In [35]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [36]:
```
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
import warnings
warnings.filterwarnings("ignore")
```

In [37]:
```
data=pd.read_csv(r'C:\Users\18F17955\Downloads\HousePricePrediction.csv')
data.head()
```

Out[37]:

	Id	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt	YearRemodA
0	0	60	RL	8450.0	Inside	1Fam	5	2003	20
1	1	20	RL	9600.0	FR2	1Fam	8	1976	19
2	2	60	RL	11250.0	Inside	1Fam	5	2001	20
3	3	70	RL	NaN	Corner	1Fam	5	1915	19
4	4	60	RL	14260.0	FR2	1Fam	5	2000	20

In [38]:
```
data.shape
```

Out[38]:
```
(2919, 13)
```

In [39]:
```
data.isnull().sum()
```

Out[39]:
```
Id                0
MSSubClass        0
MSZoning          4
LotArea           9
LotConfig         0
BldgType          0
OverallCond       0
YearBuilt         0
YearRemodAdd      0
Exterior1st       1
BsmtFinSF2        1
TotalBsmtSF       1
SalePrice      1459
dtype: int64
```

In [40]:
```
data=data.dropna()
```

In [41]:
```
data.isnull().sum()
```

Out[41]:
```
Id              0
MSSubClass      0
MSZoning        0
LotArea         0
LotConfig       0
BldgType        0
OverallCond     0
YearBuilt       0
YearRemodAdd    0
Exterior1st     0
BsmtFinSF2      0
TotalBsmtSF     0
SalePrice       0
dtype: int64
```

In [42]:
```
data.shape
```

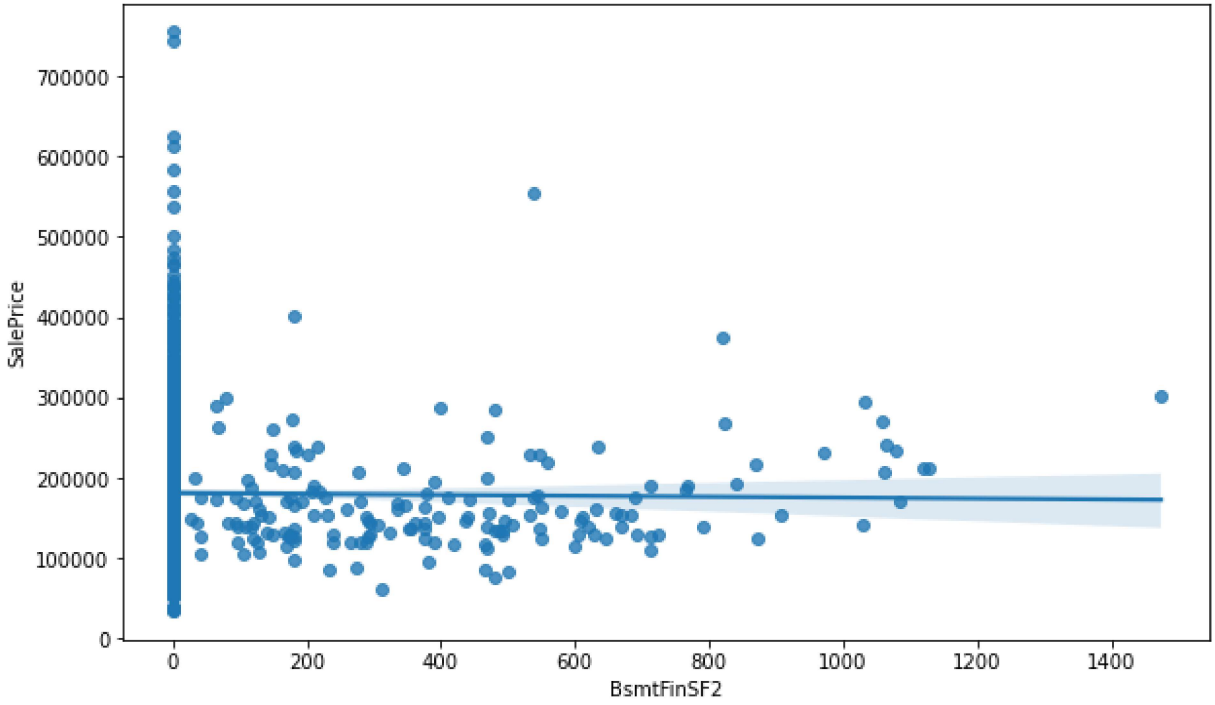
Out[42]:
```
(1451, 13)
```

In [43]:
```
data.dtypes
```

Out[43]:
```
Id               int64
MSSubClass       int64
MSZoning        object
LotArea        float64
LotConfig       object
BldgType        object
OverallCond      int64
YearBuilt        int64
YearRemodAdd     int64
Exterior1st     object
BsmtFinSF2     float64
TotalBsmtSF    float64
SalePrice      float64
dtype: object
```

