Centro de Enseñanza Técnica Industrial

Plantel Colomos

Ingeniería en Desarrollo de Software

Inteligencia Artificial

REPORTE DE PRÁCTICA

IDENTIFICACIÓN DE LA PRÁCTICA

Práctica	**2**	**Nombre de la práctica**	**Regresión lineal multivariable**
Fecha	**03/02/2025**	**Nombre del profesor**	**Alma Nayeli Rodríguez Vázquez**
Nombre del estudiante		**Jesús Alberto Aréchiga Carrillo**	

OBJETIVO

El objetivo de esta práctica consiste en implementar el método de regresión lineal multivariable para predicción.

PROCEDIMIENTO

Realiza la implementación siguiendo estas instrucciones.

Implementa el método de regresión lineal multivariable en Python. Para ello, considera los siguientes requerimientos:

- Utiliza el set de datos del archivo “dataset_RegresionLinealMultivariable.csv”.
- Utiliza los siguientes valores para los parámetros iniciales:
  `a=vector de ceros, beta=0.8, iteraciones=600`
- Reporta el error `J` y el valor final del vector `a`. Además, reporta el valor de `h` para los siguientes datos de prueba
  - Dato de prueba 1: `x1= 3000, x2=4, y=539900`
  - Dato de prueba 2: `x1= 1985, x2=4, y=299900`
  - Dato de prueba 3: `x1= 1534, x2=3, y=314900`
- Comprueba tus resultados con los siguientes:

```
J= 2043280050.6028 a0= 340412.6596 a1= 110631.0503 a2= -6649.4743
Dato de prueba 1 x1= 3000 x2= 4 Salida correcta y= 539900 Predicción h= 472277.8551
Dato de prueba 2 x1= 1985 x2= 4 Salida correcta y= 299900 Predicción h= 330979.021
Dato de prueba 3 x1= 1534 x2= 3 Salida correcta y= 314900 Predicción h= 276933.0262
```

IMPLEMENTACIÓN

Agrega el código de tu implementación en Python aquí.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Cargar datos
data = pd.read_csv('dataset_RegresionLinealMultivariable.csv')
data.head()

#Calcular el tamaño
m = np.size(data, axis = 0)
```

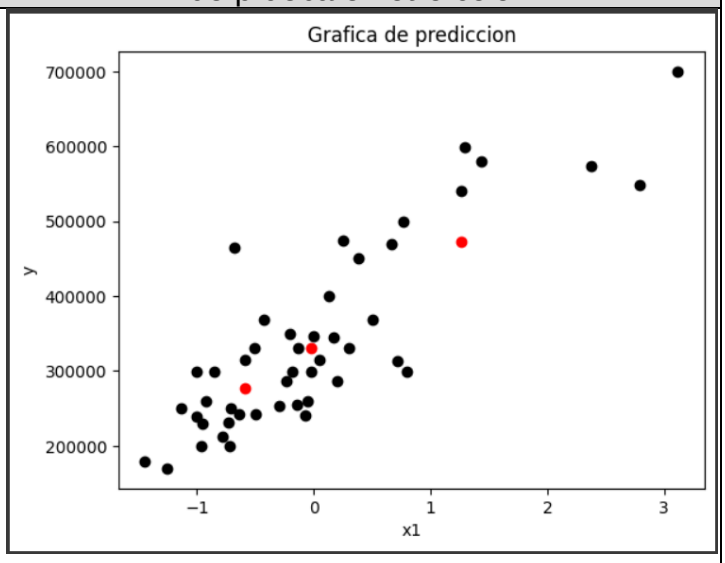
```
n = np.size(data, axis = 1) - 1

#Separar datos en X y Y
x = data
x = x.drop(columns=['y'])

y = data
y = y.drop(columns=['x1', 'x2'])

# Normalización de características (Estandarización)
media = np.mean(x, axis = 0)
sigma = np.std(x, axis = 0, ddof = 1)
x = (x - media) / sigma

#Agregar columna de unos a los datos X
m, n = np.shape(x)
x.insert(0, 'x0', np.ones((m, 1)))

#Inicializar el vector de parametros
a = np.ones((n + 1, 1))

#Inicializar parametros
beta = 0.8
iterMax = 600

#Crear los vectores J y h
J = np.zeros((iterMax, 1))
h = np.zeros((m, 1))

#Entrenamiento
for iter in range(iterMax):
  for i in range(m):
    h[i] = np.dot(a.transpose(), x.iloc[i, :])
  J[iter] = np.sum(np.power((h - y), 2)) / (2 * m)
  for j in range(n + 1):
    xj = np.mat(x[x.columns[j]])
    xj = xj.transpose()
    a[j] = a[j] - beta * (1 / m) * np.sum((h - y) * xj)

plt.plot(J)
plt.title('Grafica de convergencia')
plt.xlabel('Iteraciones')
plt.ylabel('Costo')
plt.show()

print('a = ', a, '\nJ = ', J[iter - 1])

#Crear datos de prueba
datosPrueba = pd.DataFrame({
    'x1': [3000, 1985, 1534],
    'x2': [4, 4, 3],
    'y' : [539900, 299900, 314900]
})
```

```
#Separar X y Y
datosPruebaX = datosPrueba.drop(columns=['y'])
datosPruebaY = datosPrueba.drop(columns=['x1', 'x2'])

#Normalizar los datos de prueba
datosPruebaX = (datosPruebaX - media) / sigma

datosPruebaX.insert(0, 'x0', np.ones((3, 1)))

#Prediccion de los datos de prueba
hPrueba = np.dot(datosPruebaX, a)

#Dibujar los datos de entrenamiento y los datos de prueba
plt.plot(x['x1'], y['y'], 'o', color = 'black')
plt.plot(datosPruebaX['x1'], hPrueba, 'o', color = 'red')
plt.title('Grafica de prediccion')
plt.xlabel('x1')
plt.ylabel('y')
plt.show()

#Imprimir todos los valores
print('J =', J[iter - 1], 'a0 =', a[0], 'a1 =', a[1], 'a2 =', a[2])
print('Dato de prueba 1: x1 =', datosPrueba['x1'][0], ', x2 =',
    datosPrueba['x2'][0], ', Salida y =', datosPrueba['y'][0],
    ', Prediccion h =', hPrueba[0])
print('Dato de prueba 2: x1 =', datosPrueba['x1'][1], ', x2 =',
    datosPrueba['x2'][1], ', Salida y =', datosPrueba['y'][1],
    ', Prediccion h =', hPrueba[1])
print('Dato de prueba 3: x1 =', datosPrueba['x1'][2], ', x2 =',
    datosPrueba['x2'][2], ', Salida y =', datosPrueba['y'][2],
    ', Prediccion h =', hPrueba[2])
```

Inteligencia Artificial

RESULTADOS

Agrega las imágenes con los resultados obtenidos en los espacios indicados.

Gráfica de convergencia	Gráfica de dispersión donde se aprecian los datos de entrenamiento en un color y los datos de prueba en otro color
	

Impresión de los valores de `J`, `a`, los datos de prueba `x` con la salida correcta `y` y su predicción `h`

```
#Imprimir todos los valores
print('J =', J[iter - 1], 'a0 =', a[0], 'a1 =', a[1], 'a2 =', a[2])
print('Dato de prueba 1: x1 =', datosPrueba['x1'][0], ', x2 =',
      datosPrueba['x2'][0], ', Salida y =', datosPrueba['y'][0],
      ', Prediccion h =', hPrueba[0])
print('Dato de prueba 2: x1 =', datosPrueba['x1'][1], ', x2 =',
      datosPrueba['x2'][1], ', Salida y =', datosPrueba['y'][1],
      ', Prediccion h =', hPrueba[1])
print('Dato de prueba 3: x1 =', datosPrueba['x1'][2], ', x2 =',
      datosPrueba['x2'][2], ', Salida y =', datosPrueba['y'][2],
      ', Prediccion h =', hPrueba[2])
```

```
J = [2.04328005e+09] a0 = [340412.65957447] a1 = [110631.05027885] a2 = [-6649.47427082]
Dato de prueba 1: x1 = 3000 , x2 = 4 , Salida y = 539900 , Prediccion h = [472277.85514636]
Dato de prueba 2: x1 = 1985 , x2 = 4 , Salida y = 299900 , Prediccion h = [330979.02101847]
Dato de prueba 3: x1 = 1534 , x2 = 3 , Salida y = 314900 , Prediccion h = [276933.02614885]
```

CONCLUSIONES

Escribe tus observaciones y conclusiones.
En la vida real, los valores de entrenamiento son más de uno, por eso es por lo que la regresión lineal multivariable es más factible que la univariable. Es necesario normalizar los datos de entrenamiento para que el algoritmo aprenda de manera más eficiente y estable, mejorando tanto la velocidad de convergencia como la calidad del modelo. De la misma manera, es necesario normalizar los valores de prueba o los valores reales en la práctica para que se puedan alinear con el entrenamiento y den resultados correctos.