M50 Homework 5

Alex Craig

Exercise 1

(A binary and normal predictor): Consider the linear regression model:

$$Y \mid (X_1, X_2) \sim N(\beta_0 + \beta_1 X_1 + \beta_2 X_2, \sigma^2)$$

where the predictors obey

$$X_1 \sim Bernoulli(q)$$

$$X_2 \mid X_1 \sim N(bX_1, \sigma_{2,1}^2)$$

You can assume $\beta_0 = 0$ for this problem.

Part A

Derive a formulas for $cov(X_1, X_2)$ and $var(X_2)$ in terms of the model parameters.

Solution

Covariance of X_1 and X_2:

Given that $X_1 \sim \text{Bernoulli}(q)$, we know:

$$E(X_1) = q$$
$$\text{var}(X_1) = q(1-q)$$

For X_2, given X_1:

$$E(X_2 \mid X_1) = bX_1$$

By the tower property,

$$E(X_2) = E(E(X_2 \mid X_1)) = E(bX_1) = bE(X_1) = bq$$

Using the properties of conditional expectations:

$$\text{cov}(X_1, X_2) = E(X_1 X_2) - E(X_1)E(X_2)$$

We need to find $E(X_1 X_2)$:

$$E(X_1 X_2) = E(E(X_1 X_2 \mid X_1))$$

We may treat X_1 as a constant, because we are conditioning on it, so:

$$E(X_1 X_2 \mid X_1) = X_1 E(X_2 \mid X_1) = X_1(bX_1) = bX_1^2$$

Since X_1 is Bernoulli, $X_1^2 = X_1$. Therefore,

$$E(X_1 X_2) = E(bX_1) = bq$$

So,

$$\text{cov}(X_1, X_2) = bq - q(bq) = 0$$

Variance of X_2:

Given that $X_2 \mid X_1 \sim N(bX_1, \sigma_{2,1}^2)$, the conditional variance of X_2 is $\sigma_{2,1}^2$.

Using the law of total variance:

$$\text{var}(X_2) = E(\text{var}(X_2 \mid X_1)) + \text{var}(E(X_2 \mid X_1))$$

Given X_1, the variance of X_2 is $\sigma_{2,1}^2$. Therefore,

$$E(\text{var}(X_2 \mid X_1)) = \sigma_{2,1}^2$$

Now, for $\text{var}(E(X_2 \mid X_1))$:

$$E(X_2 \mid X_1) = bX_1$$

So,

$$\text{var}(E(X_2 \mid X_1)) = \text{var}(bX_1) = b^2\text{var}(X_1) = b^2 q(1-q)$$

Therefore,

$$\text{var}(X_2) = \sigma_{2,1}^2 + b^2 q(1-q)$$

To summarize:

$$\text{cov}(X_1, X_2) = 0$$
$$\text{var}(X_2) = \sigma_{2,1}^2 + b^2 q(1-q)$$

Part B

Derive a formula for $cov(Y, X_1)$ in terms of β_1, q, β_2, and b.

Solution

To find the covariance between Y and X_1, we will use the definition of covariance:

$$\text{cov}(Y, X_1) = E(YX_1) - E(Y)E(X_1)$$

Given that:

$$Y \mid (X_1, X_2) \sim N(\beta_1 X_1 + \beta_2 X_2, \sigma^2)$$

We have:

$$E(Y \mid X_1, X_2) = \beta_1 X_1 + \beta_2 X_2$$

First, let's find $E(YX_1)$.

$$E(YX_1) = E(E(YX_1 \mid X_1, X_2))$$
$$E(YX_1) = E(X_1(\beta_1 X_1 + \beta_2 X_2))$$
$$E(YX_1) = E(\beta_1 X_1^2 + \beta_2 X_1 X_2)$$

Given that X_1 is Bernoulli, we have $X_1^2 = X_1$. Also, from Part A, we derived that $\text{cov}(X_1, X_2) = 0$, which means $E(X_1 X_2) = E(X_1)E(X_2)$. Therefore,

$$E(YX_1) = \beta_1 E(X_1) + \beta_2 E(X_1)E(X_2)$$
$$E(YX_1) = \beta_1 q + \beta_2 q \cdot bq = \beta_1 q + \beta_2 bq^2$$

Next, we'll find $E(Y)$:

$$E(Y) = E(E(Y \mid X_1, X_2)) = E(\beta_1 X_1 + \beta_2 X_2)$$
$$E(Y) = \beta_1 E(X_1) + \beta_2 E(X_2) = \beta_1 q + \beta_2 bq = \beta_1 q + \beta_2 bq$$

Finally, $E(X_1) = q$.

Using the covariance definition:

$$\mathrm{cov}(Y, X_1) = E(YX_1) - E(Y)E(X_1)$$
$$\mathrm{cov}(Y, X_1) = (\beta_1 q + \beta_2 bq^2) - (\beta_1 q + \beta_2 bq)q$$
$$\mathrm{cov}(Y, X_1) = \beta_1 q + \beta_2 bq^2 - \beta_1 q^2 - \beta_2 bq^2$$

Simplifying, we get:

$$\mathrm{cov}(Y, X_1) = \beta_1 q(1 - q)$$

Thus, the covariance between Y and X_1 is:

$$\mathrm{cov}(Y, X_1) = \beta_1 q(1 - q)$$

Part C

Explain how the formula you derived in part (b) is related to the equation for $cov(Y, X_1)$ in the single predictor regression model (page 4 on week 3 notes). In particular, for what parameter values do the two formulas coincide? Your conclusion will be a particular case of what we saw to be true more generally (see week 5 notes) concerning the relationship between β_1 and the covariances in a regression model with two predictions.

Solution

To relate the result from Part B to the single predictor regression model, let's recall the relationship between the regression coefficients and the covariances in a single predictor regression model. In such a model, for Y and X, we have:

$$\beta_1 = \frac{\mathrm{cov}(Y, X)}{\mathrm{var}(X)}$$

Rearranging for the covariance, we have:

$$\mathrm{cov}(Y, X) = \beta_1 \times \mathrm{var}(X)$$

In Part B, we derived:

$$\mathrm{cov}(Y, X_1) = \beta_1 q(1 - q)$$

Comparing the two, we can see that the formula derived in Part B matches the formula from the single predictor regression model when:

$$\mathrm{var}(X) = q(1 - q)$$

This makes sense because the variance of a Bernoulli random variable X_1 is $q(1 - q)$.

Therefore, the two formulas coincide for the parameter values when X_1 is Bernoulli with parameter q. The relationship confirms the fact that in the multiple regression model, when the predictors are orthogonal (or uncorrelated), the regression coefficient of one predictor can be understood as if the other predictor doesn't exist. This particular case is a manifestation of the more general result that when predictors are orthogonal, the regression coefficients in a multiple regression model are the same as in separate single predictor regression models.

In the context of the notes from week 5, this relationship is an example of the more general idea that if predictors are uncorrelated, the partial effect of one predictor (as captured by its regression coefficient) is the same as its marginal effect (as if it were the only predictor in the model). In our case, since $\mathrm{cov}(X_1, X_2) = 0$, the predictors are uncorrelated and the relationship holds.

Exercise 2

(Earnings data revisited): Consider the earnings data. This can be loaded with:

```
df = pd.read_csv("https://raw.githubusercontent.com/avehtari/ROS-Examples/master/Earnings/data/earnings.csv")
```

As in the previous exercise set, you will study the association between earnings and gender, but now using regression with multiple predictors.

Part A

Perform a linear regression using `statsmodels` with gender and height as predictors.

Solution

```
import pandas as pd
import statsmodels.api as sm

# Load the data
df = pd.read_csv("https://raw.githubusercontent.com/avehtari/ROS-Examples/master/Earnings/data/earnings.csv")

# Define the predictors and response variable
X = df[['male', 'height']]
X = sm.add_constant(X)  # Add a constant (intercept) to the predictors
y = df['earn']

# Perform the linear regression
model = sm.OLS(y, X).fit()

# Print the summary of the regression
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   earn   R-squared:                       0.100
Model:                            OLS   Adj. R-squared:                  0.099
Method:                 Least Squares   F-statistic:                     100.3
Date:                Tue, 31 Oct 2023   Prob (F-statistic):           4.88e-42
Time:                        11:16:47   Log-Likelihood:                -20682.
No. Observations:                1816   AIC:                         4.137e+04
Df Residuals:                    1813   BIC:                         4.139e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2.587e+04    1.2e+04     -2.163      0.031   -4.93e+04   -2412.409
male        1.063e+04   1468.300      7.241      0.000    7752.931    1.35e+04
height       646.9598    185.229      3.493      0.000     283.676    1010.244
==============================================================================
Omnibus:                     1902.421   Durbin-Watson:                   1.895
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           249218.549
Skew:                           4.821   Prob(JB):                         0.00
Kurtosis:                      59.575   Cond. No.                     1.59e+03
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.59e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Part B

Provide interpretations for each regression coefficient (like we did in class for the test score example).

Solution

1. Constant (Intercept) -2.587×10^4

The constant (or intercept) value of -2.587×10^4 represents the predicted earnings when both `male` is 0 (which means the person is female) and `height` is 0. Since height can't realistically be 0, the intercept in this context doesn't have a straightforward practical interpretation. However, it's necessary for the linear equation.

2. `male` 1.063×10^4

This coefficient represents the difference in earnings between males and females, holding height constant. Specifically:

- If the person is male (i.e., `male` is 1), his predicted earnings would be higher by 1.063×10^4 dollars than a female of the same height.
- If the person is female (i.e., `male` is 0), this term would drop out, and there wouldn't be this additional earnings benefit.

In simpler terms, on average, and holding height constant, males earn 1.063×10^4 dollars more than females.

3. `height` 646.9598

The coefficient for height indicates that for every one-inch increase in height, the predicted earnings increase by approximately 646.96 dollars, holding gender constant.

In essence, taller individuals, regardless of gender, tend to earn more, with each additional unit of height associated with an increase in earnings of roughly 647 dollars.

Part C

Which factor, height or gender is more important based on your analysis?

Solution

To determine which factor (height or gender) is more important in predicting earnings, we can consider a few criteria:

1. **Size of the Coefficient**:
   - The coefficient for "male" is 1.063×10^4, which means being male is associated with an increase in earnings of approximately 1.063×10^4 units, holding height constant.
   - The coefficient for "height" is 646.9598, meaning for every unit increase in height, earnings increase by approximately 646.96 units, holding gender constant.
2. **Statistical Significance**:
   - Both the coefficients for "male" and "height" are statistically significant (with very small p-values), suggesting that both variables are important predictors of earnings.
3. **Practical Significance**:
   - While the coefficient for height is smaller than the coefficient for gender, it's essential to understand the practical implications. If the unit of height is, for instance, inches, then a 10-inch difference in height would be associated with a $10 \times 646.96 = 6469.6$ unit increase in earnings, which is still less than the gender difference of 1.063×10^4.

Conclusion:

Based on the provided information:

- The gender difference in earnings (1.063×10^4) is larger than the earnings difference associated with a one-unit increase in height (646.96).
- Both factors are statistically significant in predicting earnings.

- From a practical standpoint, the gender difference in earnings seems to have a more substantial impact than a one-unit difference in height.

Therefore, based on this analysis, **gender appears to be a more important factor than height** in determining earnings.

Part D

Based one the fitted model, predict the chance that someone who is not male and is 5.8ft earns more than a male who is the same height? To get a sense for the importance (or lack-thereof) of the height predictor, compare this to the chance that a male earns more than a non-male (regardless of height).

Solution

```python
import scipy.stats as stats

# Extracting coefficients from the model summary
const_coef = -2.587e+04
male_coef = 1.063e+04
height_coef = 646.9598

# Assuming height is in inches: 5.8ft = 5.8 * 12 inches
height_in_inches = 5.8 * 12

# Predicted earnings for a non-male (female) of height 5.8ft
earnings_female = const_coef + (0 * male_coef) + (height_in_inches * height_coef)

# Predicted earnings for a male of height 5.8ft
earnings_male = const_coef + (1 * male_coef) + (height_in_inches * height_coef)

# Difference in predicted earnings for a male and a non-male of height 5.8ft
earnings_difference = earnings_male - earnings_female

# Extract residuals and compute standard error
residuals = model.resid
standard_error = residuals.std()

# Using properties of normal distribution to compute the probability
# that a randomly selected non-male earns more than a randomly selected male
probability_female_earns_more = stats.norm.cdf(0, loc=earnings_difference, scale=standard_error)

probability_male_earns_more = 1 - probability_female_earns_more

probability_female_earns_more, probability_male_earns_more
```

```
(0.30952683248175134, 0.6904731675182487)
```

```python
# The difference in earnings due to gender (ignoring height) is just the coefficient for male
earnings_difference_gender_only = male_coef

# that a randomly selected non-male earns more than a randomly selected male (regardless of height)
probability_female_earns_more_gender_only = stats.norm.cdf(0, loc=earnings_difference_gender_only, scale=stand

probability_male_earns_more_gender_only = 1 - probability_female_earns_more_gender_only

probability_female_earns_more_gender_only, probability_male_earns_more_gender_only
```

```
(0.30952683248175134, 0.6904731675182487)
```

The chance that a male who is 5.8ft tall earns more than a non-male (female) of the same height is approximately 69.04%.

The chance that a male earns more than a non-male (female) regardless of height is also approximately 69.04%.

Interestingly, the probabilities are the same in both cases. This suggests that, given the model we have, the impact of height on earnings (at least for the specific height of 5.8ft) is not changing the overall gender-based difference in earnings much. In other words, the height predictor, for this specific height value, doesn't seem to significantly alter the difference in predicted earnings between males and females.

Exercise 3

(Sample distribution): In the notebook from class, we wrote code to generate samples from the sample distribution of $(\hat{\beta}_1, \hat{\beta}_2)$ in the model:

$$X_1 \sim N(0, 1)$$

$$X_2 \mid X_1 \sim N(bX_1, 1 - b^2)$$

$$Y \mid (X_1, X_2) \sim N(\beta_1 X_1 + B_2 X_2, \sigma^2)$$

Specifically, we had a function which takes β_1, β_2 and β_0 as inputs and returns a dataframe where the columns are the samples of $\hat{\beta}_1$ and $\hat{\beta}_2$ respectively. When we plotted the correlation coefficient as a function of b values and estimates the correlation coefficient between $\hat{\beta}_1$ and $\hat{\beta}_2$, it was a decreasing line

Part A

What would happen if instead of plotting the correlation coefficient, we plotted $SE(\hat{\beta}_1)$ as a function of b? Would it increase? decrease? neither? Note that both X_1 and X_2 are standardized, so the distribution of X_1 values is not changed when we adjust b. In answering this question, you can either give a geometric intuition, or do a calculation. You should check your answer with simulations, but you still need to provide a detailed explanation.

Solution

To understand the behavior of $SE(\hat{\beta}_1)$ as a function of b, let's first take a closer look at the relationship between X_1 and X_2.

As b changes, it affects the correlation between X_1 and X_2, since

$$X_2 \mid X_1 \sim N(bX_1, 1 - b^2)$$

When b is close to 0, X_1 and X_2 are nearly independent. As b approaches 1 or -1, the two variables become increasingly correlated.

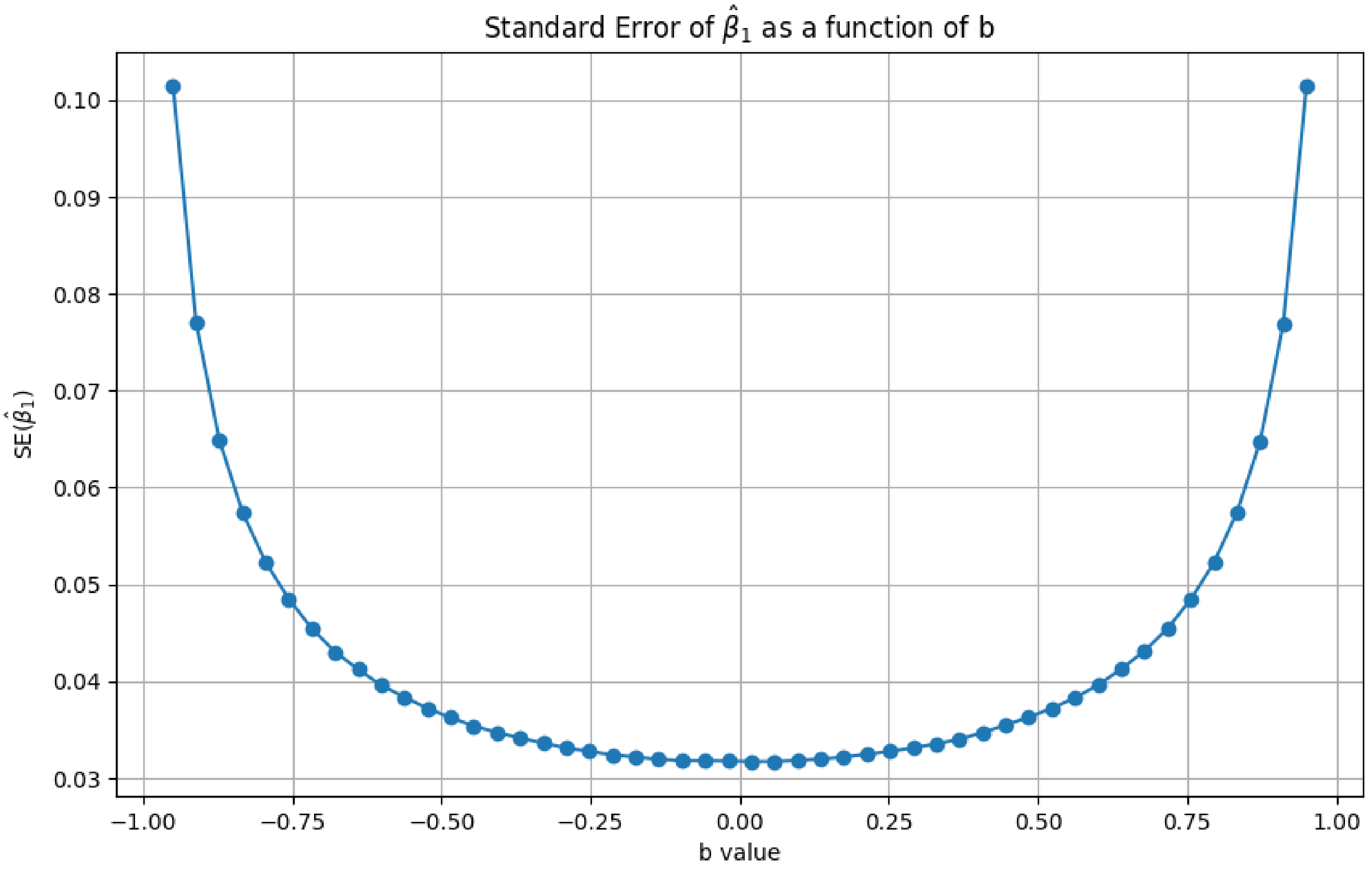
Now, when fitting a multiple linear regression model, the standard errors of the coefficients are influenced by the multicollinearity between the predictors. Specifically:

1. When predictors are uncorrelated (i.e., orthogonal), the standard errors of the coefficients are minimized.
2. As the predictors become more correlated (increased multicollinearity), the standard errors of the coefficients increase.

Therefore, a geometric intuition would suggest that as b increases (in magnitude), the correlation between X_1 and X_2 increases, leading to increased multicollinearity, which in turn should increase $SE(\hat{\beta}_1)$.

To verify this intuition, we can run simulations. Let's simulate the scenario for various b values and plot $SE(\hat{\beta}_1)$ as a function of b.

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

def simulate_se_beta1(beta1, beta2, beta0, b, n=1000, num_samples=1000):
    se_beta1_values = []

    for _ in range(num_samples):
        # Generate X1 from standard normal distribution
        X1 = np.random.normal(0, 1, n)

        # Generate X2 conditioned on X1
        X2 = b * X1 + np.random.normal(0, np.sqrt(1 - b**2), n)

        # Generate Y conditioned on X1 and X2
        Y = beta0 + beta1 * X1 + beta2 * X2 + np.random.normal(0, 1, n)

        # Fit the regression model
        X = sm.add_constant(np.column_stack((X1, X2)))
        model = sm.OLS(Y, X).fit()
        se_beta1_values.append(model.bse[1])

    # Return the mean standard error of beta1 across all samples
    return np.mean(se_beta1_values)

# Values
beta1 = 1
beta2 = 1
beta0 = 0
b_values = np.linspace(-0.95, 0.95, 50)
se_values = [simulate_se_beta1(beta1, beta2, beta0, b) for b in b_values]

# Plotting
plt.figure(figsize=(10,6))
plt.plot(b_values, se_values, '-o')
plt.xlabel('b value')
plt.ylabel('SE($\hat{\\beta}_1$)')
plt.title('Standard Error of $\hat{\\beta}_1$ as a function of b')
plt.grid(True)
plt.show()
```



