Homework 1 Solution

AI 539 - Machine Learning for Non-AI Majors
Instructor: Alireza Aghasi

April 7, 2024

Q1. (a) Consider the following system of two equations, where x and y are the unknowns:

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}. \tag{1}$$

Show that if $a_1 b_2 - a_2 b_1 \neq 0$, then simultanuosly solving the system above for x and y yields

$$x = \frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - a_2 b_1}, \qquad y = \frac{a_1 c_2 - c_1 a_2}{a_1 b_2 - a_2 b_1}.$$

Solution. This is a straightforward problem, that you are all familiar with from basic algebra. While you can use the Cramer's rule, I go through the derivation to keep the solution self-contained. We can multiply the first equation by a_2 and the second by a_1 to get

$$\begin{cases} a_2 a_1 x + a_2 b_1 y = a_2 c_1 \\ a_1 a_2 x + a_1 b_2 y = a_1 c_2 \end{cases}.$$

Now subtracting the resulting first equation from the second one gives:

$$(a_1 b_2 - a_2 b_1) y = a_1 c_2 - a_2 c_1,$$

therefore

$$y = \frac{a_1 c_2 - c_1 a_2}{a_1 b_2 - a_2 b_1}.$$

We can go through a similar process for x by multiplying the first equation in (1) by b_2 and the second one by b_1, which after a similar procedure yield

$$x = \frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - a_2 b_1}.$$

(b) The goal is to find the optimal values of β_1 and β_2 which fit the general model

$$y = \beta_1 g(x) + \beta_2 f(x) \tag{2}$$

to the data points $(x_1, y_1), \cdots, (x_n, y_n)$. Here $f(x) \in \mathbb{R}$ and $g(x) \in \mathbb{R}$ are some arbitrary functions of x. Use the result in part (a) to mathematically show that these optimal values are

$$\hat{\beta}_1 = \frac{\left(\sum_{i=1}^n y_i g_i\right)\left(\sum_{i=1}^n f_i^2\right) - \left(\sum_{i=1}^n y_i f_i\right)\left(\sum_{i=1}^n g_i f_i\right)}{\left(\sum_{i=1}^n g_i^2\right)\left(\sum_{i=1}^n f_i^2\right) - \left(\sum_{i=1}^n f_i g_i\right)^2}, \tag{3}$$

$$\hat{\beta}_2 = \frac{\left(\sum_{i=1}^n y_i f_i\right)\left(\sum_{i=1}^n g_i^2\right) - \left(\sum_{i=1}^n y_i g_i\right)\left(\sum_{i=1}^n g_i f_i\right)}{\left(\sum_{i=1}^n g_i^2\right)\left(\sum_{i=1}^n f_i^2\right) - \left(\sum_{i=1}^n f_i g_i\right)^2}. \tag{4}$$

For brevity we used the notations $g_i = g(x_i)$ and $f_i = f(x_i)$.

Hint: In the class we went through the exercise of showing that to fit a simple linear model

$$y = \beta_0 + \beta_1 x$$

to the data points $(x_1, y_1), \cdots, (x_n, y_n)$, the optimal choice of parameters are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}.$$

You should be able to follow a similar process to derive the equations (3) and (4).

Solution. We would need to minimize the difference between the model outout and the response variables y_i, or in other words:

$$\underset{\beta_1,\beta_2}{\text{minimize}} \sum_{i=1}^{n} (y_i - \beta_1 g(x_i) - \beta_2 f(x_i))^2$$

Let's denote the objective by

$$RSS = \sum_{i=1}^{n} (y_i - \beta_1 g_i - \beta_2 f_i)^2,$$

then in order to minimize C we would need to take derivatives wrt β_1 and β_2 and set them to zero:

$$\frac{\partial RSS}{\partial \beta_1} = -2 \sum_{i=1}^{n} g_i (y_i - \beta_1 g_i - \beta_2 f_i) = 0 \tag{5}$$

$$\frac{\partial RSS}{\partial \beta_2} = -2 \sum_{i=1}^{n} f_i (y_i - \beta_1 g_i - \beta_2 f_i) = 0. \tag{6}$$

Or after rearranging the terms we end up with the following system of two equations and two unknowns:

$$\beta_1 \sum_{i=1}^{n} g_i^2 + \beta_2 \sum_{i=1}^{n} g_i f_i = \sum_{i=1}^{n} g_i y_i, \tag{7}$$

$$\beta_1 \sum_{i=1}^{n} f_i g_i + \beta_2 \sum_{i=1}^{n} f_i^2 = \sum_{i=1}^{n} f_i y_i. \tag{8}$$

We now can use the result in part (a) to solve these two equations for β_1 and β_2, which would give:

$$\hat{\beta}_1 = \frac{\left(\sum_{i=1}^{n} y_i g_i\right)\left(\sum_{i=1}^{n} f_i^2\right) - \left(\sum_{i=1}^{n} y_i f_i\right)\left(\sum_{i=1}^{n} g_i f_i\right)}{\left(\sum_{i=1}^{n} g_i^2\right)\left(\sum_{i=1}^{n} f_i^2\right) - \left(\sum_{i=1}^{n} f_i g_i\right)^2},$$

$$\hat{\beta}_2 = \frac{\left(\sum_{i=1}^{n} y_i f_i\right)\left(\sum_{i=1}^{n} g_i^2\right) - \left(\sum_{i=1}^{n} y_i g_i\right)\left(\sum_{i=1}^{n} g_i f_i\right)}{\left(\sum_{i=1}^{n} g_i^2\right)\left(\sum_{i=1}^{n} f_i^2\right) - \left(\sum_{i=1}^{n} f_i g_i\right)^2}.$$

(c) Now for the model defined in (2), assume that $g(x) = 0$, which reduces our model to $y = \beta f(x)$. Show that fitting this simplified model to the data points $(x_1, y_1), \cdots, (x_n, y_n)$, yields the following expression for the optimal value of β:

$$\hat{\beta} = \frac{\sum_{i=1}^{n} f_i y_i}{\sum_{i=1}^{n} f_i^2}. \tag{9}$$

Solution. We form the RSS as

$$RSS = \sum_{i=1}^{n}(y_i - \beta f(x_i))^2 = \sum_{i=1}^{n}(y_i - \beta f_i)^2$$

and set the derivative of the objective to zero

$$\frac{dRSS}{d\beta} = -2\sum_{i=1}^{n} f_i(y_i - \beta f_i) = 0.$$

Therefore

$$\beta \sum_{i=1}^{n} f_i^2 = \sum_{i=1}^{n} y_i f_i,$$

or

$$\hat{\beta} = \frac{\sum_{i=1}^{n} f_i y_i}{\sum_{i=1}^{n} f_i^2}.$$

(d) Is it possible to derive (9) by simply setting $g(x_i) = g_i = 0$ in (4)?

Solution. No. Setting $g_i = 0$ in (4), yields

$$\hat{\beta}_2 = \hat{\beta} = \frac{\left(\sum_{i=1}^{n} y_i f_i\right)(0) - (0)\,(0)}{(0)\left(\sum_{i=1}^{n} f_i^2\right) - (0)^2} = \frac{0}{0},$$

which is mathematically meaningless.

(e) In part (c) you showed that assuming that our data follows the model $y = \beta f(x) + \epsilon$, where ϵ is zero mean iid Gaussian noise, then the best estimate for $\hat{\beta}$ is obtained via (9). Using the basic properties of the expectation mentioned in the class, prove that $\hat{\beta}$ is an unbiased estimate of β. Basically you would need to show that

$$\mathbb{E}(\hat{\beta}) = \beta.$$

Solution. In the expression for $\hat{\beta}$, only the quantities y_i are random, therefore

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}\left(\frac{\sum_{i=1}^{n} f_i y_i}{\sum_{i=1}^{n} f_i^2}\right) = \frac{\mathbb{E}\left(\sum_{i=1}^{n} f_i y_i\right)}{\sum_{i=1}^{n} f_i^2} = \frac{\sum_{i=1}^{n} f_i \mathbb{E}(y_i)}{\sum_{i=1}^{n} f_i^2} = \frac{\sum_{i=1}^{n} f_i \mathbb{E}(\beta f_i + \epsilon_i)}{\sum_{i=1}^{n} f_i^2} = \frac{\sum_{i=1}^{n} \beta f_i^2}{\sum_{i=1}^{n} f_i^2} = \beta.$$

(f) Assume that for the true regression function we have $g(x) = 0$ and $f(x) = x^2$, meaning that our observations are in the form of $y_i = \beta x_i^2 + \epsilon_i$. Moreover, assume that ϵ_i represents i.i.d zero-mean noise, that is $\mathbb{E}\epsilon_i = 0$, $var(\epsilon_i) = \sigma^2$. Use (9) to derive the optimal fit $\hat{\beta}$ for the model $y = \beta x^2$, and then using the properties mentioned in the class about the variance of the sum of independent random variables, show that:

$$var(\hat{\beta}) = \frac{\sigma^2}{\sum_{i=1}^{n} x_i^4}.$$

Solution. After plugging $f_i = x_i^2$ in (9), we get

$$\hat{\beta} = \frac{\sum_{i=1}^n x_i^2 y_i}{\sum_{i=1}^n x_i^4}.$$

Notice that in this expression for $\hat{\beta}$, only the quantities y_i are random. But we know that $y_i = \beta x_i^2 + \epsilon_i$ and since the term βx_i^2 is a non-random parameter

$$var(y_i) = var(\epsilon_i) = \sigma^2.$$

Since the y_i's are independent of each other, by the properties of the variance stated in the class we have

$$var(\hat{\beta}) = \frac{\sum_{i=1}^n x_i^4 var(y_i)}{\left(\sum_{i=1}^n x_i^4\right)^2} = \frac{\sigma^2 \sum_{i=1}^n x_i^4}{\left(\sum_{i=1}^n x_i^4\right)^2} = \frac{\sigma^2}{\sum_{i=1}^n x_i^4}. \tag{10}$$

(g) Using the outcome of part (e), is it true that for the model $y = \beta x^2$, having samples with larger x can reduce the uncertainty in evaluating $\hat{\beta}$? (uncertainty has a similar interpretation as the variance here)

Solution. Yes, based on the equation

$$var(\hat{\beta}) = \frac{\sigma^2}{\sum_{i=1}^n x_i^4},$$

having larger values of x_i makes $var(\hat{\beta})$ smaller.

For the solutions to Q2 and Q3 see the attached notebook files.

Question 2

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
import statsmodels.api as sm
```

In [2]:
```
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize)
from sklearn.metrics import mean_squared_error
```

Parts (a-c)

In [3]:
```python
# reading the CSV file
df = pd.read_csv('fat.csv')
train = df.iloc[0:200,]
test = df.iloc[200:,]
terms = df.columns.drop('brozek')

# constructing the test and training data
Xtr = MS(terms).fit_transform(train)
ytr = train['brozek']
Xts = MS(terms).fit_transform(test)
yts = test['brozek']
model = sm.OLS(ytr, Xtr)
results = model.fit()
results.summary()
```

Out [3]:

OLS Regression Results

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.859e+04
Date:	Tue, 09 Apr 2024	**Prob (F-statistic):**	7.25e-285
Time:	12:09:06	**Log-Likelihood:**	61.275
No. Observations:	200	**AIC:**	-86.55
Df Residuals:	182	**BIC:**	-27.18
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	11.7984	4.641	2.542	0.012	2.641	20.956
siri	0.8845	0.013	66.439	0.000	0.858	0.911
density	-9.5175	4.172	-2.281	0.024	-17.750	-1.285
age	-0.0007	0.002	-0.417	0.677	-0.004	0.003
weight	0.0116	0.005	2.429	0.016	0.002	0.021
height	0.0004	0.005	0.075	0.940	-0.009	0.010
adipos	-0.0213	0.016	-1.361	0.175	-0.052	0.010
free	-0.0134	0.006	-2.324	0.021	-0.025	-0.002
neck	-0.0037	0.011	-0.325	0.746	-0.026	0.019
chest	0.0034	0.005	0.631	0.529	-0.007	0.014
abdom	0.0005	0.005	0.090	0.929	-0.010	0.011
hip	-0.0037	0.007	-0.496	0.620	-0.018	0.011
thigh	0.0199	0.008	2.579	0.011	0.005	0.035
knee	-0.0305	0.013	-2.386	0.018	-0.056	-0.005
ankle	0.0037	0.010	0.359	0.720	-0.017	0.024
biceps	-0.0159	0.009	-1.871	0.063	-0.033	0.001
forearm	0.0196	0.010	1.891	0.060	-0.001	0.040
wrist	0.0340	0.027	1.246	0.214	-0.020	0.088

Omnibus:	146.785	**Durbin-Watson:**	1.875
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8218.452
Skew:	2.048	**Prob(JB):**	0.00
Kurtosis:	34.136	**Cond. No.**	1.48e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.48e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [4]:

