ML_PS1

Clarice Tee

2025-01-21

Table of contents

1. Familarize w data

```python
# Import required libraries
import numpy as np
import os
import statsmodels.api as sm
import pandas as pd
from sklearn.preprocessing import SplineTransformer
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
from pygam import LinearGAM, s, te
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
# Display all columns in pandas
pd.set_option('display.max_columns', None)
```

```python
# Load the data
directory = r"C:\Users\clari\OneDrive\Documents\Machine Learning\mp1"
acs_path = os.path.join(directory, "acs_usa.csv")
acs_data = pd.read_csv(acs_path)
acs_data.describe()
print(acs_data.shape)
```

```
(9048, 25)
```

2.a. Make educdc crowsswalk

```
# Check unique values of EDUCD in the original data
acs_data['EDUCD'].unique()
```

```
array([ 65,  71,  63,  64,  81,  50, 114,  40, 101,   2,  61, 115,  26,
        30, 116,  23,  15,  25,  22,  17,  14,  16,  11,  12],
       dtype=int64)
```

```
# read in crosswalk data
crosswalk_path = os.path.join(directory, "Education-Crosswalk.csv")
crosswalk = pd.read_csv(crosswalk_path)
crosswalk.head()
```

	educd	educdc
0	2	0.0
1	10	0.0
2	11	2.0
3	12	0.0
4	13	2.5

```
# Capitalize the column names in the crosswalk
crosswalk = crosswalk.rename(columns={'educd': 'EDUCD', 'educdc': 'EDUCDC'})
# map the matching values of EDUCD between the two dataframes (i.e. merge on
↪  EDUCD)

acs_data = acs_data.merge(crosswalk, on='EDUCD', how='left')

acs_data.head()
```

	YEAR	SAMPLE	SERIAL	CBSERIAL	HHWT	CLUSTER	STRATA	GQ	PERNU
0	2023	202301	1472	2023010069599	8100.0	2023000014721	120201	4	1
1	2023	202301	2120	2023010102972	648.0	2023000021201	140401	4	1
2	2023	202301	2282	2023010110952	6642.0	2023000022821	240001	4	1
3	2023	202301	3578	2023010171383	43578.0	2023000035781	80001	4	1
4	2023	202301	3902	2023000014636	11988.0	2023000039021	150201	1	1

2.b. Making dummy variables

i. HIGHSCHOOL 12-15 years of schooling = hs diploma

```
acs_data['HSDIP'] = np.where((acs_data['EDUCDC'] == 12) | (acs_data['EDUCDC']
↪  == 13) | (acs_data['EDUCDC'] == 14) | (acs_data['EDUCDC'] == 15),
                             1,
                             0
                             )
```

ii. COLLEGE #if more than 16 years of schooling, has college diploma daw

```
acs_data['COLDIP'] = np.where((acs_data['EDUCDC'] >= 16),
    1,
    0
)
```

iii. & iv. RACE

```
# WHITE = 1, 0 othewise. BLACK =1, 0 otherwise
acs_data['WHITE'] = np.where(acs_data['RACE'] == 1, 1, 0)
acs_data['BLACK'] = np.where(acs_data['RACE'] == 2, 1, 0)
```

v. HISPANIC

```
acs_data['HISPANIC'] = np.where((acs_data['HISPAN'] == 1) | (acs_data['HISPAN']
↪  == 2) | (acs_data['HISPAN'] == 3) | (acs_data['HISPAN'] == 4),
    1,
    0
)
```

vi. MARRIED

```
acs_data['MAR'] = np.where(
    (acs_data['MARST'] == 1) | (acs_data['MARST'] == 2),
    1,
    0
)
```

vii. FEMALE

```
acs_data['SEX'].unique()
acs_data['FEMALE'] = np.where(acs_data['SEX'] == 2, 1, 0)
```

viii. VET (1 = VET, 0 = NOT)

```
acs_data['VET'] = np.where(acs_data['VETSTAT'] == 2, 1, 0)
```

2.c.Interaction term Create an interaction between each of the education dummy variables (hsdip and coldip) and the continuous measure of education (educdc

```
acs_data['HSDIP_EDUCDC'] = acs_data['HSDIP'] * acs_data['EDUCDC']
acs_data['COLDIP_EDUCDC'] = acs_data['COLDIP'] * acs_data['EDUCDC']
```

2.d. Create vairables

Age squared

```
# creating new variable "AGEQ" in the data, which is the square of AGE
acs_data['AGEQ'] = np.power(acs_data['AGE'], 2)
```

The natural log of incwage

```
# We first drop those with 0 income. This is because ln(0) is undefined
acs_data = acs_data[acs_data.INCWAGE != 0]
# creating new variable "INCWAGE_log" in the data, the natural log of INCWAGE
acs_data['LNINCWAGE'] = np.log(acs_data['INCWAGE'])
```

```
acs_data[['INCWAGE','LNINCWAGE']].head(10)
```

	INCWAGE	LNINCWAGE
0	4500	8.411833
1	5500	8.612503
2	24000	10.085809
3	26000	10.165852
4	40000	10.596635
5	70000	11.156251
6	60000	11.002100
7	30000	10.308953
8	5000	8.517193
9	118000	11.678440

0.1 Data Analysis Questions

4.1 Compute descriptive (summary) statistics for the following variables: year, incwage, lnincwage, educdc, f emale, age, ageq, white, black, hispanic, married, nchild, vet, hsdip, coldip, and the interaction terms. In other words, compute sample means, standard deviations, etc.

