dutta_174_ps1

October 10, 2023

[1]:
```
# Importing necessary libraries for data analysis
import pandas as pd

# Loading the uploaded data into a Pandas DataFrame
df = pd.read_csv('ps1data.csv')

# Displaying the first few rows of the DataFrame for a quick overview
df.head()
```

[1]:

	student_id	treatment	student_age	student_female	student_grade	BL_math_percent	BL_hindi_percent	BL_ses_index	EL_math_percent	EL_hindi_percent	EL_ses_index	add1
0	434	0	10	0	6	0.338745	0.181693	-0.602710	0.580087	0.670878	-0.548381	NaN
1	304	0	10	0	6	0.378779	0.299358	-2.111449	0.359146	0.353417	-2.065724	NaN
2	195	0	10	0	6	0.337856	0.312899	-0.583376	0.336297	0.514733	-0.575815	NaN
3	554	1	10	1	6	0.357186	0.405422	-0.559469	0.458891	0.552098	-0.714794	NaN
4	457	0	10	0	5	0.311081	0.465632	0.678304	0.619417	0.738115	0.758112	NaN

[2]:
```
# Generating summary statistics for the dataset
summary_stats = df.describe()

# Extracting specific information for answering the question
age_range = summary_stats.loc[['min', 'max'], 'student_age']
grade_range = summary_stats.loc[['min', 'max'], 'student_grade']
average_scores_baseline = summary_stats.loc['mean', ['BL_math_percent',␣
↪'BL_hindi_percent']]
```

```
average_scores_endline = summary_stats.loc['mean', ['EL_math_percent',␣
 ↪'EL_hindi_percent']]

summary_stats, age_range, grade_range, average_scores_baseline,␣
 ↪average_scores_endline
```

[2]: (

	student_id	treatment	student_age	student_female	student_grade
count	533.000000	533.000000	533.000000	533.000000	533.000000
mean	310.245779	0.493433	12.412758	0.771107	7.181989
std	177.594732	0.500427	1.356683	0.420515	1.101389
min	1.000000	0.000000	10.000000	0.000000	4.000000
25%	157.000000	0.000000	11.000000	1.000000	6.000000
50%	314.000000	0.000000	12.000000	1.000000	7.000000
75%	459.000000	1.000000	13.000000	1.000000	8.000000
max	619.000000	1.000000	15.000000	1.000000	9.000000

	BL_math_percent	BL_hindi_percent	BL_ses_index	EL_math_percent
count	533.000000	533.000000	533.000000	533.000000
mean	0.316018	0.434963	-0.052643	0.504212
std	0.124357	0.167175	1.657388	0.179286
min	0.009710	0.040948	-5.548198	-0.008850
25%	0.234079	0.309451	-1.264941	0.373951
50%	0.313007	0.418445	0.043285	0.501580
75%	0.388263	0.548276	1.325737	0.636368
max	0.758028	0.923115	4.116564	1.007436

	EL_hindi_percent	EL_ses_index	add1
count	533.000000	533.000000	0.0
mean	0.555292	-0.058806	NaN
std	0.192862	1.660641	NaN
min	0.071790	-5.680749	NaN
25%	0.414091	-1.251783	NaN
50%	0.566267	0.033842	NaN
75%	0.698551	1.300833	NaN
max	1.004655	4.127560	NaN

,
min 10.0
max 15.0
Name: student_age, dtype: float64,
min 4.0
max 9.0
Name: student_grade, dtype: float64,
BL_math_percent 0.316018
BL_hindi_percent 0.434963
Name: mean, dtype: float64,
EL_math_percent 0.504212
EL_hindi_percent 0.555292
Name: mean, dtype: float64)

[3]:
```
import statsmodels.api as sm

# Initialize a dictionary to store results
results_dict = {}

# List of variables to check for balance between treatment and control groups
check_vars = ['student_age', 'student_female', 'BL_ses_index',␣
 ↪'BL_math_percent', 'BL_hindi_percent']

# Run a separate regression for each variable to check for balance
for var in check_vars:
    X = df['treatment']
    X = sm.add_constant(X)  # Adding a constant term for the intercept
    y = df[var]

    model = sm.OLS(y, X).fit()
    results_dict[var] = model.params

# Display results
results_dict
```

[3]:
```
{'student_age': const        12.337037
 treatment     0.153457
 dtype: float64,
 'student_female': const        0.770370
 treatment    0.001493
 dtype: float64,
 'BL_ses_index': const        0.041434
 treatment   -0.190659
 dtype: float64,
 'BL_math_percent': const        0.322679
 treatment   -0.013500
 dtype: float64,
 'BL_hindi_percent': const        0.429797
 treatment    0.010469
 dtype: float64}
```

[4]:
```
# Initialize a dictionary to store results for the treatment effects
treatment_effects = {}

# List of outcome variables to check for treatment effects
outcome_vars = ['EL_math_percent', 'EL_hindi_percent']

# Run a separate regression for each outcome variable to estimate treatment␣
 ↪effects
for var in outcome_vars:
    X = df['treatment']
```

```
    X = sm.add_constant(X)  # Adding a constant term for the intercept
    y = df[var]

    model = sm.OLS(y, X).fit()
    treatment_effects[var] = {'params': model.params, 'pvalues': model.pvalues}

# Display results
treatment_effects
```

[4]:
```
{'EL_math_percent': {'params': const        0.465990
  treatment    0.077461
  dtype: float64,
  'pvalues': const        5.160598e-178
  treatment     4.665160e-07
  dtype: float64},
 'EL_hindi_percent': {'params': const        0.523077
  treatment    0.065289
  dtype: float64,
  'pvalues': const        4.764119e-184
  treatment     8.483582e-05
  dtype: float64}}
```

[5]:
```
# Initialize a dictionary to store results for the treatment effects with␣
 ↪controls
treatment_effects_controls = {}

# Run a separate regression for each outcome variable to estimate treatment␣
 ↪effects, now with controls
for var in outcome_vars:
    X = df[['treatment'] + check_vars]  # Adding control variables
    X = sm.add_constant(X)  # Adding a constant term for the intercept
    y = df[var]

    model = sm.OLS(y, X).fit()
    treatment_effects_controls[var] = {'params': model.params, 'pvalues': model.
 ↪pvalues}

# Display results
treatment_effects_controls
```

[5]:
```
{'EL_math_percent': {'params': const              0.067954
  treatment           0.079136
  student_age         0.009214
  student_female     -0.022815
  BL_ses_index        0.008676
  BL_math_percent     0.409524
  BL_hindi_percent    0.394224
```

```
 dtype: float64,
 'pvalues': const              2.628722e-01
 treatment           4.237725e-10
 student_age         4.636140e-02
 student_female      1.380862e-01
 BL_ses_index        2.728390e-02
 BL_math_percent     4.206404e-13
 BL_hindi_percent    1.960331e-20
 dtype: float64},
'EL_hindi_percent': {'params': const               0.016395
 treatment           0.066230
 student_age         0.013516
 student_female      0.035501
 BL_ses_index        0.024447
 BL_math_percent     0.280761
 BL_hindi_percent    0.514150
 dtype: float64,
 'pvalues': const              7.900800e-01
 treatment           2.244402e-07
 student_age         4.076495e-03
 student_female      2.324027e-02
 BL_ses_index        1.593795e-09
 BL_math_percent     7.019191e-07
 BL_hindi_percent    3.412046e-31
 dtype: float64}}
```

[ ]: