hw3nk

September 19, 2024

[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

[3]:
```
#Question 5 - 6.27
X = np.array([
    [1, 7, 33],
    [1, 4, 41],
    [1, 16, 7],
    [1, 3, 49],
    [1, 21, 5],
    [1, 8, 31]
])

y = np.array([42, 33, 75, 28, 91, 55])

#a
XTX_inv = np.linalg.inv(X.T @ X)
b = XTX_inv @ X.T @ y

print("(a) b:")
print(b)

#b
y_hat = X @ b
e = y - y_hat

print("\n(b) e:")
print(e)

#c
H = X @ XTX_inv @ X.T

print("\n(c) H:")
print(H)
```

```python
#d
SSR = e.T @ e

print("\n(d) SSR:")
print(SSR)

#e
n, p = X.shape
s2 = SSR / (n - p + 1)
cov_b = s2 * XTX_inv

print("\n(e) Estimate of sigma^2:")
print(s2)

print("\n(e) cov_b:")
print(cov_b)

#f
X_h = np.array([1, 10, 30])
Y_h_hat = X_h @ b

print("\n(f) Y_h_hat:")
print(Y_h_hat)

#g
s2_Y_h = s2 * (X_h @ XTX_inv @ X_h.T)

print("\n(g) (s^2(Y_h):")
print(s2_Y_h)
```

```
(a) b:
[33.93210327  2.7847614  -0.26441893]

(b) e:
[-2.69960842 -1.22997279 -1.63735316 -1.32985996 -0.08999801  6.98679233]

(c) H:
[[ 0.23143293  0.25167585  0.21178735  0.14886839 -0.05475543  0.21099091]
 [ 0.25167585  0.31240459  0.09437844  0.26627729 -0.14787283  0.22313666]
 [ 0.21178735  0.09437844  0.70442026 -0.31917435  0.10446672  0.20412159]
 [ 0.14886839  0.26627729 -0.31917435  0.61425632  0.14143492  0.14833743]
 [-0.05475543 -0.14787283  0.10446672  0.14143492  0.94039955  0.01632707]
 [ 0.21099091  0.22313666  0.20412159  0.14833743  0.01632707  0.19708635]]

(d) SSR:
62.073538196057626

(e) Estimate of sigma^2:
```

```
15.518384549014407

(e) cov_b:
[[ 5.36603352e+02 -2.56191874e+01 -1.01962028e+01]
 [-2.56191874e+01  1.24624977e+00  4.83050531e-01]
 [-1.01962028e+01  4.83050531e-01  1.96850816e-01]]

(f) Y_h_hat:
53.847149399348694

(g) (s^2(Y_h):
4.068464775116091
```

[6]:
```
#Question 1
insurancedata=pd.read_csv('../data/insurance.csv')
insurancedata.sample(10)
```

[6]:

	age	sex	bmi	children	smoker	region	charges
1328	23	female	24.225	2	no	northeast	22395.74424
670	30	male	31.570	3	no	southeast	4837.58230
524	42	male	26.070	1	yes	southeast	38245.59327
1133	52	female	18.335	0	no	northwest	9991.03765
741	27	male	29.150	0	yes	southeast	18246.49550
606	27	female	25.175	0	no	northeast	3558.62025
347	46	male	33.345	1	no	northeast	8334.45755
624	59	male	28.785	0	no	northwest	12129.61415
92	59	male	29.830	3	yes	northeast	30184.93670
1334	18	female	31.920	0	no	northeast	2205.98080

[8]:
```
insurance=insurancedata[['charges','age','bmi','children']].copy()
insurance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   charges   1338 non-null   float64
 1   age       1338 non-null   int64
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
dtypes: float64(2), int64(2)
memory usage: 41.9 KB
```

[11]:
```
reg = smf.ols('charges~age+bmi+children',data=insurance).fit()

# print(dir(reg)) #members of the object provided by the modelling
```

[12]:
```
reg.summary()
```

[12]:

Dep. Variable:	charges	**R-squared:**	0.120
Model:	OLS	**Adj. R-squared:**	0.118
Method:	Least Squares	**F-statistic:**	60.69
Date:	Thu, 19 Sep 2024	**Prob (F-statistic):**	8.80e-37
Time:	16:47:18	**Log-Likelihood:**	-14392.
No. Observations:	1338	**AIC:**	2.879e+04
Df Residuals:	1334	**BIC:**	2.881e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-6916.2433	1757.480	-3.935	0.000	-1.04e+04	-3468.518
age	239.9945	22.289	10.767	0.000	196.269	283.720
bmi	332.0834	51.310	6.472	0.000	231.425	432.741
children	542.8647	258.241	2.102	0.036	36.261	1049.468

Omnibus:	325.395	**Durbin-Watson:**	2.012
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	603.372
Skew:	1.520	**Prob(JB):**	9.54e-132
Kurtosis:	4.255	**Cond. No.**	290.

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

[13]:
```
sm.stats.anova_lm(reg,typ=1)
```

[13]:

	df	sum_sq	mean_sq	F	PR(>F)
age	1.0	1.753019e+10	1.753019e+10	135.546341	6.627851e-30
bmi	1.0	5.446449e+09	5.446449e+09	42.112843	1.211545e-10
children	1.0	5.715190e+08	5.715190e+08	4.419080	3.572625e-02
Residual	1334.0	1.725261e+11	1.293299e+08	NaN	NaN

[14]:
```
sm.stats.anova_lm(reg,typ=2)
```

[14]:

	sum_sq	df	F	PR(>F)
age	1.499426e+10	1.0	115.938067	5.533923e-26
bmi	5.417280e+09	1.0	41.887301	1.354882e-10
children	5.715190e+08	1.0	4.419080	3.572625e-02
Residual	1.725261e+11	1334.0	NaN	NaN

[15]:
```
#Question 2
column_names = ['Y', 'X1', 'X2', 'X3', 'X4']
propertydata = pd.read_csv('../data/property.txt', delim_whitespace=True,␣
    ↪names=column_names)

propertydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 81 entries, 0 to 80
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Y       81 non-null     float64
 1   X1      81 non-null     int64
 2   X2      81 non-null     float64
 3   X3      81 non-null     float64
 4   X4      81 non-null     int64
dtypes: float64(3), int64(2)
memory usage: 3.3 KB
```

```
/var/folders/86/c2gz31wn29b2r_53d_q3g3hc0000gn/T/ipykernel_66472/396240547.py:2:
FutureWarning: The 'delim_whitespace' keyword in pd.read_csv is deprecated and
will be removed in a future version. Use ``sep='\s+'`` instead
  propertydata = pd.read_csv('../data/property.txt', delim_whitespace=True,
names=column_names)
```

[32]:
```python
#a
print("(a)")
model = smf.ols('Y ~ X1 + X2 + X3 + X4', data=propertydata).fit()
print(model.summary())
```

```
(a)
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.585
Model:                            OLS   Adj. R-squared:                  0.563
Method:                 Least Squares   F-statistic:                     26.76
Date:                Thu, 19 Sep 2024   Prob (F-statistic):           7.27e-14
Time:                        20:27:27   Log-Likelihood:                -122.75
No. Observations:                  81   AIC:                             255.5
Df Residuals:                      76   BIC:                             267.5
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     12.2006      0.578     21.110      0.000      11.049      13.352
X1            -0.1420      0.021     -6.655      0.000      -0.185      -0.100
X2             0.2820      0.063      4.464      0.000       0.156       0.408
X3             0.6193      1.087      0.570      0.570      -1.545       2.784
X4          7.924e-06   1.38e-06      5.722      0.000    5.17e-06    1.07e-05
==============================================================================
Omnibus:                        1.922   Durbin-Watson:                   1.580
Prob(Omnibus):                  0.383   Jarque-Bera (JB):                1.301
Skew:                           0.148   Prob(JB):                        0.522
Kurtosis:                       3.545   Cond. No.                     1.74e+06
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.74e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[34]:
```python
#b
print("(b)")
fitted_values = model.fittedvalues
residuals = model.resid
print("\nFitted Values:")
print(fitted_values.head(6))
print("\nResiduals:")
print(residuals.head(6))
sigma_squared = (residuals ** 2).sum() / (len(propertydata) - len(model.params))
print("\nSigma^2:")
print(sigma_squared)
```

```
(b)

Fitted Values:
0    14.535672
1    13.513806
2    11.091053
3    15.133568
4    13.686716
5    13.687185
dtype: float64

Residuals:
0   -1.035672
1   -1.513806
2   -0.591053
3   -0.133568
4    0.313284
5   -3.187185
dtype: float64

Sigma^2:
1.2925078150380076
```

[35]:
```python
#c
print("(c)")
p_val_b2 = model.pvalues['X2']
beta_2 = model.params['X2']
print("\nPval B2:")
print(p_val_b2)
```

```
print("\nEstimate B2:")
print(beta_2)
```

```
(c)

Pval B2:
2.747396037799102e-05

Estimate B2:
0.28201652995092874
```

[41]:

```
#d
print("(d)")
type1 = sm.stats.anova_lm(model, typ=1)
print("\nType 1 results:")
print(type1)

p_val_t1_b1 = type1.loc['X1', 'PR(>F)']
p_val_t1_b2 = type1.loc['X2', 'PR(>F)']

print("\nPval Beta 1:")
print(p_val_t1_b1)
print("\nPval Beta 2:")
print(p_val_t1_b2)

type2 = sm.stats.anova_lm(model, typ=2)
print("\nType 2 results:")
print(type2)

p_val_t2_b1 = type2.loc['X1', 'PR(>F)']
p_val_t2_b2 = type2.loc['X2', 'PR(>F)']

print("\nPval Beta 1:")
print(p_val_t2_b1)
print("\nPval Beta 2:")
print(p_val_t2_b2)
```

```
(d)

Type 1 results:
            df     sum_sq    mean_sq          F        PR(>F)
X1         1.0  14.818520  14.818520  11.464936  1.125291e-03
X2         1.0  72.802011  72.802011  56.326167  9.699085e-11
X3         1.0   8.381417   8.381417   6.484616  1.290389e-02
X4         1.0  42.324958  42.324958  32.746385  1.975990e-07
Residual  76.0  98.230594   1.292508        NaN           NaN

Pval Beta 1:
```

```
0.0011252906812985895

Pval Beta 2:
9.6990847500436e-11

Type 2 results:
             sum_sq    df          F        PR(>F)
X1        57.242762   1.0  44.288136  3.894322e-09
X2        25.758955   1.0  19.929439  2.747396e-05
X3         0.419746   1.0   0.324753  5.704457e-01
X4        42.324958   1.0  32.746385  1.975990e-07
Residual  98.230594  76.0        NaN           NaN

Pval Beta 1:
3.894321773978801e-09

Pval Beta 2:
2.7473960377990986e-05
```

[39]:

```
#e
print("(e)")
new_property = pd.DataFrame({'X1': [4], 'X2': [10], 'X3': [0.1], 'X4': [80000]})

predictions = model.get_prediction(new_property)
prediction_summary = predictions.summary_frame(alpha=0.10)

print("\nPrediction Interval 90%")
print(prediction_summary[['obs_ci_lower', 'obs_ci_upper']])
```

```
(e)

Prediction Interval 90%
   obs_ci_lower  obs_ci_upper
0     13.228907     17.068083
```