Week_2_Multiple_Regression_GradedAssignment

August 20, 2017

1 Table of Contents

2 Notes

The notebook and other code can be found on my github repo.

3 Week 2: Multiple Regression Graded Assignment

3.1 Questions:

This test exercise is of an applied nature and uses data that are available in the data file `TestExer2-GPA-round2`. The exercise is based on ***Exercise 3.14 of 'Econometric Methods with Applications in Business and Economics'***. The question of interest is whether the study results of students in Economics can be predicted from the scores on entrance tests taken before they start their studies. More precisely, you are asked to investigate whether verbal and mathematical entrance tests predict freshman grades of students in Economics.

A

1. Regress FGPA on a constant and SATV. Report the coefficient of SATV and its standard error and p-value (give your answers with 3 decimals).
2. Determine a 95% confidence interval (with 3 decimals) for the effect on FGPA of an increase by 1 point in SATV.

B

Answer questions a_i and a_2 also for the regression of FGPA on a constant, SATV, SATM, and FEM.

C

Determine the (4 Œ 4) correlation matrix of FGPA, SATV, SATM, and FEM. Use these correlations to explain the differences between the outcomes in parts (A) and (B).

D

1. Perform an `F-test` on the significance (at the 5% level) of the effect of SATV on FGPA, based on the regression in part (b) and another regression. Note: Use the F-test in terms of SSR or R^2 and use 6 decimals in your computations. The relevant critical value is 3.9. 2. Check numerically that $F = t^2$.

TestExer2-GPA Data

Data on student learning of 609 students (dataset 1 in 'Econometric Methods with Applications in Business and Economics').

```
FGPA: Freshman grade point average (scale 0-4)
SATV: Score on SAT Verbal test (scale 0-10)
SATM: Score on SAT Mathematics test (scale 0-10)
FEM: Gender dummy (1 for females, 0 for males)
```

3.1.1 Solution:

Get Data

```
In [1]: import numpy as np
        import pandas as pd

        from sklearn.metrics import r2_score
        from sklearn.linear_model import LinearRegression
        from sklearn.feature_selection import f_regression as fp
```

```
In [2]: file_location="./data/TestExer2-GPA-round2.xls"

        TestExer = pd.read_excel(file_location)
        TestExer.head() # sample data
```

```
Out[2]:    Observation   FGPA  SATM  SATV  FEM
        0            1  2.518   4.0   4.0    1
        1            2  2.326   4.9   3.1    0
        2            3  3.003   4.4   4.0    1
        3            4  2.111   4.9   3.9    0
        4            5  2.145   4.3   4.7    0
```

Part A Solution

Part A Code

```
In [3]: SATV = TestExer.SATV.values.reshape(-1,1) # independent variable
        FGPA = TestExer.FGPA # dependent variable

In [4]: model_a = LinearRegression()
        model_a.fit(X=SATV, y=FGPA)
        print("The coefficient rounded to 3 decimals is: ", round(model_a.coef_[0], 3))
        print("The intercept rounded to 3 decimals is: ", round(model_a.intercept_, 3))
```

```
The coefficient rounded to 3 decimals is:  0.063
The intercept rounded to 3 decimals is:  2.442
```

```
In [5]: F_test, P_value = fp(SATV, FGPA)
        print("The P-Value rounded to 3 decimals is: ", round(P_value[0], 3))
```

```
The P-Value rounded to 3 decimals is:  0.023
```

```
In [6]: FGPA_pred = model_a.predict(SATV)
        SSE_a = ((FGPA - FGPA_pred)**2).sum() # sum of squared error
        s_a = np.sqrt((SSE_a)/(len(FGPA)-2)) # standard error for part a
        s_b_sq = s_a**2 / ((SATV - SATV.mean())**2).sum() # std. error sq of slope
        s_b = np.sqrt(s_b_sq)

In [7]: lower_limit = (model_a.coef_ - 1.96*s_b)
        upper_limit = model_a.coef_ + 1.96*s_b
        conf_interval = [lower_limit[0], upper_limit[0]]
```

Summary Part A

```
In [8]: # Part a - 1: coefficient of SATV and its standard error and p-value
        print("The coefficient rounded to 3 decimals is: ", round(model_a.coef_[0], 3))
        print("The Standard error of Slope rounded to 3 decimals is: ", round(s_b, 3))
        print("The P-Value rounded to 3 decimals is: ", round(P_value[0], 3))

        # Part a - 2: 95% confidence interval (with 3 decimals)
        print("\n")
        print("The 95% confidence interval for effect on FGPA with an increase by 1 point is: "
```

```
The coefficient rounded to 3 decimals is:  0.063
The Standard error of Slope rounded to 3 decimals is:  0.028
The P-Value rounded to 3 decimals is:  0.023


The 95% confidence interval for effect on FGPA with an increase by 1 point is:
 [0.0088651437165872399, 0.11730654703910486]
```