In [44]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="BsmtFinSF2", y="SalePrice", data=data)
```

Out[44]: <AxesSubplot:xlabel='BsmtFinSF2', ylabel='SalePrice'>

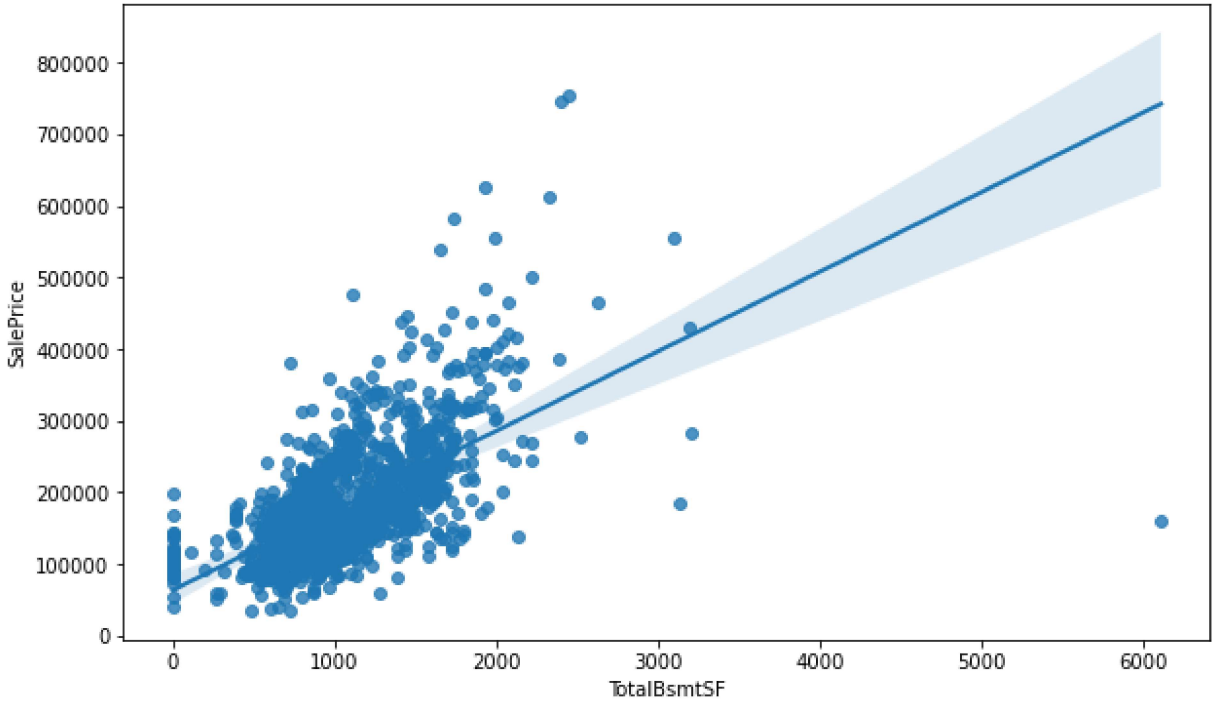


In [45]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(data['BsmtFinSF2'], data['SalePrice'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is -0.011518533534484295  with a P-value of
P = 0.6610961159429988
```

In [46]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="TotalBsmtSF", y="SalePrice", data=data)
```

Out[46]: <AxesSubplot:xlabel='TotalBsmtSF', ylabel='SalePrice'>

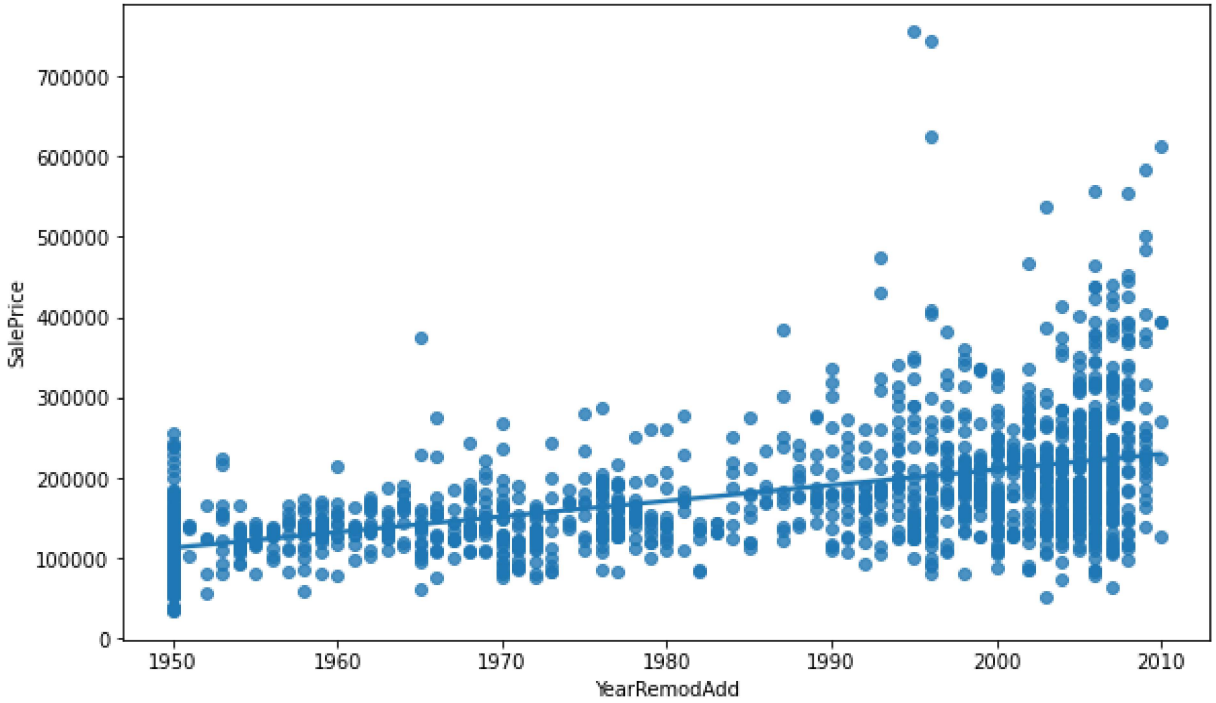


In [47]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(data['TotalBsmtSF'], data['SalePrice'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.612344890924304  with a P-value of P =
4.633323060317264e-150
```

In [48]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="YearRemodAdd", y="SalePrice", data=data)
```

Out[48]: <AxesSubplot:xlabel='YearRemodAdd', ylabel='SalePrice'>

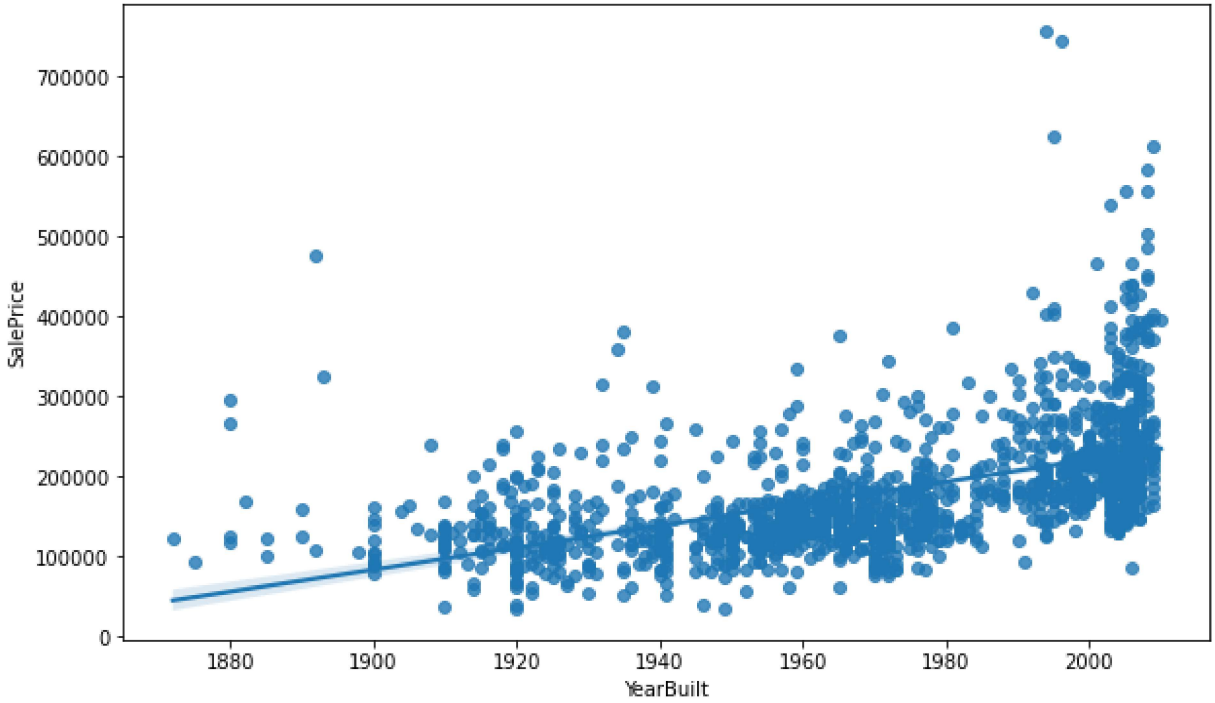


In [49]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(data['YearRemodAdd'], data['SalePrice'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.5052358636155805  with a P-value of P
= 7.637899965293881e-95
```

In [50]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="YearBuilt", y="SalePrice", data=data)
```

Out[50]: <AxesSubplot:xlabel='YearBuilt', ylabel='SalePrice'>

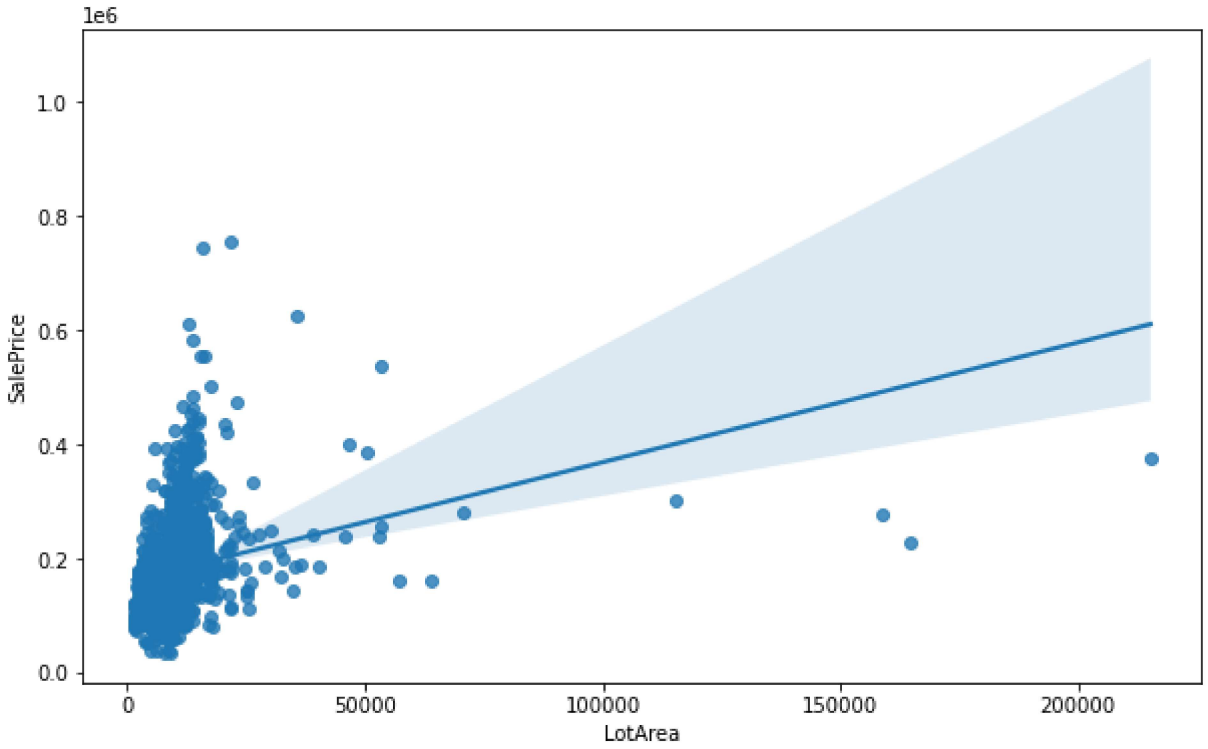


In [51]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(data['YearBuilt'], data['SalePrice'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.5216973113065799  with a P-value of P
= 4.41208869870221e-102
```

