Funciones Lineales y No Lineales

1. Introducción

Las funciones son relaciones matemáticas que asignan un valor de salida a un valor de entrada. En matemáticas, las funciones pueden clasificarse en varias categorías, pero aquí nos centraremos en dos tipos fundamentales: las funciones lineales y las funciones no lineales. La comprensión de estas funciones es esencial en diversas disciplinas, como la física, la economía, la ingeniería, y en la modelización de fenómenos naturales y sociales.

2. Funciones en una Dimensión

2.1 Función Lineal

Una función lineal es una función que puede representarse en la forma:

$$f(x)=mx+b$$

donde:

- $f(x)$ es el valor de salida de la función.
- x es la variable independiente (entrada).
- m es la pendiente de la línea, que indica la inclinación de la función.
- b es la intersección con el eje y, es decir, el valor de $f(x)$ cuando $x=0$.

Ejemplo de Función Lineal

Consideremos la función:

$$f(x)=2x+3$$

En este caso:

- La pendiente $m=2$, lo que significa que por cada unidad que aumentamos x, $f(x)$ aumenta en 2 unidades.
- La intersección con el eje y $b=3$, indicando que la línea cruza el eje y en el punto (0, 3).
- El dominio y rango de esta función son ambos todos los números reales, es decir, $(-\infty,\infty)$.

2.2 Características de la Función Lineal

- **Gráfica**: La gráfica de una función lineal es una línea recta en el plano cartesiano, que se puede dibujar utilizando dos puntos.
- **Pendiente**: La pendiente m determina la inclinación de la línea.
  - Si $m>0$, la línea sube de izquierda a derecha.
  - Si $m<0$, la línea baja de izquierda a derecha.
  - Si $m=0$, la línea es horizontal.

- **Intersección**: El punto donde la línea cruza el eje y está dado por b.
- **Propiedades Adicionales**:
  - **Adición**: Si tenemos dos funciones lineales $f(x)=m_1 x+b_1$ y $g(x)=m_2 x+b_2$, entonces la suma de estas funciones es también una función lineal:

$$(f+g)(x)=(m_1+m_2)x+(b_1+b_2)$$

- **Dominio y Rango**: El dominio y rango de una función lineal son todos los números reales, es decir, $(-\infty,\infty)$.

2.3 Función No Lineal

Una función no lineal no puede representarse como una línea recta y puede adoptar diversas formas, como cuadráticas, cúbicas, exponenciales, logarítmicas, entre otras. La forma general de una función cuadrática es:

$$f(x)=a x^2+b x+c$$

donde a, b, y c son constantes. Esta forma representa una parábola que puede abrirse hacia arriba o hacia abajo, dependiendo del signo de a.

Ejemplo de Función No Lineal

Consideremos la función:

$$f(x)=x^2-4x+4$$

Esta función es cuadrática, y podemos factorizarla como:

$$f(x)=(x-2)^2$$

La gráfica de esta función es una parábola que tiene su vértice en el punto (2, 0) y abre hacia arriba.

- El dominio de esta función sigue siendo $(-\infty,\infty)$, pero el rango es ¿, ya que no puede tomar valores negativos.

2.4 Características de la Función No Lineal

- **Gráfica**: La gráfica de una función no lineal puede ser una parábola, una hipérbola, una curva, o cualquier forma no lineal.
- **Variabilidad**: En una función no lineal, la tasa de cambio no es constante, lo que significa que la pendiente puede variar en diferentes puntos.
- **Intersecciones**: Puede haber múltiples intersecciones con los ejes, dependiendo de la forma de la función.
- **Dominio y Rango**: El dominio y rango pueden ser más limitados que en funciones lineales. Por ejemplo, en una función cuadrática con un vértice mínimo, el rango será ¿, donde k es el valor mínimo.

3. Funciones en Múltiples Dimensiones

3.1 Función Lineal en Múltiples Dimensiones

En el caso de múltiples dimensiones (por ejemplo, en R^2 o R^3), una función lineal se puede representar como:

$$f(x) = A\,x + b$$

donde:

- $f(x)$ es el vector de salida.
- x es el vector de entrada, por ejemplo, en R^2: $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.
- A es una matriz que representa la transformación lineal, en este caso, una matriz 2×2.
- b es un vector de traslación.

Ejemplo de Función Lineal en 2D

Tomemos la función:

$$f(x) = \begin{pmatrix} 2 & 3 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Al calcular esta función, obtenemos:

$$f(x) = \begin{pmatrix} 2\,x_1 + 3\,x_2 + 1 \\ x_1 + 4\,x_2 - 1 \end{pmatrix}$$

La representación gráfica de esta función en 2D es una línea recta.

3.2 Características de la Función Lineal en Múltiples Dimensiones

- **Gráfica**: La representación gráfica de una función lineal en 2D es una línea recta, y en 3D es un plano.
- **Propiedades**: Conserva la propiedad de superposición; si $f(x_1)$ y $f(x_2)$ son soluciones, entonces:

$$f(c\,x_1 + d\,x_2) = c\,f(x_1) + d\,f(x_2)$$

para cualquier $c, d \in R$.

- **Espacios Vectoriales**: Las funciones lineales se asocian con espacios vectoriales, lo que significa que la combinación lineal de funciones lineales también es lineal.
- **Determinante**: Para funciones lineales representadas por matrices, el determinante de la matriz A puede indicar si la transformación es invertible. Si $det(A) \neq 0$, la transformación es invertible.

3.3 Función No Lineal en Múltiples Dimensiones

Una función no lineal en múltiples dimensiones puede tener la forma:

$$f(x)=g(x)$$

donde g no es una transformación lineal. Por ejemplo, consideremos una función cuadrática que se puede expresar en forma matricial:

$$f(x)=x^T A x+b^T x+c$$

donde:

- x es un vector de entrada en R^2, por ejemplo, $x=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.
- A es una matriz simétrica, 2×2, que determina la forma de la superficie.
- b es un vector de coeficientes lineales, 2×1.
- c es una constante.

Ejemplo de Función No Lineal en 2D

Supongamos que tenemos:

$$A=\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, b=\begin{pmatrix} -3 \\ -4 \end{pmatrix}, c=5$$

Entonces, la función es:

$$f(x)=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}+\begin{pmatrix} -3 \\ -4 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}+5$$

Al expandir esto, obtenemos:

$$f(x_1, x_2)=2 x_1^2+2 x_2^2+2 x_1 x_2-3 x_1-4 x_2+5$$

Esta función representa una superficie no lineal en el espacio tridimensional.

- El análisis de esta función puede involucrar la identificación de puntos críticos (máximos, mínimos o puntos de silla) utilizando derivadas parciales.

3.4 Características de la Función No Lineal en Múltiples Dimensiones

- **Gráfica**: La gráfica puede ser superficies curvas o formas más complejas, dependiendo de la función. En 3D, la representación puede ser más complicada, mostrando crestas y valles.
- **Tasa de Cambio**: La tasa de cambio varía en diferentes puntos, lo que complica el análisis. Esto puede ser medido utilizando derivadas parciales en el contexto multivariable. Por ejemplo, la derivada parcial de f respecto a x_1 se denota como:

$$\frac{\partial f}{\partial x_1}=4 x_1+2 x_2-3$$

y respecto a x_2 como:

$$\frac{\partial f}{\partial x_2} = 4x_2 + 2x_1 - 4.$$

- **Comportamiento**: Puede tener múltiples máximos y mínimos locales, lo que es crucial para el estudio de optimización. Por ejemplo, en la función $f(x, y) = x^2 - y^2$, hay un mínimo en $(0, 0)$ y el comportamiento de la superficie cambia alrededor de ese punto.
- **Aplicaciones**: Las funciones no lineales se utilizan ampliamente en modelos de crecimiento poblacional, economía, biología y muchas áreas de ingeniería, donde el comportamiento no es lineal. En sistemas de control, por ejemplo, se utilizan funciones no lineales para modelar el comportamiento de sistemas complejos.

4. Comparación entre Funciones Lineales y No Lineales

4.1 Complejidad

- **Funciones Lineales**: Son más simples de analizar y resolver. Sus soluciones pueden expresarse de forma cerrada y son generalmente más fáciles de manipular matemáticamente.
- **Funciones No Lineales**: Requieren métodos más avanzados para su análisis, como el uso de cálculo diferencial e integral, y pueden tener múltiples soluciones.

4.2 Soluciones y Comportamiento

- **Funciones Lineales**: La solución a un sistema de ecuaciones lineales puede encontrarse usando técnicas como la eliminación de Gauss o la regla de Cramer. Además, el comportamiento de la solución es predecible y estable.
- **Funciones No Lineales**: Pueden tener soluciones múltiples o ninguna solución, dependiendo de la naturaleza de la función. Además, el comportamiento puede ser caótico y menos predecible, lo que complica la modelización.

Las funciones lineales son más simples y se caracterizan por su representación gráfica como líneas rectas, mientras que las funciones no lineales son más complejas y pueden representar una variedad de formas y comportamientos en matemáticas. La comprensión de estas diferencias es crucial en muchos campos, como la economía, la física y la ingeniería. Además, el análisis de funciones no lineales es fundamental en problemas de optimización y en el estudio de sistemas dinámicos, lo que resalta la importancia de estas funciones en la modelización de situaciones del mundo real.

Sistemas Lineales y No Lineales

1. Introducción

Los sistemas de ecuaciones son conjuntos de ecuaciones que comparten variables comunes. Estos sistemas pueden clasificarse en lineales y no lineales, y su estudio es fundamental en diversas áreas de las matemáticas, la física, la ingeniería y la economía. La capacidad de resolver estos sistemas es crucial para modelar y comprender fenómenos del mundo real.

2. Sistema Lineal

2.1 Definición

Un sistema lineal se puede representar en la forma matricial:

$$A\,x = b$$

donde:

- A es una matriz que contiene los coeficientes del sistema, donde cada fila representa una ecuación y cada columna representa una variable.
- x es el vector de variables que queremos encontrar, es decir, los valores desconocidos que satisfacen todas las ecuaciones.
- b es un vector de constantes que representa los términos independientes de cada ecuación.

