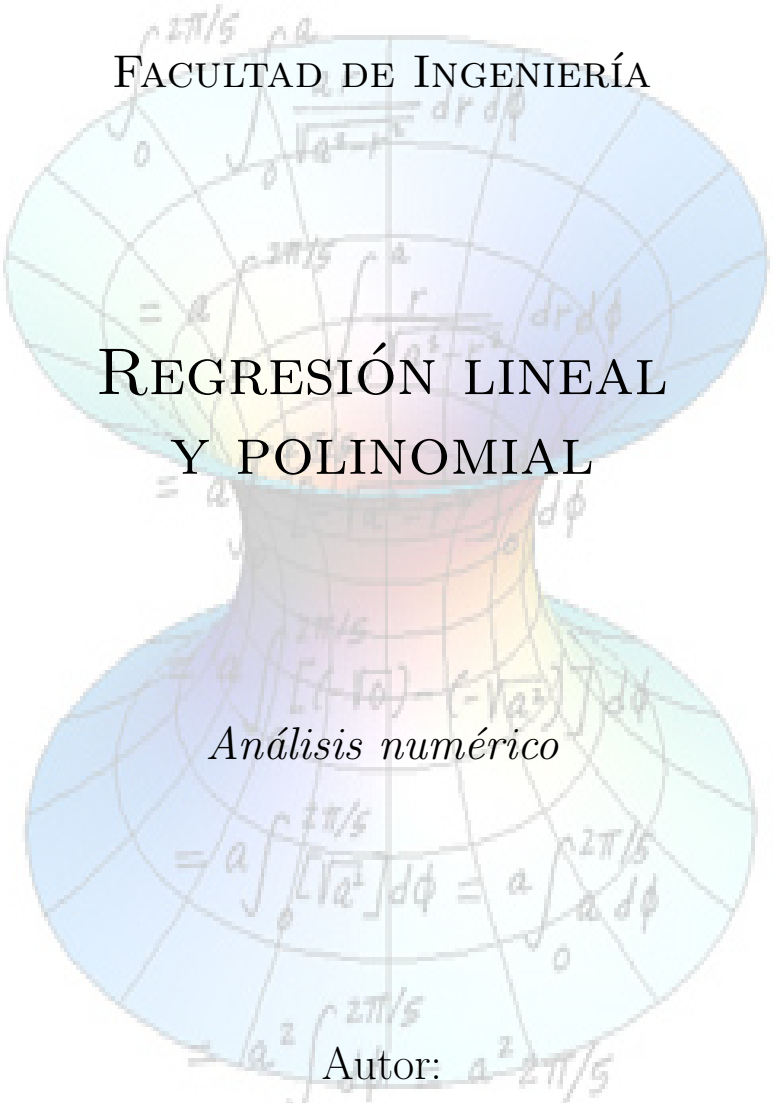
UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA

Universidad Autónoma de Querétaro

Facultad de Ingeniería



Regresión lineal y polinomial

Análisis numérico

Autor:
David Gómez Torres

6 Octubre del 2021

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

1.1. Regresión lineal

El ejemplo más simple de una aproximación por mínimos cuadrados es ajustar una línea recta a un conjunto de observaciones definidas por puntos: (x_1,y_1), (x_2,y_2),...,(x_n, y_n).

La expresión matemática para la línea recta es

$$y = a_0 + a_1 x + e \tag{1.1}$$

donde a_0 y a_1 son coeficientes que representan la intersección con el eje y y la pendiente, respectivamente, e es el error, o diferencia, entre el modelo y las observaciones.

1.1.1. Criterio para un mejor juste

La estrategia que supera las deficiencias de los procedimientos mencionados consiste en minimizar la suma de los cuadrados de los residuos entre la y medida y la y calculada con el modelo lineal:

$$S_r = \sum_{i=1}^{n} (y_{i,medida} - y_{i,modelo})^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_1)^2 \tag{1.2}$$

1.1.2. Ajuste de una línea recta por mínimos cuadrados

Para determinar los valores de a_0 y a_1 , la ecuación (1.2) se deriva con respecto a cada uno de los coeficientes:

$$\sum y_i - \sum a_0 - \sum a_1 x_1 = 0$$
$$\sum y_i x_i - \sum a_0 x_i - \sum a_1 x_1^2 = 0$$

Estas se pueden reescribir como

$$n a_0 + \left(\sum x_i\right) a_1 = \sum y_i$$
$$\left(\sum x_i\right) a_0 + \left(x_i^2\right) a_1 = \sum x_i y_i$$

para dar lugar a las **ecuaciones normales** que se resuelven de manera simultanea:

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

Si este resultado se acomoda, obtenemos

$$a_0 = \bar{y} - a_1 \bar{x} \tag{1.3}$$

donde $\bar{y}$ y $\bar{x}$ son las medias de y y x, respectivamente.

1.1.3. Cuantificación del error

Si además de estos criterios se satisfacen, una "desviación estándar" para la línea de regresión se determina como sigue:

$$S_{y/x} = \sqrt{\frac{S_r}{n-2}} \tag{1.4}$$

donde a $s_{y/x}$ se le llama **error estándar de la estimación**.

Una representación alternativa para r que es más conveniente para implementarse en una computadora es:

$$r = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{\sqrt{n\sum x_i^2 - (\sum x_i^2)}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \tag{1.5}$$

1.1.4. Linealización de relaciones no lineales

En algunos casos, se pueden utilizar transformaciones para expresar los datos en una forma que sea compatible con la regresión lineal.

- **Modelo exponencial:**

$$y = a_1 e^{\beta_1 x} \tag{1.6}$$

donde a_1 y β_1 son constantes.

Para linealizar la ecuación (1.6), se le aplica el logaritmo natural, obteniendo:

$$\ln y = \ln a_1 + \beta_1 x \ln e \tag{1.7}$$

- **Modelo de potencias:**

$$y = a_2 x^{\beta_2} \tag{1.8}$$

donde a_2 y β_2 son constantes.

Para linealizar la ecuación (1.8), utilizamos el logaritmo de base 10:

$$\log y = \beta_2 \log x + \log a_2 \tag{1.9}$$

- **Modelo de razón del crecimiento:**

$$y = a_3 \frac{x}{\beta_3 + x} \tag{1.10}$$

La ecuación (1.10) es linealizada al invertirla para dar resultado:

$$\frac{1}{y} = \frac{\beta_3}{a_3}\frac{1}{x} + \frac{1}{a_3} \tag{1.11}$$

1.2. Regresión polinomial

El procedimiento de mínimos cuadrados se puede extender fácilmente al ajuste de datos con un polinomio de grado superior. Por ejemplo, suponga que ajustamos un polinomio de segundo grado o cuadrático:

$$y = a_0 + a_1 x + a_2 x^2 + e$$

En este caso, la suma de los cuadrados de los residuos es

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i - a_2 x_i^2)^2$$

El caso bidimensional se extiende con facilidad a un polinomio de m-ésimo grado como sigue:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + e \tag{1.12}$$

En este caso, el error estándar se formula como sigue:

$$S_{y/x} = \sqrt{\frac{S_r}{n - (m+1)}} \tag{1.13}$$

Un problema potencial en la implementación de la regresión polinomial en la computadora es que las ecuaciones normales algunas veces están mal condicionadas. Esto se presenta especialmente cuando se plantean polinomios de grado superior.

1.3. Regresión lineal múltiple

Una extensión útil de la regresión lineal es el caso en el que y es una función lineal de dos o más variables independientes. Por ejemplo, y podría ser una función lineal de x_1 y x_2, como en:

$$y = a_0 + a_1 x_1 + a_2 x_2 + e$$

En este caso bidimensional, la "línea" de regresión se convierte en un "plano".Ver figura()

Como en los casos anteriores, los "mejores" valores para los coeficientes se determinan al realizar la suma de los cuadrados de los residuos,

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_{1i} - a_2 x_{2i})^2$$

El caso bidimensional anterior fácilmente se extiende a m-dimensiones así:

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_m x_m + e$$

donde el error estándar se formula como

$$S_{y/x} = \sqrt{\frac{S_r}{n - (m+1)}}$$

Aunque puede haber ciertos casos donde una variable esté linealmente relacionada con dos o más variables, la regresión lineal múltiple tiene además utilidad en la obtención de ecuaciones de potencias de la forma general

$$y = a_0 x_1^{a_1} x_2^{a_2} \cdots x_m^{a_m} \tag{1.14}$$

Tales ecuaciones son extremadamente útiles cuando se ajustan datos experimentales. Para usar regresión lineal múltiple, la ecuación se transforma al aplicar logaritmos:

$$\log y = \log a_0 + a_1 \log x_1 + a_2 \log x_2 + \cdots + a_m \log x_m \tag{1.15}$$

1.4. Regresión no lineal

En el presente contexto, tales modelos se definen como aquellos que tienen dependencia no lineal de sus parámetros.

$$f(x) = a_0(1 - e^{-a_1 x}) + e \tag{1.16}$$

Como en el caso de los mínimos cuadrados lineales, la regresión no lineal se basa en la determinación de los valores de los parámetros que minimizan la suma de los cuadrados de los residuos. Sin embargo, en el caso no lineal, la solución debe realizarse en una forma iterativa.

El **método de Gauss-Newton** es un algoritmo para minimizar la suma de los cuadrados de los residuos entre los datos y las ecuaciones no lineales. El concepto clave detrás de esta técnica es que se utiliza una expansión en serie de Taylor para expresar la ecuación no lineal original en una forma lineal aproximada.

El modelo no lineal se puede escribir como la siguiente ecuación:

$$y_i - f(x_i)_j = \frac{\partial f(x_i)_j}{\partial a_0}\Delta a_0 + \frac{\partial f(x_i)_j}{\partial a_1}\Delta a_1 + e_i \tag{1.17}$$

o en forma matricial

$$\{D\} = [Z_j]\{\Delta A\} + \{E\} \tag{1.18}$$

Este procedimiento se repite hasta que la solución converge, es decir, hasta que

$$|\varepsilon_a|_k = \left|\frac{a_{k,j+1} - a_{k,j}}{a_{k,j+1}}\right| 100\,\% \tag{1.19}$$

está por debajo de un criterio de terminación aceptable.

CAPÍTULO 2

METODOLOGÍA

2.1. Modelo lineal

Problema 17.5

Con el mismo enfoque que se empleó para obtener las ecuaciones, obtenga el ajuste por mínimos cuadrados del modelo siguiente:

$$y = a_1 x + e$$

Es decir, determine la pendiente que resulta en el ajuste por mínimos cuadrados para una línea recta con intersección en el origen. Ajuste los datos siguientes con dicho modelo e ilustre el resultado con una gráfica:

