

Creado por:

Isabel Maniega

```
In [1]: from IPython import display
```

3.2.1 – Comprender y aplicar el bootstrap para distribuciones de muestreo.

El **bootstrapping** (o bootstrap) es un método de remuestreo propuesto por Bradley Efron en 1979. Se utiliza para aproximar la distribución en el muestreo de un estadístico. Se usa frecuentemente para aproximar el sesgo o la varianza de un análisis estadístico, así como para construir intervalos de confianza o realizar contrastes de hipótesis sobre parámetros de interés. En la mayor parte de los casos no pueden obtenerse expresiones cerradas para las aproximaciones bootstrap y por lo tanto es necesario obtener remuestras en un ordenador para poner a prueba el método. La enorme potencia de cálculo de los ordenadores actuales facilita considerablemente la aplicabilidad de este método tan costoso computacionalmente.

El método bootstrap reemplaza procedimientos analíticos complejos por un análisis empírico intensivo de la computadora. El método bootstrap se basa en gran medida en el MCM (Método de MonteCarlo), donde se extraen varias remuestras aleatorias de una muestra original dada.

El bootstrapping en estadística es una técnica de remuestreo que se utiliza para estimar la precisión y variabilidad de una medida estadística. Implica extraer múltiples muestras aleatorias con reemplazo del conjunto de datos original, creando nuevas muestras "bootstrap". Estas muestras se utilizan luego para calcular la estadística de interés, como la media o la desviación estándar. Al tomar muestras repetidas de los datos originales, el bootstrapping proporciona una estimación de la distribución de muestreo y permite el cálculo de intervalos de confianza. Esta técnica es particularmente útil cuando la distribución de datos subyacente es desconocida o no normal. Por lo tanto, el bootstrapping ayuda a los estadísticos a realizar inferencias más sólidas y extraer conclusiones confiables de sus datos.

Enfoque

Una distribución de muestreo describe la probabilidad de obtener cada valor posible de un estadístico de una muestra aleatoria de una población; en otras palabras, qué proporción de todas las muestras aleatorias de ese tamaño ofrecerá ese valor. El procedimiento de bootstrap es un método que estima la distribución de muestreo al tomar múltiples muestras con reemplazo de una sola muestra aleatoria. Estas nuevas muestras se denominan muestras repetidas. Cada muestra tiene el mismo tamaño que la muestra original.

La muestra original representa la población de la cual se extrajo. Por lo tanto, las muestras repetidas de esta muestra original representan lo que obtendríamos si tomáramos muchas muestras de la población. La distribución bootstrap de un estadístico, basada en las muestras repetidas.

Luego, se puede calcular la estadística o el parámetro de interés para cada muestra bootstrap y utilizar los resultados para estimar su distribución, media, desviación estándar, intervalo de confianza o cualquier otra medida de incertidumbre.

Bootstrap es útil para el aprendizaje automático porque puede ayudar a evaluar y comparar el rendimiento de sus modelos sin hacer suposiciones sobre la distribución subyacente de los datos o los modelos. Por ejemplo, puede utilizar Bootstrap para:

- Calcule la exactitud , precisión , recuperación , puntuación F1 , curva ROC o cualquier otra métrica de su modelo y su intervalo de confianza .
- Compare el rendimiento de dos o más modelos y pruebe si la diferencia es estadísticamente significativa o no.
- Seleccione el mejor modelo entre un conjunto de candidatos según el método de clasificación bootstrap o los métodos de agregación bootstrap .

A continuación se muestra un proceso paso a paso de bootstrap paramétrico:

- Elija un modelo paramétrico: seleccione un modelo estadístico que represente mejor la distribución subyacente de los datos. Por ejemplo, puede suponer una distribución normal.
- Estimar parámetros: utilice el conjunto de datos original para estimar los parámetros de la distribución elegida (por ejemplo, media y desviación estándar para una distribución normal).
- Generar remuestreos: extrae aleatoriamente muestras con reemplazo de la distribución estimada. Estos remuestreos se tratan como si fueran nuevos conjuntos de datos.
- Calcular estadísticas: para cada nueva muestra, calcule la estadística deseada (por ejemplo, media, mediana, varianza).
- Analizar resultados: agregue las estadísticas de las remuestras para crear una distribución y estimar intervalos de confianza u otras propiedades de interés.

El bootstrapping en estadística es una técnica versátil con una amplia gama de aplicaciones. A continuación, se indican algunas de las más comunes:

Estimación de errores estándar e intervalos de confianza

- Estadísticas complejas: cuando se trabaja con estadísticas que carecen de soluciones analíticas para errores estándar (por ejemplo, mediana, percentiles), el bootstrap puede ser una estimación confiable.
- Tamaños de muestra pequeños: a veces los métodos tradicionales pueden no ser precisos con muestras pequeñas, pero el bootstrapping aún puede proporcionar estimaciones razonables.

Prueba de hipótesis

- Pruebas no paramétricas: el bootstrap se puede utilizar para crear pruebas de hipótesis cuando no se cumplen los supuestos paramétricos.
- Estadísticas de prueba complejas: para estadísticas de prueba complejas sin distribuciones conocidas, puede ayudar a determinar valores p.

Evaluación del modelo

- Validación del modelo: el bootstrap en estadística se puede utilizar para evaluar la estabilidad y confiabilidad de un modelo remuestreando los datos y reajustando el modelo varias veces.
- Intervalos de predicción: puede ayudar a estimar intervalos de predicción para nuevos puntos de datos.

Otras aplicaciones

- Detección de valores atípicos: también puede ayudar en la identificación de valores atípicos mediante el seguimiento de puntos de datos inusuales comparando sus distribuciones bootstrap con los datos originales.
- Análisis de series de tiempo: análisis de datos dependientes del tiempo mediante el arranque de los residuos o la creación de series de tiempo sustitutas.
- Análisis de supervivencia: este enfoque también ayuda a estimar las curvas de supervivencia y los intervalos de confianza.

¿Cuáles son las limitaciones de bootstrap? Bootstrap no es una solución única para todos los problemas de estimación. Antes de aplicarlo, debe ser consciente de sus limitaciones y suposiciones. Por ejemplo, se requiere una muestra grande y representativa de la población para capturar la verdadera variabilidad y sesgo en los datos. Además, puede no funcionar bien para muestras pequeñas o dispersas, o para muestras con valores atípicos o valores extremos. Además, puede no ser válido para algunas estadísticas o parámetros que dependen del orden o la estructura de los datos, como series temporales, datos espaciales o datos jerárquicos. Por último, puede ser computacionalmente intensivo y consumir mucho tiempo, especialmente para conjuntos de datos grandes o complejos, ya que implica generar y analizar muchas muestras de arranque.

Diferenciar entre tipos de datos discretos y continuos en el contexto del bootstrap.

En estadística, las variables desempeñan un papel crucial en la comprensión y el análisis de los datos. Dos tipos fundamentales de variables son las variables discretas y las continuas. Las variables discretas tienen valores distintos y separados con espacios entre ellos, mientras que las variables continuas tienen una secuencia ininterrumpida de valores.

¿Qué son las variables discretas?

Una variable discreta es un tipo de variable que solo puede adoptar valores específicos o distintos. Estos valores suelen ser números enteros. Las variables discretas suelen

representar recuentos o categorías.

Ejemplos de variables discretas son:

- Número de alumnos en un aula: Es una variable discreta porque sólo puede tomar valores numéricos enteros (por ejemplo, 25 alumnos, 30 alumnos).
- Resultados de lanzar un dado de seis caras: el resultado será (1, 2, 3, 4, 5 o 6), que son discretos porque constan de categorías distintas y separadas.
- Número de libros en un estante: El número de libros es discreto porque no puede tomar valores fraccionarios o continuos (por ejemplo, 5 libros, 10 libros, 15 libros).

¿Qué son las variables continuas?

Una variable continua es un tipo de variable que puede tomar cualquier valor dentro de un rango determinado. A diferencia de las variables discretas, que constan de valores distintos e independientes, las variables continuas pueden representar una cantidad infinita de valores posibles, incluidos valores fraccionarios y decimales. Las variables continuas suelen representar medidas o cantidades.

Ejemplos de variables continuas son:

- Altura: La altura es una variable continua porque puede tomar cualquier valor dentro de un rango (por ejemplo, 150,5 cm, 162,3 cm, 175,9 cm).
- Peso: El peso es continuo porque se puede medir con precisión y puede tomar cualquier valor dentro de un rango (por ejemplo, 55,3 kg, 68,7 kg, 72,1 kg).
- Tiempo: El tiempo se puede medir con precisión y puede tomar cualquier valor (por ejemplo, 10:30:15.5 AM, 10:45:30.75 AM).

Aspecto	Variables discretas	Variable continua
Naturaleza de los valores	Sólo pueden tomar valores específicos o discretos.	Pueden tomar cualquier valor dentro de un rango específico.
Escala de medición	Las variables discretas normalmente se miden en una escala nominal u ordinal.	Las variables continuas normalmente se miden en una escala de intervalo o de razón.
Representación	Las variables discretas a menudo se representan mediante gráficos de barras o histogramas.	Las variables continuas a menudo se representan mediante gráficos de líneas o curvas suaves.
Ejemplos	Los ejemplos incluyen el número de estudiantes en una clase o los resultados de tirar un dado.	Los ejemplos incluyen medidas como longitud, tiempo o temperatura.
Distribuciones de probabilidad	Las variables discretas tienen funciones de masa de probabilidad (PMF)	Las variables continuas tienen funciones de densidad de probabilidad (PDF).
Aplicaciones	Se emplean en diversos contextos y aplicaciones matemáticas donde se cuentan cantidades.	A menudo se emplean en diversas ramas de las matemáticas, incluido el cálculo, las ecuaciones diferenciales y el análisis real, así como en campos aplicados como la física, la ingeniería y la estadística.

Reconocer situaciones y tipos de datos en los que el bootstrap es un método eficaz para estimar distribuciones de muestreo.

Aplicaciones del método bootstrap

Entrenamiento de algoritmos de aprendizaje automático

Los algoritmos de aprendizaje automático se pueden entrenar con una muestra inicial. El bootstrapping añade otra dimensión a este proceso al volver a muestrear esta muestra inicial para producir muestras simuladas, a las que se exponen los algoritmos después del entrenamiento. Este proceso proporciona una imagen más clara de cómo funcionan los algoritmos de aprendizaje automático fuera del entrenamiento.

Prueba de hipótesis

Los métodos estadísticos tradicionales suelen intentar hacer generalizaciones sobre un conjunto de datos basándose en una única muestra. Al extraer información de miles de muestras simuladas, el método bootstrap permite determinar cálculos más precisos.

Creación de intervalos de confianza

Al calcular una estadística de interés, el método bootstrap puede generar miles de muestras simuladas, cada una de las cuales presenta su propia estadística de interés. Los equipos pueden desarrollar un intervalo de confianza que sea más preciso, ya que se basa en una colección más grande de muestras en lugar de solo una muestra o unas pocas muestras.

Cálculo del error estándar

El bootstrapping es más eficaz que los métodos tradicionales para calcular el error estándar, ya que genera muchas muestras simuladas al azar. Esto facilita la determinación de las medias de diferentes muestras y la estimación de una distribución de muestreo que refleje mejor el conjunto de datos más amplio y que pueda utilizarse para hallar el error estándar.

Ventajas de las estadísticas de bootstrap

Una comparación de los resultados derivados del enfoque tradicional y el enfoque bootstrap.

Ahora que comprendemos el método bootstrap, debemos tener en cuenta que los resultados obtenidos son básicamente idénticos a los del método tradicional. Además, el método bootstrap siempre funcionará porque no presupone ninguna distribución subyacente de los datos.

Esto contrasta con el enfoque tradicional que, en teoría, supone que los datos se distribuyen normalmente. Si sabemos cómo funciona el enfoque bootstrap, es posible que nos preguntemos si este enfoque se basa demasiado en los datos observados. Es

una buena pregunta, dado que las remuestras se derivan de la muestra inicial y, por ello, es lógico suponer que un valor atípico distorsionará las estimaciones de las remuestras.

“Las ventajas del bootstrap son que es una forma sencilla de derivar las estimaciones de los errores estándar y los intervalos de confianza, y es conveniente porque evita el costo de repetir el experimento para obtener otros grupos de datos muestreados”.

Si bien esto es cierto, si se considera el enfoque tradicional, un valor atípico dentro del conjunto de datos también sesgará la media e inflará el error estándar de la estimación. Si bien puede ser tentador pensar que un valor atípico puede aparecer varias veces dentro de los datos remuestreados y sesgar los resultados y, por lo tanto, mejorar el enfoque tradicional, el enfoque bootstrap depende tanto de los datos como el enfoque tradicional.

“Las ventajas del bootstrapping son que es una forma sencilla de derivar las estimaciones de los errores estándar y los intervalos de confianza, y es conveniente ya que evita el costo de repetir el experimento para obtener otros grupos de datos muestreados. Aunque es imposible conocer el intervalo de confianza real para la mayoría de los problemas, el bootstrapping es asintóticamente consistente y más preciso que el uso de los intervalos estándar obtenidos utilizando la varianza de la muestra y el supuesto de normalidad”, según el autor Graysen Cline en su libro , *Nonparametric Statistical Methods Using R*.

Ambos enfoques requieren el uso de muestras seleccionadas adecuadamente para hacer inferencias sobre las poblaciones. Sin embargo, la mayor diferencia entre estos dos métodos es la mecánica detrás de la estimación de la distribución de muestreo. El procedimiento tradicional requiere que uno tenga una estadística de prueba que satisfaga supuestos particulares para lograr resultados válidos, y esto depende en gran medida del diseño experimental. El enfoque tradicional también utiliza la teoría para decir cómo debería ser la distribución de muestreo, pero los resultados se desmoronan si los supuestos de la teoría no se cumplen.

Por otro lado, el método bootstrap toma los datos de la muestra original y luego los vuelve a muestrear para crear muchas muestras. Este enfoque no se basa en la teoría, ya que se puede observar la distribución de la muestra y no hay que preocuparse por ninguna suposición. Esta técnica permite realizar estimaciones precisas de las estadísticas, lo que es crucial cuando se utilizan datos para tomar decisiones.

Limitaciones de las estadísticas de bootstrap

El bootstrapping ofrece muchos beneficios en comparación con los métodos estadísticos tradicionales, pero hay algunas desventajas a considerar:

- Requiere mucho tiempo: se necesitan miles de muestras simuladas para que el bootstrap sea preciso.
- Computacionalmente exigente: debido a que el bootstrap requiere miles de muestras y lleva más tiempo completarlo, también demanda mayores niveles de potencia computacional.

- A veces incompatible: el bootstrapping no siempre es la mejor opción para una situación, especialmente cuando se trata de datos espaciales o series de tiempo .
- Propenso a sesgos: el bootstrap no siempre tiene en cuenta la variabilidad de las distribuciones, lo que genera errores y sesgos al realizar los cálculos.

¿Se puede utilizar el bootstrap para cualquier tipo de datos?

El bootstrapping se puede aplicar a distintos tipos de datos, incluidos los numéricos, categóricos e incluso los de series temporales. Es un método versátil que no requiere suposiciones específicas sobre la distribución o la estructura de los datos. Sin embargo, la eficacia del bootstrapping puede depender de la naturaleza de los datos y de la pregunta de investigación.

Por ejemplo, el método bootstrap puede ser menos eficaz para muestras de tamaño pequeño, ya que el proceso de remuestreo puede no captar adecuadamente la variabilidad de la población. Además, ciertos tipos de datos, como los altamente correlacionados o dependientes, pueden requerir técnicas de bootstrap especializadas para tener en cuenta la estructura específica de los datos.

Tipos de método Bootstrap

El método Bootstrap se puede clasificar en varios tipos, cada uno adaptado a diferentes necesidades estadísticas y características de los datos. Estas variaciones permiten flexibilidad en el enfoque y la aplicación. Estos son algunos de los tipos más destacados:

1. Bootstrap no paramétrico: la forma más común, no asume ninguna distribución subyacente específica. Remuestrea directamente a partir de los datos, manteniendo la distribución empírica de la muestra original.
2. Bootstrap paramétrico: asume que los datos siguen una distribución específica. Implica ajustar un modelo a los datos y luego remuestrear a partir de este modelo. Es ideal para situaciones en las que se conoce la distribución subyacente o se puede aproximar razonablemente.
3. Bootstrap suavizado: mejora el Bootstrap no paramétrico agregando una pequeña cantidad de ruido aleatorio a los remuestreos. Este tipo es útil para datos con valores discretos o cuando se requiere una estimación más suave.
4. Bootstrap en bloque: diseñado para datos con una estructura inherente, como series temporales, donde las observaciones son dependientes. Remuestrea bloques de datos en lugar de observaciones individuales para preservar la estructura interna.
5. Bootstrap salvaje: se utiliza particularmente en el análisis de regresión. Es adecuado para modelos con errores heterocedásticos e implica un remuestreo de los residuos del modelo ajustado.
6. Bootstrap bayesiano: en lugar de remuestrear puntos de datos, implica remuestrear pesos asociados con cada observación. Este enfoque se alinea más

estrechamente con los métodos inferenciales bayesianos.

7. Bootstrap agrupado: se aplica cuando los datos se organizan en clústeres o grupos. Implica remuestrear clústeres completos en lugar de observaciones individuales para tener en cuenta las correlaciones dentro del grupo.

Cada tipo aborda desafíos y escenarios específicos en el análisis estadístico. Esto muestra la adaptabilidad y la amplia aplicabilidad del método Bootstrap.

Demuestre su competencia en la aplicación de métodos de bootstrap utilizando Python para generar y analizar distribuciones de muestreo.

```
In [2]: import pandas as pd
import numpy as np
import random
```

```
In [3]: # Creamos una muestra aleatoria de 10,000 personas que tienen una edad p
data = np.random.normal(loc = 34, size = 10000)
data
```

```
Out[3]: array([33.580226 , 33.24465275, 34.29120269, ..., 35.74359521,
              35.36097503, 31.44994397], shape=(10000,))
```

```
In [4]: # Comprobamos cuál es el promedio de edad de nuestra base
data.mean()
```

```
Out[4]: np.float64(34.006230725807555)
```

```
In [5]: # Vamos a crear 40 muestras de tamaño 5 para estimar el promedio
promedio = []
for i in range(40):
    muestra = random.sample(data.tolist(), 5)
    print('Lista: ', i, muestra)
    prom = np.mean(muestra)
    promedio.append(prom)
```


Lista: 0 [34.59827032264446, 33.214525742015795, 33.91318747417202, 35.08455869259418, 31.872530098810543]

Lista: 1 [33.92658809147659, 31.681606369203685, 32.80754869213708, 33.37814398447943, 34.46072813065105]

Lista: 2 [35.25250378974439, 34.49965097280347, 37.35698753154476, 33.16250775637381, 34.88948824296923]

Lista: 3 [35.39243838684612, 33.47803141607893, 33.48067744409053, 33.74414767136443, 32.08990466175514]

Lista: 4 [34.21499439002191, 34.59485796514964, 34.96342289600963, 34.31365170607519, 35.296779735327874]

Lista: 5 [34.57608631657874, 33.60262230318072, 34.28053887596791, 34.0261495499985, 34.31326566923845]

Lista: 6 [33.293243710395856, 34.78399129988628, 33.97042600048418, 34.28714323979266, 34.331827444684485]

Lista: 7 [34.32128572052728, 35.512023902282905, 33.70073796965353, 34.04590148197259, 33.98026376014855]

Lista: 8 [32.64113551263233, 32.223778058545875, 34.3559816514147, 34.272845048618336, 34.31182790727442]

Lista: 9 [34.22534897339543, 34.167870525848734, 33.31423843829395, 32.33921542086716, 32.70173877729508]

Lista: 10 [33.57465450187374, 34.06493061553607, 34.99696828946662, 34.773935791876795, 35.5245902783739]

Lista: 11 [34.53320340646105, 35.08002772871288, 35.281987583141486, 34.0656136099896, 34.371763421407856]

Lista: 12 [34.43622553936382, 33.885130707657076, 34.067906913515486, 34.038523437402475, 32.359667725245764]

Lista: 13 [33.14506422548223, 35.276574017558026, 34.175059084598004, 33.4054481984603, 35.15779974874415]

Lista: 14 [34.81521673473674, 34.113020816061855, 32.43431906287408, 33.66711102911786, 36.4460203267042]

Lista: 15 [35.1909014787645, 33.55424654254386, 34.47846681885506, 35.977861683729586, 36.15662778621656]

Lista: 16 [35.76447391991118, 34.978386995713784, 34.141225017577284, 32.837288730836796, 32.902465336737144]

Lista: 17 [35.55329129529035, 34.99764518681727, 34.334369555363295, 33.82148961722238, 33.269749895704166]

Lista: 18 [33.36658315923295, 35.53209840947308, 32.60641955818614, 32.41449451960551, 34.68435701896734]

Lista: 19 [33.30907554424197, 33.00530726319298, 35.7228852204288, 33.077249203823946, 33.496068382559365]

Lista: 20 [32.884241730554436, 33.21462470943568, 35.15779974874415, 34.82088365299948, 34.25960089029228]

Lista: 21 [33.06726295155963, 32.156953525166195, 35.0286417306223, 33.822591764212675, 35.007035556892085]

Lista: 22 [33.74986372895576, 35.23013604973241, 34.741156256424354, 35.22020990658393, 35.194629827879766]

Lista: 23 [34.22924261105238, 34.36626342107779, 35.84012857367448, 33.92916952018842, 34.23583774396996]

Lista: 24 [35.30858356664179, 34.59322857325941, 33.79620384375761, 34.64889138259299, 32.40447655917903]

Lista: 25 [34.153436198265275, 33.446184992452615, 33.503845248505094, 32.747220447536414, 32.131914654313995]

Lista: 26 [33.53079039770142, 33.05359305413297, 34.71885428993437, 33.269749895704166, 33.54035461420341]

Lista: 27 [33.955511170935324, 33.67868303001296, 32.613090771298005, 34.408057198954985, 35.04311309743762]

Lista: 28 [34.1786775546091, 34.6736023675578, 33.506261961888924, 35.02194224778281, 33.469890502371435]

Lista: 29 [34.208738709124354, 33.82080947193985, 35.819421625581995, 34.603290694832324, 34.66452010621334]

```
Lista: 30 [35.42433606929641, 32.88378292674043, 33.73756941113901, 34.56
732923702318, 35.894172724837944]
Lista: 31 [32.24810423861261, 34.78737534163626, 33.64361090635842, 35.09
604478594178, 32.56827508798396]
Lista: 32 [36.492052652394996, 35.55163153974991, 35.11409314525826, 32.2
6717804822398, 34.96759768625911]
Lista: 33 [35.30641796563604, 34.507175187174, 33.822772326108215, 35.373
31971465295, 33.0687144753333]
Lista: 34 [33.42404597949974, 34.931125244470046, 33.708266060235445, 30.
52456862052295, 33.867998221086005]
Lista: 35 [33.18925652858654, 32.8854797829398, 33.55173306940303, 35.464
99010660728, 34.332846354019544]
Lista: 36 [33.8349197353841, 34.404802334517655, 33.36148014791046, 34.09
864961276154, 34.18295385370501]
Lista: 37 [35.27480324340086, 33.32737908822496, 35.17250998103115, 34.42
483213126385, 30.572099242267118]
Lista: 38 [34.26509581997874, 35.69135116309076, 34.643294468682434, 34.3
1382662277446, 34.951836466017866]
Lista: 39 [33.287545195579185, 34.2545507016916, 33.04990257155276, 34.00
847669935423, 35.13836994609089]
```

In [6]: promedio

```
Out[6]: [np.float64(33.7366144660474),
np.float64(33.25092305358957),
np.float64(35.032227658687134),
np.float64(33.63703991602703),
np.float64(34.67674133851685),
np.float64(34.15973254299287),
np.float64(34.13332633904869),
np.float64(34.31204256691697),
np.float64(33.56111363569713),
np.float64(33.349682427140074),
np.float64(34.587015895425424),
np.float64(34.66651914994257),
np.float64(33.75749086463692),
np.float64(34.23198905496854),
np.float64(34.29513759389895),
np.float64(35.071620862021916),
np.float64(34.12476800015524),
np.float64(34.395309110079495),
np.float64(33.720790533093),
np.float64(33.72211712284941),
np.float64(34.06743014640521),
np.float64(33.81649710569057),
np.float64(34.827199153915245),
np.float64(34.520128373992605),
np.float64(34.15027678508617),
np.float64(33.19652030821468),
np.float64(33.62266845033527),
np.float64(33.93969105372778),
np.float64(34.17007492684202),
np.float64(34.62335612153837),
np.float64(34.50143807380739),
np.float64(33.668682072106606),
np.float64(34.87851061437725),
np.float64(34.4156799337809),
np.float64(33.29120082516284),
np.float64(33.88486116831124),
np.float64(33.97656113685575),
np.float64(33.75432473723758),
np.float64(34.77308090810885),
np.float64(33.947769022853734)]
```

```
In [7]: np.mean(promedio)
```

```
Out[7]: np.float64(34.11120382625213)
```

Otro ejemplo:

```
In [8]: # Import necessary libraries
import pandas as pd
import numpy as np
from scipy import stats
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [9]: # Import the weight-height dataset
data = pd.read_csv('weight-height.csv')
```

```
# Display first 5 rows
data.head()
```

```
Out[9]:
```

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

```
In [10]: # Convert inches to centimeters
data["Height(cm)"] = data["Height"]*2.54

# Get summary statistics of Heights in centimeters
data['Height(cm)'].describe()
```

```
Out[10]: count    10000.000000
mean         168.573602
std           9.772721
min          137.828359
25%          161.304276
50%          168.447898
75%          175.702625
max          200.656806
Name: Height(cm), dtype: float64
```

```
In [11]: # Extract 500 random heights
heights = data['Height(cm)'].sample(500).reset_index(drop=True)

# Display Summary Statistics of heights in cm
heights.describe()
```

```
Out[11]: count    500.000000
mean    168.010439
std      9.522859
min     142.439897
25%     161.278866
50%     167.644466
75%     174.795268
max     196.714415
Name: Height(cm), dtype: float64
```

```
In [12]: # Create a function to get x, y for of ecdf
def get_ecdf(data):
    """Returns x,y for ecdf"""
    # Get lenght of the data into n
    n = len(data)

    # We need to sort the data
    x = np.sort(data)

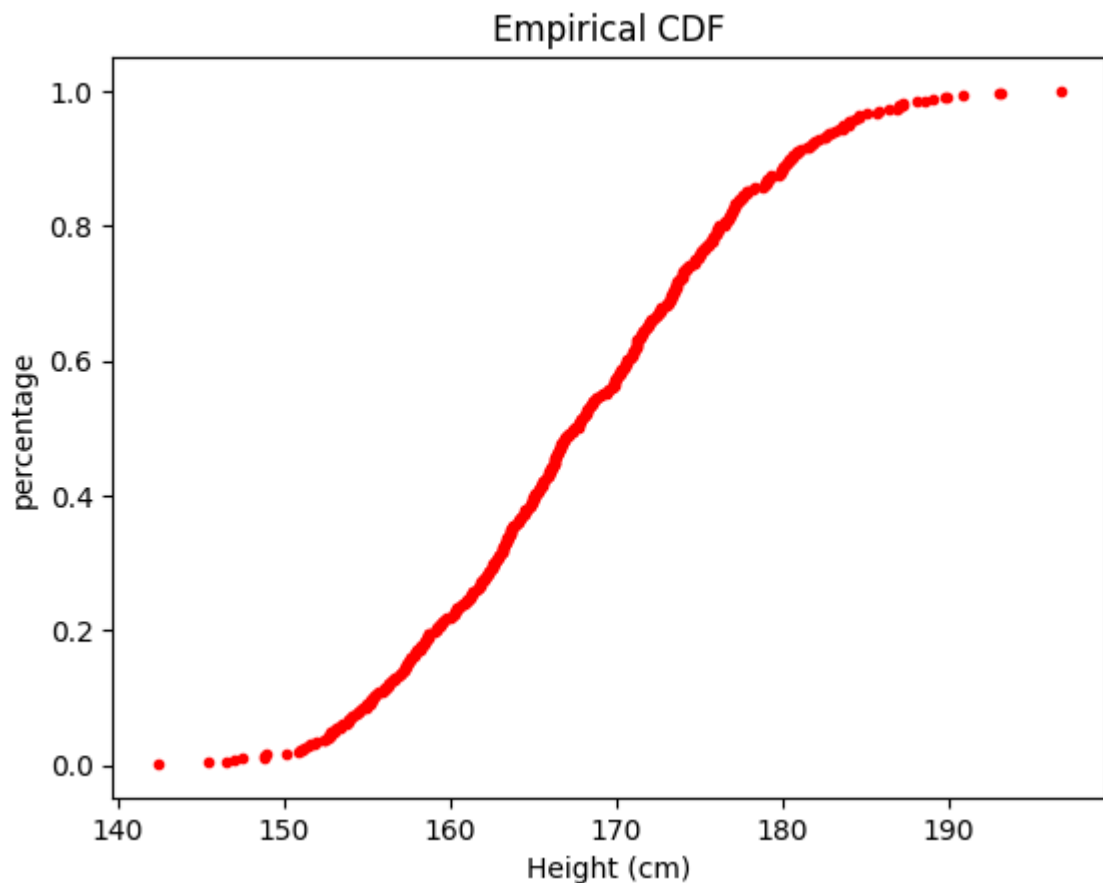
    # the function will show us cumulative percentages of corresponding d
    y = np.arange(1,n+1)/n

    return x,y
```

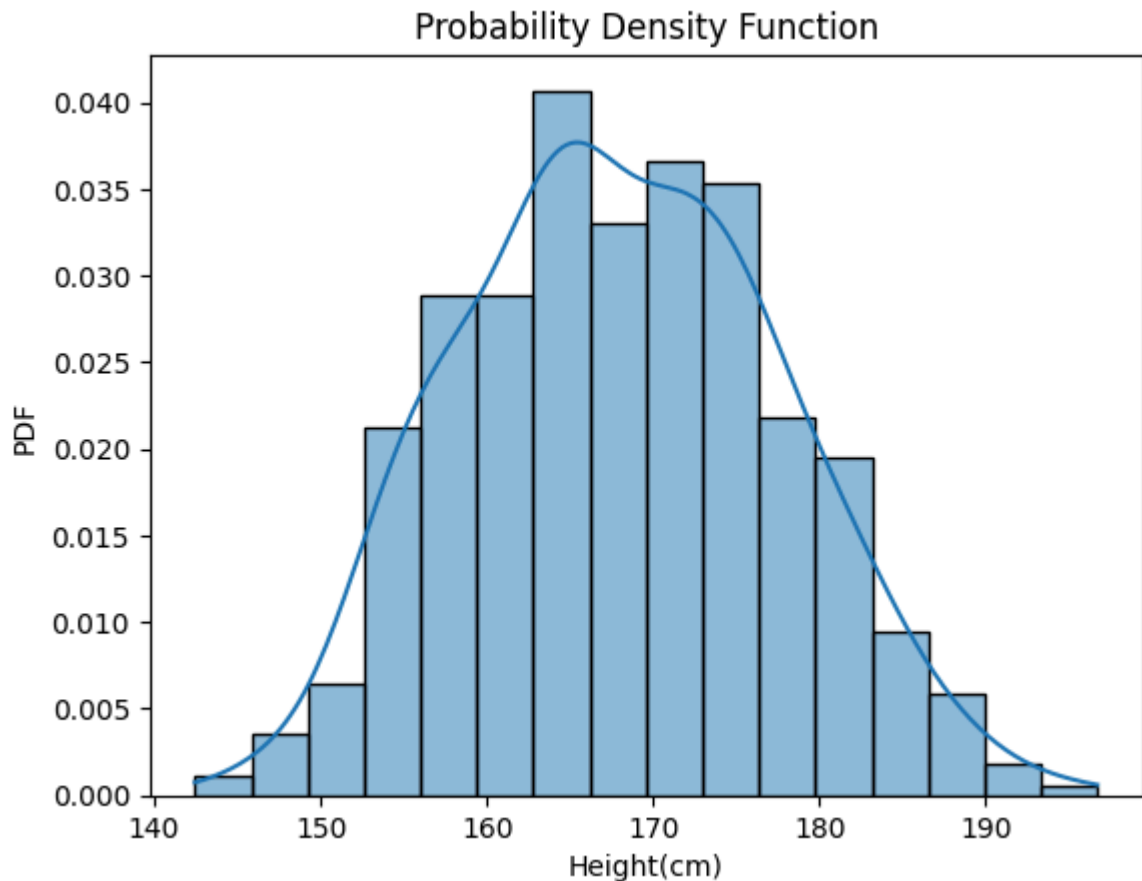
```
# Create a function to plot ecdf
def plot_ecdf(data,labelx,labely,title,color):
    """Plot ecdf"""
    # Call get_ecdf function and assign the returning values
    x, y = get_ecdf(data)

    plt.plot(x,y,marker='.',linestyle='none',c=color)
    plt.xlabel(labelx)
    plt.ylabel(labely)
    plt.title(title)

# Plotting Empirical CDF
plot_ecdf(heights,"Height (cm)","percentage","Empirical CDF","r")
plt.show()
```



```
In [13]: # Plotting PDF
sns.histplot(heights, kde=True, stat="density")
plt.xlabel("Height(cm)")
plt.ylabel("PDF")
plt.title("Probability Density Function")
plt.show()
```



```
In [14]: def draw_bs_replicates(data, func, size):
    """creates a bootstrap sample, computes replicates and returns replicates
    # Create an empty array to store replicates
    bs_replicates = np.empty(size)

    # Create bootstrap replicates as much as size
    for i in range(size):
        # Create a bootstrap sample
        bs_sample = np.random.choice(data, size=len(data))
        # Get bootstrap replicate and append to bs_replicates
        bs_replicates[i] = func(bs_sample)

    return bs_replicates

# Draw 15000 bootstrap replicates
bs_replicates_heights = draw_bs_replicates(heights, np.mean, 15000)

# Print empirical mean
print("Empirical mean: " + str(heights.mean()))

# Print the mean of bootstrap replicates
print("Bootstrap replicates mean: " + str(np.mean(bs_replicates_heights)))
```

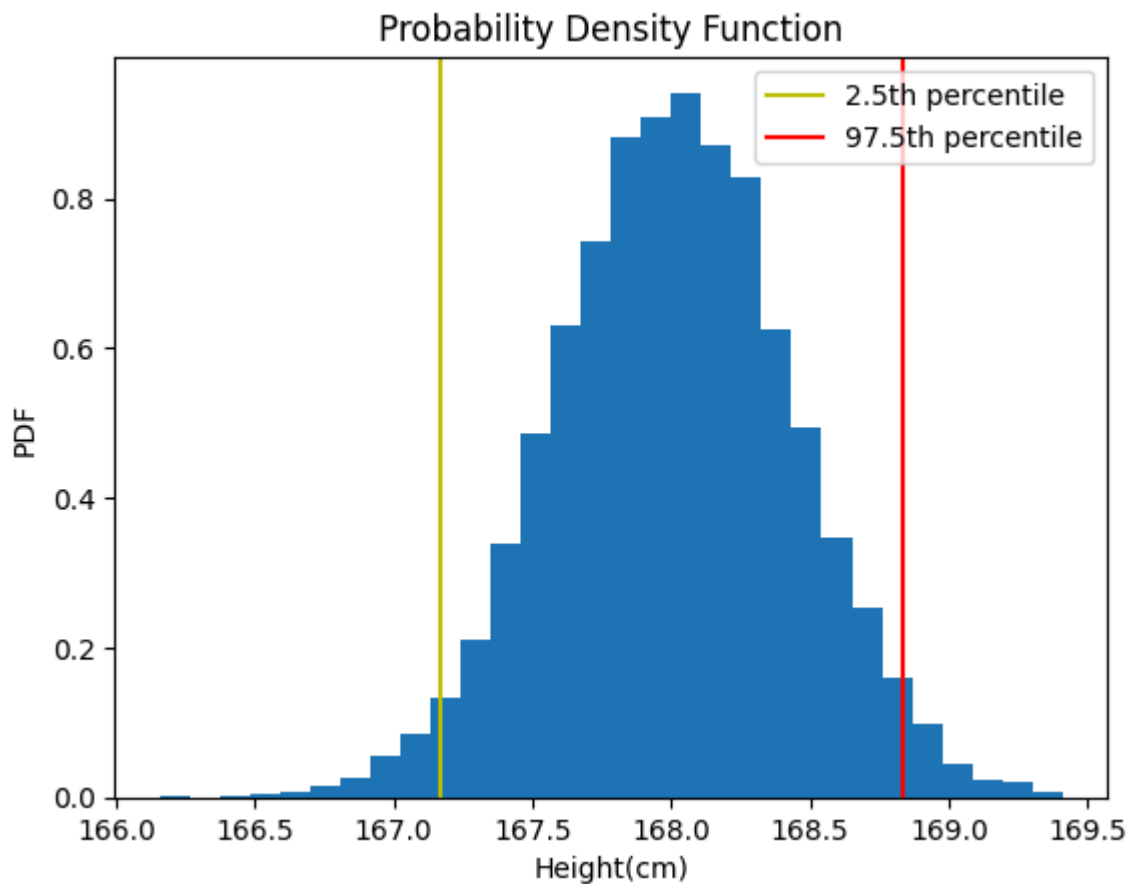
Empirical mean: 168.0104388231265

Bootstrap replicates mean: 168.00793501498194

```
In [15]: # Plot the PDF for bootstrap replicates as histogram
plt.hist(bs_replicates_heights, bins=30, density=True)

# Showing the related percentiles
plt.axvline(x=np.percentile(bs_replicates_heights, [2.5]), ymin=0, ymax=1,
plt.axvline(x=np.percentile(bs_replicates_heights, [97.5]), ymin=0, ymax=1)
```

```
plt.xlabel("Height(cm)")
plt.ylabel("PDF")
plt.title("Probability Density Function")
plt.legend()
plt.show()
```



```
In [16]: # Get the corresponding values of 2.5th and 97.5th percentiles
conf_interval = np.percentile(bs_replicates_heights,[2.5,97.5])

# Print the interval
print("The confidence interval: ",conf_interval)
```

The confidence interval: [167.1655074 168.83172736]

Nuestros límites se encuentran en 167,39 y 169,13. Por lo tanto, podemos afirmar que, si realizamos el mismo experimento con toda la población, la media de las alturas estará entre 167,39 y 169,13 cm con un 95 % de probabilidad.

Analizar la confiabilidad y validez de los resultados obtenidos mediante bootstrap en diversos escenarios estadísticos

¿Qué son las pruebas de calidad de datos?

Las pruebas de calidad de datos implican la evaluación de los datos para garantizar que cumplan con estándares específicos de precisión, integridad, consistencia y otros. Estas pruebas identifican discrepancias y errores en los datos que pueden conducir a análisis o toma de decisiones deficientes. Las pruebas son un proceso continuo que se lleva a

cabo en diversas etapas del manejo de datos, desde la recopilación hasta el procesamiento y el almacenamiento, y que proporciona una visión general de su estado.

Las organizaciones utilizan las pruebas de calidad de datos para mantener su integridad, fiabilidad y confianza. Este proceso ayuda a garantizar que las fuentes de datos produzcan resultados precisos al utilizarlas en análisis, informes o tareas operativas. Al implementar las pruebas de calidad de datos, las empresas pueden minimizar los riesgos asociados con datos inexactos y aumentar la confianza en las decisiones basadas en datos.

Dimensiones clave de la calidad de datos y comprobaciones relevantes de datos

1. Precisión

La precisión en la calidad de los datos garantiza que la información represente correctamente las entidades del mundo real que modela. Es esencial para prevenir errores en las decisiones y operaciones basadas en datos. Los datos precisos deben verificarse con fuentes confiables y refinarse periódicamente para mantener su precisión. Los errores y las discrepancias pueden distorsionar los resultados y generar desconfianza en los activos de datos.

Las comprobaciones relevantes incluyen:

- Verificación de la fuente (Source verification): Comparar los datos con fuentes de referencia fiables para confirmar su exactitud.
- Análisis de umbrales (Threshold analysis): Validar que los valores se encuentren dentro de los umbrales o rangos esperados.
- Detección de anomalías (Anomaly detection): Identificar valores atípicos mediante métodos estadísticos para detectar posibles inexactitudes.
- Validación entre campos (Cross-field validation): Asegurar que las relaciones entre los campos sean lógicas (p. ej., fecha de nacimiento \leq fecha actual).
- Comparación histórica (historical comparison): Evaluar los datos con registros históricos para detectar desviaciones.

2. Integridad

La integridad de los datos se refiere a la medida en que todos los datos necesarios están disponibles. La falta de datos puede dar lugar a interpretaciones incorrectas y a procesos de toma de decisiones deficientes. Para garantizar la integridad, las empresas deben identificar los puntos de datos esenciales y verificar que todos los campos se completen correctamente.

Las comprobaciones relevantes incluyen:

- Comprobaciones de valores nulos (Null value checks): Identificar e informar de los valores que faltan en los campos obligatorios.
- Validación de campos obligatorios (Mandatory field validation): Confirmar que todos los campos críticos estén completados.

- Integridad a nivel de registro (Record-level completeness): Verificar que los conjuntos de datos incluyan todos los registros requeridos.
- Integridad entre sistemas (Cross-system completeness): Asegurar que las transferencias de datos entre sistemas no omitan elementos críticos. Verificación de rango y enumeración (Range and numeration verification): Validar que los campos que permiten un número limitado de valores conocidos se rellenen con los valores adecuados.

3. Consistencia

La consistencia en la calidad de los datos implica mantener la uniformidad en todos los conjuntos de datos, independientemente de la fuente o el formato. La inconsistencia en los datos puede deberse a diversos formatos, diferentes escalas o errores de versiones. Mantener la consistencia requiere políticas coherentes de gestión de datos y comprobaciones periódicas para detectar anomalías.

Las comprobaciones relevantes incluyen:

- Estandarización del formato (Format standarization): Asegurar la consistencia del formato de los datos en todos los conjuntos de datos (p. ej., formatos de fecha, precisión numérica).
- Consistencia a nivel de campo (Field-level consistency): Comparar la uniformidad de los valores repetidos en los conjuntos de datos (p. ej., ID de cliente).
- Validación del control de versiones (Version control validation): Verificar que se utilicen las versiones más recientes de los datos en todos los sistemas.
- Comprobaciones entre sistemas (Cross-system checks): Asegurar la consistencia entre conjuntos de datos o sistemas integrados.
- Consistencia del esquema (Scheme consistency): Confirmar que los esquemas de la base de datos se ajusten a un estándar unificado.

4. Oportunidad

La oportunidad se refiere a la disponibilidad de los datos cuando se necesitan para la toma de decisiones. La información retrasada u obsoleta reduce la eficacia de la información. Garantizar la oportunidad requiere procesos eficientes de recopilación de datos y actualizaciones inmediatas.

Las comprobaciones relevantes incluyen:

- Validación de la marca de tiempo (Timestamp validation): Confirmar que las marcas de tiempo en las entradas de datos se ajusten a los requisitos de procesamiento.
- Monitoreo de la latencia (Latency monitoring): Medir y monitorear los retrasos en los procesos de captura o transferencia de datos.
- Cumplimiento de la programación (Scheduling adherence): Validar que las actualizaciones o entradas de datos se realicen según el cronograma definido.
- Mecanismos de alert (Alert mechanisms): Implementar alertas para actualizaciones de datos retrasadas o registros obsoletos.

5. Singularidad

La Singularidad de los datos garantiza que cada entrada sea distinta, sin duplicaciones. Los datos duplicados pueden saturar los conjuntos de datos, distorsionar el análisis y aumentar los costos de almacenamiento. Es especialmente vital para identificar y gestionar la asignación de recursos, los registros de clientes y las cadenas de suministro.

Las comprobaciones relevantes incluyen:

- Detección de duplicados (Duplicate detection): Utilizar algoritmos para identificar entradas duplicadas en los conjuntos de datos.
- Validación de clave principal (Primary key validation): Asegurarse de que las restricciones de clave principal se apliquen en las bases de datos.
- Comprobación de fusión (Merge check): Confirmar que los conjuntos de datos fusionados no introduzcan registros duplicados.
- Resolución de entidades (Entity resolution): Realizar la correspondencia de entidades para garantizar una representación clara de individuos u objetos.
- Limpieza del almacenamiento (Storage cleanup): Eliminar periódicamente datos redundantes o duplicados.

6. Validez

La validación comprueba que los valores de los datos coincidan con los formatos definidos y se encuentren dentro de los rangos esperados. Los datos no válidos pueden surgir debido a errores de entrada o integraciones de datos defectuosas. Por lo tanto, la validez es crucial para garantizar la precisión y la usabilidad de los datos.

Las comprobaciones relevantes incluyen:

- Validación de formato (Format validation): Verificar si los valores de los datos cumplen con los formatos predefinidos (p. ej., correo electrónico, número de teléfono).
- Validación de rango (Range validation): Asegurarse de que los valores numéricos o de fecha se encuentren dentro de los rangos aceptados.
- Restricciones de dominio (Domain constraints): Verificar que los valores pertenezcan a un conjunto definido de categorías o códigos válidos.
- Validación de dependencia (Dependency validation): Verificar que los valores de los datos cumplan con las reglas de interdependencia (p. ej., que el código postal coincida con la ciudad).
- Coincidencia de expresiones regulares (Regular expression matching): Usar patrones de expresiones regulares para validar campos de texto (p. ej., códigos postales, números de identificación).

3.2.2 – Explicar cuándo y cómo utilizar la regresión lineal y logística.

Comprender la teoría, los supuestos y la base matemática de la regresión lineal.

Regresión Lineal Simple

El objetivo de un modelo de regresión es tratar de explicar la relación que existe entre una variable dependiente (variable respuesta) Y un conjunto de variables independientes (variables explicativas) X_1, \dots, X_n .

En un modelo de regresión lineal simple tratamos de explicar la relación que existe entre la variable respuesta Y y una única variable explicativa X.

```
In [2]: # get the image  
display.Image("./images/rls.png")
```

Out[2]: **Regresión Lineal Simple**

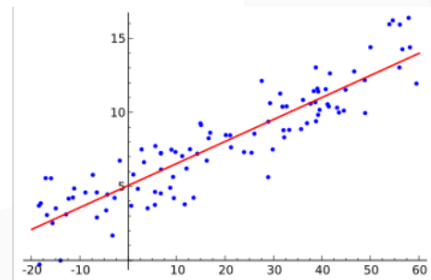
- Basado en la ecuación de la recta:

$$y = b + mx$$

Diagrama de la ecuación de la recta $y = b + mx$ con etiquetas:

- y : Variable dependiente
- b : Constante
- m : Coeficiente
- x : Variable Independiente

[Ir al ejemplo práctico Regresión Lineal Simple](#)



Si todos los puntos (valores medidos) estuvieran exactamente en una línea recta, la estimación sería perfecta. Sin embargo, casi nunca es así y, por tanto, en la mayoría de los casos hay que encontrar una línea recta que se aproxime lo más posible a los puntos de datos individuales. Así pues, se intenta que el error en la estimación sea lo más pequeño posible, de modo que la distancia entre el valor estimado y el valor real sea lo menor posible. Esta distancia o error se denomina "residuo", se abrevia como "e" (error) y se puede representar con la letra griega épsilon (ϵ).

Al calcular la recta de regresión, se intenta determinar los coeficientes de regresión (a y b) de modo que la suma de los residuos al cuadrado sea mínima. (MCO- "Mínimos cuadrados ordinarios")

El coeficiente de regresión b puede tener ahora distintos signos, que pueden interpretarse del siguiente modo

- $b > 0$: existe una correlación positiva entre x e y (a mayor x, mayor y)
- $b < 0$: existe una correlación negativa entre x e y (cuanto mayor es x, menor es y)
- $b = 0$: no hay correlación entre x e y

Los coeficientes de regresión estandarizados suelen designarse con la letra "beta". Son valores comparables entre sí. Aquí ya no importa la unidad de medida de la variable.

Regresión lineal múltiple

A diferencia de la regresión lineal simple, la regresión lineal múltiple permite considerar más de dos variables independientes. El objetivo es estimar una variable en función de otras variables. La variable que hay que estimar se llama variable dependiente (criterio). Las variables que se utilizan para la predicción se denominan variables independientes (predictores).

La regresión lineal múltiple se utiliza con frecuencia en la investigación social empírica, así como en los estudios de mercado. En ambos ámbitos interesa averiguar qué influencia tienen distintos factores en una variable

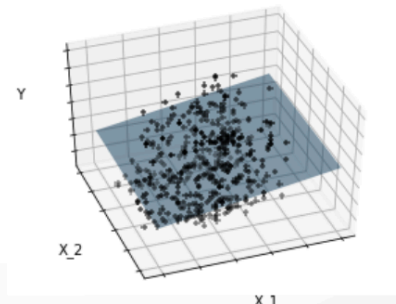
```
In [3]: # get the image
display.Image("./images/rlm.png")
```

Out[3]: **Regresión Lineal Multiple**

- Basado en la ecuación:

$$y = b_0 + m_1 x_1 + m_2 x_2 + \dots + m_n x_n$$

[Ir al ejemplo práctico Regresión Lineal Multiple](#)



Regresión lineal polinomial

Predicción de una variable de respuesta cuantitativa a partir de una variable predictora cuantitativa, donde la relación se modela como una función polinomial de orden n

En estadística, la regresión polinomial es un modelo de análisis de regresión en el que la relación entre la variable independiente X y la variable dependiente Y se modela con un polinomio de n-ésimo grado en X. La regresión polinomial se ajusta a una relación no lineal entre el valor de X y la media condicional correspondiente de Y, denotada $E[Y|X]$. Aunque la regresión polinomial ajusta un modelo no lineal a los datos, como problema de estimación estadística es lineal, en el sentido de que la función de regresión $E[Y|X]$ es lineal en los parámetros desconocidos que se estiman a partir de los datos. Por esta razón, la regresión polinomial se considera un caso especial de regresión lineal múltiple.

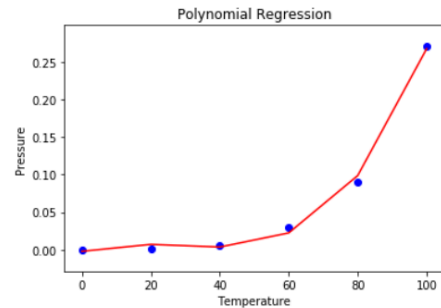
```
In [4]: # get the image
display.Image("./images/rlp.png")
```

Out[4]:

Regresión Lineal Polinomial

- Basado en la ecuación:

$$y = b_0 + m_1 x_1 + m_2 x_1^2 + \dots + m_n x_1^n$$

[Ir al ejemplo práctico Regresión Lineal Polinomial](#)<https://datatab.es/tutorial/linear-regression>

Explicar los conceptos, casos de uso y fundamentos estadísticos de la regresión logística.

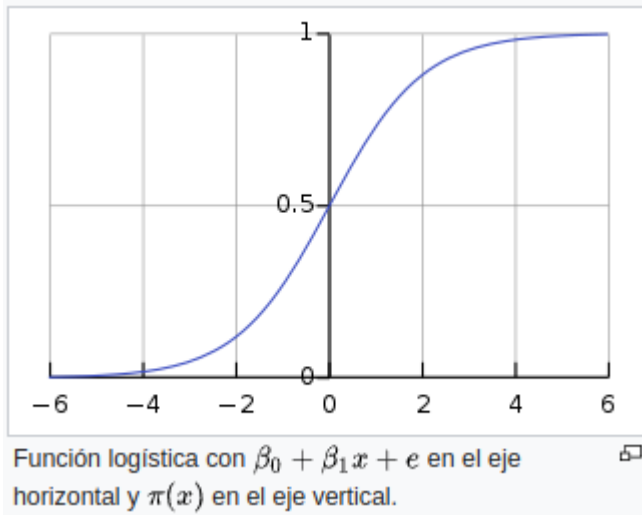
Regresión Logística

Es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes o predictoras.

Trata de correlacionar la probabilidad de una variable cualitativa binaria (asumiremos que puede tomar los valores reales "0" y "1") con una variable escalar x . La idea es que la regresión logística aproxime la probabilidad de obtener "0" (no ocurre cierto suceso) o "1" (ocurre el suceso) con el valor de la variable explicativa x . En esas condiciones, la probabilidad aproximada del suceso se aproximará mediante una función logística

```
In [5]: # get the image
display.Image("./images/rl.png")
```

Out[5]:



Existen tres tipos de modelos de regresión logística, que se definen en función de la respuesta categórica.

- **Regresión logística binaria:** en este enfoque, la respuesta o variable dependiente es de naturaleza dicotómica, es decir, solo tiene dos resultados posibles (p. ej., 0 o 1). Algunos ejemplos populares de su uso incluyen la predicción de si un correo electrónico es spam o no spam, o si un tumor es maligno o no. Dentro de la regresión logística, este es el enfoque más utilizado y, de manera más general, es uno de los clasificadores más comunes para la clasificación binaria.
- **Regresión logística multinomial:** en este tipo de modelo de regresión logística, la variable dependiente tiene tres o más resultados posibles; sin embargo, estos valores no tienen un orden especificado. Por ejemplo, los estudios cinematográficos quieren predecir qué género de película es probable que vea un espectador para comercializar las películas de manera más efectiva. Un modelo de regresión logística multinomial puede ayudar al estudio a determinar la fuerza de influencia que la edad, el sexo y el estado civil de una persona pueden tener en el tipo de película que prefiere. Luego, el estudio puede orientar una campaña publicitaria de una película específica hacia un grupo de personas que probablemente vayan a verla.
- **Regresión logística ordinal:** este tipo de modelo de regresión logística se utiliza cuando la variable de respuesta tiene tres o más resultados posibles, pero en este caso, estos valores tienen un orden definido. Algunos ejemplos de respuestas ordinales incluyen las escalas de calificación de la A a la F o las escalas de calificación del 1 al 5.

Casos de uso de regresión logística

La regresión logística se usa comúnmente para problemas de predicción y clasificación. Algunos de estos casos de uso incluyen:

- **Detección de fraudes:** los modelos de regresión logística pueden ayudar a los equipos a identificar anomalías en los datos, que son predictivas de fraude. Determinados comportamientos o características pueden tener una mayor

asociación con las actividades fraudulentas, lo que resulta especialmente útil para las instituciones bancarias y otras entidades financieras a la hora de proteger a sus clientes. Las empresas basadas en SaaS también han empezado a adoptar estas prácticas para eliminar las cuentas de usuario falsas de sus conjuntos de datos cuando realizan análisis de datos en torno al rendimiento empresarial.

- **Predicción de enfermedades:** en medicina, este enfoque analítico se puede utilizar para predecir la probabilidad de enfermedad o dolencia para una población determinada. Las organizaciones de atención médica pueden establecer atención preventiva para las personas que muestran una mayor propensión a enfermedades específicas.
- **Predicción de abandono:** los comportamientos específicos pueden ser indicativos de abandono en diferentes funciones de una organización. Por ejemplo, los equipos de Recursos Humanos y de gestión pueden querer saber si hay personas de alto rendimiento dentro de la empresa que corren el riesgo de abandonar la organización. Este tipo de información puede suscitar conversaciones para comprender las áreas problemáticas dentro de la empresa, como la cultura o la compensación. Alternativamente, la organización de ventas puede querer saber cuáles de sus clientes corren el riesgo de llevar su actividad a otra parte. Esto puede hacer que los equipos establezcan una estrategia de retención para evitar la pérdida de ingresos.

Desarrollar la capacidad de elegir entre regresión lineal y logística en función de la naturaleza de los datos y la pregunta de investigación.

Aplicar los conceptos de datos discretos y continuos en la elección e implementación de modelos de regresión lineal y logística.

Los modelos de regresión lineal se utilizan para identificar la relación entre una variable dependiente continua y una o más variables independientes. Cuando solo hay una variable independiente y una variable dependiente, se conoce como regresión lineal simple, pero a medida que aumenta el número de variables independientes, se denomina regresión lineal múltiple. Para cada tipo de regresión lineal, busca trazar una línea de mejor ajuste a través de un conjunto de puntos de datos, que normalmente se calcula utilizando el método de mínimos cuadrados.

Al igual que la regresión lineal, la regresión logística también se utiliza para estimar la relación entre una variable dependiente y una o más variables independientes, pero se utiliza para hacer una predicción sobre una variable categórica frente a una continua. Una variable categórica puede ser verdadera o falsa, sí o no, 1 o 0, etcétera. La unidad de medida también difiere de la regresión lineal, ya que produce una probabilidad, pero la función logit transforma la curva S en línea recta.

Si bien ambos modelos se utilizan en el análisis de regresión para hacer predicciones sobre resultados futuros, la regresión lineal suele ser más fácil de entender. La regresión lineal tampoco requiere un tamaño de muestra tan grande como la regresión logística, que necesita una muestra adecuada para representar los valores en todas las categorías de respuesta. Sin una muestra más grande y representativa, es posible que el modelo no tenga suficiente poder estadístico para detectar un efecto significativo.

Diferencias clave entre la regresión lineal y la regresión logística

La regresión logística y la regresión lineal son muy diferentes en sus planteamientos matemáticos.

Valor del resultado

El resultado de la regresión lineal es una escala de valores continua. Esto incluye, por ejemplo, números, kilómetros, precio y peso.

Por el contrario, el resultado del modelo de regresión logística es la probabilidad de que se produzca un suceso categórico fijo. Por ejemplo, 0,76 podría significar una probabilidad del 76 % de llevar una camiseta azul, y 0,22 podría significar una probabilidad del 22 % de votar "sí".

Relación variable

En el análisis de regresión, una línea de regresión es la forma de la línea gráfica que representa la relación entre cada variable independiente y la variable dependiente.

En la regresión lineal, la línea de regresión es recta. Cualquier cambio en una variable independiente tiene un efecto directo en la variable dependiente.

En la regresión logística, la línea de regresión es una curva en forma de S, también conocida como curva sigmoidea.

Tipo de distribución matemática

La regresión lineal sigue una distribución normal o gaussiana de la variable dependiente. Una distribución normal se representa mediante una línea continua en un gráfico.

Una regresión logística sigue una distribución binomial. La distribución binomial se suele representar como un gráfico de barras.

Cuándo utilizar la regresión lineal y cuándo la regresión logística

Puede utilizar la regresión lineal cuando desee predecir una variable dependiente continua a partir de una escala de valores. Utilice la regresión logística cuando espere un resultado binario (por ejemplo, sí o no).

Ejemplos de regresión lineal:

- Predecir la estatura de un adulto en función de la estatura de la madre y del padre

- Predecir el volumen de ventas de calabazas en función del precio, la época del año y la ubicación de la tienda
- Predecir el precio de un billete de avión en función del origen, el destino, la época del año y la compañía aérea
- Predecir el número de “me gusta” en las redes sociales en función del autor de la publicación, su número de seguidores orgánicos, el contenido de la publicación y la hora del día en que se ha publicado

Ejemplos de regresión logística:

- Predecir si una persona padecerá una enfermedad cardíaca en función del IMC, el hábito de fumar y la predisposición genética
- Predecir qué artículos de ropa serán los más populares en función del color, la talla, el tipo y el precio
- Predecir si un empleado renunciará ese año en función del salario, los días en la oficina, el número de reuniones, el número de correos electrónicos enviados, el equipo y la permanencia
- Predecir qué miembros del equipo de ventas tendrán más de un millón de dólares en contratos en un año en función de las ventas del año anterior, la permanencia en el puesto y la tasa de comisiones

Demostrar la aplicación de modelos de regresión lineal y logística en conjuntos de datos utilizando Python, incluida la estimación de parámetros y el ajuste del modelo.

Regresión Lineal Simple

```
In [21]: # pip install scikit-learn
```

```
In [22]: import numpy as np
from sklearn import datasets, linear_model
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score
```

```
In [23]: dataset = datasets.load_diabetes()
print(dataset.DESCR)
# Crear un DataFrame con los datos
data = pd.DataFrame(dataset.data, columns=dataset.feature_names)
data['level'] = dataset.target
data
```

```
.. _diabetes_dataset:
```

```
Diabetes dataset
```

```
-----
```

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of $n = 442$ diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

****Data Set Characteristics:****

:Number of Instances: 442

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression one year after baseline

:Attribute Information:

- age age in years
- sex
- bmi body mass index
- bp average blood pressure
- s1 tc, total serum cholesterol
- s2 ldl, low-density lipoproteins
- s3 hdl, high-density lipoproteins
- s4 tch, total cholesterol / HDL
- s5 ltg, possibly log of serum triglycerides level
- s6 glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the square root of `n_samples` (i.e. the sum of squares of each column totals 1).

Source URL:

<https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>

For more information see:

Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," *Annals of Statistics* (with discussion), 407-499. (https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)

Out[23]:

	age	sex	bmi	bp	s1	s2	s3	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002
...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039

442 rows × 11 columns



```
In [24]: # Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

```
In [25]: # Use only one feature (bmi)
diabetes_X = diabetes_X[:, np.newaxis, 2]

from sklearn.model_selection import train_test_split
# Separo los datos de "train" entrenamiento y "test" prueba para probar l
X_train, X_test, y_train, y_test = train_test_split(diabetes_X, diabetes_y,
```

```
In [26]: # Create linear regression object
regr = linear_model.LinearRegression()
```

```
In [27]: # Train the model using the training sets
regr.fit(X_train, y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(X_test)
```

```
In [28]: # The coefficients
print("Coefficients: \n", regr.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, diabetes_y_
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test, diabetes_y_
print('Valor de la intersección o coeficiente "b":')
print(regr.intercept_)
print()
print('La ecuación del modelo es igual a:')
print('y = ', regr.coef_, 'x ', regr.intercept_)
```

Coefficients:

[961.28582159]

Mean squared error: 4208.02

Coefficient of determination: 0.33

Valor de la intersección o coeficiente "b":

153.2607973612545

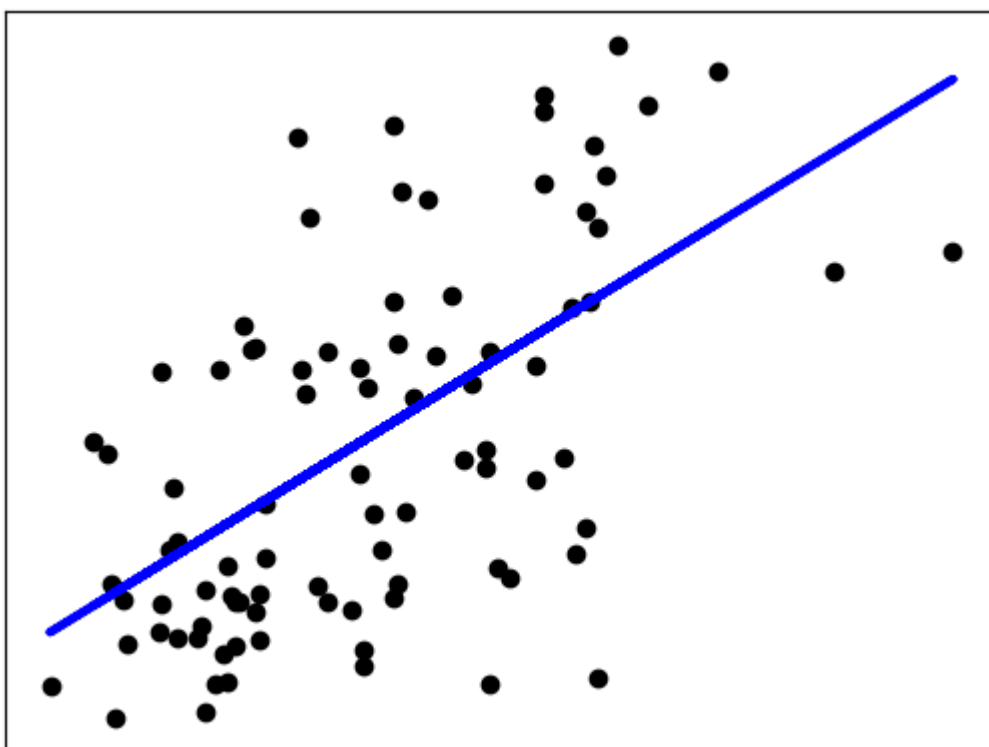
La ecuación del modelo es igual a:

$y = [961.28582159] x + 153.2607973612545$

```
In [29]: # Plot outputs
plt.scatter(X_test, y_test, color="black")
plt.plot(X_test, diabetes_y_pred, color="blue", linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



```
In [30]: print("Precisión del modelo:")
print(regr.score(X_train, y_train))
```

Precisión del modelo:

0.34838459897089347

Regresión Líneal Múltiple

```
In [31]: data
```

```
Out[31]:
```

	age	sex	bmi	bp	s1	s2	s3
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142
...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816

442 rows × 11 columns



```
In [32]: # Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

In [33]: from sklearn.model_selection import train_test_split
# Separo los datos de "train" entrenamiento y "test" prueba para probar l
X_train, X_test, y_train, y_test = train_test_split(diabetes_X, diabetes_

In [34]: lr_multiple = linear_model.LinearRegression()

In [35]: lr_multiple.fit(X_train, y_train)

Out[35]:
▼ LinearRegression ⓘ ?
LinearRegression()

In [36]: y_pred = lr_multiple.predict(X_test)
y_pred
```

```
Out[36]: array([124.20702615, 191.06204032, 288.29840359, 102.37948554,
 192.32260124,  91.41430331, 168.96822655,  99.65058889,
 148.20571883, 110.8849482 ,  57.54470203, 102.28232108,
 156.75102572, 151.19714641,  79.67485923, 158.92485797,
  87.38795309, 280.48229438, 154.88894835, 245.17092909,
  78.94795224, 125.97778591, 116.80339999, 262.4855481 ,
 154.67341443, 186.88453233, 164.09783075, 193.14046207,
 109.82666288, 168.64833131, 155.2753602 , 127.2598638 ,
 122.27904358, 157.62468675,  86.05572464, 123.79378559,
 283.26539588, 169.56797607,  66.73217594,  77.06821601,
 207.781719 , 116.81253899, 115.2054581 , 159.48340447,
 134.28746902, 107.55299534, 114.96527759,  83.27045169,
 140.55079406, 176.61864082, 293.8204808 , 246.9710692 ,
 182.79265564, 108.05673728, 134.22730122, 161.43284428,
 197.24410617, 220.22775341, 153.15694755, 122.81413245,
 231.3937887 , 170.71746664, 122.06088985, 172.56995893,
 161.36697796, 244.53718193, 292.63957893, 192.01463823,
 120.02368014, 192.30421283, 152.33481088, 181.67857424,
 221.73537284, 142.75687792, 101.01189928,  90.87102672,
 180.0951872 , 177.03293765,  69.87842152, 161.47701635,
 141.52538935, 102.95682691, 146.30213708, 142.90006701,
 189.66527664, 244.89615287, 158.64498796, 257.54940227,
 100.47894482])
```

```
In [37]: y_test
```

```
Out[37]: array([ 63., 283., 242.,  88.,  48., 101., 225., 118., 172.,  88.,  85.,
 104., 118., 210.,  37., 206.,  71., 243., 209., 259.,  59.,  44.,
  79., 242., 185.,  91., 127.,  68.,  69., 128.,  55., 145.,  64.,
 141.,  80., 144., 281., 184.,  52.,  48., 198.,  96.,  61., 110.,
  60., 152., 200.,  42., 115., 283., 270., 243., 144.,  69.,  49.,
  58., 281., 248., 136., 214., 236., 122.,  84., 235., 120., 252.,
 230., 257., 177., 272., 134., 161., 332., 187.,  65.,  91., 262.,
 143., 143., 252., 185.,  94., 200., 168.,  67., 132., 262., 310.,
 69.] )
```

```
In [38]: print('DATOS DEL MODELO REGRESIÓN LINEAL MULTIPLE')
print()
print('Valor de las pendientes o coeficientes "a":')
print(lr_multiple.coef_)
print('Valor de la intersección o coeficiente "b":')
print(lr_multiple.intercept_)
```

DATOS DEL MODELO REGRESIÓN LINEAL MULTIPLE

Valor de las pendientes o coeficientes "a":

```
[ -4.06085951 -259.69150028  557.96150037  329.09753441 -611.57040659
  322.34758136  66.49490052  250.09941704  619.31839018  59.31447795]
```

Valor de la intersección o coeficiente "b":

```
153.1194547267447
```

```
In [39]: print("Precisión del modelo:")
print(lr_multiple.score(X_train, y_train))
```

Precisión del modelo:

```
0.527869788470257
```

Regresión Líneal Polinomial

In [40]: data

Out[40]:

	age	sex	bmi	bp	s1	s2	s3	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002
...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039

442 rows × 11 columns



In [41]: `# Load the diabetes dataset`
`diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)`

In [42]: `from sklearn.model_selection import train_test_split`
`X_train, X_test, y_train, y_test = train_test_split(diabetes_X, diabetes_y,`

In [43]: `## Cargar el modelo:`

In [44]: `from sklearn.preprocessing import PolynomialFeatures`

In [45]: `# se define el grado de polinomio`
`poli_reg = PolynomialFeatures(degree= 2)`

In [46]: `# se transforman las características existentes en características de mayor grado`
`X_train_poli = poli_reg.fit_transform(X_train)`
`X_test_poli = poli_reg.fit_transform(X_test)`

In [47]: `pr = linear_model.LinearRegression()`

In [48]: `pr.fit(X_train_poli, y_train)`

Out[48]: `LinearRegression` ⓘ ⓘ
`LinearRegression()`

In [49]: `y_pred = pr.predict(X_test_poli)`
`y_pred`

```
Out[49]: array([ 62.18978875, 176.97559866, 256.65791585, 140.64517018,
 216.64984008,  79.72424599, 229.07043331, 217.0095926 ,
 104.98006978, 155.74135822, 133.74993498, 220.449707 ,
 131.96046565, 195.03397876, 215.13499058, 182.70531385,
 122.23631834, 170.03862448, 113.27695138, 108.26596349,
 188.89418692, 297.26956124, 174.91952705, 198.20835727,
  93.63382709, 102.52912297, 189.98222759, 183.28004252,
 225.38251647, 234.93193443, 123.01868487, 103.07943153,
 208.09794495, 138.31504254, 131.50384406, 121.91771703,
  87.35944768, 294.21563901, 153.08114297,  92.7202512 ,
 177.42123112, 171.31336692, 101.00318658,  70.62421872,
  80.46455507, 115.82863755, 264.27921489, 172.06796472,
 142.01414106, 255.98470281, 199.91877172, 101.05335109,
 171.20578222, 161.07069795, 145.85416772, 305.76806352,
 147.39001859, 234.88251424,  73.3203688 , 169.23153249,
 166.90516729, 158.948224 , 376.6776279 ,  75.40199627,
  60.29034069, 144.73426264, 276.15017365, 209.08698323,
 148.37419838,  57.17025024, 119.00044179, 208.30332536,
 141.44423942, 150.06710965, 208.42636599,  67.7993641 ,
 157.46848742,  49.48908286, 193.75308008,  92.79501353,
 168.69434926, 291.7524701 , 199.9748806 , 291.74170341,
 150.300785 , 169.92829736,  76.87844816, 155.38006474,
 227.8927297 ])
```

```
In [50]: y_test
```

```
Out[50]: array([ 72., 190., 258.,  55., 241.,  55., 280., 259.,  67., 182., 134.,
 168., 131., 122., 306., 281.,  65., 196., 125.,  60., 311., 308.,
  48., 198., 111., 252., 173., 142., 131., 233., 113.,  97., 265.,
 202., 183., 160.,  70., 263., 100.,  77., 178., 172., 178., 137.,
 170.,  71., 217., 181.,  97., 310., 178.,  96., 242.,  93.,  66.,
 132., 197., 257.,  55., 131., 113., 170., 245.,  81., 135., 134.,
 274., 233., 154.,  94., 146., 275.,  85., 162., 200.,  77.,  73.,
  72., 237.,  71., 131.,  84., 198., 268.,  59.,  52.,  54., 121.,
 150.] )
```

```
In [51]: print("Valor de pendiente o coeficiente 'a':")
print(pr.coef_)
```

Valor de pendiente o coeficiente 'a':

```
[-6.35237602e-09  5.95683486e+01 -2.24825143e+02  5.20232077e+02
 3.16993264e+02  7.18628281e+02 -7.47926979e+02 -4.94817844e+02
 4.09486424e+01  3.09261922e+02  9.82979652e+01  1.82470693e+03
 2.97197495e+03 -1.81131529e+03  1.02961273e+03  1.32271810e+03
-7.35433735e+03  2.85081773e+03  4.84234414e+03  1.59816522e+03
 1.85026023e+03 -1.35760263e+00  2.48241058e+03  1.53657614e+03
 6.54469842e+03 -7.14801547e+03  3.58048118e+02  2.06084084e+03
-3.36905008e+03  1.09372883e+03 -1.38174062e+02  3.73625909e+03
-5.22975877e+03  2.79125799e+03  1.52418830e+03 -1.46697100e+03
 5.67380189e+01  2.64034967e+03  4.33932064e+01 -2.33742661e+03
 5.90721556e+03  4.22704963e+02 -3.53860480e+03  2.35823014e+02
-2.86969132e+03  7.59121430e+04 -1.20097636e+05 -1.03641356e+05
-5.41635852e+04 -7.24936220e+04  2.97213884e+03  4.95088787e+04
 8.50632962e+04  4.19393707e+04  6.21467650e+04 -4.96761296e+03
 2.77287154e+04  2.17490636e+04  3.86250021e+04  5.41127736e+03
 5.27774800e+03  1.13026424e+04  8.83822354e+03  1.51306951e+04
-5.55536375e+02  7.05431711e+02]
```

```
In [52]: print("Precisión del modelo: ")
print(pr.score(X_train_poli, y_train))
```


Precisión del modelo:
0.6026934321011834

Logistic Regression

```
In [53]: from sklearn import datasets
```

```
In [54]: dataset = datasets.load_breast_cancer()  
dataset
```

34/63

```

1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1)),
'frame': None,
'target_names': array(['malignant', 'benign'], dtype='<U9'),
'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagno
stic) dataset\n-----\n\n**Data Se
t Characteristics:**\n\nNumber of Instances: 569\n\nNumber of Attribut
es: 30 numeric, predictive attributes and the class\n\nAttribute Inform
ation:\n    - radius (mean of distances from center to points on the per
imeter)\n    - texture (standard deviation of gray-scale values)\n    -
perimeter\n    - area\n    - smoothness (local variation in radius lengt
hs)\n    - compactness (perimeter^2 / area - 1.0)\n    - concavity (seve
rity of concave portions of the contour)\n    - concave points (number o
f concave portions of the contour)\n    - symmetry\n    - fractal dimensi
on ("coastline approximation" - 1)\n\n    The mean, standard error, and
"worst" or largest (mean of the three\n    worst/largest values) of thes
e features were computed for each image,\n    resulting in 30 features.
For instance, field 0 is Mean Radius, field\n    10 is Radius SE, field
20 is Worst Radius.\n\n    - class:\n                - WDBC-Malignant\n
- WDBC-Benign\n\nSummary Statistics:\n\n=====
===== \n                                Min    Max
\n===== \nradius (mean):
6.981  28.11\ntexture (mean):                9.71  39.28\nperime
ter (mean):                43.79  188.5\narea (mean):
143.5  2501.0\nsmoothness (mean):                0.053  0.163\ncompa
ctness (mean):                0.019  0.345\nconcavity (mean):
0.0    0.427\nconcave points (mean):                0.0    0.201\nsymmet
ry (mean):                0.106  0.304\nfractal dimension (mean):
0.05    0.097\nradius (standard error):                0.112  2.873\ntextur
e (standard error):                0.36    4.885\nperimeter (standard erro
r):                0.757  21.98\narea (standard error):                6.802
542.2\nsmoothness (standard error):                0.002  0.031\ncompactness
(standard error):                0.002  0.135\nconcavity (standard error):
0.0    0.396\nconcave points (standard error):                0.0    0.053\nsymmet
ry (standard error):                0.008  0.079\nfractal dimension (standar
d error): 0.001  0.03\nradius (worst):                7.93  3
6.04\ntexture (worst):                12.02  49.54\nperimeter (wor
st):                50.41  251.2\narea (worst):
185.2  4254.0\nsmoothness (worst):                0.071  0.223\ncompa
ctness (worst):                0.027  1.058\nconcavity (worst):
0.0    1.252\nconcave points (worst):                0.0    0.291\nsymmet
ry (worst):                0.156  0.664\nfractal dimension (worst):
0.055  0.208\n===== \n\nMi
ssing Attribute Values: None\n\nClass Distribution: 212 - Malignant, 35
7 - Benign\n\nCreator: Dr. William H. Wolberg, W. Nick Street, Olvi L.
Mangasarian\n\nDonor: Nick Street\n\nDate: November, 1995\n\nThis is a
copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://g
oo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine n
eedle\naspirate (FNA) of a breast mass. They describe\ncharacteristics
of the cell nuclei present in the image.\n\nSeparating plane described a
bove was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennet
t, "Decision Tree\nConstruction Via Linear Programming." Proceedings of
the 4th\nMidwest Artificial Intelligence and Cognitive Science Societ
y,\npp. 97-101, 1992], a classification method which uses linear\nprogra
mming to construct a decision tree. Relevant features\nwere selected us
ing an exhaustive search in the space of 1-4\nfeatures and 1-3 separatin
g planes.\n\nThe actual linear program used to obtain the separating pla

```

ne\nin the 3-dimensional space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. dropdown:: References\n\n - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction\n for breast tumor diagnosis. IS &T/SPIE 1993 International Symposium on\n Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n San Jose, CA, 1993.\n - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and\n prognosis via linear programming. Operations Research, 43(4), pages 570-577,\n July-August 1995.\n - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)\n 163-171.\n',

```
'feature_names': array(['mean radius', 'mean texture', 'mean perimeter',
                        'mean area',
                        'mean smoothness', 'mean compactness', 'mean concavity',
                        'mean concave points', 'mean symmetry', 'mean fractal dimension',
                        'radius error', 'texture error', 'perimeter error', 'area error',
                        'smoothness error', 'compactness error', 'concavity error',
                        'concave points error', 'symmetry error',
                        'fractal dimension error', 'worst radius', 'worst texture',
                        'worst perimeter', 'worst area', 'worst smoothness',
                        'worst compactness', 'worst concavity', 'worst concave points',
                        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv',
'data_module': 'sklearn.datasets.data'}
```

```
In [55]: print("Características del dataset:")
print(dataset.DESCR)
```

Características del dataset:

.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset

****Data Set Characteristics:****

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter² / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.

- class:
 - WDBC-Malignant
 - WDBC-Benign

:Summary Statistics:

=====	=====	=====
	Min	Max
=====	=====	=====
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54

perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

=====

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
<https://goo.gl/U2Uwz2>

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in:
 [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
 cd math-prog/cpo-dataset/machine-learn/WDBC/

.. dropdown:: References

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.

- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) 163-171.

```
In [56]: print("Información en el Dataset:")
         print(dataset.keys())
```

```
Información en el Dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
In [57]: dataset.data
```

```
Out[57]: array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
                1.189e-01],
               [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
                8.902e-02],
               [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
                8.758e-02],
               ...,
               [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
                7.820e-02],
               [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
                1.240e-01],
               [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
                7.039e-02]], shape=(569, 30))
```

```
In [58]: dataset.feature_names
```

```
Out[58]: array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
                'mean smoothness', 'mean compactness', 'mean concavity',
                'mean concave points', 'mean symmetry', 'mean fractal dimension',
                'radius error', 'texture error', 'perimeter error', 'area error',
                'smoothness error', 'compactness error', 'concavity error',
                'concave points error', 'symmetry error',
                'fractal dimension error', 'worst radius', 'worst texture',
                'worst perimeter', 'worst area', 'worst smoothness',
                'worst compactness', 'worst concavity', 'worst concave points',
                'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
In [59]: import pandas as pd

         df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
         df
```

Out[59]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows × 30 columns

In [60]: `X = dataset.data`In [61]: `y = dataset.target`
In [62]: `from sklearn.model_selection import train_test_split`
`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)`
In [63]: `from sklearn.preprocessing import StandardScaler`In [64]: `escalar = StandardScaler()`
In [65]: `X_train = escalar.fit_transform(X_train)`
`X_test = escalar.fit_transform(X_test)`

In [66]: `from sklearn.linear_model import LogisticRegression`
`algoritmo = LogisticRegression()`
In [67]: `algoritmo.fit(X_train, y_train)`Out[67]: `LogisticRegression``LogisticRegression()`
In [68]: `y_pred = algoritmo.predict(X_test)`
`y_pred`


```
Out[68]: array([1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
                1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
                0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
                1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
                0, 0, 1, 1])
```

```
In [69]: y_test
```

```
Out[69]: array([1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
                1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
                1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
                0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
                1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
                0, 0, 0, 0, 1, 1])
```

```
In [70]: from sklearn.metrics import confusion_matrix
```

```
In [71]: matriz = confusion_matrix(y_test, y_pred)
```

```
In [72]: matriz
```

```
Out[72]: array([[36,  1],
                [ 0, 77]])
```

```
In [73]: # Calculo de precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred)
print("Precisión del modelo:")
print(precision)
```

Precisión del modelo:
0.9871794871794872

```
In [74]: # calculo para la exactitud del modelo

from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print("Exactitud del modelo:")
print(exactitud)
```

Exactitud del modelo:
0.9912280701754386

```
In [75]: # Calcular la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred)
print("Sensibilidad del modelo")
print(sensibilidad)
```

Sensibilidad del modelo
1.0

```
In [76]: # Calculo el puntaje F1 del modelo
from sklearn.metrics import f1_score

puntajef1 = f1_score(y_test, y_pred)
```

```
print("Puntaje F1 del modelo")
print(puntajef1)
```

Puntaje F1 del modelo
0.9935483870967742

```
In [77]: # Calculo la curva ROC -AUC del modelo
from sklearn.metrics import roc_auc_score

roc_auc = roc_auc_score(y_test, y_pred)
print("Curva ROC - AUC del modelo:")
print(roc_auc)
```

Curva ROC - AUC del modelo:
0.9864864864864865

Interpretar con precisión los resultados de los análisis de regresión, incluidos los coeficientes y las estadísticas de ajuste del modelo.

```
In [78]: df = pd.DataFrame({'Restaurante': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                           'Poblac. estudiantil (en miles)': [2, 6, 8, 8, 12, 16,
                           'Ventas trimestrales(miles de $)': [58, 105, 88, 118,
                           df
```

```
Out[78]:
```

	Restaurante	Poblac. estudiantil (en miles)	Ventas trimestrales(miles de \$)
0	1	2	58
1	2	6	105
2	3	8	88
3	4	8	118
4	5	12	117
5	6	16	137
6	7	20	157
7	8	20	169
8	9	22	149
9	10	26	202

```
In [79]: X = df['Poblac. estudiantil (en miles)']
y = df['Ventas trimestrales(miles de $)']
```

```
In [80]: # Plot outputs
plt.scatter(X, y, color="black")

plt.xlabel(('Poblac. estudiantil (en miles)'))
plt.ylabel(('Ventas trimestrales(miles de $)'))

plt.show()
```

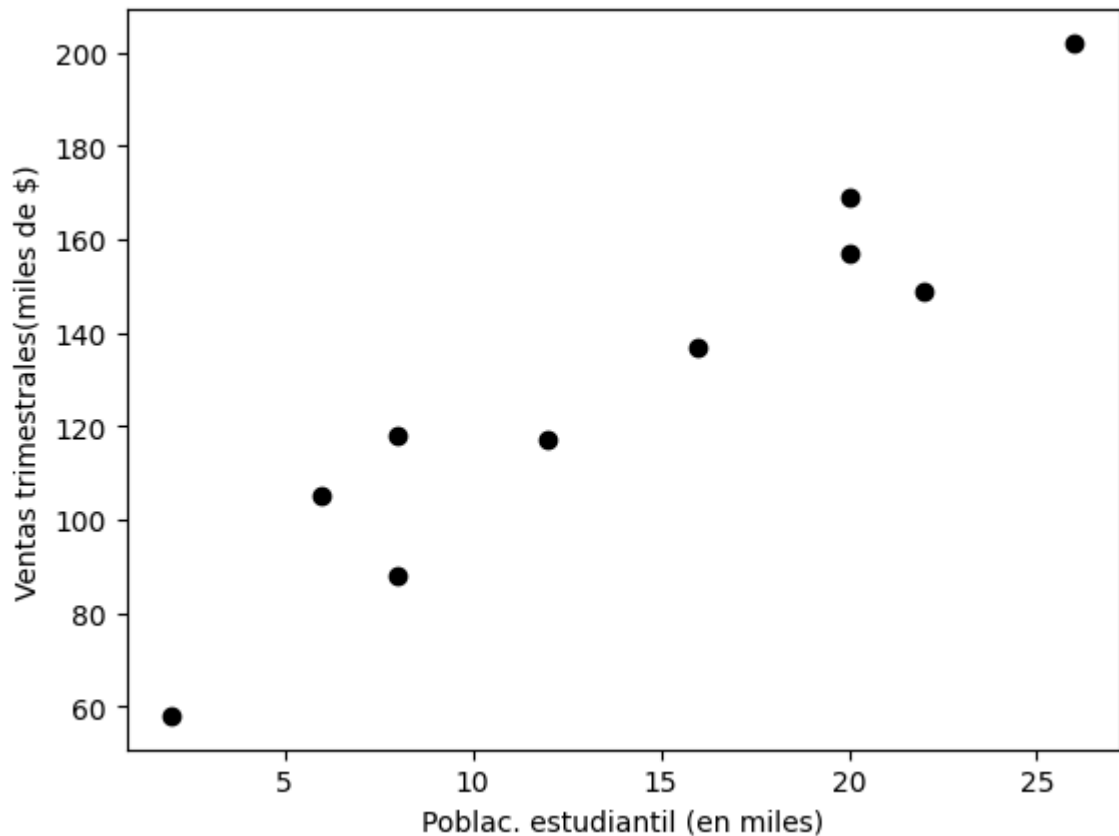


Diagrama de dispersión (y línea de regresión estimada)

$$\hat{y} = b_0 + b_1 x$$

El método de mínimos cuadrados consiste en hallar los valores b_0 y b_1 que hacen mínima la suma de los cuadrados de las desviaciones entre los valores observados de la variable dependiente, y_i , y los valores estimados de la misma, \hat{y}_i . Es decir se minimiza la suma: $\sum (y_i - \hat{y}_i)^2$.

Al aplicar el método se llega al siguiente sistema de ecuaciones simultáneas (llamadas ecuaciones normales de la recta de regresión de y en x), cuya solución da los valores de b_0 y b_1 :

$$\sum y_i = nb_0 + (\sum x_i)b_1$$

$$\sum x_i y_i = (\sum x_i)b_0 + (\sum x_i^2)b_1$$

Las soluciones son las siguientes:

$$b_1 = \frac{\sum x_i y_i - \frac{(\sum x_i)(\sum y_i)}{n}}{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}$$

$$\text{que también es } b_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{S_{xy}}{S_x^2}$$

$$\text{y } b_0 = \bar{y} - b_1 \bar{x}$$

Determine la ecuación de regresión con los datos dados.

```
In [81]: x_total = X.sum()
x_total # 140
```

Out[81]: np.int64(140)

```
In [82]: y_total = y.sum()
y_total # 1300
```

Out[82]: np.int64(1300)

```
In [83]: df['xiyi'] = X * y
df
```

Out[83]:

	Restaurante	Poblac. estudiantil (en miles)	Ventas trimestrales(miles de \$)	xiyi
0	1	2	58	116
1	2	6	105	630
2	3	8	88	704
3	4	8	118	944
4	5	12	117	1404
5	6	16	137	2192
6	7	20	157	3140
7	8	20	169	3380
8	9	22	149	3278
9	10	26	202	5252

```
In [84]: xy_total = df['xiyi'].sum()
xy_total # 21040
```

Out[84]: np.int64(21040)

```
In [85]: df['xi²'] = X ** 2
df
```

Out[85]:

	Restaurante	Poblac. estudiantil (en miles)	Ventas trimestrales(miles de \$)	xiyi	xi²
0	1	2	58	116	4
1	2	6	105	630	36
2	3	8	88	704	64
3	4	8	118	944	64
4	5	12	117	1404	144
5	6	16	137	2192	256
6	7	20	157	3140	400
7	8	20	169	3380	400
8	9	22	149	3278	484
9	10	26	202	5252	676

```
In [86]: xi2_total = df['xi²'].sum()
xi2_total # 2528
```

Out[86]: np.int64(2528)

```
In [87]: Sxy = (X - X.mean())*(y - y.mean())
Sxy = Sxy.sum() / len(X) - 1
Sxy
```

Out[87]: np.float64(283.0)

```
In [88]: Sx2 = (X - X.mean())**2
Sx2 = Sx2.sum() / len(X) - 1
Sx2
```

Out[88]: np.float64(55.8)

```
In [89]: b1 = Sxy / Sx2
b1
```

Out[89]: np.float64(5.07168458781362)

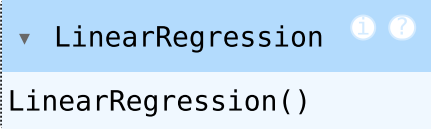
```
In [90]: b0 = y.mean() - (b1 * X.mean())
b0
```

Out[90]: np.float64(58.99641577060932)

Ecuación de la recta: $\$y = 58.996 + 5.0716x\$$

Calculo usando Sklearn:

```
In [91]: x = df.iloc[:,1:2]
y = df.iloc[:,2:3]
# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(x, y)
```

Out[91]: 

```
In [92]: # The coefficients
print("Coefficients: \n", regr.coef_)
print('Valor de la intersección o coeficiente "b":')
print(regr.intercept_)
print()
print('La ecuación del modelo es igual a:')
print('y = ', regr.coef_, 'x ', regr.intercept_)
```

Coefficients:

[[5.]]

Valor de la intersección o coeficiente "b":

[60.]

La ecuación del modelo es igual a:

y = [[5.]] x [60.]

El coeficiente de determinación (r2)

El coeficiente de determinación en la regresión lineal simple es una medida de la bondad de ajuste de la recta estimada a los datos reales. Suma de cuadrados debida al error: $SCE = \sum (y_i - \hat{y}_i)^2$

Suma de cuadrados total: $SCT = \sum (y_i - \bar{y})^2$

Suma de cuadrados debida a la regresión: $SCR = \sum (\hat{y}_i - \bar{y})^2$

Relación entre SCT, SCR y SCE: $SCT = SCR + SCE$

Coeficiente de determinación : $r^2 = \frac{SCR}{SCT} = \frac{SCT - SCE}{SCT} = 1 - \frac{SCE}{SCT}$

Expresado r^2 en porcentaje, se puede interpretar como el porcentaje de la variabilidad total de "Y" que se puede explicar aplicando la ecuación de regresión.

```
In [93]: df['y_resultante'] = b0 + b1*x
df
```

```
Out[93]:
```

	Restaurante	Poblac. estudiantil (en miles)	Ventas trimestrales(miles de \$)	xiyi	xi²	y_resultante
0	1	2	58	116	4	69.139785
1	2	6	105	630	36	89.426523
2	3	8	88	704	64	99.569892
3	4	8	118	944	64	99.569892
4	5	12	117	1404	144	119.856631
5	6	16	137	2192	256	140.143369
6	7	20	157	3140	400	160.430108
7	8	20	169	3380	400	160.430108
8	9	22	149	3278	484	170.573477
9	10	26	202	5252	676	190.860215

```
In [94]: df['y - y_resultante'] = df['Ventas trimestrales(miles de $)'] - df['y_re
df
```

Out[94]:

	Restaurante	Poblac. estudiantil (en miles)	Ventas trimestrales(miles de \$)	xiyi	xi ²	y_resultante	y - y_resultante
0	1	2	58	116	4	69.139785	-11.139785
1	2	6	105	630	36	89.426523	15.573477
2	3	8	88	704	64	99.569892	-11.569892
3	4	8	118	944	64	99.569892	18.430108
4	5	12	117	1404	144	119.856631	-2.856631
5	6	16	137	2192	256	140.143369	-3.143369
6	7	20	157	3140	400	160.430108	-3.430108
7	8	20	169	3380	400	160.430108	8.569892
8	9	22	149	3278	484	170.573477	-21.573477
9	10	26	202	5252	676	190.860215	11.139785

```
In [95]: SCE = (df['y - y_resultante']**2).sum()
SCE
```

```
Out[95]: np.float64(1532.9187703138455)
```

```
In [96]: SCT = ((y - y.mean())**2).sum()
SCT
```

```
Out[96]: Ventas trimestrales(miles de $)    15730.0
dtype: float64
```

```
In [97]: SCR = SCT - SCE
SCR
```

```
Out[97]: Ventas trimestrales(miles de $)    14197.08123
dtype: float64
```

```
In [98]: coeficiente_determinacion = SCR / SCT
coeficiente_determinacion
```

```
Out[98]: Ventas trimestrales(miles de $)    0.902548
dtype: float64
```

El 90.25% de la variación en las ventas se puede explicar con la relación lineal entre la población estudiantil y las ventas.

```
In [99]: X = df.iloc[:, 1:2]
y = df.iloc[:, 2:3]
```

```
In [100]: print("Precisión del modelo:")
print(regr.score(X, y))
```

```
Precisión del modelo:
0.9027336300063573
```

El coeficiente de correlación lineal (r)

Es una medida descriptiva que mide la intensidad de asociación lineal entre las dos variables, x y y . Los valores del coeficiente de correlación lineal siempre están entre -1 y $+1$. -1 significa una relación lineal negativa perfecta, $+1$ significa una relación lineal positiva perfecta. Los valores cercanos a cero indican que las variables x y y no tienen relación lineal. El coeficiente de correlación lineal se relaciona con el coeficiente de determinación así:

$$r = (\pm b_1) \sqrt{r^2}$$

b_1 es la pendiente la recta de regresión de y en x .

El coeficiente de determinación es más general que el coeficiente de correlación lineal.

PRUEBAS DE SIGNIFICACIÓN PARA LA REGRESIÓN LINEAL

La ecuación de regresión lineal simple indica que el valor medio o valor esperado de y es una función lineal de x : $E(y/x) = \beta_0 + \beta_1 x$. Si $\beta_1 = 0$ entonces $E(y/x) = \beta_0$ y en este caso el valor medio no depende del valor de x , y concluimos que x y y no tienen relación lineal. En forma alternativa, si el valor $\beta_1 \neq 0$ llegamos a la conclusión que las dos variables se relacionan (más específicamente, que hay una componente lineal en el modelo). Existen dos pruebas, por lo menos, que se pueden utilizar para tal fin. En ambas se requiere una estimación de σ^2 , la varianza de ϵ en el modelo de regresión.

Cuadrados medios del error CME (es una estimación de σ^2)

$$S^2 = \text{CME} = \text{SCE} / (n-2)$$

$n-2$ son los grados de libertad asociados a SCE. 2 son los parámetros estimados en la regresión lineal (β_0 y β_1) y n es el número de pares de datos.

```
In [101... CME = SCE / (len(df)-2)
CME
```

```
Out[101... np.float64(191.61484628923068)
```

Error estándar de estimación (s)

Es la raíz cuadrada de s^2 , $S = \sqrt{\text{CME}}$ y es el estimador de la desviación estándar σ .

```
In [102... import math

S = math.sqrt(CME)
S
```

```
Out[102... 13.84250144624268
```

Distribución muestral de b_1

b_1 es un estadístico con distribución normal de media $\mu_{b_1} = \beta_1$ y desviación estándar $\sigma_{b_1} = \frac{\sigma}{\sqrt{\sum (x_i - \bar{x})^2}}$. Si sustituimos σ por

su estimación muestral, s , obtenemos un estimador de σ_{b1} que denotaremos por s_{b1} . $s_{b1} = \frac{s}{\sqrt{2} \{\sum (x_i - \bar{x})^2\}}$. Con esta información podemos construir un estadístico t .

$t = \frac{b_1 - \beta_1}{s_{b1}}$ el cual se distribuye con $v=n-2$ g.l.

```
In [103... sb1 = math.sqrt(S / Sx2)
sb1
```

```
Out[103... 0.4980697768594643
```

```
In [104... y.mean()
```

```
Out[104... Ventas trimestrales(miles de $)    130.0
dtype: float64
```

```
In [105... t = (b1 - y.mean()) / sb1 # ???
t
```

```
Out[105... Ventas trimestrales(miles de $)    -250.824927
dtype: float64
```

Prueba t de significación en la regresión

$H_0: \beta_1 = 0$

$H_1: \beta_1 \neq 0$

Estadístico de contraste bajo H_0 , $t_c = \frac{b_1 - 0}{s_{b1}}$

Decisión: Se rechaza H_0 en favor de H_1 si $|t_c| > t_{\alpha/2}$ o si p-valor $< \alpha$.

```
In [106... tc = (b1 - 0) / sb1
tc
```

```
Out[106... np.float64(10.182678860365082)
```

```
In [107... tc > t

# True --> se rechaza H0 y se acepta H1 -->
# beta1 es distinto de 0 llegamos a la conclusión que las dos variables
```

```
Out[107... Ventas trimestrales(miles de $)    True
dtype: bool
```

Prueba de significancia usando el estadístico F (es una prueba más general)

Se usan dos estimaciones de σ^2 , una basada en CME y la otra basada en CMR.

$CME = \frac{SCE}{n-2}$ y $CMR = \frac{SCR}{\text{número de variables independiente}} = \frac{SCR}{1}$

CME es un estimador insesgado de σ^2 , mientras que CMR lo es sólo si H_0 es cierta. Si H_0 es falsa, CMR tiende a sobreestimar σ^2 .

El estadístico de contraste, bajo H_0 es una F. $F = \text{CMR} / \text{CME}$ con 1 gl en el numerador y $n-2$ en el denominador. Los datos se acomodan en una tabla ANOVA. Se rechaza H_0 en favor de H_1 si $F_c > F_{\alpha}$ o también si el p-valor (α)

```
In [108... CME = SCE / len(df) - 2
CME
```

```
Out[108... np.float64(151.29187703138456)
```

```
In [109... CMR = SCR / 1
CMR
```

```
Out[109... Ventas trimestrales(miles de $)    14197.08123
dtype: float64
```

```
In [110... F = CMR / CME
F
```

```
Out[110... Ventas trimestrales(miles de $)    93.839018
dtype: float64
```

```
In [111... anova = pd.DataFrame({'Fuente de variación': ['Regresion', 'Error', 'Total'],
                        'Suma de cuadrados': [SCR, SCE, SCT],
                        'Grados de libertad': ['1', 'n-2', 'n-1'],
                        'Cuadrados medios': [CMR, CME, None],
                        'F': [F, None, None]})
anova
```

```
Out[111...
```

	Fuente de variación	Suma de cuadrados	Grados de libertad	Cuadrados medios	F
0	Regresion	Ventas trimestrales(miles de \$) 14197.08123...	1	Ventas trimestrales(miles de \$) 14197.08123...	Ventas trimestrales(miles de \$) 93.839018 d...
1	Error	1532.91877	n-2	151.291877	None
2	Total	Ventas trimestrales(miles de \$) 15730.0 dty...	n-1	None	None

Uso de la ecuación de regresión lineal para evaluar y predecir.

El modelo de regresión lineal simple es un supuesto acerca de la relación entre x y y . Si los resultados tienen una relación estadísticamente significativa entre x y y , y si el ajuste que proporciona la ecuación de regresión parece bueno, ésta podría utilizarse para estimaciones y predicciones.

Ecuación de la recta: $\hat{y} = 58.996 + 5.0716x$

```
In [112... x = 10
y = 60 + 5*x
print('El valor de y para un valor de x = 10 es ', y)
```

El valor de y para un valor de x = 10 es 110

```
In [113... # Make predictions using the testing set
y_pred = regr.predict([[10]])
y_pred
```

```
/home/isabelmaniega/Documentos/PCAD_ES/env/lib/python3.12/site-packages/sk
learn/utils/validation.py:2739: UserWarning: X does not have valid feature
names, but LinearRegression was fitted with feature names
warnings.warn(
```

```
Out[113... array([[110.]])
```

Coeficientes para regresión logística

```
In [114... from sklearn import datasets
```

```
In [115... dataset = datasets.load_breast_cancer()
dataset
```

```

Out[115... {'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e
-01,
    1.189e-01],
    [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
    8.902e-02],
    [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
    8.758e-02],
    ...,
    [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
    7.820e-02],
    [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
    1.240e-01],
    [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
    7.039e-02]], shape=(569, 30)),
  'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
    0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0,
    1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
0,
    1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1,
    1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0,
    0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1,
    1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1,
    1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0,
    0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0,
    1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
1,
    1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
    0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
0,
    0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0,
    0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
0,
    1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1,
    1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
0,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1,
1,
    1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0,
    1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1,
    1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1,

```

```

1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1)),
'frame': None,
'target_names': array(['malignant', 'benign'], dtype='<U9'),
'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagno
stic) dataset\n-----\n\n**Data Se
t Characteristics:**\n\nNumber of Instances: 569\n\nNumber of Attribut
es: 30 numeric, predictive attributes and the class\n\nAttribute Inform
ation:\n    - radius (mean of distances from center to points on the per
imeter)\n    - texture (standard deviation of gray-scale values)\n    -
perimeter\n    - area\n    - smoothness (local variation in radius lengt
hs)\n    - compactness (perimeter^2 / area - 1.0)\n    - concavity (seve
rity of concave portions of the contour)\n    - concave points (number o
f concave portions of the contour)\n    - symmetry\n    - fractal dimensi
on ("coastline approximation" - 1)\n\n    The mean, standard error, and
"worst" or largest (mean of the three\n    worst/largest values) of thes
e features were computed for each image,\n    resulting in 30 features.
For instance, field 0 is Mean Radius, field\n    10 is Radius SE, field
20 is Worst Radius.\n\n    - class:\n                - WDBC-Malignant\n
- WDBC-Benign\n\nSummary Statistics:\n\n=====
===== \n                                Min    Max
\n===== \nradius (mean):
6.981  28.11\ntexture (mean):                                9.71  39.28\nperime
ter (mean):                                43.79  188.5\narea (mean):
143.5  2501.0\nsmoothness (mean):                                0.053  0.163\ncompa
ctness (mean):                                0.019  0.345\nconcavity (mean):
0.0    0.427\nconcave points (mean):                                0.0    0.201\nsymmet
ry (mean):                                0.106  0.304\nfractal dimension (mean):
0.05   0.097\nradius (standard error):                                0.112  2.873\ntextur
e (standard error):                                0.36   4.885\nperimeter (standard erro
r):                                0.757  21.98\narea (standard error):                                6.802
542.2\nsmoothness (standard error):                                0.002  0.031\ncompactness
(standard error):                                0.002  0.135\nconcavity (standard error):
0.0    0.396\nconcave points (standard error):                                0.0    0.053\nsymmet
ry (standard error):                                0.008  0.079\nfractal dimension (standar
d error): 0.001  0.03\nradius (worst):                                7.93   3
6.04\ntexture (worst):                                12.02  49.54\nperimeter (wor
st):                                50.41  251.2\narea (worst):
185.2  4254.0\nsmoothness (worst):                                0.071  0.223\ncompa
ctness (worst):                                0.027  1.058\nconcavity (worst):
0.0    1.252\nconcave points (worst):                                0.0    0.291\nsymmet
ry (worst):                                0.156  0.664\nfractal dimension (worst):
0.055  0.208\n===== \n\nMi
ssing Attribute Values: None\n\nClass Distribution: 212 - Malignant, 35
7 - Benign\n\nCreator: Dr. William H. Wolberg, W. Nick Street, Olvi L.
Mangasarian\n\nDonor: Nick Street\n\nDate: November, 1995\n\nThis is a
copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://g
oo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine n
eedle\naspirate (FNA) of a breast mass. They describe\ncharacteristics
of the cell nuclei present in the image.\n\nSeparating plane described a
bove was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennet
t, "Decision Tree\nConstruction Via Linear Programming." Proceedings of
the 4th\nMidwest Artificial Intelligence and Cognitive Science Societ
y,\npp. 97-101, 1992], a classification method which uses linear\nprogra
mming to construct a decision tree. Relevant features\nwere selected us
ing an exhaustive search in the space of 1-4\nfeatures and 1-3 separatin
g planes.\n\nThe actual linear program used to obtain the separating pla

```

ne\nin the 3-dimensional space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. dropdown:: References\n\n - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction\n for breast tumor diagnosis. IS &T/SPIE 1993 International Symposium on\n Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n San Jose, CA, 1993.\n - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and\n prognosis via linear programming. Operations Research, 43(4), pages 570-577,\n July-August 1995.\n - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)\n 163-171.\n',

```
'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
                        'mean smoothness', 'mean compactness', 'mean concavity',
                        'mean concave points', 'mean symmetry', 'mean fractal dimension',
                        'radius error', 'texture error', 'perimeter error', 'area error',
                        'smoothness error', 'compactness error', 'concavity error',
                        'concave points error', 'symmetry error',
                        'fractal dimension error', 'worst radius', 'worst texture',
                        'worst perimeter', 'worst area', 'worst smoothness',
                        'worst compactness', 'worst concavity', 'worst concave points',
                        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv',
'data_module': 'sklearn.datasets.data'}
```

La matriz de confusión como herramienta del aprendizaje automático

Imagine que tiene una prueba médica que verifica la presencia o ausencia de una enfermedad; en este caso si utilizáramos un algoritmo de aprendizaje automático basado en clasificación en este caso con variables categóricas, y por el cual usemos posiblemente un algoritmo de árboles decisorios). De cualquier manera, en la vida real, hay dos posibles verdades: lo que se está probando es verdadero o no. La persona está enferma o no lo está; la imagen es un perro, o no lo es.

Debido a esto, también hay dos resultados de prueba posibles: un resultado de prueba positivo (la prueba predice que la persona está enferma o no, o bien en el otro ejemplo la imagen es un perro, o no). Estas 4 opciones se resumen en el siguiente cuadro, debido a que surgen dos posibles valores reales y dos posibles valores de predicción o predictivos



No description has been provided for this image

MATRIZ DE CONFUSIÓN BINARIA

Estas 4 opciones se conocen como: la matriz de confusión.

Veamos de nuevo esos 4 resultados posibles:

- *Verdadero positivo*: El valor real es positivo y la prueba predijo también que era positivo. O bien una persona está enferma y la prueba así lo demuestra.(VP)

- *Verdadero negativo*: El valor real es negativo y la prueba predijo también que el resultado era negativo. O bien la persona no está enferma y la prueba así lo demuestra.(VN)
- *Falso negativo*: El valor real es positivo, y la prueba predijo que el resultado es negativo. La persona está enferma, pero la prueba dice de manera incorrecta que no lo está. Esto es lo que en estadística se conoce como error tipo II.(FN)
- *Falso positivo*: El valor real es negativo, y la prueba predijo que el resultado es positivo. La persona no está enferma, pero la prueba nos dice de manera incorrecta que sí lo está. Esto es lo que en estadística se conoce como error tipo I (FP)

A partir de estas 4 opciones surgen las métricas de la matriz de confusión: Por una parte la exactitud y la precisión y por otra la Sensibilidad y la Especificidad. Veámoslas en detalle:

La Exactitud (en inglés Accuracy) y la Precisión (en inglés Precision)

1. La Exactitud

la Exactitud (en inglés, "Accuracy") se refiere a lo cerca que está el resultado de una medición del valor verdadero. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. Se representa como la proporción de resultados verdaderos (tanto verdaderos positivos (VP) como verdaderos negativos (VN)) dividido entre el número total de casos examinados (verdaderos positivos, falsos positivos, verdaderos negativos, falsos negativos)

En forma práctica, la Exactitud es la cantidad de predicciones positivas que fueron correctas.

$$\frac{VP+VN}{VP+FP+FN+VN}$$

2. La Precisión

La Precisión (en inglés "Precision") Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión. Se representa por la proporción de verdaderos positivos dividido entre todos los resultados positivos (tanto verdaderos positivos, como falsos positivos).

En forma práctica es el porcentaje de casos positivos detectados.

Se calcula como: $\frac{VP}{VP+FP}$

3. Sesgo (también llamado Bias, ó Inaccuracy):

Es la diferencia entre el valor medio y el verdadero valor de la magnitud medida. El sesgo pertenece al concepto de exactitud.

La Sensibilidad (en Inglés Recall o sensitivity) y la Especificidad)(En inglés Specificity)

La sensibilidad y la especificidad son dos valores que nos indican la capacidad de nuestro estimador para discriminar los casos positivos, de los negativos. La sensibilidad se representa como la fracción de verdaderos positivos, mientras que la especificidad, es la fracción de verdaderos negativos.

1. La Sensibilidad ("Recall" o «Sensitivity»),

También se conoce como Tasa de Verdaderos Positivos (True Positive Rate) ó TP. Es la proporción de casos positivos que fueron correctamente identificadas por el algoritmo. Se calcula así: $\frac{VP}{VP+FN}$, o lo que sería igual : Verdaderos positivos / Total Enfermos

En el área de la salud decimos que la sensibilidad es la capacidad de poder detectar correctamente la enfermedad entre los enfermos.

2. La Especificidad («Specificity»)

También conocida como la Tasa de Verdaderos Negativos, ("true negative rate") o TN. Se trata de los casos negativos que el algoritmo ha clasificado correctamente. Expresa cuan bien puede el modelo detectar esa clase. Se calcula: $\frac{VN}{VN+FP}$,

En términos de salud: Verdaderos Negativos / Total Sanos

En el area de la salud decimos que la especificidad es la capacidad de poder identificar los casos de pacientes sanos entre todos los sanos)

Mediciones de la precisión de una prueba

Al calcular las relaciones entre estos valores, podemos medir cuantitativamente la precisión de nuestras pruebas.

1. La tasa de falsos positivos se calcula como $\frac{FP}{FP + TN}$, donde FP es el número de falsos positivos y TN es el número de verdaderos negativos (FP + TN es el número total de negativos). Es la probabilidad de que se produzca una falsa alarma: que se dé un resultado positivo cuando el valor verdadero sea negativo.

Hay muchas otras medidas posibles de precisión de la prueba y tasa de error.

A continuación, se muestra un breve resumen de los más comunes:

2. La **tasa de falsos negativos**, también llamada **tasa de error**, es la probabilidad de que la prueba pase por alto un verdadero positivo. Se calcula como $\frac{FN}{FN + VP}$, donde FN es el número de falsos negativos y VP es el número de verdaderos positivos
3. La **tasa de verdadero positivos** (TVP, también llamada **sensibilidad**) se calcula como $\frac{VP}{VP + FN}$. La tasa de verdaderos positivos es la probabilidad de que un resultado positivo real dé positivo.
4. La **tasa de verdaderos negativos** (también llamada especificidad), que es la probabilidad de que un resultado negativo real dé un resultado negativo. Se calcula

como $\frac{VN}{VN + FP}$.

a) El **valor predictivo positivo** es la probabilidad de que, si ha obtenido un resultado positivo en la prueba, realmente tenga la enfermedad. Se calcula como $\frac{VP}{VP + FP}$.

b) El **valor predictivo negativo**, por el contrario es la probabilidad de que, si ha obtenido un resultado negativo en la prueba, en realidad no tenga la enfermedad.

EL F1 SCORE

Esta es otra métrica muy empleada porque nos resume la precisión y sensibilidad en una sola métrica. Por ello es de gran utilidad cuando la distribución de las clases es desigual, por ejemplo cuando el número de pacientes con una condición es del 15% y el otro es 85% , lo que en el campo de la salud es bastante común.

Se calcula: $F_1 = \frac{2 * Precision * Sensibilidad}{Precision + Sensibilidad}$

RESUMEN

Conforme a estas nuevas métricas podemos obtener cuatro casos posibles para cada clase:

- Alta precisión y alto recall: el modelo de Machine Learning escogido maneja perfectamente esa clase.
- Alta precisión y bajo recall: el modelo de Machine Learning escogido no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
- Baja precisión y alto recall: El modelo de Machine Learning escogido detecta bien la clase, pero también incluye muestras de la otra clase.
- Baja precisión y bajo recall: El modelo de Machine Learning escogido no logra clasificar la clase correctamente.

```
In [116... import pandas as pd

df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df
```

Out[116...

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows × 30 columns

In [117...] `X = dataset.data`In [118...] `y = dataset.target`
In [119...] `from sklearn.model_selection import train_test_split`
`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)`
In [120...] `from sklearn.preprocessing import StandardScaler`In [121...] `escalar = StandardScaler()`
In [122...] `X_train = escalar.fit_transform(X_train)`
`X_test = escalar.fit_transform(X_test)`

In [123...] `from sklearn.linear_model import LogisticRegression`
`algoritmo = LogisticRegression()`
In [124...] `algoritmo.fit(X_train, y_train)`Out[124...] `LogisticRegression``LogisticRegression()`
In [125...] `y_pred = algoritmo.predict(X_test)`
`y_pred`

```
Out[125...] array([0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
        1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
        1, 1, 1, 0])
```

```
In [126...] y_test
```

```
Out[126...] array([0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
        1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
        1, 1, 1, 0])
```

```
In [127...] # Matriz de confusión
# [[VP, FP,
# [FN, VN]]

y_pred = y_pred.tolist()
y_test = y_test.tolist()
```

```
In [128...] from sklearn.metrics import confusion_matrix
```

```
In [129...] matriz = confusion_matrix(y_test, y_pred)
```

```
In [130...] matriz
```

```
Out[130...] array([[39,  1],
        [ 2, 72]])
```

```
In [131...] """
- Verdadero positivo: El valor real es positivo y la prueba predijo t
- Verdadero negativo: El valor real es negativo y la prueba predijo t
- Falso negativo: El valor real es positivo, y la prueba predijo que
  Cancer Maligno pero realmente es Benigno (FN)
- Falso positivo: El valor real es negativo, y la prueba predijo que
  Cancer Benigno pero realmente es Maligno (FP)
M = malignant(1), B = benign (0)
"""

VP = 0
VN = 0
FP = 0

for n, value in enumerate(y_test):
    if value == y_pred[n]:
        if y_pred[n] == 0:
            # Cancer Benigno: Verdadero Positivo
            VP += 1
        else:
            # Cancer Maligno: Verdadero Negativo
            VN += 1
    else:
        # Falso Positivo
        # Prueba Positiva (0), pero tiene cancer Maligno
        if y_pred[n] == 1:
            FP += 1
```

```
# False negativo
# Prueba Negativa (1), pero tiene cancer Benigno
FN = len(y_pred) - (VP + VN + FP)

matriz_confusion = [[VP, FP], [FN, VN]]
matriz_confusion
```

Out[131...] [[39, 1], [2, 72]]

```
In [132...] # Calculo de precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred)
print("Precisión del modelo:")
print(precision)
```

Precisión del modelo:
0.9863013698630136

```
In [133...] precision = VP / (VP+FP)
precision
```

Out[133...] 0.975

```
In [134...] # calculo para la exactitud del modelo

from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print("Exactitud del modelo:")
print(exactitud)
```

Exactitud del modelo:
0.9736842105263158

```
In [135...] exactitud = (VP+VN) / ( VP+FP+FN+VN)
exactitud
```

Out[135...] 0.9736842105263158

```
In [136...] # Calcular la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred)
print("Sensibilidad del modelo")
print(sensibilidad)
```

Sensibilidad del modelo
0.972972972972973

```
In [137...] # En el área de la salud decimos que la sensibilidad es la capacidad de d

sensibilidad = VP / (VP+FN)
sensibilidad
```

Out[137...] 0.9512195121951219

```
In [138...] # Calculo el puntaje F1 del modelo
from sklearn.metrics import f1_score
```

```
puntajef1 = f1_score(y_test, y_pred)
print("Puntaje F1 del modelo")
print(puntajef1)
```

Puntaje F1 del modelo
0.9795918367346939

```
In [139... F1 = (2 * precision * sensibilidad) / (precision + sensibilidad)
F1
```

Out[139... 0.9629629629629629

```
In [140... # En el area de la salud decimos que la especificidad es la capacidad de
# entre todos los sanos
```

```
especificidad = VN / (VN+FP)
especificidad
```

Out[140... 0.9863013698630136

Identificar limitaciones, supuestos y sesgos potenciales en los modelos de regresión lineal y logística y su impacto en los resultados.

Limitaciones de la regresión logística:

- Límite de decisión lineal: el límite de decisión lineal puede limitar la capacidad de la regresión logística para captar relaciones complejas en los datos. Las relaciones no lineales pueden requerir modelos más sofisticados.
- Sensibilidad a los valores atípicos: si bien los valores atípicos afectan a muchos modelos, la regresión logística puede ser particularmente sensible. Los valores extremos pueden distorsionar los coeficientes e influir desproporcionadamente en las predicciones.
- Impacto de las variables irrelevantes: la inclusión de variables irrelevantes puede provocar un sobreajuste y reducir la interpretabilidad del modelo. La selección de características se vuelve crucial para mantener la eficiencia del modelo.
- Supuesto de independencia: si bien se supone que existe independencia, lograrla en datos del mundo real puede ser un desafío. Las series temporales o los datos espaciales pueden violar este supuesto.

Abordar los desafíos:

- Multicolinealidad: detecte la multicolinealidad utilizando factores de inflación de varianza (VIF) y considere eliminar variables altamente correlacionadas o utilizar técnicas de regularización.
- Valores atípicos: identifique y maneje los valores atípicos mediante técnicas o transformaciones robustas. Los análisis de sensibilidad pueden ayudar a medir su impacto.
- Variables irrelevantes: realice una selección exhaustiva de características para incluir solo variables relevantes. Las técnicas de regularización como la

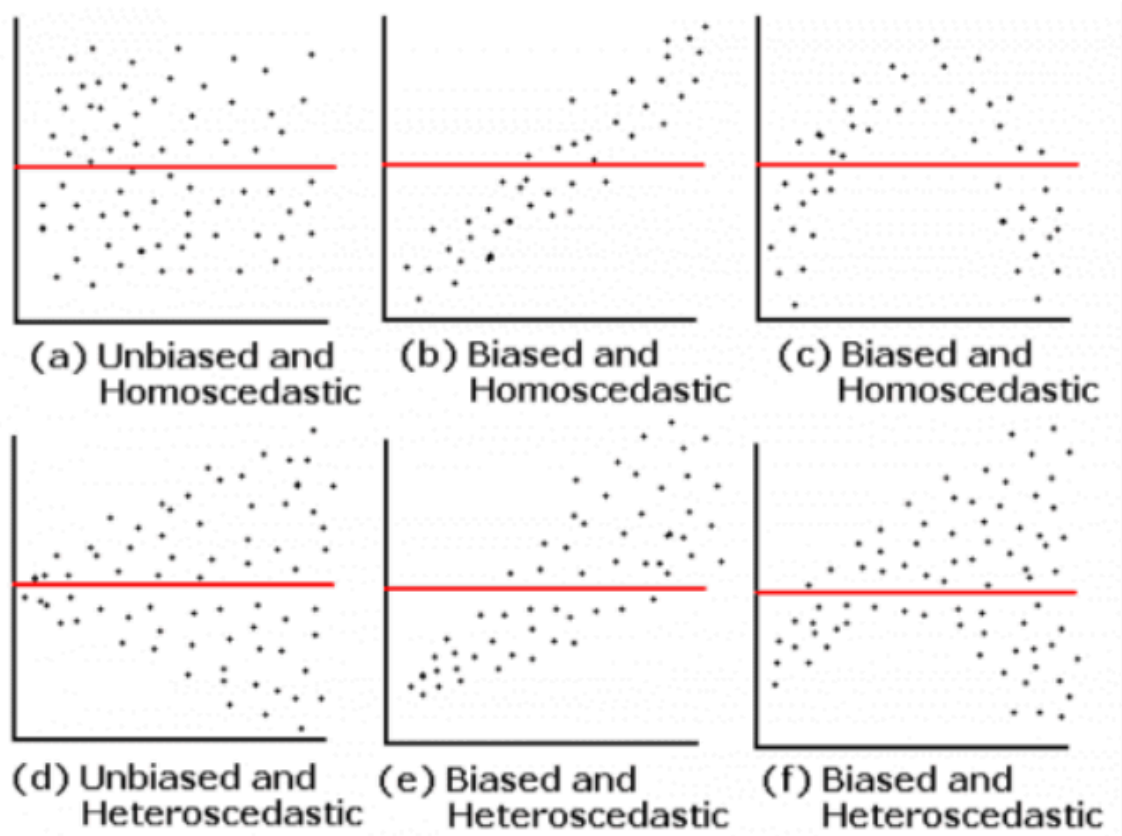
regularización L1 o L2 pueden ayudar a mitigar el impacto de las variables irrelevantes.

Patrones que puedes encontrar

- **Tendencias.** Cuando los residuos se separan del cero de manera sistemática (no aleatoria), tanto si aumentan como si disminuyen para valores de predicciones mayores, el patrón nos sugiere que la función de regresión no es lineal. A este patrón se le suele llamar tendencia, sesgo o "bias" en inglés.
- **Dispersión irregular.** Si observas un patrón de dispersión no aleatorio de los residuos, la variabilidad de los residuos es mayor para ciertos valores predichos por el modelo, esto indica que no se cumple el supuesto de varianza constante en los errores del modelo. Puedes observar alguno de los siguientes casos:
 - un patrón de "abanico". Es decir, los residuos son cercanos a 0 para valores de x pequeños y están más extendidos para valores de x grandes.
 - un patrón de "canalización". Es decir, los residuos se separan para valores de x pequeños pero se acercan a 0 para valores de x grandes.
 - un patrón más complejo.
- **Datos extremos (Outliers).** Ocurre cuando uno o más residuos se apartan del patrón aleatorio del resto. Incluso, podemos observar que si eliminamos el/los outlier el patrón de los residuos cambia.

```
In [6]: # get the image
display.Image("./images/patrones.png")
```

Out[6]: **Ejemplos de patrones:**



- Un modelo válido implica encontrar un patrón de residuos al azar, es decir, que no haya sesgos en los residuos (tendencias) ni una dispersión (varianza) no constante ni valores que desvíen el comportamiento observado (outliers); esto ocurre solamente en la figura "a".
- Las figuras "b" y "c" tienen problemas de tendencia, lo cual podría indicar que la relación entre las variables estudiadas no es la indicada o que existe correlación en los residuales (e.g. si se trata de una serie temporal).
- Las figuras "d", "e" y "f" tienen problemas de dispersión irregular. En todos los casos la varianza de los residuos aumenta con los valores ajustados, esto indica que la variabilidad de los errores aumenta al aumentar su media.

Entonces, ¿cómo puedes mejorar el modelo?

- Si encuentras problemas de tendencia podrías necesitar términos de mayor orden (cuadrática o cúbica) o nuevas variables explicativas, o incluir términos de interacción entre las variables explicativas. Agrega los términos y reajusta el modelo.
- Si encuentras problemas de dispersión irregular utiliza pruebas de igualdad de varianza (complementarias a los análisis gráficos), considera utilizar transformaciones de las variables o modelar la heterogeneidad encontrada con modelos generalizados (GLM) o modelos mixtos (MM).
- Si encuentras posibles valores extremos (outliers) o puntos de influencia verifica que no sean errores de medición y considera realizar análisis robustos.

Creado por:

Isabel Maniega