2.2 Ejemplos de Sistema Lineal

Ejemplo 1

Consideremos el siguiente sistema de ecuaciones lineales:

$$\begin{array}{ll} 2\,x_1 + 3\,x_2 & ¿\,5 \\ 4\,x_1 - x_2 & ¿\,1 \end{array}$$

Este sistema se puede escribir en forma matricial como:

$$A = \begin{pmatrix} 2 & 3 \\ 4 & -1 \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, b = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

En este caso, la matriz A contiene los coeficientes de las variables x_1 y x_2 en las ecuaciones, lo que permite representar el sistema de manera compacta.

Ejemplo 2

Consideremos otro sistema de ecuaciones lineales:

$$\begin{array}{ll} x_1 - 2\,x_2 & ¿\,3 \\ 3\,x_1 + 4\,x_2 & ¿\,10 \end{array}$$

Este sistema se puede escribir en forma matricial como:

$$A = \begin{pmatrix} 1 & -2 \\ 3 & 4 \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, b = \begin{pmatrix} 3 \\ 10 \end{pmatrix}$$

Aquí, también se observa que las ecuaciones son lineales en función de las variables x_1 y x_2.

2.3 Solución del Sistema Lineal

La solución de un sistema lineal se puede encontrar utilizando varios métodos, como la eliminación de Gauss, la regla de Cramer, o la inversión de matrices (si es aplicable). Un sistema lineal puede tener:

- **Una solución única**: Cuando las ecuaciones son independientes y se intersectan en un solo punto.
- **Sin solución**: Si las ecuaciones son inconsistentes y no se intersectan en ningún punto (por ejemplo, dos líneas paralelas).
- **Infinitas soluciones**: Si las ecuaciones son dependientes (una es múltiplo de la otra) y se representan en la misma línea.

2.4 Sistemas Lineales con Funciones No Lineales

Es importante mencionar que una matriz A puede contener funciones no lineales en sus entradas. Por ejemplo, si definimos una matriz A como:

$$A=\begin{pmatrix} f_1(x) & f_2(y) \\ g(z) & h(w) \end{pmatrix}$$

donde f_1, f_2, g, y h son funciones no lineales de las variables x, y, z, y w, entonces, el sistema aún puede representarse como $A\,x=b$. Sin embargo, la resolución del sistema puede volverse complicada debido a la no linealidad de las funciones.

2.5 Ejemplos de Sistemas Lineales con Funciones No Lineales

Ejemplo 1

Consideremos el siguiente sistema donde la matriz A tiene funciones no lineales:

$$\begin{aligned} x_1\sin(x_1)+2\,x_2 &\quad ¿5 \\ 4\,x_1-x_2e^{x_2} &\quad ¿1 \end{aligned}$$

Este sistema se puede escribir en forma matricial como:

$$A=\begin{pmatrix} \sin(x_1) & 2 \\ 4 & -e^{x_2} \end{pmatrix}, x=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, b=\begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

En este caso, la matriz A incluye una función trigonométrica y una función exponencial, lo que hace que el sistema sea más complicado de resolver.

Ejemplo 2

Consideremos otro sistema donde la matriz A contiene funciones no lineales:

$$\begin{aligned} x_1\cos(x_2)+3\,x_2 &\quad ¿4 \\ 2\,x_1+x_2\ln(x_1) &\quad ¿2 \end{aligned}$$

Este sistema se puede escribir en forma matricial como:

$$A=\begin{pmatrix}\cos(x_2) & 3\\ 2 & \ln(x_1)\end{pmatrix}, x=\begin{pmatrix}x_1\\ x_2\end{pmatrix}, b=\begin{pmatrix}4\\ 2\end{pmatrix}$$

Aquí, la matriz A incluye funciones trigonométricas y logarítmicas, lo que añade un nivel de complejidad en la resolución del sistema.

3. Sistema No Lineal

3.1 Definición

Un **sistema no lineal** es aquel que no puede llevarse a la forma $A\,x=b$. Un caso particular de sistemas no lineales se puede representar como:

$$x^T A\,x+B\,x=c$$

donde:

- x es un vector de variables.
- A es una matriz que contiene los coeficientes de las variables cuadráticas.
- B es una matriz que contiene los coeficientes de las variables lineales.
- c es una constante que representa el resultado esperado.

3.2 Ejemplos de Sistema No Lineal

Ejemplo 1

Consideremos el siguiente sistema no lineal:

$$x_1^2+2x_1x_2+x_2^2+3x_1-x_2=7$$

Este sistema se puede escribir en forma matricial como:

$$x^T A\,x+B\,x=c$$

donde:

$$A=\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}, B=\begin{pmatrix}3 & -1\end{pmatrix}, c=7$$

En este caso, el término cuadrático $x^T A\,x$ representa las interacciones entre las variables, mientras que Bx representa los términos lineales.

Ejemplo 2

Consideremos otro sistema no lineal:

$$x_1^2-4x_2^2+3x_1+2=0$$

Este sistema puede ser escrito como:

$$x^T A\, x + B\, x = c$$

donde:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -4 \end{pmatrix}, B = \begin{pmatrix} 3 & 0 \end{pmatrix}, c = -2$$

3.3 Solución del Sistema No Lineal

Resolver un sistema no lineal puede ser más complicado que resolver un sistema lineal. Generalmente se requieren métodos numéricos como el método de Newton-Raphson, el método de la bisección, o algoritmos de optimización. Dependiendo de la naturaleza del sistema, puede haber múltiples soluciones o ninguna solución, lo que dificulta el análisis. La convergencia de estos métodos también depende de la elección de un valor inicial adecuado.

3.4 Sistemas No Lineales con Funciones No Lineales

En un sistema no lineal, las matrices A y B pueden contener funciones no lineales. Por ejemplo, si definimos un sistema no lineal como:

$$x^T A(x) x + B(x)\, x = c$$

donde:

- $A(x)$ es una matriz cuyas entradas son funciones no lineales de las variables x.
- $B(x)$ es un vector donde cada componente es una función no lineal de las variables x.
- c sigue siendo una constante.

Esto complica aún más la resolución del sistema, ya que ahora tanto los términos cuadráticos como lineales son no lineales.

3.5 Ejemplos de Sistemas No Lineales con Funciones No Lineales

Ejemplo 1

Consideremos el siguiente sistema:

$$f(x_1, x_2) = x_1^4 + x_1^2 x_2 + x_1 x_2^2 + x_2^2 \sin(x_2) + x_1 + x_2 e^{x_2} - 5 = 0$$

Podemos representar este sistema como:

$$x^T A(x) x + B(x)\, x = c$$

donde:

$$A(x) = \begin{pmatrix} x_1^2 & x_1 \\ x_2 & \sin(x_2) \end{pmatrix}, B(x) = \begin{pmatrix} 1 & e^{x_2} \end{pmatrix}, c = 5$$

En este caso, el sistema incluye tanto términos cuadráticos como funciones trigonométricas y exponenciales, lo que aumenta la complejidad del sistema.

Ejemplo 2

Consideremos otro sistema no lineal:

$$g(x_1, x_2) = x_1^2 \ln(x_2) + 2x_1 x_2^2 + x_1 - x_2 \tan(x_2) = 0$$

Este sistema se puede escribir como:

$$x^T A(x) x + B(x) x = c$$

donde:

$$A(x) = \begin{pmatrix} \ln(x_2) & 0 \\ 0 & 2x_1 \end{pmatrix}, B(x) = \begin{pmatrix} 1 & -\tan(x_2) \end{pmatrix}, c = 0$$

Aquí, el sistema incluye un término logarítmico y cuadráticos, lo que complica aún más la resolución del sistema.

4. Comparación entre Sistemas Lineales y No Lineales

4.1 Complejidad

- **Sistemas Lineales**: Generalmente más simples de resolver. Su análisis puede llevarse a cabo mediante álgebra lineal y métodos bien establecidos, como la eliminación de Gauss o la regla de Cramer.
- **Sistemas No Lineales**: Requieren métodos más complejos, como el método de Newton-Raphson, la búsqueda de mínimos, o métodos de optimización global. La solución puede no ser única y a menudo involucra técnicas de cálculo numérico.

4.2 Soluciones y Comportamiento

- **Sistemas Lineales**: La solución es más estable y predecible. Se pueden utilizar gráficos para visualizar las soluciones y sus intersecciones. Además, el comportamiento de la solución depende linealmente de los parámetros.
- **Sistemas No Lineales**: Pueden tener múltiples máximos y mínimos locales, lo que complica la identificación de soluciones globales. El comportamiento de la solución puede ser altamente sensible a las condiciones iniciales.

Tanto los sistemas lineales como los no lineales son fundamentales en la matemática aplicada y en la modelización de fenómenos en el mundo real. Mientras que los sistemas lineales son más directos y fáciles de manejar, los sistemas no lineales ofrecen desafíos adicionales que requieren enfoques más sofisticados para su análisis y solución. Comprender estas diferencias es crucial para aplicar las técnicas adecuadas en situaciones prácticas, ya que el tipo de sistema determinará el enfoque y las herramientas matemáticas que se utilizarán.

La Importancia de Distinguir un Sistema Lineal y No Lineal en Ciencia de Datos

1. Introducción

En la ciencia de datos, la capacidad para distinguir entre un sistema lineal y uno no lineal es crucial para abordar correctamente los problemas que surgen en diversos campos como la economía, la biomedicina, la ingeniería y las ciencias sociales. Esta distinción afecta directamente el tipo de técnicas matemáticas y computacionales que se deben utilizar para construir modelos predictivos, analizar tendencias y realizar inferencias sobre los datos. En esencia, el tipo de sistema con el que trabajamos determina qué tan sencillo o complejo será encontrar una solución óptima, y qué tan interpretables serán los resultados.

2. Relevancia de los Sistemas Lineales

Los sistemas lineales son ampliamente utilizados en ciencia de datos debido a su simplicidad y solvencia computacional. Esto se debe a que en estos sistemas las relaciones entre las variables son proporcionales, lo que significa que un cambio en una variable tiene un efecto directo y constante en la otra. Estos sistemas pueden ser resueltos de manera eficiente con herramientas estándar como la **eliminación gaussiana**, la **descomposición LU** y otros métodos algebraicos.

En el contexto de la ciencia de datos, los sistemas lineales permiten la aplicación de técnicas de regresión lineal, regresión múltiple, y modelos como el de mínimos cuadrados. Estas herramientas son esenciales para el análisis exploratorio de datos y para la construcción de modelos predictivos que permiten a las empresas y científicos de datos extraer información útil sobre las relaciones lineales entre las variables. Por ejemplo, en marketing, la **regresión lineal múltiple** se utiliza para modelar cómo las variables, como el precio y la publicidad, influyen en las ventas.

2.1 Beneficios de los Sistemas Lineales

- **Eficiencia Computacional**: Los sistemas lineales, al ser más sencillos, pueden resolverse de manera rápida incluso cuando se trata de grandes cantidades de datos. Esto es especialmente importante cuando se manejan datos a gran escala, como en big data, donde la eficiencia en el cálculo es una prioridad.
- **Interpretabilidad**: Uno de los principales atractivos de los sistemas lineales es que sus soluciones son interpretables. Los coeficientes de los modelos lineales tienen una interpretación clara y directa, lo que permite comprender de forma sencilla cómo cada variable independiente está afectando al resultado. Esto es crucial en áreas como la medicina, las finanzas o las ciencias sociales, donde la transparencia en la interpretación de los modelos es importante para la toma de decisiones.
- **Soluciones Únicas**: En los sistemas lineales, si el sistema es determinado, las soluciones son únicas. Esto es particularmente útil en situaciones donde se necesita una única respuesta clara y definitiva.

2.2 Limitaciones de los Sistemas Lineales

Aunque los sistemas lineales son poderosos en su simplicidad, tienen limitaciones importantes. La principal es que no pueden capturar relaciones más complejas entre las variables. Muchos fenómenos del mundo real, como el crecimiento poblacional, los mercados financieros o el comportamiento humano, no siguen patrones lineales simples.

- **No Capturan No Linealidades**: Los sistemas lineales son insuficientes cuando las relaciones entre las variables son curvas, exponenciales o más complicadas. En estos casos, un modelo lineal podría subestimar o sobreestimar los resultados, ofreciendo una representación pobre de los datos.
- **Soluciones Imprecisas en Fenómenos Complejos**: En fenómenos complejos, como los procesos biológicos o las interacciones en redes neuronales, un enfoque lineal puede ser demasiado simplista y generar resultados imprecisos.

3. Los Desafíos de los Sistemas No Lineales

En la ciencia de datos, los sistemas no lineales son comunes cuando las relaciones entre las variables no pueden describirse mediante una combinación lineal. Un sistema no lineal puede tener múltiples soluciones, ninguna solución, o soluciones que dependen de las condiciones iniciales, lo que lo hace más difícil de resolver. Estos sistemas requieren enfoques más avanzados y, generalmente, métodos numéricos iterativos como el **método de Newton-Raphson** o el **descenso por gradiente**.

La capacidad para reconocer cuándo un sistema es no lineal es esencial porque permite seleccionar modelos más avanzados que puedan capturar mejor la complejidad de los datos. Por ejemplo, en muchos problemas de predicción en ciencia de datos, un modelo no lineal como una **red neuronal** o una **máquina de soporte vectorial** puede ofrecer mejores resultados que un modelo lineal.

3.1 Ventajas de los Sistemas No Lineales

- **Modelan Relaciones Complejas**: Los sistemas no lineales pueden capturar fenómenos complejos como interacciones entre variables, efectos de saturación, o crecimiento exponencial, lo que los hace más adecuados para modelar procesos naturales, biológicos o sociales.
- **Predicciones Más Precisas**: Cuando los datos muestran claras no linealidades, como en los casos de fenómenos económicos o climatológicos, un sistema no lineal puede ajustarse mejor a los datos, proporcionando predicciones más precisas.
- **Aplicaciones en Machine Learning**: Muchas de las técnicas avanzadas de aprendizaje automático, como las **redes neuronales** o los **modelos de árboles de decisión**, son inherentemente no lineales. Estas técnicas permiten descubrir patrones no lineales en los datos que serían imposibles de detectar con un enfoque lineal.

3.2 Desventajas de los Sistemas No Lineales

- **Complejidad Computacional**: Los sistemas no lineales requieren mayor potencia computacional para ser resueltos. A menudo implican resolver ecuaciones de forma iterativa, lo cual puede llevar tiempo y esfuerzo computacional significativo.

- **Falta de Interpretabilidad**: Los modelos no lineales, como las redes neuronales profundas, a menudo son "cajas negras", lo que significa que son difíciles de interpretar. Esto es un problema en áreas como la salud o las finanzas, donde los usuarios necesitan comprender cómo un modelo está tomando sus decisiones.
- **Sensibilidad a las Condiciones Iniciales**: Muchos sistemas no lineales son sensibles a las condiciones iniciales, lo que puede llevar a múltiples soluciones o soluciones que dependen de pequeños cambios en los datos iniciales.

4. Impacto en la Eficiencia Computacional

La elección entre un sistema lineal y no lineal afecta significativamente la **eficiencia computacional**. Mientras que los sistemas lineales pueden resolverse con algoritmos eficientes de álgebra lineal, los sistemas no lineales suelen necesitar métodos más sofisticados que pueden ser computacionalmente costosos.

4.1 Sistemas Lineales y Escalabilidad

En proyectos de ciencia de datos que implican grandes cantidades de datos, como análisis de redes sociales, comercio electrónico o procesamiento de imágenes, la escalabilidad es una preocupación clave. Los sistemas lineales permiten resolver problemas rápidamente incluso cuando se trata de matrices de gran tamaño o cuando se requiere el procesamiento en tiempo real. Los **algoritmos basados en matrices dispersas** y la **descomposición en valores singulares (SVD)** son ejemplos de técnicas que permiten manejar grandes datasets de forma eficiente.

4.2 Sistemas No Lineales y Complejidad

Los sistemas no lineales, por otro lado, no escalan tan fácilmente. A menudo, es necesario utilizar métodos iterativos como el **método de Newton-Raphson** o técnicas de optimización no lineal, lo que puede requerir muchos recursos computacionales y tiempo de procesamiento. Este es un desafío importante cuando se trabaja con grandes cantidades de datos en aplicaciones de inteligencia artificial o simulaciones físicas complejas.

5. Aplicaciones Prácticas

5.1 FinTech y Predicción Financiera

En el sector financiero, la capacidad de distinguir entre un sistema lineal y no lineal tiene implicaciones significativas para la predicción de precios, el análisis de riesgo y la gestión de portafolios. Mientras que los modelos lineales pueden ser útiles para analizar relaciones simples entre factores macroeconómicos y precios de activos, los modelos no lineales, como las **redes neuronales recurrentes (RNNs)**, se han demostrado más eficaces en capturar patrones temporales complejos y no lineales en los precios de mercado.

5.2 Biología Computacional y Medicina

En biología y medicina, muchos sistemas no lineales capturan interacciones complejas entre genes, proteínas o variables fisiológicas. Por ejemplo, el crecimiento de tumores o la propagación de epidemias a menudo sigue patrones no lineales, lo que requiere el uso de **modelos de crecimiento exponencial** o **modelos logísticos** para realizar predicciones precisas.

5.3 Marketing y Ciencia del Comportamiento

En el marketing, los sistemas no lineales permiten modelar fenómenos como la saturación publicitaria o el comportamiento de compra de los consumidores. A medida que las campañas publicitarias aumentan, el impacto marginal de cada nueva inversión no es constante, lo que requiere un modelo no lineal para capturar la dinámica real del retorno de inversión.

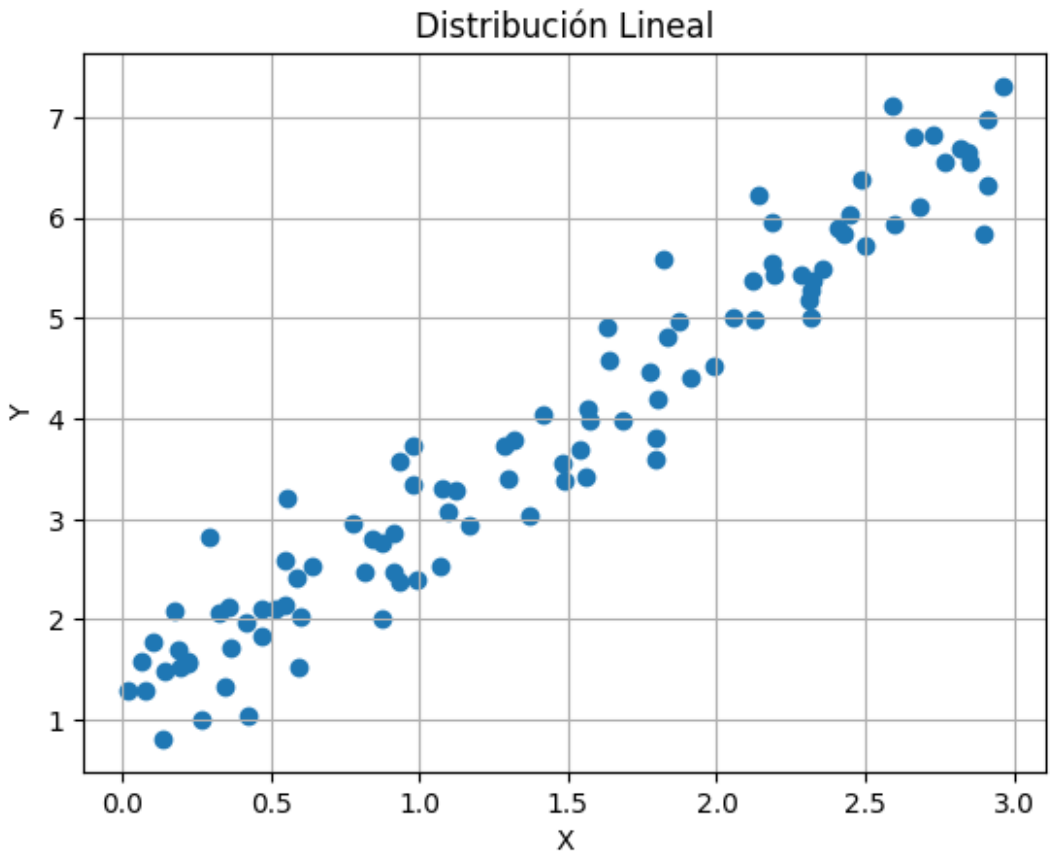
6. Ejemplos

A continuación, se presentan ejemplos en Python que ilustran distribuciones de puntos con comportamiento lineal y no lineal, junto con sus gráficas de dispersión.

6.1 Ejemplo de Comportamiento Lineal

En este ejemplo, generamos un conjunto de datos que sigue una relación lineal y mostramos la gráfica de dispersión.

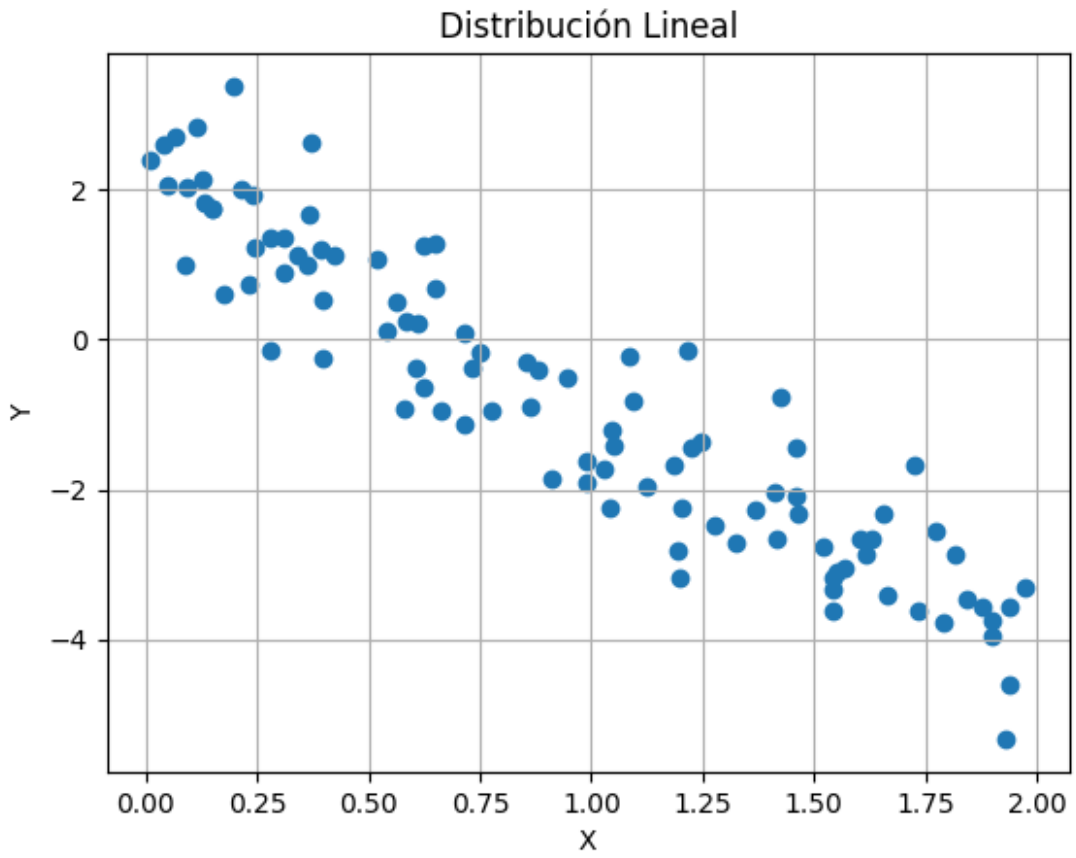
```
import numpy as np
import matplotlib.pyplot as plt

# Generar datos lineales
np.random.seed(42)
x = 3*np.random.rand(100)
y = 2 * x + 1 + np.random.normal(0, 0.5, 100)  # y = 2x + 1 + ruido

# Gráfica de dispersión
plt.scatter(x, y)
plt.title('Distribución Lineal')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```



6.2 Ejemplo de Comportamiento Lineal

Generamos otro conjunto de datos que sigue una relación lineal diferente y mostramos la gráfica de dispersión.

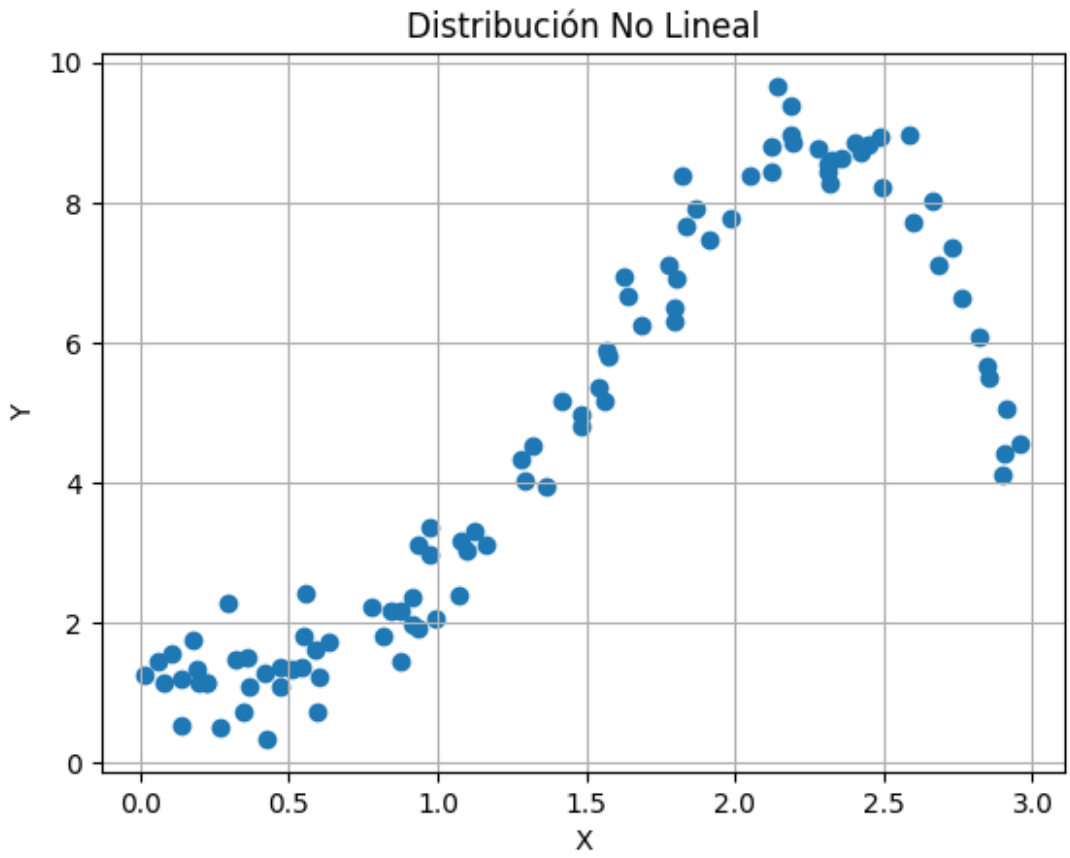
```
# Generar otro conjunto de datos lineales
np.random.seed(42)
x = 2*np.random.rand(100)
y = -3 * x + 2 + np.random.normal(0, 0.8, 100)  # y = -3x + 2 + ruido

# Gráfica de dispersión
plt.scatter(x, y)
plt.title('Distribución Lineal')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```



6.3 Ejemplo de Comportamiento No Lineal

En este ejemplo, generamos un conjunto de datos que sigue una relación con una contribución cuadrática y mostramos la gráfica de dispersión.

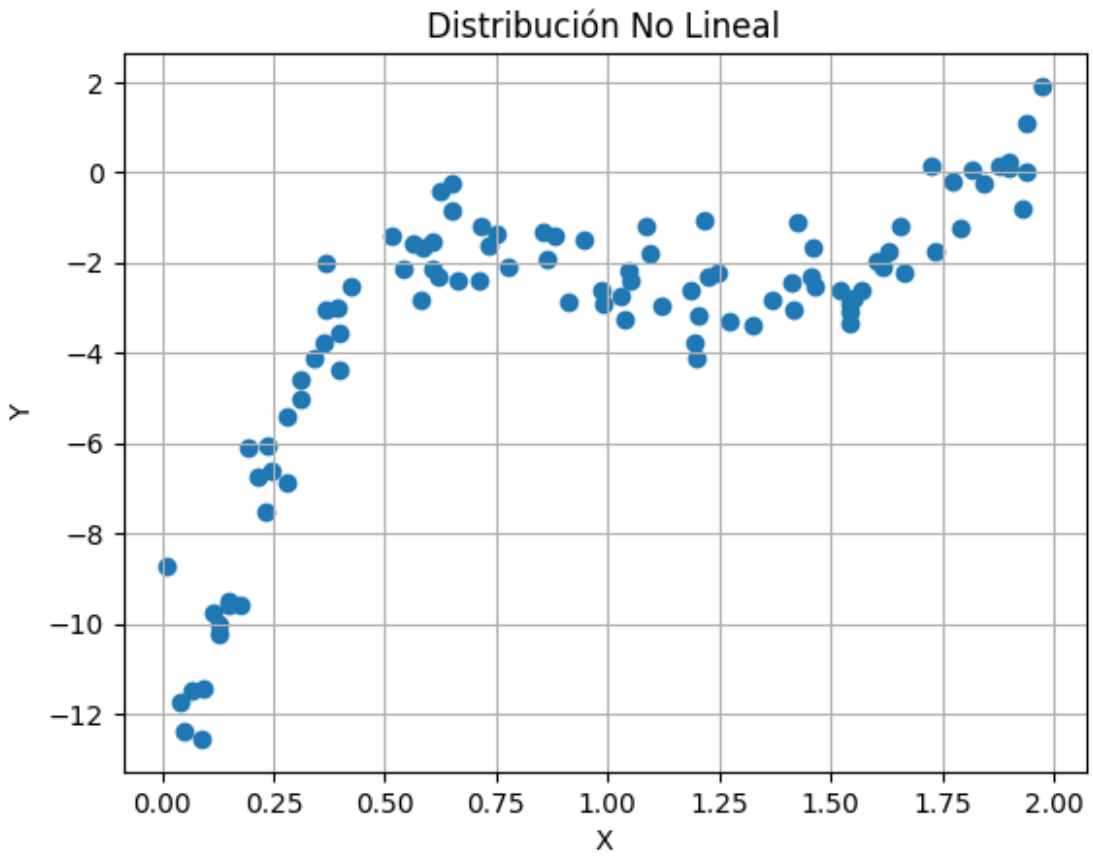
```
# Generar datos no lineales
np.random.seed(42)
x = 3*np.random.rand(100)
y = 2*np.sin(x)*(x**2) + 1 + np.random.normal(0, 0.5, 100)  # y =
3sin(x)x^2 + 1 + ruido

# Gráfica de dispersión
plt.scatter(x, y)
plt.title('Distribución No Lineal')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```



6.4 Ejemplo de Comportamiento No Lineal

Generamos otro conjunto de datos que sigue una relación con una contribución cúbica y mostramos la gráfica de dispersión.

```
# Generar datos no lineales
np.random.seed(42)
x = 2*np.random.rand(100)
y = 10*x*(np.log(x)**3) - 3 * x + 1 + np.random.normal(0, 0.8, 100)  #
y = 10x(log(x))^3 - 3x + 1 + ruido

# Gráfica de dispersión
plt.scatter(x, y)
plt.title('Distribución No Lineal')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```



La distinción entre sistemas lineales y no lineales es fundamental en ciencia de datos porque afecta directamente la precisión, interpretabilidad y eficiencia del análisis. Los sistemas lineales son rápidos, eficientes y fáciles de interpretar, mientras que los sistemas no lineales, aunque más complejos, capturan relaciones más ricas y permiten obtener predicciones más precisas en muchos contextos. En definitiva, un científico de datos debe ser capaz de reconocer y seleccionar el modelo adecuado para maximizar el valor de los datos en cada contexto específico.

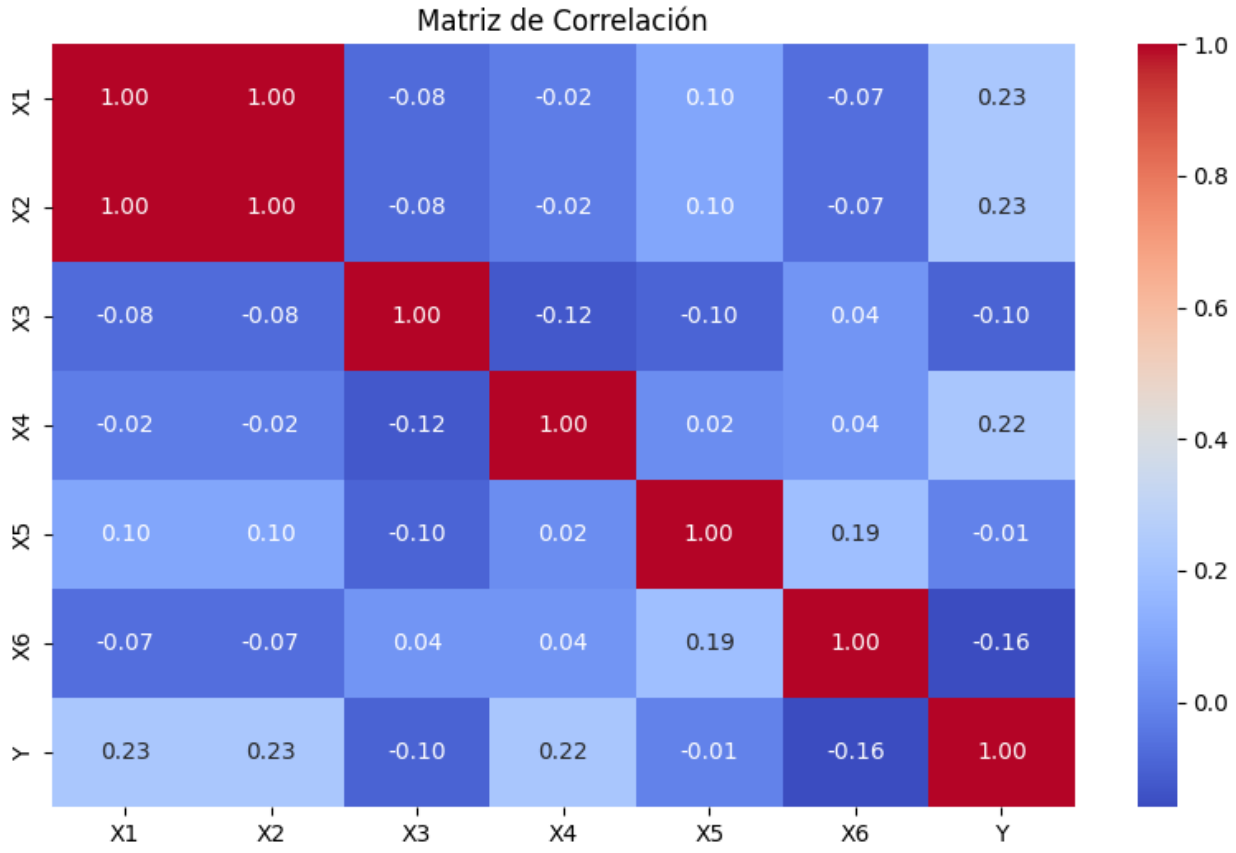
Ejercicio en Python: Uso de una Matriz de Correlación para la Selección de Variables en un Sistema Lineal

En el contexto de la ciencia de datos, la **selección de variables** es una tarea esencial para optimizar el rendimiento de los modelos predictivos y evitar redundancias. Cuando tratamos con sistemas lineales, es crucial identificar variables que están altamente correlacionadas entre sí, ya que esto podría llevar a **multicolinealidad**, una situación en la que dos o más variables independientes están fuertemente correlacionadas, lo que puede distorsionar los resultados de los modelos lineales.

En este ejercicio, utilizaremos una **matriz de correlación** para detectar y eliminar variables redundantes en un conjunto de datos. Esto nos ayudará a simplificar el modelo, reduciendo el riesgo de sobreajuste y mejorando la interpretabilidad.

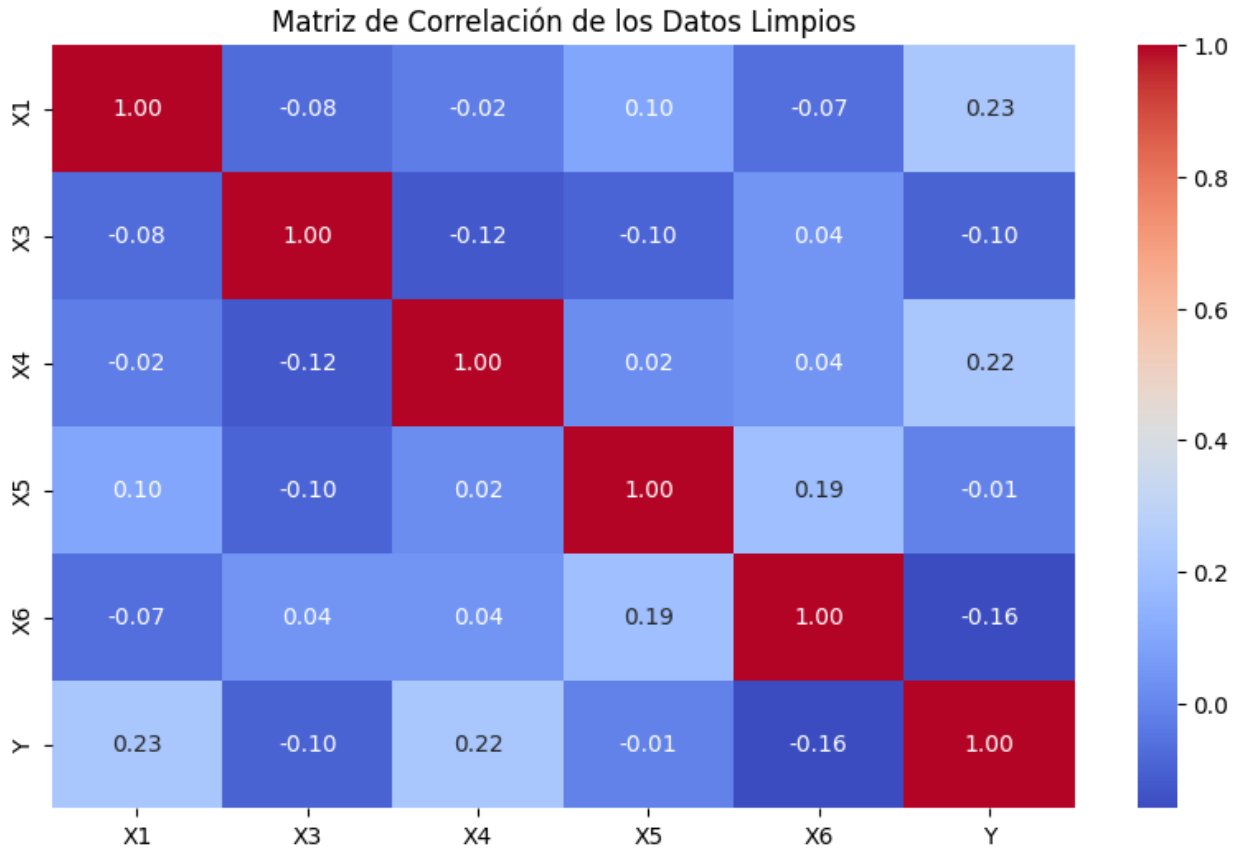
```
# 1. Importar Librerías Necesarias

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 2. Crear un DataFrame de Ejemplo con Variables Correlacionadas

# Generamos un conjunto de datos con **variables altamente
correlacionadas**
# para demostrar cómo una matriz de correlación puede ayudar a
identificar esas relaciones.

np.random.seed(42)
X1 = np.random.rand(100)
X2 = (X1 * 0.4) + 0.2
data = pd.DataFrame({
    'X1': X1,
    'X2': X2,  # Alta correlación con X1
    'X3': (X1 + X2)*np.random.normal(0, 1, 100),  # Relación con X1 y
X2
    'X4': np.random.rand(100) + np.random.normal(0, 1, 100),  #
Variable independiente
    'X5': np.random.rand(100) + np.random.normal(0, 1, 100),  #
Variable independiente
    'X6': np.random.rand(100) + np.random.normal(0, 1, 100),  #
Variable independiente
    'Y': np.random.normal(0, 1, 100)    # Variable objetivo
})

# 3. Calcular la Matriz de Correlación

# La **matriz de correlación** nos muestra el grado de relación lineal
entre las variables,
# donde 1 indica una correlación perfecta y 0 significa que no hay
correlación.

corr_matrix = data.corr()


# 4. Visualizar la Matriz de Correlación

# Usamos `seaborn` para generar una visualización gráfica de la matriz
de correlación.
# Las zonas más oscuras indican una mayor correlación.
```

```
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación')
plt.show()

# 5. Identificar Columnas con Correlación Alta

# Definimos un **umbral de correlación** de 0.8 para identificar
variables que están
# fuertemente correlacionadas y, por lo tanto, pueden ser redundantes.


threshold = 0.8
to_drop = []
for column in corr_matrix.columns:
    if any((corr_matrix[column].abs() > threshold) &
(corr_matrix.index != column)):
        to_drop.append(column)

print(f"Variables altamente correlacionadas con otras: {to_drop}")
```




```
Variables altamente correlacionadas con otras: ['X1', 'X2']
```

```
# 6. Eliminar Variables Redundantes

# Eliminamos las variables que están altamente correlacionadas, como `X2`,
# ya que no aporta información adicional al modelo que `X1` no proporcione.

data_cleaned = data.drop(columns=['X2'])

# 7. Visualizar la Nueva Matriz de Correlación

# Volvemos a calcular y visualizar la **matriz de correlación** tras eliminar
# las variables redundantes.

corr_matrix = data_cleaned.corr()

plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación de los Datos Limpios')
plt.show()
```



```
# 8. Dividir los Datos en Entrenamiento y Prueba

# Dividimos los datos en conjuntos de **entrenamiento y prueba** para
evaluar
# el modelo de regresión lineal.

X = data_cleaned.drop(columns='Y')
y = data_cleaned['Y']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


# 9. Aplicar un Modelo de Regresión Lineal

# Entrenamos un **modelo de regresión lineal** usando los datos de
entrenamiento.

model = LinearRegression()
model.fit(X_train, y_train)

# Predecir los valores en el conjunto de prueba
y_pred = model.predict(X_test)

# 10. Comparar Valores Predichos vs Reales

# Comparamos los primeros 10 valores predichos con los valores reales
y calculamos
# el **error porcentual**.

comparison_df = pd.DataFrame({
    'Valor Real': y_test.head(10).values,
    'Valor Predicho': y_pred[:10]
})

comparison_df['Error Porcentual'] = ((comparison_df['Valor Real'] -
comparison_df['Valor Predicho']) / comparison_df['Valor Real']) * 100

print("\nComparación de los primeros 10 valores entre el valor
predicho y el valor real:\n", comparison_df)

# 11. Evaluar el Modelo

# Calculamos el **error cuadrático medio (MSE)** y el valor **R²**
para evaluar
# el rendimiento del modelo.


mse = mean_squared_error(y_test, y_pred)
print(f"\nError cuadrático medio (MSE) del modelo: {mse:.4f}")
```

```
r2 = r2_score(y_test, y_pred)
print(f"\nValor R² del modelo: {r2:.4f}")


Comparación de los primeros 10 valores entre el valor predicho y el valor real:
    Valor Real  Valor Predicho  Error Porcentual
0    1.053153       -0.349489        133.184977
1   -0.703176        0.236094        133.575406
2    0.380198        0.620051        -63.086462
3    0.184836        0.032804         82.252143
4    0.493318       -0.381028        177.237774
5   -1.351685       -0.301508         77.693886
6    1.029156        0.083102         91.925233
7    0.271579        0.030975         88.594581
8   -1.348185        0.087065        106.457952
9   -0.544919        0.321894        159.071924

Error cuadrático medio (MSE) del modelo: 0.6358

Valor R² del modelo: -0.0728
```

Este ejercicio demuestra la importancia de usar una **matriz de correlación** en sistemas lineales para detectar variables redundantes, lo cual permite simplificar el modelo, reducir la multicolinealidad y mejorar la eficiencia computacional y la interpretabilidad. Este enfoque es esencial en el contexto de la ciencia de datos para garantizar la calidad del análisis y la fiabilidad de los modelos predictivos.

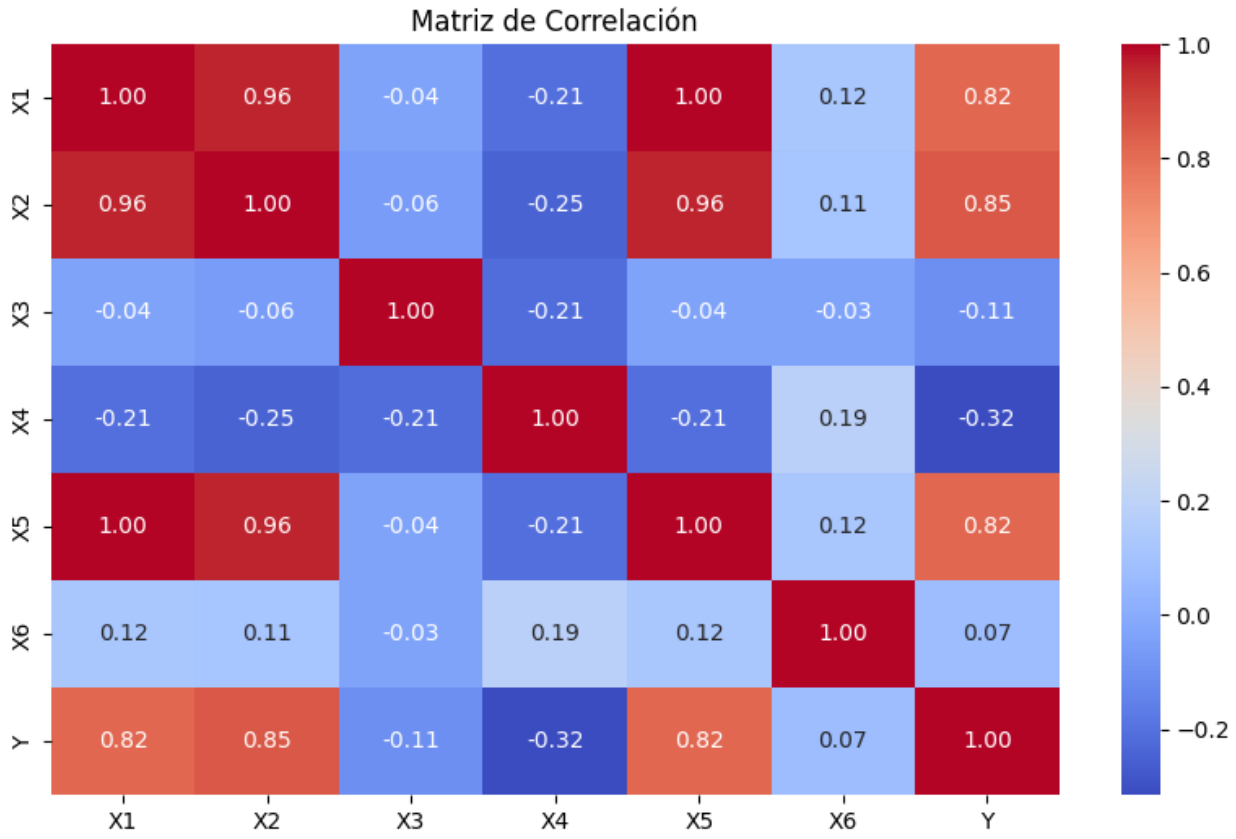
Ejercicio en Python: Uso de una Matriz de Correlación para la Selección de Variables en un Sistema No Lineal

En el contexto de la ciencia de datos, la **selección de variables** sigue siendo crucial cuando trabajamos con sistemas no lineales. Aunque la matriz de correlación es más comúnmente usada para sistemas lineales, también puede ser útil en sistemas no lineales para eliminar variables redundantes que afecten el rendimiento de un modelo, como la **regresión polinómica**. En este ejercicio, aplicaremos un modelo de regresión polinómica en lugar de un modelo lineal para capturar las relaciones no lineales entre las variables.

```
# 1. Importar Librerías Necesarias

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
# 2. Crear un DataFrame de Ejemplo con Variables Correlacionadas

# Generamos un conjunto de datos donde las relaciones entre las
variables pueden no ser estrictamente lineales.

np.random.seed(42)
X1 = np.random.rand(100)
X2 = (X1 ** 2) + 0.1 * np.random.rand(100)
data = pd.DataFrame({
    'X1': X1,
    'X2': X2,  # Relación no lineal con X1
    'X3': np.random.rand(100),  # Variable independiente
    'X4': np.random.rand(100),  # Variable independiente
    'X5': (2*X1) + 5, # Alta correlación con X1
    'X6': np.random.rand(100),  # Variable independiente
    'Y': (2*X1**2 + 3*X2**2) + np.random.normal(0, 1, 100)  # Relación
no lineal con X1 y X2
})

# 3. Calcular la Matriz de Correlación

# Aunque estamos trabajando con relaciones no lineales, la matriz de
correlación puede ofrecer
# algunas pistas iniciales sobre la redundancia entre variables. Sin
embargo, la correlación lineal
# podría no ser suficiente para capturar todas las dependencias en
sistemas no lineales.

corr_matrix = data.corr()


### # 4. Visualizar la Matriz de Correlación

# Usamos `seaborn` para visualizar la matriz de correlación,
recordando que las relaciones no lineales
# no siempre se reflejan completamente en una matriz de correlación
lineal.


plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación')
plt.show()


# 5. Identificar Columnas con Correlación Alta

# Definimos un umbral de correlación para identificar posibles
variables correlacionadas de manera lineal.
```

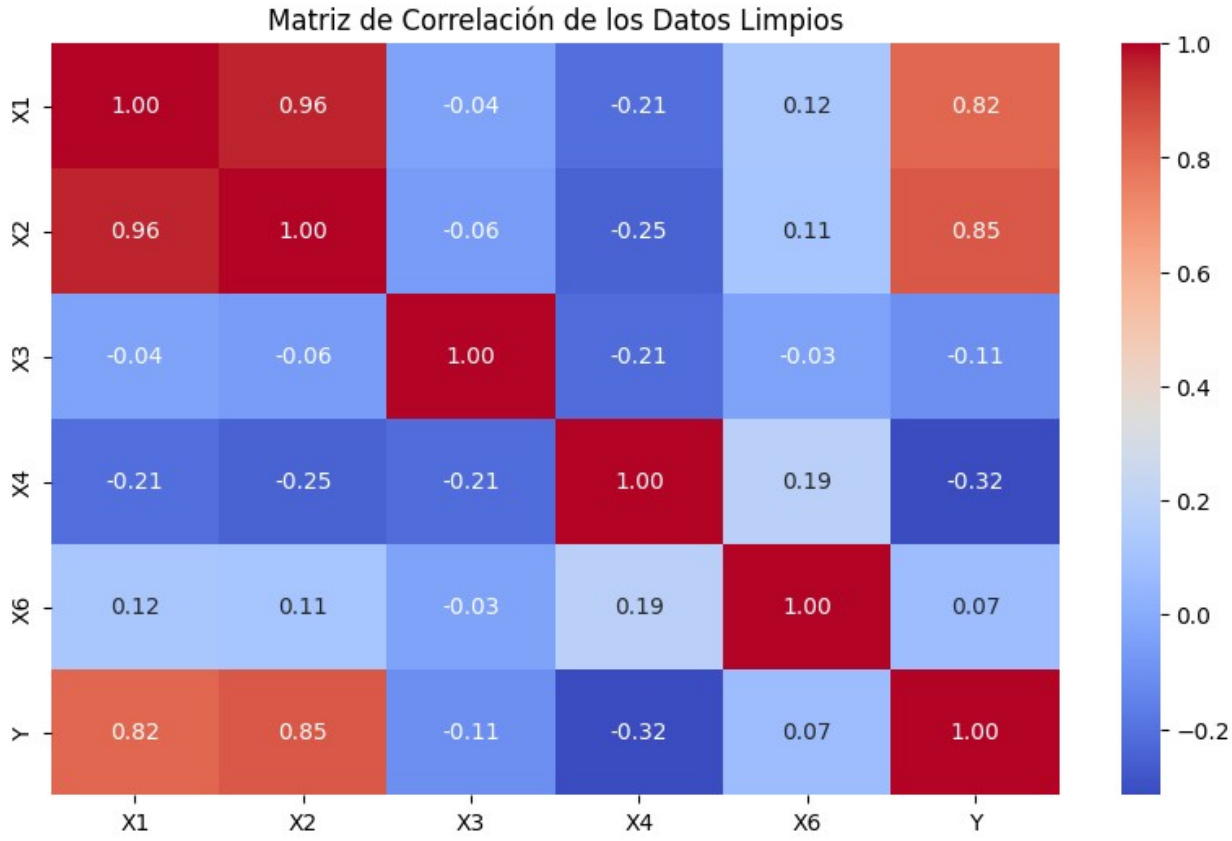
```
# Aunque no todas las relaciones no lineales aparecerán aquí, es útil
reducir la multicolinealidad lineal.


threshold = 0.8
to_drop = []
for column in corr_matrix.columns:
    if any((corr_matrix[column].abs() > threshold) &
(corr_matrix.index != column)):
        to_drop.append(column)

print(f"Variables altamente correlacionadas con otras: {to_drop}")
```




```
Variables altamente correlacionadas con otras: ['X1', 'X2', 'X5', 'Y']
```

```
# 6. Eliminar Variables Redundantes

# Eliminamos las variables que están altamente correlacionadas, como
`X5`,
# ya que no aporta información adicional al modelo que `X1` no
proporcione.

data_cleaned = data.drop(columns=['X5'])
```

```
### # 7. Visualizar la Nueva Matriz de Correlación

# Volvemos a calcular la matriz de correlación de los datos limpios,
eliminando las variables redundantes.

corr_matrix = data_cleaned.corr()

plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación de los Datos Limpios')
plt.show()
```




```
# 8. Aplicar Regresión Polinómica

# Dividimos los datos en **entrenamiento y prueba** y luego aplicamos
un **modelo de regresión polinómica** para
# capturar la relación no lineal entre las variables. En este caso,
generamos características polinómicas de grado 2.

X = data_cleaned.drop(columns='Y')
y = data_cleaned['Y']
```

```
# Dividir los datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Crear características polinómicas de grado 2
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Aplicar regresión lineal a los términos polinómicos
model = LinearRegression()
model.fit(X_train_poly, y_train)

# 9. Predecir y Evaluar el Modelo

# Utilizamos el modelo polinómico para predecir los valores y luego
evaluamos su rendimiento.

y_pred = model.predict(X_test_poly)

# Comparar los primeros 10 valores entre el valor predicho y el valor
real
comparison_df = pd.DataFrame({
    'Valor Real': y_test.head(10).values,
    'Valor Predicho': y_pred[:10]
})

comparison_df['Error Porcentual'] = ((comparison_df['Valor Real'] -
comparison_df['Valor Predicho']) / comparison_df['Valor Real']) * 100

print("\nComparación de los primeros 10 valores entre el valor
predicho y el valor real:\n", comparison_df)

### # 10. Evaluar el Modelo

# Calculamos el **error cuadrático medio (MSE)** y el valor **R²**
para medir la calidad del ajuste
# de la regresión polinómica.

mse = mean_squared_error(y_test, y_pred)
print(f"\nError cuadrático medio (MSE) del modelo: {mse:.4f}")

r2 = r2_score(y_test, y_pred)
print(f"\nValor R² del modelo: {r2:.4f}")
```

```
Comparación de los primeros 10 valores entre el valor predicho y el
valor real:
    Valor Real  Valor Predicho  Error Porcentual
0   -0.542472        0.448266        182.633891
1    3.711657        4.036052         -8.739907
```

```
2    2.408853        3.582134        -48.707028
3   -0.573611        1.508604        363.001153
4   -1.021992       -0.747443         26.864037
5    2.009990        0.846650         57.877893
6    0.332164        0.315672          4.965185
7    3.924838        4.630264        -17.973381
8   -1.709694       -0.733735         57.083851
9    0.684027        1.409638       -106.079207

Error cuadrático medio (MSE) del modelo: 0.8487

Valor R² del modelo: 0.7238
```

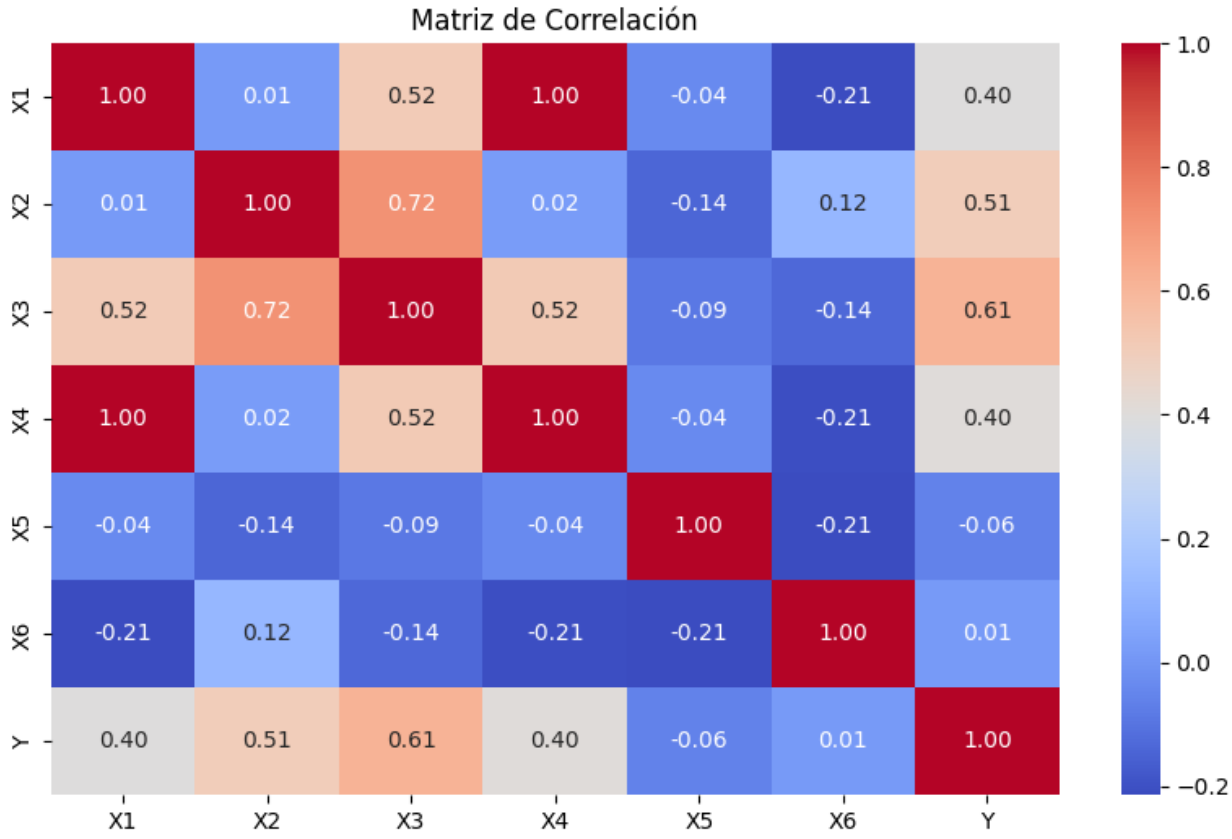
En este ejercicio, utilizamos una **regresión polinómica** para capturar las relaciones no lineales en los datos. Aunque la matriz de correlación es útil para detectar multicolinealidad lineal, en sistemas no lineales es importante considerar métodos adicionales como la **regresión polinómica** o modelos más avanzados para capturar la verdadera naturaleza de las relaciones entre variables. Este enfoque es fundamental en ciencia de datos para modelos no lineales y mejora significativamente la precisión predictiva cuando las relaciones entre las variables no son lineales.

Ejercicio en Python: Uso de una Matriz de Correlación para la Selección de Variables en un Sistema No Lineal con Funciones de Base Radial

En este ejercicio, vamos a trabajar con un sistema no lineal utilizando **regresión con funciones de base radial (RBF)**. Esta técnica se basa en el uso de funciones no lineales para modelar la relación entre las variables, siendo útil en contextos donde las relaciones entre las variables no pueden ser capturadas adecuadamente por modelos polinómicos o lineales. Aunque el enfoque de **matriz de correlación** sigue siendo útil para eliminar variables redundantes, en sistemas no lineales complejos es necesario usar técnicas avanzadas como RBF para obtener un mejor ajuste.

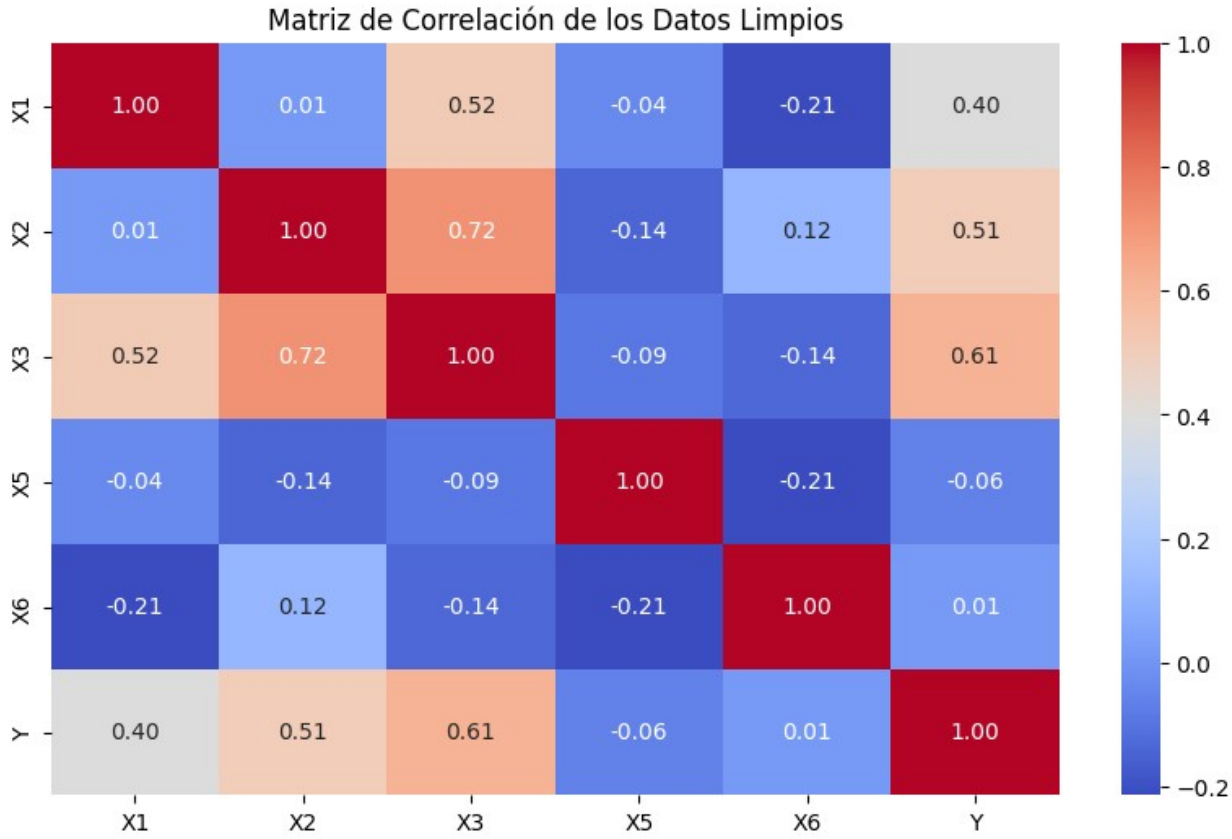
```python
# 1. Importar Librerías Necesarias

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.kernel_ridge import KernelRidge

# 2. Crear un DataFrame de Ejemplo con Variables Correlacionadas

# Generamos un conjunto de datos con relaciones no lineales que se
modelarán usando funciones de base radial.
```

```
np.random.seed(42)
X1 = np.random.rand(100)
X2 = np.sin(X1 * 2 * np.pi)**2 +  np.random.rand(100)
data = pd.DataFrame({
    'X1': X1,
    'X2': X2,  # Relación no lineal con X1
    'X3': X1*X2**2, # Relación no lineal con X1 y X2
    'X4': X1 + np.sin(X1),  # Relación con X1
    'X5': np.random.rand(100),  # Variable independiente
    'X6': np.random.rand(100),  # Variable independiente
    'Y': np.sin(X1 * 2 * np.pi)**2 + np.cos(X2 * 2 * np.pi) + X1*X2  #
Relación no lineal con X1 y X2
})

# 3. Calcular la Matriz de Correlación

# Aunque trabajamos con relaciones no lineales, calculamos una matriz
de correlación para identificar cualquier
# redundancia entre las variables que podrían ser eliminadas.


corr_matrix = data.corr()


# 4. Visualizar la Matriz de Correlación

# Visualizamos la matriz de correlación para tener una idea de las
dependencias lineales.


plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación')
plt.show()


# 5. Identificar y Eliminar Variables Redundantes

# Definimos un umbral de correlación y eliminamos variables
redundantes, aunque en este caso podrían
# no reflejar todas las relaciones no lineales.


threshold = 0.8
to_drop = []
for column in corr_matrix.columns:
    if any((corr_matrix[column].abs() > threshold) &
(corr_matrix.index != column)):
        to_drop.append(column)

print(f"Variables altamente correlacionadas con otras: {to_drop}")
```




```
Variables altamente correlacionadas con otras: ['X1', 'X4']
```

```
# 6. Eliminar Variables Redundantes

# Eliminamos las variables que están altamente correlacionadas, como `X4`,
# ya que no aporta información adicional al modelo que `X1` no proporcione.

data_cleaned = data.drop(columns=['X4'])

# 7. Visualizar la Nueva Matriz de Correlación

# Visualizamos la nueva matriz de correlación después de limpiar las variables redundantes.

corr_matrix = data_cleaned.corr()

plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Matriz de Correlación de los Datos Limpios')
plt.show()
```




```
# 8. Aplicar Regresión con Funciones de Base Radial (RBF)

# Dividimos los datos en **entrenamiento y prueba** y luego aplicamos
un modelo de regresión con
# funciones de base radial para capturar las relaciones no lineales.

X = data_cleaned.drop(columns='Y')
y = data_cleaned['Y']

# Dividir los datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Aplicar Kernel Ridge con RBF como kernel
model = KernelRidge(kernel='rbf', alpha=1.0, gamma=1.0)
model.fit(X_train, y_train)

# 9. Predecir y Evaluar el Modelo

# Utilizamos el modelo para predecir los valores de prueba y evaluar
su rendimiento.

y_pred = model.predict(X_test)
```

```python
# Comparar los primeros 10 valores entre el valor predicho y el valor
real
comparison_df = pd.DataFrame({
    'Valor Real': y_test.head(10).values,
    'Valor Predicho': y_pred[:10]
})

comparison_df['Error Porcentual'] = ((comparison_df['Valor Real'] -
comparison_df['Valor Predicho']) / comparison_df['Valor Real']) * 100

print("\nComparación de los primeros 10 valores entre el valor
predicho y el valor real:\n", comparison_df)


# 10. Evaluar el Modelo

# Calculamos el **error cuadrático medio (MSE)** y el valor **R²**
para medir la precisión del ajuste.

mse = mean_squared_error(y_test, y_pred)
print(f"\nError cuadrático medio (MSE) del modelo: {mse:.4f}")

r2 = r2_score(y_test, y_pred)
print(f"\nValor R² del modelo: {r2:.4f}")
```

```
Comparación de los primeros 10 valores entre el valor predicho y el
valor real:
    Valor Real  Valor Predicho  Error Porcentual
0    1.200539        0.831457         30.742998
1    1.818525        1.406686         22.646833
2    1.715522        1.553147          9.465055
3    1.321186        1.081620         18.132659
4    1.130184        0.903636         20.045234
5    1.405740        1.165041         17.122543
6    1.299639        1.059127         18.506065
7    2.227126        1.429390         35.819070
8   -0.323978       -0.271175         16.298545
9   -0.273828        0.186217        168.005082

Error cuadrático medio (MSE) del modelo: 0.2429

Valor R² del modelo: 0.7417
```

En este ejercicio, utilizamos **regresión con funciones de base radial (RBF)** para modelar las relaciones no lineales en el conjunto de datos. La regresión RBF es especialmente útil cuando las relaciones entre las variables no pueden ser capturadas adecuadamente por modelos lineales o polinomiales. Aunque la matriz de correlación nos ayuda a identificar y eliminar redundancias lineales, la verdadera complejidad de un sistema no lineal requiere modelos avanzados como el RBF para capturar con precisión la dinámica subyacente.