x	2	4	6	7	10	11	14	17	20
y	1	2	5	2	8	7	6	9	12

1. Gráfica del ajuste lineal a los datos

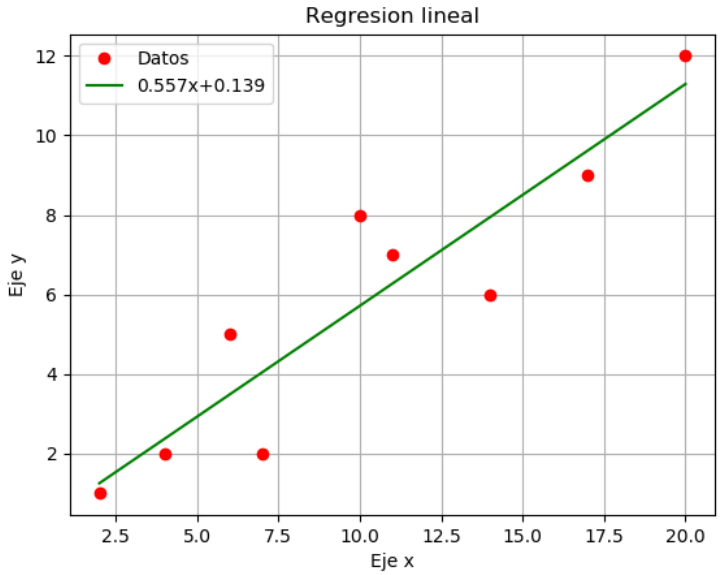


Figura 2.1: Línea ajustada a los datos

2. Solución por la regresión lineal

```python
import math
import numpy as np
import matplotlib.pyplot as plt

#arreglo con valores
x=[2,4,6,7,10,11,14,17,20]
y=[1,2,5,2,8,7,6,9,12]

n = len(x) #numeros de datos
#convertir en vectores
x = np.array(x)
y = np.array(y)

#sumatorias
sumx = sum(x)
sumy = sum(y)
sumx2 = sum(x**2)
sumy2 = sum(y**2)
sumxy = sum(x*y)

#promedio de datos
xm = sumx/n
ym = sumy/n
#valores de la recta
a1 = (sumx*sumy-n*sumxy)/((sumx)**2 - n*sumx2) #pendiente (m)
a0 = ym - a1*xm #interseccion con el eje (b)
#errores del metodo
st = sum((y - ym)**2)
sr = sum((y - a1*x - a0)**2)
syx = math.sqrt(sr/(n-2)) #error estand.
r = math.sqrt((st - sr)/st) #coef.correlacion
print("Error estandar: ", syx)
print("Coeficiente de correlacion: ", r)
print('Pendiente                    Interseccion con el eje X')
print(f'{a1:10} ==> {a0:10f}')

#graficar
plt.plot(x,y, 'o', label='Datos',color='red') #puntos
plt.plot(x, a1*x+a0, label='0.557x+0.139',color='green') #recta
#titulo de ejes
plt.xlabel('Eje x')
plt.ylabel('Eje y')
plt.grid()#generar malla
plt.title('Regresion lineal')#titulo de grafica
plt.legend()#mostrar recuadro
plt.show()#mostrar grafica
```

2.2. Modelo exponencial

Problema 17.9

Ajuste a un modelo exponencial a

x	0.4	0.8	1.2	1.6	2	2.3
y	800	975	1500	1950	2900	3600

Grafique los datos y la ecuación tanto en papel milimétrico como en semilogarítmico.

1. Gráfica linealizada

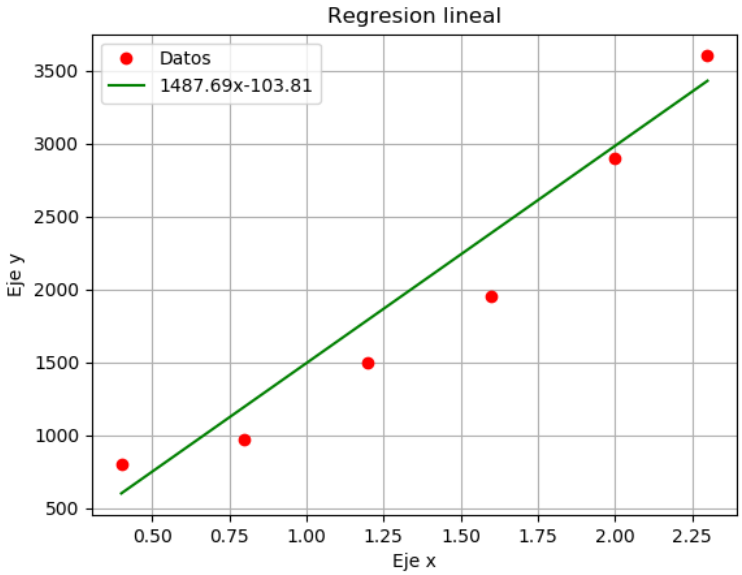


Figura 2.2: Linealización de un modelo exponencial

2. Método de exponencial

```
import math
import numpy as np
import matplotlib.pyplot as plt

#arreglo con valores
x=[0.4,0.8,1.2,1.6,2,2.3]
y=[800,975,1500,1950,2900,3600]
#numeros de datos
n = len(x)
#convertir en vectores
x = np.array(x)
y = np.array(y)
#sumatorias
sumx = sum(x)
sumy = sum(y)
sumx2 = sum(x**2)
sumy2 = sum(y**2)
sumxy = sum(x*y)

#promedio de datos
xm = sumx/n
ym = sumy/n
#valores de la recta
a1 = (sumx*sumy-n*sumxy)/((sumx)**2 - n*sumx2) #pendiente (m)
a0 = ym - a1*xm #interseccion con el eje (b)
#errores del metodo
st = sum((y - ym)**2)
sr = sum((y - a1*x - a0)**2)

syx = math.sqrt(sr/(n-2)) #error estand.
r = math.sqrt((st - sr)/st) #coef.correlacion
print("Error estandar: ", syx)
print("Coeficiente de correlacion: ", r)
print('Pendiente                    Interseccion con el eje X')
print(f'{a1:10} ==> {a0:10f}')
#graficar
plt.plot(x,y, 'o', label='Datos',color='red') #puntos
plt.plot(x, np.log(a0)+a1*x, label='1487.69x-103.81',color='green')
#titulo de ejes
plt.xlabel('Eje x')
plt.ylabel('Eje y')
plt.grid()#generar malla
plt.title('Regresion lineal')#titulo de grafica
plt.legend()#mostrar recuadro
plt.show()#mostrar grafica
```

2.3. Modelo de potencias

Ejercicio 17.17

Ajuste una ecuación cúbica a los datos siguientes:

x	3	4	5	7	8	9	11	12
y	1.6	3.6	4.4	3.4	2.2	2.8	3.8	4.6

Además de los coeficientes, determine r^2 y $s_{y/x}$.

1. Gráfica del modelo cúbico linealizado

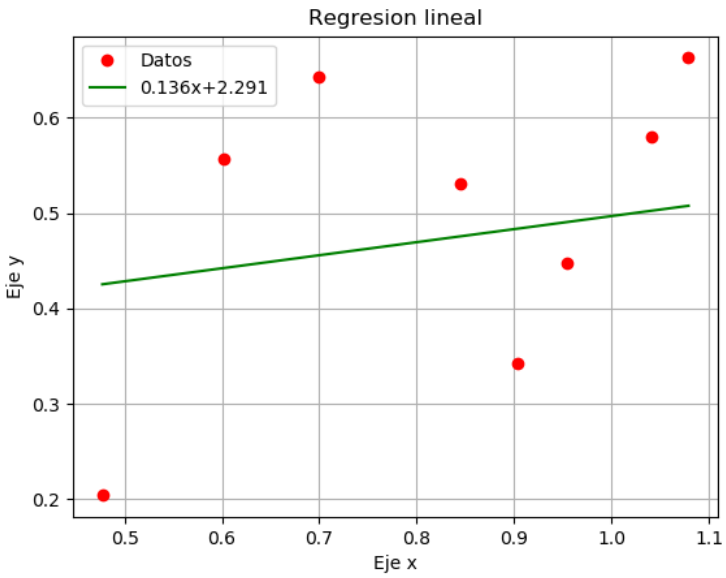


Figura 2.3: Modelo de potencias

2. Coeficiente de determinación r^2 y error estándar $S_{y/x}$

$$r^2 = 0.1816$$
$$S_{y/x} = 1.018$$

Ver Figura (3.3) en el anexo.

3. Método de potencias

```python
import math
import numpy as np
import matplotlib.pyplot as plt

#arreglo con valores
x=[3,4,5,7,8,9,11,12]
y=[1.6,3.6,4.4,3.4,2.2,2.8,3.8,4.6]
#numeros de datos
n = len(x)
#convertir en vectores
x = np.array(x)
y = np.array(y)

#sumatorias
sumx = sum(x)
sumy = sum(y)
sumx2 = sum(x**2)
sumy2 = sum(y**2)
sumxy = sum(x*y)

#promedio de datos
xm = sumx/n
ym = sumy/n
#valores de la recta
a1 = (sumx*sumy-n*sumxy)/((sumx)**2 - n*sumx2) #pendiente (m)
a0 = ym - a1*xm #interseccion con el eje (b)
#errores del metodo
st = sum((y - ym)**2)
sr = sum((y - a1*x - a0)**2)
syx = math.sqrt(sr/(n-2)) #error estand.
r2 = (st - sr)/st #coef.correlacion
print("Error estandar: ", syx)
print("Coeficiente de determinacion: ", r2)
print('Pendiente                    Interseccion con el eje X')
print(f'{a1:10} ==> {a0:10f}')

#graficar
plt.plot(np.log10(x),np.log10(y), 'o', label='Datos',color='red')
plt.plot(np.log10(x), a1*np.log10(x)+np.log10(a0),
label='0.136x+2.291',color='green')
#titulo de ejes
plt.xlabel('Eje x')
plt.ylabel('Eje y')
plt.grid()#generar malla
plt.title('Regresion lineal')#titulo de grafica
plt.legend()#mostrar recuadro
plt.show()#mostrar grafica
```

2.4. Regresión lineal múltiple

Ejercicios 17.19

Use regresión lineal múltiple para ajustar

x	0	0	1	2	0	1	2	2	1
x_2	0	2	2	4	4	6	6	2	1
y	14	21	11	12	23	23	14	6	11

Calcule los coeficientes, el error estándar de la estimación y el coeficiente de correlación.

```
import pandas as pd
import numpy as np

#crear dataframe
df = pd.read_csv('datos.csv')
n = len(df)#longitud de la cadena
#sumatorias para la matriz
sx1 = df['x1'].sum()
sx2 = df['x2'].sum()

df['x1_2']=df['x1']**2
sx1_2 = df['x1_2'].sum()

df['x1x2'] = df['x1']*df['x2']
sx1x2 = df['x1x2'].sum()

df['x2_2']=df['x1_2']**2
sx2_2 = df['x2_2'].sum()
#vector solucion
sy = df['y'].sum()
df['x1y'] = df['x1']*df['y']
sx1y = df['x1y'].sum()
df['x2y'] = df['x2']*df['y']
sx2y = df['x2y'].sum()

#crear matriz
A = np.zeros(shape=(3,3))
b = np.zeros(shape=(3,1))
#Llenar componentes de la matriz
A[0,0] = n
A[0,1] = sx1
A[0,2] = sx2
A[1,0] = sx1
A[1,1] = sx1_2
A[1,2] = sx1x2
A[2,0] = sx2
A[2,1] = sx1x2
A[2,2] = sx2_2
#Llenar vector sol.
b[0] = sy
b[1] = sx1y
b[2] = sx2y
#solucionar sistema
x = np.linalg.solve(A,b)
a0 = x[0].item
a1 = x[1].item
a2 = x[2].item
```

2.5. Regresión lineal a datos físicos

Ejercicios 17.28

Un objeto se suspende en un túnel de viento y se mide la fuerza para varios niveles de velocidad del viento. A continuación están tabulados los resultados.

v, m/s	10	20	30	40	50	60	70	80
F, N	25	70	380	550	610	1220	830	1450

Emplee regresión por mínimos cuadrados para ajustar estos datos con a) una línea recta, b) una ecuación de potencias basada en transformaciones logarítmicas. Muestre los resultados gráficamente.

1. Método lineal

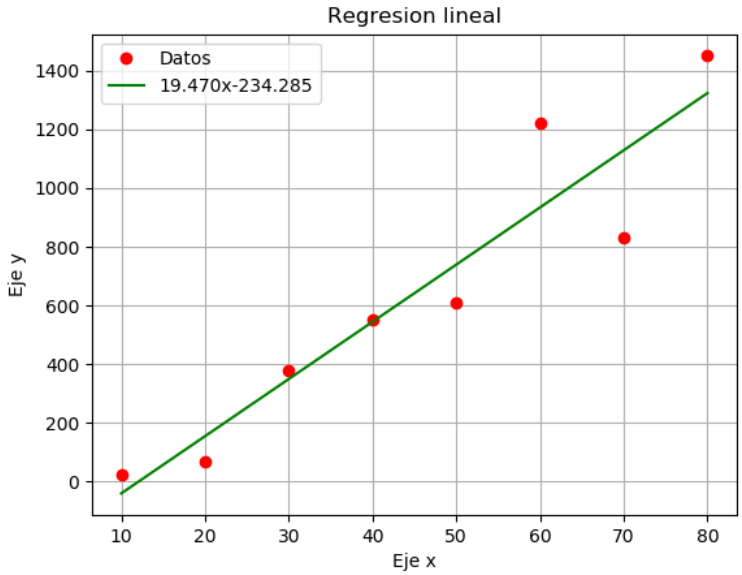


Figura 2.4: Modelo lineal

```python
import math
import numpy as np
import matplotlib.pyplot as plt

#arreglo con valores
x=[10,20,30,40,50,60,70,80]
y=[25,70,380,550,610,1220,830,1450]
#numeros de datos
n = len(x)
#convertir en vectores
x = np.array(x)
y = np.array(y)

#sumatorias
sumx = sum(x)
sumy = sum(y)
sumx2 = sum(x**2)
sumy2 = sum(y**2)
sumxy = sum(x*y)

#promedio de datos
xm = sumx/n
ym = sumy/n
#valores de la recta
a1 = (sumx*sumy-n*sumxy)/((sumx)**2 - n*sumx2) #pendiente (m)
a0 = ym - a1*xm #interseccion con el eje (b)

#errores del metodo
st = sum((y - ym)**2)
sr = sum((y - a1*x - a0)**2)
syx = math.sqrt(sr/(n-2)) #error estand.
r = math.sqrt((st - sr)/st) #coef.correlacion
print("Error estandar: ", syx)
print("Coeficiente de correlacion: ", r)
print('Pendiente                    Interseccion con el eje X')
print(f'{a1:10} ==> {a0:10f}')

#graficar
plt.plot(x,y, 'o', label='Datos',color='red') #puntos
plt.plot(x, a1*x+a0, label='19.470x-234.285',color='green')
#titulo de ejes
plt.xlabel('Eje x')
plt.ylabel('Eje y')
plt.grid()#generar malla
plt.title('Regresion lineal')#titulo de grafica
plt.legend()#mostrar recuadro
plt.show()#mostrar grafica
```

2. Modelo de potencias

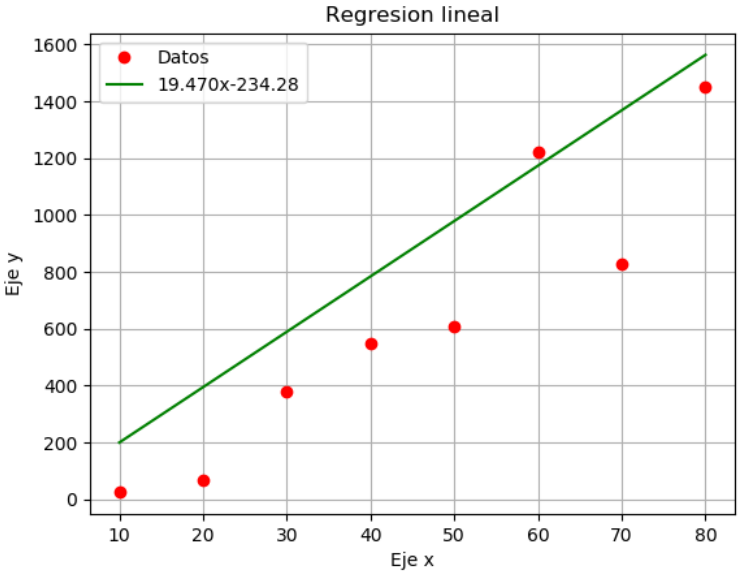


Figura 2.5: Linealización de potencias

```python
import math
import numpy as np
import matplotlib.pyplot as plt

#arreglo con valores
x=[10,20,30,40,50,60,70,80]
y=[25,70,380,550,610,1220,830,1450]
#numeros de datos
n = len(x)
#convertir en vectores
x = np.array(x)
y = np.array(y)
#sumatorias
sumx = sum(x)
sumy = sum(y)
sumx2 = sum(x**2)
sumy2 = sum(y**2)
sumxy = sum(x*y)

#promedio de datos
xm = sumx/n
ym = sumy/n
#valores de la recta
a1 = (sumx*sumy-n*sumxy)/((sumx)**2 - n*sumx2) #pendiente (m)
a0 = ym - a1*xm #interseccion con el eje (b)
#errores del metodo
st = sum((y - ym)**2)
sr = sum((y - a1*x - a0)**2)
syx = math.sqrt(sr/(n-2)) #error estand.
r = math.sqrt((st - sr)/st) #coef.correlacion
print("Error estandar: ", syx)
print("Coeficiente de correlacion: ", r)
print('Pendiente                    Interseccion con el eje X')
print(f'{a1:10} ==> {a0:10f}')

#graficar
plt.plot(x,y, 'o', label='Datos',color='red') #puntos
plt.plot(x, np.log(abs(a0))+a1*x, label='19.470x-234.28'
,color='green') #recta
#titulo de ejes
plt.xlabel('Eje x')
plt.ylabel('Eje y')
plt.grid()#generar malla
plt.title('Regresion lineal')#titulo de grafica
plt.legend()#mostrar recuadro
plt.show()#mostrar grafica
```

CAPÍTULO 3

ANEXOS

3.1. Anexo A: Evidencia de los código reportados

3.1.1. Ejercicio 17.5

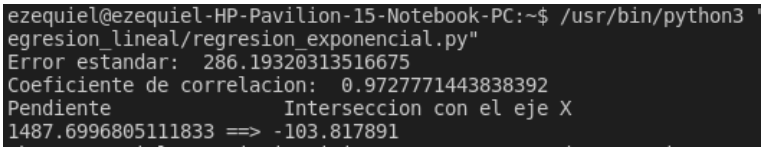


Figura 3.1: Método lineal

3.1.2. Ejercicio 17.9

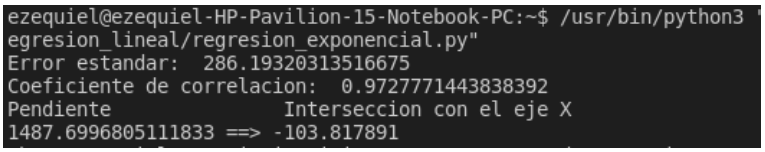


Figura 3.2: Método exponencial

3.1.3. Ejercicio 17.17

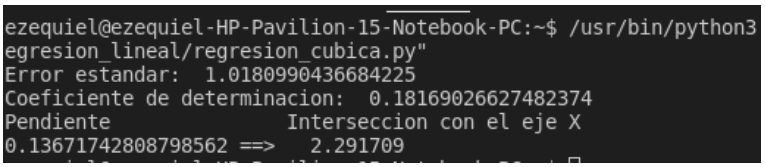


Figura 3.3: Método cúbico

3.1.4. Ejercicio 17.19

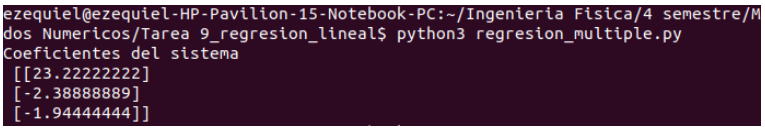


Figura 3.4: Método regresión múltiple

3.1.5. Ejercicio 17.28

1. Lineal

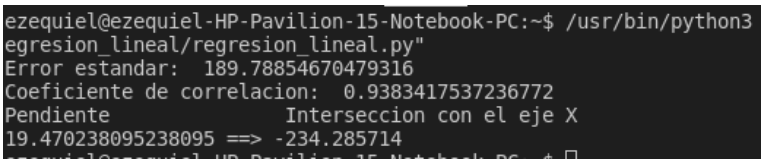


Figura 3.5: Modelo lineal

2. Potencias

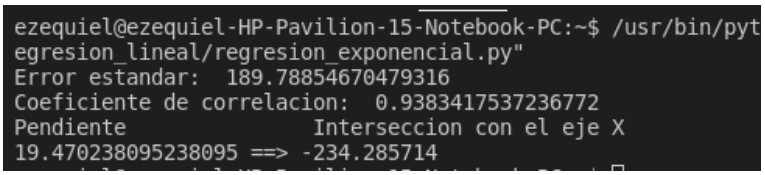


Figura 3.6: Modelo de potencias

CAPÍTULO 4

BIBLIOGRAFÍA

1. Chapra, S. C., Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: Mcgraw-hill.