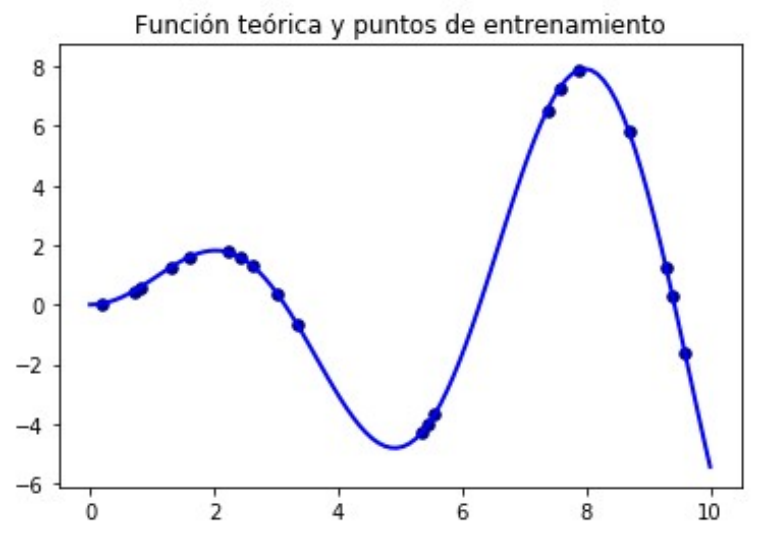
ML_REGRESION_04	Modelo de Regresión Polinómica	ML

El problema consiste en comprobar cuál es el grado del polinomio (N) que mejor se ajusta a un conjunto de valores generados según una función teórica. Y utilizar ese modelo para realizar predicciones.

Un modelo de regresión polinómico tiene la forma:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 + \ldots + \theta_N x_1^N$$

La función teórica f(x) = x * Sin (x) y los datos utilizados para el ajuste son:



Para realizar el ajuste de los modelos polinómicos se utiliza la función polyfit de la librería numpy.

SOLUCIÓN

Importar las librerías necesarias para realizar la práctica.

```
# importar las librerías
import numpy as np
import matplotlib.pyplot as plt
```

Definir la función teórica:

```
# Utilizamos una función teórica para generar los puntos
def f(x):
    """ función para aproximar mediante interpolación polinómica """
    return x * np.sin(x)
```

Generar los puntos X para representar la función:

```
# Generar los puntos X usados para representar la función
x_plot = np.linspace(0, 10, 100)
```

Generar de forma aleatoria los puntos X y las etiquetas y (ajustadas a la función teórica)

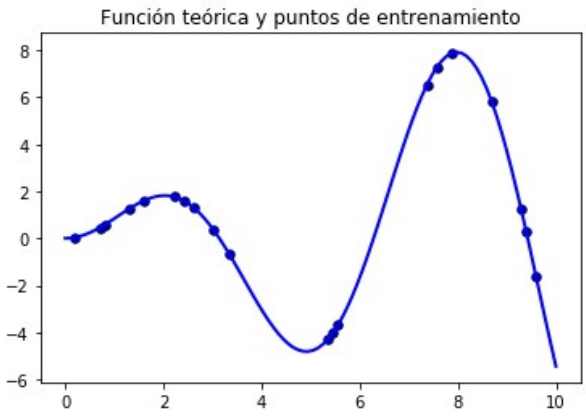
```
# Generar de forma aleatoria la muestra de puntos x y las etiquetas y
x = np.linspace(0, 10, 100)
rng = np.random.RandomState(0)
rng.shuffle(x)
x = np.sort(x[:20])
y = f(x)
```

Definir el color y grosor de línea para la visualización:

```
# crear la lista de colores
colors = ['red','orange', 'green']

# Definir la anchura de la linea a dibujar
lw = 2
```

Dibujar la gráfica teórica y los puntos entrenamiento:

```
# Dibujar la gráfica teórica f(x)
plt.plot(x_plot, f(x_plot), color='blue', linewidth=lw,label="Función
teórica")

# y los puntos utilizados para entrenamiento
plt.scatter(x, y, color='navy', s=30, marker='o', label="Puntos de
entrenamiento")

plt.title("Función teórica y puntos de entrenamiento")
plt.show()
```



Realizamos la regresión polinómica utilizando la librería polyfit de Numpy:

```
# REGRESION POLINOMICA (utilizando la función polyfit de la librería numpy)
print("Ajuste de Regresión polinómica")
# Ajuste para ecuaciones de grado 3, 4 y 5
# Polinomio de grado 3: y = t0+t1*X+t2*X^2+t3*X^3
# Polinomio de grado 4: y = t0+t1*X+t2*X^2+t3*X^3+t4*X^4
# Polinomio de grado 5: y = t0+t1*X+t2*X^2+t3*X^3+t4*X^4+t5*X^5
```

Dibujamos de nuevo la gráfica teórica y los puntos de entrenamiento:

```
# Dibujar la gráfica teórica f(x)
plt.title("Regresión polinómica")
plt.plot(x_plot, f(x_plot), color='blue', linewidth=lw,label="Función
teórica")

# y los puntos utilizados para entrenamiento
plt.scatter(x, y, color='navy', s=30, marker='o', label="Puntos de
entrenamiento")
```

Realizamos el ajuste polinómico para los grados N=3,4 y 5, y escribimos la ecuación de cada polinomio y el error cuadrático medio (ECM)

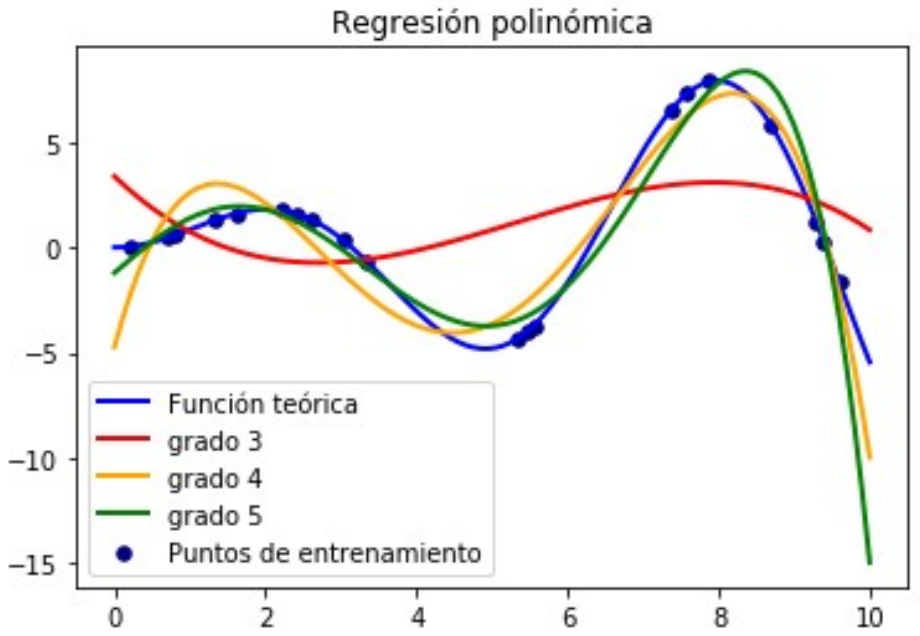
```
# Realizar el ajuste de los polinomios de grados 3,4 y 5
for count, degree in enumerate([3,4,5]):
    # Ajuste del polinomio de grado 'degree' a los datos de entrenamiento x,y
    coeffs = np.polyfit(x,y,deg=degree)
    # Determinar y escribir la forma del polinomio
    p = np.poly1d(np.polyfit(x, y, deg=degree), variable='X')
    print("Polinomio de grado ",degree," : ")
    print(p)
    print("")

    y_pred = np.polyval(np.poly1d(coeffs), x)
    print("Error cuadrático medio (ECM): ",1/20*(sum((y-y_pred)**2)))
    print("")

    # Dibujar la gráfica del polinomio
    # Calcular la y de la gráfica 'y_plot'
    y_plot = np.polyval(np.poly1d(coeffs), x_plot)

    # Dibujar la gráfica
    plt.plot(x_plot, y_plot, color=colors[count], linewidth=lw,label="grado
%d" % degree)
```

```
Ajuste de Regresión polinómica
```

```
Polinomio de grado  3  :
          3          2
-0.05329 X + 0.8502 X - 3.427 X + 3.377

Error cuadrático medio (ECM):  9.59074349743502

Polinomio de grado  4  :
          4        3         2
-0.06679 X + 1.25 X - 7.211 X + 13.38 X - 4.741

Error cuadrático medio (ECM):  1.5359390833908724

Polinomio de grado  5  :
           5          4          3          2
-0.009309 X + 0.1626 X - 0.7565 X + 0.1762 X + 3.023 X - 1.207

Error cuadrático medio (ECM):  0.7573371110922194
```

Dibujar el gráfico completo

```
# Leyenda del gráfico
plt.legend(loc='lower left')
# Dibujar el gráfico
plt.show()
```



Utilizar el modelo de regresión polinómica para predecir valores:

```
# Predecir para un valor de X=6 con el modelo de regresión polinómica de
grado 5
coeffs = np.polyfit(x,y,deg=5)
```

```
y_pred = np.polyval(np.poly1d(coeffs), 6)
print("Predicción para X=6: y=",y_pred)

Predicción para X=6: y= -1.8108241323328411
```