Creado por:

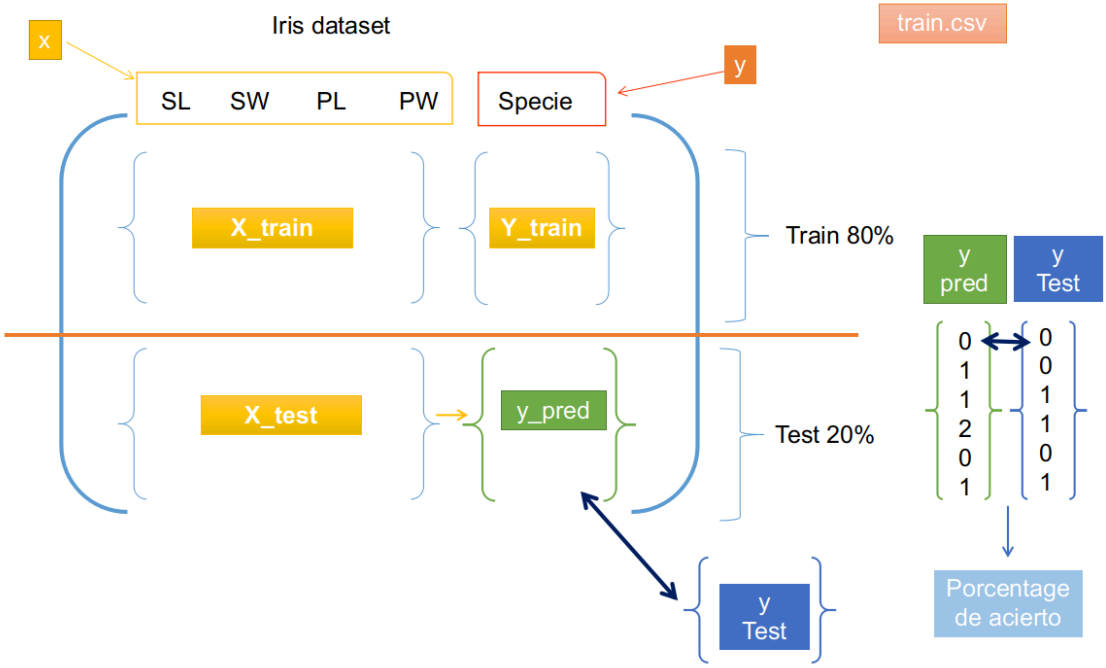
Isabel Maniega

Fundamentos de Aprendizaje Automático

Explicar el concepto de dividir datos en conjuntos de entrenamiento y prueba, particularmente para proyectos de aprendizaje automático, enfatizando la importancia de este paso para la validación del modelo.

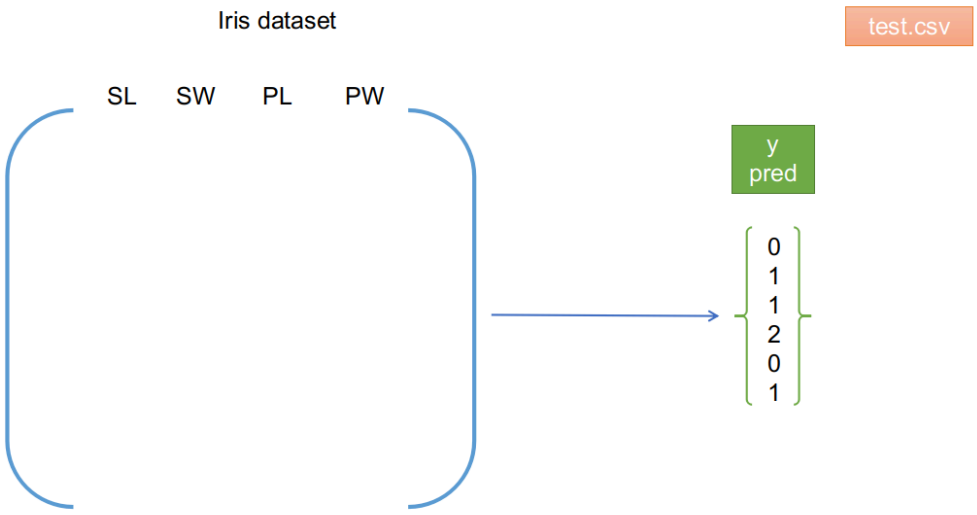
In [1]:
```
from IPython import display

display.Image("./images/train.png")
```

Out[1]:



In [2]:
```
display.Image("./images/test.png")
```

Out[2]:



En la ciencia de datos o el aprendizaje automático, la división de datos entra en escena cuando los datos dados se dividen en dos o más subconjuntos para que un modelo pueda entrenarse, probarse y evaluarse.

En la práctica o en proyectos de la vida real, la división de datos es un aspecto importante y se vuelve imprescindible cuando los modelos se basan en los datos, ya que garantiza la creación de modelos de aprendizaje automático. Por lo general, creamos dos o tres partes del conjunto de datos principal.

Si hay dos divisiones, significa que una se utilizará para el entrenamiento y otra para las pruebas, o si hay tres divisiones, significa que hay conjuntos de entrenamiento, prueba y validación.

Al realizar tareas de aprendizaje automático supervisado, siempre se recomienda dividir los datos en tres conjuntos: conjunto de entrenamiento, conjunto de prueba y conjunto de validación. Entonces, en el procedimiento cuando se trata de la división de datos, primero, dividimos aleatoriamente los datos en tres conjuntos:

- **Conjunto de entrenamiento:** Un subconjunto del conjunto de datos principal se incorporará al modelo para que este pueda aprender los patrones de datos.
- **Conjunto de validación:** Este conjunto se utiliza para comprender el rendimiento del modelo en comparación con otros modelos y opciones de hiperparámetros.
- **Conjunto de prueba:** Este conjunto verifica la precisión del modelo final.

Datos de entrenamiento

Un subconjunto de datos es responsable de entrenar el modelo. Por lo general, el modelo de aprendizaje automático aprende a predecir al comprender los patrones y las relaciones ocultas dentro de los datos. El modelo aprenderá de los patrones y las relaciones entre las variables de peso y tono en nuestro ejemplo.

Al tomar datos de entrenamiento de todos los datos, se debe tener en cuenta una mayor representatividad de los datos. Esto significa que los datos extraídos deben tener suficiente población para cada clase de datos. Con esta calidad, también se debe garantizar que los datos extraídos sean imparciales porque los datos sesgados pueden generar un modelo inexacto.

El ejemplo anterior representa un problema de clasificación de datos en las clases masculina y femenina como una tarea de clasificación binaria. Para resolver este problema, podemos utilizar un modelo de árbol de decisiones simple.

Un árbol de decisiones aprenderá dividiendo los datos en nodos, utilizando la característica seleccionada (Ninguno, Peso, Tono de voz o Peso y Tono de voz).

Datos de validación

Al crear un modelo de aprendizaje automático, generalmente intentamos entrenar más de un modelo cambiando los parámetros del modelo o utilizando diferentes algoritmos. Por ejemplo, al crear el modelo de árbol de decisiones para nuestros datos, realizamos un ajuste de hiperparámetros y descubrimos que varios modelos funcionaban bien en esas condiciones. Por lo tanto, debemos elegir un modelo final utilizando diferentes parámetros.

Se ha visto que si usamos los mismos datos para el entrenamiento y el ajuste de un modelo, tr representa un exceso de aptitud y se vuelve incapaz de generalizarse.

Aquí, el conjunto de validación de los datos entra en escena y funciona como datos independientes e imparciales, lo que también ayuda en la comparación del rendimiento de diferentes modelos.

Como estos datos ayudan a elegir el mejor algoritmo o parámetro del modelo, llevamos el modelo a producción después de aproximar el rendimiento del modelo. Se sugiere no utilizar los datos de prueba para evaluar el modelo antes de seleccionar el óptimo.

Datos de prueba

Como se discutió en el tema anterior, después de entrenar, validar y seleccionar un modelo, debemos llevarlo a producción después de probar su rendimiento para este subconjunto extraído de datos que se denomina datos de prueba.

Debemos ser muy cuidadosos con este paso porque si se realiza antes de tiempo puede generar un sobreajuste y conducir a un rendimiento poco confiable. El conjunto de prueba debe usarse como la forma final de evaluación cuando se completa el uso del conjunto de validación y se selecciona el modelo final.

Palabras finales

Hemos discutido la división de datos en el aprendizaje automático utilizando los puntos "¿Qué es la división de datos?", "¿Cómo funciona?" y "¿Qué son los conjuntos de entrenamiento, prueba y validación?". Podemos obtener las siguientes conclusiones:

- La *división de datos* se convierte en un paso necesario a seguir en el modelado de aprendizaje automático porque ayuda desde el entrenamiento hasta la evaluación del modelo.
- Deberíamos dividir todo nuestro conjunto de datos en tres subconjuntos de datos.
- La cantidad de datos de entrenamiento debería ser mayor que la de los otros dos datos. Además, debería ser imparcial respecto de cualquier clase o categoría, de modo que el modelo pueda aprender adecuadamente de los datos.
- Deberíamos utilizar el conjunto de validación para evaluar varios modelos y encontrar el modelo con mejor rendimiento.
- Después de encontrar el modelo con mejor rendimiento, utilizamos el conjunto de prueba para cuantificar el rendimiento del modelo.
- La división de datos es un subpaso simple en el modelado de aprendizaje automático o el modelado de datos, mediante el cual podemos tener una comprensión realista del rendimiento del modelo. Además, ayuda al modelo a generalizarse bien a datos desconocidos o no vistos.

Estadística descriptiva

Comprender y aplicar medidas estadísticas en el análisis de datos. Aplicar las estadísticas descriptivas de Python para el análisis de conjuntos de datos.

Las medidas que indican el centro aproximado de una distribución se denominan **medidas de tendencia central**.

Las medidas que describen la dispersión de los datos son **medidas de dispersión**. Estas medidas incluyen la media, la mediana, la moda, el rango, los cuartiles superior e inferior, la varianza y la desviación estándar.

In [3]:
```
# pip install pandas
```

In [4]:
```
# pip install scipy
```

In [5]:
```
# pip install statsmodels
```

In [6]:
```
# pip install seaborn
```

In [7]:
```
import pandas as pd
import numpy as np

# Preprocesado y análisis
# ==============================================================================
import statsmodels.api as sm
from scipy import stats
```

```
# Gráficos
# ================================================================
import matplotlib.pyplot as plt
import seaborn as sns
from IPython import display
```

Ejemplo 1: Notas

In [8]:
```
df = pd.DataFrame({"notas_1": [15, 16, 15, 17, 14, 14, 14, 10, 15, 25],
                   "notas_2": [16, 21, 16, 16, 13, 15, 15, 19, 22, 15],
                   "notas_3": [17, 22, 15, 22, 14, 15, 16, 15, 24, 16]})
df.head()
```

Out[8]:

	notas_1	notas_2	notas_3
0	15	16	17
1	16	21	22
2	15	16	15
3	17	16	22
4	14	13	14

Tendencia Central

Media: La media de un conjunto de datos es la suma de todos los valores de un conjunto de datos dividida por el número de valores del conjunto.

media_1 = (15 + 16 + 15 + 17 + 14) / 5

In [9]:
```
media_1 = sum(df.notas_1)/ len(df.notas_1)
media_1
```

Out[9]: 15.5

Como calcular la media de las distintas notas:

In [10]:
```
media_1 = df["notas_1"].mean()
media_1
```

Out[10]: np.float64(15.5)

In [11]:
```
media_2 = df["notas_2"].mean()
media_2
```

Out[11]: np.float64(16.8)

In [12]:
```
media_3 = df["notas_3"].mean()
media_3
```

Out[12]: np.float64(17.6)

Mediana: La mediana de un conjunto de datos es el "elemento medio" cuando los datos están organizados en orden ascendente.

mediana_1 = (10, 14, 14, 14, **15**, **15**, 15, 16, 17, 25)

mediana_1 = 15

In [13]:
```
mediana_1 = df.notas_1.sort_values(ascending=True)
mediana_1
```

Out[13]:
```
7    10
6    14
4    14
5    14
2    15
0    15
8    15
1    16
3    17
9    25
Name: notas_1, dtype: int64
```

Como calcular la mediana de las distintas notas:

In [14]:
```
mediana_1 = df["notas_1"].median()
mediana_1
```

Out[14]:
```
np.float64(15.0)
```

In [15]:
```
mediana_2 = df["notas_2"].median()
mediana_2
```

Out[15]:
```
np.float64(16.0)
```

In [16]:
```
mediana_3 = df["notas_3"].median()
mediana_3
```

Out[16]:
```
np.float64(16.0)
```

Moda: La moda es la medida que aparece con mayor frecuencia en un conjunto de datos.

moda_1 = (10, **14, 14, 14, 15, 15, 15,** 16, 17, 25)

moda_1 = (14, 15,)

Como calcular la moda de las distintas notas:

In [17]:
```
moda_1 = df["notas_1"].mode()
moda_1
```

Out[17]:
```
0    14
1    15
Name: notas_1, dtype: int64
```

In [18]:
```
moda_2 = df["notas_2"].mode()
moda_2
```

Out[18]:
```
0    15
1    16
Name: notas_2, dtype: int64
```

In [19]:
```
moda_3 = df["notas_3"].mode()
moda_3
```

Out[19]:
```
0    15
Name: notas_3, dtype: int64
```

In [20]:
```
df.notas_3.value_counts()
```

Out[20]:
```
notas_3
15    3
22    2
16    2
17    1
14    1
24    1
Name: count, dtype: int64
```

Resultados Nota_1:

In [21]:
```
print(f"Media: {media_1}, Mediana: {mediana_1}, Moda: \n{moda_1}")
```

```
Media: 15.5, Mediana: 15.0, Moda:
0    14
1    15
Name: notas_1, dtype: int64
```

Resultados Nota_2:

In [22]:
```
print(f"Media: {media_2}, Mediana: {mediana_2}, Moda: \n{moda_2}")
```

```
Media: 16.8, Mediana: 16.0, Moda:
0    15
1    16
Name: notas_2, dtype: int64
```

Resultados Nota_2:

In [23]:
```
print(f"Media: {media_3}, Mediana: {mediana_3}, Moda: \n{moda_3}")
```

```
Media: 17.6, Mediana: 16.0, Moda:
0    15
Name: notas_3, dtype: int64
```

Variabilidad

Varianza

La varianza mide el grado de dispersión de un conjunto de valores. Cuantifica la distancia que separa cada punto de datos del conjunto de la media.

Variance = (∑(Each value — Mean)^2)/ Number of values

Variance = (suma((15 -15.5)^2, (16-15.5)^2, (15-15.5)^2, (17-15.5)^2, (14-15.5)^2, (14-15.5)^2, (14-15.5)^2, (10-15.5)^2, (15-15.5)^2, (25-15.5)^2)/10)

In [24]:
```
Variance = ((15 -15.5)**2 + (16-15.5)**2 + (15-15.5)**2 + (17-15.5)**2 +
Variance
```

Out[24]: 13.05

In [25]:
```
varianza_1 = df["notas_1"].var(ddof=0)
varianza_1
```

Out[25]: np.float64(13.05)

In [26]:
```
varianza_2 = df["notas_2"].var(ddof=0)
varianza_2
```

Out[26]: np.float64(7.56)

In [27]:
```
varianza_3 = df["notas_3"].var(ddof=0)
varianza_3
```

Out[27]: np.float64(11.84)

La varianza nos ayuda a comprender en qué medida los valores individuales se desvían de la media. Una varianza más alta indica un conjunto de datos más disperso.

Desviación estándar

La desviación estándar es la raíz cuadrada de la varianza. Proporciona una medida más interpretable de la dispersión de los datos.

In [28]:
```
import math

STD = math.sqrt(((15 -15.5)**2 + (16-15.5)**2 + (15-15.5)**2 + (17-15.5)*
STD
```

Out[28]: 3.6124783736376886

Calculamos la desviación estandar de los datos:

Delta Degrees of Freedom --> ddof

https://stackoverflow.com/questions/41204400/what-is-the-difference-between-numpy-var-and-statistics-variance-in-python

In [29]:
```
std_1 = df["notas_1"].std(ddof=0)
std_1
```

Out[29]: np.float64(3.6124783736376886)

In [30]:
```
std_2 = df["notas_2"].std(ddof=0)
std_2
```

Out[30]: np.float64(2.749545416973504)

In [31]:
```
std_3 = df["notas_3"].std(ddof=0)
std_3
```

Out[31]: np.float64(3.4409301068170506)

La desviación estándar es una métrica muy utilizada en estadística. Proporciona una medida de la distancia típica entre cada punto de datos y la media, lo que permite comprender mejor la variabilidad del conjunto de datos.

Rango intercuartílico o RIQ

El rango es la diferencia entre los valores más altos y más bajos de un conjunto de datos. Para determinar el rango:

1. Identifique el valor más alto de su conjunto de datos. Esto se llama máximo. 10, 14, 14, 14, 15, 15, 15, 16, 17, **25**
2. Identifique el valor más bajo de su conjunto de datos. Esto se llama mínimo. **10**, 14, 14, 14, 15, 15, 15, 16, 17, 25
3. Reste el mínimo del máximo. 25-10 = **15**

Como calcular la IQR de las distintas notas:

In [32]:
```
rango_1 = df["notas_1"].max() - df["notas_1"].min()
iqr_1 = df["notas_1"].quantile(0.75) - df["notas_1"].quantile(0.25)
print(f"IQR de las notas 1: {iqr_1}, rango: {rango_1}")
```

```
IQR de las notas 1: 1.75, rango: 15
```

In [33]:
```
rango_2 = df["notas_2"].max() - df["notas_2"].min()
iqr_2 = df["notas_2"].quantile(0.75) - df["notas_2"].quantile(0.25)
print(f"IQR de las notas 2: {iqr_2}, rango: {rango_2}")
```

```
IQR de las notas 2: 3.25, rango: 9
```

In [34]:
```
rango_3 = df["notas_3"].max() - df["notas_3"].min()
iqr_3 = df["notas_3"].quantile(0.75) - df["notas_3"].quantile(0.25)
print(f"IQR de las notas 3: {iqr_3}, rango: {rango_3}")
```

```
IQR de las notas 3: 5.75, rango: 10
```

Encontrar valores atípicos

- **Notas 1:**

--> Superiores:

In [35]:
```
superiores_1 = df["notas_1"].quantile(0.75) + 1.5 * iqr_1
print(superiores_1)
```

```
18.375
```

In [36]:
```
df.notas_1
```

Out[36]:
```
0    15
1    16
2    15
3    17
4    14
5    14
6    14
7    10
8    15
9    25
Name: notas_1, dtype: int64
```

Todos los valores superiores a 18.375 son outliers, en nuestro caso es el valor 25.

--> Inferiores

In [37]:
```
inferiores_1 = df["notas_1"].quantile(0.25) - 1.5 * iqr_1
inferiores_1
```

Out[37]:
```
np.float64(11.375)
```

Todos los valores inferiores a 11.375 son considerados outliers, en este caso el valor 10.

- **Notas 2:**

--> Superiores:

In [38]:
```
superiores_2 = df["notas_2"].quantile(0.75) + 1.5 * iqr_2
print(superiores_2)
```

```
23.125
```

In [39]:
```
df.notas_2
```

Out[39]:
```
0    16
1    21
2    16
3    16
4    13
5    15
6    15
7    19
8    22
9    15
Name: notas_2, dtype: int64
```

--> Inferiores

In [40]:
```
inferiores_2 = df["notas_2"].quantile(0.25) - 1.5 * iqr_2
inferiores_2
```

Out[40]: np.float64(10.125)

- **Notas 3:**

--> Superiores:

In [41]:
```
superiores_3 = df["notas_3"].quantile(0.75) + 1.5 * iqr_3
print(superiores_3)
```

```
29.375
```

In [42]:
```
df.notas_3
```

Out[42]:
```
0    17
1    22
2    15
3    22
4    14
5    15
6    16
7    15
8    24
9    16
Name: notas_3, dtype: int64
```

--> Inferiores

In [43]:
```
inferiores_3 = df["notas_3"].quantile(0.25) - 1.5 * iqr_3
inferiores_3
```

Out[43]: np.float64(6.375)

Máximos, Mínimos, cuartiles (Q3, Q1), Mediana/Media (Q2)

Los **cuartiles** de un grupo de datos son las medianas de las mitades superior e inferior de ese conjunto. El cuartil inferior, Q1, es la mediana de la mitad inferior, mientras que el cuartil superior, Q3, es la mediana de la mitad superior. Si su conjunto de datos tiene una cantidad impar de puntos de datos, no tendrá en cuenta la mediana al encontrar estos valores, pero si su conjunto de datos contiene una cantidad par de puntos de datos, tendrá en cuenta ambos valores medios que utilizó para encontrar la mediana como partes de las mitades superior e inferior.

Ordene los datos de menor a mayor. 10, 14, 14, 14, 15, 15, 15, 16, 17, 25

Q = a/4 * N donde:

- a es el cuartil 1, 2, 3
- N es el numero de datos

Q = 0.25 * 10 = 2.5 tomamos los valores entre la posición 2, 3 => 14, 14

con esos datos Q1 = valor 2 + 0.25 (valor 3 - valor 2);

Q1 = 14 + 0.25(14 -14) = 14

Q = 0.75 * 10 = 7.5 buscamos los datos entre la posición 7, 8 => 15, 16

Q3 = 15 + 0.75 (16-15) = 15.75

- **Notas 1:**

In [44]:
```
max_1 = df["notas_1"].max()
min_1 = df["notas_1"].min()
q3_1 = df["notas_1"].quantile(0.75)
q1_1 = df["notas_1"].quantile(0.25)
mediana_1 = df["notas_1"].median()
media_1 = df["notas_1"].mean()
```

In [45]:
```
print(f"Maximo: {max_1}, Mínimo: {min_1}, Q3: {q3_1}, Q1: {q1_1}, Media:
```

```
Maximo: 25, Mínimo: 10, Q3: 15.75, Q1: 14.0, Media: 15.5
```

- **Notas 2:**

In [46]:
```
max_2 = df["notas_2"].max()
min_2 = df["notas_2"].min()
q3_2 = df["notas_2"].quantile(0.75)
q1_2 = df["notas_2"].quantile(0.25)
mediana_2 = df["notas_2"].median()
media_2 = df["notas_2"].mean()
```

In [47]:
```
print(f"Maximo: {max_2}, Mínimo: {min_2}, Q3: {q3_2}, Q1: {q1_2}, Media:
```

```
Maximo: 22, Mínimo: 13, Q3: 18.25, Q1: 15.0, Media: 16.8
```

- **Notas 3:**

In [48]:
```
max_3 = df["notas_3"].max()
min_3 = df["notas_3"].min()
q3_3 = df["notas_3"].quantile(0.75)
q1_3 = df["notas_3"].quantile(0.25)
mediana_3 = df["notas_3"].median()
media_3 = df["notas_3"].mean()
```

In [49]:
```
print(f"Maximo: {max_3}, Mínimo: {min_3}, Q3: {q3_3}, Q1: {q1_3}, Media:
```

```
Maximo: 24, Mínimo: 14, Q3: 20.75, Q1: 15.0, Media: 17.6
```

In [50]:
```
df.describe()
```

Out[50]:

	notas_1	notas_2	notas_3
count	10.000000	10.000000	10.000000
mean	15.500000	16.800000	17.600000
std	3.807887	2.898275	3.627059
min	10.000000	13.000000	14.000000
25%	14.000000	15.000000	15.000000
50%	15.000000	16.000000	16.000000
75%	15.750000	18.250000	20.750000
max	25.000000	22.000000	24.000000

Resultados notas 1:

In [51]:
```
print(f'Desviación estándar: {std_1}, Varianza: {varianza_1}, Rango: {ran
 Outlier Sup: {superiores_1}, Outlier Inf: {inferiores_1}')
```

```
Desviación estándar: 3.6124783736376886, Varianza: 13.05, Rango: 15,
IQR: 1.75 Outlier Sup: 18.375, Outlier Inf: 11.375
```

Resultados notas 2:

In [52]:
```
print(f'Desviación estándar: {std_2}, Varianza: {varianza_2}, Rango: {ran
 Outlier Sup: {superiores_2}, Outlier Inf: {inferiores_2}')
```

```
Desviación estándar: 2.749545416973504, Varianza: 7.56, Rango: 9,
IQR: 3.25 Outlier Sup: 23.125, Outlier Inf: 10.125
```

Resultados notas 3:

In [53]:
```
print(f'Desviación estándar: {std_3}, Varianza: {varianza_3}, Rango: {ran
 Outlier Sup: {superiores_3}, Outlier Inf: {inferiores_3}')
```

```
Desviación estándar: 3.4409301068170506, Varianza: 11.84, Rango: 10,
IQR: 5.75 Outlier Sup: 29.375, Outlier Inf: 6.375
```

Aplicar medidas de confianza en cálculos estadísticos para evaluar la confiabilidad de los datos.

¿Qué es un intervalo de confianza?

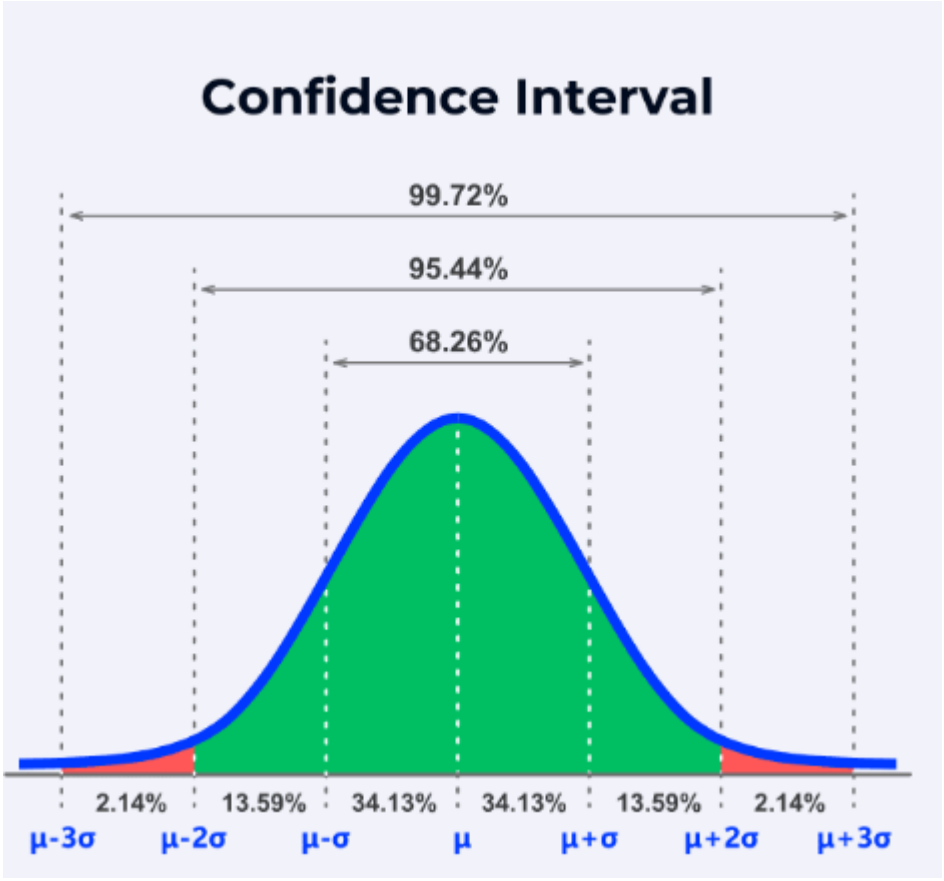
Un intervalo de confianza es una herramienta estadística que se utiliza para estimar el rango de valores dentro del cual es probable que se encuentre un parámetro de población, como una media o proporción de población. Proporciona una medida de incertidumbre en torno a una estimación puntual derivada de datos de muestra.

Los intervalos de confianza se construyen en función de estadísticas de muestra, como la media o proporción de muestra, y suelen ir acompañados de un nivel de confianza específico, como el 95 % o el 99 %. El nivel de confianza indica la probabilidad de que el

intervalo calculado contenga el parámetro poblacional verdadero en un muestreo repetido.

In [54]:
```
display.Image("./images/image_6.png")
```

Out[54]:



Importancia del intervalo de confianza en el análisis estadístico

- Cuantificación de la incertidumbre: los intervalos de confianza proporcionan una medida de la incertidumbre en torno a las estimaciones puntuales, lo que permite a los investigadores evaluar la fiabilidad y precisión de sus hallazgos. Los intervalos de confianza ayudan a evitar el exceso de confianza en las estimaciones estadísticas al reconocer y cuantificar la incertidumbre.
- Inferencia y toma de decisiones: los intervalos de confianza desempeñan un papel crucial en la inferencia estadística y la toma de decisiones. Permiten a los investigadores realizar inferencias sobre los parámetros poblacionales basándose en datos de muestra, lo que orienta las decisiones en investigación, negocios, atención médica y formulación de políticas.
- Comparación de grupos o tratamientos: los intervalos de confianza facilitan las comparaciones entre grupos o tratamientos al proporcionar un rango de valores plausibles para los parámetros poblacionales. Ya sea que se comparen medias, proporciones u otras estadísticas, los intervalos de confianza ayudan a evaluar la importancia y la magnitud de las diferencias.
- Determinación del tamaño de la muestra: los intervalos de confianza informan sobre la determinación del tamaño de la muestra para los estudios de investigación. Al especificar el nivel deseado de precisión y confianza, los investigadores pueden

calcular el tamaño de la muestra necesario para alcanzar los objetivos de su estudio y, al mismo tiempo, minimizar los costos y los recursos.

- Comunicación de resultados: los intervalos de confianza ofrecen una forma concisa de comunicar la precisión y la incertidumbre de las estimaciones estadísticas a las partes interesadas, incluidos los investigadores, los responsables de las políticas y el público en general. Proporcionan una indicación clara del rango dentro del cual es probable que se encuentre el verdadero parámetro de la población.
- Robustez ante los supuestos: a diferencia de las estimaciones puntuales, que pueden ser sensibles a los valores atípicos o las violaciones de los supuestos distributivos, los intervalos de confianza son más robustos y proporcionan una imagen más completa de la incertidumbre subyacente. Ofrecen un enfoque flexible para el análisis estadístico, en particular en situaciones en las que los supuestos paramétricos pueden no cumplirse.
- Control de calidad y mejora de procesos: en el control de calidad y la mejora de procesos, los intervalos de confianza se utilizan para monitorear y evaluar el desempeño de los sistemas y procesos. Al realizar un seguimiento de los intervalos de confianza a lo largo del tiempo, las organizaciones pueden identificar tendencias, detectar desviaciones del desempeño esperado e implementar acciones correctivas según sea necesario.
- Reproducibilidad científica: los intervalos de confianza contribuyen a la transparencia y reproducibilidad de la investigación científica al cuantificar la incertidumbre inherente a las estimaciones estadísticas. Los estudios de replicación pueden utilizar intervalos de confianza para evaluar la coherencia y la generalización de los hallazgos en diferentes muestras o entornos.
- Toma de decisiones en situaciones de incertidumbre: en contextos de toma de decisiones, los intervalos de confianza proporcionan a los responsables de la toma de decisiones un marco para considerar la incertidumbre y la variabilidad en sus elecciones. Ya sea que se evalúe la eficacia de las intervenciones, se evalúen los riesgos o se asignen recursos, los intervalos de confianza permiten tomar decisiones más fundamentadas y sólidas.

Entendimiento de los intervalos de confianza

Los intervalos de confianza son una piedra angular de la inferencia estadística, ya que nos permiten estimar parámetros de población con un cierto grado de incertidumbre. En esencia, un intervalo de confianza es un rango de valores derivados de datos de muestra que probablemente contengan el parámetro de población real.

Imagina que estás tratando de estimar la altura promedio de todos los adultos de un país. En lugar de basarte únicamente en la altura media de la muestra, que podría variar de una muestra a otra, un intervalo de confianza proporciona un rango de valores plausibles dentro de los cuales se espera que se encuentre la media de la población real. Este rango se expresa con un nivel de confianza específico, normalmente el 95 % o el 99 %.

Interpretación del intervalo de confianza

Interpretar un intervalo de confianza implica comprender qué representa el intervalo y qué no. Es fundamental comprender que el nivel de confianza asociado con un intervalo se refiere al porcentaje de intervalos de confianza, derivados de un muestreo repetido, que contendrían el parámetro de población real. Por ejemplo, si construimos 100 intervalos de confianza con un nivel de confianza del 95 %, esperaríamos que aproximadamente 95 de ellos contuvieran el parámetro de población real.

Al comunicar los resultados de un intervalo de confianza, es fundamental enfatizar que proporciona un rango de valores plausibles, no una estimación puntual específica. Además, el intervalo de confianza solo cuantifica la incertidumbre debido a la variabilidad del muestreo y no tiene en cuenta otras fuentes de incertidumbre o sesgo.

¿Cómo calcular el intervalo de confianza? El cálculo de un intervalo de confianza depende de varios factores, incluido el tamaño de la muestra, la variabilidad de la población y el nivel de confianza deseado. Para datos distribuidos normalmente con una desviación estándar poblacional conocida, la fórmula para calcular un intervalo de confianza para la media poblacional (μ) es:

$$CI = \bar{x} \pm Z(\sigma/\sqrt{n})$$

Donde:

- $\bar{x}$ es la media de la muestra.
- σ es la desviación estándar de la población.
- n es el tamaño de la muestra.
- Z es el valor crítico de la distribución normal estándar correspondiente al nivel de confianza deseado.

Para los casos en los que se desconoce la desviación estándar de la población o el tamaño de la muestra es pequeño, se utiliza la distribución t en lugar de la distribución normal estándar. Este ajuste tiene en cuenta la incertidumbre adicional introducida al estimar la desviación estándar de la población a partir de los datos de la muestra.

Tabla pequeña de valores z para intervalos de confianza

Nivel de confianza	z
0,70	1,04
0,75	1,15
0,80	1,28
0,85	1,44
0,90	1,645
0,92	1,75
0,95	1,96
0,96	2,05
0,98	2,33
0,99	2,58

Ejemplo:

Supongamos que queremos estimar el tiempo promedio que los clientes pasan en una tienda. Recopilamos una muestra de 100 clientes y descubrimos que el tiempo promedio que pasan es de 30 minutos, con una desviación estándar de 5 minutos. Si queremos construir un intervalo de confianza del 95 % para el tiempo promedio que pasa la población, podemos usar la fórmula:

`x̄ = 30`

`σ = 5`

`n = 100`

`Z = 95% --> 1.96`

```
CI = 30 ± 1.96(5/√100)

CI = 30 ± 0.98
```

Por lo tanto, el intervalo de confianza del 95 % para el tiempo medio de permanencia de los clientes en la tienda es de aproximadamente 29,02 a 30,98 minutos.

Implicaciones de los niveles de confianza

La elección del nivel de confianza tiene varias implicaciones:

- Precisión frente a certeza: un nivel de confianza más alto (p. ej., 99 %) da como resultado un intervalo de confianza más amplio, lo que refleja una mayor certeza de que el intervalo contiene el parámetro verdadero pero menos precisión sobre su valor. Por el contrario, un nivel de confianza más bajo (p. ej., 90 %) produce un intervalo más estrecho, lo que ofrece más precisión pero menos certeza.
- Significación estadística: en las pruebas de hipótesis, un intervalo de confianza que no contiene el valor de la hipótesis nula (a menudo cero) indica un resultado estadísticamente significativo en el nivel de confianza elegido. Por ejemplo, un intervalo de confianza del 95 % que no incluye el cero sugiere un efecto estadísticamente significativo con un riesgo del 5 % de un falso positivo (error de tipo I).
- Interpretación: Los niveles de confianza deben interpretarse en el contexto del estudio y del proceso de toma de decisiones. Proporcionan un rango de valores plausibles para el parámetro de interés, pero no garantizan que el valor verdadero se encuentre dentro de un intervalo único calculado a partir de una muestra.
- Falsos positivos: Incluso con un nivel de confianza alto, siempre existe la posibilidad de observar un resultado estadísticamente significativo por pura casualidad. Esto se conoce como falso positivo y el riesgo es igual al 100 % menos el nivel de confianza.

Analizar y evaluar las relaciones de datos.

La correlación es el análisis estadístico de la relación o dependencia entre dos variables. La correlación nos permite estudiar tanto la fuerza como la dirección de la relación entre dos conjuntos de variables.

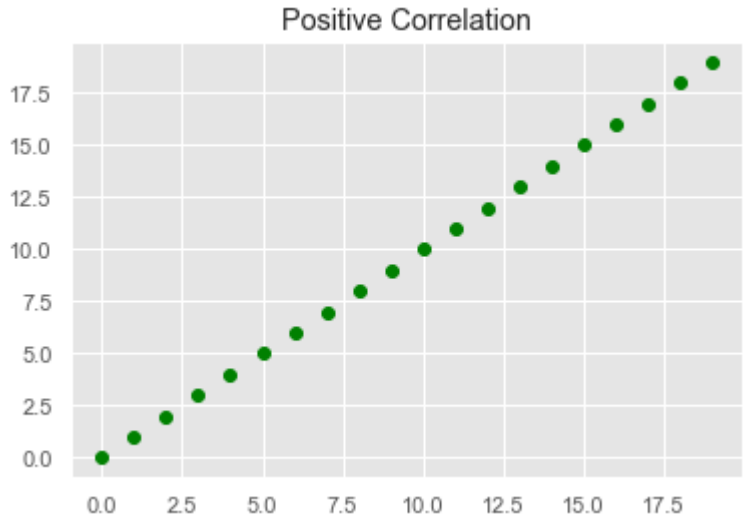
El estudio de la correlación es fundamental en el campo del aprendizaje automático. Por ejemplo, algunos algoritmos no funcionarán correctamente si dos o más variables están estrechamente relacionadas, lo que generalmente se conoce como multicolinealidad . La correlación también es la base del Análisis de Componentes Principales , una técnica de reducción de dimensionalidad lineal que es muy útil en proyectos de aprendizaje automático.

Tipos:

Correlación positiva: se dice que dos variables están correlacionadas positivamente cuando sus valores se mueven en la misma dirección. Por ejemplo, en la imagen a continuación, a medida que aumenta el valor de X, también lo hace el valor de Y a una tasa constante:

In [55]:
```
display.Image("./images/posit.png")
```

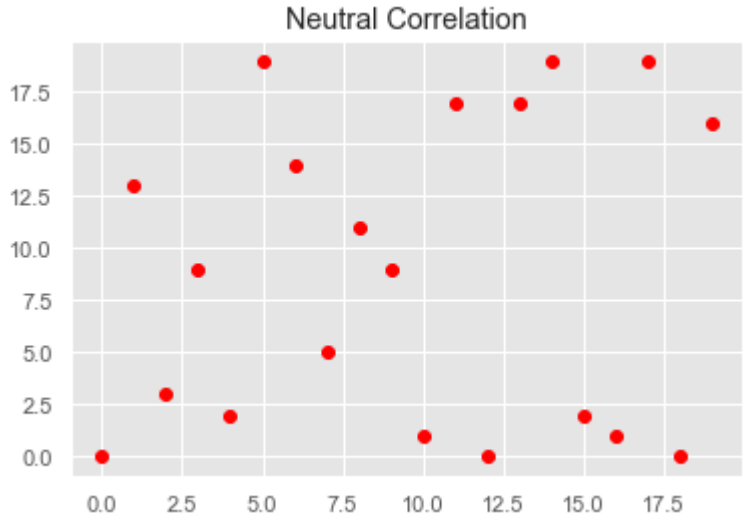
Out[55]:



Correlación Neutral: No hay relación en el cambio de las variables X e Y. En este caso los valores son completamente aleatorios y no muestran ningún signo de correlación, como se muestra en la siguiente imagen:

In [56]:
```
display.Image("./images/neutra.png")
```

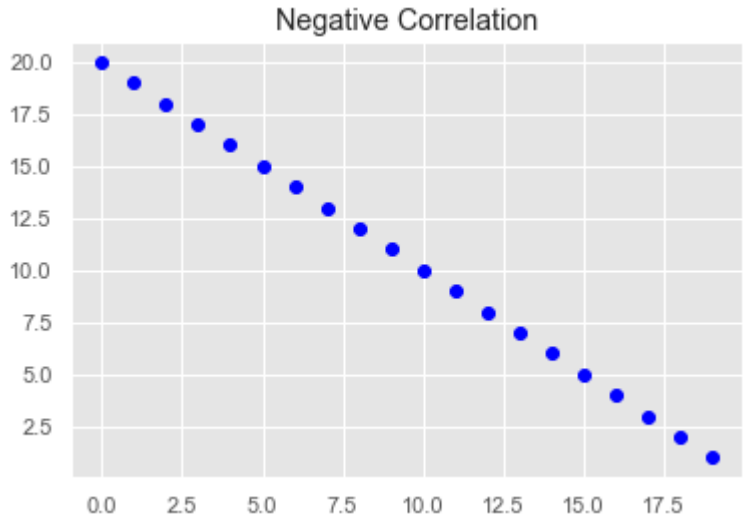
Out[56]:



Correlación negativa: finalmente, las variables X e Y estarán negativamente correlacionadas cuando sus valores cambien en direcciones opuestas, por lo que aquí, a medida que aumenta el valor de X, el valor de Y disminuye a una tasa constante:

In [57]:
```
display.Image("./images/negat.png")
```

Out[57]:



Coeficientes de correlación

Un coeficiente de correlación es un resumen estadístico que mide la fuerza y la dirección con la que se asocian dos variables entre sí.

Una de las ventajas de los coeficientes de correlación es que estiman la correlación entre dos variables de forma estandarizada, por lo que el valor del coeficiente siempre estará en la misma escala, variando de -1,0 a 1,0.

1. Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson (r) es una puntuación que mide la fuerza de una relación lineal entre dos variables. Se calcula dividiendo la covarianza de las variables X e Y por el producto de la desviación estándar de cada variable, como se muestra en la siguiente fórmula:

In [58]:
```
display.Image("./images/pearson.png")
```

Out[58]:

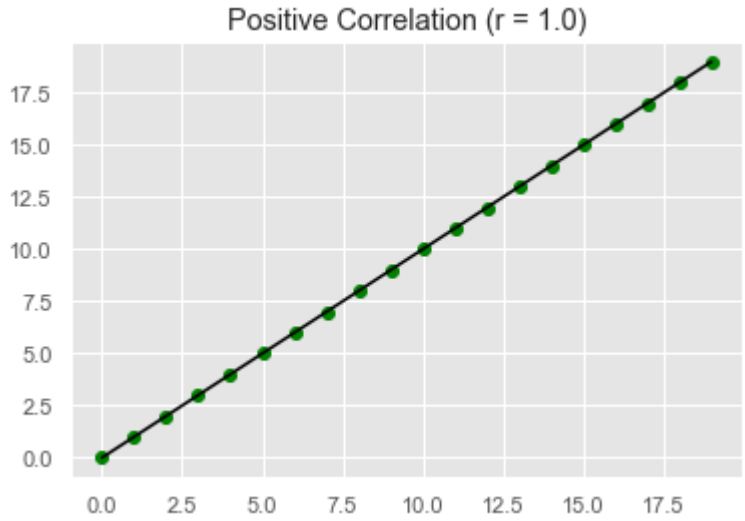
$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

El coeficiente se basa en dos supuestos. Primero, asume que las variables siguen una distribución normal o gaussiana. Si los datos no se distribuyen normalmente, entonces otros coeficientes pueden ser más confiables.

En segundo lugar, supone que existe una relación lineal entre las dos variables, lo que significa que los cambios en los datos se pueden modelar mediante una función lineal (es decir, sus valores aumentan o disminuyen simultáneamente a una tasa constante). Si la relación entre las dos variables es más cercana a una línea recta, entonces su correlación (lineal) es más fuerte y el valor absoluto del coeficiente de correlación de Pearson es más alto. Por ejemplo, en la siguiente imagen, todos los puntos de datos se pueden modelar perfectamente utilizando una línea recta, lo que da como resultado un coeficiente de correlación igual a 1,0.

In [59]:
```
display.Image("./images/r1.png")
```

Out[59]:



Un coeficiente de -1,0 indica una correlación negativa perfecta, mientras que un coeficiente de 1,0 muestra una correlación positiva perfecta. Por el contrario, un coeficiente de 0,0 indica que no existe una correlación lineal entre las variables.

Calculo:

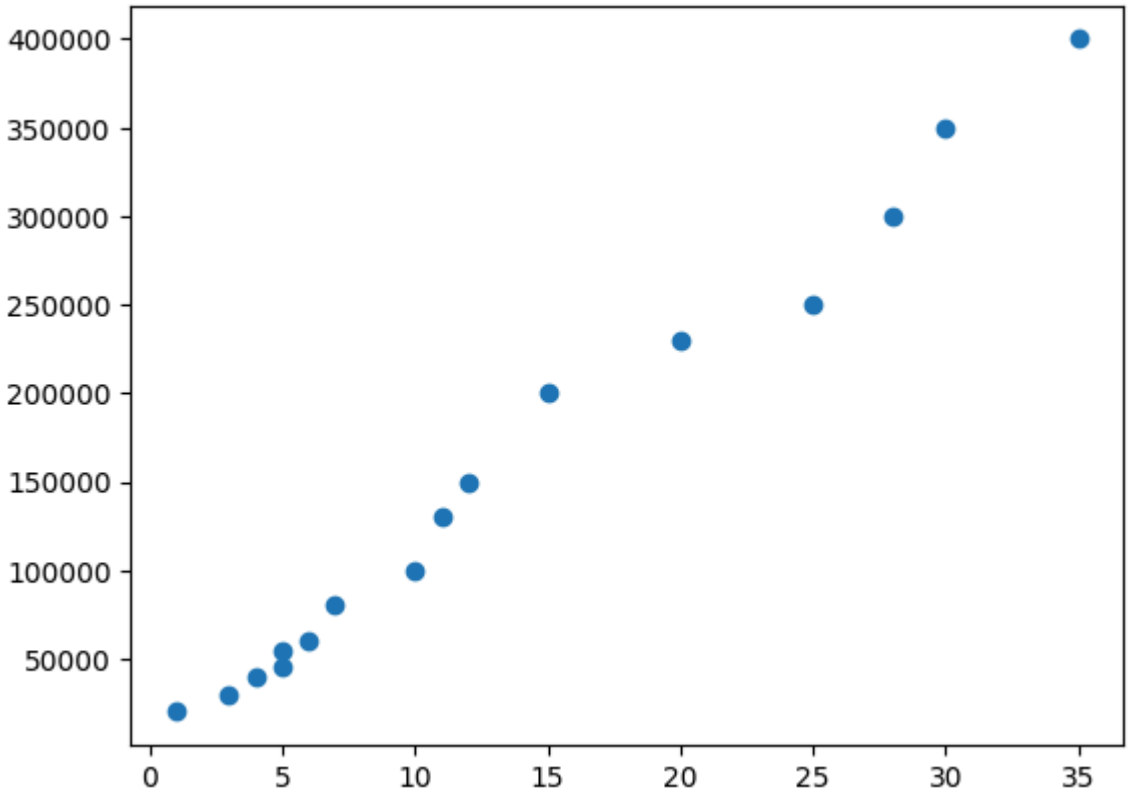
In [60]:
```
experience = [1, 3, 4, 5, 5, 6, 7, 10, 11, 12, 15, 20, 25, 28, 30,35]

salary = [20000, 30000, 40000, 45000, 55000, 60000, 80000, 100000, 130000
```

In [61]:
```
import pandas as pd

df = pd.DataFrame({"Experience": experience, "Salary": salary})
df
```

Out[61]:

	Experience	Salary
0	1	20000
1	3	30000
2	4	40000
3	5	45000
4	5	55000
5	6	60000
6	7	80000
7	10	100000
8	11	130000
9	12	150000
10	15	200000
11	20	230000
12	25	250000
13	28	300000
14	30	350000
15	35	400000

In [62]:
```
import matplotlib.pyplot as plt

plt.scatter(df.Experience, df.Salary)
#plt.plot(df.Experience, df.Salary, color='red', linewidth=2)
plt.show()
```



- scipy librería

In [63]:
```
import scipy.stats as stats

corr, _ = stats.pearsonr (experience, salary)
corr
```

Out[63]: np.float64(0.9929845761480397)

In [64]:
```
# Otros coeficientes:

spearman_corr, _ = stats.spearmanr(experience, salary)
print("spearman:", spearman_corr)

kendall_corr, _ = stats.kendalltau(experience, salary)
print("Kendall:", kendall_corr)
```

```
spearman: 0.9992644353546791
Kendall: 0.9958246164193105
```

- Numpy librería

In [65]:
```
import numpy as np

np.corrcoef(df.Experience, df.Salary)
```

Out[65]:
```
array([[1.        , 0.99298458],
       [0.99298458, 1.        ]])
```

Una matriz de correlación es una tabla que muestra los coeficientes de correlación entre variables. Cada celda de la tabla muestra la correlación entre dos variables. La diagonal de la matriz incluye los coeficientes entre cada variable y ella misma, que siempre es

igual a 1,0. Los demás valores de la matriz representan la correlación entre experiencia y salario. En este caso, como solo estamos calculando correlación para dos variables, los valores son los mismos.

- Pandas librería

In [66]:
```
df['Experience'].corr(df['Salary'])
```

Out[66]:
```
np.float64(0.9929845761480398)
```

In [67]:
```
print(df['Experience'].corr(df['Salary'], method='spearman'))
print(df['Experience'].corr(df['Salary'], method='kendall'))
```

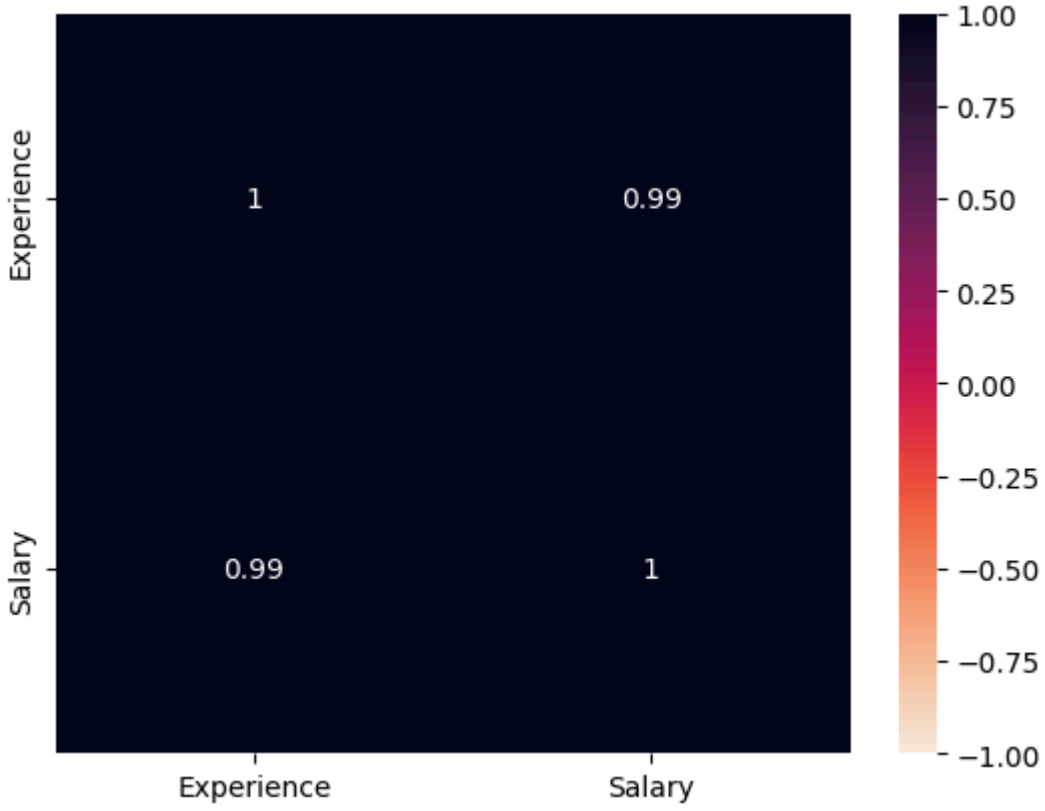
```
0.9992644353546791
0.9958246164193105
```

In [68]:
```
df.corr()
```

Out[68]:

	Experience	Salary
Experience	1.000000	0.992985
Salary	0.992985	1.000000

In [69]:
```
import seaborn as sns

sns.heatmap(df.corr(), vmin=-1, vmax=1,
annot=True,cmap="rocket_r")
plt.show()
```



Conclusión

La correlación solo cuantifica la fuerza y la dirección de la relación entre dos variables. Puede haber una fuerte correlación entre dos variables, pero no nos permite concluir que una causa la otra. Cuando las correlaciones fuertes no son causales, las llamamos correlaciones espurias.

Relación entre correlación y grado de significancia

Una correlación se considera significativa cuando su p-valor es menor que 0.05, lo que indica que la correlación observada es poco probable que se deba al azar. En otras palabras, si el p-valor es menor que 0.05, se rechaza la hipótesis nula de que no hay correlación entre las variables, y se concluye que existe una correlación significativa.

Desarrollemos este tema:

- P-valor (p-value): es una medida de la probabilidad de obtener los resultados observados (o resultados aún más extremos) si la hipótesis nula fuera verdadera. En el contexto de la correlación, la hipótesis nula es que no hay correlación entre las variables.
- Nivel de significancia: generalmente 0.05, es un umbral predefinido para determinar si un resultado es significativo. Si el p-valor es menor o igual que el nivel de significancia, se considera que la correlación es significativa.
- Hipótesis nula: establece que no hay relación entre las variables. Si la hipótesis nula se rechaza, se concluye que hay una relación significativa entre las variables.
- Hipótesis alternativa: establece que sí hay una relación entre las variables. Si la hipótesis nula se rechaza, la hipótesis alternativa se considera válida, lo que indica que hay una correlación significativa entre las variables.

En resumen, para determinar si una correlación es significativa, se compara el p-valor con el nivel de significancia predefinido (generalmente 0.05). Si el p-valor es menor o igual que el nivel de significancia, se concluye que la correlación es significativa.

Estadística Inferencial

In [70]:
```
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[70]:

	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 [71]:
```
X = df['Poblac. estudiantil (en miles)']
y = df['Ventas trimestrales(miles de $)']
```

In [72]:
```
# Plot outputs
plt.scatter(X, y, color="black")

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

plt.show()
```

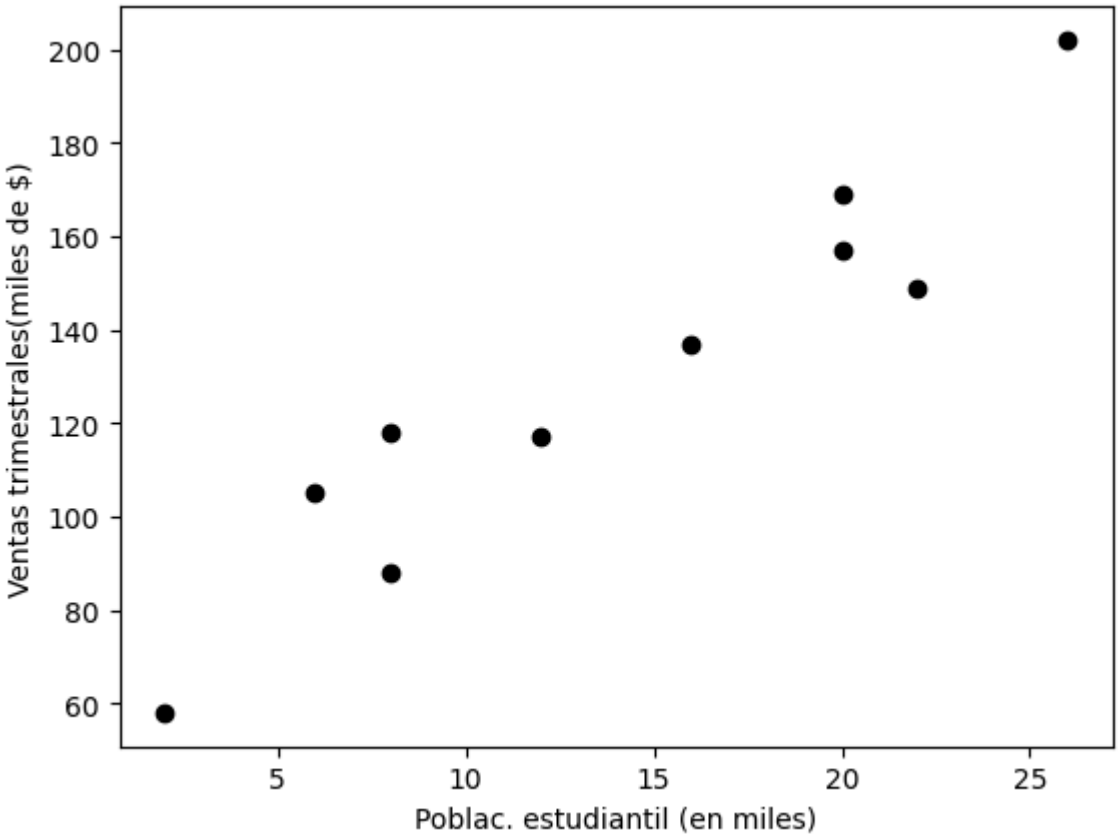


Diagrama de dispersión (y línea de regresión estimada)

$ŷ = b_0 + b_1 x$

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, yi, y los valores estimados de la misma, ŷi. Es decir se minimiza la suma: $Σ(yi – ŷi)^2$.

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_0y b_1:

$$∑y_i = nb_0 + (∑x_i)b_1$$

$$∑x_iy_i = (∑x_i)b_0 + (∑x_i^2)b_1$$

Las soluciones son las siguientes:

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

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

y $b_0 = y - b_1x$

Determine la ecuación de regresión con los datos dados.

In [73]:
```
x_total = X.sum()
x_total # 140
```

Out[73]:
```
np.int64(140)
```

In [74]:
```
y_total = y.sum()
y_total # 1300
```

Out[74]:
```
np.int64(1300)
```

In [75]:
```
df['xiyi'] = X * y
df
```

Out[75]:

	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 [76]:
```
xy_total = df['xiyi'].sum()
xy_total # 21040
```

Out[76]:
```
np.int64(21040)
```

In [77]:
```
df['xi²'] = X ** 2
df
```

Out[77]:

	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 [78]:
```
xi2_total = df['xi²'].sum()
xi2_total # 2528
```

Out[78]:
```
np.int64(2528)
```

In [79]:
```
Sxy = (X - X.mean())*(y - y.mean())
Sxy = Sxy.sum() / len(X) -1
Sxy
```

Out[79]:
```
np.float64(283.0)
```

In [80]:
```
Sx2 = (X - X.mean())**2
Sx2 = Sx2.sum() / len(X) -1
Sx2
```

Out[80]:
```
np.float64(55.8)
```

In [81]:
```
b1 = Sxy / Sx2
b1
```

Out[81]:
```
np.float64(5.07168458781362)
```

In [82]:
```
b0 = y.mean() - (b1 * X.mean())
b0
```

Out[82]:
```
np.float64(58.99641577060932)
```

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

Calculo usando Skelearn:

In [83]:
```
# pip install scikit-learn
```

In [84]:
```
from sklearn import datasets, linear_model
```

In [85]:
```
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[85]:
```
▾ LinearRegression
▶ Parameters
```

In [86]:
```
# 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: $SCE = Σ(y_i – ŷ_i)^2$

Suma de cuadrados total: $SCT = Σ(y_i – y)2$

Suma de cuadrados debida a la regresión: $SCR = Σ(ŷi - y)^2$

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

Coeficiente de determinación : $r² = \frac{SCR}{SCT}= \frac{SCT-SCE}{SCT} = 1-\frac{SCE}{SCT}$

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 [87]:
```
df['y_resultante'] = b0 + b1*x
df
```

Out[87]:

	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 [88]:
```
df['y - y_resultante'] = df['Ventas trimestrales(miles de $)'] - df['y_re
df
```

Out[88]:

	Restaurante	Poblac. estudiantil (en miles)	Ventas trimestrales(miles de $)	xiyi	xi²	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 [89]:
```
SCE = (df['y - y_resultante']**2).sum()
SCE
```

Out[89]:
```
np.float64(1532.9187703138455)
```

In [90]:
```
SCT = ((y - y.mean())**2).sum()
SCT
```

Out[90]:
```
Ventas trimestrales(miles de $)    15730.0
dtype: float64
```

In [91]:
```
SCR = SCT - SCE
SCR
```

Out[91]:
```
Ventas trimestrales(miles de $)    14197.08123
dtype: float64
```

In [92]:
```
coeficiente_determinacion = SCR / SCT
coeficiente_determinacion
```

Out[92]:
```
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 [93]:
```
X = df.iloc[:, 1:2]
y = df.iloc[:, 2:3]
```

In [94]:
```
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 tiene relación lineal. El coeficiente de correlación lineal se relaciona con el coeficiente de determinación así:

$r = (± b_1)\sqrt[2]{r²}$

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: $E(y/x) = β_0 + β_1 x$. Si $β_1=0$ entonces $E(y/x) = β_0$ 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 β1 ≠ 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 $\sigma²$, la varianza de ϵ en el modelo de regresión.

Cuadrados medios del error CME (es una estimación de $\sigma²$)

$S^2 = CME = SCE/(n-2)$

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 [95]:
```
CME = SCE / (len(df)-2)
CME
```

Out[95]:
```
np.float64(191.61484628923068)
```

Error estándar de estimación (s)

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

In [96]:
```
import math

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

Out[96]:
```
13.84250144624268
```

Distribución muestral de b1

b_1 es un estadístico con distribución normal de media $μ_{b1} = β_1$ y desviación estándar $\sigma_{b1} = \frac{\sigma}{\sqrt[2]{Σ(x_i - x)²}}$. Si sustituimos σ por su estimación muestral, s, obtenemos un estimador de σ_{b1} que denotaremos por s_{b1}. $s_{b1·}=\frac{S}{\sqrt[2]{Σ(x_i - x)²}}$. Con esta información podemos construir un estadístico t.

$t = \frac{b_1 - β_1}{S_b1}$ el cual se distribuye con v=n-2 g.l.

In [97]:
```
sb1 = math.sqrt(S / Sx2)
sb1
```

Out[97]:
```
0.4980697768594643
```

In [98]:
```
y.mean()
```

Out[98]:
```
Ventas trimestrales(miles de $)    130.0
dtype: float64
```

In [99]:
```
t = (b1 - y.mean()) /sb1 # ???
t
```

Out[99]:
```
Ventas trimestrales(miles de $)   -250.824927
dtype: float64
```

Prueba t de significación en la regresión

H0: $β_1 = 0$

H1: $β_1 ≠ 0 $

Estadístico de contraste bajo H0, $t_c = \frac{b_1 - 0}{S_{b1}}$

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

In [100…
```
tc = (b1 -0) /sb1
tc
```

Out[100…
```
np.float64(10.182678860365082)
```

In [101…
```
tc > t

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

Out[101…
```
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 $CMR = \frac{SCR}{número de variables independiente} = \frac{SCR}{1}$

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=CMR/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 [102…
```
CME = SCE / len(df) -2
CME
```

Out[102…
```
np.float64(151.29187703138456)
```

In [103…
```
CMR = SCR / 1
CMR
```

Out[103…
```
Ventas trimestrales(miles de $)    14197.08123
dtype: float64
```

In [104…
```
F = CMR/CME
F
```

Out[104…
```
Ventas trimestrales(miles de $)    93.839018
dtype: float64
```

In [105…
```
anova = pd.DataFrame({'Fuente de variación': ['Regresion', 'Error', 'Tota
                      '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[105…

	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: $y = 58.996 + 5.0716x$

In [106…
```
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 [107…
```
# Make predictions using the testing set
y_pred = regr.predict([[10]])
y_pred
```

```
/home/isabelmaniega/Documentos/IA_Python/env/lib/python3.12/site-packages/
sklearn/utils/validation.py:2749: UserWarning: X does not have valid featu
re names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[107… `array([[110.]])`

Coeficientes para regresión logística

In [108…
```
from sklearn import datasets
```

In [109…
```
dataset = datasets.load_breast_cancer()
dataset
```

Out[109…

```
{'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, 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, 0, 1, 1, 1, 1, 1, 0, 1, 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,
        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, 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\n:Number of Instances: 569\n\n:Number of Attribut
es: 30 numeric, predictive attributes and the class\n\n:Attribute 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 dimens
ion ("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\n:Summary 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\n:Mi
ssing Attribute Values: None\n\n:Class Distribution: 212 - Malignant, 35
7 - Benign\n\n:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L.
Mangasarian\n\n:Donor: Nick Street\n\n:Date: 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 Lin
early 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. Man
gasarian. 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 diagnosi
s and\n    prognosis via linear programming. Operations Research, 43(4),
pages 570-577,\n    July-August 1995.\n  - W.H. Wolberg, W.N. Street, an
d O.L. Mangasarian. Machine learning techniques\n    to diagnose breast
cancer from fine-needle aspirates. Cancer Letters 77 (1994)\n    163-17
1.\n',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimete
r', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimensio
n',
        'radius error', 'texture error', 'perimeter error', 'area erro
r',
        '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 par el cual usemos posiblemente un algoritmo de arboles 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, ó 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 ó 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 tambien 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 tambien 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 silo 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 (tambien 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 Especificity)

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 de poder detectar correctamente la enfermedad entre los enfermos.

2. La Especificidad («Especificity»)

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 [110…

```
import pandas as pd

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

Out[110…

	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 [111…
```
X = dataset.data
```

In [112…
```
y = dataset.target
```

In [113…
```
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 [114…
```
from sklearn.preprocessing import StandardScaler
```

In [115…
```
escalar = StandardScaler()
```

In [116…
```
X_train = escalar.fit_transform(X_train)
X_test = escalar.fit_transform(X_test)
```

In [117…
```
from sklearn.linear_model import LogisticRegression

algoritmo = LogisticRegression()
```

In [118…
```
algoritmo.fit(X_train, y_train)
```

Out[118…

▾ LogisticRegression

▶ Parameters

In [119…
```
y_pred = algoritmo.predict(X_test)
y_pred
```

Out[119…
```
array([1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1])
```

In [120…
```
y_test
```

Out[120…
```
array([1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1])
```

In [121…
```
# Matriz de confusión
# [[VP, FP,
# [FN, VN]]

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

In [122…
```
from sklearn.metrics import confusion_matrix
```

In [123…
```
matriz = confusion_matrix(y_test, y_pred)
```

In [124…
```
matriz
```

Out[124…
```
array([[42,  1],
       [ 0, 71]])
```

In [125…
```
"""
    - 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[125… [[42, 1], [0, 71]]

In [126…
```
# 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:
0.9861111111111112
```

In [127…
```
precision = VP / (VP+FP)
precision
```

Out[127… 0.9767441860465116

In [128…
```
# 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 [129…
```
exactitud = (VP+VN) /( VP+FP+FN+VN)
exactitud
```

Out[129… 0.9912280701754386

In [130…
```
# 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 [131…
```
# En el área de la salud decimos que la sensibilidad es la capacidad de d

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

Out[131… 1.0

In [132…
```
# 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
0.993006993006993
```

In [133…
```
F1 = (2 * precision * sensibilidad) / (precision + sensibilidad)
F1
```

Out[133… 0.988235294117647

In [134…
```
# En el area de la salud decimos que la especificidad es  la capacidad de
# entre todos los sanos

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

Out[134… 0.9861111111111112

Creado por:

Isabel Maniega