

Clase 1: Programación en Python

Curso de Python para la Ciencia de Datos
4ta Jornadas de Ingeniería Estadística UBB

Rodrigo Salas Fuentes, Dr. -Ing.

¹Escuela de Ingeniería C. Biomédica. Universidad de Valparaíso

²Centro de Investigación y Desarrollo en Ingeniería en Salud **CINGS-UV**

³Instituto Milenio Intelligent Healthcare Engineering - **iHealth**

rodrigo.salas@uv.cl



Tabla de Contenidos

- 1 Contenido del Curso
- 2 Metodología de Ciencia de Datos
- 3 Análisis de datos con PANDAS
- 4 Visualización de datos con Seaborn
- 5 Análisis de Clustering con Scikit-Learn
 - Definición de Análisis de clusters
 - Medidas de Similaridad entre Objetos
 - Método de K-Medias
 - Métodos Jerárquicos Aglomerativos - AGNES
 - Medidas de Calidad y Determinación del Número de Clusters
- 6 Taller

- 1 Contenido del Curso
- 2 Metodología de Ciencia de Datos
- 3 Análisis de datos con PANDAS
- 4 Visualización de datos con Seaborn
- 5 Análisis de Clustering con Scikit-Learn
- 6 Taller

Contenido del Curso

Contenido	Fechas
Programación con Python Historia de Python Instalación de Python: Anaconda, Google Colab. Variables, Expresiones, Sentencias en Python Estructuras de Control Estructuras de Datos Funciones Taller: "Programación con Python"	19 de enero, 2022
Análisis Exploratorio de datos con Python Toolbox la Ciencia de Datos: PANDAS, Seaborn, Scikit-Learn Manipulación de datos con Python Análisis Exploratorio de los Datos Técnicas de Clustering Taller: "Análisis Exploratorio de datos con Python"	20 de enero, 2022
Análisis Multi-Dimensional de Datos con Python Análisis de Componentes Principales Técnicas de Proyección de Datos Taller: "Análisis Multi-Dimensional de Datos"	21 de enero, 2022

Profesor: Rodrigo Salas Fuentes

- Ingeniero C. Informático. UTFSM (2002)
- Doctor en Ingeniería Informática. UTFSM (2010)
- Profesor Titular, Escuela de Ingeniería C. Biomédica, Universidad de Valparaíso. (since 2007)
- Investigador Principal del Centro de Investigación y Desarrollo en Ingeniería en Salud (CINGS-UV)
- Investigador Principal del Instituto Milenio Intelligent Healthcare Engineering (iHealth).
- Áreas de Investigación: Ciencias de la Computación, Inteligencia Artificial, Ciencia de Datos.



- e-mail: rodrigo.salas@uv.cl
- Google Scholar
- LinkedIn: [rodrigo-salas-fuentes](https://www.linkedin.com/in/rodrigo-salas-fuentes/)
- Twitter: [@rod_salas_f](https://twitter.com/rod_salas_f)
- GitHub: [link](https://github.com/rodrigo-salas-fuentes)
- Web-Page: [link](https://rodrigo-salas-fuentes.github.io)

Material de Estudio

Apuntes del Curso:

El material del curso se encuentra disponible en mi Github personal:

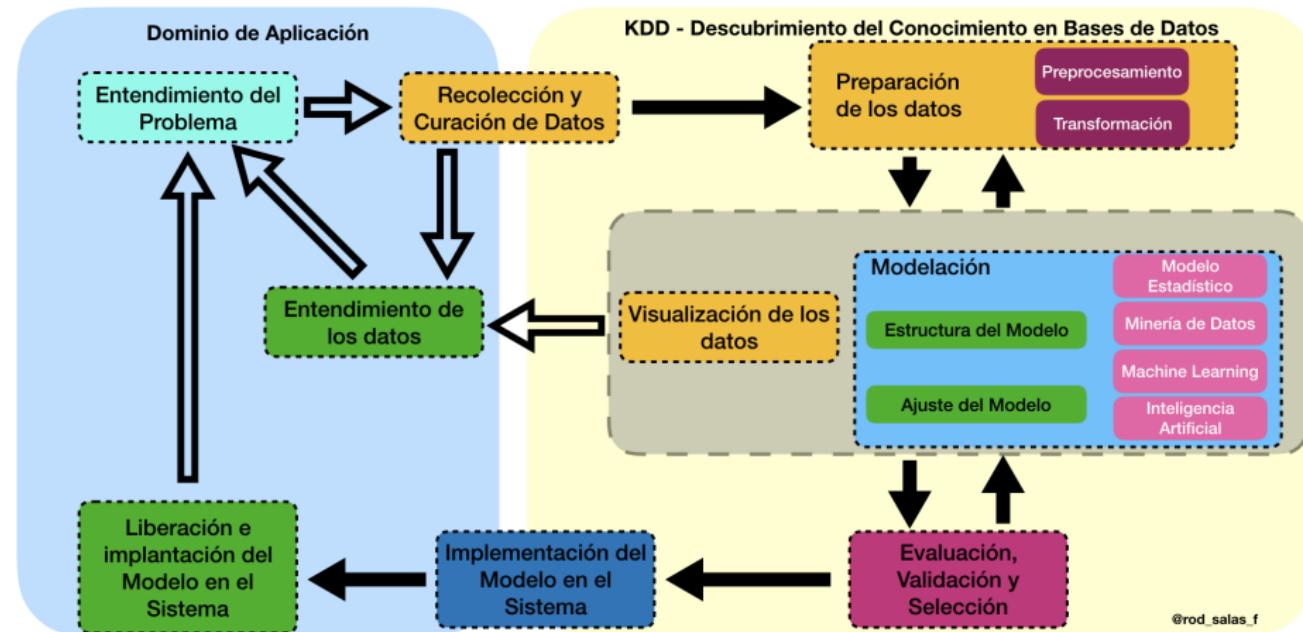


<https://github.com/rodsalasf/Python>

- 1 Contenido del Curso
- 2 Metodología de Ciencia de Datos
- 3 Análisis de datos con PANDAS
- 4 Visualización de datos con Seaborn
- 5 Análisis de Clustering con Scikit-Learn
- 6 Taller

Metodología de Ciencia de Datos

Metodología de Ciencia de Datos



Propuesta de metodología desarrollada por Rodrigo Salas en base a la metodología KDD.

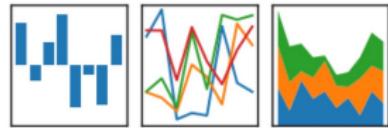
- 1 Contenido del Curso
- 2 Metodología de Ciencia de Datos
- 3 Análisis de datos con PANDAS
- 4 Visualización de datos con Seaborn
- 5 Análisis de Clustering con Scikit-Learn
- 6 Taller

Análisis de datos con PANDAS



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



DataFrame

El DataFrame es la estructura básica de PANDAS.

```
class pandas.DataFrame(data=None, index=None, columns=None,  
dtype=None, copy=None)
```

Parámetros:

- **data:** ndarray (structured or homogeneous), Iterable, dict, or DataFrame.
- **index:** Index or array-like Index to use for resulting frame.
- **columns:** Index or array-like Column labels to use for resulting frame when data does not have them, defaulting to RangeIndex(0, 1, 2, ..., n).
- **dtype:** type, default None Data type to force.
- **copy:** bool or None, default None Copy data from inputs.

Importación de los Toolboxes

Importación de los Toolboxes

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

Creación de un DataFrame

El DataFrame puede ser creado a partir de un diccionario, un arreglo u otra estructura. El usuario puede definir la columna índice.

Ejemplo de creación de un DataFrame

```
d = {'col1': [1, 2], 'col2': [3, 4]}  
df1 = pd.DataFrame(data=d)
```

```
df2 = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]),  
...      columns=['a', 'b', 'c'])
```

```
df3 = pd.DataFrame(data=d, index=[7,8])
```

Accediendo a los datos I

- **at**: Access a single value for a row/column label pair.
- **columns**: The column labels of the DataFrame.
- **iloc**: Purely integer-location based indexing for selection by position.
- **index**: The index (row labels) of the DataFrame.
- **loc**: Access a group of rows and columns by label(s) or a boolean array.
- **shape**: Return a tuple representing the dimensionality of the DataFrame.
- **size**: Return an int representing the number of elements in this object.
- **values**: Return a Numpy representation of the DataFrame.
- **head([n])**: Return the first n rows.
- **tail([n])**: Return the last n rows.

Accediendo a los datos II

Ejemplo de acceso a los valores

```
df = pd.DataFrame('x': [1, 2, 3], 'y': [3, 4, 5], index = [3, 7, 12])
df.at[7,'y']
df.x
df['x']
df.loc[7]
df.iloc[1]
df.values
df.index
df.columns
df.shape
df.size
df.head()
df.tail()
```

Resúmenes y métricas estadísticas I

- `describe([percentiles, include, exclude, ...]):` Generate descriptive statistics.

Resumen Estadístico

```
d = np.random.rand(10,3)*10  
df = pd.DataFrame(data=d, columns=['A','B','C'])  
df.describe()
```

Resúmenes y métricas estadísticas II

- `corr([method, min_periods])`: Compute pairwise correlation of columns, excluding NA/null values.
- `count([axis, level, numeric_only])`: Count non-NA cells for each column or row.
- `cov([min_periods, ddof])`: Compute pairwise covariance of columns, excluding NA/null values.
- `kurt([axis, skipna, level, numeric_only])`: Return unbiased kurtosis over requested axis.
- `mad([axis, skipna, level])`: Return the mean absolute deviation of the values over the requested axis.
- `max([axis, skipna, level, numeric_only])`: Return the maximum of the values over the requested axis.
- `mean([axis, skipna, level, numeric_only])`: Return the mean of the values over the requested axis.
- `median([axis, skipna, level, numeric_only])`: Return the median of the values over the requested axis.

Resúmenes y métricas estadísticas III

- `min([axis, skipna, level, numeric_only])`: Return the minimum of the values over the requested axis.
- `quantile([q, axis, numeric_only, interpolation])`: Return values at the given quantile over requested axis.
- `sem([axis, skipna, level, ddof, numeric_only])`: Return unbiased standard error of the mean over requested axis.
- `skew([axis, skipna, level, numeric_only])`: Return unbiased skew over requested axis.
- `std([axis, skipna, level, ddof, numeric_only])`: Return sample standard deviation over requested axis.
- `sum([axis, skipna, level, numeric_only, ...])`: Return the sum of the values over the requested axis.
- `var([axis, skipna, level, ddof, numeric_only])`: Return unbiased variance over requested axis.

Resúmenes y métricas estadísticas IV

Estadísticas

```
df.count()  
df.sum()  
df.mean()  
df.median()  
df.quantile(0.25)  
df.max()  
df.min()  
df.sem()  
df.var()  
df.std()  
df.mad()  
df.corr()  
df.cov()  
df.skew()  
df.kurt()
```

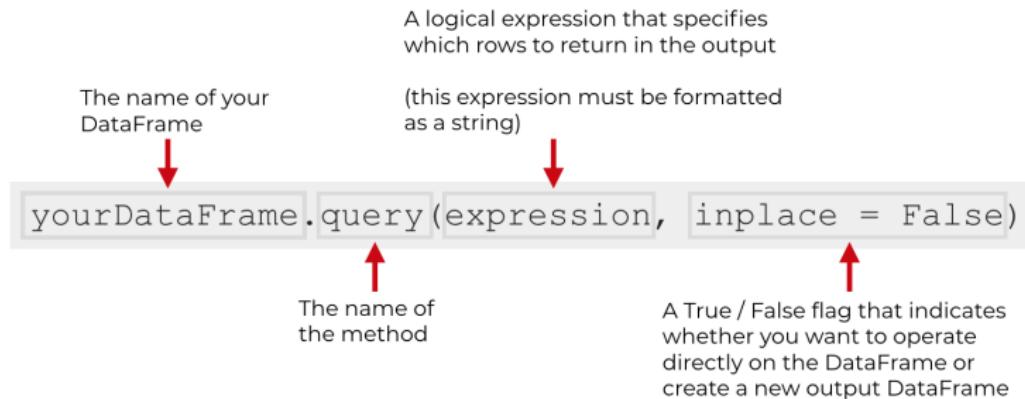
Manipulación de datos faltantes

Los valores que faltan pueden manejarse descartando/eliminando filas o columnas, o pueden imputarse.

- `drop([labels, axis, index, columns, level, ...])`: Drop specified labels from rows or columns.
- `dropna([axis, how, thresh, subset, inplace])`: Remove missing values.
- `fillna([value, method, axis, inplace, ...])`: Fill NA/NaN values using the specified method.
- `interpolate([method, axis, limit, inplace, ...])`: Fill NaN values using an interpolation method.

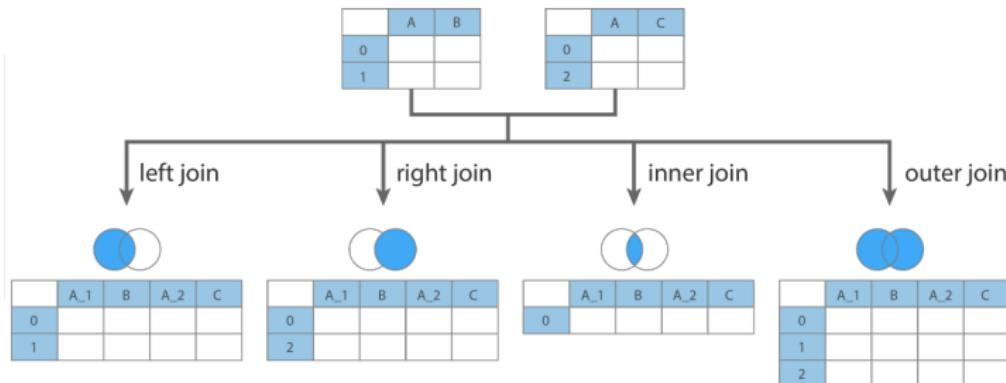
Consultas de datos (Query)

- `filter([items, like, regex, axis])`: Subset the dataframe rows or columns according to the specified index labels.
- `query(expr[, inplace])`: Query the columns of a DataFrame with a boolean expression.



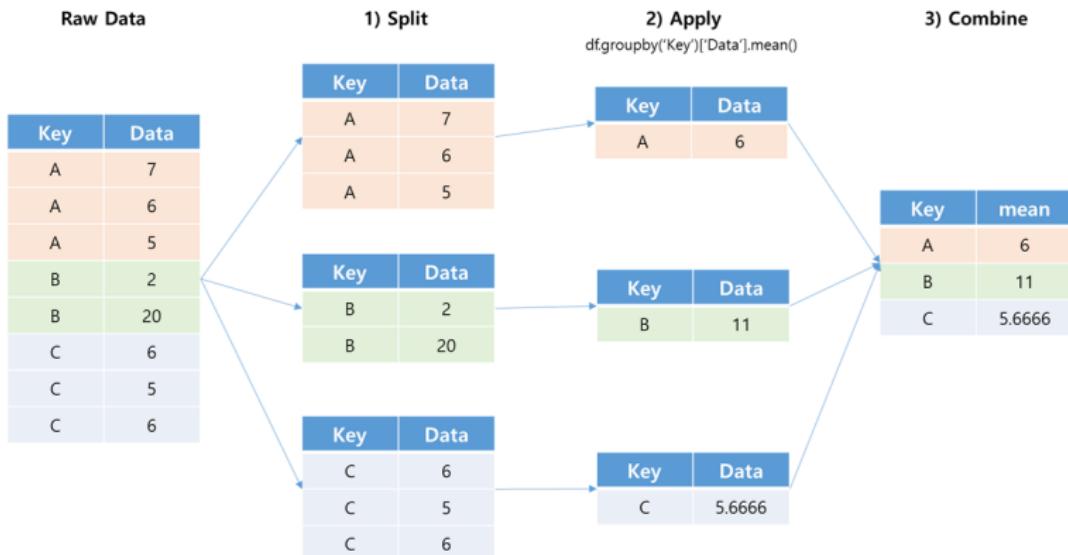
Combinación de DataFrame

- `pandas.concat(objs, axis=0, join='outer', ...)`: Concatenate pandas objects along a particular axis with optional set logic along the other axes.
- `merge(right[, how, on, left_on, right_on, ...])`: Merge DataFrame or named Series objects with a database-style join.