In [52]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="LotArea", y="SalePrice", data=data)
```

Out[52]: <AxesSubplot:xlabel='LotArea', ylabel='SalePrice'>

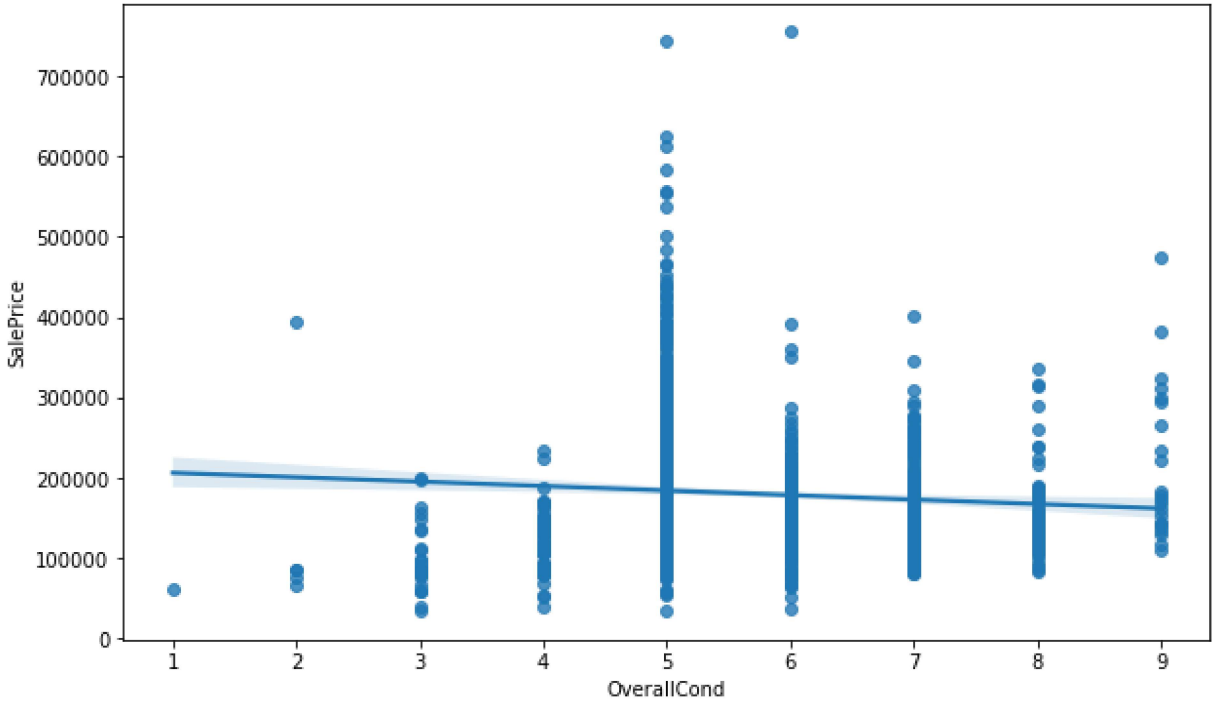


In [53]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(data['LotArea'], data['SalePrice'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.2642200501536714  with a P-value of P
= 1.3331559408474299e-24
```

In [54]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="OverallCond", y="SalePrice", data=data)
```

Out[54]: <AxesSubplot:xlabel='OverallCond', ylabel='SalePrice'>

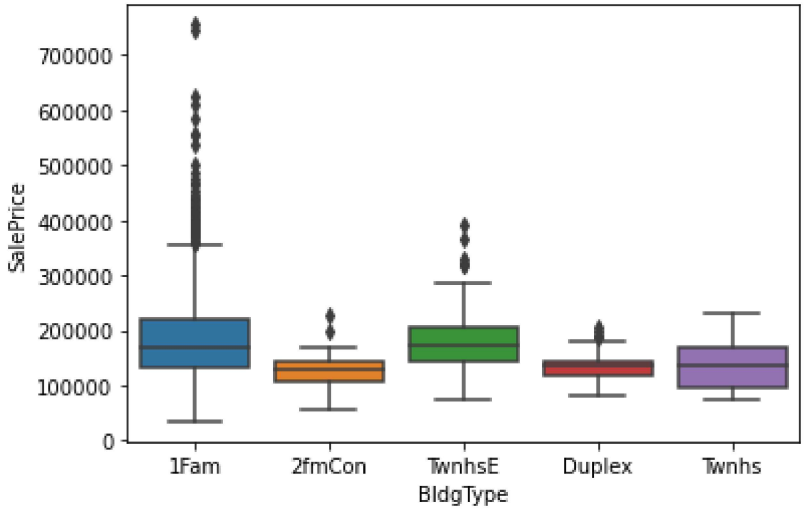


In [55]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(data['OverallCond'], data['SalePrice'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is -0.07800508309965179  with a P-value of
P = 0.0029457820372020068
```

