Problem 4.

Refer to Real estate sales data set in Appendix C.7. Residential sales that occurred during the year 2002 were available from a city in the midwest. Data on 522 arms-length transactions include sales price, style, finished square feet, number of bedrooms, pool, lot size, year built, air conditioning, and whether or not the lot is adjacent to a highway. The city tax assessor was interested in predicting sales price based on the demographic variable information given above.

a) Select a random sample of 300 observations to use in the model-building data set. Develop a best subset model for predicting sales price. Justify your choice of model. Assess your model's ability to predict and discuss its use as a tool for predicting sales price.

b) Fit the regression model identified above to the validation data set. Compare the estimated regression coefficients and their estimated standard errors with those obtained in a). Also compare the error mean square and coefficients of multiple determination. Does the model fitted to the validation data set yield similar estimates as the model fitted to the model-building data set?

c) Calculate the mean squared prediction error (9.20) and compare it to MSE obtained from the model-building data set. Is there evidence of a substantial bias problem in MSE here?

In [1]:
```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

In [2]:
```
df = pd.read_csv('APPENC07.txt', sep = '\s+', header =None, names=['Y','X2','X3','N.1'
df.head()
```

Out[2]:

	Y	X2	X3	N.1	X7	N.2	X4	X6	N.3	X1	X5	X8
1	360000	3032	4	4	1	2	0	1972	2	1	22221	0
2	340000	2058	4	2	1	2	0	1976	2	1	22912	0
3	250000	1780	4	3	1	2	0	1980	2	1	21345	0
4	205500	1638	4	2	1	2	0	1963	2	1	17342	0
5	275500	2196	4	3	1	2	0	1968	2	7	21786	0

In [3]:
```
y = df['Y']
x1 = df['X1']
x2 = df['X2']
x3 = df['X3']
x4= df['X4']
```

```
x5= df['X5']
x6= df['X6']
x7= df['X7']
x8= df['X8']
df1 = pd.DataFrame({'Y':y,'X1':x1,'X2':x2,'X3':x3,
                    'X4':x4,'X5':x5,'X6':x6,'X7':x7,'X8':x8})
df1.head()
```

Out[3]:

	Y	X1	X2	X3	X4	X5	X6	X7	X8
1	360000	1	3032	4	0	22221	1972	1	0
2	340000	1	2058	4	0	22912	1976	1	0
3	250000	1	1780	4	0	21345	1980	1	0
4	205500	1	1638	4	0	17342	1963	1	0
5	275500	7	2196	4	0	21786	1968	1	0

a) Select a random sample of 300 observations to use in the model-building data set. Develop a best subset model for predicting sales price. Justify your choice of model. Assess your model's ability to predict and discuss its use as a tool for predicting sales price.

In [4]:
```
data1=df1.sample(n =300)
data1
```

Out[4]:

	Y	X1	X2	X3	X4	X5	X6	X7	X8
177	259000	1	2556	3	0	80886	1957	1	0
362	242000	5	2514	4	0	17535	1953	1	0
49	178000	7	2038	2	0	47884	1918	0	0
160	437632	5	2936	4	0	22844	1980	1	0
228	274900	7	2472	3	0	22451	1969	1	0
...	...	...	...	...	...	...	...	...	...
431	175000	1	1672	3	0	22617	1949	1	0
73	920000	1	3857	4	0	32793	1997	1	0
124	400000	1	2537	3	0	11053	1993	1	0
293	235000	7	2313	4	0	24705	1972	1	0
102	570000	1	2547	2	0	21789	1996	1	0

300 rows × 9 columns

In [7]:
```
y = data1['Y']
x1 = data1['X1']
x2 = data1['X2']
x3 = data1['X3']
x4= data1['X4']
x5= data1['X5']
x6= data1['X6']
```

```
x7= data1['X7']
x8= data1['X8']
n=len(data1)
import statsmodels.api as sm
import statsmodels.formula.api as smf
model = smf.ols('y ~ x1+x2+x3+x4+x5+x6+x7+x8', data=data1)
results = model.fit()
sse = np.sum((results.fittedvalues - data1.Y)**2)
mse = sse/(n-9)
```

In [8]:

```
anova_result = sm.stats.anova_lm(results, typ=2)
print(anova_result)
```

```
                sum_sq     df           F        PR(>F)
x1        1.938366e+11    1.0   37.056153  3.615607e-09
x2        1.871538e+12    1.0  357.785930  1.359362e-52
x3        4.154982e+10    1.0    7.943168  5.157782e-03
x4        1.753401e+10    1.0    3.352014  6.814532e-02
x5        1.191716e+11    1.0   22.782289  2.876781e-06
x6        3.258508e+11    1.0   62.293596  6.060009e-14
x7        3.066298e+08    1.0    0.058619  8.088626e-01
x8        1.018909e+10    1.0    1.947869  1.638806e-01
Residual  1.522188e+12  291.0         NaN           NaN
```

Regression of Y on X1

In [9]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model1 = smf.ols('y ~ x1', data=data1)
results1 = model1.fit()
sse1 = np.sum((results1.fittedvalues - data1.Y)**2)
ssr1 = np.sum((results1.fittedvalues - data1.Y.mean())**2)
sstoX1 = ssr1+sse1
R2_X1 = ssr1/sstoX1
print('R^2 =',R2_X1)

n=len(y)
p1=2
R2a_X1 = 1 - (sse1/(n-p1))/(sstoX1/(n-1))
print('R^2a =',R2a_X1)

Cp1 = sse1/mse - (n-2*p1)
print('Cp=',Cp1)
aic1 = n * math.log(sse1/n)+ 2*p1
print('AICp=',aic1)
bic1 = n * math.log(sse1/n)+ p1*math.log(n)
print('BICp=',bic1)
```

```
R^2 = 0.08469827523087155
R^2a = 0.08162679293298858
Cp= 798.8302794215367
AICp= 7105.725715435214
BICp= 7113.133280384526
```

Regression of Y on X2

In [10]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

```
model2 = smf.ols('y ~ x2', data=data1)
results2 = model2.fit()
sse2 = np.sum((results2.fittedvalues - data1.Y)**2)
ssr2 = np.sum((results2.fittedvalues - data1.Y.mean())**2)
sstoX2 = ssr2+sse2
R2_X2 = ssr2/sstoX2
print('R^2 =',R2_X2)

n=len(y)
p1=2
R2a_X2 = 1 - (sse2/(n-p1))/(sstoX2/(n-1))
print('R^2a =',R2a_X2)

Cp2 = sse2/mse - (n-2*p1)
print('Cp=',Cp2)

aic2 = n * math.log(sse2/n)+ 2*p1
print('AICp=',aic2)
bic2 = n * math.log(sse2/n)+ p1*math.log(n)
print('BICp=',bic2)
```

```
R^2 = 0.6267146045117246
R^2a = 0.6254619689563947
Cp= 150.502112677098
AICp= 6836.652564736088
BICp= 6844.0601296854
```

Regression of Y on X3

In [11]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model3 = smf.ols('y ~ x3', data=data1)
results3 = model3.fit()
sse3 = np.sum((results3.fittedvalues - data1.Y)**2)
ssr3 = np.sum((results3.fittedvalues - data1.Y.mean())**2)
sstoX3 = ssr3+sse3
R2_X3 = ssr3/sstoX3
print('R^2 =',R2_X3)

n=len(y)
p1=2
R2a_X3 = 1 - (sse3/(n-p1))/(sstoX3/(n-1))
print('R^2a =',R2a_X3)

Cp3 = sse3/mse - (n-2*p1)
print('Cp=',Cp3)

aic3 = n * math.log(sse3/n)+ 2*p1
print('AICp=',aic3)
bic3 = n * math.log(sse3/n)+ p1*math.log(n)
print('BICp=',bic3)
```

```
R^2 = 0.14682337004154908
R^2a = 0.14396036121618516
Cp= 724.5198820191725
AICp= 7084.639564558229
BICp= 7092.047129507541
```

Regression of Y on X4

In [12]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model4 = smf.ols('y ~ x4', data=data1)
results4 = model4.fit()
sse4 = np.sum((results4.fittedvalues - data1.Y)**2)
ssr4 = np.sum((results4.fittedvalues - data1.Y.mean())**2)
sstoX4 = ssr4+sse4
R2_X4 = ssr4/sstoX4
print('R^2 =',R2_X4)

n=len(y)
p1=2
R2a_X4 = 1 - (sse4/(n-p1))/(sstoX4/(n-1))
print('R^2a =',R2a_X4)

Cp4 = sse4/mse - (n-2*p1)
print('Cp=',Cp4)

aic4 = n * math.log(sse4/n)+ 2*p1
print('AICp=',aic4)
bic4 = n * math.log(sse4/n)+ p1*math.log(n)
print('BICp=',bic4)
```

```
R^2 = 0.022425970865090683
R^2a = 0.019145521102893137
Cp= 873.3167602660997
AICp= 7125.471792642476
BICp= 7132.879357591789
```

Regression of Y on X5

In [13]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model5 = smf.ols('y ~ x5', data=data1)
results5 = model5.fit()
sse5 = np.sum((results5.fittedvalues - data1.Y)**2)
ssr5 = np.sum((results5.fittedvalues - data1.Y.mean())**2)
sstoX5 = ssr5+sse5
R2_X5 = ssr5/sstoX5
print('R^2 =',R2_X5)

n=len(y)
p1=2
R2a_X5 = 1 - (sse5/(n-p1))/(sstoX5/(n-1))
print('R^2a =',R2a_X5)

Cp5 = sse5/mse - (n-2*p1)
print('Cp=',Cp5)

aic5 = n * math.log(sse5/n)+ 2*p1
print('AICp=',aic5)
bic5 = n * math.log(sse5/n)+ p1*math.log(n)
print('BICp=',bic5)
```

```
R^2 = 0.043908814742777787
R^2a = 0.040700455060706663
Cp= 847.620241480111
AICp= 7118.805573177114
BICp= 7126.213138126426
```

Regression of Y on X6

In [14]:
```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
model6 = smf.ols('y ~ x6', data=data1)
results6 = model6.fit()
sse6 = np.sum((results6.fittedvalues - data1.Y)**2)
ssr6 = np.sum((results6.fittedvalues - data1.Y.mean())**2)
sstoX6 = ssr6+sse6
R2_X6 = ssr6/sstoX6
print('R^2 =',R2_X6)

n=len(y)
p1=2
R2a_X6 = 1 - (sse6/(n-p1))/(sstoX6/(n-1))
print('R^2a =',R2a_X6)

Cp6 = sse6/mse - (n-2*p1)
print('Cp=',Cp6)

aic6 = n * math.log(sse6/n)+ 2*p1
print('AICp=',aic6)
bic6 = n * math.log(sse6/n)+ p1*math.log(n)
print('BICp=',bic6)
```

```
R^2 = 0.3081472378283676
R^2a = 0.3058255842640333
Cp= 531.5537261966687
AICp= 7021.761534470902
BICp= 7029.1690994202145
```

Regression of Y on X7

In [15]:
```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
model7 = smf.ols('y ~ x7', data=data1)
results7 = model7.fit()
sse7 = np.sum((results7.fittedvalues - data1.Y)**2)
ssr7 = np.sum((results7.fittedvalues - data1.Y.mean())**2)
sstoX7 = ssr7+sse7
R2_X7 = ssr7/sstoX7
print('R^2 =',R2_X7)

n=len(y)
p1=2
R2a_X7 = 1 - (sse7/(n-p1))/(sstoX7/(n-1))
print('R^2a =',R2a_X7)

Cp7 = sse7/mse - (n-2*p1)
print('Cp=',Cp7)

aic7 = n * math.log(sse7/n)+ 2*p1
```

```
print('AICp=',aic7)
bic7 = n * math.log(sse7/n)+ p1*math.log(n)
print('BICp=',bic7)
```

```
R^2 = 0.09314014438001295
R^2a = 0.0900969905020933
Cp= 788.732610304038
AICp= 7102.945963198573
BICp= 7110.353528147885
```

Regression of Y on X8

In [16]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model8 = smf.ols('y ~ x8', data=data1)
results8 = model8.fit()
sse8 = np.sum((results8.fittedvalues - data1.Y)**2)
ssr8 = np.sum((results8.fittedvalues - data1.Y.mean())**2)
sstoX8 = ssr8+sse8
R2_X8 = ssr8/sstoX8
print('R^2 =',R2_X8)

n=len(y)
p1=2
R2a_X8 = 1 - (sse8/(n-p1))/(sstoX8/(n-1))
print('R^2a =',R2a_X8)

Cp8 = sse8/mse - (n-2*p1)
print('Cp=',Cp8)

aic8 = n * math.log(sse8/n)+ 2*p1
print('AICp=',aic8)
bic8 = n * math.log(sse8/n)+ p1*math.log(n)
print('BICp=',bic8)
```

```
R^2 = 0.006037557885916396
R^2a = 0.002702113449291943
Cp= 892.9196193842672
AICp= 7130.459412410572
BICp= 7137.866977359885
```

In [17]:

```
n = len(data1)
p = 7
AIC_full = n*math.log(np.sum(results.resid**2))- n*math.log(n) + 2*p
AIC_full
```

Out[17]:

```
6718.216305304037
```

In [18]:

```
BIC_full = n*math.log(np.sum(results.resid**2)) - n*math.log(n) + math.log(n)*p
BIC_full
```

Out[18]:

```
6744.14278262663
```

In [19]:

```
table = {'R^2_a,p':[R2a_X1,R2a_X2,R2a_X3,R2a_X4,R2a_X5,R2a_X6,R2a_X7,R2a_X8],
        'Cp':[Cp1,Cp2,Cp3,Cp4,Cp5,Cp6,Cp7,Cp8],
        'AICp':[aic1,aic2,aic3,aic4,aic5,aic6,aic7,aic8],
        'BICp':[bic1,bic2,bic3,bic4,bic5,bic6,bic7,bic8]}
t = pd.DataFrame(table)
```

```
t.index = ['X1','X2','X3','X4','X5','X6','X7','X8']
print(t)
```

```
     R^2_a,p          Cp         AICp         BICp
X1  0.081627  798.830279  7105.725715  7113.133280
X2  0.625462  150.502113  6836.652565  6844.060130
X3  0.143960  724.519882  7084.639565  7092.047130
X4  0.019146  873.316760  7125.471793  7132.879358
X5  0.040700  847.620241  7118.805573  7126.213138
X6  0.305826  531.553726  7021.761534  7029.169099
X7  0.090097  788.732610  7102.945963  7110.353528
X8  0.002702  892.919619  7130.459412  7137.866977
```

The model contains Y and X2 is the best subset contains 1 predictor

b) Fit the regression model identified above to the validation data set. Compare the estimated regression coefficients and their estimated standard errors with those obtained in a). Also compare the error mean square and coefficients of multiple determination. Does the model fitted to the validation data set yield similar estimates as the model fitted to the model-building data set?

In [20]:

```
data2=df1.sample(n =222)
y = data2['Y']
x1 = data2['X1']
x2 = data2['X2']
x3 = data2['X3']
x4= data2['X4']
x5= data2['X5']
x6= data2['X6']
x7= data2['X7']
x8= data2['X8']
import statsmodels.api as sm
import statsmodels.formula.api as smf
model1 = smf.ols('y ~ x1+x2+x3+x4+x5+x6', data=data2)
results1 = model1.fit()
results1.summary()
```

Out[20]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.798
Model:	OLS	**Adj. R-squared:**	0.792
Method:	Least Squares	**F-statistic:**	141.6
Date:	Sun, 11 Dec 2022	**Prob (F-statistic):**	7.85e-72
Time:	23:00:19	**Log-Likelihood:**	-2779.4
No. Observations:	222	**AIC:**	5573.
Df Residuals:	215	**BIC:**	5597.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.059e+06	6.78e+05	-5.985	0.000	-5.4e+06	-2.72e+06
x1	-1.3e+04	2294.936	-5.663	0.000	-1.75e+04	-8472.079
x2	184.2558	9.949	18.520	0.000	164.645	203.866
x3	-1.837e+04	5230.271	-3.512	0.001	-2.87e+04	-8060.674
x4	5215.5223	1.7e+04	0.306	0.760	-2.83e+04	3.88e+04
x5	0.9922	0.462	2.146	0.033	0.081	1.904
x6	2035.7612	348.454	5.842	0.000	1348.939	2722.584

Omnibus:	31.162	**Durbin-Watson:**	2.080
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	81.908
Skew:	0.598	**Prob(JB):**	1.64e-18
Kurtosis:	5.725	**Cond. No.**	3.87e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.87e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

The regression model identified above to the validation data set:

Y^ = −4.071e + 06 + (-1.14e+04)X1 + (163.3282)X2 + (-8168.8044)X3 + (-1.025e+04)X4 + (0.4746)X5 + (2100.6208)X6

c) Calculate the mean squared prediction error (9.20) and compare it to MSE obtained from the model-building data set. Is there evidence of a substantial bias problem in MSE here?

In [ ]:

In [ ]: