Q2

April 18, 2024

The goal of this question is to calculate body fat using Brozek's equation. In this problem, the first column **brozek** is the response variable, and all other columns are treated as the features (we may not use all the features, please only use the ones stated in each question). To start, read the data file fat.csv in the homework folder (containing 252 samples) and split it into two sets. Set 1 includes the first 200 rows of the data (do not count the row associated with the feature/response names), and set 2, which includes the last 52 rows of the data. Name the first set **train** and the second set **test**.

[1]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
import statsmodels.api as sm
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize)
```

[2]:
```
Fat = pd.read_csv("fat.csv")
train,test = np.split(Fat,[int(200)])
y_train = train['brozek']
X_train = pd.DataFrame({'intercept': np.ones(train.shape[0]),
                  'siri':    train['siri'],
                  'density': train['density'],
                  'age':     train['age'],
                  'weight':  train['weight'],
                  'height':  train['height'],
                  'adipos':  train['adipos'],
                  'free':    train['free'],
                  'neck':    train['neck'],
                  'chest':   train['chest'],
                  'abdom':   train['abdom'],
                  'hip':     train['hip'],
                  'thigh':   train['thigh'],
                  'knee':    train['knee'],
                  'ankle':   train['ankle'],
                  'biceps':  train['biceps'],
                  'forearm': train['forearm'],
                  'wrist':   train['wrist']})
```

(a) As a first modeling attempt, consider a linear model using all the 17 features, that is brozek = β_0 + $\beta_1$siri + ... + β_{17}wrist report the fitted parameters, the 95% confidence interval for each estimated parameter and the p-values. What is the R2 value, and based on that how good do you see the model fitting the training data?

[3]:
```python
firstFittedModel = sm.OLS(y_train, X_train).fit()
firstFittedModel.summary()
```

[3]:

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.859e+04
Date:	Thu, 18 Apr 2024	**Prob (F-statistic):**	7.25e-285
Time:	18:26:29	**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.

(b) Based on α = 0.05 and the calculated p-values, 10 features seem problematic.

feature	p-value
age	0.677
height	0.940
adipos	0.175
neck	0.746
chest	0.529
abdom	0.929
hip	0.620
ankle	0.720
biceps	0.063
wrist	0.214

[4]:
```python
X_test = pd.DataFrame({'intercept': np.ones(test.shape[0]),
                      'siri':    test['siri'],
                      'density': test['density'],
                      'age':     test['age'],
                      'weight':  test['weight'],
                      'height':  test['height'],
                      'adipos':  test['adipos'],
                      'free':    test['free'],
                      'neck':    test['neck'],
                      'chest':   test['chest'],
                      'abdom':   test['abdom'],
                      'hip':     test['hip'],
                      'thigh':   test['thigh'],
                      'knee':    test['knee'],
                      'ankle':   test['ankle'],
                      'biceps':  test['biceps'],
                      'forearm': test['forearm'],
                      'wrist':   test['wrist']})
```

(c) Calculate the prediction error e_1 using your test file.

[5]:
```python
y_actual = np.array(test['brozek'].values)

# Get prediction error and prediction
def get_prediction_error(model, X_test):
    # the actual value from test dataset
    # the value predicted by the full-features model
    y_prediction = np.array(model.predict(X_test))
    return (np.sqrt(sum(np.square(y_actual - y_prediction))), y_prediction)
```

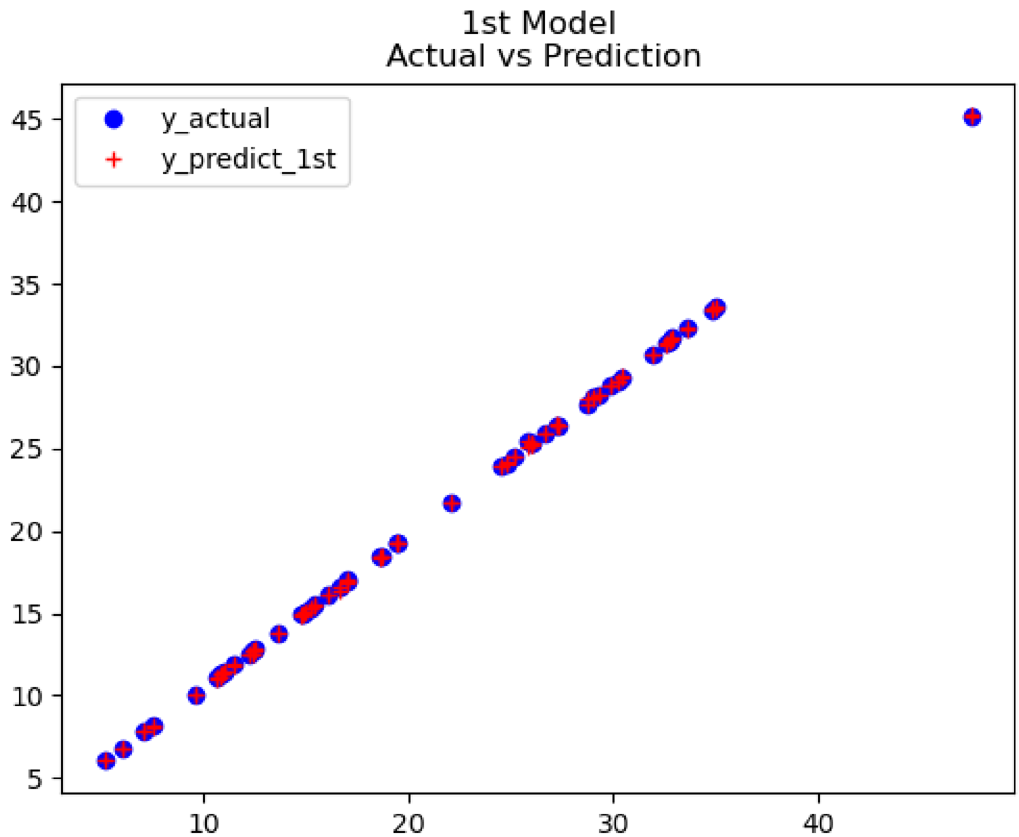
[6]:
```python
# Calculate Prediction Error of the first model
e_1, y_pred_first = get_prediction_error(firstFittedModel, X_test)
print('Prediction Error e_1 =', e_1)
```

```
Prediction Error e_1 = 0.753742369836337
```

[7]:
```
import matplotlib.pyplot as plt

def plot_act_vs_pred(title, label_pred, x_test, y_pred):
    fig, ax = plt.subplots()
    ax.plot(test['siri'], y_actual, "bo", label="y_actual")
    ax.plot(np.hstack((test['siri'], x_test['siri'])), np.hstack((y_actual,␣
 ↪y_pred)), "r+", label=label_pred)
    plt.title(title)
    ax.legend(loc="best")
```

[8]:
```
# Plot Actual vs Prediction of the first model
plot_act_vs_pred('1st Model\n Actual vs Prediction', 'y_predict_1st', X_test,␣
 ↪y_pred_first)
```



(d) Now consider a second model which uses the features indicated in part (a), with the only difference that density is replaced by inverse density. In other words, your model has the the following parametric form:brozek = β_0 + β_1 siri + $\frac{\beta_2}{density}$ + β_3 age + ... + β_{17} wrist Repeat the steps in part (a) and report the values.

[9]:
```
X_train_second = pd.DataFrame({'intercept': np.ones(train.shape[0]),
                  'siri':    train['siri'],
                  'density': 1/train['density'],
                  'age':     train['age'],
                  'weight':  train['weight'],
                  'height':  train['height'],
                  'adipos':  train['adipos'],
                  'free':    train['free'],
                  'neck':    train['neck'],
                  'chest':   train['chest'],
                  'abdom':   train['abdom'],
                  'hip':     train['hip'],
                  'thigh':   train['thigh'],
                  'knee':    train['knee'],
                  'ankle':   train['ankle'],
                  'biceps':  train['biceps'],
                  'forearm': train['forearm'],
                  'wrist':   train['wrist']})
secondFittedModel = sm.OLS(y_train, X_train_second).fit()
secondFittedModel.summary()
```

[9]:

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.852e+04
Date:	Thu, 18 Apr 2024	**Prob (F-statistic):**	1.00e-284
Time:	18:26:29	**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
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

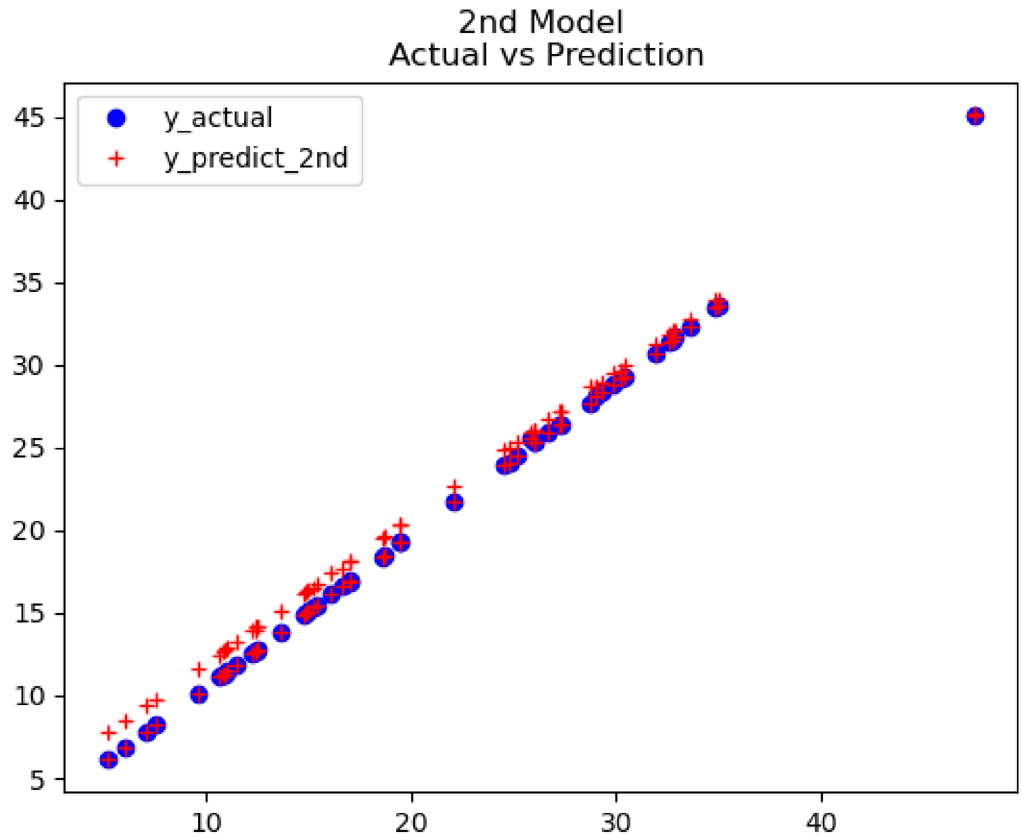
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.58e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

(e) Use a similar formulation in part (c) and calculate the test error (we will call this test error e_2 which corresponds to our second model).

[10]:
```
# Calculate the Prediction Error of the second model
e_2, y_pred_second = get_prediction_error(secondFittedModel, X_test)
print('Second Prediction Error e_2 =', e_2)
```

```
Second Prediction Error e_2 = 7.766138616196848
```

[11]:
```
# Plot Actual vs Prediction of the second model
plot_act_vs_pred('2nd Model\n Actual vs Prediction', 'y_predict_2nd', X_test,␣
↪y_pred_second)
```



(f) Now consider a third model which only uses siri and density as the features, but follows a para- metric formulation as: brozek = β_0 + β_1 siri + β_2 $siri^2$ + $\frac{\beta_3}{density}$ + β_4 density Repeat the steps in part (a) and report the values.

[12]:
```
X_train_third = pd.DataFrame({'intercept': np.ones(train.shape[0]),
                    'siri':        train['siri'],
                    'siri_square': train['siri'] ** 2,
                    'inv_density': 1/train['density'],
                    'density':     train['density']})