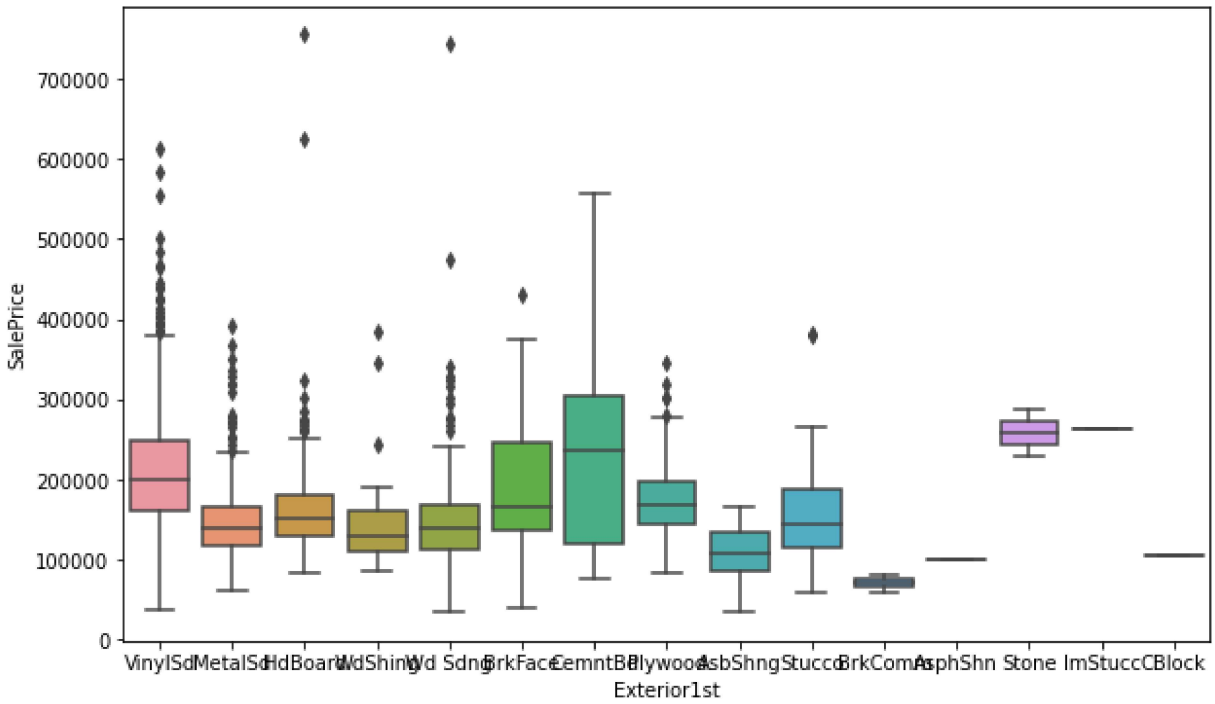
In [56]:
```
sns.boxplot(x="BldgType", y="SalePrice", data=data)
```

Out[56]: <AxesSubplot:xlabel='BldgType', ylabel='SalePrice'>



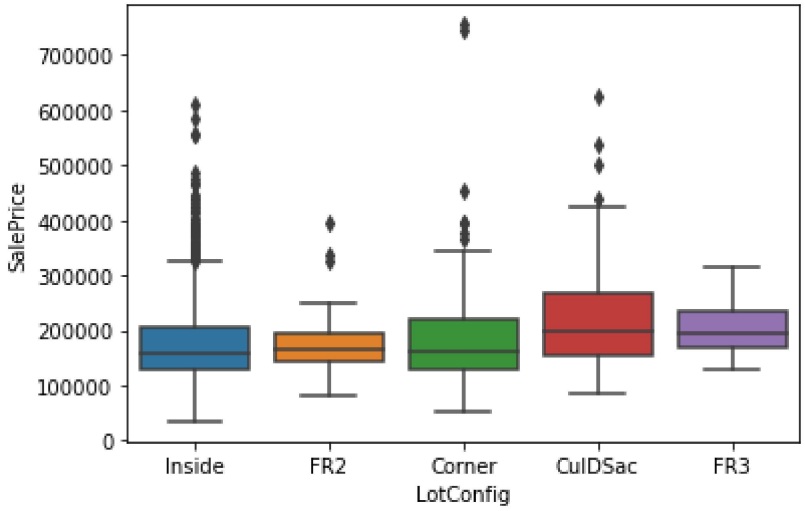
In [57]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="Exterior1st", y="SalePrice", data=data)
```

Out[57]: <AxesSubplot:xlabel='Exterior1st', ylabel='SalePrice'>



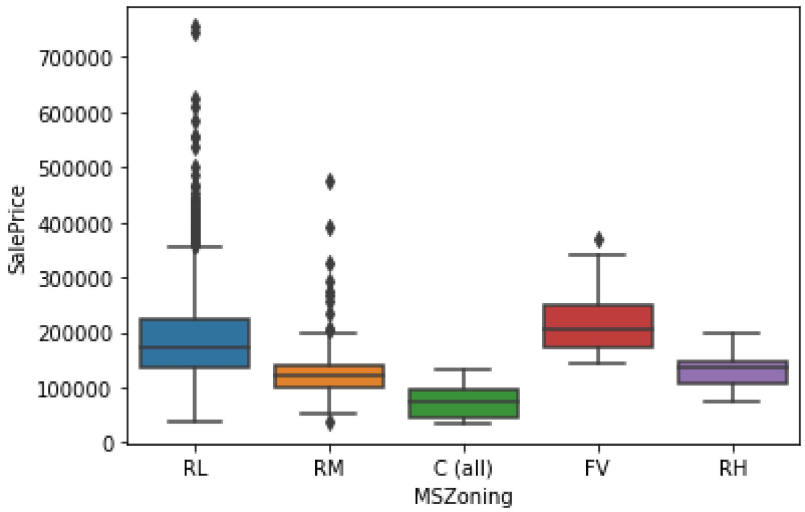
In [58]:
```
sns.boxplot(x="LotConfig", y="SalePrice", data=data)
```

Out[58]: <AxesSubplot:xlabel='LotConfig', ylabel='SalePrice'>



In [59]:
```
sns.boxplot(x="MSZoning", y="SalePrice", data=data)
```

Out[59]: <AxesSubplot:xlabel='MSZoning', ylabel='SalePrice'>



In [60]:
```
data.drop(['LotArea', 'BldgType',
'TotalBsmtSF','BsmtFinSF2'], axis = 1, inplace = True)
```

In [61]:
```
data.shape
```

Out[61]:
```
(1451, 9)
```

In [62]:
```
data.describe()
```

Out[62]:

	Id	MSSubClass	OverallCond	YearBuilt	YearRemodAdd	SalePrice
count	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000	1451.000000
mean	733.944176	57.022743	5.578222	1971.287388	1984.917988	180951.926947
std	419.108340	42.354320	1.115370	30.201292	20.634575	79510.569664
min	0.000000	20.000000	1.000000	1872.000000	1950.000000	34900.000000
25%	371.500000	20.000000	5.000000	1954.000000	1967.000000	130000.000000
50%	734.000000	50.000000	5.000000	1973.000000	1994.000000	163000.000000
75%	1096.500000	70.000000	6.000000	2000.000000	2004.000000	214000.000000
max	1459.000000	190.000000	9.000000	2010.000000	2010.000000	755000.000000

In [63]:
```
data.describe(include=['object'])
```

Out[63]:

	MSZoning	LotConfig	Exterior1st
count	1451	1451	1451
unique	5	5	15
top	RL	Inside	VinylSd
freq	1143	1046	512

In [65]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
data.LotConfig = labelencoder.fit_transform(data.LotConfig)
data.MSZoning = labelencoder.fit_transform(data.MSZoning)
data.Exterior1st = labelencoder.fit_transform(data.Exterior1st)
data.BldgType = labelencoder.fit_transform(data.YearRemodAdd)
```

In [66]:
```
data.head(10)
```

Out[66]:

	Id	MSSubClass	MSZoning	LotConfig	OverallCond	YearBuilt	YearRemodAdd	Exterior1st	Sa
0	0	60	3	4	5	2003	2003	12	2
1	1	20	3	2	8	1976	1976	8	1
2	2	60	3	4	5	2001	2002	12	2
4	4	60	3	2	5	2000	2000	12	2
5	5	50	3	4	5	1993	1995	12	1
7	7	60	3	0	6	1973	1973	6	2
9	9	190	3	0	6	1939	1950	8	1
11	11	60	3	4	5	2005	2006	14	3
13	13	20	3	4	5	2006	2007	12	2
15	15	45	4	0	8	1929	2001	13	1

In [67]:
```
import scipy.stats as stats
data = stats.zscore(data)
data = stats.zscore(data)
```

In [68]:
```
data
```

Out[68]:

	Id	MSSubClass	MSZoning	LotConfig	OverallCond	YearBuilt	YearRemodAdd	Exter
0	-1.751808	0.070318	-0.044622	0.604070	-0.518592	1.050404	0.876599	0.7
1	-1.749421	-0.874421	-0.044622	-0.630444	2.172027	0.156094	-0.432336	-0.5
2	-1.747034	0.070318	-0.044622	0.604070	-0.518592	0.984158	0.828120	0.7
4	-1.742260	0.070318	-0.044622	-0.630444	-0.518592	0.951036	0.731162	0.7
5	-1.739873	-0.165867	-0.044622	0.604070	-0.518592	0.719178	0.488766	0.7
...	...	...	...	...	...	...	...	
1455	1.721045	0.070318	-0.044622	0.604070	-0.518592	0.917913	0.731162	0.7
1456	1.723432	-0.874421	-0.044622	0.604070	0.378281	0.222339	0.149413	-0.1
1457	1.725819	0.306503	-0.044622	0.604070	3.068900	-1.003196	1.022036	-1.4
1458	1.728206	-0.874421	-0.044622	0.604070	0.378281	-0.705093	0.537246	-0.5
1459	1.730593	-0.874421	-0.044622	0.604070	0.378281	-0.208255	-0.965605	-1.1

1451 rows × 9 columns

In [69]:
```
x_train=data.iloc[:,0:6]
y_train=data.iloc[:,7]
x_test=data.iloc[:,0:6]
y_test=data.iloc[:,7]
```

In [70]:
```
x_train
```

Out[70]:

	Id	MSSubClass	MSZoning	LotConfig	OverallCond	YearBuilt
0	-1.751808	0.070318	-0.044622	0.604070	-0.518592	1.050404
1	-1.749421	-0.874421	-0.044622	-0.630444	2.172027	0.156094
2	-1.747034	0.070318	-0.044622	0.604070	-0.518592	0.984158
4	-1.742260	0.070318	-0.044622	-0.630444	-0.518592	0.951036
5	-1.739873	-0.165867	-0.044622	0.604070	-0.518592	0.719178
...	...	...	...	...	...	...
1455	1.721045	0.070318	-0.044622	0.604070	-0.518592	0.917913
1456	1.723432	-0.874421	-0.044622	0.604070	0.378281	0.222339
1457	1.725819	0.306503	-0.044622	0.604070	3.068900	-1.003196
1458	1.728206	-0.874421	-0.044622	0.604070	0.378281	-0.705093
1459	1.730593	-0.874421	-0.044622	0.604070	0.378281	-0.208255

1451 rows × 6 columns

In [71]:
```
rg = LinearRegression()
mdl=rg.fit(x_train,y_train)
```

In [72]:
```
y_pred1 = rg.predict(x_test)
```

In [73]:
```
print('The R-square for Multiple Linear regression is: ',
rg.score(x_train,y_train))
```

```
The R-square for Multiple Linear regression is:  0.02110709029983371
```

