625.661 - Homework Five

Eric Niblock

March 21, 2022

1. Randomly select only 14 rows of Table B.4 and randomly select 5 regressors from that table. Consider the property valuation data found in Table B.4. Use the all-possible-regressions method to find the "best" set of regressors.

The attached PDF provides results for the problem. The five regressors chosen at random were: x_1, x_2, x_4, x_6, and x_8. Using the all-possible-regressions method, use of only x_1 provides the lowest value of C_p while using x_1 and x_4 still provides a low value for C_p, while also possessing a higher value regarding R^2_{Adj}. Depending on which metric is chosen, one of these two models would represent the best subset of regressors.

2. Randomly select only 14 rows of Table B.11. Table B.11 presents data on the quality of Pinot Noir wine. Build an appropriate regression model for quality y using the all-possible-regressions approach. Use C_p as the model selection criterion, and incorporate the region information by using indicator variables.

The attached PDF provides results for the problem. The region information was encoded into 'iv1' and 'iv2'. Using C_p as the selection criteria, we achieve the best results by using the regressors 'Body' and 'iv1', which achieve the lowest value of C_p.

3. Randomly select only 20 rows of Table B.2. Consider the solar thermal energy test data in Table B.2.

(a) Use forward selection to specify a subset regression model.

The attached PDF provides results for the problem. A cutoff value of $F_{in} = 4.0$ was used. The forward selection was stopped on the third round given insufficient F-values, and regressors x_3 and x_4 were included in the model.

(b) Use backward elimination to specify a subset regression model.

The attached PDF provides results for the problem. A cutoff value of $F_{out} = 4.0$ was used. The backward elimination was stopped on the fourth round given insufficient F-values, and regressors x_3 and x_4 remained in the model.

(c) Use stepwise regression to specify a subset regression model.

The attached PDF provides results for the problem. A cutoff value of $F_{in} = F_{out} = 4.0$ was used. Given that stepwise regression is simply a modification of forward selection, with a reassessment of regressors at each step, we generated a model at each of the forward selection steps, and confirmed that all regressors remained significant within the model, by examining t-values. Again, regressors x_3 and x_4 were included in the model.

(d) Apply all-possible-regressions to the data. Evaluate R_p^2, C_p, and MS_{Res} for each model. Which subset model do you recommend?

The attached PDF provides results for the problem. The all-possible-regressions method suggests that we should use all of the regressors, since this produces the lowest value of C_p, and one of the highest values of R_{adj}^2. However, depending on the practical implications of using more regressors (cost, complexity, etc.) using just x_3 and x_4 still provides a low value of C_p and a high R_{adj}^2 by comparison.

(e) Compare and contrast the models produced by the variable selection in parts (a)-(d).

Forward selection, backward elimination, and stepwise regression all produced the same model selection (use of x_3 and x_4, only). The all-possible-regressions method resulted in the selection of all the regressors, though this only produced marginal gains in R^2_{adj}, and reduction of C_p. Therefore, all of the models are essentially in agreement, with slight exception given to the all-possible-regressions method.

4. Randomly select only 20 rows of Table B.1.

(a) Calculate the PRESS statistic for this model. What comments can you make about the likely predictive performance of this model?

The attached PDF provides results for the problem. The PRESS statistic was calculated as being 172.08. This large value for the PRESS statistic suggests that the model will have poor predictive performance, because the residuals associated with the out-of-sample population are large.

(b) Delete half the observations (chosen at random), and refit the regression model. Have the regression coefficients changed dramatically? How well does this model predict the number of games won for the deleted observations?

The attached PDF provides results for the problem. The coefficients in the newly produced model have changed dramatically. In fact, the number of regressors is equivalent to the number of data points, meaning that the model can fit all of the training data exactly. However, as per the bias-variance tradeoff, having such a complex model will significantly hamper the predictive capability of the model. Therefore, the model does not do well at predicting the number of games won concerning the deleted observations.

Import Tools

In [224]:
```
import pandas as pd
import random
import numpy as np
import statsmodels.api as sm
from itertools import chain, combinations
```

In [225]:
```
def powerset(iterable):

    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))
```

Problem 1

In [226]:
```
n = 14

df = pd.read_excel(r'C:\Users\maste\Downloads\linear_regression_5e_data_sets\line
                   '\Appendices\data-table-B4.XLS')
a = np.random.choice(['x1','x2','x3','x4','x5','x6','x7','x8','x9'], size=5, repl

sample = df.sample(n)
X = np.array(sample[a])
y = np.array(sample[['y']])
```

```
WARNING *** OLE2 inconsistency: SSCS size is 0 but SSAT size is non-zero
```

In [227]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X)
results = mod.fit()
s2_full = results.mse_resid
```

In [228]:
```
combos = list(powerset([1,2,3,4,5]))
```

In [229]:
```
X = sm.add_constant(X)
allruns = []
for c in combos:
    mod = sm.OLS(y, X[:,[0]+list(c)])
    results = mod.fit()
    msr = round(results.mse_resid,3)
    r2 = round(results.rsquared,3)
    r2a = round(results.rsquared_adj,3)
    cp = round((results.ssr/s2_full) - n + (2*(len(list(c))+1)),3)
    res = round(results.ssr,3)

    usestr = ''
    if c != ():
        for i in a[np.array(c)-1]:
            usestr += i

    run = [usestr,res,r2,r2a,msr,cp]
    allruns.append(run)
```

In [230]:
```
df = pd.DataFrame(allruns, columns = ['Regressors','SS_Res', 'R2', 'R2_Adj','MS_F
df
```

Out[230]:

	Regressors	SS_Res	R2	R2_Adj	MS_Res	C_p
0		453.314	0.000	0.000	34.870	103.121
1	x6	364.617	0.196	0.129	30.385	82.596
2	x4	226.122	0.501	0.460	18.843	47.425
3	x8	380.829	0.160	0.090	31.736	86.713
4	x2	278.440	0.386	0.335	23.203	60.711
5	x1	42.123	0.907	0.899	3.510	0.697
6	x6x4	225.651	0.502	0.412	20.514	49.305
7	x6x8	211.640	0.533	0.448	19.240	45.747
8	x6x2	254.315	0.439	0.337	23.120	56.585
9	x6x1	42.090	0.907	0.890	3.826	2.689
10	x4x8	166.952	0.632	0.565	15.177	34.398
11	x4x2	214.713	0.526	0.440	19.519	46.528
12	x4x1	34.359	0.924	0.910	3.124	0.726
13	x8x2	231.594	0.489	0.396	21.054	50.815
14	x8x1	42.103	0.907	0.890	3.828	2.692
15	x2x1	41.756	0.908	0.891	3.796	2.604
16	x6x4x8	144.991	0.680	0.584	14.499	30.821
17	x6x4x2	213.626	0.529	0.387	21.363	48.251
18	x6x4x1	33.066	0.927	0.905	3.307	2.397
19	x6x8x2	159.207	0.649	0.543	15.921	34.432
20	x6x8x1	42.089	0.907	0.879	4.209	4.689
21	x6x2x1	41.743	0.908	0.880	4.174	4.601
22	x4x8x2	161.252	0.644	0.538	16.125	34.951
23	x4x8x1	33.778	0.925	0.903	3.378	2.578
24	x4x2x1	33.247	0.927	0.905	3.325	2.443
25	x8x2x1	41.756	0.908	0.880	4.176	4.604
26	x6x4x8x2	138.096	0.695	0.560	15.344	31.070
27	x6x4x8x1	33.059	0.927	0.895	3.673	4.395
28	x6x4x2x1	31.516	0.930	0.900	3.502	4.004
29	x6x8x2x1	41.738	0.908	0.867	4.638	6.600
30	x4x8x2x1	32.721	0.928	0.896	3.636	4.310
31	x6x4x8x2x1	31.502	0.931	0.887	3.938	6.000

Problem 2

In [215]:
```
n = 14

df = pd.read_excel(r'C:\Users\maste\Downloads\linear_regression_5e_data_sets\line
                   '\Appendices\data-table-B11.XLS')


sample = df.sample(n)
```

```
WARNING *** OLE2 inconsistency: SSCS size is 0 but SSAT size is non-zero
```

In [216]:
```
iv1 = []
iv2 = []
for i in list(sample['Region']):
    if i == 1:
        iv1.append(0)
        iv2.append(0)
    if i == 2:
        iv1.append(1)
        iv2.append(0)
    if i == 3:
        iv1.append(0)
        iv2.append(1)
```

In [217]:
```
sample['iv1'] = iv1
sample['iv2'] = iv2
X = np.array(sample[['Clarity', 'Aroma', 'Body', 'Flavor', 'Oakiness', 'iv1', 'iv
y = np.array(sample[['Quality']])
a = np.array(['Clarity', 'Aroma', 'Body', 'Flavor', 'Oakiness', 'iv1', 'iv2'])
```

In [218]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X)
results = mod.fit()
s2_full = results.mse_resid
```

In [219]:
```
combos = list(powerset([1,2,3,4,5,6,7]))
```

In [220]:
```
X = sm.add_constant(X)
allruns = []
for c in combos:
    mod = sm.OLS(y, X[:,[0]+list(c)])
    results = mod.fit()
    msr = round(results.mse_resid,3)
    r2 = round(results.rsquared,3)
    r2a = round(results.rsquared_adj,3)
    cp = round((results.ssr/s2_full) - n + (2*(len(list(c))+1)),3)
    res = round(results.ssr,3)

    usestr = ''
    if c != ():
        for i in a[np.array(c)-1]:
            usestr += i

    run = [usestr,res,r2,r2a,msr,cp]
    allruns.append(run)
```

In [223]:
```
df = pd.DataFrame(allruns, columns = ['Regressors','SS_Res', 'R2', 'R2_Adj','MS_R
print(df.to_string())
```

```
                Regressors  SS_Res     R2  R2_Adj  MS_Res     C_
p
0                           55.144  0.000   0.000   4.242  47.00
1
1                  Clarity  31.359  0.431   0.384   2.613  23.55
3
2                    Aroma  31.079  0.436   0.389   2.590  23.25
3
3                     Body  19.272  0.651   0.621   1.606  10.62
1
4                   Flavor  23.589  0.572   0.537   1.966  15.23
9
5                 Oakiness  54.348  0.014  -0.068   4.529  48.15
1
6                      iv1  37.863  0.313   0.256   3.155  30.51
1
7                      iv2  25.308  0.541   0.503   2.109  17.07
9
8             ClarityAroma  27.447  0.502   0.412   2.495  21.36
7
9              ClarityBody  17.381  0.685   0.627   1.580  10.59
7
10           ClarityFlavor  19.923  0.639   0.573   1.811  13.31
7
11         ClarityOakiness  31.324  0.432   0.329   2.848  25.51
5
12              Clarityiv1  19.252  0.651   0.587   1.750  12.59
9
13              Clarityiv2  18.746  0.660   0.598   1.704  12.05
7
14               AromaBody  18.206  0.670   0.610   1.655  11.48
0
15             AromaFlavor  23.184  0.580   0.503   2.108  16.80
6
16           AromaOakiness  29.490  0.465   0.368   2.681  23.55
3
17                Aromaiv1  13.589  0.754   0.709   1.235   6.53
9
18                Aromaiv2  21.736  0.606   0.534   1.976  15.25
6
19              BodyFlavor  18.063  0.672   0.613   1.642  11.32
6
20            BodyOakiness  17.531  0.682   0.624   1.594  10.75
8
21                 Bodyiv1   9.063  0.836   0.806   0.824   1.69
7
22                 Bodyiv2  15.930  0.711   0.659   1.448   9.04
4
23          FlavorOakiness  14.239  0.742   0.695   1.294   7.23
6
24               Flavoriv1  14.812  0.731   0.683   1.347   7.84
9
25               Flavoriv2  16.209  0.706   0.653   1.474   9.34
2
```

```
26                    Oakinessiv1  37.311  0.323  0.200  3.392  31.92
1
27                    Oakinessiv2  24.887  0.549  0.467  2.262  18.62
8
28                         iv1iv2  18.448  0.665  0.605  1.677  11.73
9
29               ClarityAromaBody  13.888  0.748  0.673  1.389   8.85
9
30             ClarityAromaFlavor  19.741  0.642  0.535  1.974  15.12
2
31           ClarityAromaOakiness  26.902  0.512  0.366  2.690  22.78
4
32                ClarityAromaiv1  12.561  0.772  0.704  1.256   7.44
0
33                ClarityAromaiv2  18.605  0.663  0.561  1.861  13.90
7
34              ClarityBodyFlavor  16.548  0.700  0.610  1.655  11.70
6
35            ClarityBodyOakiness  16.359  0.703  0.614  1.636  11.50
4
36                 ClarityBodyiv1   7.597  0.862  0.821  0.760   2.12
9
37                 ClarityBodyiv2  14.384  0.739  0.661  1.438   9.39
0
38          ClarityFlavorOakiness  13.944  0.747  0.671  1.394   8.91
9
39               ClarityFlavoriv1  11.321  0.795  0.733  1.132   6.11
3
40               ClarityFlavoriv2  14.329  0.740  0.662  1.433   9.33
1
41             ClarityOakinessiv1  17.887  0.676  0.578  1.789  13.13
9
42             ClarityOakinessiv2  18.280  0.669  0.569  1.828  13.55
9
43                   Clarityiv1iv2  11.886  0.784  0.720  1.189   6.71
7
44                AromaBodyFlavor  16.315  0.704  0.615  1.631  11.45
6
45              AromaBodyOakiness  16.530  0.700  0.610  1.653  11.68
7
46                   AromaBodyiv1   9.051  0.836  0.787  0.905   3.68
5
47                   AromaBodyiv2  14.744  0.733  0.652  1.474   9.77
6
48            AromaFlavorOakiness  13.931  0.747  0.672  1.393   8.90
5
49                 AromaFlavoriv1  11.946  0.783  0.718  1.195   6.78
1
50                 AromaFlavoriv2  16.118  0.708  0.620  1.612  11.24
6
51               AromaOakinessiv1  13.461  0.756  0.683  1.346   8.40
3
52               AromaOakinessiv2  21.735  0.606  0.488  2.174  17.25
6
53                    Aromaiv1iv2  11.258  0.796  0.735  1.126   6.04
5
54             BodyFlavorOakiness  12.961  0.765  0.694  1.296   7.86
```

```
8
55                  BodyFlavoriv1   8.703  0.842   0.795   0.870   3.31
2
56                  BodyFlavoriv2  14.704  0.733   0.653   1.470   9.73
2
57                BodyOakinessiv1   9.061  0.836   0.786   0.906   3.69
5
58                BodyOakinessiv2  15.626  0.717   0.632   1.563  10.71
9
59                     Bodyiv1iv2   8.021  0.855   0.811   0.802   2.58
2
60             FlavorOakinessiv1  11.527  0.791   0.728   1.153   6.33
3
61             FlavorOakinessiv2  13.192  0.761   0.689   1.319   8.11
5
62                   Flavoriv1iv2  10.313  0.813   0.757   1.031   5.03
5
63                 Oakinessiv1iv2  15.956  0.711   0.624   1.596  11.07
3
64         ClarityAromaBodyFlavor  12.232  0.778   0.680   1.359   9.08
8
65       ClarityAromaBodyOakiness  13.367  0.758   0.650   1.485  10.30
2
66            ClarityAromaBodyiv1   7.249  0.869   0.810   0.805   3.75
6
67            ClarityAromaBodyiv2  10.916  0.802   0.714   1.213   7.67
9
68     ClarityAromaFlavorOakiness  13.192  0.761   0.654   1.466  10.11
4
69          ClarityAromaFlavoriv1  10.707  0.806   0.720   1.190   7.45
6
70          ClarityAromaFlavoriv2  13.081  0.763   0.657   1.453   9.99
6
71        ClarityAromaOakinessiv1  12.192  0.779   0.681   1.355   9.04
5
72        ClarityAromaOakinessiv2  18.273  0.669   0.521   2.030  15.55
1
73             ClarityAromaiv1iv2  10.079  0.817   0.736   1.120   6.78
4
74      ClarityBodyFlavorOakiness  12.812  0.768   0.664   1.424   9.70
8
75           ClarityBodyFlavoriv1   7.404  0.866   0.806   0.823   3.92
2
76           ClarityBodyFlavoriv2  13.504  0.755   0.646   1.500  10.44
8
77         ClarityBodyOakinessiv1   7.531  0.863   0.803   0.837   4.05
8
78         ClarityBodyOakinessiv2  14.316  0.740   0.625   1.591  11.31
8
79              ClarityBodyiv1iv2   6.700  0.878   0.824   0.744   3.16
9
80       ClarityFlavorOakinessiv1  10.391  0.812   0.728   1.155   7.11
8
81       ClarityFlavorOakinessiv2  12.740  0.769   0.666   1.416   9.63
1
82            ClarityFlavoriv1iv2   8.211  0.851   0.785   0.912   4.78
6
```

83	ClarityOakinessiv1iv2	9.279	0.832	0.757	1.031	5.929
84	AromaBodyFlavorOakiness	10.385	0.812	0.728	1.154	7.111
85	AromaBodyFlavoriv1	8.696	0.842	0.772	0.966	5.305
86	AromaBodyFlavoriv2	12.794	0.768	0.665	1.422	9.689
87	AromaBodyOakinessiv1	9.050	0.836	0.763	1.006	5.683
88	AromaBodyOakinessiv2	14.489	0.737	0.620	1.610	11.503
89	AromaBodyiv1iv2	8.020	0.855	0.790	0.891	4.581
90	AromaFlavorOakinessiv1	11.093	0.799	0.709	1.233	7.869
91	AromaFlavorOakinessiv2	12.691	0.770	0.668	1.410	9.579
92	AromaFlavoriv1iv2	9.609	0.826	0.748	1.068	6.282
93	AromaOakinessiv1iv2	10.476	0.810	0.726	1.164	7.209
94	BodyFlavorOakinessiv1	8.347	0.849	0.781	0.927	4.931
95	BodyFlavorOakinessiv2	12.411	0.775	0.675	1.379	9.279
96	BodyFlavoriv1iv2	7.598	0.862	0.801	0.844	4.129
97	BodyOakinessiv1iv2	7.876	0.857	0.794	0.875	4.426
98	FlavorOakinessiv1iv2	9.896	0.821	0.741	1.100	6.589
99	ClarityAromaBodyFlavorOakiness	8.958	0.838	0.736	1.120	7.585
100	ClarityAromaBodyFlavoriv1	6.849	0.876	0.798	0.856	5.329
101	ClarityAromaBodyFlavoriv2	9.196	0.833	0.729	1.149	7.839
102	ClarityAromaBodyOakinessiv1	7.187	0.870	0.788	0.898	5.690
103	ClarityAromaBodyOakinessiv2	10.911	0.802	0.678	1.364	9.674
104	ClarityAromaBodyiv1iv2	6.193	0.888	0.818	0.774	4.626
105	ClarityAromaFlavorOakinessiv1	10.281	0.814	0.697	1.285	9.000
106	ClarityAromaFlavorOakinessiv2	11.466	0.792	0.662	1.433	10.269
107	ClarityAromaFlavoriv1iv2	8.204	0.851	0.758	1.026	6.778
108	ClarityAromaOakinessiv1iv2	8.628	0.844	0.746	1.078	7.232
109	ClarityBodyFlavorOakinessiv1	7.396	0.866	0.782	0.925	5.914
110	ClarityBodyFlavorOakinessiv2	12.142	0.780	0.642	1.518	10.991
111	ClarityBodyFlavoriv1iv2	6.456	0.883	0.810	0.807	4.90

```
7
112              ClarityBodyOakinessiv1iv2   6.237  0.887   0.816   0.780   4.67
4
113            ClarityFlavorOakinessiv1iv2   8.194  0.851   0.759   1.024   6.76
7
114             AromaBodyFlavorOakinessiv1   8.173  0.852   0.759   1.022   6.74
5
115             AromaBodyFlavorOakinessiv2   9.920  0.820   0.708   1.240   8.61
4
116                  AromaBodyFlavoriv1iv2   7.534  0.863   0.778   0.942   6.06
1
117                AromaBodyOakinessiv1iv2   7.875  0.857   0.768   0.984   6.42
6
118              AromaFlavorOakinessiv1iv2   9.561  0.827   0.718   1.195   8.23
0
119               BodyFlavorOakinessiv1iv2   7.587  0.862   0.776   0.948   6.11
8
120     ClarityAromaBodyFlavorOakinessiv1   6.725  0.878   0.774   0.961   7.19
5
121     ClarityAromaBodyFlavorOakinessiv2   8.156  0.852   0.725   1.165   8.72
7
122           ClarityAromaBodyFlavoriv1iv2   5.652  0.897   0.810   0.807   6.04
8
123         ClarityAromaBodyOakinessiv1iv2   5.691  0.897   0.808   0.813   6.08
9
124       ClarityAromaFlavorOakinessiv1iv2   8.173  0.852   0.725   1.168   8.74
5
125        ClarityBodyFlavorOakinessiv1iv2   6.236  0.887   0.790   0.891   6.67
2
126          AromaBodyFlavorOakinessiv1iv2   7.472  0.865   0.748   1.067   7.99
4
127  ClarityAromaBodyFlavorOakinessiv1iv2   5.608  0.898   0.780   0.935   8.00
0
```

Problem 3(a)

In [289]:
```
n = 20

df = pd.read_excel(r'C:\Users\maste\Downloads\linear_regression_5e_data_sets\line
                   '\Appendices\data-table-B2.XLS')

sample = df.sample(n)

X = np.array(sample[['x1','x2','x3','x4','x5']])
y = np.array(sample[['y']])
```

```
WARNING *** OLE2 inconsistency: SSCS size is 0 but SSAT size is non-zero
```

In [290]:
```
for k in range(1,6):
    X = sm.add_constant(X)
    mod = sm.OLS(y, X[:,[0,k]])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    F = ess_model/mse_model
    print('x'+str(k)+', F-stat, SS_R     ', round(F,5), '      ', round(ess_model,5
```

```
x1, F-stat, SS_R     7.33697       2836.5042
x2, F-stat, SS_R     0.03151       17.11627
x3, F-stat, SS_R     0.70183       367.59273
x4, F-stat, SS_R     63.77444       7639.2436
x5, F-stat, SS_R     1.58842       794.30316
```

In [291]:
```
SSx4 = 7639.2436
```

In [292]:
```
for k in range(1,6):
    X = sm.add_constant(X)
    mod = sm.OLS(y, X[:,[0,k,4]])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    print('x'+str(k)+'|x4'+' F(|), SS_R(|)    ', round((ess_model-SSx4)/mse_model
```

```
x1|x4 F(|), SS_R(|)     0.30771 38.33322
x2|x4 F(|), SS_R(|)     1.82766 209.30317
x3|x4 F(|), SS_R(|)     22.72605 1233.45931
x4|x4 F(|), SS_R(|)     -0.0 -0.0
x5|x4 F(|), SS_R(|)     7.07588 633.68656
```

In [293]:
```
SSx3x4 = 1233.45931 + SSx4
```

In [294]:
```
for k in range(1,6):
    X = sm.add_constant(X)
    mod = sm.OLS(y, X[:,[0,k,4,3]])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    print('x'+str(k)+'|x4x3'+' F(|), SS_R(|)    ', round((ess_model-SSx3x4)/mse_m
```

```
x1|x4x3 F(|), SS_R(|)     1.26565 67.63649
x2|x4x3 F(|), SS_R(|)     0.00367 0.21152
x3|x4x3 F(|), SS_R(|)     -0.0 -0.0
x4|x4x3 F(|), SS_R(|)     -0.0 -0.0
x5|x4x3 F(|), SS_R(|)     0.06573 3.77498
```

Problem 3(b)

In [328]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X[:,:])
results = mod.fit()
fmse_model = results.mse_resid
fess_model = results.ess
```

In [329]:
```
for k in range(1,6):
    X = sm.add_constant(X)
    d = [0,1,2,3,4,5]
    d.remove(k)
    mod = sm.OLS(y, X[:,d])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    print('x|'+'Without x'+str(k) +' F(|), SS_R(|)     ', round((fess_model-ess_mo
          round(fess_model-ess_model,5))
```

