Proyecto 1 - Regresión

Alex Loja
Ciencia de la Computación
UTEC
alex.loja@utec.edu.pe

Tania Barreda
Bioingeniería
UTEC
tania.barreda@utec.edu.pe

Jean Ycarrayme Valdivia
Bioingeniería
UTEC
jean.ycarrayme@utec.edu.pe

Melisa Rivera
Ciencia de la Computación
UTEC
melisa.rivera@utec.edu.pe

Melanie Cortez
Ciencia de la Computación
UTEC
melanie.cortez@utec.edu.pe

Fernando Padilla
Ciencia de la Computación
UTEC
fernando.padilla@utec.edu.pe

Keywords–Regresión, Serie Temporal, Regresión Polinomial, Regularización

1. INTRODUCCIÓN

Recientemente, las técnicas de aprendizaje automático (ML) han ganado amplia difusión, ya que se han aplicado con éxito en varios campos de investigación como en el cuidado de la salud [1], finanzas [2], vehículos autónomos [3], procesamiento natural de lenguaje [4] y reconocimiento de imágenes [5]. La regresión, es una técnica de aprendizaje automático utilizada principalmente para predicción y previsión, en algunos casos, para determinar relaciones causales entre variables independientes y dependientes, aunque por sí sola solo muestra relaciones entre una variable dependiente y un conjunto fijo de variables [6]. La regresión lineal es un método sencillo para analizar relaciones lineales entre variables continuas, lo que la hace ideal para muestras pequeñas y de fácil interpretación [7]. A diferencia de la regresión lineal, las técnicas de aprendizaje automático para regresión no lineal nos permiten modelar relaciones más complejas entre múltiples variables o un conjunto de datos más disperso [8].

En este proyecto nos centraremos en la regresión no lineal, específicamente en la regresión polinomial. La regresión polinómica es un tipo de análisis de regresión que utiliza un modelo polinómico de grado n para describir la relación entre variables independientes y dependientes. Es un caso especial de la regresión múltiple lineal (MLR) en el que la ecuación polinómica de los datos se ajusta a la interacción curvilínea entre las variables dependientes e independientes. El modelo polinómico se expresa como: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_h x^h + \epsilon$, donde h es el grado del polinomio [6].

2. DATASET

A. Descripción

El conjunto de datos propuesto consiste de una serie temporal unidimensional, la cual contiene una secuencia de valores registrados sobre un determinado periodo de tiempo. Cada punto representa el valor observado en ese intervalo de tiempo específico. En este caso tenemos 100 datos registrados en *datos.txt* y, al no tener mayor información temporal, asumiremos que empezamos en un instante de tiempo $t = 0$ hasta llegar a $t = 99$.

Visualizamos la data con *scatterplot* y graficamos la serie temporal con *lineplot*.

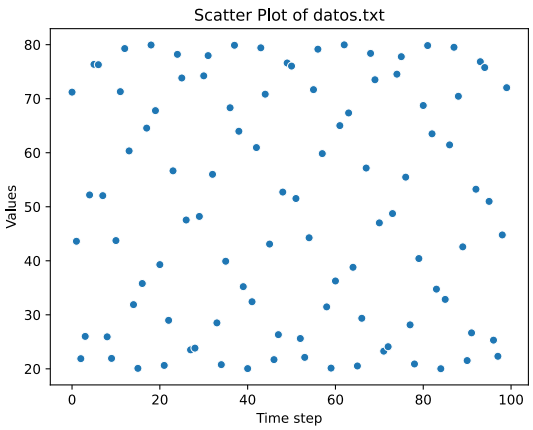

Fig 2.1: Gráfico de puntos con los valores de datos.txt por cada instante de tiempo.

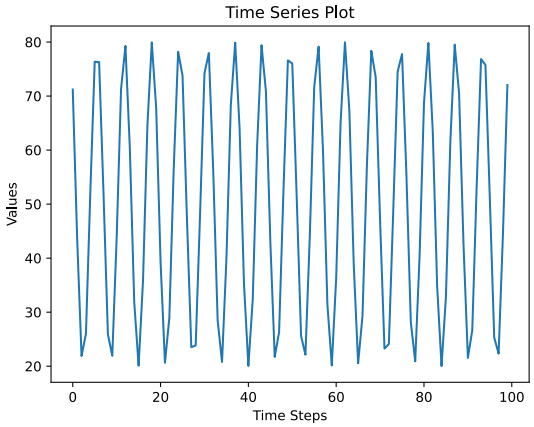

Fig 2.2: Gráfico de la serie temporal a partir de los datos.txt por cada instante de tiempo.

Como observamos, los puntos están ampliamente distribuidos y no siguen algún comportamiento ascendente o descendente característico de una función lineal. Por ende, la carencia de alguna relación evidente entre los datos nos sugiere que no siguen una tendencia lineal. Entonces, debido a la alta dispersión de los datos y a la ausencia de su similitud con otros modelos que faciliten su regresión paramétrica o con alguna forma definida, la regresión polinómica puede ser una herramienta útil debido a su factor de ajuste en el grado de su polinomio para garantizar una mejor aproximación a este tipo de distribuciones complejas, aunque previniendo el riesgo de sobreajuste de los datos por el grado excesivo del polinomio [9].

B. Análisis

Estacionaridad

Para comprender mejor el concepto de estacionaridad, podemos imaginar que los datos estacionarios no muestran una tendencia de crecimiento o decrecimiento constante a lo largo del tiempo, incluso si parecen aleatorios a primera vista. Sin embargo, lo que realmente define a estos datos es que mantienen un patrón consistente en sus propiedades estadísticas, como la media, la varianza y la autocorrelación.

- En el caso de una serie temporal, donde el tiempo se toma como la variable independiente, verificar la estacionaridad es fundamental para una regresión efectiva y garantizar que las predicciones no estén sesgadas.
- La regresión polinómica suele asumir que los datos son estacionarios; para modelar una serie temporal no estacionaria, se realiza el proceso de media móvil autorregresiva integrada (ARIMA) [10].

Una serie temporal Y_t se considera **estacionaria** si satisface las siguientes condiciones [11]:

1) **Media:** La media de Y_t permanece constante a lo largo del tiempo. Matemáticamente:
$$E(Y_t) = \mu, \ \forall t$$
donde μ es una constante escalar.
2) **Varianza:** La varianza de Y_t es constante a lo largo del tiempo. Matemáticamente:
$$\text{Var}(Y_t) = \sigma^2, \ \forall t$$
donde σ^2 es una constante escalar.
3) **Covarianza:** La covarianza entre Y_t y Y_{t-s} depende únicamente del retraso s, no del tiempo específico t. Matemáticamente:
$$\text{Cov}(Y_t, Y_{t-s}) = \gamma_s$$
donde γ_s es una función del retraso s.

Aunque métodos informales, como la simple observación, pueden sugerir que los datos son estacionarios, esto no es suficiente para una confirmación robusta. Para comprobar las condiciones planteadas en [11] y determinar si una serie es estacionaria se propone **Prueba de Dickey-Fuller**. Si bien esta prueba solo detecta la estacionaridad en aproximadamente el 30% de los casos, sigue siendo una herramienta valiosa. La prueba se basa en la hipótesis nula de que los datos no son estacionarios. Si el p-valor resultante es menor a 0.05, podemos rechazar la hipótesis nula y concluir que los datos son estacionarios.

Hipótesis:

- **Hipótesis nula ($\mathbf{H}_0$)**: La serie temporal tiene una raíz unitaria (no es estacionaria).
- **Hipótesis alternativa ($\mathbf{H}_1$)**: La serie temporal no tiene una raíz unitaria (es estacionaria).

La prueba se basa en el siguiente modelo de regresión:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \epsilon_t \tag{1}$$

donde:

- $\Delta y_t = y_t - y_{t-1}$ es la diferencia primera de la serie.
- α es un término constante.
- βt es un término de tendencia (opcional, dependiendo de la especificación).
- γ es el coeficiente que indica la presencia de una raíz unitaria.
- ϵ_t es el término de error.

Estadístico de prueba
Se calcula el estadístico de prueba para el coeficiente γ. Este valor se compara con los valores críticos de la distribución de Dickey-Fuller para determinar la significancia.

Decisión
Compara el estadístico de prueba con los valores críticos para decidir si se rechaza o no la hipótesis nula. Si el estadístico es menor que el valor crítico, se rechaza la hipótesis nula, indicando que la serie es estacionaria.

Para comprobar si los datos que tenemos son estacionarios, usamos la función *adfuller* de la biblioteca *statsmodels*. Obteniendo los siguientes resultados:

```
ADF Statistic: -32859592747.05899
p-value: 0.0
Critical Values:
   1%: -3.4989097606014496
   5%: -2.891516256916761
   10%: -2.5827604414827157
The time series is stationary (reject the null hypothesis).
```

Dado que el valor del estadístico ADF (-32,859,592,747.05899) es mucho menor que los valores críticos en los niveles del 1%, 5% y 10%, y el valor p es 0.0, se rechaza la hipótesis nula. Por lo tanto, la serie temporal es estacionaria.

3. METODOLOGÍA

A. Regresión polinomial

En la regresión polinomial, la relación entre la variable independiente x y la variable dependiente y se modela como un polinomio. La ecuación general está dada por:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon$$

En forma matricial, esto puede escribirse como:

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Donde:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

es el vector $n \times 1$ de valores observados,

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}$$

es la matriz $n \times (n+1)$ de variables independientes (también conocida como la matriz de Vandermonde),

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$

es el vector $(n+1) \times 1$ de coeficientes, y

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

es el vector $n \times 1$ de errores.

Estimación por Mínimos Cuadrados

Para estimar los coeficientes $\boldsymbol{\beta}$, minimizamos la suma de los cuadrados de las diferencias entre los valores observados y los valores predichos. Esto se expresa como:

$$\min_{\boldsymbol{\beta}} \|\mathbf{y} - X\boldsymbol{\beta}\|^2$$

La solución a este problema de minimización se obtiene utilizando la ecuación normal:

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

Ejemplo: Ajuste de un Polinomio Cuadrático

Consideremos ajustar un polinomio cuadrático ($n = 2$) a los siguientes datos:

x	y
1	1.2
2	2.8
3	7.4
4	13.5
5	25.2

El modelo cuadrático es:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

En forma matricial:

$$\mathbf{y} = \begin{bmatrix} 1.2 \\ 2.8 \\ 7.4 \\ 13.5 \\ 25.2 \end{bmatrix}, \quad X = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \\ 1 & 5 & 5^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix}$$

La ecuación normal para estimar los coeficientes es:

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

Ahora, calculemos los componentes paso a paso.

Paso 1: Calcular $X^T X$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} = \begin{bmatrix} 5 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix}$$

Paso 2: Calcular $X^T \mathbf{y}$

$$X^T \mathbf{y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 \end{bmatrix} \begin{bmatrix} 1.2 \\ 2.8 \\ 7.4 \\ 13.5 \\ 25.2 \end{bmatrix} = \begin{bmatrix} 50.1 \\ 178.6 \\ 779.0 \end{bmatrix}$$

Paso 3: Resolver para $\boldsymbol{\beta}$

Finalmente, resolvemos para $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

$$\beta = \begin{bmatrix} 2.56 & -3.3 & 0.5 \\ -3.3 & 2.67 & -0.43 \\ 0.5 & -0.43 & 0.07 \end{bmatrix} \begin{bmatrix} 50.1 \\ 178.6 \\ 779 \end{bmatrix} = \begin{bmatrix} -71.624 \\ -23.438 \\ 2.782 \end{bmatrix}$$

Finalmente

$$y = -71.624 - 23.438x + 2.782x^2$$

B. Función de perdida

Dado un modelo polinomial de la forma:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon$$

La predicción $\hat{y}_i$ para un punto de datos i es:

$$\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_n x_i^n$$

El residuo o error para cada punto de datos es:

$$\epsilon_i = y_i - \hat{y}_i$$

La suma de los errores al cuadrado (SSE) es:

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

En forma matricial, la SSE puede escribirse como:

$$\text{SSE} = (\mathbf{y} - X\boldsymbol{\beta})^T (\mathbf{y} - X\boldsymbol{\beta})$$

Donde:

- $\mathbf{y}$ es el vector de valores observados,
- X es la matriz de diseño,
- $\boldsymbol{\beta}$ es el vector de coeficientes.

Considerando nuevamente el conjunto de datos anterior.

El modelo cuadrático es:

$$\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

Los residuos o errores son:

$$\boldsymbol{\epsilon} = \mathbf{y} - \hat{\mathbf{y}} = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ y_3 - \hat{y}_3 \\ y_4 - \hat{y}_4 \\ y_5 - \hat{y}_5 \end{bmatrix}$$

La suma de los errores al cuadrado (SSE) es:

$$\text{SSE} = \sum_{i=1}^{5} (y_i - \hat{y}_i)^2$$

O en forma matricial:

$$\text{SSE} = (\mathbf{y} - X\boldsymbol{\beta})^T (\mathbf{y} - X\boldsymbol{\beta})$$

El objetivo es minimizar esta suma para encontrar los valores óptimos de β_0, β_1 y β_2.

C. Técnicas de regularización

1) L1: La regularización L1 (también conocida como Lasso) introduce un término de penalización en la función de pérdida con el objetivo de reducir la magnitud de los coeficientes de los parámetros del modelo. Este enfoque es especialmente útil para evitar el sobreajuste (*overfitting*) y puede llevar a que algunos coeficientes se vuelvan exactamente cero, simplificando el modelo.

Función de pérdida con regularización L1

La regularización L1 modifica la función de pérdida añadiendo una penalización basada en la suma de los valores absolutos de los coeficientes β_j. La nueva función de pérdida se define como:

$$\text{L1 loss} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p} |\beta_j|$$

Donde:

- $\lambda \geq 0$ es el parámetro de regularización que controla la magnitud de la penalización.
- β_j son los coeficientes del modelo.

Este término de penalización induce un sesgo hacia soluciones en las que los coeficientes β_j sean pequeños o incluso cero, lo que puede simplificar el modelo y reducir el sobreajuste.

Gradiente de la función de pérdida con regularización L1

Para optimizar la función de pérdida con regularización L1, se utiliza el método del descenso por gradiente. El gradiente de la función de pérdida respecto a cada coeficiente β_j tiene dos componentes:

$$\frac{\partial}{\partial \beta_j} \text{L1 loss} = -2 \sum_{i=1}^{n} (y_i - \hat{y}_i)\, x_i^j + \lambda \cdot \text{sign}(\beta_j)$$

Donde $\text{sign}(\beta_j)$ es la función signo, que toma los valores:

$$\text{sign}(\beta_j) = \begin{cases} 1 & \text{si } \beta_j > 0 \\ -1 & \text{si } \beta_j < 0 \\ 0 & \text{si } \beta_j = 0 \end{cases}$$

El término $\lambda \cdot \text{sign}(\beta_j)$ actúa como una fuerza que empuja los coeficientes hacia cero, siendo más fuerte cuanto mayor sea el valor de λ.

Propiedades Matemáticas de L1

Algunas propiedades clave de la regularización L1:

- Sparsidad: La regularización L1 tiende a generar modelos ”esparsos”, es decir, con muchos coeficientes β_j iguales a cero. Esto significa que el modelo selecciona automáticamente las características más relevantes.
- Penalización absoluta: A diferencia de la regularización L2 (Ridge), que penaliza los coeficientes al cuadrado, L1 penaliza los valores absolutos de los coeficientes, lo que permite que algunos coeficientes se reduzcan exactamente a cero.
- Dependencia de λ: El parámetro λ controla la fuerza de la penalización. Si $\lambda = 0$, no hay regularización y el modelo es equivalente a la regresión polinomial estándar. Si λ es muy grande, todos los coeficientes tienden a ser cero.

Minimización de la función de pérdida

La minimización de la función de pérdida con regularización L1 requiere resolver el siguiente problema de optimización:

$$\min_{\boldsymbol{\beta}} \left(\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right)$$

Esto implica encontrar los coeficientes $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$ que minimicen tanto el error de predicción como la penalización sobre los coeficientes. Este problema se puede resolver usando técnicas de optimización numérica como el descenso por gradiente.

La actualización de los coeficientes en cada iteración del descenso por gradiente es:

$$\beta_j^{(t+1)} = \beta_j^{(t)} - \eta \left(-2 \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right) x_i^j + \lambda \cdot \text{sign}(\beta_j^{(t)}) \right)$$

Donde η es la tasa de aprendizaje, y $\beta_j^{(t)}$ es el valor del coeficiente β_j en la iteración t.

Dado el mismo conjunto de datos. El modelo cuadrático sin regularización es:

$$\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

La función de pérdida con regularización L1 es:

$$\text{L1 loss} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{2} |\beta_j|$$

Esto se traduce en:

$$\text{L1 loss} = \sum_{i=1}^{5} (y_i - \hat{y}_i)^2 + \lambda \left(|\beta_0| + |\beta_1| + |\beta_2| \right)$$

Aplicando un valor de $\lambda = 0.1$, la penalización sería:

$$0.1 \left(|\beta_0| + |\beta_1| + |\beta_2| \right)$$

El objetivo es minimizar tanto los errores como la suma de los valores absolutos de los coeficientes. Esto reduce el sobreajuste y simplifica el modelo, haciendo que algunos coeficientes se vuelvan cero.

2) L2: La regularización L2 (conocida como Ridge) es una técnica utilizada para evitar el sobreajuste (*overfitting*) en los modelos de regresión. A diferencia de la regularización L1, que penaliza los valores absolutos de los coeficientes, L2 penaliza los cuadrados de los coeficientes. Esto conduce a soluciones con coeficientes más pequeños, aunque no necesariamente cero, haciendo que el modelo sea más robusto y menos sensible a las fluctuaciones en los datos.

Función de pérdida con regularización L2

En la regularización L2, se introduce una penalización basada en los cuadrados de los coeficientes del modelo. La nueva función de pérdida se define como:

$$\text{L2 loss} = \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=0}^{p} \beta_j^2$$

Donde:

- $\lambda \geq 0$ es el parámetro de regularización que controla la magnitud de la penalización.
- β_j son los coeficientes del modelo.

El término $\lambda \sum_{j=0}^{p} \beta_j^2$ introduce una penalización en la magnitud de los coeficientes, lo que evita que estos se vuelvan demasiado grandes. A diferencia de la regularización L1, L2 no fuerza a que los coeficientes sean exactamente cero, sino que los reduce de manera continua.

Gradiente de la función de pérdida con regularización L2

Para minimizar la función de pérdida con regularización L2, se utiliza el método del descenso por gradiente. El gradiente de la función de pérdida con respecto a cada coeficiente β_j es:

$$\frac{\partial}{\partial \beta_j}\text{L2 loss} = -2\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)x_i^j + 2\lambda\beta_j$$

El término adicional $2\lambda\beta_j$ reduce los coeficientes β_j conforme aumenta λ, pero nunca los fuerza a ser exactamente cero. La regularización L2 introduce una contracción continua de los coeficientes, manteniéndolos pequeños pero presentes en el modelo.

Minimización de la función de pérdida

La minimización de la función de pérdida regularizada con L2 requiere resolver el siguiente problema de optimización:

$$\min_{\boldsymbol{\beta}}\left(\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 + \lambda\sum_{j=0}^{p}\beta_j^2\right)$$

Este problema se puede resolver utilizando algoritmos de optimización, como el descenso por gradiente. En cada iteración, los coeficientes β_j se actualizan siguiendo la regla:

$$\beta_j^{(t+1)} = \beta_j^{(t)} - \eta\left(-2\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)x_i^j + 2\lambda\beta_j^{(t)}\right)$$

Donde η es la tasa de aprendizaje y $\beta_j^{(t)}$ es el valor del coeficiente en la iteración t.

3) Comparación entre L1 y L2: Desde un punto de vista matemático, L1 y L2 tienen comportamientos distintos:

- L1 (Lasso): Penaliza los valores absolutos de los coeficientes $|\beta_j|$, lo que puede llevar a que algunos coeficientes se reduzcan exactamente a cero, produciendo un modelo más esparso.
- L2 (Ridge): Penaliza los cuadrados de los coeficientes β_j^2, lo que tiende a reducir todos los coeficientes, pero sin eliminarlos completamente.

La regularización L2 es útil cuando se desea controlar el tamaño de los coeficientes sin eliminar características por completo, mientras que L1 es más apropiada cuando se busca un modelo más esparso con selección de características.

D. Mean Squared Error (MSE)

El MSE es una métrica utilizada para evaluar la precisión de un modelo de regresión. El MSE mide el promedio de los cuadrados de los errores, donde un error es la diferencia entre el valor real y el valor predicho [12]. Matemáticamente, el MSE se define como:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

donde:

- n es el número de observaciones,
- y_i es el valor real de la i-ésima observación,
- $\hat{y}_i$ es el valor predicho para la i-ésima observación.

El MSE proporciona una medida de la magnitud promedio de los errores cuadráticos en un conjunto de predicciones. Un MSE más bajo indica un mejor ajuste del modelo a los datos [12].

4. IMPLEMENTACIÓN

Colab: https://colab.research.google.com/drive/1SVXCqpyp9lsyHJqjv_GNsdAswfMGwDLn?usp=sharing

Para la parte de la implementación, escribimos una clase llamada PolynomialRegression. En esta clase definimos las funciones necesarias para llevar a cabo la regresión polinómica.

```
class PolynomialRegression:
    def __init__(self, degree, learning_rate, iterations, alpha=0.1)

    def transform(self, X)

    def normalize(self, X)

    def fit(self, X, y)

    def fit_with_l1(self, X, y)

    def fit_with_l2(self, X, y)

    def predict(self, X)
```

Primero, tomamos como parámetros el grado de la función, la taza de aprendizaje, el número de iteraciones y el valor de penalización. Posteriormente, en transform tomamos como entrada X y creamos la matriz de Vandermonde. Luego, en nomalize, calculamos la media y la desviación estándar para normalizar los datos en X. Estos pasos se desarrollan en nuestras tres funciones fit.

Definimos tres funciones:

- *fit(X, y)*: primero calculamos las matrices $X^T X$ y $X^T y$. Como el dataset proporcionado no es grande podemos dessarrollar directamente para encontrar los valores de β, si este no es el caso tendriamos que iterar aplicando la tasa de aprendizaje.
- *fit_with_l1(X, y)*: realiza una regresión lineal con regularización L1 (Lasso) utilizando descenso por coordenadas. Después de aplicar la transformación y normalización inicializa el el vector de pesos W. Finalmente, actualiza iterativamente cada peso usando descenso por coordenadas, aplicando regularización L1 (controlada por alpha) a todos los pesos excepto al término de sesgo.
- *fit_with_l2(X, y)*: Realiza una regresión lineal con regularización L2 (Ridge). Calcula dos productos matriciales: `XtX` que es $X^T \cdot X$ y `Xty` que es $X^T \cdot y$. A continuación, añade un término de regularización L2 mediante la matriz de identidad `reg_term`, donde el primer valor (correspondiente al término de sesgo) no se regulariza. Finalmente, resuelve la ecuación normal $(X^T X + \alpha I)W = X^T y$ para encontrar los pesos `W` que minimizan la función de costo con regularización L2.

5. EXPERIMENTACIÓN

Como la regresión polinómica depende en gran medida del grado. Se experimentó con distintos grados del polinomio (10, 40, 100) y con tres enfoques: sin regularización, con regularización L1 y con regularización L2.
Los gráficos generados mostraron cómo el ajuste del polinomio varía según el grado y el tipo de regularización.

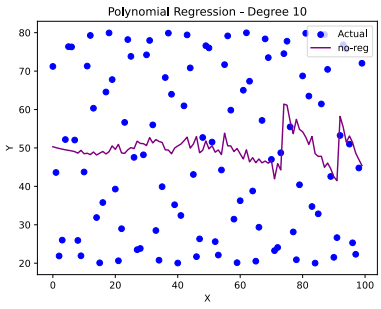

Fig 5.1: no-reg-10

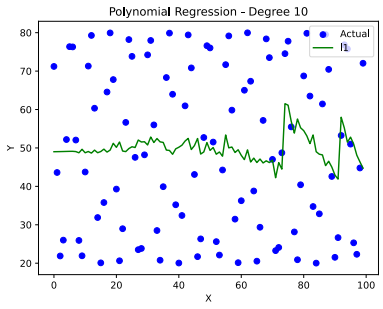

Fig 5.2: l1-10

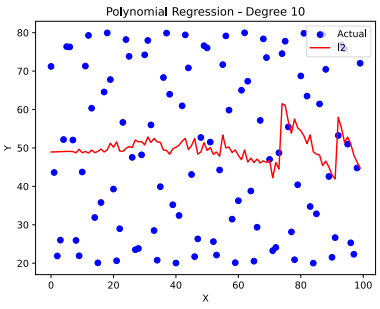

Fig 5.3: l2-10

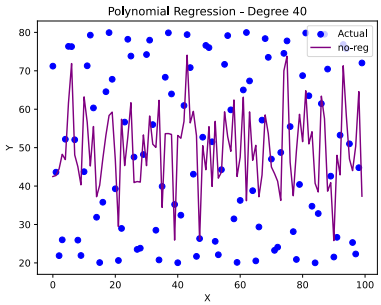

Fig 5.4: no-reg-40

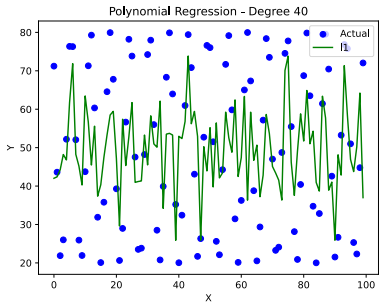

Fig 5.5: l1-40

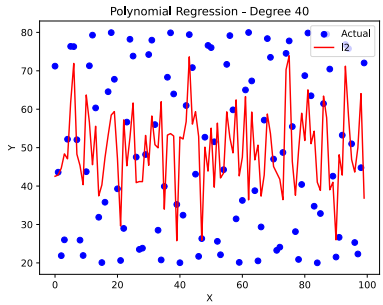

Fig 5.6: l2-40

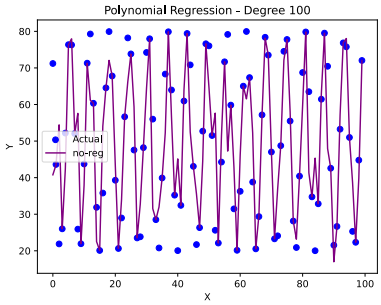

Fig 5.7: no-reg-100

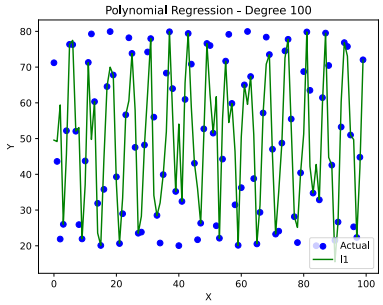

Fig 5.8: l1-100

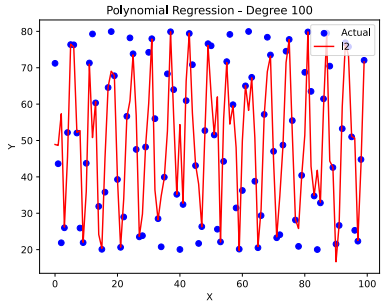

Fig 5.9: l2-100

También se calculó el MSE para comparar los experimentos realizados.

Grado	Sin Regularización	Regularización L1 (Lasso)	Regularización L2 (Ridge)
10	443.5016616148007	444.56987346497175	444.5538175678486
40	319.89644548677126	328.64375084132394	328.5296121954328
100	9.432633851127381e-06	24.90225730205064	28.004391931984838

Table 5.1: Resultados del MSE para diferentes grados y métodos de regularización

Para calcular el tiempo de ejecución se uso la función *timeit* y se calculó el promedio de ejecutar 100 veces las funciones.

Función	degree = 10		degree = 40		degree = 100	
	Fit	Predict	Fit	Predict	Fit	Predict
fit(X, y)	0.00017	0.00012	0.00125	0.00025	0.00390	0.00065
fit_with_l1(X, y)	0.16672	0.00011	0.62633	0.00018	15.82854	0.00060
fit_with_l2(X, y)	0.00022	0.00012	0.00132	0.00018	0.00375	0.00066

Table 5.2: Tiempo de ejecución de las funciones para diferentes grados

6. Discusión

Se experimentó con distintos grados del polinomio (10, 40, 100) y con tres enfoques: sin regularización, con regularización L1 y con regularización L2.

Análisis de la Regresión

Los gráficos generados mostraron cómo el ajuste del polinomio varía según el grado y el tipo de regularización.

- **Grado 10**: El modelo no capturó las tendencias generales, el ajuste fue pobre y con alta varianza.
- **Grado 40**: Ajuste más cercano a los datos.
- **Grado 100**: Ajuste mucho más preciso y que captura las tendencias generales.

Análisis del MSE

Los resultados sugieren que aumentar el grado del polinomio mejora el ajuste del modelo, como se refleja en la reducción del MSE. Sin embargo, esto puede llevar a sobreajuste, especialmente en el caso del polinomio de grado 100 sin regularización.

Las técnicas de regularización L1 y L2 mitigaron este riesgo al penalizar los coeficientes grandes, reduciendo así el riesgo de sobreajuste, aunque el MSE fue más alto en comparación con el modelo sin regularización. La regularización L1 tiende a reducir algunos coeficientes a cero, lo que simplifica el modelo.

Análisis del Tiempo de Ejecución

- **Sin regularización**: La función fit sin regularización es muy rápida, incluso con el aumento en el grado del polinomio, lo que lo hace adecuado para cálculos simples.
- **L1 (Lasso)**: Esta regularización es significativamente más costosa en términos de tiempo de ejecución, especialmente para grados más altos. Esto se debe a las iteraciones adicionales necesarias para la minimización de la función de costo.
- **L2 (Ridge)**: El tiempo de ejecución de L2 es similar a la versión sin regularización, ya que no requiere tanto tiempo computacional como L1.

Estas observaciones refuerzan la idea de que, aunque L1 es más costosa, puede ofrecer un modelo más simplificado, mientras que L2 mantiene un buen equilibrio entre complejidad computacional y ajuste del modelo.

7. CONCLUSIONES

- **Regresión polinomial como herramienta eficiente**: La regresión polinomial demostró ser adecuada para ajustar series temporales no lineales. Sin embargo, conforme se aumenta el grado del polinomio, se incrementa el riesgo de sobreajuste, como fue evidente en los resultados para el grado 100. Por lo tanto, es crucial equilibrar el grado del polinomio para evitar sobreajustes sin perder capacidad predictiva.
- **Regularización como mitigación del sobreajuste**: Las técnicas de regularización L1 (Lasso) y L2 (Ridge) ayudaron a controlar el sobreajuste en los modelos de mayor grado. L1 promovió la generación de modelos más simples, eliminando algunos coeficientes al reducirlos a cero, mientras que L2 mantuvo los coeficientes más pequeños sin eliminarlos. Ambos enfoques mejoraron la robustez del modelo.
- **Eficiencia computacional**: La regresión sin regularización es la más rápida, pero en situaciones donde el modelo sobreajusta, el costo computacional adicional de L1 y L2 es justificado. Aunque L1 fue significativamente más costosa en términos de tiempo de ejecución, su capacidad para simplificar el modelo puede ser útil en datasets grandes y dispersos. L2, por otro lado, fue más eficiente y mantuvo un ajuste adecuado.
- **Mejora del ajuste con grados altos**: Aumentar el grado del polinomio mejoró el ajuste, como lo mostró la reducción del MSE, especialmente en el grado 100 sin regularización. No obstante, los modelos con grados más altos requieren regularización para ser efectivos sin caer en sobreajustes.
- **Selección de modelo balanceado**: El ajuste del modelo debe buscar un equilibrio entre precisión (minimizar MSE) y complejidad computacional. Modelos con grado 100 y regularización L2 demostraron ser una opción balanceada, evitando sobreajuste y manteniendo tiempos de ejecución razonables.

REFERENCES

[1] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[2] A. Bahrammirzaee, "A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert systems, and hybrid intelligent systems," *Neural Computing and Applications*, vol. 19, no. 8, pp. 1165–1195, 2010.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[6] D. Maulud and A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 140–147, 2020.

[7] T. M. Hope, "Chapter 4 - linear regression," in *Machine Learning*, A. Mechelli and S. Vieira, Eds. Academic Press, 2020, pp. 67–81. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128157398000043

[8] D. Liang, D. A. Frederick, E. E. Lledo, N. Rosenfield, V. Berardi, E. Linstead, and U. Maoz, "Examining the utility of nonlinear machine learning approaches versus linear regression for predicting body image

outcomes: The u.s. body project i," *Body Image*, vol. 41, pp. 32–45, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1740144522000122

[9] P. Vinuesa, "Tema 9 - regresión lineal simple y polinomial: teoría y práctica," 2016. [Online]. Available: https://www.ccg.unam.mx/~vinuesa/R4biosciences/docs/Tema9_regresion.html#calculo-del-grado-de-ajuste-r2

[10] D. J. Bartholomew, "Time series analysis forecasting and control." 1971.

[11] R. Mushtaq, "Augmented dickey fuller test," 2011.

[12] D. I. MacKenzie, J. D. Nichols, J. A. Royle, K. H. Pollock, L. L. Bailey, and J. E. Hines, "Chapter 3 - fundamental principals of statistical inference," in *Occupancy Estimation and Modeling (Second Edition)*, second edition ed., D. I. MacKenzie, J. D. Nichols, J. A. Royle, K. H. Pollock, L. L. Bailey, and J. E. Hines, Eds. Boston: Academic Press, 2018, pp. 71–111. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124071971000041