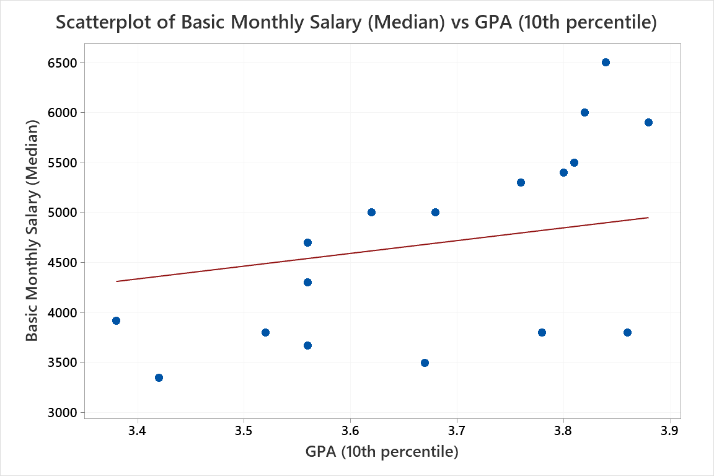
Dataset 3: GPA Salary 2022 NUS

Model 1

1a)



Equation

Predicted Basic Monthly Salary (Median) = 1275 × GPA (10th percentile)

1b)

n = 17,

To derive ,

1c)

1c(i)

Code:

#Model 1

monthlySalary = df['Basic Monthly Salary (Median)']

gpa = df['GPA (10th percentile)']

b = 2

rate = 0.001

epsilon = 0.001

diff = 1

max\_iter = 500

iter = 1

E = lambda b: (1/17)\*(np.sum([(monthlySalary - b \* gpa)\*\*2]))

deriv = lambda b: (-2/17)\* (np.sum([gpa \* (monthlySalary - b \* gpa)]))

while diff > epsilon and iter < max\_iter:

b\_new = b - rate \* deriv(b)

print("Iteration ", iter, ": b-value is: ", b\_new,"E(b) is: ", E(b\_new) )

diff = abs(b\_new - b)

iter = iter + 1

b = b\_new

print("Number of iterations is: ", iter)

print("The local minimum occurs when b is: ", b)

print("Minimum error is: ", E(b))

Output:

Number of iterations is: 383

The local minimum occurs when b is: 1275.29106526093

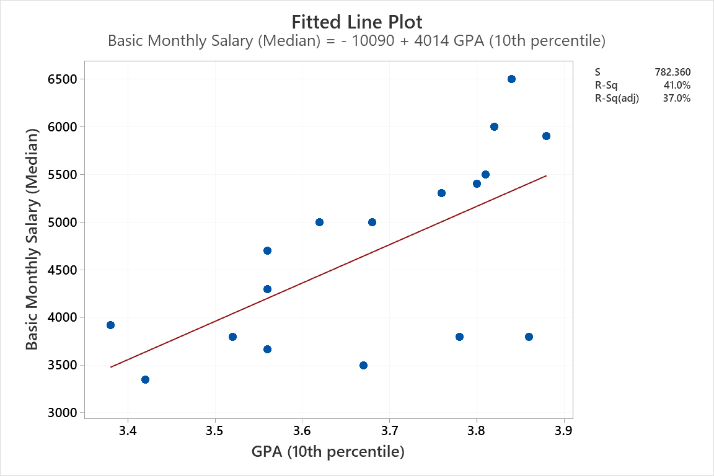
Minimum error is: 714816.1106934531

1c(ii)

Predicted Basic Monthly Salary (Median) = 1275.29106526093 × GPA (10th percentile)

Model 2

2a)



Predicted Basic Monthly Salary (Median) = -10090 + 4014 × GPA (10th percentile)

2b)

To derive ),

To derive ),

2c(i)

#Model 2

monthlySalary = df['Basic Monthly Salary (Median)']

gpa = df['GPA (10th percentile)']

next\_x = 4

next\_y = 5

alpha = 0.0679

epsilon = 0.001

max\_iters = 30000

iteration = 1

partialf\_x = lambda x,y: (-2/17)\*(np.sum((monthlySalary - (x + y \* gpa))))

partialf\_y = lambda x,y: (-2/17)\*(np.sum((gpa)\*(monthlySalary - (x + y \* gpa))))

func = lambda x,y: (1/17)\*(np.sum((monthlySalary - (x + y \* gpa))\*\*2))

next\_func = func(next\_x,next\_y)

for n in range(max\_iters):

current\_x = next\_x

current\_y = next\_y

current\_func = next\_func

next\_x = current\_x-alpha\*partialf\_x(current\_x,current\_y)

next\_y = current\_y-alpha\*partialf\_y(current\_x,current\_y)

next\_func = func(next\_x,next\_y)

iteration = iteration + 1

change\_func = abs(next\_func-current\_func)

print("Iteration",n+1,": x = ",next\_x,", y = ",next\_y,", f(x,y) = ",next\_func)

if change\_func<epsilon:

break

print("Number of iterations is: ", iteration)

print("The local minimum occurs when b is: ", next\_y)

print("Minimum error is: ", next\_func)

Output:

Number of iterations is: 26027

The local minimum occurs when b is: 4004.304236205563

Minimum error is: 540079.6723763739

2c(ii)

Predicted Basic Monthly Salary (Median) = -10053.4768 + 4004.304236 × GPA (10th percentile)

2(d)

Learning rate is chosen as 0.0679 and was adjusted smaller to prevent it from not converging. Maximum iteration is set as 30 000 as many iterations are required to reach convergence. Initial values were randomly chosen, and epsilon is chosen as 0.001.

Model 3

3(a)

Predictor ‘w’ is chosen as the number of months of job experience.

Data collection procedure: I assigned realistic values to each row by examining their GPA and median salary. For instance, a person with lower GPA but higher median salary than another person usually means that the person has more months of job experience.

Data: [2,24,0,2,12,15,20,24,15,20,20,12]

Code:

newDF = df.copy()

#Extract the first 12 rows

newDF = df.head(12)

#predictor w

newDF['Months Of Work Experience'] = [2,24,0,2,12,15,20,24,15,20,20,12]

3(b)

The general formula for error function is defined as:

We obtain the error function where n = 12 as there are 12 rows of data.

To determine how

1. ) changes with a while keeping b and c constant.
2. ) changes with b while keeping a and c constant.
3. ) changes with c while keeping a and b constant.

We will perform partial differential on the error function and derive ),) and). We will then pluck in these functions into our gradient descent algorithm later.

Gradient descent will then find the values a, b and c in the regression line:

1. At n-th point a = , y = , it will compute ),) and).
2. It will then update to the (n+1)-th point using ),) and ) to compute ).

To derive ), we differentiate the error function with respect to a. ‘b’ and ‘c’ are treated as constants.

To derive ), we differentiate the error function with respect to b. ‘a’ and ‘c’ are treated as constants.

To derive ), we differentiate the error function with respect to c. ‘a’ and ‘b’ are treated as constants.

3(c)

Code:

#Model 3

newDF = df.copy()

#Extract the first 12 rows

newDF = df.head(12)

#predictor w

newDF['Months Of Job Experience'] = [2,24,0,2,12,15,20,24,15,20,20,12]

newDF

newDFSalary = newDF['Basic Monthly Salary (Median)']

newDFGpa = newDF['GPA (10th percentile)']

newDFExperience = newDF['Months Of Job Experience']

next\_x = 4

next\_y = 5

next\_c = 6

alpha = 0.0037

epsilon = 0.001

max\_iters = 400000

iteration = 1

partialf\_x = lambda x,y,c: (-2/12)\*(np.sum((newDFSalary - (x + y \* newDFGpa + c \* newDFExperience))))

partialf\_y = lambda x,y,c: (-2/12)\*(np.sum((newDFGpa)\*(newDFSalary - (x + y \* newDFGpa + c \* newDFExperience))))

partialf\_c = lambda x,y,c: (-2/12)\*(np.sum((newDFExperience)\*(newDFSalary - (x + y \* newDFGpa + c \* newDFExperience))))

func = lambda x,y,c: (1/12)\*(np.sum((newDFSalary - (x + y \* newDFGpa + c \* newDFExperience))\*\*2))

next\_func = func(next\_x,next\_y,next\_c)

for n in range(max\_iters):

current\_x = next\_x

current\_y = next\_y

current\_c = next\_c

current\_func = next\_func

next\_x = current\_x-alpha\*partialf\_x(current\_x,current\_y,current\_c)

next\_y = current\_y-alpha\*partialf\_y(current\_x,current\_y,current\_c)

next\_c = current\_c-alpha\*partialf\_c(current\_x,current\_y,current\_c)

next\_func = func(next\_x,next\_y,next\_c)

iteration = iteration + 1

change\_func = abs(next\_func-current\_func)

print("Iteration",n+1,": a = ",next\_x,", b = ",next\_y,", c = ",next\_c,", f(a,b,c) = ",next\_func)

if change\_func<epsilon:

break

print("Number of iterations is: ", iteration)

print("The local minimum occurs when a is: ", next\_x)

print("The local minimum occurs when b is: ", next\_y)

print("The local minimum occurs when c is: ", next\_c)

print("Minimum error is: ", next\_func)

Output:

Number of iterations is: 364128

The local minimum occurs when a is: -6491.570409892604

The local minimum occurs when b is: 2712.4157094736984

The local minimum occurs when c is: 80.97953786174492

Minimum error is: 277004.45415598276

Explanation of Code:

1. I extracted the first 12 rows of the dataset and inserted data for months of job experience.
2. I set a learning rate of 0.0037, an epsilon of 0.001, maximum iteration of 100 000 and set an initial point of a=4, b=5 and c=6.
3. I defined the error function and partial derivative function of a, b and c.
4. At n-th point a = , y = , it will compute ),) and).
5. It will then update to the (n+1)-th point using ),) and ) to compute ).
6. An epsilon of 0.001 and maximum iteration of 400 000 is set as the stopping criterion.

To verify that the regression line obtained for model 3 is correct, I used Minitab and obtained the regression equation to cross-reference to my equation.

1. Open Minitab with the dataset opened.
2. Input the data for ‘Months Of Work Experience’ manually.
3. (Stat -> Regression -> Regression -> Fit Regression Model) to obtain the regression equation.

Our equation:

Minitab equation:

A black numbers and symbols

Description automatically generated with medium confidence

Cross referencing my equation to Minitab’s equation, the values a, b and c are somewhat similar to the Minitab’s equation. This verifies that my regression line for model 3 is correct.