```
# predicting the test
new_predictions = results.get_prediction(Xts);
y_hat = new_predictions.predicted_mean
e1 = np.linalg.norm(yts- y_hat)
print(e1)
```

```
0.7537423698361571
```

Part(a) Multiple R-squared: 0.999, Adjusted R-squared: 0.999 F-statistic: 1.859e+04 on 17 and 182 DF, p-value: < 2.2e-16 R-squared is very close to 1, indicating that the model very accurately fits part (b), as observed above, the p-values associated with the following features seem problematic: age, height, adipos, neck, chest, abdom, hip, ankle, biceps, forearm and wrist

parts (d-e)

In [5]:
```
df2 = df.copy() #use this instead of df2=df since otherwise any changes to df2 applies to df as well
df2['density'] = 1/df2['density']
# renaming density to inv_density
df2.rename(columns = {'density':'inv_density'}, inplace = True)

train2 = df2.iloc[0:200,]
test2 = df2.iloc[200:,]
terms = df2.columns.drop('brozek')

# constructing the test and training data
Xtr2 = MS(terms).fit_transform(train2)
Xts2 = MS(terms).fit_transform(test2)
model = sm.OLS(ytr, Xtr2)
results2 = model.fit()
results2.summary()
```

Out[5]:

OLS Regression Results

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.852e+04
Date:	Tue, 09 Apr 2024	**Prob (F-statistic):**	1.00e-284
Time:	12:09:50	**Log-Likelihood:**	60.917
No. Observations:	200	**AIC:**	-85.83
Df Residuals:	182	**BIC:**	-26.46
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	-8.0827	4.536	-1.782	0.076	-17.032	0.867
siri	0.8865	0.013	67.497	0.000	0.861	0.912
inv_density	10.3645	4.867	2.129	0.035	0.761	19.968
age	-0.0007	0.002	-0.423	0.673	-0.004	0.003
weight	0.0109	0.005	2.287	0.023	0.001	0.020
height	0.0005	0.005	0.094	0.925	-0.009	0.010
adipos	-0.0206	0.016	-1.320	0.189	-0.051	0.010
free	-0.0125	0.006	-2.161	0.032	-0.024	-0.001
neck	-0.0038	0.011	-0.332	0.740	-0.026	0.019
chest	0.0031	0.005	0.583	0.561	-0.008	0.014
abdom	0.0005	0.005	0.098	0.922	-0.010	0.011
hip	-0.0039	0.007	-0.529	0.598	-0.019	0.011
thigh	0.0199	0.008	2.580	0.011	0.005	0.035
knee	-0.0305	0.013	-2.386	0.018	-0.056	-0.005
ankle	0.0036	0.010	0.352	0.725	-0.017	0.024
biceps	-0.0158	0.009	-1.850	0.066	-0.033	0.001
forearm	0.0193	0.010	1.858	0.065	-0.001	0.040
wrist	0.0339	0.027	1.239	0.217	-0.020	0.088

Omnibus:	144.182	**Durbin-Watson:**	1.875
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8277.916
Skew:	1.977	**Prob(JB):**	0.00
Kurtosis:	34.268	**Cond. No.**	1.58e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.58e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [6]:
```
# predicting the test
new_predictions = results2.get_prediction(Xts2);
y_hat2 = new_predictions.predicted_mean
e2 = np.linalg.norm(yts- y_hat2)
print(e2)
```

```
0.754275370669583
```

Multiple R-squared: 0.999, Adjusted R-squared: 0.999 F-statistic: 1.852e+04 on 17 and 182 DF, p-value: < 2.2e-16

part (e), as observed above, e2 = 0.7542248120179155

Parts (f-i)

In [7]:
```
df3 = pd.DataFrame(columns=['siri', 'sq_siri', 'inv_density', 'density'])
df3['siri'] = df['siri']
df3['sq_siri'] = np.square(df['siri'])
df3['inv_density'] = 1/df['density']
df3['density'] = df['density']

Train3 = df3.iloc[0:200,]
Test3 = df3.iloc[200:,]

Xtr3 = MS(['siri', 'sq_siri', 'inv_density', 'density']).fit_transform(Train3)
Xts3 = MS(['siri', 'sq_siri', 'inv_density', 'density']).fit_transform(Test3)

# constructing the test and training data
model = sm.OLS(ytr, Xtr3)
results3 = model.fit()
results3.summary()
```

Out[7]:

OLS Regression Results

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	7.648e+04
Date:	Tue, 09 Apr 2024	**Prob (F-statistic):**	2.51e-310
Time:	12:10:22	**Log-Likelihood:**	51.125
No. Observations:	200	**AIC:**	-92.25
Df Residuals:	195	**BIC:**	-75.76
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	-1298.6590	567.083	-2.290	0.023	-2417.062	-180.256
siri	0.9618	0.027	35.651	0.000	0.909	1.015
sq_siri	-0.0030	0.001	-2.275	0.024	-0.006	-0.000
inv_density	706.0142	305.636	2.310	0.022	103.237	1308.791
density	598.1825	262.950	2.275	0.024	79.592	1116.773

Omnibus:	138.737	**Durbin-Watson:**	1.955
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	12413.875
Skew:	1.696	**Prob(JB):**	0.00
Kurtosis:	41.447	**Cond. No.**	2.68e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.68e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [8]:
```
# predicting the test
new_predictions = results3.get_prediction(Xts3);
y_hat3 = new_predictions.predicted_mean
e3 = np.linalg.norm(yts- y_hat3)
print(e3)
```

```
0.5633616703887403
```

part (i): as we can see model 3 presents the lowest test error and also is the smallest model, so we pick model 3.

Question 3

In [2]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
import statsmodels.api as sm
```

In [3]:
```
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize)
from sklearn.metrics import mean_squared_error
```

Parts (a-b)

In [4]:
```
# reading the CSV file
df = pd.read_csv('fat.csv')
exdf = df.copy() # copying the dataframe
exdf.insert(18, "sq_siri", np.square(df['siri']), True)
exdf.insert(19, "inv_density", 1/df['density'], True)

train = exdf.iloc[0:200,]
test = exdf.iloc[200:,]
terms = exdf.columns.drop('brozek')

# constructing the test and training data
Xtr = MS(terms).fit_transform(train)
Xts = MS(terms).fit_transform(test)
ytr = train['brozek']
yts = test['brozek']
print(Xtr.shape)
print(Xts.shape)
model = sm.OLS(ytr, Xtr)
results = model.fit()
results.summary()
```

```
(200, 20)
(52, 20)
```

Out [4]:

OLS Regression Results

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.703e+04
Date:	Tue, 09 Apr 2024	**Prob (F-statistic):**	6.09e-282
Time:	13:38:25	**Log-Likelihood:**	64.717
No. Observations:	200	**AIC:**	-89.43
Df Residuals:	180	**BIC:**	-23.47
Df Model:	19		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
intercept	-950.8057	580.610	-1.638	0.103	-2096.483	194.871
siri	0.9305	0.029	32.063	0.000	0.873	0.988
density	436.7337	269.081	1.623	0.106	-94.225	967.693
age	-0.0007	0.002	-0.414	0.680	-0.004	0.003
weight	0.0162	0.006	2.809	0.006	0.005	0.028
height	-0.0005	0.005	-0.096	0.924	-0.010	0.009
adipos	-0.0235	0.016	-1.505	0.134	-0.054	0.007
free	-0.0204	0.007	-2.780	0.006	-0.035	-0.006
neck	-0.0018	0.011	-0.163	0.870	-0.024	0.020
chest	0.0056	0.005	1.026	0.306	-0.005	0.016
abdom	-0.0006	0.005	-0.109	0.913	-0.011	0.010
hip	0.0008	0.008	0.106	0.916	-0.014	0.016
thigh	0.0174	0.008	2.264	0.025	0.002	0.033
knee	-0.0290	0.013	-2.296	0.023	-0.054	-0.004
ankle	0.0061	0.010	0.591	0.555	-0.014	0.026
biceps	-0.0169	0.008	-2.011	0.046	-0.034	-0.000
forearm	0.0219	0.010	2.110	0.036	0.001	0.042
wrist	0.0343	0.027	1.270	0.206	-0.019	0.088
sq_siri	-0.0026	0.001	-1.956	0.052	-0.005	2.26e-05
inv_density	518.6225	312.993	1.657	0.099	-98.985	1136.230

Omnibus:	98.592	**Durbin-Watson:**	1.907
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6766.061
Skew:	0.902	**Prob(JB):**	0.00
Kurtosis:	31.437	**Cond. No.**	3.19e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.19e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [5]:

```
# predicting the test
new_predictions = results.get_prediction(Xts);
y_hat = new_predictions.predicted_mean
efull = np.linalg.norm(yts- y_hat)
print(efull)
```

```
0.8565466791374616
```

parts (c-d)

In [7]:
```
seldf = exdf.copy()
terms = seldf.columns.drop('brozek')
for p in range(0,19):
    Xtr2 = MS(terms).fit_transform(seldf.iloc[0:200,])
    model = sm.OLS(ytr, Xtr2)
    results2 = model.fit()
    # finding the maximum p-value other than the intercept
    maxp = results2.pvalues[1:].max()
    if maxp>=0.03:
        # dropping the index corresponding to largest p-value
        terms = terms.drop(results2.pvalues[1:].idxmax())

    else:
        print(results2.summary())
        break


# constructing the test and training data
Xtr = MS(terms).fit_transform(train)
Xts = MS(terms).fit_transform(test)

# predicting the test accuracy
new_predictions2 = results2.get_prediction(Xts);
y_hat2 = new_predictions2.predicted_mean
esel = np.linalg.norm(yts- y_hat2)
print('==============================================================================')
print(esel)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 brozek   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.999
Method:                 Least Squares   F-statistic:                 1.021e+05
Date:                Tue, 09 Apr 2024   Prob (F-statistic):          1.02e-312
Time:                        13:39:38   Log-Likelihood:                 50.772
No. Observations:                 200   AIC:                            -93.54
Df Residuals:                     196   BIC:                            -80.35
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      1.3644      0.245      5.577      0.000       0.882       1.847
siri           0.9252      0.002    447.284      0.000       0.921       0.929
thigh          0.0143      0.005      3.107      0.002       0.005       0.023
knee          -0.0259      0.010     -2.540      0.012      -0.046      -0.006
==============================================================================
Omnibus:                      186.283   Durbin-Watson:                   1.924
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            13933.309
Skew:                           2.955   Prob(JB):                         0.00
Kurtosis:                      43.461   Cond. No.                     1.35e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.35e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
==============================================================================
0.6684465807081803
```

part (e)

Yes, esel < efull, which indicates the process has reduced the overfitting associated with the full model.

part (f)

No, e3 < esel indicates that the model proposed in part (f) of question 2 is better than the model gnerated by the feature selection scheme in terms of test accuracy.