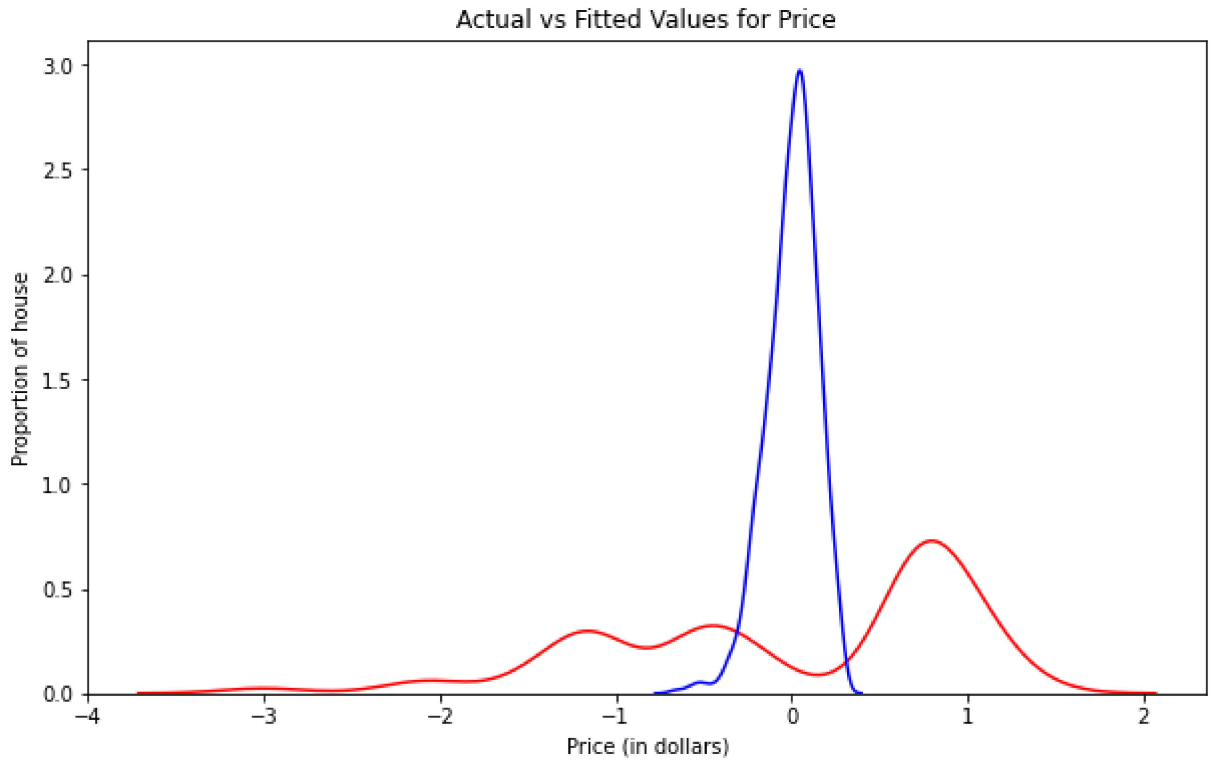
In [74]:
```
mse1 = mean_squared_error(y_test, y_pred1)
print('The mean square error for Multiple Linear Regression: ', mse1)
```

```
The mean square error for Multiple Linear Regression:  0.9788929097001663
```

In [75]:
```
mae1= mean_absolute_error(y_test, y_pred1)
print('The mean absolute error for Multiple Linear Regression: ', mae1)
```

```
The mean absolute error for Multiple Linear Regression:  0.8569591371252484
```

In [76]:
```
plt.figure(figsize=(10,6))
ax1 = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(y_pred1, hist=False, color="b", label="Fitted Values" , ax=ax1)
plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion of house')
plt.show()
plt.close()
```



In [77]:
```
rf = RandomForestRegressor()
model=rf.fit(x_train,y_train)
```

In [78]:
```
y_pred2 = rf.predict(x_test)
```

In [79]:
```
print('The R-square for Random Forest is: ', rf.score(x_train,y_train))
```

```
The R-square for Random Forest is:  0.8740550168660304
```

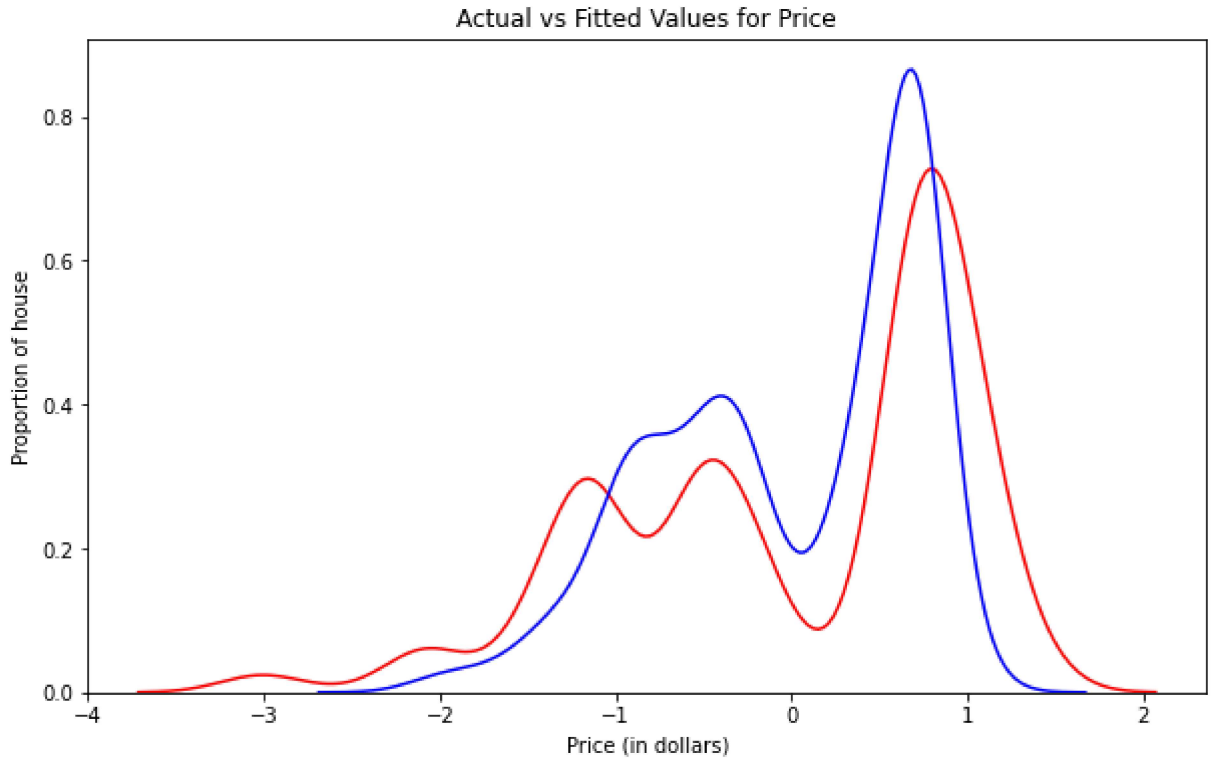
In [80]:
```
mse2 = mean_squared_error(y_test, y_pred2)
print('The mean square error of price and predicted value is: ', mse2)
```

```
The mean square error of price and predicted value is:  0.12594498313396957
```