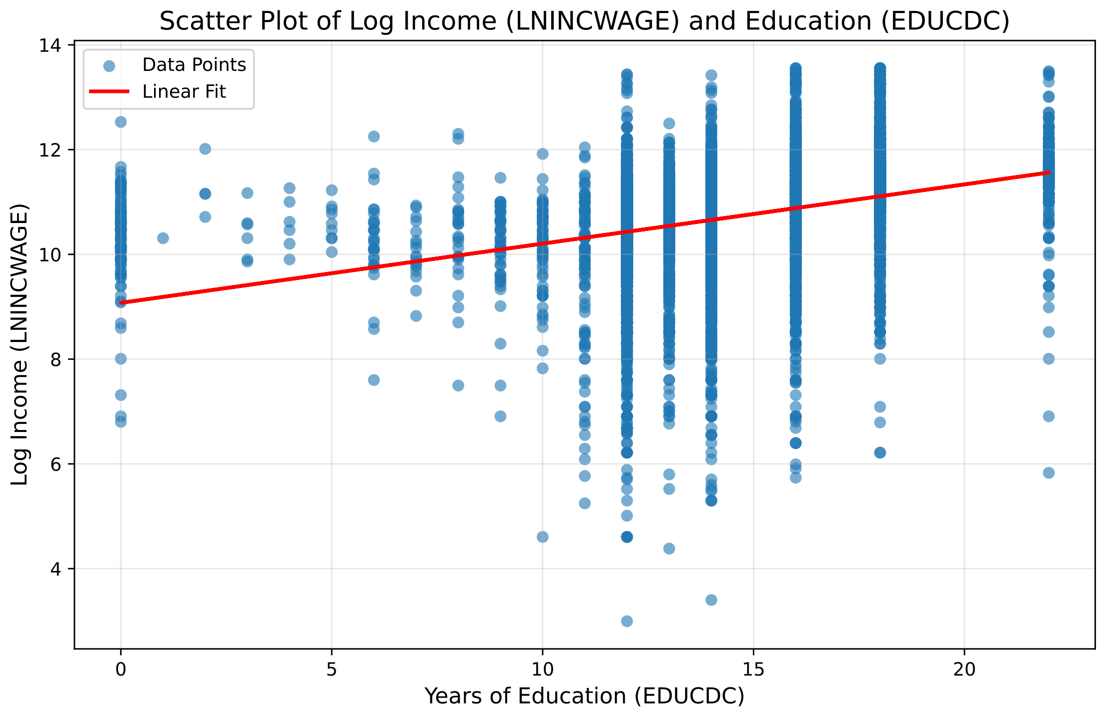
```python
def acs_summary_statistics(data, variables):

    acs_summary_stats = []

    for var in variables:
        if var in data.columns:  # Ensure the variable exists in the DataFrame
            stats = {
                'Variable': var,
                'Mean': data[var].mean(),
                'Std Dev': data[var].std(),
                'Min': data[var].min(),
                'Max': data[var].max(),
                'Count': data[var].count()
            }
            acs_summary_stats.append(stats)

    # Convert summary stats to a DataFrame
    return pd.DataFrame(acs_summary_stats)


# List of variables to compute summary statistics for
variables = [
    'YEAR', 'INCWAGE', 'LNINCWAGE', 'EDUCDC', 'FEMALE', 'AGE', 'AGEQ',
    'WHITE', 'BLACK', 'HISPANIC', 'MAR', 'NCHILD', 'VET',
    'HSDIP', 'COLDIP', 'HSDIP_EDUCDC', 'COLDIP_EDUCDC'
]

# Call function
acs_summary_stats = acs_summary_statistics(acs_data, variables)

# Display the results with formatting
pd.set_option('display.float_format', lambda x: '%.3f' % x)
print(acs_summary_stats.to_string(index=False))
```

```
 Variable      Mean   Std Dev      Min        Max  Count
     YEAR  2023.000     0.000 2023.000   2023.000   8610
  INCWAGE 70235.557 82434.439   20.000 770000.000   8610
LNINCWAGE    10.693     1.079    2.996     13.554   8610
   EDUCDC    14.310     3.054    0.000     22.000   8610
   FEMALE     0.488     0.500    0.000      1.000   8610
```

```
          AGE     41.627     13.291    18.000     65.000    8610
         AGEQ   1909.434   1118.953   324.000   4225.000    8610
        WHITE      0.663      0.473     0.000      1.000    8610
        BLACK      0.082      0.275     0.000      1.000    8610
     HISPANIC      0.163      0.370     0.000      1.000    8610
          MAR      0.557      0.497     0.000      1.000    8610
       NCHILD      0.780      1.097     0.000      6.000    8610
          VET      0.041      0.198     0.000      1.000    8610
        HSDIP      0.537      0.499     0.000      1.000    8610
       COLDIP      0.415      0.493     0.000      1.000    8610
 HSDIP_EDUCDC      6.939      6.484     0.000     14.000    8610
COLDIP_EDUCDC      7.060      8.435     0.000     22.000    8610
```

4.2 Scatter plot ln (incwage) and EDUCDC. Include a linear fit line. Be sure to label all axes and include an informative title.

```python
# Extract data for the scatter plot
X = acs_data[['EDUCDC']].values
y = acs_data['LNINCWAGE'].values
# Fit a linear regression model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

# Create a figure and axis object
fig, ax = plt.subplots(figsize=(10, 6))

# Scatterplot of education vs. log income
ax.scatter(acs_data['EDUCDC'], acs_data['LNINCWAGE'],
           alpha=0.6, s=30, label='Data Points')

# Linear fit line
ax.plot(acs_data['EDUCDC'], y_pred, color="red",
        linewidth=2, label="Linear Fit")

# Setting the title, legend, and labels
ax.set_title(
    'Scatter Plot of Log Income (LNINCWAGE) and Education (EDUCDC)',
     ↪  fontsize=14)
ax.set_xlabel('Years of Education (EDUCDC)', fontsize=12)
ax.set_ylabel('Log Income (LNINCWAGE)', fontsize=12)
ax.legend()
ax.grid(alpha=0.3)


plt.show()
```



4.3 LNINCWAGE MODEL

```
import statsmodels.formula.api as smf

LNINCWAGE_reg = smf.ols(
    'LNINCWAGE ~ EDUCDC + FEMALE + AGE + AGEQ + WHITE + BLACK + HISPANIC+ MAR +
     ↪  NCHILD + VET', data=acs_data).fit()
print(LNINCWAGE_reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              LNINCWAGE   R-squared:
0.282
Model:                            OLS   Adj. R-squared:
0.281
Method:                 Least Squares   F-statistic:
337.3
Date:                Thu, 30 Jan 2025   Prob (F-statistic):
0.00
Time:                        22:58:05   Log-Likelihood:
-11445.
No. Observations:                8610   AIC:
2.291e+04
```

```
Df Residuals:                    8599   BIC:
2.299e+04
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
                 0.975]
------------------------------------------------------------------------------
Intercept      6.1306      0.116     52.856      0.000       5.903
6.358
EDUCDC         0.0994      0.003     29.325      0.000       0.093
0.106
FEMALE        -0.3832      0.020    -19.100      0.000      -0.423
-0.344
AGE            0.1493      0.006     26.016      0.000       0.138
0.161
AGEQ          -0.0016   6.76e-05    -23.066      0.000      -0.002
-0.001
WHITE         -0.0070      0.028     -0.253      0.801      -0.061
0.047
BLACK         -0.1202      0.043     -2.791      0.005      -0.205
-0.036
HISPANIC      -0.0211      0.033     -0.640      0.522      -0.086
0.043
MAR            0.2053      0.023      8.867      0.000       0.160
0.251
NCHILD        -0.0082      0.010     -0.794      0.427      -0.028
0.012
VET           -0.0132      0.050     -0.261      0.794      -0.112
0.086
==============================================================================
Omnibus:                     2566.758   Durbin-Watson:
1.845
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
11879.119
Skew:                          -1.379   Prob(JB):
0.00
Kurtosis:                       8.050   Cond. No.
2.62e+04
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 2.62e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

4.3.a The model explains 28.1% of the variation in log wages (adjusted r2 value)

4.3.b. (b) What is the return to an additional year of education?11 Is this statistically significant?Is it practically significant? Briefly explain. note that your answer should be in terms of wages, not log wages.

```python
beta_EDUCDC = LNINCWAGE_reg.params.iloc[1]

# Convert
change_wages_EDUCDC = (np.exp(beta_EDUCDC) - 1) * 100

mean_wage = acs_data['INCWAGE'].mean()  # Calculate mean directly from the data

# Calculate dollar change
dollar_change_EDUCDC = mean_wage * (change_wages_EDUCDC / 100)

print("Additional year of education means a 10.45 percent increase in wages.
 ↪  Dollar change in wages for an additional year of education is:",
 ↪  dollar_change_EDUCDC)
```

```
Additional year of education means a 10.45 percent increase in wages.
Dollar change in wages for an additional year of education is:
7341.363608190367
```

Based on the p-value, it is statistically significant at the 1% level.

4.3.c.

Here, we need to take the derivative of the model w rt AGE and equate it to 0. We are left with Beta(3) + 2Beta(4)AGE

```python
# Getting the coefficients for AGE and AGE^2
beta_AGE = LNINCWAGE_reg.params.iloc[3]
beta_AGE2 = LNINCWAGE_reg.params.iloc[4]

# Calculate the age at which the highest wage occurs
max_age = -beta_AGE / (2 * beta_AGE2)
print(max_age)
```

```
47.861998868799084
```

We must check whether this is a local max or minumum by taking the second derivative wrt age, leaving us w 2Beta(4).

```
second_deriv = (2 * beta_AGE2)
print(second_deriv)
```

```
-0.0031195396677629757
```

Now that we know it is negative, we know this is the maximum. therefore:

```
print(f"The age at which the model predicts the highest wage is approximately
↪  48 years.")
```

```
The age at which the model predicts the highest wage is approximately 48
years.
```

4.3.d.

```
# Getting the Coefficient for FEMALE
beta_female = LNINCWAGE_reg.params.iloc[2]

# Convert log-wage to percentage change
percentage_change_female = (np.exp(beta_female) - 1) * 100

print(f"Being female is associated with a {percentage_change_female:.1f}%
↪  change in wages.")
```

```
Being female is associated with a -31.8% change in wages.
```

The model predicts that being a women means that, all else equal, wages are lower by about 31.8%. This makes sense because of gender discrimination against women, where their work is undervalued relative to men or they may take lower paying jobs due to having to work part-time to take on child rearing work.

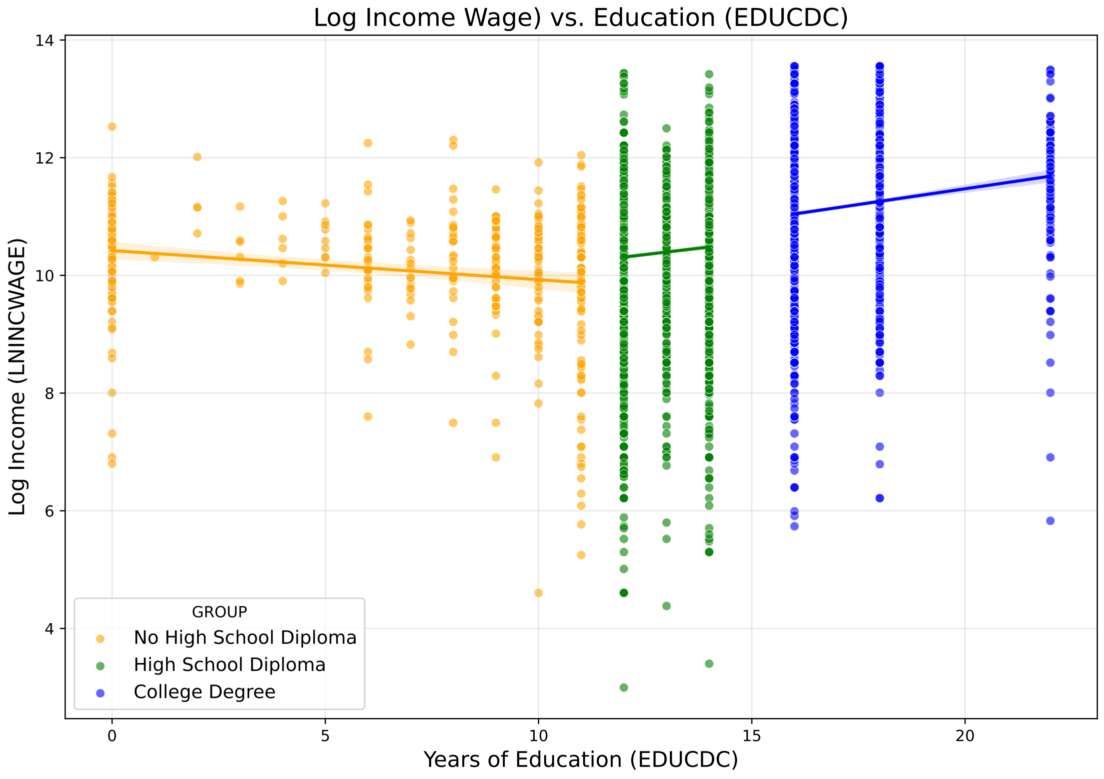
4.3.e. All else equal, white individuals have approximately a .7% decrease in their wages compared to other races, although, this is not significant because we cannot even reject the null hypothesis at a significance level of 10%.

All else equal, black individuals have 12.02% less than other races and this proves to be a significant value at the 1% level.

4.4

```python
## Create degree groups
acs_data['GROUP'] = 'No High School Diploma'  # Default group
acs_data.loc[(acs_data['HSDIP'] == 1) & (acs_data['COLDIP'] == 0), 'GROUP'] =
 ↪  'High School Diploma'
acs_data.loc[(acs_data['COLDIP'] == 1), 'GROUP'] = 'College Degree'

