Implementação do Método dos Míninos quadrados

Instruções: Implemente o Método dos Míninos quadrados em Python:

- Básico - Linear
- Quadrático.
- Robusto (com peso).

  Teste o funcionamento usando:
- “Alps Water”
- “Books x Grades”
- “US Census Dataset”

Não é permitido usar bibliotecas prontas. Proibidas: Sklearn, NumPy, SciPy.

Básico - Linear

$\beta = (X^t.X)^{-1}.X^T.y$

Definição das funções

In [76]:
```python
#Função para calcular a matriz transposta:

def transposta(X):
    """Retorna a transposta da matriz de entrada"""
    return [[X[j][i] for j in range(len(X))] for i in range(len(X[0]))]
```

In [77]:
```python
#Multiplicação de duas matrizes:

def multiply_matrix_matrix(mult1,mult2):
    """Retorna o produto escalar de duas matrizes"""
    x = []
    for i in range(0,len(mult1)):
        y=[]
        for j in range(0,len(mult2[0])):
            total = 0
            for k in range(0,len(mult1[0])):
                total = total + mult1[i][k]*mult2[k][j]
            y.append(total)
        x.append(y)

    return x
```

In [78]:
```python
#Multiplicação da matriz transposta por um vetor:

def multiply_matrix_vector(mult1,mult2):
    """Retorna o produto escalar de uma matriz com um vetor"""
    x=[]
    total = 0
    for i in range(len(mult1)):
        total=0
        for j in range(len(mult2)):
            total = total + mult1[i][j] * mult2[j]
        x.append(total)
    return x
```

In [79]:
```python
#Cálculo da inversa:

def Determinante(matriz):
    """Cálculo do determinante da matriz inserida"""
    #Determinante da matriz 2x2
    if len(matriz) == 2:
        return (matriz[0][0]*matriz[1][1])-(matriz[0][1]*matriz[1][0])

    determinant = 0
    for c in range(len(matriz)):
        determinant = determinant + ((-1)**c)*matriz[0][c]*Determinante([row[:c] + row[c+1:] for
                                                                        row in (matriz[:0]+matriz[0+1
:])])
    return determinant

def matriz_inversa(m):
    """Cálculo da matriz inversa através do método dos cofatores."""
    determinant = Determinante(m)

    #Fórmula para cáculo da matriz 2x2:
    if len(m) == 2:
        a = m[0][0]
        b = m[0][1]
        c = m[1][0]
        d = m[1][1]

        inverse= [d, -b, -c, a]
        calc_inverse=[]
        for i in inverse:
            x = (1/(a*d-b*c)) * i
            calc_inverse.append(x)
        col = 2
        inverse_matrix = [calc_inverse[i:i+col] for i in range(0, len(calc_inverse), col)]
        return inverse_matrix

    else:
    #Para os demais tamanho, matriz de cofatores:
        cofactors = []
        for r in range(len(m)):
            cofactorRow = []
            for c in range(len(m)):
                minor = [row[:c] + row[c+1:] for row in (m[:r]+m[r+1:])]
                cofactorRow.append(((-1)**(r+c)) * Determinante(minor))
            cofactors.append(cofactorRow)
        cofactors = transposta(cofactors)
        for r in range(len(cofactors)):
            for c in range(len(cofactors)):
                cofactors[r][c] = cofactors[r][c]/determinant
        return cofactors

#Obs: Para construir a matriz de cofatores, consultei o site:
#https://stackoverflow.com/questions/32114054/matrix-inversion-without-numpy
```

Testando nos Datasets:

In [80]:
```python
#Biblioteca Pandas - para carregar os arquivos txt disponibilizados no Moodle
import pandas as pd
```

Alps Water

- **Input:** Boiling
- **Output:** Pressure

Variáveis relevantes:

Xl_alps - Input X para o método dos mínimos quadrados linear;
Y_alps - Saída Y (igual para todos os métodos);
Xlt_alps - Transposta da input Xl_alps;
ml_alps - multiplicação $(Xt.X)$ para o método dos mínimos quadrados linear;
p1l_alps - $(Xt.X)^{-1}$ para o método dos mínimos quadrados linear;
p2l_alps - $(Xt.y)$ para o método dos mínimos quadrados linear;
betthal_alps - Resposta do exercício, coeficientes β0 e β1.

In [97]:
```python
#Importando o dataset
alps_water = pd.read_csv('DataSets/alpswater1.txt',sep='\t',header=None)
alps_water.columns= ['Row','Pressure','Boiling']
alps_water.drop(columns='Row', inplace=True)
alps_water.head()

# Defining input
x_input = [i for i in alps_water['Boiling']]
Xl_alps = [[1,i] for i in x_input]

#Definindo a saída
Y_alps = alps_water['Pressure']

#Traposta da matriz de entrada
Xlt_alps = transposta(Xl_alps)
Xlt_alps

#Produto escalar entre os elementos de duas matrizes
ml_alps = multiply_matrix_matrix(Xlt_alps,Xl_alps)
ml_alps

#Matriz inversa do produto escalar
p1l_alps = matriz_inversa(ml_alps)
p1l_alps

#Produto escalar da matriz X transposta pelo vetor y
p2l_alps= multiply_matrix_vector(Xlt_alps,Y_alps)
p2l_alps

#Resultado final
print('Método Linear - Alps Water:')
betthal_alps = multiply_matrix_vector(p1l_alps,p2l_alps)
print('Coeficiente β0 =', betthal_alps[0])
print('Coeficiente β1 =', betthal_alps[1])
```

```
Método Linear - Alps Water:
Coeficiente β0 = -81.06372712864686
Coeficiente β1 = 0.522892400784599
```

Books, attend and grade

- **Inputs:** Books and Attend
- **Output:** Grade

Variáveis relevantes:

Xl_books - Input X para o método dos mínimos quadrados linear;
Y_books - Saída Y (igual para todos os métodos);
Xlt_books - Transposta da input Xl_books;
ml_books - multiplicação $(Xt.X)$ para o método dos mínimos quadrados linear;
p1l_books - $(Xt.X)^{-1}$ para o método dos mínimos quadrados linear;
p2l_books- $(Xt.y)$ para o método dos mínimos quadrados linear;
betthal_books - Resposta do exercício, coeficientes β0, β1 e β2.

In [98]:
```python
#Importando o dataset
books_attend_grade = pd.read_csv('DataSets/Books_attend_grade.txt',sep='\t',header=None)
books_attend_grade.columns=['Books', 'Attend', 'Grade']
books_attend_grade.head()

#Cálculo da matriz X
x_input = [i for i in books_attend_grade['Books']]
Xl_books=[[1,i,j] for i, j in zip(x_input, books_attend_grade['Attend'])]
Xl_books

#Cálculo da matriz Y
Y_books = [i for i in books_attend_grade['Grade']]
Y_books

#Cálculo da matrix X transposta
Xlt_books = transposta(Xl_books)
Xlt_books

#Produto escalar da X transposta pela entrada X
ml_books = multiply_matrix_matrix(Xlt_books,Xl_books)
ml_books

#Cálculo da inversa da matrix 3x3 (Inversa da Multiplicação de X transposta por X)
p1l_books = matriz_inversa(ml_books)
p1l_books

#Multiplicação da matriz X transposta pelo vetor y
p2l_books = multiply_matrix_vector(Xlt_books,Y_books)
p2l_books

#Resultado final
print('Método Linear - Books, Attend and Grade:')
betthal_books = multiply_matrix_vector(p1l_books,p2l_books)
print('Coeficiente β0 =', betthal_books[0])
print('Coeficiente β1 =', betthal_books[1])
print('Coeficiente β2 =', betthal_books[2])
```

```
Método Linear - Books, Attend and Grade:
Coeficiente β0 = 37.379185204571286
Coeficiente β1 = 4.036892611009321
Coeficiente β2 = 1.2834772747099308
```

US-Census

- **Input:** x
- **Output:** y

Variáveis relevantes:

Xl_us - Input X para o método dos mínimos quadrados linear;
Y_us - Saída Y (igual para todos os métodos);
Xlt_us - Transposta da input Xl_us;
ml_us - multiplicação $(Xt.X)$ para o método dos mínimos quadrados linear;
p1l_us - $(Xt.X)^{-1}$ para o método dos mínimos quadrados linear;
p2l_us- $(Xt.y)$ para o método dos mínimos quadrados linear;
betthal_us - Resposta do exercício, coeficientes β0 e β1.

In [96]:
```python
us_census = pd.read_csv('DataSets/US-Census.txt', sep='\t', header=None)
us_census.columns = ['x','y'] #atribuição de nomes às colunas
us_census

#Definindo a matriz de entrada (X)
x_input = [i for i in us_census['x']]
Xl_us = [[1,i] for i in x_input]
Xl_us

#Definindo a matriz de saída(y)
Y_us = [i for i in us_census['y']]
Y_us

#Cálculo da matriz X transposta
Xlt_us = transposta(Xl_us)
Xlt_us

#Multiplicação de X transposta pela entrada X
ml_us = multiply_matrix_matrix(Xlt_us,Xl_us)
ml_us

#Cálculo da inversa da matrix 2x2 (Inversa da Multiplicação de X transposta por X)
p1l_us = matriz_inversa(ml_us)
p1l_us

#Multiplicação da matriz X transposta pelo vetor y
p2l_us = multiply_matrix_vector(Xlt_us,Y_us)
p2l_us

#Resultado final
print('Método Linear - US-Census:')
betthal_us = multiply_matrix_vector(p1l_us,p2l_us)
print('Coeficiente β0 =', betthal_us[0])
print('Coeficiente β1 =', betthal_us[1])
```

```
Método Linear - US-Census:
Coeficiente β0 = -3783.9455909089884
Coeficiente β1 = 2.025302727272731
```

Quadrático

X = [1 x x^2]
$\beta = (X^t.X)^{-1}.X^T.y$

Testando nos Datasets:

Alps Water

Variáveis relevantes:

- Xq_alps - Input X para o método dos mínimos quadrados quadrático;
- Y_alps - Saída Y (igual para todos os métodos);
- Xqt_alps - Transposta da input Xq_alps;
- mq_alps - multiplicação $(X^t.X)$ para o método dos mínimos quadrados quadrático;
- p1q_alps - $(X^t.X)^{-1}$ para o método dos mínimos quadrados quadrático;
- p2q_alps - $(X^t.y)$ para o método dos mínimos quadrados quadrático;
- betthaq_alps - Resposta do exercício, coeficientes β0, β1 e β2.

In [99]:
```python
#Dataset já foi importado anteriormente
alps_water

#Definição da matriz X, incluindo x^2
x_input= [i for i in alps_water['Boiling']]
Xq_alps = [[1,i,i**2] for i in x_input]
Xq_alps

#Definição do output
Y_alps = [i for i in alps_water['Pressure']]
Y_alps

#Cálculo da transposta de X
Xqt_alps = transposta(Xq_alps)
Xqt_alps

#Transposta de X escalar X
mq_alps = multiply_matrix_matrix(Xqt_alps,Xq_alps)
mq_alps

#Matriz inversa da (transposta de X escalar X)
p1q_alps = matriz_inversa(mq_alps)
p1q_alps

#Multiplicação da matriz X transposta pelo vetor y
p2q_alps = multiply_matrix_vector(Xqt_alps,Y_alps)
p2q_alps

#Resultado final
print('Método Quadrático - Alps Water:')
betthaq_alps = multiply_matrix_vector(p1q_alps,p2q_alps)
print('Intercept β0 =', betthaq_alps[0])
print('Linear Coeficiente β1 =', betthaq_alps[1])
print('Quadratic Coeficiente β2 =', betthaq_alps[2])
```

```
Método Quadrático - Alps Water:
Intercept β0 = 38.82929486036301
Linear Coeficiente β1 = -0.6547706308192573
Quadratic Coeficiente β2 = 0.002889712064870764
```

Books, attend and grade

Variáveis relevantes:

- Xq_books - Input X para o método dos mínimos quadrados quadrático;
- Y_books - Saída Y (igual para todos os métodos);
- Xqt_books - Transposta da input Xq_books;
- mq_books - multiplicação $(Xt.X)$ para o método dos mínimos quadrados quadrático;
- p1q_books - $(Xt.X)^{-1}$ para o método dos mínimos quadrados quadrático;
- p2q_books - $(Xt.y)$ para o método dos mínimos quadrados quadrático;
- betthaq_books- Resposta do exercício, coeficientes β0, β1, β2, β3 e β4.

In [100]:
```python
#Dataset already imported
books_attend_grade

#Definição da matriz X, incluindo x^2
x_input = [i for i in books_attend_grade['Books']]
Xq_books = [[1,i,j,i**2,j**2] for i, j in zip(x_input, books_attend_grade['Attend'])]
Xq_books

#Cálculo da matriz Y
Y_books = [i for i in books_attend_grade['Grade']]
Y_books

#Cálculo da matrix X transposta
Xqt_books = transposta(Xq_books)
Xqt_books

#Multiplicação de X transposta pela entrada X
mq_books = multiply_matrix_matrix(Xqt_books,Xq_books)
mq_books

#Cálculo da inversa da matrix 3x3 (Inversa da Multiplicação de X transposta por X)
p1q_books = matriz_inversa(mq_books)
p1q_books

#Multiplicação da matriz X transposta pelo vetor y
p2q_books = multiply_matrix_vector(Xqt_books,Y_books)
p2q_books

#Resultado final
betthaq_books = multiply_matrix_vector(p1q_books,p2q_books)
print('Método Quadrático - Books, attend and grades:')
print('Coeficiente β0 =', betthaq_books[0])
print('Coeficiente β1 =', betthaq_books[1])
print('Coeficiente β2 =', betthaq_books[2])
print('Coeficiente β3 =', betthaq_books[3])
print('Coeficiente β4 =', betthaq_books[4])
```

```
Método Quadrático - Books, attend and grades:
Coeficiente β0 = 66.52409983168764
Coeficiente β1 = 6.161855083001456
Coeficiente β2 = -3.4318843608712086
Coeficiente β3 = -0.48861319886634647
Coeficiente β4 = 0.1662748150648099
```

US-Census

Variáveis relevantes:

- Xq_us - Input X para o método dos mínimos quadrados quadrático;
- Y_us - Saída Y (igual para todos os métodos);
- Xqt_us - Transposta da input Xq_us;
- mq_us - multiplicação $(X^t.X)$ para o método dos mínimos quadrados quadrático;
- p1q_us - $(X^t.X)^{-1}$ para o método dos mínimos quadrados quadrático;
- p2q_us - $(X^t.y)$ para o método dos mínimos quadrados quadrático;
- betthaq_us - Resposta do exercício, coeficientes β0, β1 e β2.

In [101]:
```python
#Dataset already imported
us_census
#Definição da matriz de entrada (X), incluindo x^2
x_input = [i for i in us_census['x']]
Xq_us = [[1,i,i**2] for i in x_input]
Xq_us

#Definindo a matrix de saída(y)
Y_us = [i for i in us_census['y']]
Y_us

#Cálculo da matrix X transposta
Xqt_us = transposta(Xq_us)
Xqt_us

#Multiplicação de X transposta pela entrada X
mq_us= multiply_matrix_matrix(Xqt_us,Xq_us)
mq_us

#Cálculo da inversa da matrix 2x2 (Inversa da Multiplicação de X transposta por X)
p1q_us = matriz_inversa(mq_us)
p1q_us

#Multiplicação da matriz X transposta pelo vetor y
p2q_us = multiply_matrix_vector(Xqt_us,Y_us)
p2q_us

#Resultado final
betthaq_us = multiply_matrix_vector(p1q_us,p2q_us)
print('Método Quadrático - US-Census:')
print('Coeficiente β0 =', betthaq_us[0])
print('Coeficiente β1 =', betthaq_us[1])
print('Coeficiente β2 =', betthaq_us[2])
```

```
Método Quadrático - US-Census:
Coeficiente β0 = 32294.01736307144
Coeficiente β1 = -34.98747000005096
Coeficiente β2 = 0.009490454545471039
```

Robusto (com peso)

- $\beta = (X^T xW.X)^{-1}.X^T xW.y$

  Critério adotado para cálculo do peso:
- $W = abs(1/(yobs - ypred))$

Alps Water

Variáveis relevantes:

- Xr_alps - Input X para o método dos mínimos quadrados robustos;
- Xrt_alps - Transposta da input Xr_alps;
- Y_alps - Saída Y (igual para todos os métodos);
- W_alps - Peso para $W = abs(1/(yobs - (\beta0 + \beta1 * X1))$
  Obs: coeficientes obtidos através do método linear (betthal_alps[0] e betthal_alps[1])
- XtW_alps - Cálculo do produto vetorial entre $X^T xW$
- mr_alps - multiplicação $(X^T xW.X)$ para o método dos mínimos quadrados robustos;
- p1r_alps - $(X^T xW.X)$ para o método dos mínimos quadrados robustos;
- p2r_alps - $X^T xW.y$ para o método dos mínimos quadrados robustos;
- betthar_alps - Resposta do exercício, coeficientes robustos β0 e β1.

In [102]:
```python
#Dataset já importado anteriormente
alps_water

#A entrada para o método dos mínimos quadrados robustos é igual à entrada do método linear
Xr_alps = Xl_alps

#Transposta da entrada
Xrt_alps = transposta(Xr_alps)

#A saída do dataset é a mesma
Y_alps

#Cálculo do peso W
W_alps=[]
for i in range(len(Y_alps)):
    w=1/(Y_alps[i]-(betthal_alps[0]*Xr_alps[i][0]+betthal_alps[1]*Xr_alps[i][1]))
    if w<0:
        w=-w
    W_alps.append(w)

#Produto vetorial da X transposta com W_alps
Z=[]
for i,j in zip(Xrt_alps[0],W_alps):
    Z.append(i*j)

for i,j in zip(Xrt_alps[1],W_alps):
    Z.append(i*j)

XtW_alps = [Z[:len(Z)//2],Z[len(Z)//2:]]

#(X transposta vetorial W_alps) escalar com a entrada X
mr_alps = multiply_matrix_matrix(XtW_alps,Xr_alps)

#Inversa de do resultado anterior
p1r_alps = matriz_inversa(mr_alps)

#Cálculo do segundo elemento do produto escalar total
p2r_alps = multiply_matrix_vector(XtW_alps,Y_alps)

# Cálculo do Bettha através do método robusto
betthar_alps = multiply_matrix_vector(p1r_alps,p2r_alps)

#Respostas

print('Método Robusto - Alps Water:')

print('Coeficiente β0 =', betthar_alps[0])
print('Coeficiente β1 =', betthar_alps[1])
```

```
Método Robusto - Alps Water:
Coeficiente β0 = -81.36733077487588
Coeficiente β1 = 0.5243294760032597
```

In [105]:
```python
#Relevante:
print('Valor relevante: \nPeso W calculado:\n',W_alps)
```

```
Valor relevante:
Peso W calculado:
 [6.615717874720793, 3.91031831657218, 59.95568090952846, 123.07409021305531, 19.60108352980896, 8.89
1750989844958, 10.48699628945288, 10.006537357313336, 4.409124625060843, 5.4196679414565665, 3.888543
893640207, 1.5385989989840607, 128.713972933644, 3.9741256597416235, 14.346848941484716, 7.0014575863
19023, 6.025559864993501]
```

Books, attend and grade

Variáveis relevantes:

- Xr_books - Input X para o método dos mínimos quadrados robustos;
- Xrt_books - Transposta da input Xr_books;
- Y_books - Saída Y (igual para todos os métodos);
- W_books - Peso para $W = abs(1/(yobs - (\beta0 + \beta1 * X1 + \beta2 * X2))$
  Obs: coeficientes obtidos através do método linear (betthal_books[0] e betthal_books[1] e betthal_books[2])
- XtW_books - Cálculo do produto vetorial entre $X^T xW$
- mr_books - multiplicação $(X^T xW.X)$ para o método dos mínimos quadrados robustos;
- p1r_books - $(X^T xW.X)^{-1}$ para o método dos mínimos quadrados robustos;
- p2r_books - $X^T xW.y$ para o método dos mínimos quadrados robustos;
- betthar_books - Resposta do exercício, coeficientes robustos β0, β1 e β2.

In [106]:
```python
#Definição da matriz X do método robusto (igual ao método linear)
Xr_books=Xl_books

#Transposta de X
Xrt_books = transposta(Xr_books)

#Definição da matriz Y (igual para todos os métodos)
Y_books

#Cálculo da matriz W
W_books=[]
for i in range(len(Y_books)):
    w=1/(Y_books[i]-(betthal_books[0]*Xr_books[i][0]+betthal_books[1]*Xr_books[i][1]+betthal_books[2]*
Xr_books[i][2]))
    if w<0:
        w=-w
    W_books.append(w)
W_books

#Produto vetorial: X transposto x W
Z=[]
for i,j in zip(Xrt_books[0],W_books):
    Z.append(i*j)

for i,j in zip(Xrt_books[1],W_books):
    Z.append(i*j)

for i,j in zip(Xrt_books[2],W_books):
    Z.append(i*j)

n = 3 #separando a lista em 03 partes para representar a matriz
XtW_books = [Z[i::n] for i in range(n)]
XtW_books

#Multiplicação da transposta de X pelo produto vetorial
mr_books  = multiply_matrix_matrix(XtW_books,Xr_books)
mr_books

#Inversa do elemento 01 do produto escalar
p1r_books = matriz_inversa(mr_books)

#Cálculo do elemento 02 do produto escalar
p2r_books = multiply_matrix_vector(XtW_books,Y_books)

#Cálculo do coeficiente bettha
betthar_books = multiply_matrix_vector(p1r_books,p2r_books)

#Resposta - coeficientes método robusto dataset Books, attend and grade
print('Método Robusto - Books, attend and grade:')

print('Coeficiente β0 =', betthar_books[0])
print('Coeficiente β1 =', betthar_books[1])
print('Coeficiente β2 =', betthar_books[2])
```

```
Método Robusto - Books, attend and grade:
Coeficiente β0 = -818.8498826007017
Coeficiente β1 = 85.33311830852267
Coeficiente β2 = 46.5306542633009
```

In [107]:
```python
#Relevante:
print('Valor relevante: \nPeso W calculado:\n',W_books)
```

```
Valor relevante:
Peso W calculado:
 [0.2544218079640258, 0.2726105258151234, 0.19179287774647127, 0.06671734150338997, 0.73446870675075
8, 0.11358861953609173, 0.08669772648202224, 0.12814436055259634, 0.04936326852618996, 0.422180399898
34494, 0.097283727656483, 0.14495447647951382, 0.04847463275063433, 0.13791485445128246, 0.0544870149
50061925, 0.07546685363854524, 0.0712987088110479, 0.06897771808494317, 0.048071963263508995, 0.09150
400883828357, 0.04843786511683539, 0.44747803208037473, 0.030964669806083815, 0.07276949407331455, 0.
062167314299350275, 0.06281426310644295, 3.876301907208429, 0.19515741301934358, 0.13195488462149368,
7.237199318140946, 0.0637307064918253, 0.058402632219830164, 0.06034558547536571, 0.05938033719826099
6, 0.2629019902847974, 0.06755068511233059, 0.9604685457653549, 0.23169043263837652, 0.04543101801466
875, 0.08484612526405702]
```

US-Census

Variáveis relevantes:

- Xr_us - Input X para o método dos mínimos quadrados robustos;
- Xrt_us- Transposta da input Xr_us;
- Y_us - Saída Y (igual para todos os métodos);
- W_us - Peso para $W = abs(1/(yobs - (\beta0 + \beta1 * X1))$
  Obs: coeficientes obtidos através do método linear (betthal_us[0] e betthal_us[1])
- XtW_us - Cálculo do produto vetorial entre $X^T xW$
- mr_us - multiplicação $(X^T xW.X)$ para o método dos mínimos quadrados robustos;
- p1r_us - $(X^T xW.X)^{-1}$ para o método dos mínimos quadrados robustos;
- p2r_us - $X^T xW.y$ para o método dos mínimos quadrados robustos;
- betthar_us - Resposta do exercício, coeficientes robustos β0 e β1.

In [108]:
```python
#Entrada igual ao método linear
Xr_us=Xl_us

#Transposta da entrada
Xrt_us= transposta(Xr_us)

#Saida é a mesma para todos os métodos
Y_us

#Cálculo do coeficiente W
W_us=[]
for i in range(len(Y_us)):
    w= (1/(Y_us[i]-(betthal_us[0]*Xr_us[i][0]+betthal_us[1]*Xr_us[i][1])))
    if w<0: #módulo não pode ser negativo
        w=-w
    W_us.append(w)

#Produto vetorial: X transposto x W
Z=[]
for i,j in zip(Xrt_us[0],W_us):
    Z.append(i*j)

for i,j in zip(Xrt_us[1],W_us):
    Z.append(i*j)

XtW_us  = [Z[:len(Z)//2],Z[len(Z)//2:]]
XtW_us

#Produto escalar entre as matrizes
mr_us  = multiply_matrix_matrix(XtW_us,Xr_us)

#Inversa do step anterior
p1r_us = matriz_inversa(mr_us)

#Multiplicação do produto vetorial entre Xt e W com a saída Y
p2r_us = multiply_matrix_vector(XtW_us,Y_us)

#Cálculo da resposta
betthar_us = multiply_matrix_vector(p1r_us,p2r_us)

#Resultados, coeficiente bettha robusto
print('Método Robusto - US-Census:')
print('Coeficiente β0 =', betthar_us[0])
print('Coeficiente β1  =', betthar_us[1])
```

```
Método Robusto - US-Census:
Coeficiente β0 = -3778.840082823066
Coeficiente β1  = 2.022789272835894
```

In [109]:
```python
#Relevante:
print('Valor relevante: \nPeso W calculado:\n',W_books)
```

```
Valor relevante:
Peso W calculado:
 [0.2544218079640258, 0.2726105258151234, 0.19179287774647127, 0.06671734150338997, 0.73446870675075
8, 0.11358861953609173, 0.08669772648202224, 0.12814436055259634, 0.04936326852618996, 0.422180399898
34494, 0.097283727656483, 0.14495447647951382, 0.04847463275063433, 0.13791485445128246, 0.0544870149
50061925, 0.07546685363854524, 0.0712987088110479, 0.06897771808494317, 0.048071963263508995, 0.09150
400883828357, 0.04843786511683539, 0.44747803208037473, 0.030964669806083815, 0.07276949407331455, 0.
062167314299350275, 0.06281426310644295, 3.876301907208429, 0.19515741301934358, 0.13195488462149368,
7.237199318140946, 0.0637307064918253, 0.058402632219830164, 0.06034558547536571, 0.05938033719826099
6, 0.2629019902847974, 0.06755068511233059, 0.9604685457653549, 0.23169043263837652, 0.04543101801466
875, 0.08484612526405702]
```

Summary: Results

> Aprentação do resultado de β por datasets. Comparação dos resultados de cada método.

Alps Water

In [79]:
```python
#Resultado final
print('Método Linear - Alps Water:')
betthal_alps = multiply_matrix_vector(p1l_alps,p2l_alps)
print('Coeficiente β0 =', betthal_alps[0])
print('Coeficiente β1 =', betthal_alps[1])
print('\n')
print('Método Quadrático - Alps Water:')
betthaq_alps = multiply_matrix_vector(p1q_alps,p2q_alps)
print('Intercept β0 =', betthaq_alps[0])
print('Linear Coeficiente β1 =', betthaq_alps[1])
print('Quadratic Coeficiente β2 =', betthaq_alps[2])
print('\n')
print('Método Robusto - Alps Water:')
print('Coeficiente β0 =', betthar_alps[0])
print('Coeficiente β1 =', betthar_alps[1])
```

```
Método Linear - Alps Water:
Coeficiente β0 = -81.06372712864686
Coeficiente β1 = 0.522892400784599


Método Quadrático - Alps Water:
Intercept β0 = 38.82929486036301
Linear Coeficiente β1 = -0.6547706308192573
Quadratic Coeficiente β2 = 0.002889712064870764


Método Robusto - Alps Water:
Coeficiente β0 = -81.36733077487588
Coeficiente β1 = 0.5243294760032597
```

Books, attend and grade

In [83]:
```python
#Resultado final
print('Método Linear - Books, Attend and Grade:')
betthal_books = multiply_matrix_vector(p1l_books,p2l_books)
print('Coeficiente β0 =', betthal_books[0])
print('Coeficiente β1 =', betthal_books[1])
print('Coeficiente β2 =', betthal_books[2])
print('\n')
betthaq_books = multiply_matrix_vector(p1q_books,p2q_books)
print('Método Quadrático - Books, attend and grades:')
print('Coeficiente β0 =', betthaq_books[0])
print('Coeficiente β1 =', betthaq_books[1])
print('Coeficiente β2 =', betthaq_books[2])
print('Coeficiente β3 =', betthaq_books[3])
print('Coeficiente β4 =', betthaq_books[4])
print('\n')
print('Método Robusto - Books, attend and grade:')

print('Coeficiente β0 =', betthar_books[0])
print('Coeficiente β1 =', betthar_books[1])
print('Coeficiente β2 =', betthar_books[2])
```

```
Método Linear - Books, Attend and Grade:
Coeficiente β0 = 37.379185204571286
Coeficiente β1 = 4.036892611009321
Coeficiente β2 = 1.2834772747099308


Método Quadrático - Books, attend and grades:
Coeficiente β0 = 66.52409983168764
Coeficiente β1 = 6.161855083001456
Coeficiente β2 = -3.4318843608712086
Coeficiente β3 = -0.48861319886634647
Coeficiente β4 = 0.1662748150648099


Método Robusto - Books, attend and grade:
Coeficiente β0 = -818.8498826007017
Coeficiente β1 = 85.33311830852267
Coeficiente β2 = 46.5306542633009
```

US-Census

In [85]:

```python
#Resultado final
print('Método Linear - US-Census:')
betthal_us = multiply_matrix_vector(p1l_us,p2l_us)
print('Coeficiente β0 =', betthal_us[0])
print('Coeficiente β1 =', betthal_us[1])
print('\n')

print('Método Quadrático - US-Census:')
print('Coeficiente β0 =', betthaq_us[0])
print('Coeficiente β1 =', betthaq_us[1])
print('Coeficiente β2 =', betthaq_us[2])

print('\n')
print('Método Robusto - US-Census:')
print('Coeficiente β0 =', betthar_us[0])
print('Coeficiente β1  =', betthar_us[1])
```

```
Método Linear - US-Census:
Coeficiente β0 = -3783.9455909089884
Coeficiente β1 = 2.025302727272731


Método Quadrático - US-Census:
Coeficiente β0 = 32294.01736307144
Coeficiente β1 = -34.98747000005096
Coeficiente β2 = 0.009490454545471039


Método Robusto - US-Census:
Coeficiente β0 = -3778.840082823066
Coeficiente β1  = 2.022789272835894
```