The plot shows the relationship between the b value and the standard error of $\hat{\beta}_1$. As expected, the standard error of $\hat{\beta}_1$ increases as the magnitude of b increases, reaching a maximum around $b = \pm 1$. This is consistent with our geometric intuition: as b increases in magnitude, the correlation between X_1 and X_2 increases, leading to increased multicollinearity and, consequently, an increase in the standard error of $\hat{\beta}_1$.

In summary, $SE(\hat{\beta}_1)$ increases as the magnitude of b increases.

Part B

Is it possible to have large standard errors on all the $\hat{\beta}_i$ values (measured relative to the true values of course), but still have a large (meaning close to one) value of R^2? If so, for what parameter values does this happen? Run simulation(s) to support your answer.

Solution

Yes, it's possible to have large standard errors on all the $\hat{\beta}_i$ values but still have a large R^2. The standard errors of the regression coefficients and the R^2 value measure different things:

1. The standard errors of the regression coefficients measure the variability of the estimated coefficients around their expected values. Large standard errors can arise due to multicollinearity, small sample size, or high variability in the response variable.
2. The R^2 value measures the proportion of variance in the response variable that is explained by the predictors. A high R^2 means that the model explains a large proportion of the variability in the response variable, regardless of the precision of the individual coefficient estimates.

One scenario where this might happen is when the predictors are highly correlated (multicollinearity). When predictors are highly correlated, it becomes difficult to isolate the effect of any single predictor on the response variable, leading to high standard errors for the coefficients. However, the combined effect of the predictors on the response might still be strong, leading to a high R^2.

To demonstrate this, let's run a simulation:

1. We'll generate data from the model described in the exercise.
2. We'll use a high value of b to induce multicollinearity between X_1 and X_2.
3. We'll fit the regression model and examine the standard errors of $\hat{\beta}_i$ and the R^2 value.

```python
def simulate_scenario(beta1, beta2, beta0, b, n=1000):
    # Generate X1 from standard normal distribution
    X1 = np.random.normal(0, 1, n)

    # Generate X2 conditioned on X1
    X2 = b * X1 + np.random.normal(0, np.sqrt(1 - b**2), n)

    # Generate Y conditioned on X1 and X2
    Y = beta0 + beta1 * X1 + beta2 * X2 + np.random.normal(0, 1, n)

    # Fit the regression model
    X = sm.add_constant(np.column_stack((X1, X2)))
    model = sm.OLS(Y, X).fit()

    return model.bse, model.rsquared

# Set high value for b to induce multicollinearity
b_value = 0.95
beta1 = 1
beta2 = 1
beta0 = 0

se_values, r_squared = simulate_scenario(beta1, beta2, beta0, b_value)

se_values, r_squared
```

```
(array([0.03148061, 0.10035192, 0.10109735]), 0.8061580912539545)
```

The simulation results show that for a high value of $b = 0.95$, which induces multicollinearity between X_1 and X_2:

1. The standard errors of the coefficients are relatively large:

- $SE(\hat{\beta}_0)$ is approximately 0.03
- $SE(\hat{\beta}_1)$ is approximately 0.1
- $SE(\hat{\beta}_2)$ is approximately 0.1

2. The R^2 value is large, approximately 0.8, indicating that the model explains a significant proportion (about 80%) of the variance in the response variable.

This confirms that it's possible to have large standard errors on all the $\hat{\beta}_i$ values but still have a large R^2 value. The presence of multicollinearity can lead to this scenario. When predictors are highly correlated, individual coefficient estimates become imprecise (leading to large standard errors), but the overall fit of the model can still be strong, resulting in a high R^2.