# Create the plot
plt.figure(figsize=(12, 8))

# Colors per grp
group_colors = {
    'No High School Diploma': 'orange',
    'High School Diploma': 'green',
    'College Degree': 'blue'
}

# Plot scatter points and regression lines for each group
for group, color in group_colors.items():
    subset = acs_data[acs_data['GROUP'] == group]  # Filter data for each group

    # Scatterplot for this group
    sns.scatterplot(
        x='EDUCDC',
        y='LNINCWAGE',
        data=subset,
        color=color,
        alpha=0.6,
        label=group
    )

    # Regression line for this group
    sns.regplot(
        x='EDUCDC',
        y='LNINCWAGE',
        data=subset,
        scatter=False,
        color=color,
        line_kws={'linewidth': 2},
        truncate=True
    )

# Add labels, title, and legend
plt.title('Log Income Wage) vs. Education (EDUCDC)', fontsize=16)
plt.xlabel('Years of Education (EDUCDC)', fontsize=14)
plt.ylabel('Log Income (LNINCWAGE)', fontsize=14)  # Fix label
plt.legend(title='GROUP', fontsize=12)
plt.grid(alpha=0.3)
```

```python
# Show the plot
plt.show()
```



4.5a LNINCNWAGES = beta0 + beta1(hsdip) + beta2(coldip) + beta3(educdc × hsdip) + beta4(educdc × coldip) + beta5(age) + beta6(age^2) + beta7(female) + beta8(white) + beta9(black) + beta10(hispanic) + beta11(married) + beta12(nchild) + beta13(vet) + epsilon

Using theory, intuition, and/or common sense, explain/justify why you think your model is the best possible representation of the way the world works (in other words, why you think you are correctly modeling f (X) but not over-fitting e).

By now including the dummy variables for HSDIP, COLDIP and their interaction terms with EDUCDC, the regression model can show differential returns to education based on educational attainment. We increase flexibbility of the model. Now, there can be different slopes for the relationship between educational attainment and wage. It shows us potential “sheepskin effects”, so we can see the added value of completing a HS or College–the degrees have a signalling effect that help employers quickly screen competence. https://www.econlib.org/archives/2012/01/the_present_val.html

4.5b

```python
NEW_reg = smf.ols(
    'LNINCWAGE ~ HSDIP + COLDIP + HSDIP_EDUCDC + COLDIP_EDUCDC + FEMALE + AGE +
     ↪  AGEQ + WHITE + BLACK + HISPANIC+ MAR + NCHILD + VET',
     ↪  data=acs_data).fit()
print(NEW_reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              LNINCWAGE   R-squared:
0.306
Model:                            OLS   Adj. R-squared:
0.305
Method:                 Least Squares   F-statistic:
292.0
Date:                Thu, 30 Jan 2025   Prob (F-statistic):
0.00
Time:                        22:58:09   Log-Likelihood:
-11296.
No. Observations:                8610   AIC:
2.262e+04
Df Residuals:                    8596   BIC:
2.272e+04
Df Model:                          13
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025
0.975]
---------------------------------------------------------------------------------
Intercept         7.1693      0.118     60.790      0.000       6.938
7.401
HSDIP            -0.7122      0.189     -3.761      0.000      -1.083
-0.341
COLDIP           -0.1929      0.177     -1.090      0.276      -0.540
0.154
HSDIP_EDUCDC      0.0833      0.014      5.855      0.000       0.055
0.111
COLDIP_EDUCDC     0.0708      0.010      7.063      0.000       0.051
0.090
FEMALE           -0.3967      0.020    -20.056      0.000      -0.435
-0.358
AGE               0.1368      0.006     24.069      0.000       0.126
0.148
```

```
AGEQ              -0.0014    6.7e-05    -21.149      0.000      -0.002
-0.001
WHITE              0.0238      0.027      0.875      0.382      -0.030
0.077
BLACK             -0.0745      0.042     -1.756      0.079      -0.158
0.009
HISPANIC          -0.0090      0.032     -0.278      0.781      -0.073
0.055
MAR                0.1833      0.023      8.039      0.000       0.139
0.228
NCHILD            -0.0029      0.010     -0.290      0.772      -0.023
0.017
VET                0.0199      0.050      0.399      0.690      -0.078
0.117
==============================================================================
Omnibus:                     2730.221   Durbin-Watson:
1.856
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
13032.061
Skew:                          -1.465   Prob(JB):
0.00
Kurtosis:                       8.267   Cond. No.
4.41e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 4.41e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

REsults: The following variables are statistically significant at the 10% level: HSDIP, HSDIP_EDUCDC, COLDIP_EDUCDC, FEMALE, AGE, AGEQ, BLACK.

Importantly, our interaction terms proved to be statistically significant, confirming the sheepskin effect or the idea that an HS degree and a college degree bumps up wages.

4.5c

```python
# 22 year old, female, not white, black or hispanic. Not married, no children,
↪  not a veteran, HSDIP.
# vs all else equal, but COLDIP
female_1 = acs_data.loc[
    (acs_data['AGE'] == 22) & (acs_data['FEMALE'] == 1) & (acs_data['WHITE'] ==
↪   0) & (
        acs_data['BLACK'] == 0) & (acs_data['NCHILD'] == 0) & (acs_data['VET']
 ↪   == 0) & (acs_data['MAR'] == 0) & (acs_data['HSDIP'] == 1) &
 ↪   (acs_data['COLDIP'] == 0) & (acs_data['HSDIP_EDUCDC'] == 12)&
 ↪   (acs_data['COLDIP_EDUCDC'] == 0)
]
# Expectation
female_1_prediction = LNINCWAGE_reg.get_prediction(female_1)
# Display prediction summary
prediction_summary = female_1_prediction.summary_frame(alpha=0.05)
print(prediction_summary)

# Convert log wages to actual wages
predicted_wage = np.exp(prediction_summary['mean'][0])
print(f"Predicted wage with a high school diploma: ${predicted_wage:.2f}")

# alpha : significance level for the confidence interval.
# Fix alpha to 0.05 so that the confidence interval should have 95% coverage.
```

```
    mean  mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower
    obs_ci_upper
0   9.470    0.035          9.402          9.539         7.675
11.265
1   9.470    0.035          9.402          9.539         7.675
11.265
2   9.449    0.033          9.385          9.513         7.655
11.244
3   9.449    0.033          9.385          9.513         7.655
11.244
4   9.470    0.035          9.402          9.539         7.675
11.265
5   9.449    0.033          9.385          9.513         7.655
11.244
6   9.449    0.033          9.385          9.513         7.655
11.244
7   9.449    0.033          9.385          9.513         7.655
11.244
8   9.470    0.035          9.402          9.539         7.675
11.265
9   9.449    0.033          9.385          9.513         7.655
11.244
10 9.449    0.033          9.385          9.513         7.655
11.244
11 9.449    0.033          9.385          9.513         7.655
11.244
```

```
12 9.470    0.035          9.402          9.539         7.675
11.265
Predicted wage with a high school diploma: $12966.88
```

COLDIP

```
# 22 year old, female, not white, black or hispanic. Not married, no children,
 ↪  not a veteran, HSDIP.
# vs all else equal, but COLDIP
female_2 = acs_data.loc[
    (acs_data['AGE'] == 22) & (acs_data['FEMALE'] == 1) & (acs_data['WHITE'] ==
 ↪  0) & (
        acs_data['BLACK'] == 0) & (acs_data['NCHILD'] == 0) & (acs_data['VET']
 ↪  == 0) & (acs_data['MAR'] == 0) & (acs_data['HSDIP'] == 0) &
 ↪  (acs_data['COLDIP'] == 1) & (acs_data['HSDIP_EDUCDC'] == 0)&
 ↪  (acs_data['COLDIP_EDUCDC'] == 16)
]
# Expectation
female_2_prediction = LNINCWAGE_reg.get_prediction(female_2)
# Display prediction summary
prediction_summary_2 = female_2_prediction.summary_frame(alpha=0.05)
print(prediction_summary_2)

# Convert log wages to actual wages
predicted_wage_2 = np.exp(prediction_summary_2['mean'][0])
print(f"Predicted wage with a college diploma: ${predicted_wage_2:.2f}")

# alpha : significance level for the confidence interval.
# Fix alpha to 0.05 so that the confidence interval should have 95% coverage.
```

```
   mean  mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower
   obs_ci_upper
0 9.868    0.035          9.800          9.936         8.073
11.662
1 9.868    0.035          9.800          9.936         8.073
11.662
2 9.847    0.035          9.779          9.915         8.052
11.641
3 9.868    0.035          9.800          9.936         8.073
11.662
4 9.847    0.035          9.779          9.915         8.052
11.641
5 9.847    0.035          9.779          9.915         8.052
11.641
6 9.868    0.035          9.800          9.936         8.073
11.662
```

```
7 9.868    0.035          9.800          9.936         8.073
11.662
8 9.868    0.035          9.800          9.936         8.073
11.662
9 9.868    0.035          9.800          9.936         8.073
11.662
Predicted wage with a college diploma: $19299.12
```

4.5d Those with college degrees have a higher predicted wage than those without. They earn about 6, 332 USD more on average.

4.5e SInce we know that a college degree will increase the average expected wage , it would be good to create policy to expand access to college education. Howeever, what our model does is to test correlation and not causation, so we do need to consider how other factors influence this (ommitted variables like household size or grades in school). The legislation should also try to ensure access to good quality education and ensure that The quality and type of education matter, not just access. ANd we should consider who (race, gender, origin, socioeconomic status) is being provided with this subsidy.