In [81]:
```
mae2= mean_absolute_error(y_test, y_pred2)
print('The mean absolute error of price and predicted value is: ', mae2)
```

```
The mean absolute error of price and predicted value is:  0.2549282220625497
```

In [82]:
```
plt.figure(figsize=(10,6))
ax1 = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(y_pred2, hist=False, color="b", label="Fitted Values" , ax=ax1)
plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion of house')
plt.show()
plt.close()
```



In [83]:
```
LassoModel=Lasso()
lm=LassoModel.fit(x_train,y_train)
```

In [84]:
```
y_pred3 = lm.predict(x_test)
```

In [85]:
```
print('The R-square for LASSO is: ', lm.score(x_train,y_train))
```

```
The R-square for LASSO is:  0.0
```

In [86]:
```
mae3= mean_absolute_error(y_test, y_pred3)
print('The mean absolute error of price and predicted value is: ', mae3)
```

```
The mean absolute error of price and predicted value is:  0.8858233347881364
```

In [87]:
```
mse3 = mean_squared_error(y_test, y_pred3)
print('The mean square error of price and predicted value is: ', mse3)
```

```
The mean square error of price and predicted value is:  1.0
```

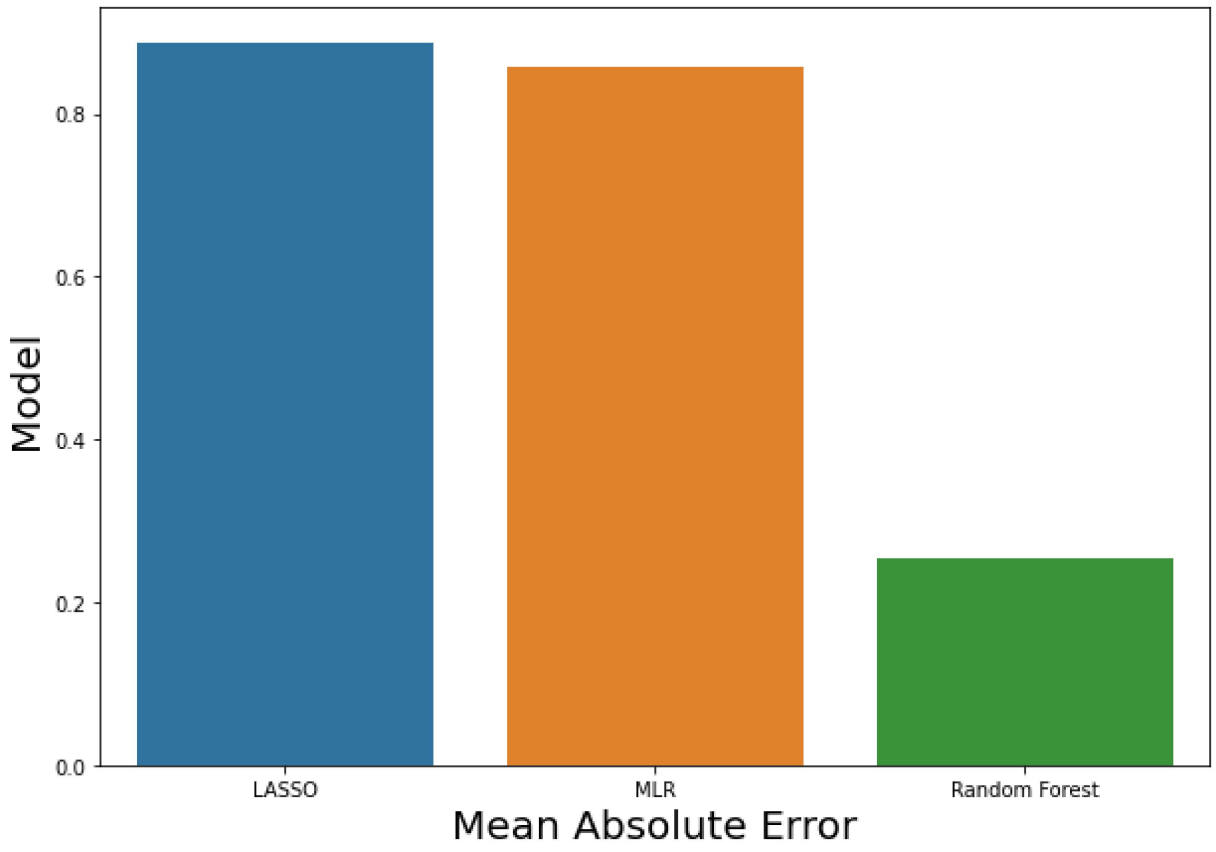
In [88]:
```
scores = [('MLR', mae1),
 ('Random Forest', mae2),
 ('LASSO', mae3)
 ]
```

In [89]:
```
mae = pd.DataFrame(data = scores, columns=['Model', 'MAE Score'])
mae
```

Out[89]:

	Model	MAE Score
0	MLR	0.856959
1	Random Forest	0.254928
2	LASSO	0.885823

In [91]:
```
mae.sort_values(by=(['MAE Score']), ascending=False, inplace=True)
f, axe = plt.subplots(1,1, figsize=(10,7))
sns.barplot(x = mae['Model'], y=mae['MAE Score'], ax = axe)
axe.set_xlabel('Mean Absolute Error', size=20)
axe.set_ylabel('Model', size=20)
plt.show()
```



In [ ]: