**Q2.**

import numpy as np

a = np.array([[6,2,-5], [3,3,-2], [7,5,-3]])

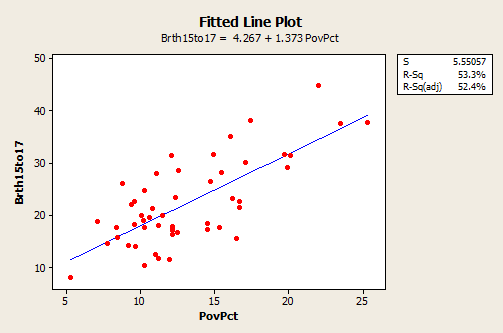
b = np.array([13,13,26])

x = np.linalg.solve(a, b)

print(x)

**Q3. The dataset of size *n* = 51 are for the 50 states and the District of Columbia in the United States. The variables are *y* = year 2002 birth rate per 1000 females 15 to 17 years old and *x* = poverty rate, which is the percent of the state’s population living in households with incomes below the federally defined poverty level. (Data source: *Mind On Statistics*, 3rd edition, Utts and Heckard).**

The plot of the data below (birth rate on the vertical) shows a generally linear relationship, on average, with a positive slope. As the poverty level increases, the birth rate for 15 to 17 year old females tends to increase as well.



The equation should really state that it is for the “average” birth rate (or “predicted” birth rate would be okay too) because a regression equation describes the average value of*y* as a function of one or more x-variables. In statistical notation, the equation could be written ^y=4.267+1.373xy^=4.267+1.373x.

* The interpretation of the slope (value = 1.373) is that the 15 to 17 year old birth rate increases 1.373 units, on average, for each one unit (one percent) increase in the poverty rate.
* The interpretation of the intercept (value=4.267) is that if there were states with poverty rate = 0, the predicted average for the 15 to 17 year old birth rate would be 4.267 for those states. Since there are no states with poverty rate = 0 this interpretation of the intercept is not practically meaningful for this example.

In the graph with a regression line present, we also see the information that *s* = 5.55057 and *r*2 = 53.3%.

**Python code using numpy:**

|  |
| --- |
| X = Q3Dataset[‘PovPct’] |
| y = Q3Dataset['Brth15to17'] |
|  |
| # Calculate the terms needed for the numerator and denominator of beta |
| df['xycov'] = (df['X'] - xmean) \* (df['y'] - ymean) |
| df['xvar'] = (df['X'] - xmean)\*\*2 |
|  |
| # Calculate beta and alpha |
| beta = df['xycov'].sum() / df['xvar'].sum() |
| alpha = ymean - (beta \* xmean) |
| print(f'alpha = {alpha}') |
| print(f'beta = {beta}')   |  | | --- | | # Plot regression against actual data  plt.figure(figsize=(12, 6)) | | plt.plot(X, ypred) # regression line | | plt.plot(X, y, 'ro') # scatter plot showing actual data | | plt.title('Actual vs Predicted') | | plt.xlabel('X') | | plt.ylabel('y') | |  | | plt.show()  def rmse(predictions, targets):  return np.sqrt(((predictions - targets) \*\* 2).mean())  rmse\_val = rmse(np.array(X), np.array(ypred))  print("rms error is: " + str(rmse\_val)) | |

**With Sklearn:**

|  |
| --- |
|  |
|  |  |
|  | from sklearn.linear\_model import LinearRegression  # Build linear regression model using PovPct as predictors |
|  | # Split data into predictors X and output Y |
|  |  |
|  | X = Q3Dataset[‘PovPct’] |
|  | y = Q3Dataset['Brth15to17'] |
|  |  |
|  | # Initialise and fit model |
|  | lm = LinearRegression() |
|  | model = lm.fit(X, y)   |  | | --- | | print(f'alpha = {model.intercept\_}') | | print(f'betas = {model.coef\_}')  from sklearn.metrics import mean\_squared\_error  from math import sqrt  rms = sqrt(mean\_squared\_error(y\_actual, y\_predicted)) |  | |