4.5f

```
NEW_reg_r_squared = NEW_reg.rsquared
NEW_reg_adjusted_r_squared = round(NEW_reg.rsquared_adj,4)

print(f"R-squared: {NEW_reg_r_squared:.4f}")
print(f"{(NEW_reg_adjusted_r_squared*100)} percent of the variation in logwages
 ↪  can be explained by the model")
```

```
R-squared: 0.3063
30.53 percent of the variation in logwages can be explained by the model
```

versus

```
OLD_reg_r_squared = LNINCWAGE_reg.rsquared
OLD_reg_adjusted_r_squared = LNINCWAGE_reg.rsquared_adj

print(f"R-squared: {OLD_reg_r_squared:.4f}")
print(f"Adjusted R-squared: {OLD_reg_adjusted_r_squared:.4f}")
```

```
R-squared: 0.2817
Adjusted R-squared: 0.2809
```

We see that the new model has a higher R 2 and adjusted R2, meaning it better explains the variation in log wages compared to the model from 3.

4.5g

For both 22 year olds, the CIs are pretty narrow, meaning the estimates are likely quite precise. High school diploma, the model predicts a log wage of 9.470154 with a 95% confidence interval of [9.401522, 9.538785]. For the college graduate, the predicted log wage is 9.867815 with a confidence interval of [9.799870, 9.935759]. But, the model only explains 30.53% of the variation in log wages, so there's still quite a bit of variation that is not explained.

So, while our model is quite precise, we must be wary. We may need to explore a bit more and use data from previous years as well.

4.6a

```python
# Predictors from Question 3 (excluding AGE and AGE_SQUARED)
predictors_q3 = ["EDUCDC", "FEMALE", "WHITE", "BLACK", "HISPANIC",
                 "MAR", "NCHILD", "VET"]

# Define the age variable and the target variable
X_age = acs_data[["AGE"]]
y = acs_data["LNINCWAGE"]

# Define the knots for the spline (18 and 65)
knots = np.array([18, 65]).reshape(-1, 1)

# Create the spline transformer
spline_transformer = SplineTransformer(degree=3, knots=knots,
 ↪  include_bias=False)

# Transform the age variable using the spline
X_splines = spline_transformer.fit_transform(X_age)

# Combine the spline-transformed age variable with the other predictors
X_controls = acs_data[predictors_q3]
X = np.hstack([X_splines, X_controls])

# Fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Print the intercept and coefficients
print("Intercept:", model.intercept_)
spline_columns = [f"spline_{i}" for i in range(X_splines.shape[1])]
all_columns = spline_columns + predictors_q3
coefficients = pd.Series(model.coef_, index=all_columns)
print("Coefficients:")
print(coefficients)

# Calculate the adjusted R-squared
r_squared = model.score(X, y)
```

```
n = len(y)
p = X.shape[1]
adjusted_r_squared = 1 - ((1 - r_squared) * (n - 1)) / (n - p - 1)
print("The Adjusted R-squared of the model with the two age knots is",
 ↪  adjusted_r_squared)
```

```
Intercept: 14.075815645865092
Coefficients:
spline_0   -22.189
spline_1    -1.931
spline_2    -6.236
EDUCDC       0.096
FEMALE      -0.379
WHITE       -0.013
BLACK       -0.130
HISPANIC    -0.028
MAR          0.190
NCHILD      -0.008
VET         -0.016
dtype: float64
The Adjusted R-squared of the model with the two age knots is
0.2914488777671098
```

4.6b The adjusted R2 of .291 is slightly higher than the one inmodel 3. The adjusted R2 of .291 is higher than the one inmodel 3. This is probably because, as we learned, splines capture nonlinear relationsships. In this case, the nonlinear effects of age, which thus improves the model's fit (it explains more of the vairaion).

4.6c

```
# Define the knots for the spline (24 and 55)
knots_24_55 = np.array([24, 55]).reshape(-1, 1)

# Create the spline transformer
spline_transformer_24_55 = SplineTransformer(degree=3, knots=knots_24_55,
 ↪  include_bias=False)

# Transform the age variable using the spline
X_splines_24_55 = spline_transformer_24_55.fit_transform(X_age)

# Combine the spline-transformed age variable with the other predictors
X_24_55 = np.hstack([X_splines_24_55, X_controls])

# Fit the linear regression model
model_24_55 = LinearRegression()
model_24_55.fit(X_24_55, y)
```

```python
# Print the intercept and coefficients
print("Intercept (knots at 24 and 55):", model_24_55.intercept_)
spline_columns_24_55 = [f"spline_24_55_{i}" for i in
↪  range(X_splines_24_55.shape[1])]
all_columns_24_55 = spline_columns_24_55 + predictors_q3
coefficients_24_55 = pd.Series(model_24_55.coef_, index=all_columns_24_55)
print("Coefficients (knots at 24 and 55):")
print(coefficients_24_55)

# Calculate the adjusted R-squared
r_squared_24_55 = model_24_55.score(X_24_55, y)
n_24_55 = len(y)
p_24_55 = X_24_55.shape[1]
adjusted_r_squared_24_55 = 1 - ((1 - r_squared_24_55) * (n_24_55 - 1)) /
↪  (n_24_55 - p_24_55 - 1)
print("Adjusted R-squared (knots at 24 and 55):", adjusted_r_squared_24_55)
```

```
Intercept (knots at 24 and 55): 12.615808625187793
Coefficients (knots at 24 and 55):
spline_24_55_0   -13.896
spline_24_55_1    -1.510
spline_24_55_2    -4.272
EDUCDC             0.102
FEMALE            -0.385
WHITE             -0.015
BLACK             -0.127
HISPANIC          -0.016
MAR                0.211
NCHILD             0.008
VET               -0.019
dtype: float64
Adjusted R-squared (knots at 24 and 55): 0.2673685975131527
```

Model of my choosing

```python
# Knots for the spline (21 and 65)
knots_21_65 = np.array([21, 65]).reshape(-1, 1)

# Create the spline transformer
spline_transformer_21_65 = SplineTransformer(degree=3, knots=knots_21_65,
↪  include_bias=False)

# Transform the age variable using the spline
X_splines_21_65 = spline_transformer_21_65.fit_transform(X_age)
```

```
# Combine the spline-transformed age variable with the other predictors
X_21_65 = np.hstack([X_splines_21_65, X_controls])

# Fit the linear regression model
model_21_65 = LinearRegression()
model_21_65.fit(X_21_65, y)

# Print the intercept and coefficients
print("Intercept (knots at 21 and 65):", model_21_65.intercept_)
spline_columns_21_65 = [f"spline_21_65_{i}" for i in
 ↪  range(X_splines_21_65.shape[1])]
all_columns_21_65 = spline_columns_21_65 + predictors_q3
coefficients_21_65 = pd.Series(model_21_65.coef_, index=all_columns_21_65)
print("Coefficients (knots at 21 and 65):")
print(coefficients_21_65)

# Getting the ajusted R-squared
r_squared_21_65 = model_21_65.score(X_21_65, y)
n_21_65 = len(y)
p_21_65 = X_21_65.shape[1]
adjusted_r_squared_21_65 = 1 - ((1 - r_squared_21_65) * (n_21_65 - 1)) /
 ↪  (n_21_65 - p_21_65 - 1)
print("Adjusted R-squared (knots at 21 and 65):", adjusted_r_squared_21_65)
```

```
Intercept (knots at 21 and 65): 13.613217907092636
Coefficients (knots at 21 and 65):
spline_21_65_0   -18.378
spline_21_65_1    -1.803
spline_21_65_2    -5.629
EDUCDC             0.098
FEMALE            -0.380
WHITE             -0.013
BLACK             -0.127
HISPANIC          -0.022
MAR                0.198
NCHILD            -0.006
VET               -0.014
dtype: float64
Adjusted R-squared (knots at 21 and 65): 0.2819850168684307
```

Since my model of 21 and 65 have a higher adjusted R2, it is a more powerful model that fits the data better than the suggested one. I chose 21, thinking this is the age in which many students graduate from university and 65 is when they retire, which would likely improve the model.

4.6d

```
# Create a test dataframe for predictions
df_test = pd.DataFrame({
    "EDUCDC": [16, 16],
    "FEMALE": [1, 1],
    "AGE": [17, 50],
    "WHITE": [0, 0],
    "BLACK": [0, 0],
    "HISPANIC": [0, 0],
    "MAR": [0, 0],
    "NCHILD": [0, 0],
    "VET": [0, 0]
})

# Transform AGE into splines using the transformer from 6.c (knots at 24 and
 ↪  55)
splines_test = spline_transformer_24_55.transform(df_test[["AGE"]])

# Combine spline-transformed AGE with other predictors from Question 3
X_test = np.hstack([splines_test, df_test[predictors_q3]])

# Predict log wages using the model with knots at 24 and 55
log_wage_pred = model_24_55.predict(X_test)

# Convert log wages to actual wages
wage_pred = np.exp(log_wage_pred)

# Create a results dataframe
df_pred = pd.DataFrame({
    "AGE": [17, 50],
    "Predicted Log Wage": log_wage_pred,
    "Predicted Wage": wage_pred
})

print(df_pred)
```

```
   AGE  Predicted Log Wage  Predicted Wage
0   17               9.824       18470.276
1   50              10.713       44950.928
```

The knots in the spline let us reflect wage differences because of the non-linear effects of age.

Here, we are assuming both the 17 and 50 year olds have 16 years of education, even if it isn't really realistic. Predictably, the 17 year olds, who have less work experience have lower predicted wage by around 31K USD. Meanwhile,at 50, one is likely at the top–a managing partner, director, VP, department head, etc, making earnings much higher.