Part B Solution

Part B Code

```
In [9]: X = TestExer[["SATV", "SATM", "FEM"]] # independent variable
        y = TestExer.FGPA # dependent variable

In [10]: model_b = LinearRegression()
         model_b.fit(X, y)
         np.around(model_b.coef_, 3)

Out[10]: array([ 0.014,  0.173,  0.2  ])

In [11]: y_pred = model_b.predict(X)
         SSE_a2 = ((y - y_pred)**2).sum() # sum of squared error
         s_a2 = np.sqrt((SSE_a2)/(len(X)-2)) # standard error
         s_b2_sq = s_a2**2 / ((TestExer.SATV - TestExer.SATV.mean())**2).sum() # SSE SATV
         s_b2 = np.sqrt(s_b2_sq)
         s_b2

Out[11]: 0.026604468488192663

In [12]: lower_limit = (model_b.coef_[0] - 1.96*s_b2)
         upper_limit = model_b.coef_[0] + 1.96*s_b2
         conf_interval = [lower_limit, upper_limit]
         conf_interval

Out[12]: [-0.03798286169651121, 0.066306654777204016]
```

Summary Part B

```
In [13]: # Part a - 1: coefficient of SATV and its standard error
         print("The coefficient rounded to 3 decimals is: ", round(model_b.coef_[0], 3))
         print("The Standard error of Slope rounded to 3 decimals is: ", round(s_b2, 3))

         # Part a - 2: 95% confidence interval (with 3 decimals)
         print("\n")
         print("The 95% confidence interval for effect on FGPA with an increase by 1 point is:
```

```
The coefficient rounded to 3 decimals is:  0.014
The Standard error of Slope rounded to 3 decimals is:  0.027


The 95% confidence interval for effect on FGPA with an increase by 1 point is:
 [-0.03798286169651121, 0.066306654777204016]
```

Note: 0 is included in the 95% confidence limit

Part C Solution

Part C Code

```
In [14]: subset_df = TestExer[["FGPA", "SATV", "SATM", "FEM"]]
         subset_df.corr()
```

```
Out[14]:           FGPA      SATV      SATM       FEM
         FGPA  1.000000  0.092167  0.195040  0.176491
         SATV  0.092167  1.000000  0.287801  0.033577
         SATM  0.195040  0.287801  1.000000 -0.162680
         FEM   0.176491  0.033577 -0.162680  1.000000
```

Summary Part C In general, SATV has significant impact on FGPA when it was the only feature. However, since SATM and SATV are the correlated, the total significance comes from SATM influence. When there was a partial dependence (Case B), we saw that SATV does not have a significant impact.

Now FEM and SATV do not have a significant correlation so it mut be maintained in the model. The effect of SATM can be absorbed by SATV.

Part D Solution

Part D Code Suing the results of **Part B** and inferences of **Part C**, we can create a new model which looks at only `SATM` and `FEM`. We can use that to determine SSR which is defined as the sum of the squared differences between the prediction for each observation and the population. We cab then use the results fo this model to compare against the `model_b`.

```
In [15]: model_h0 = LinearRegression()
         X_h0 = TestExer[["SATM", "FEM"]]
         y_h0 = TestExer.FGPA
```

```
In [16]: model_h0.fit(X_h0, y_h0)
         y_pred_h0 = model_h0.predict(X_h0)

         SSR_h0_sq = (y_h0 - y_pred_h0)**2
         SSR_h0 = SSR_h0_sq.sum()
         r_sq_h0 = r2_score(y_h0, y_pred_h0)
```

```
In [17]: SSR_b_sq = (y - y_pred)**2
         SSR_b = SSR_b_sq.sum()
         r_sq_b = r2_score(y, y_pred)
```

```
In [18]: F = (SSR_h0 - SSR_b) / (SSR_b / 605)
```

Summary Part D

```
In [19]: # Modified Model - SSR and R-squared
         print("The Modified Model SSR rounded to 6 decimals is: ", format(SSR_h0, '.6f'))
         print("The Modified Model R^2 rounded to 6 decimals is: ", format(r_sq_h0, '.6f'))
```

```
# Part B Model - SSR and R-squared
print("\n")
print("The Part-B Model SSR rounded to 6 decimals is: ", format(SSR_b, '.6f'))
print("The Part-B Model R^2 rounded to 6 decimals is: ", format(r_sq_b, '.6f'))

# F Statistic
print("\n")
print("The F statistic rounded to 3 decimals is: ", format(F, '.3f'))
```

```
The Modified Model SSR rounded to 6 decimals is:  118.151224
The Modified Model R^2 rounded to 6 decimals is:  0.082575


The Part-B Model SSR rounded to 6 decimals is:  118.101025
The Part-B Model R^2 rounded to 6 decimals is:  0.082965


The F statistic rounded to 3 decimals is:  0.257
```

Since the value of the *F-statistic* is less than the provided critical value, we can safely conclude that the *Null hypothesis* H_0 is not rejected.