GroupBy - DataFrame

- `apply(func[, axis, raw, result_type, args])`: Apply a function along an axis of the DataFrame.
- `combine(other, func[, fill_value, overwrite])`: Perform column-wise combine with another DataFrame.
- `groupby([by, axis, level, as_index, sort, ...])`: Group DataFrame using a mapper or by a Series of columns.



Pivot Table

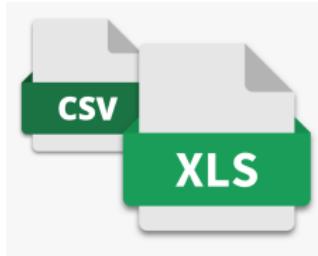
- `pivot_table([values, index, columns, ...])`: Create a spreadsheet-style pivot table as a DataFrame.

sepal.length	sepal.width	variety
5.1	3.5	Setosa
4.9	3	Setosa
4.7	3.2	Setosa
4.6	3.1	Setosa
5	3.6	Setosa
7	3.2	Versicolor
6.4	3.2	Versicolor
6.9	3.1	Versicolor
5.5	2.3	Versicolor
6.5	2.8	Versicolor
6.3	3.3	Virginica
5.8	2.7	Virginica
7.1	3	Virginica
6.3	2.9	Virginica
6.5	3	Virginica
7.6	3	Virginica
4.9	2.5	Virginica

variety	sepal.length	sepal.width
Setosa	4.86	3.28
Versicolor	6.46	2.92
Virginica	6.36	2.91

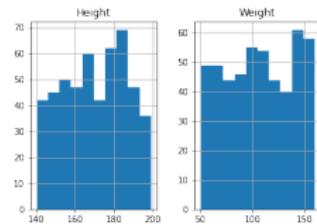
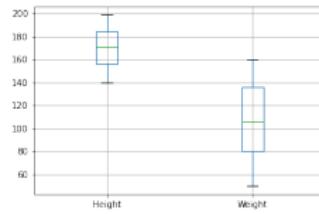
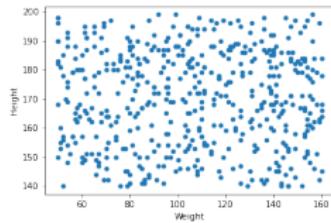
Conversión a CSV o EXCEL

- `to_csv([path_or_buf, sep, na_rep, ...])`: Write object to a comma-separated values (csv) file.
- `to_excel(excel_writer[, sheet_name, na_rep, ...])`: Write object to an Excel sheet.



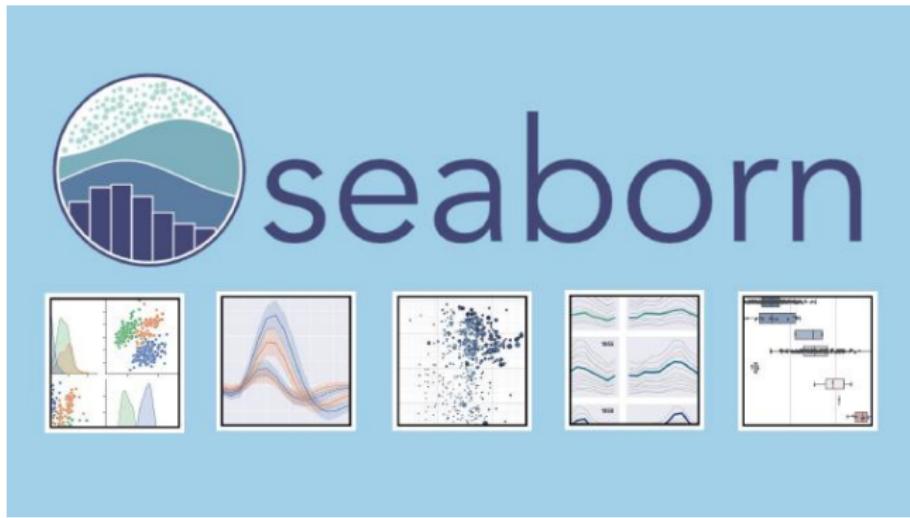
Ploting DataFrame

- `boxplot([column, by, ax, fontsize, rot, ...])`: Make a box plot from DataFrame columns.
- `plot`: for plotting a chart.
- `hist([column, by, grid, xlabelsize, xrot, ...])`: Make a histogram of the DataFrame's columns.



- 1 Contenido del Curso
- 2 Metodología de Ciencia de Datos
- 3 Análisis de datos con PANDAS
- 4 Visualización de datos con Seaborn
- 5 Análisis de Clustering con Scikit-Learn
- 6 Taller

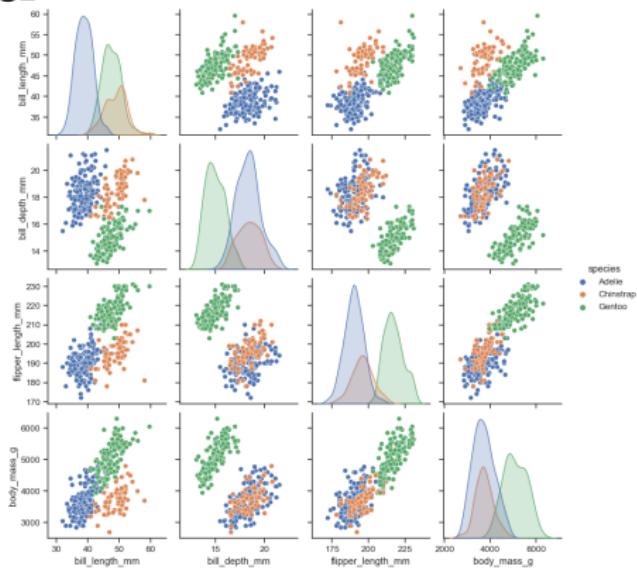
Visualización de datos con Seaborn



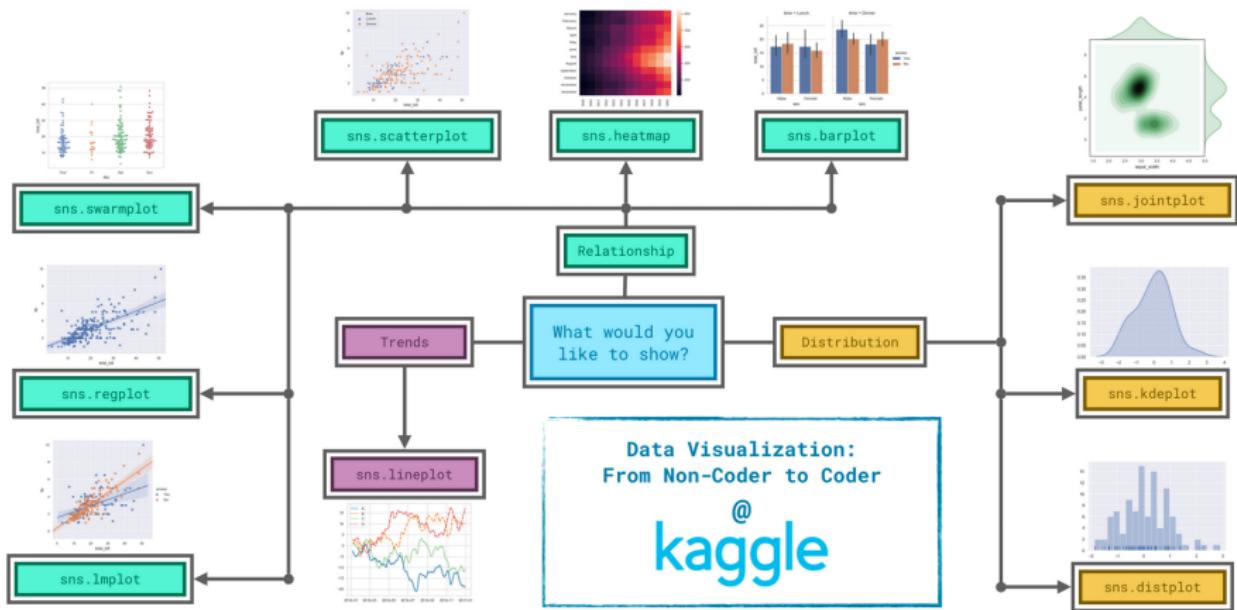
Pairplot

Plot pairwise relationships in a dataset.

```
seaborn.pairplot(data, hue=None, kind, diag_kind,...)  
    kind:{'scatter', 'kde', 'hist', 'reg'}  
    diag_kind:{'auto', 'hist', 'kde', None}
```



Galería Seaborn



<https://martinnormark.com/a-simple-cheat-sheet-for-seaborn-data-visualization-2/>

Categorical Plot

Figure-level interface for drawing categorical plots onto a FacetGrid.

```
seaborn.catplot(x, y, hue, data, col kind, ...)
```

```
kind:{'strip', 'swarm', 'box', 'violin', 'boxen', 'point', 'bar',  
      'count'}
```

Catplot

```
import seaborn as sns  
sns.set_theme(style="ticks")  
exercise = sns.load_dataset("exercise")  
g = sns.catplot(x="time", y="pulse", hue="kind", col="diet", data=exercise)
```

1 Contenido del Curso

2 Metodología de Ciencia de Datos

3 Análisis de datos con PANDAS

4 Visualización de datos con Seaborn

5 Análisis de Clustering con Scikit-Learn

- Definición de Análisis de clusters
- Medidas de Similaridad entre Objetos
- Método de K-Medias
- Métodos Jerárquicos Aglomerativos - AGNES
- Medidas de Calidad y Determinación del Número de Clusters

6 Taller

Scikit-Learn

[scikit-learn](https://scikit-learn.org/stable/) Install User Guide API Examples More ▾

scikit-learn

Machine Learning in Python

Getting Started Release Highlights for 0.24 GitHub

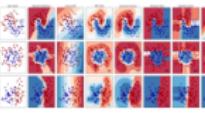
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



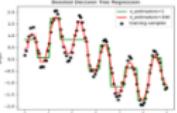
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



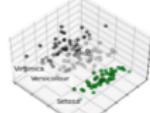
Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



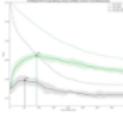
Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross-validation, metrics, and more...



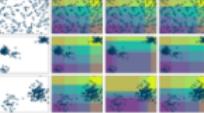
Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...



Examples

Análisis de Clusters. Definición

Análisis de Clusters

El análisis de clusters es la búsqueda de distintos grupos (cluster) en los datos, en los cuales datos del mismo cluster deben ser similares y marcadamente diferentes de los datos de las otras clases.

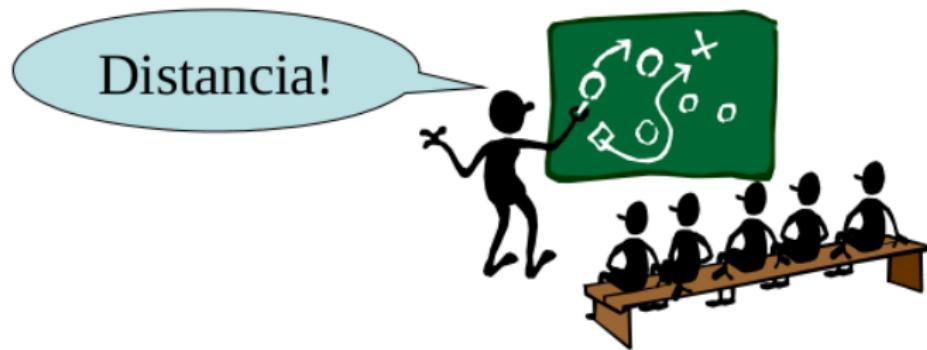
- El propósito de ello es descubrir algún patrón que permita discriminar entre los distintos grupos encontrados.
- En general, la similaridad entre los grupos se basa en la distancia.
- Los datos normalmente no tienen etiqueta alguna, pero también se pueden utilizar este tipo de algoritmos para datos etiquetados.

¿Por qué usar estos Métodos?

- La importancia de los datos radica en el análisis grupal de ellos y no en cada uno de los individuos.
- Se requiere simplificar la dimensión del conjunto de datos.
- No existe conocimiento suficiente de los conjuntos que pueden ser obtenidos a través de los datos.

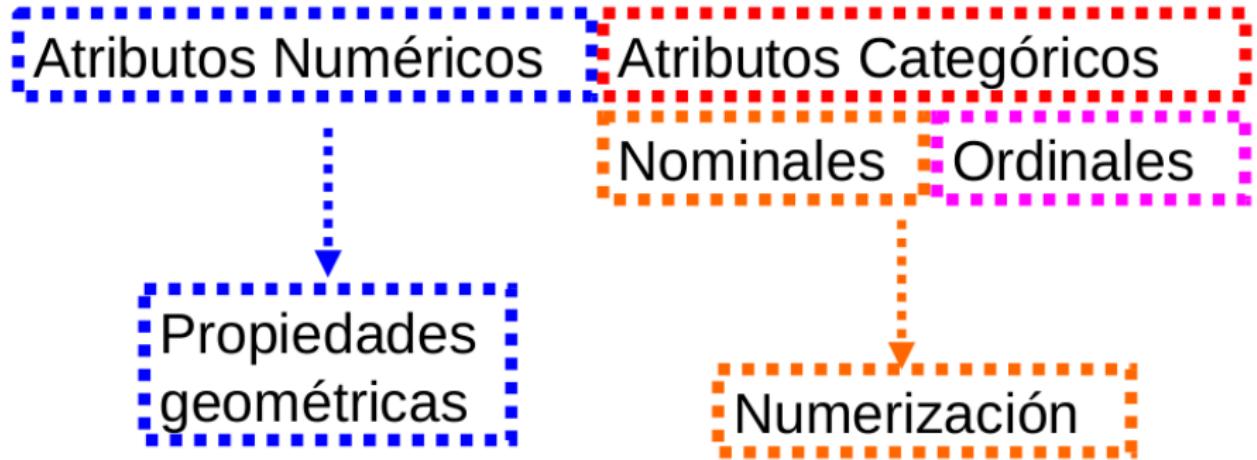
Similaridad entre dos objetos

¿Cómo saber si dos observaciones son similares?



La distancia entre las características de los objetos permite determinar el grado de similaridad que existe entre ellos.

Tipos de Atributos



Distancias para atributos Numéricos

Sean $\mathbf{x} = [x_1, \dots, x_d]^T$ y $\mathbf{y} = [y_1, \dots, y_d]^T$ vectores con d atributos.

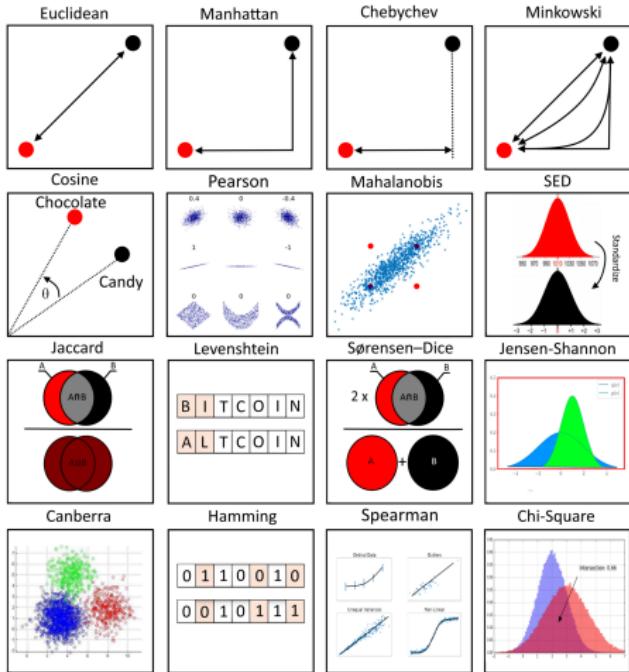
- ➊ *Distancia Euclídea:* Es la distancia clásica, y corresponde a la longitud de la recta que une dos puntos en el espacio euclídeo. Se conoce también como la métrica L_2 .

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^d (x_i - y_i)^2} = \sqrt{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}$$

- ➋ *Distancia de Manhattan:* Su nombre hace referencia a lo equivalente de ir recorriendo cuadras para llegar de un punto a otro. Se conoce también como la métrica L_1 .

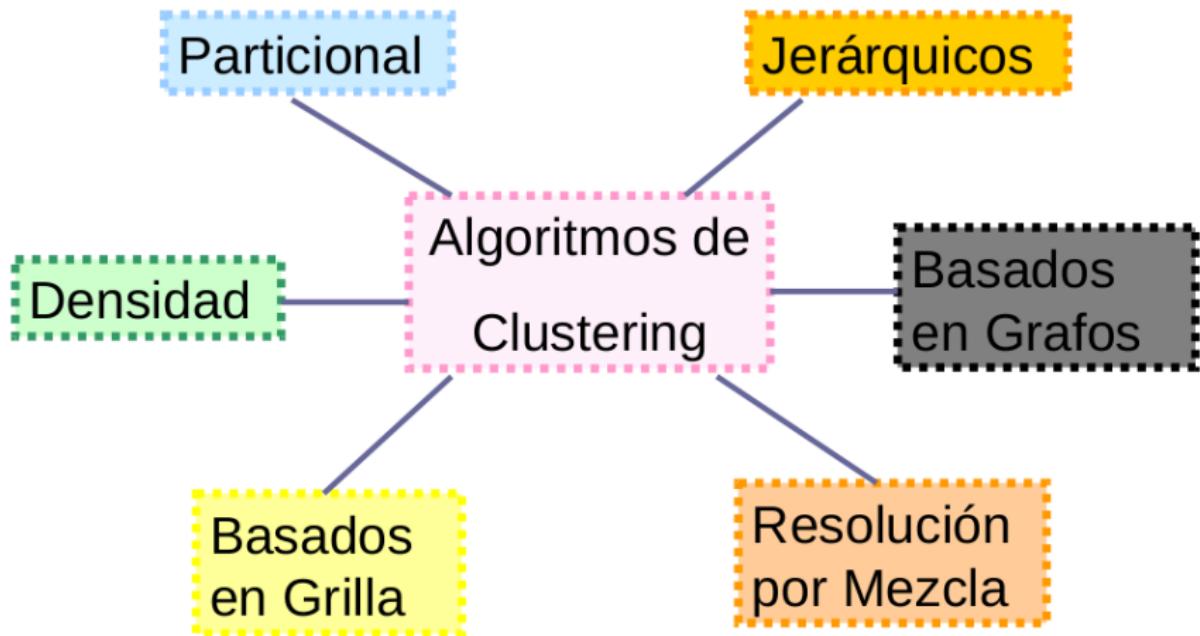
$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^d |x_i - y_i|$$

Resumen de Diferentes Métricas utilizadas en Ciencia de Datos



Mahmoud Harmouch. <https://towardsdatascience.com/>

Tipos de Algoritmos de Clustering



Familia de Métodos de Clustering

Los métodos existentes para el análisis de conglomerados, se separan en dos grandes grupos.

- **Jerárquicos:** Une o divide grupos existentes, produciendo una estructura jerárquica, la cual se muestra a través de un árbol o dendograma.
- **Partición:** Se generan K grupos a partir de los datos, este proceso se realiza localizando en forma iterativa los datos entre los grupos hasta que llegue a un estado de equilibrio. K es determinado por el usuario.

Método de *K*-Medias

Algoritmo *K*-Medias

Entrada: El conjunto de datos $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ que serán agrupados. El usuario define la cantidad de clusters K .

- ① Inicializar los centroides $\{\mathbf{q}_1, \dots, \mathbf{q}_k\}$ utilizando algún criterio de inicialización.
- ② Mientras el valor de los centroides cambien hacer:
 - ① Calcular la distancia euclídea de cada objeto \mathbf{x}_i , $i = 1..n$, a los centroides de cada cluster \mathbf{q}_k , $k = 1..K$: $d(\mathbf{x}_i, \mathbf{q}_k) = \sqrt{\sum_{j=1}^d (\mathbf{x}_i^{(j)} - \mathbf{q}_k^{(j)})^2}$
 - ② Agrupar cada objeto \mathbf{x}_i al centroide más cercano: $\eta(\mathbf{x}_i) = \arg \min_{\mathbf{q}_1, \dots, \mathbf{q}_K} \{d(\mathbf{x}_i, \mathbf{q}_k)\}$
 - ③ Actualizar la ubicación de los prototipos mediante el promedio de los datos del cual el centroide es el representante: $\mathbf{q}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathbf{x}_i$ donde $\mathbf{x}_i \in \mathcal{V}_k$, \mathcal{V}_k es el conjunto de datos cuya distancia más corta es el centroide k , n_k es la cantidad de elementos que pertenecen al conjunto \mathcal{V}_k .
- ④ Retornar los centroides $\{\mathbf{q}_1, \dots, \mathbf{q}_k\}$

Interactive K-Means I

En la siguiente aplicación se puede interactuar online con el K-Means:

<https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/>

K-Means app

K-Means Clustering Demo

This web application shows demo of simple k-means algorithm for 2D points. Just select the number of cluster and iterate.

This app is ultimately interactive. You can add more points or select template points from the right panel. More hints are available at the bottom.

Draw a point distribution:

123456

-+Iterations: 02 clusters

General Statistics:

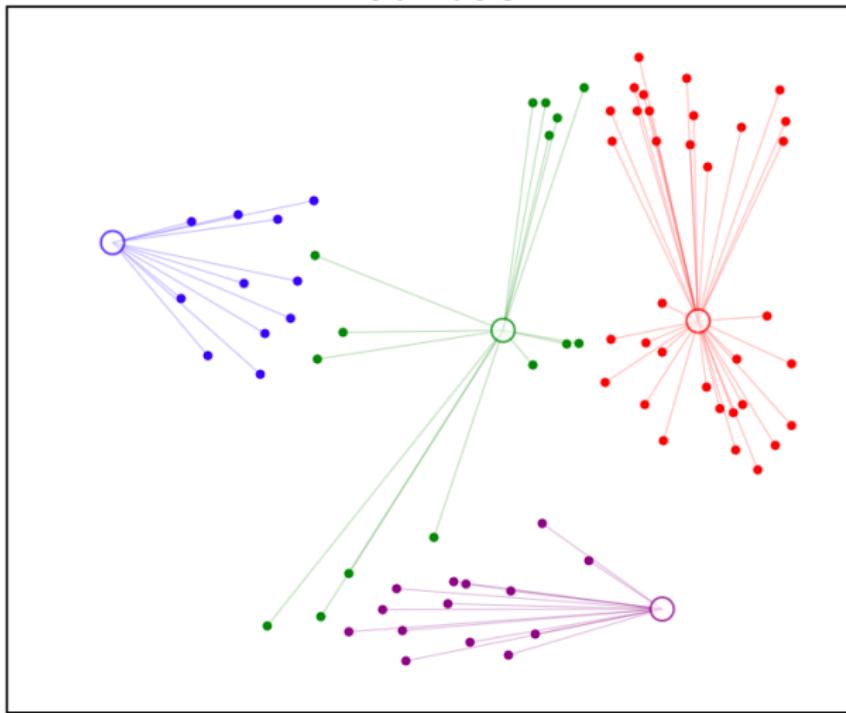
Data position: 0

Some hints for interactivity

- You can add more points by clicking or dragging in the area.
- Seed points (shown in empty circles) are randomly initialized. You can change by shift-dragging.

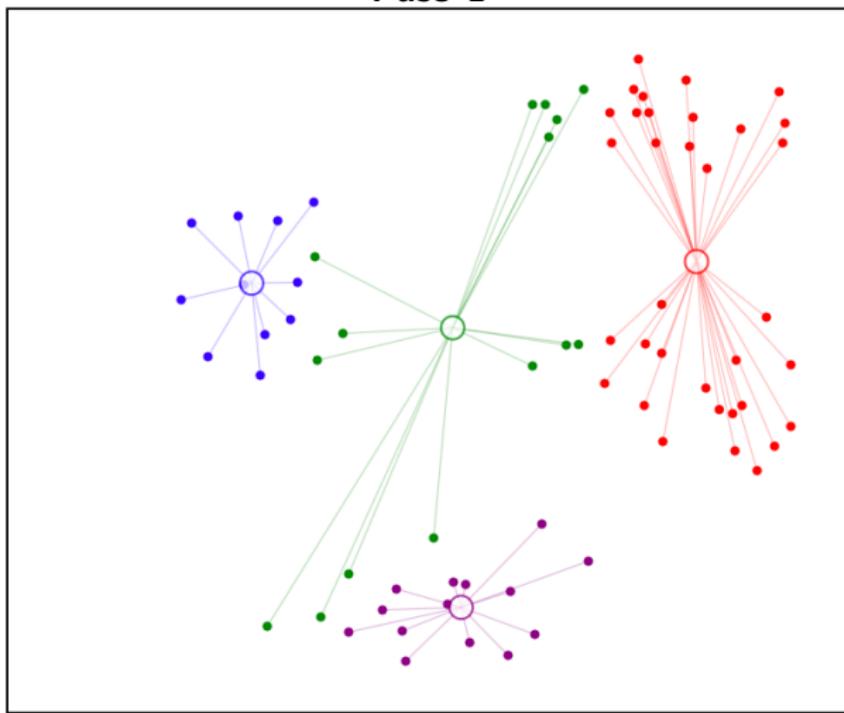
Interactive K-Means II

Inicialización



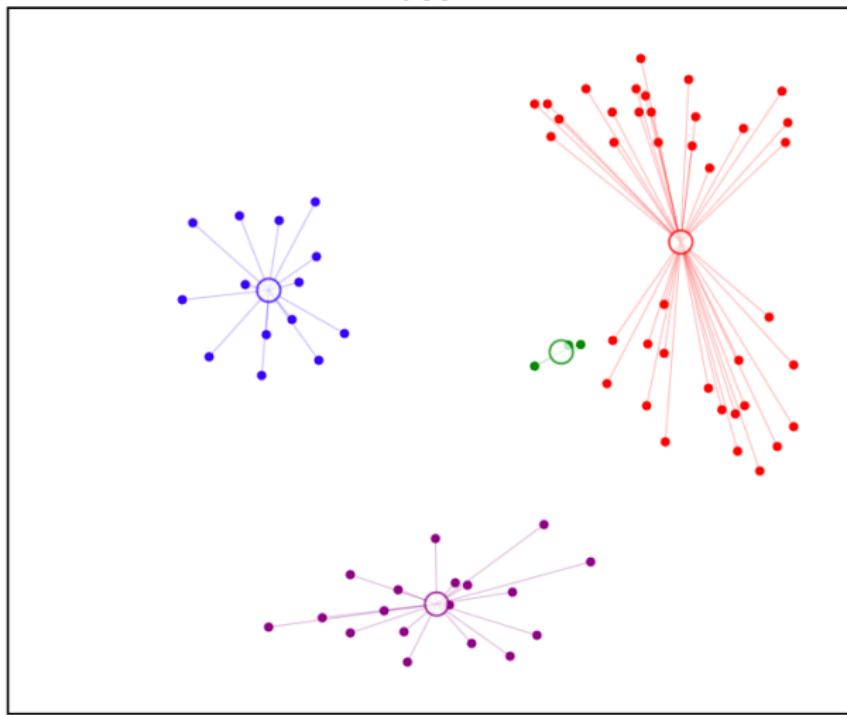
Interactive K-Means III

Paso 1



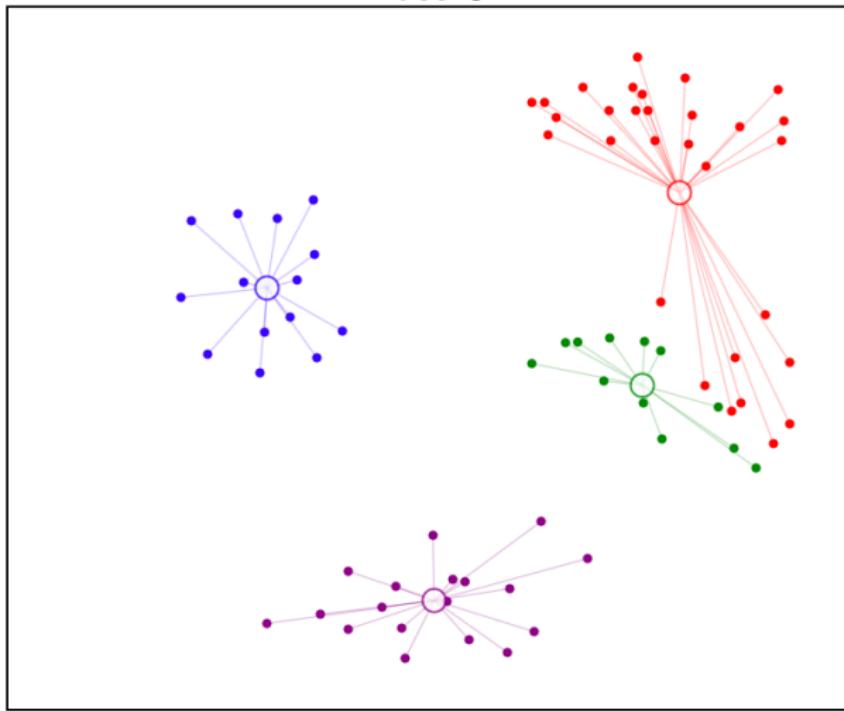
Interactive K-Means IV

Paso 2



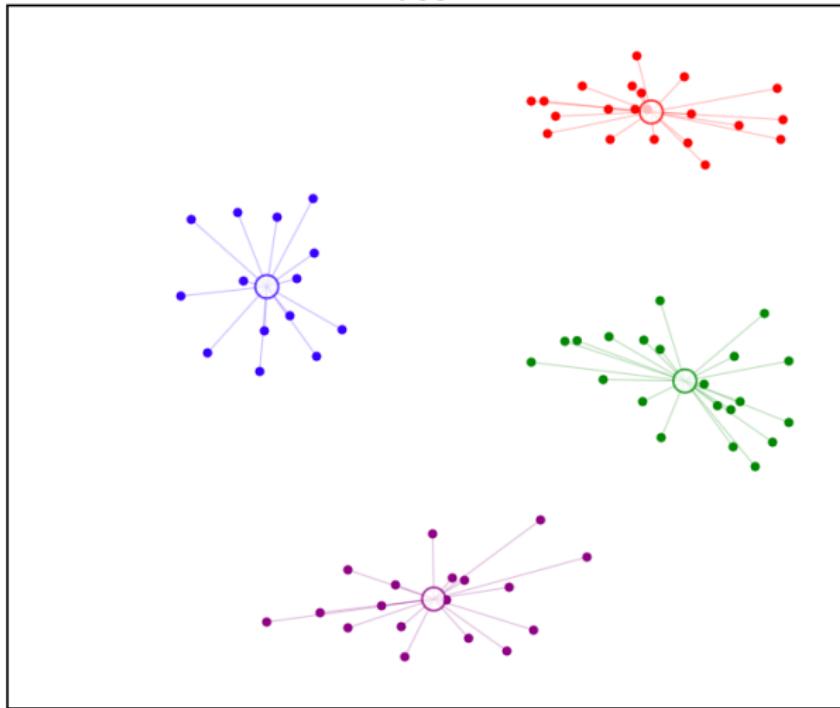
Interactive K-Means V

Paso 3



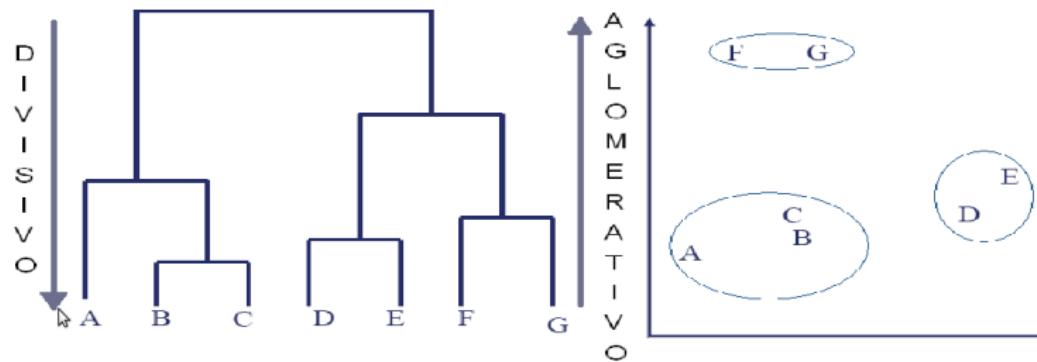
Interactive K-Means VI

Paso 4



Métodos Jerárquicos

- Los métodos jerárquicos unen o dividen grupos existentes mediante alguna función de similaridad, esta asignación es irrevocable.
- La salida de estos algoritmos corresponde a un árbol jerárquico el cual permite visualizar los cluster en distintos niveles.
- Existen dos tipos de algoritmos de aglomeración o de división.



Métodos Jerárquicos Aglomerativos

- Los algoritmos aglomerativos comienzan con tantos cluster como datos para proceder a unirlos hasta que todos los datos pertenecen a un puro cluster.
- La unión de clusters se realiza bajo un criterio de similaridad, donde los cluster más similares son unidos.
- El Método **AGNES (AGglomerative NESting)** es uno de los mas representativos de este tipo.

Algoritmo del Método AGNES

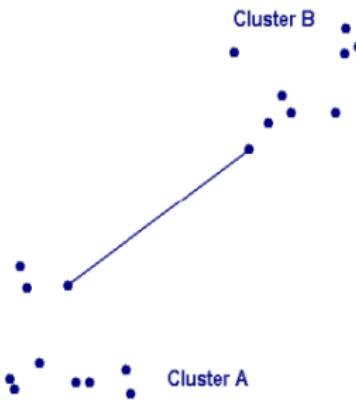
Método AGNES

- ① Generar tantos clusters, como datos existan.
- ② Seleccionar los dos cluster mas próximos y unirlos.
- ③ Recalcular las distancias entre los clusters.
- ④ Si existe más de una clase ir al paso 2.
- ⑤ Cortar el árbol donde se considere conveniente.
- ⑥ Fin del algoritmo.

Métodos AGNES I

- **Single Linkage:** La distancia entre los clusters corresponde a la mínima distancia existente entre dos puntos de los clusters.

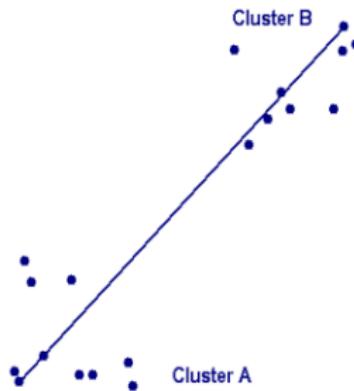
$$d(A, B) = \min_{x_i \in A, x_j \in B} d(x_i, x_j)$$



Métodos AGNES II

- **Complete Linkage:** La distancia entre los clusters corresponde a la máxima distancia existente entre dos puntos de los clusters.

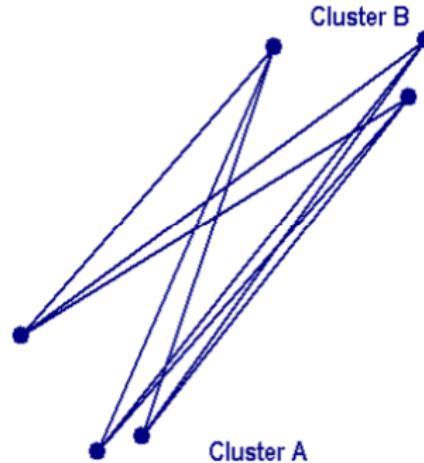
$$d(A, B) = \max_{x_i \in A, x_j \in B} d(x_i, x_j)$$



Métodos AGNES III

- **Average Linkage:** La distancia entre los clusters corresponde al promedio de todas las distancias entre los objetos del cluster A y B.

$$d(A, B) = \frac{1}{|A||B|} \sum_{x_i \in A, x_j \in B} d(x_i, x_j)$$



Métodos AGNES IV

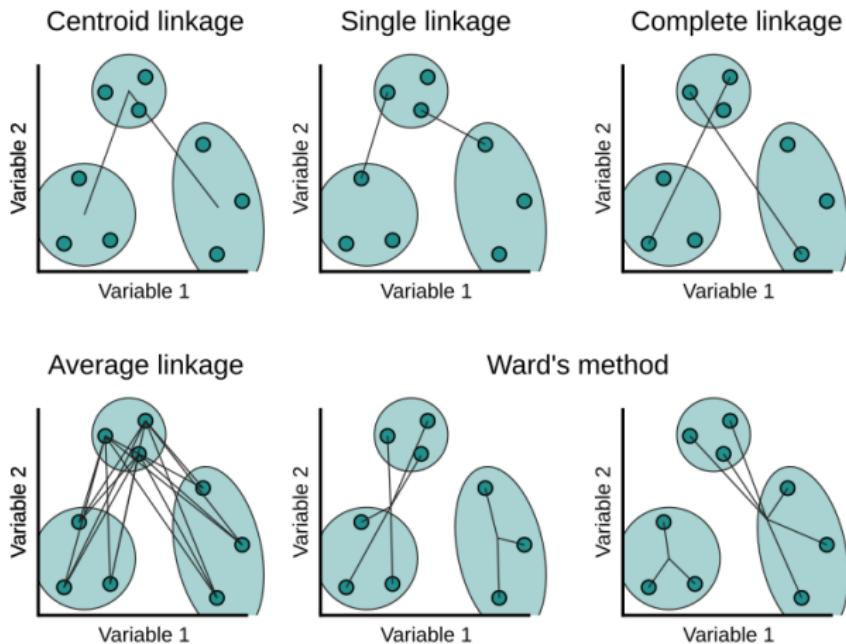
- **Método Ward:** la distancia cuantifica cuánto aumentará la suma cuadrática cuando se unen los clusters A y B:

$$\begin{aligned} W &= \sum_{\mathbf{x}_i \in A \cup B} \|\mathbf{x}_i - \mathbf{q}_{A \cup B}\|^2 \\ &\quad - \sum_{\mathbf{x}_i \in A} \|\mathbf{x}_i - \mathbf{q}_A\|^2 - \sum_{\mathbf{x}_i \in B} \|\mathbf{x}_i - \mathbf{q}_B\|^2 \\ &= \frac{n_A n_B}{n_A + n_B} \|\mathbf{q}_A - \mathbf{q}_B\|^2 \end{aligned}$$

donde \mathbf{q}_k es el centro del cluster k , y n_k es la cantidad de elementos en el cluster k .

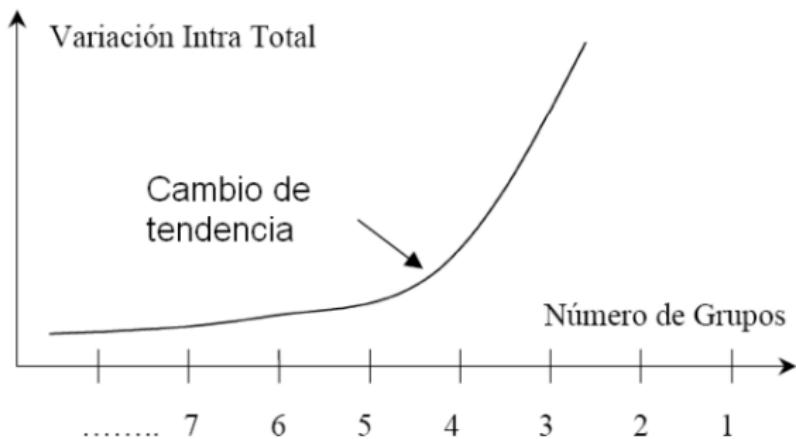
W mide el costo de combinar los clusters A y B.

Distancia entre Clusters



Medidas de Calidad

- Las Medidas de Calidad de los clusters corresponde a un índice que permite discernir entre distintas particiones realizadas por un mismo o diversos algoritmo(s) en un conjunto de datos.
- Existen diversas técnicas, dependiendo del tipo de algoritmo que se utiliza (jerárquico o de partición).



Coeficiente de Silhouette |

El coeficiente de Silhouette está dado por la siguiente expresión:

$$S(x) = \frac{a(x) - b(x)}{\max\{a(x) - b(x)\}}$$

donde $a(x) = \frac{1}{|A|-1} \sum_{y \in A} d(x, y)$ y $b(x) = \frac{1}{|B|} \sum_{y \in B} d(x, y)$

- Este coeficiente se basa en la distancia entre clusters, cada punto tiene un coeficiente de Silhouette propio determinado por $S(x)$, donde x pertenece a un cluster A .
 - $a(x)$ corresponde a la distancia promedio existente entre el punto x y los demás puntos de su mismo cluster A
 - $b(x)$ es la distancia promedio del punto x a los puntos del cluster B .
- Si $S(x)$ es cercano a 0 no existe diferencia entre los cluster, por lo cual se supone una mala asignación de K , en cambio si es cercano a 1 la elección de K fue correcta.

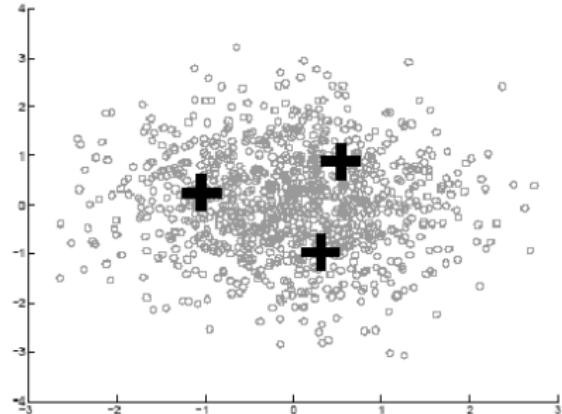
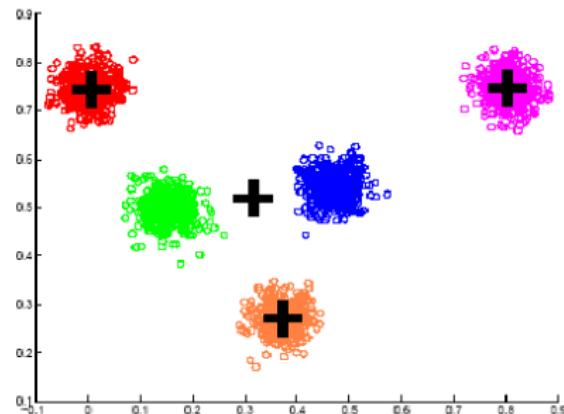
Coeficiente de Silhouette II

- Por último el $S(C)$ de un cluster corresponde al promedio de todos los $S(x)$ pertenecientes a un mismo cluster, donde $0.7 \leq S(C) \leq 1$ corresponde a un cluster fuerte, $0.5 \leq S(C) < 0.7$ es un cluster mediano, $0.25 \leq S(C) < 0.50$ es un cluster débil y $S(C) < 0.25$, no es un cluster.

$$S(A) = \frac{1}{|A|} \sum_{x \in A} S(x)$$

Determinación del número de clusters

- La determinación del número de cluster en un conjunto de datos es un problema abierto. Este consiste en determinar el número óptimo de clusters para los datos presentados.
- Es difícil definir conceptualmente y más aún estadísticamente la situación de una estructura correcta o no confusa.



Determinación del número de clusters

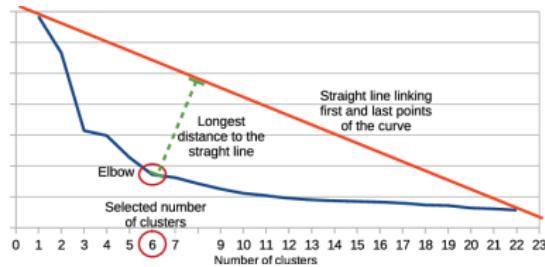
Se han desarrollado diversos métodos que solucionan el problema de la determinación del número de clusters, uno de ellos corresponde a realizar un breve estudio de los datos a analizar para determinar el número de cluster a buscar, otro método corresponde a los algoritmos dinámicos que varían el número de clusters y en base a cierta medida determinan el K óptimo.

Selección del Número de cluster usando el método de Elbow

Within Cluster Sum of Squares (WCSS)

Corresponde a la suma de todos los puntos de datos de la distancia al cuadrado entre un punto de datos \mathbf{x}_i de un grupo \mathcal{C}_k y el centroide de este grupo $\bar{\mathbf{X}}_k$

$$WCSS(K) = \sum_{k=1}^K \sum_{\mathbf{x}_i \in \mathcal{C}_k} \|\mathbf{x}_i - \bar{\mathbf{X}}_k\|^2$$



<http://dx.doi.org/10.1109/TASLP.2015.2479043>

Selección del Número de cluster usando Silhouette I

Interpretación del coeficiente de *Silhouette*:

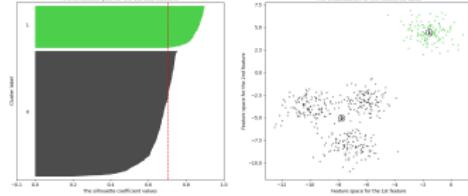
- ① Los valores cercanos a +1 indican que la muestra está lejos de los conglomerados vecinos.
- ② Un valor de 0 indica que la muestra está en o muy cerca del límite de decisión entre dos conglomerados vecinos
- ③ Valores negativos indican que esas muestras podrían haber sido asignadas al conglomerado incorrecto.

Interpretación del gráfico de *Silhouette*:

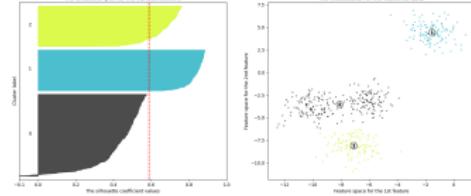
- La presencia de clústeres con puntajes de Silhouette por debajo del promedio estaría asociado a una mala elección de la cantidad de clusters.
- Amplias fluctuaciones en el tamaño de los gráficos de Silhouette podrían estar asociados a una mala elección de la cantidad de clusters.
- El grosor del gráfico de Silhouette muestra el tamaño del cluster.

Selección del Número de cluster usando Silhouette II

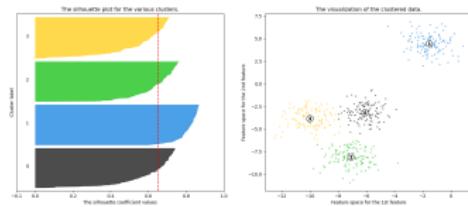
Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



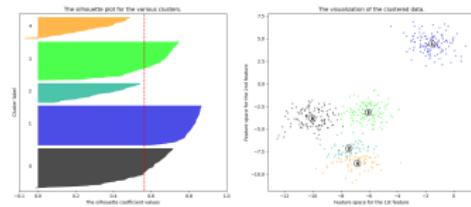
Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



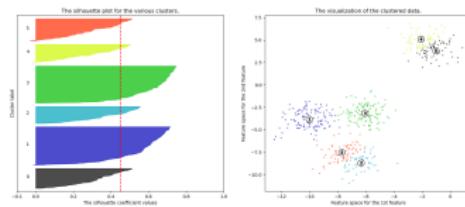
Silhouette analysis for KMeans clustering on sample data with n_clusters = 4



Silhouette analysis for KMeans clustering on sample data with n_clusters = 5



Silhouette analysis for KMeans clustering on sample data with n_clusters = 6



- 1 Contenido del Curso
- 2 Metodología de Ciencia de Datos
- 3 Análisis de datos con PANDAS
- 4 Visualización de datos con Seaborn
- 5 Análisis de Clustering con Scikit-Learn
- 6 Taller

Taller: Programación con Python

