Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario de IA II

REPORTE DE PRÁCTICA

IDENTIFICACIÓN DE LA PRÁCTICA

Práctica	**1**	**Nombre de la práctica**	**Regresión lineal univariable**
Fecha	**17/03/2024**	**Nombre del profesor**	**Alma Nayeli Rodríguez Vázquez**
Nombre del estudiante	**Ivan Barba Macias**		

OBJETIVO

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

PROCEDIMIENTO

Realiza la implementación siguiendo estas instrucciones.

Implementa el método de regresión lineal en Python y con la paquetería de sklearn. Para ello, considera los siguientes requerimientos:

- Utiliza el set de datos del archivo “dataset_RegresionLineal.csv”.
- No normalizar los datos
- Utiliza los siguientes valores para los parámetros iniciales:
  `a0=0 a1=0 beta=0.023 iteraciones=600`
- Reporta el error `J` y el valor final de `a0` y `a1`. Además, reporta el valor de `h` para el dato de prueba `x= 9.7687`, cuya salida correcta es `y= 7.5435`.
- Comprueba tus resultados con los siguientes:
  `J=4.4869 a0=-3.5657 a1=1.1599`
  `Dato de prueba x=9.7687. Salida correcta y= 7.5435. Predicción h=7.7648`

IMPLEMENTACIÓN

TODO EL CODIGO DE ESTA PRACTICA SE PUEDE ENCONTRAR EN:
https://github.com/IvanBM18/SeminarioIA2/tree/main/Regresion%20Lineal%20Univariable

Agrega el código de tu implementación en Python aquí.

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

DATASET_PATH = "./dataset_RegresionLineal.csv"
XTEST = 9.7687

class LinealRegresion:

    def __init__(self,ax : plt.Axes) -> None:
        self.weight = 0 #a1
```

```
        self.bias = 0 #a0
        self.learnRate = 0.023 #Beta
        self.iterationLimit = 600
        self.error = 0 #J
        self.ax = ax

        self.x : np.array
        self.y : np.array
        self.totalElements : int

    def loadDataset(self):

        dataset = pd.read_csv(DATASET_PATH)
        self.x = np.array(dataset['x'])
        self.y = np.array(dataset['y'])
        self.totalElements = np.size(self.x)

        self.ax.plot(self.x,self.y,'o',color = "yellow",mec = "black")

    def fit(self) -> np.array:
        m = self.totalElements
        error_list = []
        for it in range(self.iterationLimit):
            yPredicition = self.hypothesis()

            derivateW = (1 / m) * np.dot(self.x.T, yPredicition - self.y) #Forma m,
            derivateB = (1 / m) * np.sum(yPredicition - self.y)

            self.weight -= self.learnRate * derivateW
            self.bias -= self.learnRate * derivateB

            currError = (1 / (2 * m)) * np.sum(np.power((yPredicition - self.y), 2))
            error_list.append(currError)

            print(f"Iteration: {it}\tError: {currError}")
            self.plotPredicition()
        self.plotConvergence(np.array(error_list))

    def hypothesis(self):
        return np.dot(self.x,self.weight) + self.bias

    def plotPredicition(self,color : str = "yellow",visibility : int = 0.3):
        self.ax.plot(self.x,self.hypothesis(),color= color,alpha = visibility)

    def plotConvergence(self,error_list : np.array):
        fig, ax = plt.subplots()
        ax.grid()
```

```
        fig.suptitle('Grafica de convergencia')
        ax.plot(error_list,'r')
        ax.set_xlabel('Iteraciones')
        ax.set_ylabel('J (Error)')

    def predictData(self,x : int, y : int):
        predicted = self.bias + (self.weight * x)
        print(f'Predicted value for X={x} is {predicted}, Correct value is {y}')
        self.ax.plot(x,predicted,"o",color = "red",mec = "black")

if __name__ == "__main__":
    fig, ax = plt.subplots()
    ax.grid()
    fig.suptitle('Regresion Lineal')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')

    model = LinealRegresion(ax)
    model.loadDataset()
    model.plotPredicition('red',1)
    model.fit()
    model.plotPredicition('green',1)
    model.predictData(9.7687,7.5435)

    plt.show()
    print("End of program")
```

Agrega el código de tu implementación en Python con sklearn aquí.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

DATASET_PATH = "./dataset_RegresionLineal.csv"
XTEST = 9.7687

dataset = pd.read_csv(DATASET_PATH)
x = np.array(dataset['x']).reshape(-1,1)
y = np.array(dataset['y'])

plt.plot(x,y,'o',color = "yellow",mec = "black")
plt.xlabel('X')
plt.ylabel('Y')
```

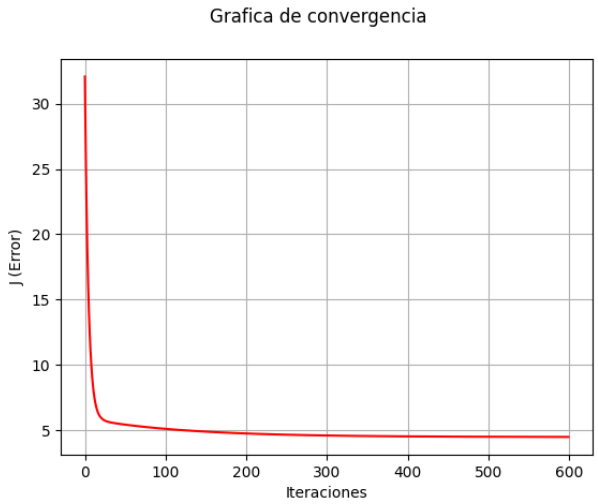
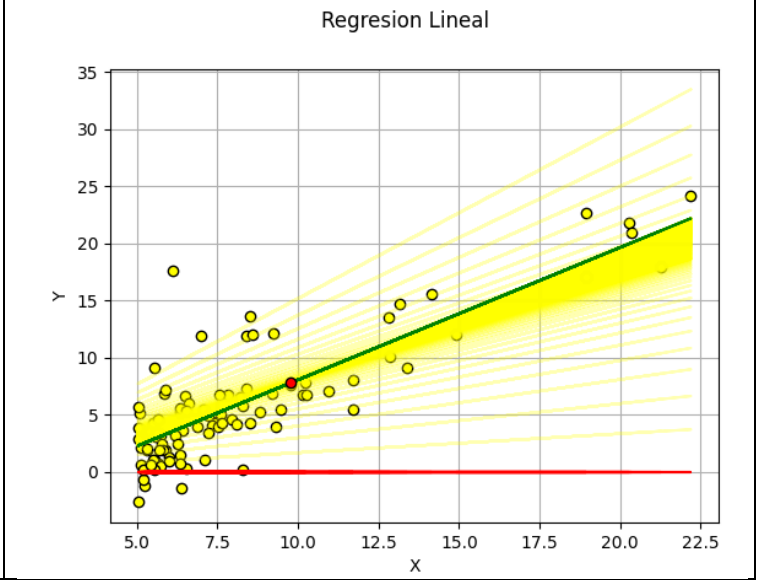
```
model = LinearRegression()
model.fit(x,y)  #Entrenamiento
predictedY = model.predict(x)  #Predicción
plt.plot(x,predictedY,"g")

predictedY = model.predict([[XTEST]])  #Predicción
plt.plot(XTEST,predictedY,"ro")
print(f'a0 = {model.intercept_}, a1 = {model.coef_}')
print(f'Valor predicho para el dato de prueba  X={XTEST}, es {predictedY[0]}')

plt.title('Regresión Lineal con SKLearn')
plt.show()
```

RESULTADOS EN PYTHON

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

Gráfica de convergencia	Gráfica del resultado final donde se aprecian los datos de entrenamiento, la recta del modelo inicial en rojo, las rectas del entrenamiento en amarillo y la final en verde)
	

Impresión de los valores de `J`, `a0`, `a1`, el dato de prueba `x` con la salida correcta `y` y su predicción `h`

```
Iteration: 599  Error: 4.4868898803743775
a0=-3.5670775888831403, a1=1.1600118333815632
Error actual, J=0
Valor predicho para x=9.7687 es h=7.7647300078713375, Valor correcto y=7.5435
```

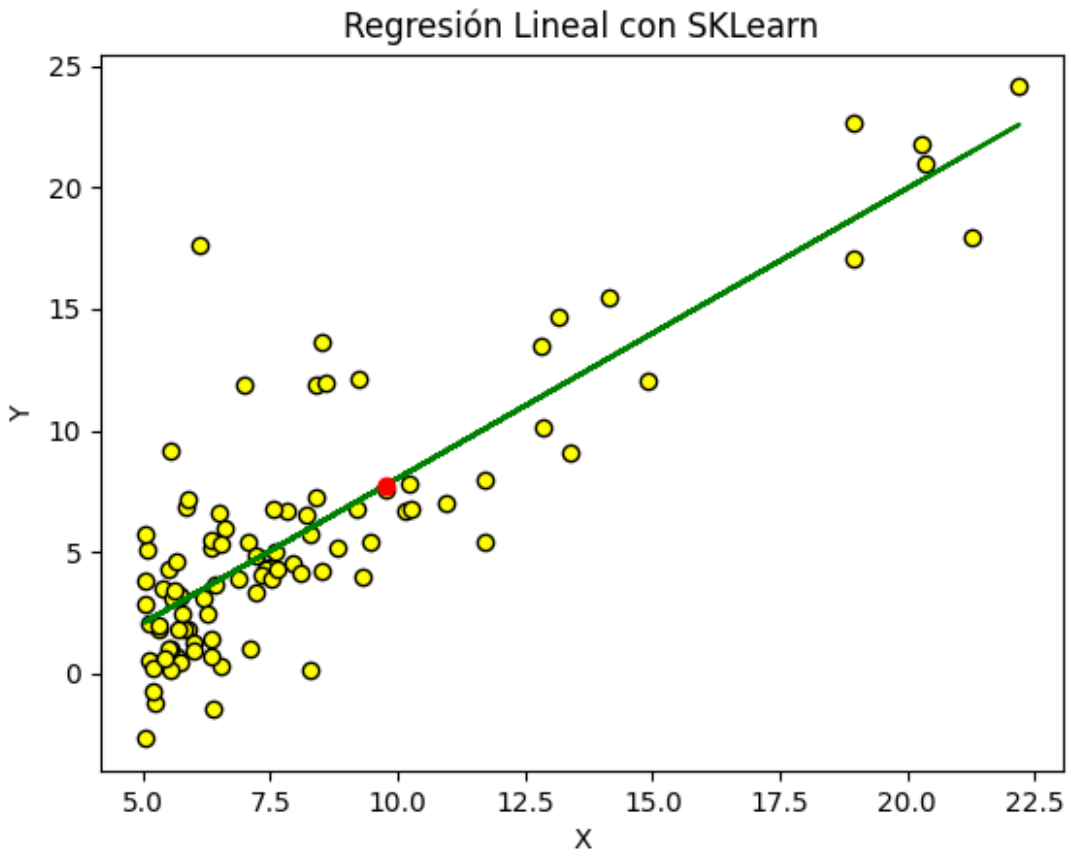
RESULTADOS EN PYTHON CON SKLEARN

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

Impresión de los valores de `a0`, `a1`, el dato de prueba `x` con la salida correcta `y` y su predicción `h`

```
a0 = -3.895780878311852, a1 = [1.19303364]
Valor predicho para el dato de prueba  X=9.7687, es 7.758606881683033
```

Gráfica del resultado final donde se aprecian los datos de entrenamiento y la recta del modelo final en verde)



Seminario de IA II

CONCLUSIONES

Escribe tus observaciones y conclusiones.
En lo personal, esta practica me resulto un repaso a las predicciones mediante regresión lineal. Ya que yo cursé el semestre pasado la catedra por lo que solo tuve que adaptar el código del semestre anterior al actual, lo que mas tuve que adaptar fue la gráfica de los datos y el nombre de los parámetros a0 (que conocía como bias o prejuicio), a1(Peso para una neurona con una entrada), y J (Error actual). En conclusión, en esta práctica aprendí a utilizar el modelo ya hecho por SKLearn, algo que no había tenido la oportunidad de utilizar, debido a que solo había tenido la oportunidad de crear los modelos desde 0, no utilizar los ya creados, lo que me abre la oportunidad a crear y experimentar con distintos modelos con mayor facilidad.