approach employed will provide accurate house price forecast.

Limitations of this work and Scope for Future Work

- The amount of data is quite limited; it would be preferable to have more data to more correctly forecast the sale price.
- There are many outliers in the given data, and I was unable to eliminate all of them due to the risk of losing data. With more data, more outliers can be removed from the dataset.