thirdFittedModel = sm.OLS(y_train, X_train_third).fit()
thirdFittedModel.summary()
```

[12]:

Dep. Variable:	brozek	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	7.648e+04
Date:	Thu, 18 Apr 2024	**Prob (F-statistic):**	2.51e-310
Time:	18:26:30	**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
siri_square	-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.

(g) Based on = 0.05 and the calculated p-values, which features seem problematic?

All the p-values of the features are less than 0.05

feature	p-value
intercept	0.023
siri	0.00
$siri^2$	0.024
$\frac{1}{density}$	0.022
density	0.024

(h) Repeat part (c) for this model and call the error e_3.

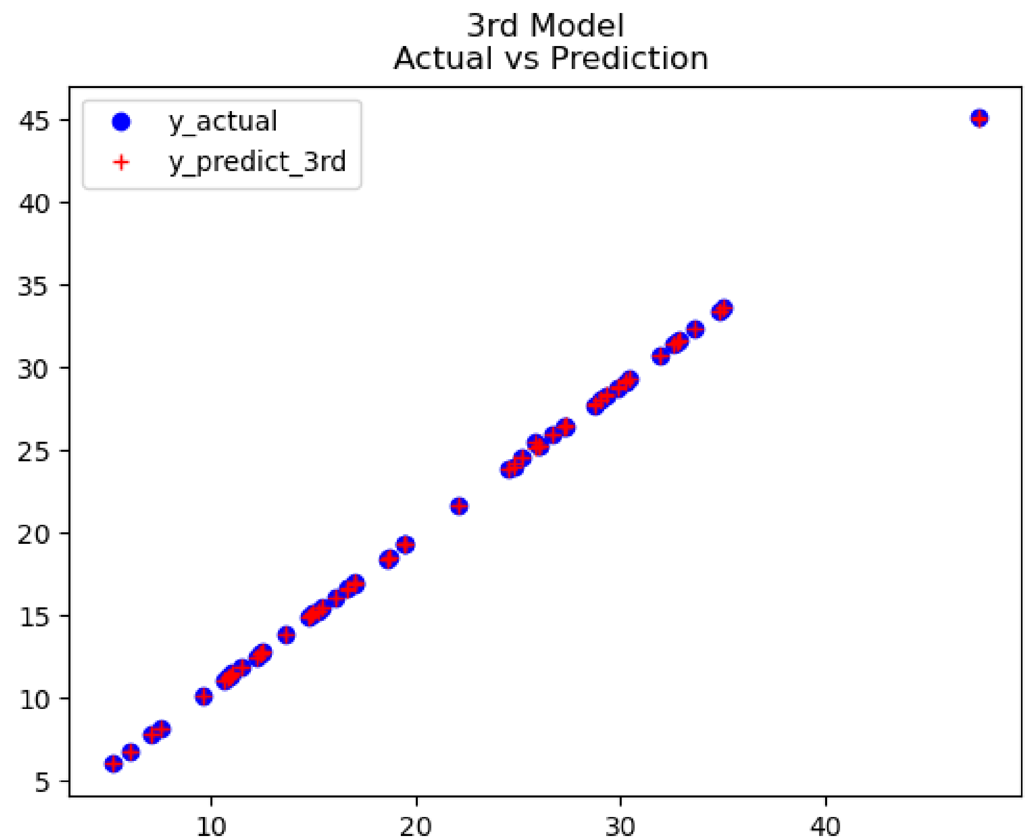
[13]:
```python
X_test_third = pd.DataFrame({'intercept': np.ones(test.shape[0]),
                    'siri':        test['siri'],
                    'siri_square': test['siri'] ** 2,
                    'inv_density': 1/test['density'],
                    'density':     test['density']})

# Calculate the Prediction Error of the third model
e_3, y_pred_third = get_prediction_error(thirdFittedModel, X_test_third)
```

```
print('Second Prediction Error e_3 =', e_3)
```

```
Second Prediction Error e_3 = 0.563361670359713
```

[14]:
```
# Plot Actual vs Prediction of the second model
plot_act_vs_pred('3rd Model\n Actual vs Prediction', 'y_predict_3rd',␣
 ↪X_test_third, y_pred_third)
```



(i) Based on the values e_1, e_2 and e_3, and the model formulations, which model would you pick and why (state two reasons)?

I will pick the third model. (1) It has the lowest prediction error *0.563361670359713* so it has the highest prediction accuracy.(2) The R-squared values of the three models are the same but the thrid model has the lowest p-values *2.51e-310*.