```
x|Without x1 F(|), SS_R(|)     2.94507 159.50673
x|Without x2 F(|), SS_R(|)     0.43206 23.40078
x|Without x3 F(|), SS_R(|)     6.10454 330.62582
x|Without x4 F(|), SS_R(|)     74.50313 4035.13872
x|Without x5 F(|), SS_R(|)     1.40724 76.21703
```

In [330]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X[:,[0,1,3,4,5]])
results = mod.fit()
fmse_model = results.mse_resid
fess_model = results.ess
```

In [331]:
```
for k in [1,3,4,5]:
    X = sm.add_constant(X)
    d = [0,1,3,4,5]
    d.remove(k)
    mod = sm.OLS(y, X[:,d])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    print('x|'+'Without x2x'+str(k) +' F(|), SS_R(|)     ', round((fess_model-ess_
          round(fess_model-ess_model,5))
```

```
x|Without x2x1 F(|), SS_R(|)     2.63389 137.25204
x|Without x2x3 F(|), SS_R(|)     5.91963 308.47171
x|Without x2x4 F(|), SS_R(|)     98.21312 5117.88637
x|Without x2x5 F(|), SS_R(|)     1.40838 73.39053
```

In [332]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X[:,[0,1,3,4]])
results = mod.fit()
fmse_model = results.mse_resid
fess_model = results.ess
```

In [333]:
```
for k in [1,3,4]:
    X = sm.add_constant(X)
    d = [0,1,3,4]
    d.remove(k)
    mod = sm.OLS(y, X[:,d])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    print('x|'+'Without x5x2x'+str(k) +' F(|), SS_R(|)    ', round((fess_model-es
          round(fess_model-ess_model,5))
```

```
x|Without x5x2x1 F(|), SS_R(|)     1.26565 67.63649
x|Without x5x2x3 F(|), SS_R(|)     23.62952 1262.76258
x|Without x5x2x4 F(|), SS_R(|)     100.43948 5367.48935
```

In [334]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X[:,[0,3,4]])
results = mod.fit()
fmse_model = results.mse_resid
fess_model = results.ess
```

In [335]:
```
for k in [3,4]:
    X = sm.add_constant(X)
    d = [0,3,4]
    d.remove(k)
    mod = sm.OLS(y, X[:,d])
    results = mod.fit()
    mse_model = results.mse_resid
    ess_model = results.ess
    print('x|'+'Without x1x5x2x'+str(k) +' F(|), SS_R(|)    ', round((fess_model
          round(fess_model-ess_model,5))
```

```
x|Without x1x5x2x3 F(|), SS_R(|)     22.72605 1233.45931
x|Without x1x5x2x4 F(|), SS_R(|)     156.70366 8505.11018
```

Problem 3(c)

In [340]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X[:,[0,4]])
results = mod.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.780
Model:                            OLS   Adj. R-squared:                  0.768
Method:                 Least Squares   F-statistic:                     63.77
Date:                Wed, 23 Mar 2022   Prob (F-statistic):           2.51e-07
Time:                        15:20:56   Log-Likelihood:                -75.182
No. Observations:                  20   AIC:                             154.4
Df Residuals:                      18   BIC:                             156.4
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        638.6626     48.443     13.184      0.000     536.889     740.437
x1           -23.3135      2.919     -7.986      0.000     -29.447     -17.180
==============================================================================
Omnibus:                        0.422   Durbin-Watson:                   2.030
Prob(Omnibus):                  0.810   Jarque-Bera (JB):                0.156
Skew:                          -0.207   Prob(JB):                        0.925
Kurtosis:                       2.878   Cond. No.                         330.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

In [341]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X[:,[0,3,4]])
results = mod.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.906
Model:                            OLS   Adj. R-squared:                  0.895
Method:                 Least Squares   F-statistic:                     81.74
Date:                Wed, 23 Mar 2022   Prob (F-statistic):           1.90e-09
Time:                        15:21:49   Log-Likelihood:                -66.694
No. Observations:                  20   AIC:                             139.4
Df Residuals:                      17   BIC:                             142.4
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        519.3443     41.106     12.634      0.000     432.617     606.071
x1             4.1474      0.870      4.767      0.000       2.312       5.983
x2           -24.9901      1.996    -12.518      0.000     -29.202     -20.778
==============================================================================
Omnibus:                        0.368   Durbin-Watson:                   2.713
Prob(Omnibus):                  0.832   Jarque-Bera (JB):                0.016
Skew:                          -0.069   Prob(JB):                        0.992
Kurtosis:                       2.983   Cond. No.                         979.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

Problem 3(d)

In [336]:
```
combos = list(powerset([1,2,3,4,5]))
a = np.array(['x1','x2','x3','x4','x5'])
```

In [337]:
```
X = sm.add_constant(X)
allruns = []
for c in combos:
    mod = sm.OLS(y, X[:,[0]+list(c)])
    results = mod.fit()
    msr = round(results.mse_resid,3)
    r2 = round(results.rsquared,3)
    r2a = round(results.rsquared_adj,3)
    cp = round((results.ssr/s2_full) - n + (2*(len(list(c))+1)),3)
    res = round(results.ssr,3)

    usestr = ''
    if c != ():
        for i in a[np.array(c)-1]:
            usestr += i

    run = [usestr,res,r2,r2a,msr,cp]
    allruns.append(run)
```

In [338]:
```
df = pd.DataFrame(allruns, columns = ['Regressors','SS_Res', 'R2', 'R2_Adj','MS_F
df
```

Out[338]:

	Regressors	SS_Res	R2	R2_Adj	MS_Res	C_p
0		9795.380	0.000	0.000	515.546	2469.585
1	x1	6958.876	0.290	0.250	386.604	1751.241
2	x2	9778.264	0.002	-0.054	543.237	2467.239
3	x3	9427.787	0.038	-0.016	523.766	2378.233
4	x4	2156.136	0.780	0.768	119.785	531.562
5	x5	9001.077	0.081	0.030	500.060	2269.868
6	x1x2	6615.172	0.325	0.245	389.128	1665.956
7	x1x3	6222.530	0.365	0.290	366.031	1566.242
8	x1x4	2117.803	0.784	0.758	124.577	523.827
9	x1x5	6900.117	0.296	0.213	405.889	1738.319
10	x2x3	9246.001	0.056	-0.055	543.882	2334.068
11	x2x4	1946.833	0.801	0.778	114.520	480.408
12	x2x5	9000.311	0.081	-0.027	529.430	2271.674
13	x3x4	922.677	0.906	0.895	54.275	220.318
14	x3x5	6569.699	0.329	0.250	386.453	1654.408
15	x4x5	1522.450	0.845	0.826	89.556	372.634
16	x1x2x3	4926.845	0.497	0.403	307.928	1239.197
17	x1x2x4	1946.600	0.801	0.764	121.663	482.349
18	x1x2x5	6280.577	0.359	0.239	392.536	1582.984
19	x1x3x4	855.041	0.913	0.896	53.440	205.142
20	x1x3x5	5899.536	0.398	0.285	368.721	1486.217
21	x1x4x5	1090.122	0.889	0.868	68.133	264.842
22	x2x3x4	922.466	0.906	0.888	57.654	222.265
23	x2x3x5	6120.514	0.375	0.258	382.532	1542.335
24	x2x4x5	1362.780	0.861	0.835	85.174	334.085
25	x3x4x5	918.902	0.906	0.889	57.431	221.360
26	x1x2x3x4	834.466	0.915	0.892	55.631	201.917
27	x1x2x3x5	4793.388	0.511	0.380	319.559	1207.305
28	x1x2x4x5	1088.875	0.889	0.859	72.592	266.525
29	x1x3x4x5	781.650	0.920	0.899	52.110	188.504
30	x2x3x4x5	917.756	0.906	0.881	61.184	223.069
31	x1x2x3x4x5	758.249	0.923	0.895	54.161	184.561

Problem 4(a)

In [361]:
```
n = 20

df = pd.read_excel(r'C:\Users\maste\Downloads\linear_regression_5e_data_sets\line
                   '\Appendices\data-table-B1.XLS')

sample = df.sample(n)

X = np.array(sample[['x1','x2','x3','x4','x5','x6','x7','x8','x9']])
y = np.array(sample[['y']])
sample
```

```
WARNING *** OLE2 inconsistency: SSCS size is 0 but SSAT size is non-zero
```

Out[361]:

	y	x1	x2	x3	x4	x5	x6	x7	x8	x9
24	6	2040	2416	38.7	50.0	0	576	54.9	2048	2628
0	10	2113	1985	38.9	64.7	4	868	59.7	2205	1917
12	9	2295	2229	37.4	53.6	-5	1037	58.8	1761	2032
14	6	2213	2140	38.8	58.3	6	819	59.2	1901	1686
21	3	1929	1606	39.7	68.8	-21	627	52.7	2592	2324
2	11	2957	1737	40.1	60.0	14	914	65.6	1847	2175
17	5	1873	2929	41.1	55.3	10	789	54.3	2861	2496
10	7	2363	1480	37.3	48.0	19	984	67.5	1984	2217
6	10	2528	2341	38.1	65.4	12	754	66.1	1564	2092
1	11	2003	2855	38.8	61.3	3	615	55.0	2096	1575
13	9	1932	2204	35.1	71.4	3	986	58.6	1709	2025
15	5	1722	1730	36.6	52.6	-19	791	54.4	2288	1835
7	11	2147	2737	37.0	78.3	-1	761	58.0	1821	1909
5	11	2309	2927	39.7	74.1	8	786	61.0	1848	2339
18	6	2118	2268	38.2	69.6	6	582	58.7	2411	2670
9	2	2566	1838	42.3	54.2	-1	797	58.9	2476	2254
23	10	2301	2835	35.3	74.1	2	683	59.7	1979	2110
11	10	2109	2191	39.5	51.9	6	700	57.2	1917	1758
25	8	2447	1638	39.9	57.1	-8	848	65.3	1786	1776
3	13	2285	2905	41.6	45.3	-4	957	61.4	1903	2476

In [362]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X)
results = mod.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.735
Model:                            OLS   Adj. R-squared:                  0.497
Method:                 Least Squares   F-statistic:                     3.089
Date:                Thu, 24 Mar 2022   Prob (F-statistic):             0.0468
Time:                        10:02:22   Log-Likelihood:                -36.513
No. Observations:                  20   AIC:                             93.03
Df Residuals:                      10   BIC:                             103.0
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -6.5928     20.444     -0.322      0.754     -52.145      38.959
x1            -0.0009      0.004     -0.239      0.816      -0.009       0.007
x2             0.0034      0.001      2.315      0.043       0.000       0.007
x3             0.1623      0.423      0.384      0.709      -0.780       1.105
x4             0.0220      0.066      0.331      0.747      -0.126       0.170
x5            -0.0023      0.071     -0.032      0.975      -0.160       0.156
x6             0.0022      0.005      0.483      0.639      -0.008       0.012
x7             0.1952      0.283      0.691      0.506      -0.435       0.825
x8            -0.0043      0.003     -1.493      0.166      -0.011       0.002
x9            -0.0015      0.002     -0.797      0.444      -0.006       0.003
==============================================================================
Omnibus:                        0.116   Durbin-Watson:                   2.465
Prob(Omnibus):                  0.944   Jarque-Bera (JB):                0.057
Skew:                          -0.062   Prob(JB):                        0.972
Kurtosis:                       2.769   Cond. No.                     1.90e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 1.9e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [363]:
```
res = results.resid
H = X@np.linalg.inv(X.T@X)@X.T
lev = np.diagonal(H)
PRESS_res = (res/(1-lev))**2

PRESS_stat = sum(PRESS_res)
print('PRESS-Statistic:   ', PRESS_stat)
```

```
PRESS-Statistic:    172.08112336483873
```

Problem 4(b)

In [364]:
```
sample1 = sample.sample(int(n/2))
X = np.array(sample1[['x1','x2','x3','x4','x5','x6','x7','x8','x9']])
y = np.array(sample1[['y']])
sample1
```

Out[364]:

	y	x1	x2	x3	x4	x5	x6	x7	x8	x9
9	2	2566	1838	42.3	54.2	-1	797	58.9	2476	2254
13	9	1932	2204	35.1	71.4	3	986	58.6	1709	2025
3	13	2285	2905	41.6	45.3	-4	957	61.4	1903	2476
17	5	1873	2929	41.1	55.3	10	789	54.3	2861	2496
23	10	2301	2835	35.3	74.1	2	683	59.7	1979	2110
14	6	2213	2140	38.8	58.3	6	819	59.2	1901	1686
15	5	1722	1730	36.6	52.6	-19	791	54.4	2288	1835
10	7	2363	1480	37.3	48.0	19	984	67.5	1984	2217
6	10	2528	2341	38.1	65.4	12	754	66.1	1564	2092
7	11	2147	2737	37.0	78.3	-1	761	58.0	1821	1909

In [359]:
```
X = sm.add_constant(X)
mod = sm.OLS(y, X)
results = mod.fit()
print(results.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                    nan
Method:                 Least Squares   F-statistic:                       nan
Date:                Thu, 24 Mar 2022   Prob (F-statistic):                nan
Time:                        10:02:04   Log-Likelihood:                 283.97
No. Observations:                  10   AIC:                            -547.9
Df Residuals:                       0   BIC:                            -544.9
Df Model:                           9                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -71.9559        inf         -0        nan         nan         nan
x1             0.0015        inf          0        nan         nan         nan
x2             0.0103        inf          0        nan         nan         nan
x3            -1.2463        inf         -0        nan         nan         nan
x4             0.0915        inf          0        nan         nan         nan
x5             0.5296        inf          0        nan         nan         nan
x6             0.0545        inf          0        nan         nan         nan
x7             1.4377        inf          0        nan         nan         nan
x8             0.0346        inf          0        nan         nan         nan
x9            -0.0506        inf         -0        nan         nan         nan
==============================================================================
Omnibus:                        0.095   Durbin-Watson:                   2.561
Prob(Omnibus):                  0.953   Jarque-Bera (JB):                0.309
Skew:                           0.106   Prob(JB):                        0.857
Kurtosis:                       2.166   Cond. No.                     4.70e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 4.7e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

C:\Users\maste\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarni
ng: kurtosistest only valid for n>=20 ... continuing anyway, n=10
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\maste\anaconda3\lib\site-packages\statsmodels\regression\linear_model.
py:1728: RuntimeWarning: divide by zero encountered in true_divide
  return 1 - (np.divide(self.nobs - self.k_constant, self.df_resid)
C:\Users\maste\anaconda3\lib\site-packages\statsmodels\regression\linear_model.
py:1728: RuntimeWarning: invalid value encountered in double_scalars
  return 1 - (np.divide(self.nobs - self.k_constant, self.df_resid)
C:\Users\maste\anaconda3\lib\site-packages\statsmodels\regression\linear_model.
py:1650: RuntimeWarning: divide by zero encountered in double_scalars
  return np.dot(wresid, wresid) / self.df_resid
```

In [ ]: