In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
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 [2]:
```
dataDiamonds=pd.read_csv(r'C:\Users\18F18013\Desktop\dataDiamonds.csv')
dataDiamonds.head()
```

Out[2]:

	number	carat	cut	color	clarity	depth	table	x	y	z	price
0	1	0.23	Ideal	E	SI2	61.5	55.0	3.95	3.98	2.43	326.0
1	2	0.21	Premium	E	SI1	59.8	61.0	3.89	3.84	2.31	326.0
2	3	0.23	Good	E	VS1	56.9	65.0	4.05	4.07	2.31	327.0
3	4	0.29	Premium	I	VS2	62.4	58.0	4.20	4.23	2.63	334.0
4	5	0.31	Good	J	SI2	63.3	58.0	4.34	4.35	2.75	335.0

In [3]:
```
dataDiamonds.shape
```

Out[3]:
```
(53940, 11)
```

In [4]:
```
dataDiamonds.isnull().sum()
```

Out[4]:
```
number     0
carat      1
cut        1
color      0
clarity    0
depth      1
table      0
x          1
y          1
z          0
price      1
dtype: int64
```

In [5]:
```
dataDiamonds=dataDiamonds.dropna()
```

In [6]:
```
dataDiamonds.isnull().sum()
```

Out[6]:
```
number     0
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
x          0
y          0
z          0
price      0
dtype: int64
```

In [7]:
```
dataDiamonds.shape
```

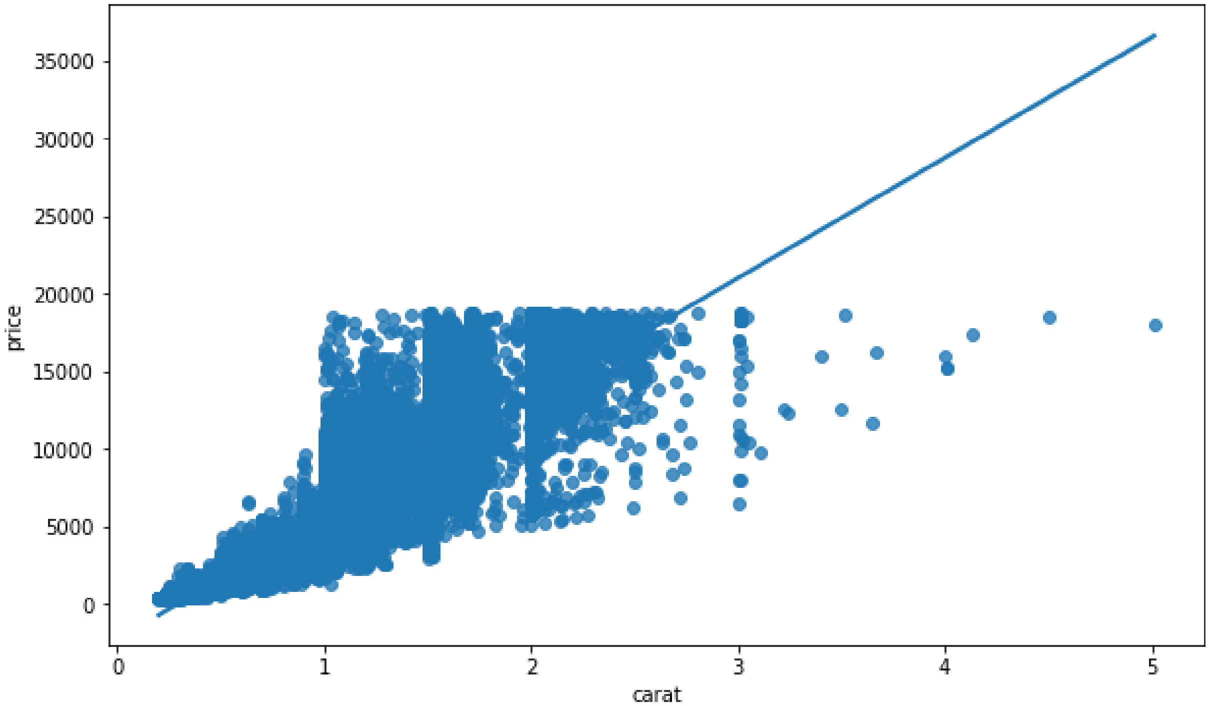
Out[7]:
```
(53934, 11)
```

In [8]:
```
dataDiamonds.dtypes
```

Out[8]:
```
number       int64
carat      float64
cut         object
color       object
clarity     object
depth      float64
table      float64
x          float64
y          float64
z          float64
price      float64
dtype: object
```

In [9]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="carat", y="price", data=dataDiamonds)
```

Out[9]: <AxesSubplot:xlabel='carat', ylabel='price'>

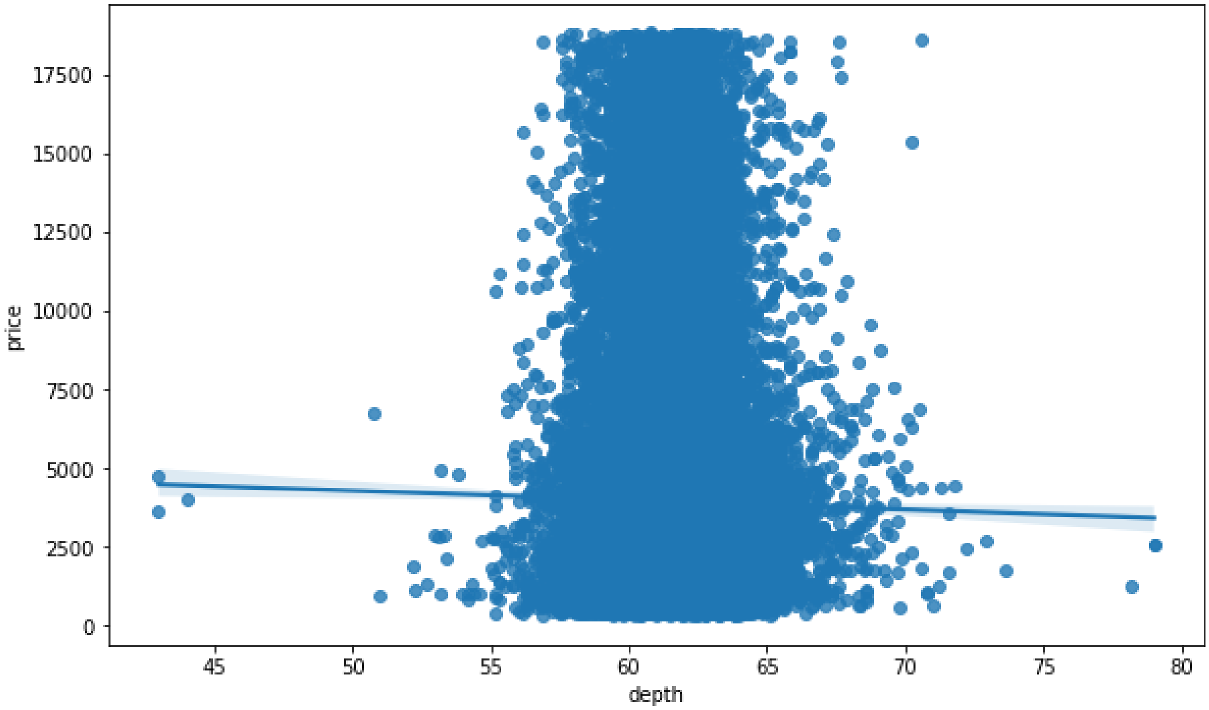


In [12]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dataDiamonds['carat'],
dataDiamonds['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.9215841496289539  with a P-value of P
= 0.0
```

In [13]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="depth", y="price", data=dataDiamonds)
```

Out[13]: <AxesSubplot:xlabel='depth', ylabel='price'>

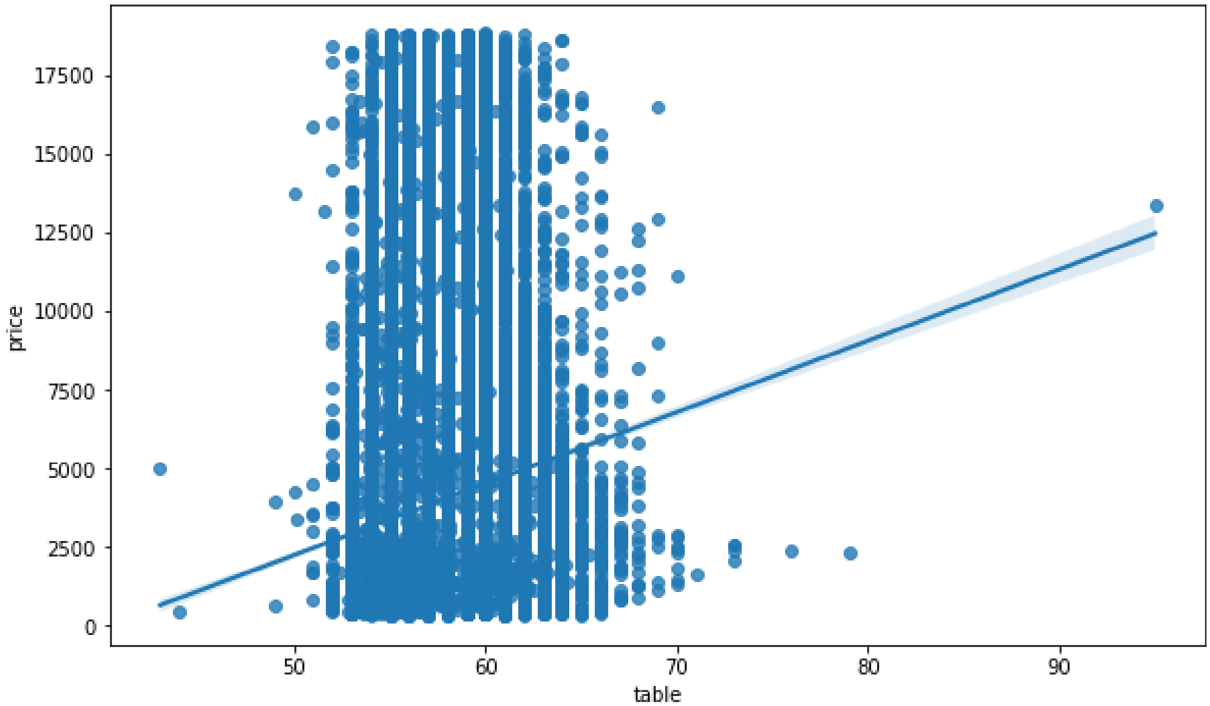


In [14]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dataDiamonds['depth'],
dataDiamonds['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is -0.010693095749490195  with a P-value of
P = 0.013015483356338804
```

In [15]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="table", y="price", data=dataDiamonds)
```

Out[15]: <AxesSubplot:xlabel='table', ylabel='price'>

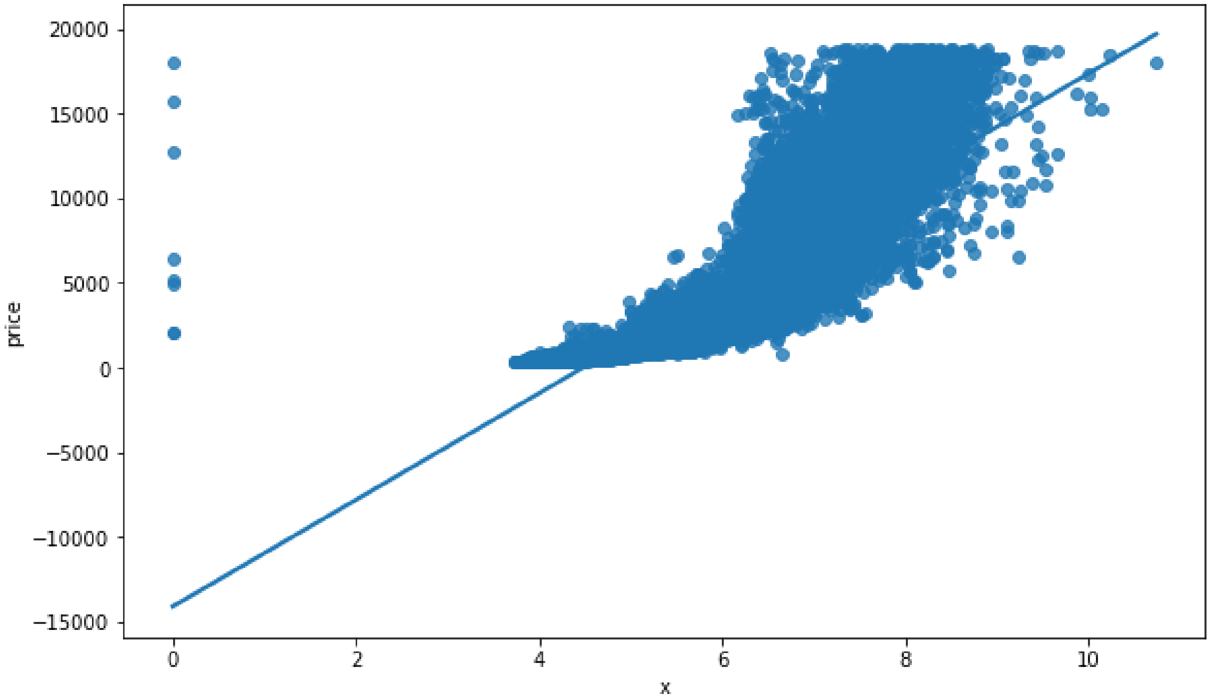


In [16]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dataDiamonds['table'],
dataDiamonds['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.12718726509573478  with a P-value of P
= 2.722006886258901e-193
```

In [17]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="x", y="price", data=dataDiamonds)
```

Out[17]: `<AxesSubplot:xlabel='x', ylabel='price'>`

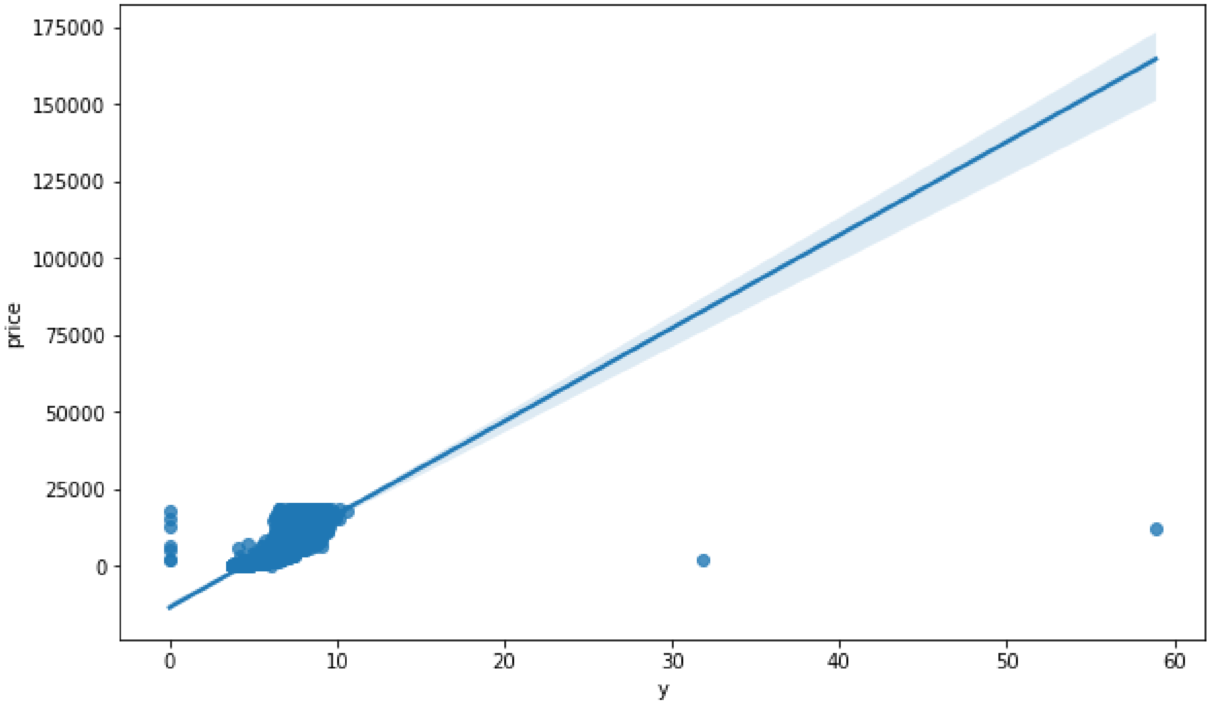


In [18]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dataDiamonds['x'],
dataDiamonds['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.884427005776203  with a P-value of P =
0.0
```

In [19]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="y", y="price", data=dataDiamonds)
```

Out[19]: <AxesSubplot:xlabel='y', ylabel='price'>

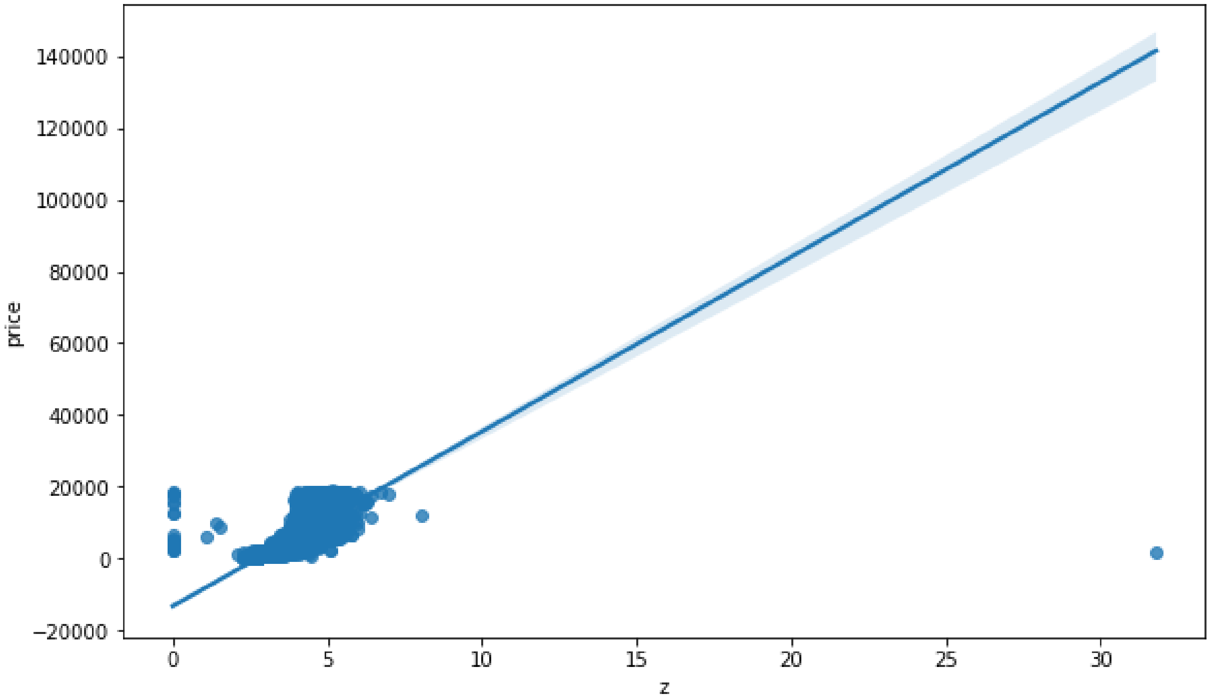


In [20]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dataDiamonds['y'],
dataDiamonds['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.8654091969013185  with a P-value of P
= 0.0
```

In [21]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="z", y="price", data=dataDiamonds)
```

Out[21]: <AxesSubplot:xlabel='z', ylabel='price'>

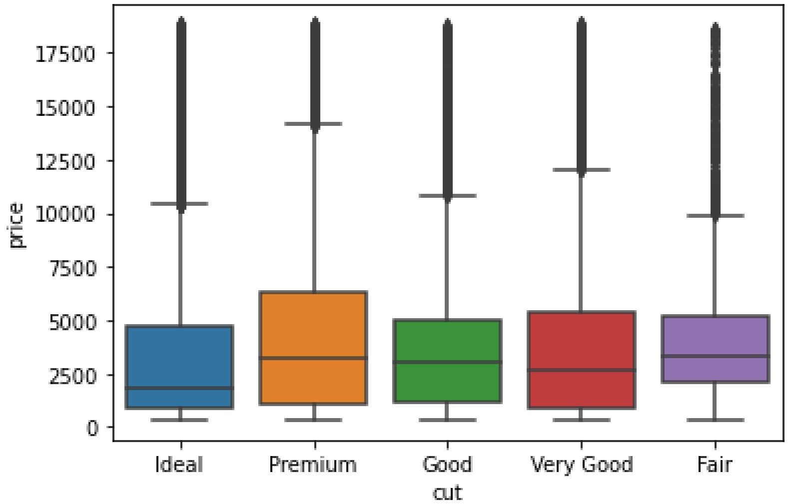


In [22]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(dataDiamonds['z'],
dataDiamonds['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of
```

```
The Pearson Correlation Coefficient is 0.8612381284689564  with a P-value of P
= 0.0
```

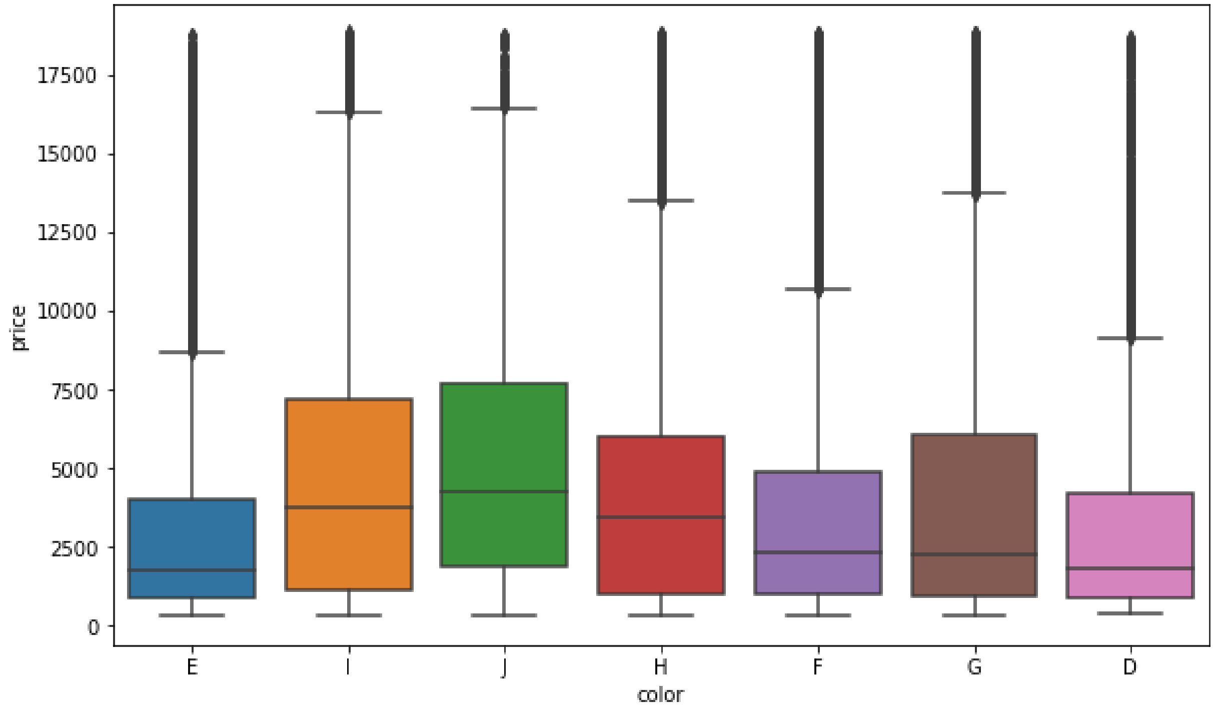
In [26]:
```
sns.boxplot(x="cut", y="price", data=dataDiamonds)
```

Out[26]: <AxesSubplot:xlabel='cut', ylabel='price'>



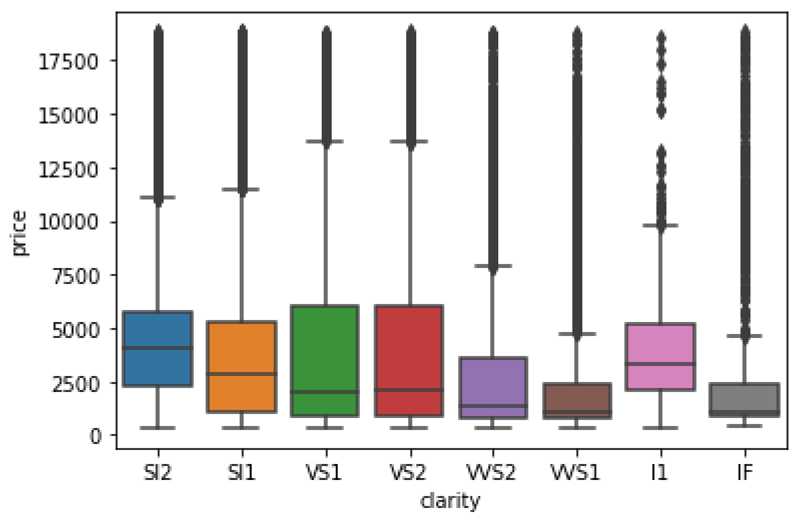
In [27]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="color", y="price", data=dataDiamonds)
```

Out[27]:
```
<AxesSubplot:xlabel='color', ylabel='price'>
```



In [28]:
```
sns.boxplot(x="clarity", y="price", data=dataDiamonds)
```

Out[28]:
```
<AxesSubplot:xlabel='clarity', ylabel='price'>
```



In [29]:
```
dataDiamonds.shape
```

Out[29]:
```
(53934, 11)
```

In [30]:
```
dataDiamonds.describe()
```

Out[30]:

	number	carat	depth	table	x	y	
count	53934.000000	53934.000000	53934.000000	53934.000000	53934.000000	53934.000000	53934
mean	26973.498220	0.797994	61.749475	57.457105	5.731313	5.734677	3
std	15569.552129	0.474009	1.432550	2.234333	1.121726	1.142108	0
min	1.000000	0.200000	43.000000	43.000000	0.000000	0.000000	0
25%	13490.250000	0.400000	61.000000	56.000000	4.710000	4.720000	2
50%	26973.500000	0.700000	61.800000	57.000000	5.700000	5.710000	3
75%	40456.750000	1.040000	62.500000	59.000000	6.540000	6.540000	4
max	53940.000000	5.010000	79.000000	95.000000	10.740000	58.900000	31

In [31]:
```
dataDiamonds.describe(include=['object'])
```

Out[31]:

	cut	color	clarity
count	53934	53934	53934
unique	5	7	8
top	Ideal	G	SI1
freq	21550	11292	13061

In [32]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
dataDiamonds.clarity=labelencoder.fit_transform(dataDiamonds.clarity)
dataDiamonds.color = labelencoder.fit_transform(dataDiamonds.color)
dataDiamonds.cut = labelencoder.fit_transform(dataDiamonds.cut)
```

In [33]:
```
dataDiamonds.head(10)
```

Out[33]:

	number	carat	cut	color	clarity	depth	table	x	y	z	price
0	1	0.23	2	1	3	61.5	55.0	3.95	3.98	2.43	326.0
1	2	0.21	3	1	2	59.8	61.0	3.89	3.84	2.31	326.0
2	3	0.23	1	1	4	56.9	65.0	4.05	4.07	2.31	327.0
3	4	0.29	3	5	5	62.4	58.0	4.20	4.23	2.63	334.0
4	5	0.31	1	6	3	63.3	58.0	4.34	4.35	2.75	335.0
5	6	0.24	4	6	7	62.8	57.0	3.94	3.96	2.48	336.0
6	7	0.24	4	5	6	62.3	57.0	3.95	3.98	2.47	336.0
7	8	0.26	4	4	2	61.9	55.0	4.07	4.11	2.53	337.0
8	9	0.22	0	1	5	65.1	61.0	3.87	3.78	2.49	337.0
9	10	0.23	4	4	4	59.4	61.0	4.00	4.05	2.39	338.0

In [34]:
```
import scipy.stats as stats
dataDiamonds = stats.zscore(dataDiamonds)
dataDiamonds = stats.zscore(dataDiamonds)
```

In [35]:
```
dataDiamonds
```

Out[35]:

	number	carat	cut	color	clarity	depth	table	x	
0	-1.732404	-1.198287	-0.538064	-0.937124	-0.484410	-0.174149	-1.099714	-1.588025	-1.536
1	-1.732339	-1.240481	0.435009	-0.937124	-1.064283	-1.360856	1.585676	-1.641515	-1.658
2	-1.732275	-1.198287	-1.511137	-0.937124	0.095463	-3.385237	3.375935	-1.498876	-1.457
3	-1.732211	-1.071706	0.435009	1.414636	0.675336	0.454107	0.242981	-1.365153	-1.317
4	-1.732147	-1.029513	-1.511137	2.002576	-0.484410	1.082363	0.242981	-1.240344	-1.212
...	...	...	...	...	...	...	...	...	
53935	1.731762	-0.164543	-0.538064	-1.525064	-1.064283	-0.662793	-0.204584	0.016660	0.022
53936	1.731826	-0.164543	-1.511137	-1.525064	-1.064283	0.942751	-1.099714	-0.036830	0.013
53937	1.731890	-0.206737	1.408081	-1.525064	-1.064283	0.733332	1.138111	-0.063575	-0.047
53938	1.731954	0.130812	0.435009	0.826696	-0.484410	-0.523181	0.242981	0.373256	0.337
53939	1.732018	-0.101253	-0.538064	-1.525064	-0.484410	0.314494	-1.099714	0.087979	0.118

53934 rows × 11 columns

In [37]:
```
x_train=dataDiamonds.iloc[:,0:5]
y_train=dataDiamonds.iloc[:,6]
x_test=dataDiamonds.iloc[:,0:5]
y_test=dataDiamonds.iloc[:,6]
```

In [38]:
```
x_train
```

Out[38]:

	number	carat	cut	color	clarity
0	-1.732404	-1.198287	-0.538064	-0.937124	-0.484410
1	-1.732339	-1.240481	0.435009	-0.937124	-1.064283
2	-1.732275	-1.198287	-1.511137	-0.937124	0.095463
3	-1.732211	-1.071706	0.435009	1.414636	0.675336
4	-1.732147	-1.029513	-1.511137	2.002576	-0.484410
...	...	...	...	...	...
53935	1.731762	-0.164543	-0.538064	-1.525064	-1.064283
53936	1.731826	-0.164543	-1.511137	-1.525064	-1.064283
53937	1.731890	-0.206737	1.408081	-1.525064	-1.064283
53938	1.731954	0.130812	0.435009	0.826696	-0.484410
53939	1.732018	-0.101253	-0.538064	-1.525064	-0.484410

53934 rows × 5 columns

In [39]:
```
rg = LinearRegression()
mdl=rg.fit(x_train,y_train)
```

In [40]:
```
y_pred1 = rg.predict(x_test)
```

In [42]:
```
print('The R-square for Multiple Linear regression is: ',
rg.score(x_train,y_train))
```

```
The R-square for Multiple Linear regression is:  0.05915398362024349
```

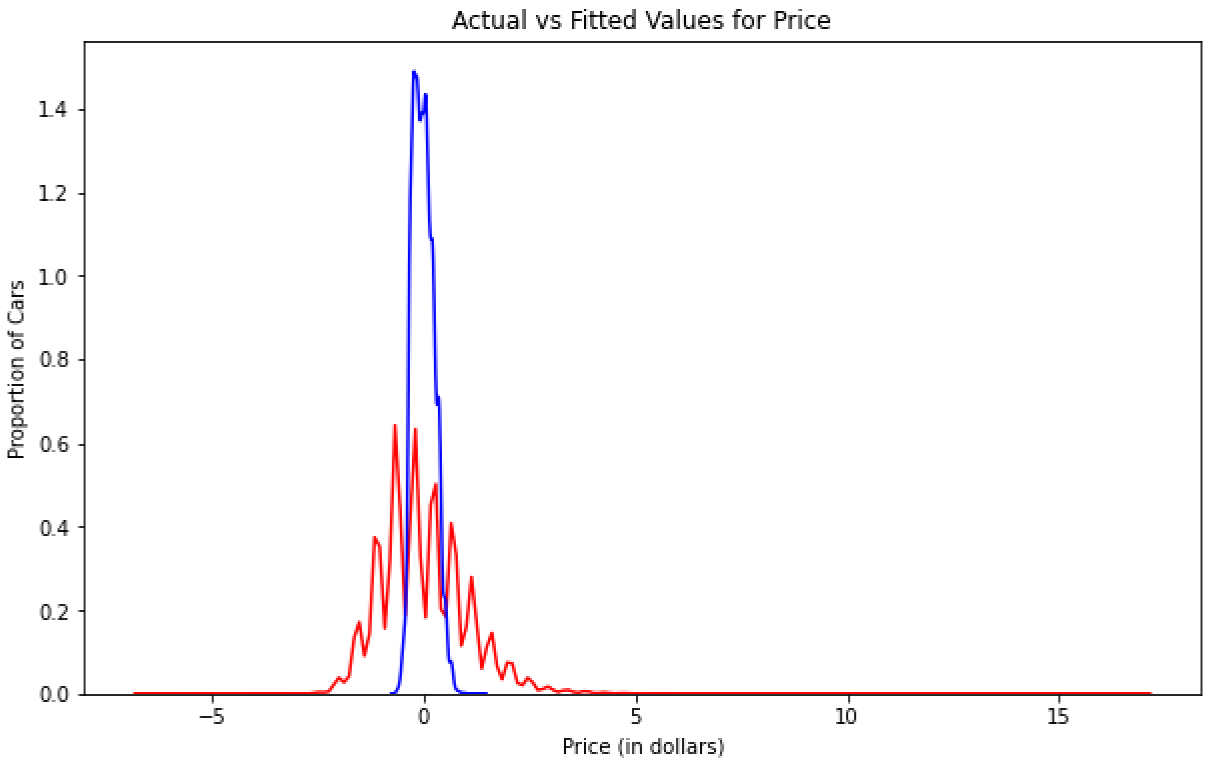
In [43]:
```
mse1 = mean_squared_error(y_test, y_pred1)
print('The mean square error for Multiple Linear Regression: ', mse1)
```

```
The mean square error for Multiple Linear Regression:  0.9408460163797565
```

In [44]:
```
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:  0.73592452542974
```

In [45]:
```
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 Cars')
plt.show()
plt.close()
```



In [46]:
```
rf = RandomForestRegressor()
model=rf.fit(x_train,y_train)
```

In [47]:
```
y_pred2 = rf.predict(x_test)
```

In [48]:
```
print('The R-square for Random Forest is: ', rf.score(x_train,y_train))
```

```
The R-square for Random Forest is:  0.8989164640256592
```

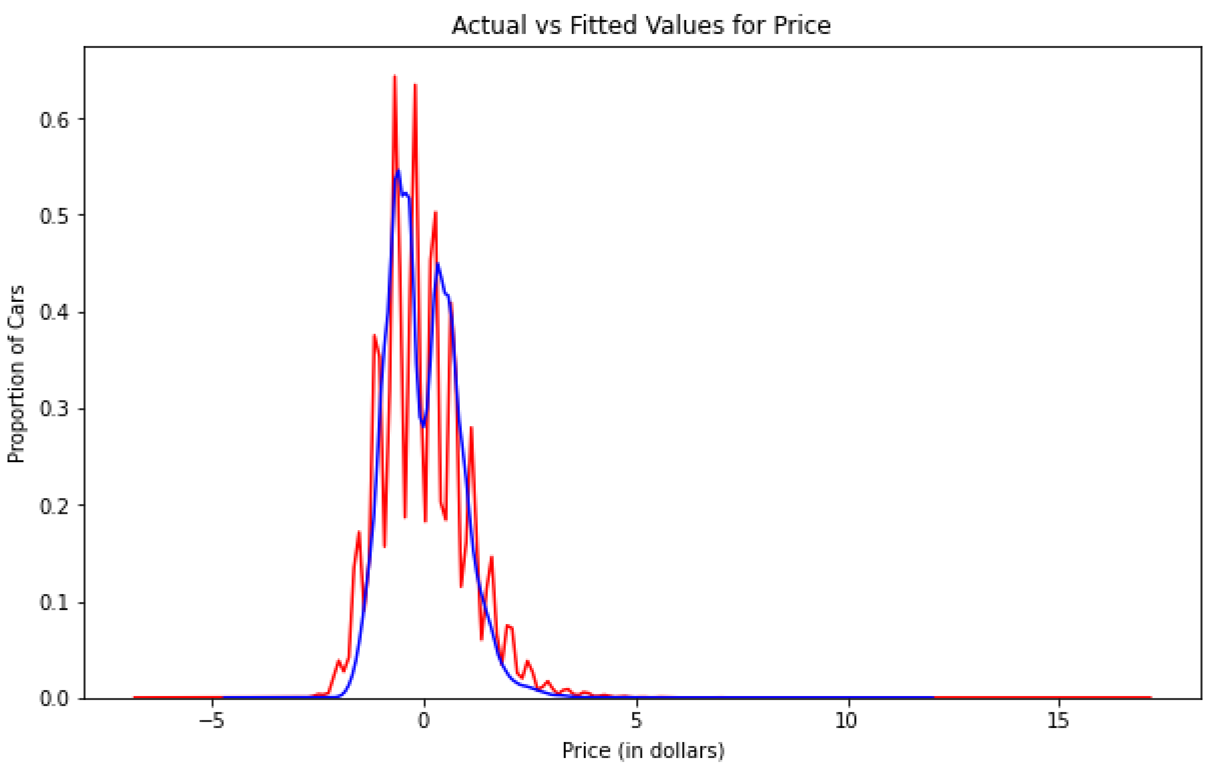
In [49]:
```
mse2 = mean_squared_error(y_test, y_pred2)
print('The mean square error of price and predicted value is: ', mse2)
```

```
The mean square error of price and predicted value is:  0.10108353597434079
```

In [50]:
```
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:  0.23087997032390323
```

In [51]:
```
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 Cars')
plt.show()
plt.close()
```



In [52]:
```
LassoModel=Lasso()
lm=LassoModel.fit(x_train,y_train)
```

In [53]:
```
y_pred3 = lm.predict(x_test)
```

In [54]:
```
print('The R-square for LASSO is: ', lm.score(x_train,y_train))
```

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

In [55]:
```
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:  0.7926247334183725
```

In [56]:
```
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
```

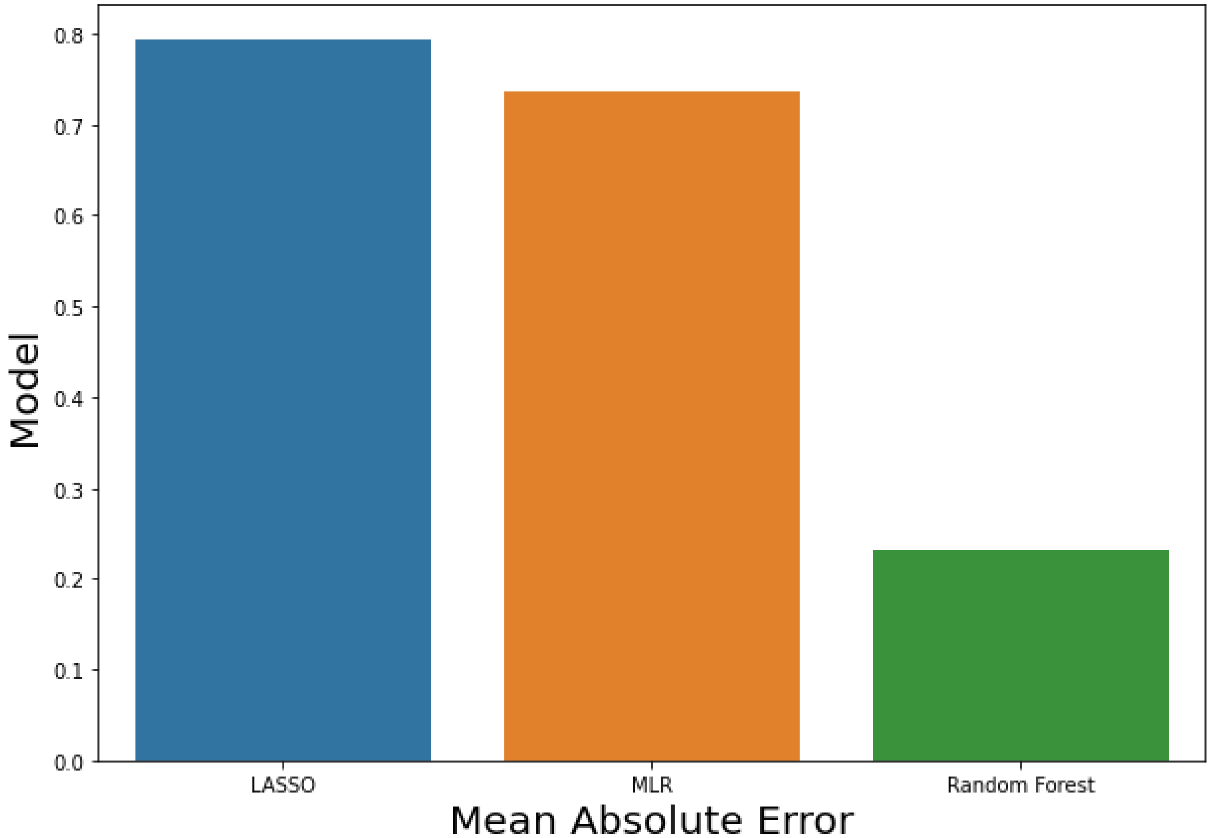
In [57]:
```
scores = [('MLR', mae1),
 ('Random Forest', mae2),
 ('LASSO', mae3)
 ]
```

In [58]:
```
mae = pd.DataFrame(data = scores, columns=['Model', 'MAE Score'])
mae
```

Out[58]:

	Model	MAE Score
0	MLR	0.735925
1	Random Forest	0.230880
2	LASSO	0.792625

In [59]:
```
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 [ ]: