

Metodologías de visualización de datos

**Breve descripción:**

Python es un lenguaje de programación muy versátil que se caracteriza por su facilidad en el manejo de “scripts” para trabajar con datos de manera fácil y efectiva, por eso la visualización de los datos utilizando Python es una de las formas más efectivas para el estudio de las ciencias de datos, junto a sus bibliotecas que permiten realizar muchas operaciones.

**Octubre 2023**

Tabla de contenido

[Introducción 1](#_Toc149213297)

[1. Python y Colab 3](#_Toc149213298)

[1.1. Listas y estructuras 5](#_Toc149213299)

[1.2. NumPy 7](#_Toc149213300)

[1.3. Condicionales y panda 13](#_Toc149213301)

[1.4. Funciones y paquetes 15](#_Toc149213302)

[1.5. Visualización y analytics 16](#_Toc149213303)

[2. Introducción a la estadística 20](#_Toc149213304)

[2.1. Análisis descriptivo 23](#_Toc149213305)

[2.2. Inferencia estadística 26](#_Toc149213306)

[2.3. Aprendizaje no supervisado 27](#_Toc149213307)

[3. EDA y tratamiento para datos 29](#_Toc149213308)

[3.1. Aprendizaje supervisado para regresiones 29](#_Toc149213310)

[3.2. Aprendizaje supervisado para clasificación 30](#_Toc149213311)

[3.3. Aprendizaje no supervisado y simulación 32](#_Toc149213312)

[Síntesis 36](#_Toc149213313)

[Material complementario 38](#_Toc149213314)

[Glosario 39](#_Toc149213315)

[Referencias bibliográficas 41](#_Toc149213316)

[Créditos 43](#_Toc149213317)

Introducción

La graficación de datos permite no solo una mejor visualización de estos sino una correcta interpretación de los mismos. Es por ello que se utilizan herramientas de proyección que permitan ver, a manera de cuadros y mapas, entre otras opciones, un panorama suficiente para determinar, de alguna manera, decisiones vitales para la empresa u orientar a un equipo a alcanzar algún objetivo. El siguiente video expone de forma introductoria lo que se desarrollará en este componente formativo:

1. Metodologías de visualización de datos



[Enlace de reproducción del video](https://youtu.be/-JuQjKfqYpY?si=KwPbl1bFKgeso197)

|  |
| --- |
| **Síntesis del video: Metodologías de visualización de datos** |
| En la actualidad, los datos y las herramientas de visualización son indispensables para la toma de decisiones.  Las herramientas de visualización proporcionan una manera de ver los datos en forma atípica y entender los patrones de estos.  Al observar un gráfico, se tiende con a identificar con rapidez las tendencias y sus valores atípicos, logrando realizar deducciones con mayor claridad y entendimiento para el grupo de trabajo.  Por ejemplo, en una hoja de cálculo con cientos de datos, es difícil captar una tendencia o encontrar algo que queramos identificar, lo que no sucede con las gráficas; es por eso que, en el campo del Big Data, reza una frase que dice: “Dice más una gráfica que mil palabras”.  De ahí que, hoy en día, las habilidades de las personas tienden a cambiar para volcarse a la interpretación de los datos, mediante la utilización de las herramientas que se han creado para ese fin.  A medida que la ciencia de datos se introduce cada vez más en la cotidianidad de las predicciones en una empresa, se torna indispensable para el futuro de las mismas, dándole sentido a los millones de datos que circulan diariamente por la red y facilitando la toma asertiva de decisiones por parte de los sectores productivos. |

# Python y Colab

**Colab**, también conocido como "Colaboratory", permite programar y ejecutar Python en su navegador, con las siguientes ventajas:

* No requiere configuración
* Da acceso gratuito a GPU
* Permite compartir contenido fácilmente

Colab puede facilitar su trabajo, ya sea estudiante, científico de datos o investigador de IA. Google Colaboratory es un entorno gratuito de Jupyter Notebook que no requiere configuración y que se ejecuta completamente en la nube. En ese sentido, Python permite diferentes ambientes en la programación, como la orientada a objetos y la funcional, pero lo más relevante de esta parte es que se utiliza como un lenguaje de scripts o intérprete.

Python tiene una funcionalidad de un modo interactivo, que trabaja de la mano con un intérprete de línea de comando que, al lanzar una orden, se obtiene un resultado. En sus inicios, las funcionalidades se quedaron limitadas con IPython, que más adelante evolucionó a Jupyter. Jupyter es un entorno que permite desarrollar líneas de Python dinámicamente; funciona como una aplicación local, donde se emula la función cliente-servidor y posibilita la ejecución de código; de esa manera, su interactividad posibilita la interpretación de su código de manera tal que pareciera la lectura de un documento a los ojos de cualquier persona.

Enseguida, se muestra la implementación y ejecución de Colab:

1. **Acceso a Google Colaboratory**. Google Colab es de carácter gratuito y para acceder a él, no es más que entrar a nuestra cuenta de Google, hacer clic derecho en nuestro Drive, ir al botón de nuevo y luego al submenú de más para seleccionar Colaboratory; creando así un nuevo cuaderno.
2. **Cuaderno de Google Colaboratory**. Una vez creado el apartado, se ha creado un cuaderno, que no es más que un documento que contiene código ejecutable, como Python; ese va a ser nuestro entorno de trabajo para Python. Es posible cambiar el nombre del documento; en tanto su extensión (\*.ipynb), viene de IPython Notebook, esto hace que nos permita ejecutar cuadernos tanto en IPython como en Jupyter y Colab.
3. **Codificando el cuaderno**. La composición de un cuaderno está dada por celdas, ya que estas son la mínima parte de ejecución dentro de un cuaderno. Es ahí donde escribimos nuestro código y lo ejecutamos; se puede realizar pulsando el botón de “play” ubicado a la izquierda de la línea o celda. Una vez realizada la ejecución, en la celda inferior, se encontrará el resultado de la ejecución inmediatamente anterior.
4. **Ejecución**. Se puede observar que las celdas de Colab son independientes, pero todas ellas trabajan con el mismo “kernel”; hay que recordar que un “kernel” es el corazón de un sistema operativo, pero, a su vez, no solo es el núcleo de un sistema, sino también un programa que controla todos los accesos al procesador y a la memoria.

Aunque el “kernel” que gobierna todas las celdas es único, las celdas se pueden ejecutar independientemente o estar conectadas por variables que se representen en diferentes partes del documento. La forma de ejecución que hace Colab es de abajo hacia arriba y el número que aparece en la parte izquierda hace referencia al orden de ejecución.

1. **Versión de Python**. Normalmente, un cuaderno en Colab utiliza el “kernel” con la versión de Python 3, pero si se quiere cambiar la versión de Python a la versión 2, se puede realizar ingresando al menú archivo y nos da la opción en ese apartado.

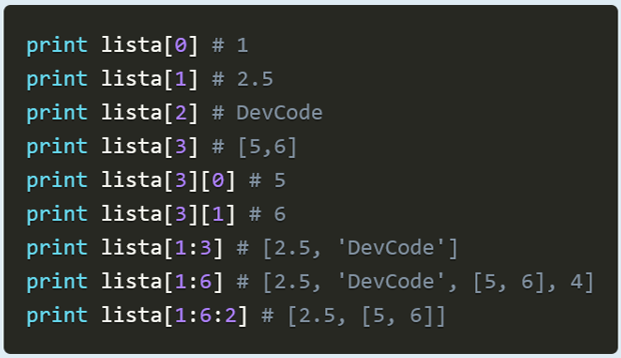
Gracias a Google Colaboratory se pueden llevar, de manera más sencilla, proyectos con esta útil y potente herramienta; estos entornos Colab son muy utilizados por la comunidad de desarrolladores a nivel mundial.

## Listas y estructuras

Python tiene un número considerable de estructuras de datos; es por eso que se dice que las listas no son más que estructuras de datos con unas características especiales, ya que permiten almacenar cualquier tipo de datos en ellas, tales como caracteres, enteros, cadenas.

Entonces, una lista es un arreglo donde es posible ingresar cualquier tipo de datos y estos están indexados para un fácil acceso.

1. Listas



De esta manera, es como se pueden insertar listas dentro de otra; por eso se dice que una lista no es más que una estructura de datos firme. Para emplear correctamente este tipo de dato, es necesario conocer los métodos y las diferentes operaciones que con la lista pueden realizar, tal como el ejemplo que se propone enseguida:

**my\_list = [2, 5, 'DevCode', 1.2, 5]**

Teniendo en cuenta el ejemplo anterior, a continuación, se muestran los métodos que son más importantes y propios de las listas:

1. **Append()**. Permite agregar nuevos elementos a una lista. Se puede agregar cualquier tipo de elemento a una lista pero, si agregamos una lista dentro de otra, simplemente se va a agregar como un solo elemento.

Ejemplo de Append:

**my\_list.append(10) # [2,5,'DevCode',1.2,5,10]**

**my\_list.append([2,5]) # [2,5,'DevCode',1.2,5,[2,5]]**

1. **Extend()**. Este método es importante y se diferencia de append, ya que agrega cualquier elemento tal como listas, pero este método agrega los elementos de una lista como elementos de la lista principal de donde está siendo agregado.

Ejemplo de Extend():

**my\_list.extend([2,5]) # [2,5,'DevCode',1.2,5,2,5]**

1. **Remove()**. Este método tiene la propiedad de remover. Solo afecta a la lista de donde esté siendo invocado. En el ejemplo inmediatamente siguiente, se muestra que se está removiendo el elemento 2 de la lista que tiene como nombre **my\_list**.

Ejemplo de Remove():

**my\_list.remove(2) # [5, ‘DevCode’, 1.2, 5]**

1. **Index()**. Este método solo devuelve el número del índice de lo que se le digite como parámetro. En este ejemplo, se está preguntando por el índice de la cadena DevCode en la lista My\_list, cuyo resultado es 2.

Ejemplo con Index():

**my\_list.index(‘DevCode’) # 2**

1. **Count()**. La función de este método es sencilla, solo muestra la cantidad de veces que un elemento de una lista se repite. En el parámetro de este método, se dice que nos muestre cuántas veces aparece el número 5 en la lista my\_list. Devuelve 2.

Ejemplo de Count():

**my\_list.count(5) # 2**

1. **Reverse()**. Reorganiza los elementos de una lista, los invierte.

Ejemplo de Reverse():

**my\_list.reverse() # [5,1.2,'DevCode',5,2]**

Se han mencionado los métodos más utilizados en Python, uno de los lenguajes de programación predilectos, hoy por hoy, por la comunidad desarrolladora del mundo, por su gran versatilidad y facilidad de interpretación, lo que también lo ha llevado a grandes controversias por esta misma comunidad.

## NumPy

Como se expuso en el anterior ítem, si se va a trabajar con Notebook en Google Colab, se debe saber que NumPy ya está instalado en el mismo.

Según NumPy v1.19 Manual (2022), NumPy es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, que incluyen manipulación matemática, lógica de formas, clasificación, selección, E/S, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.

Con base en este concepto, NumPy es imprescindible para la predicción. En el campo de los datos o ciencia de datos, circulan en las redes millones de datos todos los días, por eso, herramientas como esta se vuelven cada vez más importantes para ayudar a los desarrolladores o a los científicos de datos a tomar lo más relevante de ellos y convertirlo en información útil, que permita una predicción exacta.

NumPy es una poderosa herramienta que no solamente es utilizada en la parte científica, sino como una recolectora de grandes flujos de datos y, gracias a eso, se puede conectar con múltiples bases de datos; un punto más a su favor, que la hace más atractiva; de hecho, hay librerías, como Pandas, que están construidas sobre NumPy. Una de sus principales características es el manejo rápido de datos y el manejo de estructuras, que lo denominan por medio de “arrays” o arreglos, lo que en Python se llaman listas, pero opera los datos de una manera más ligera y eficiente.

La forma de almacenar los datos en NumPy es por medio de arreglos, lo que en Python son listas. La diferencia es que NumPy lo hace de una manera más rápida; el método invocado para la asignación de estos datos es **array()**, perteneciente a la librería NumPy.

Esta es la manera en que se estructura un arreglo en NumPy:

**import numpy as np**

**sample\_list = [1, 2, 3]**

**np.array(sample\_list)**

El resultado de la última línea de código será algo como array([1,2,3]). NumPy tiene dos tipos de arreglos que son: matrices y vectores.

Los **vectores** son sencillos y son arreglos unidimensionales que manejan una sintaxis sencilla, como la que se muestra enseguida:

**my\_vector = np.array(['este', 'es', 'un', 'vector'])**

Las **matrices** son unos arreglos bidimensionales y su sintaxis es un poco más compleja; sin embargo, su sintaxis se realiza invocando el método **np.array()**, así:

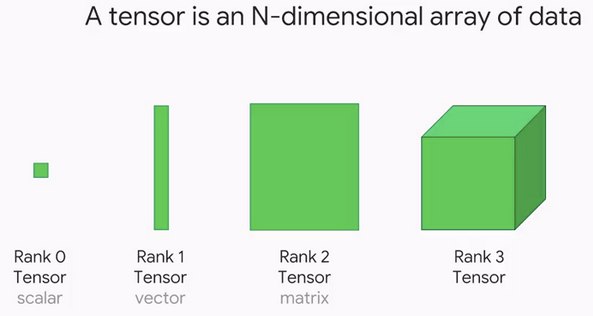
**my\_matrix = [[1,2,3],[4,5,6],[7,8,9]]**

**np.array(my\_matrix)**

Los arreglos de NumPy se pueden expandir para trabajar con matrices de uno, dos o más dimensiones; las operaciones que se pueden hacer con esta librería son extensas y es por eso que se mostrarán algunas de las más utilizadas, para conocer cómo es el trabajo y la sintaxis correcta utilizando NumPy. Es común que las listas de Python tengan gran similitud con los arreglos de NumPy, pero su gran diferencia radica en que son más rápidos, menos pesados y permiten generar “arrays” n-dimensionales.

Los “arrays” pueden variar en sus presentaciones unidimensional o vector, bidimensional o matriz y tensor.

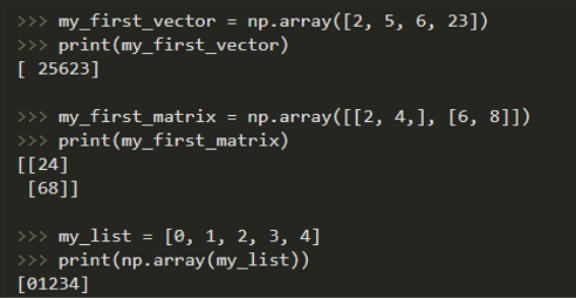
1. Tensor



Nota. Tomado de Kumar (2020).

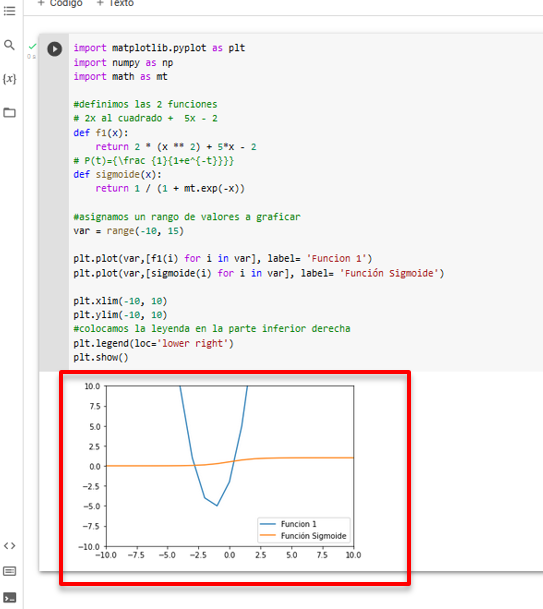
Crear un “array” con NumPy, se puede lograr a través de una lista o con una lista de listas, y la sintaxis para crearlo es **np.array(My\_list)** y, de la misma forma que las listas, es posible acceder a un valor por medio de los índices.

1. “Array”



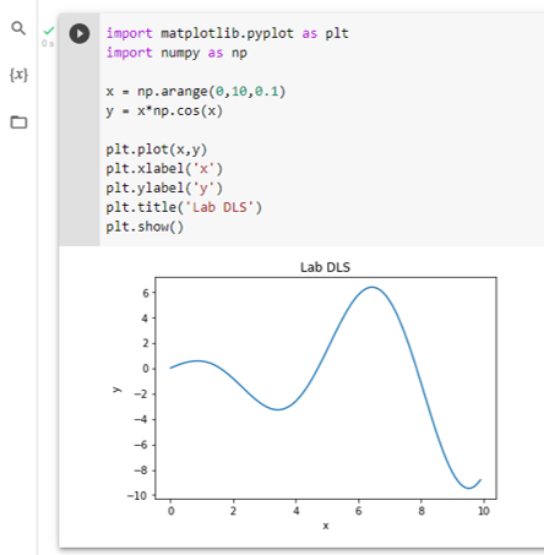
Utilizando los conceptos anteriores, cuando se trabaja con Google Colab, es posible dibujar una gráfica invocando las librerías Matplotlib y NumPy, con el código que se expone a continuación:

1. Dibujando una gráfica



De igual forma, también se pueden trazar gráficas fácilmente con Python al tener las herramientas correctas e invocando las librerías correspondientes. La siguiente es otra gráfica, la cual se obtiene con Matplotlib y NumPy:

1. Dibujando una gráfica



## Condicionales y panda

Según Datacarpentry.org (2001), Un “DataFrame” es “una estructura de datos con dos dimensiones, en la cual se pueden guardar datos de distintos tipos (como caracteres, enteros, valores de punto flotante, factores y más) en columnas”.

Es similar a una hoja de cálculo o una tabla de SQL, o el data.frame de R. Un “DataFrame” siempre tiene un índice (con inicio en 0). El índice refiere a la posición de un elemento en la estructura de datos. Resulta un poco complicado hablar de Pandas y sus condicionales sin antes hablar un poco de lo que es un “Dataframe”. Sin más preámbulos, un “Dataframe” es un marco de trabajo, donde se estipulan unos parámetros para poder ingresar los datos y estos, a su vez, ser representados con estilo. Estos datos son etiquetados, tanto en filas como en columnas.

El video que se propone enseguida, expone cómo es posible realizar lo indicado:

1. Condicionales y Pandas



[Enlace de reproducción del video](https://youtu.be/dkUlltYfeMo?si=pJ6TyNHvHzXBhmnb)

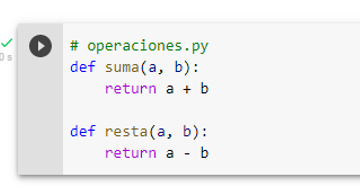
|  |
| --- |
| **Síntesis del video: Condicionales y Pandas** |
| Para poder trabajar con Pandas, veremos, a continuación, unos ejemplos que nos pondrán en contexto acerca de la utilización de los condicionales, trabajando con lo que se ha visto hasta este momento.  De este ejercicio presentado, podemos hacer una serie de ajustes a las columnas y filas para que muestren estilos diferentes. Por defecto, los “Dataframes” tienen seis cifras decimales, sin embargo, si esto no es suficiente para definir algo, o si no aporta suficiente información, se le puede indicar al “Dataframe” que muestre un formato diferente, a través del Método format() con la propiedad “style” del “Dataframe”, como veremos en nuestro siguiente ejemplo.  Veamos como quedaría. Esto gracias a los “Dataframe” y a la gran variedad de opciones que se les puede aplicar.  También podemos trabajar en una de estas posibles, como lo es la de los porcentajes en los “Dataframe”.  En esta visual, vemos que es un “Dataframe” con un formato de porcentaje, configurado para no tener la necesidad de multiplicar por 100 cada uno de sus elementos.  Finalmente, si se requiere, solo se pueden afectar algunas columnas del “Dataframe”, como, por ejemplo, algunas que tengan porcentajes y las otras que sigan igual.  De esta manera, trabajar con condicionales y Panda resulta de mucha utilidad a la hora de la clasificación de los datos. |

## Funciones y paquetes

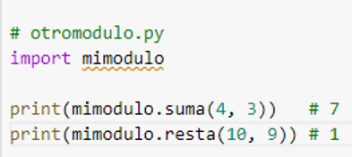
Los paquetes son partes de código que aumentan la operabilidad de Python. Un paquete se ve como una carpeta con varios contenidos, entre archivos y carpetas con sus respectivas extensiones Python (.py); la utilidad de los paquetes en la ciencia de datos es la facilidad de reutilizar código, permitiendo usar clases y objetos que fácilmente pueden ser usados por otros módulos y organizarlos de una mejor manera en namespaces.

Se demuestra esto a través del siguiente ejemplo:

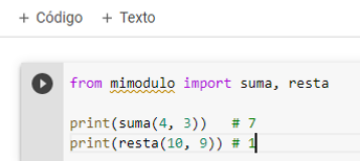
1. Se podría definir un módulo de operaciones (.py) con las funciones de suma() y resta().



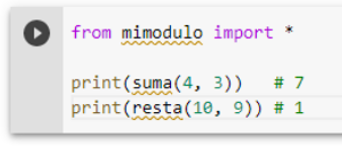
1. Una vez se haya definido ese módulo, se puede importar en otro archivo utilizando import, e importar lo que contenga.



1. Ahora bien, si se prefiere, se pueden importar los módulos de interés.



1. También, se puede importar todo el módulo utilizando “\*” y no mimodulo.\*. De esta forma, se pueden agrupar funciones en paquetes y que estos puedan ser utilizados por otros módulos.



## Visualización y “analytics”

La presentación de los datos no es algo que toque dejar pasar por desapercibido, ya que la parte visual ayuda, en gran medida, a una buena interpretación de estos, realizando grandes esfuerzos también para poder llevar a cabo una buena depuración e integración que, si no se hace de manera correcta, el usuario no va a estar satisfecho con la interpretación o, simplemente, no va a entender lo que se quiere orientar; de ahí, parte la importancia de la visualización analítica de los datos.

La visualización de la información tiene tres componentes inexorables para poder concluir con éxito un análisis, los cuales son narrativa, diseño y estadística; estas tres etapas deben ir de la mano para poder dar una buena orientación a la hora de plasmar visualmente y no perderse en el intento.

Una excelente interpretación visual debe tener los siguientes ingredientes:

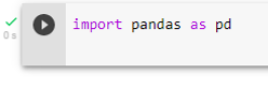
* Identificar relaciones, las sentencias y los patrones.
* Inspeccionar datos para deducir nuevas tendencias o conocimientos.
* Mejorar el entendimiento de una idea o tendencia.
* Interpretar de manera concreta la observación de evidencias reales y su variabilidad en el tiempo.
* Posibilitar retener una idea.

Los anteriores consejos son los necesarios a tener presentes a la hora de realizar una visualización precisa de lo que se quiere, realmente, mostrarle al cliente y que este interprete de manera eficaz.

Es importante determinar que para analizar datos tubulares en Python se debe hacer uso de las librerías de análisis de datos denominadas Pandas, las cuales generan gráficos de alta calidad con Matplotlib; adicionalmente, se asocian con bibliotecas que usan “arrays” de NumPy, que, como se sabe, es otra biblioteca de Python.

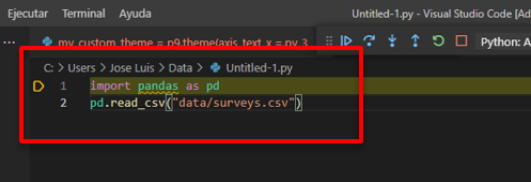
Al trabajar con Python, este no carga las bibliotecas; este proceso debe hacerse con el enunciado **import**, para acceder a las funciones de la biblioteca; la sintaxis es la siguiente: **import nombredelabiblioteca**; además de eso, se le puede agregar una abreviación para no digitar los comandos con nombres extensos, de la siguiente manera:

1. Sintaxis



Los “Dataframe” juegan una importante función en la lectura de datos y, sobre todo, en la selección de los mismos. Cuando se tenga el archivo con la información de la que se desea tomar lectura, de extensión **.CSV**, que es la extensión para abrir directamente desde un “Dataframe”, se realiza de lo siguiente:

1. “Dataframe”



En cuanto a la visualización de datos, después de introducir parámetros según el requerimiento o los datos que son objeto de proyección, aquí se visualizan mediante la librería **plotnine**:

1. Visualización mediante Plotnine



Amplíe sus conocimientos, sobre este tema, ingresando al enlace que aquí se propone; este contiene ejercicios e información de valor sobre analítica de datos y creación de gráficos con Plotnine:

<https://datacarpentry.org/python-ecology-lesson-es/07-visualization-ggplot-python.html>

# Introducción a la estadística

Hoy en día, es común escuchar el término “Big Data” y, con eso, la necesidad de conocer acerca de la probabilidad y estadística como elementos imprescindibles. Los temas de minería de datos, “machine learning”, más los millones de datos con los que se cuenta a diario, es lo que hoy se conoce como la ciencia de datos en conjunto. En este mundo, Python es el lenguaje potente que más permite trabajar con los datos y analizarlos de manera estadística.

Según López Briega (2018), la estadística suele ser definida como la ciencia de aprender de los datos o como la ciencia de obtener conclusiones en la presencia de incertidumbre.

Se relaciona principalmente con la recolección, análisis e interpretación de datos, así como con la efectiva comunicación y presentación de los resultados basados en esos datos; estos últimos, entendiéndose como cualquier clase de información grabada.

La estadística juega un rol importante en muchas disciplinas científicas; de allí que ha jugado un papel importante en la toma de decisiones, pues existe innumerable información valiosa entre los datos. La estadística brinda opciones o métodos para extraer esta información adecuadamente y entenderla, verificando la calidad de esta. Se encuentra dividida en dos ramas: estadística descriptiva y estadística inferencial, como se expone en el siguiente video:

1. Introducción a la estadística



[Enlace de reproducción del video](https://youtu.be/wMCDkknpUVw?si=hCD9uJ0A6PJRBR8e)

|  |
| --- |
| **Síntesis del video: Introducción a la estadística** |
| La estadística es muy importante para la toma de decisiones a partir de la recolección, análisis e interpretación de datos. Esta se divide en dos ramas: la estadística descriptiva y la estadística inferencial.  La primera, la estadística descriptiva, se dedica a recolectar, ordenar, analizar y representar un conjunto de datos, con el fin de describir apropiadamente las características de este. Calcula los parámetros estadísticos que describen el conjunto estudiado.  Algunas de las herramientas que utiliza son: gráficos, medidas de frecuencias, medidas de centralización, medidas de posición, medidas de dispersión, entre otras.  La estadística inferencial, por su parte, estudia cómo sacar conclusiones generales para toda la población, a partir del estudio de una muestra, el grado de fiabilidad o significación de los resultados obtenidos.  Sus principales herramientas son: el muestreo, la estimación de parámetros y el contraste de hipótesis.  A su vez, hay otro concepto importante, como lo es la probabilidad, que se encarga de qué tan probable se encuentra algo de tener algún resultado.  Cuando se está realizando algún experimento aleatorio, al calcular la probabilidad de un evento, hay que tener en cuenta todas las posibles opciones del mismo, de cuántas maneras puede ocurrir una determinada situación.  Entonces, así como la estadística nos generaliza desde los datos obtenidos acerca de algo, la probabilidad hace lo opuesto. Cuando asumimos saber cómo funcionan las cosas, entonces sabemos la clase de datos que vamos a ver y lo probable que sea volver a verlos. |

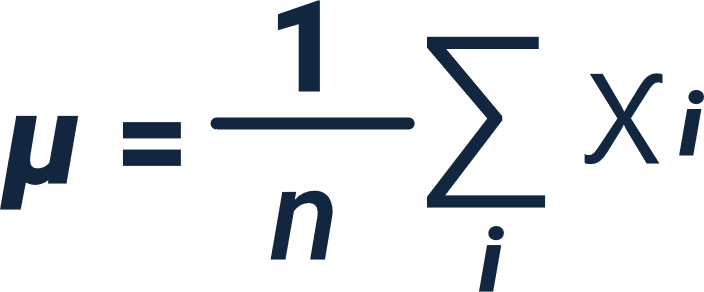
Las siguientes son algunas actividades básicas en el análisis estadístico:

* Diseño del análisis.
* Exploración de datos.
* Armado del modelo.
* Realizar estimaciones.
* Contraste de la hipótesis.

## Análisis descriptivo

Para exponer la particularidad de los datos, es necesario tener diferentes medidas, estas son algunas de ellas:

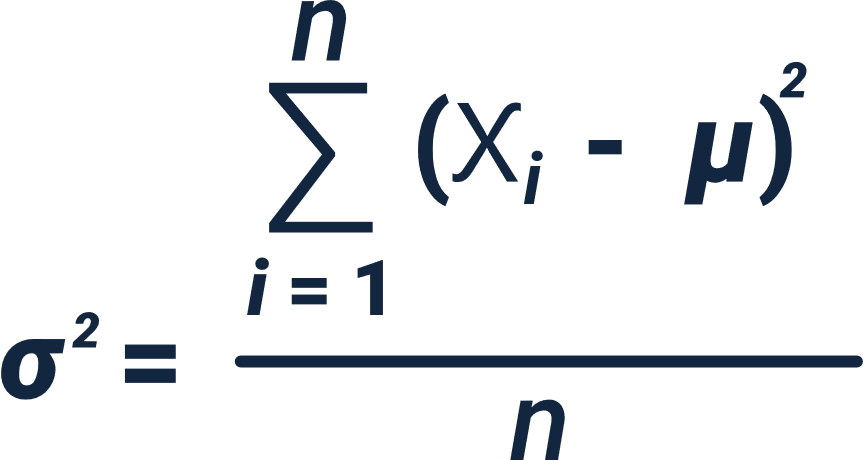
1. **Media aritmética**. Indica el resultado de haber sumado todos los datos y dividirlos por el total de los elementos sumados. Se representa por la letra griega **μ**. Dicho de otra manera, si tenemos una muestra de n valores Xi, la media aritmética μ, es la suma de los valores divididos por el número de elementos, así:



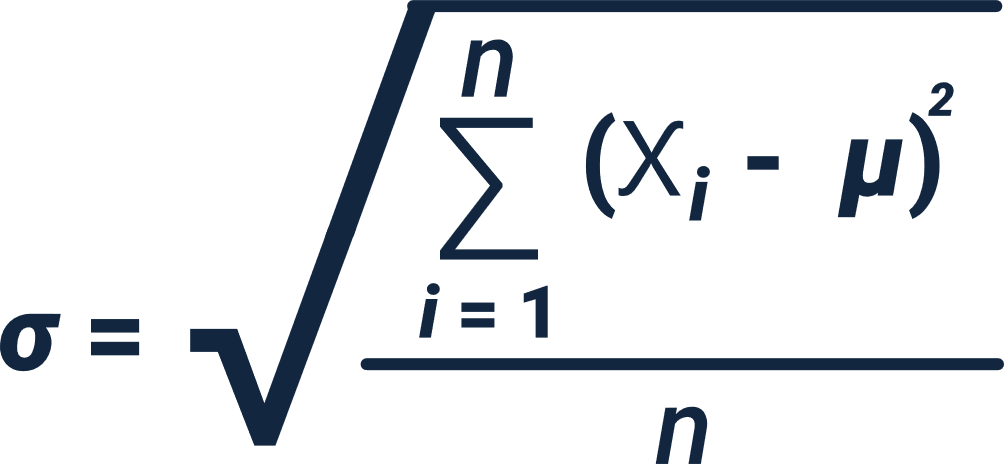
1. **Desviación respecto a la media**. La definición puntual de la desviación respecto a la media enfatiza en la diferencia en el valor absoluto entre cada valor de la variable estadística y la media aritmética.



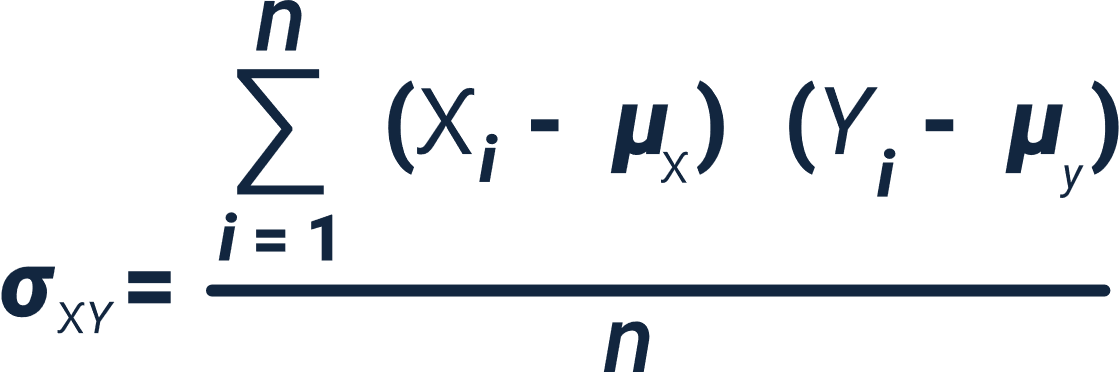
1. **Varianza**. Es la media aritmética del cuadrado de las desviaciones respecto a la media de una distribución estadística. La varianza intenta describir la dispersión de los datos. Se representa como σ2.



1. **Desviación típica**. Es la raíz cuadrada de la varianza. Se representa con la letra griega **σ**.



1. **Moda**. La moda es el valor que tiene mayor frecuencia absoluta. Se representa con M0.
2. **Mediana**. Es el valor que ocupa el lugar central de todos los datos cuando estos están ordenados de menor a mayor. Se representa con x˜
3. **Correlación**. Indica el resultado de haber sumado todos los Trata de establecer la relación o dependencia que existe entre las dos variables que intervienen en una distribución bidimensional. Es decir, determinar si los cambios en una de las variables influyen en los cambios de la otra. En caso de que suceda, diremos que las variables están correlacionadas o que hay correlación entre ellas. La correlación es positiva cuando los valores de las variables aumentan juntos; y es negativa cuando un valor de una variable se reduce cuando el valor de la otra variable aumenta.
4. **Covarianza**. Es el equivalente de la varianza aplicado a una variable bidimensional. Es la media aritmética de los productos de las desviaciones de cada una de las variables respecto a sus medias respectivas. La covarianza indica el sentido de la correlación entre las variables: si σxy>0, la correlación es directa; si σxy<0, la correlación es inversa.



1. **Valor atípico**. Es una observación que se aleja demasiado de la moda; está muy lejos de la tendencia principal del resto de los datos. Pueden ser causados por errores en la recolección de datos o medidas inusuales. Generalmente, se recomienda eliminarlos del conjunto de datos.

Como se observa, el análisis descriptivo se plantea para dar algún tipo de respuesta a algún planteamiento formulado al inicio del requerimiento estadístico; sin embargo, lo que busca este tipo de análisis estadístico es realizar una recolección de datos y tratar de organizarlos a manera representativa por medio de gráficos y medios visuales de fácil interpretación para el usuario, y así poder extraer las características que más representen a un grupo de datos, para luego trazar una tendencia de ellos.

De esta manera, hoy se puede decir que es un método que va de la mano con los sucesos y se enfoca en ellos, recolectando información y conectando los datos entre sí, de manera que sea posible conformar una información precisa y contundente; gracias a este tipo de análisis, se puede saber más acerca de una población en especial y tomar algún tipo de determinación según la información, tanto cualitativa como cuantitativa.

Python maneja unas librerías que facilitan el manejo de las matemáticas y la estadística; no es la excepción y, para fortuna de todos, la comunidad de Python es tan grande que es normal siempre encontrar librerías que solucionan cualquier situación. En ese orden de ideas, Python ofrece un abanico de librerías que ayudan a la solución de problemas estadísticos y de probabilidad; las librerías ideales para este tipo de ejercicios son las siguientes:

* NumPy
* scipy.stats
* statsmodels
* Matplotlib
* Seaborn
* Pandas
* pyMC

## Inferencia estadística

Según López Briega (2018), “La estadística inferencial estudia cómo sacar conclusiones generales para toda la población a partir del estudio de una muestra y el grado de fiabilidad o significación de los resultados obtenidos. Sus principales herramientas son el muestreo, la estimación de parámetros y el contraste de hipótesis”.

La probabilidad aporta componentes fuertes para trabajar con la estadística inferencial, teniendo entre los datos una posibilidad de reducción de errores cuando se está examinando su comportamiento en el tiempo. Cuando se habla de inferencia, se habla de concluir desde supuestos particulares o generalizados.

Hay dos métodos para la inferencia estadística:

1. **Métodos de estimación de parámetros**. Consisten en asignar un valor al parámetro caracterizado en un conjunto de datos que son el objeto de estudio. Cuando se habla de estimación, también se está contemplando la posibilidad de errores; para la reducción de estos, hay que puntualizar intervalos de confianza.
2. **Métodos de contraste de hipótesis**. Tienen como objetivo comprobar si una estimación corresponde a los valores poblacionales. Cuando se hace un contraste de hipótesis, siempre hay dos supuestos: la hipótesis nula y la hipótesis alternativa.

Cuando se traslada la estadística inferencial al plano de desarrollo en Python, se está hablando de invocar las librerías que se estudiaron en los anteriores ítems realizando básicamente las mismas operaciones, pero con distintos objetivos, dependiendo del criterio con que se utilicen.

## Aprendizaje no supervisado

Los algoritmos de lenguajes no supervisados son un conjunto de datos sin etiquetas y desordenados, que no tienden a indicar nada; estos métodos de aprendizaje no supervisado no se pueden aplicar a problemas de clasificación o de regresión, porque no se sabe el resultado que van a arrojar; sin embargo, el aprendizaje sin supervisión se dice que puede descubrir la estructura subyacente de los datos.

Los algoritmos de aprendizaje sin supervisión permiten realizar operaciones más robustas que los supervisados, de esta manera, los supervisados son usados para analizar cadenas de datos que tienen concordancia y similitud en un grupo de datos.

El aprendizaje no supervisado lo que realiza es tratar de analizar patrones que no sean reconocidos en los datos; la gran mayoría de veces, los datos son aproximaciones que el aprendizaje supervisado puede acertar, lo que hace que sean más usados en la aplicación de problemas en el mundo real.

El momento de utilizar el aprendizaje no supervisado es cuando no se tiene conocimiento de los datos que van a resultar. Trasladándolo al mundo real, es como vender un producto totalmente nuevo a un mercado incierto, donde la compañía no tiene ninguna experiencia; pero si lo que busca es comprender alguna información dentro de sus clientes existentes, el aprendizaje supervisado es la opción.

Las razones por las cuales se utiliza el aprendizaje no supervisado son las siguientes:

* Se encuentra con todo tipo de patrones desconocidos en los datos.
* Ayuda a encontrar datos útiles para la categorización.
* La facilidad para encontrar datos no etiquetados.

# EDA y tratamiento para datos

EDA, en inglés, “Exploratory Data Analysis”, hace referencia a una de las primeras cosas que tiene que realizar el científico de datos cuando revisa por primera vez un archivo en formato **.CSV**. Él debe saber comprender, descubrir posibles patrones y distribuciones estadísticas que puedan ser útiles más adelante, siempre y cuando se tenga un objetivo claro, que sepa lo que se quiere hacer con los datos, y por tanto comprender a dónde se quiere llegar con su interpretación.

EDA es una primera aproximación a los datos. Ahora bien, suponiendo que se tiene una muestra de datos suficiente, entonces, lo más seguro es que, con gran rapidez, se tenga una pregunta alusiva para poder dar inicio al proyecto. Resumiendo, lo que hace EDA es poder tener en una hora, o máximo un día, los datos para poder sacar conclusiones y poder decir al usuario si se prosigue o no con el proyecto. En caso de que el proyecto continúe, esto da más tiempo para poder determinar si se aplicarán modelos de “machine learning”.

### Trabajar con EDA

Hablando de programación y recordando que se está trabajando con Python, se encuentra con librerías denominadas Pandas, que ayudarán a manipular los datos, leerlos y transformarlos. Otra de las técnicas que le ayudará con Pandas es la visualización de los datos. Un ejemplo de ello se presenta en el PDF denominado **Anexo\_1\_Trabajar\_con\_EDA**, que usted podrá explorar desde la carpeta Anexos.

## Aprendizaje supervisado para regresiones

Cuando se habla de “*machine learning*” se hace referencia a una de las ciencias de la computación, como la inteligencia artificial, que dice que los algoritmos cada vez aprenden y se auto programan basándose en experiencias, tal como lo hacen los seres humanos.

Existen métodos de clasificación y regresión que hacen parte de una de las ramas del “*machine learning*”, más conocidos como el aprendizaje automático supervisado; estos, se organizan bajo cuatro algoritmos en los que se fundamenta el “machine learning”:

* “Machine Learning” supervisado.
* “Machine Learning” no supervisado.
* “Machine Learning” semisupervisado.
* “Machine Learning” por refuerzo.

El aprendizaje supervisado trabaja con datos que ya se encuentran etiquetados y que se conoce la respuesta de destino; con base en datos históricos, busca datos relacionados en campos especiales, denominados objetivo sujeto a datos de entrada o parámetros. A partir de estos históricos, el algoritmo aprende a etiquetar los datos de salida. Hoy en día, el aprendizaje supervisado es utilizado en dos campos: el de clasificación, como detección de fraude, y el de regresión, como el meteorológico.

Entonces, se entiende que las regresiones lineales hacen parte de los algoritmos de aprendizaje supervisado que se utilizan en las ciencias de datos, como “machine learning”, y en estadística.

## Aprendizaje supervisado para clasificación

Cuando se habla de aprendizaje supervisado, se hace referencia al entrenamiento que se realiza a los algoritmos con datos etiquetados con la respuesta correcta; cuantos más datos tenga el conjunto, más acertado y rápidamente aprende el algoritmo. Una vez concluido el entrenamiento, se le asignan nuevos datos, pero estos ya sin las etiquetas de las respuestas correctas. Este algoritmo utiliza la experiencia que obtuvo en el ejercicio anterior para apuntar o predecir un resultado.

Para construir un modelo de “machine learning”, no solamente se escoge un algoritmo de aprendizaje o una librería de “machine learning”; esto conlleva una serie de pasos que involucran lo siguiente:

1. Recolectar los datos
2. Procesar los datos
3. Explorar los datos
4. Entrenar el algoritmo
5. Evaluar el algoritmo
6. Utilizar el modelo

Las librerías que utilizan Python para “machine learning” son las mismas que se mencionaron al inicio de este documento, no es necesario volver a nombrarlas, pero sí se nombrarán los algoritmos más utilizados en la ciencia de datos:

* Regresión lineal.
* Regresión logística.
* Árboles de decisión.
* Random Forest.
* SVM (Máquinas de Vectores de Soporte).
* KNN
* K-Means

Estos algoritmos se pueden aplicar a la gran mayoría de problemas con datos, gracias a las librerías que tiene Python, como scikit-learn. Algunos ejemplos, se hallan en el PDF denominado **Anexo\_2\_Ejemplos\_de\_aprendizaje\_supervisado**, que se encuentra en la carpeta Anexos.

## Aprendizaje no supervisado y simulación

Mientras que en el aprendizaje supervisado se tienen salidas con etiquetas para el entrenamiento de los algoritmos, en el no supervisado no se sabe qué resultado esperar; de esa manera, entender el no supervisado se convierte en un ejercicio intuitivo, pues con este modelo es difícil saber si se ha realizado un trabajo exitoso o si aún es posible extraer datos de él.

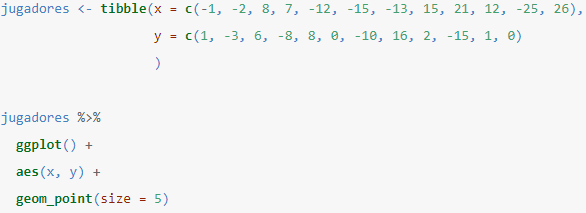
En el aprendizaje no supervisado, el reto es agrupar segmentos de datos, a eso se le llama “clustering” de datos, cuyo objetivo es encontrar grupos homogéneos de datos, ya que esta homogeneidad permite realizar análisis de comportamiento de estos grupos, para así poder sacar conclusiones y sacar posibles futuros datos. Estos algoritmos se basan en la distancia entre observaciones; los algoritmos más utilizados en el aprendizaje no supervisado son:

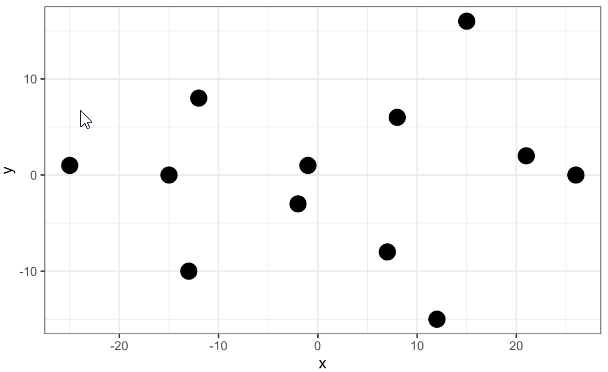
* Agrupamiento por K-medias
* Agrupamiento jerárquico

Por último, y para explicar este método, se desarrolla un ejercicio donde se utilizarán pocas variables, las cuales, poco a poco, se irán incrementando. Detalle el paso a paso de este ejercicio:

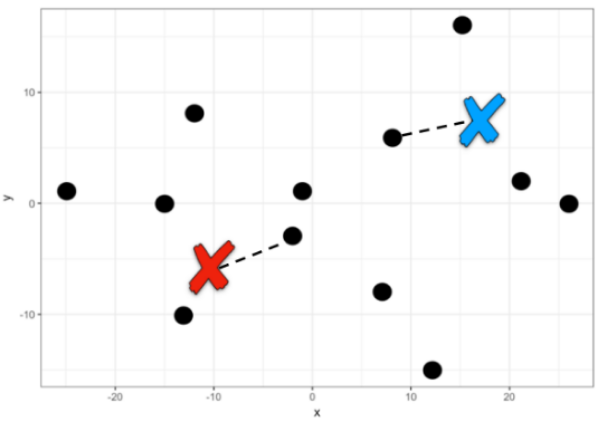
1. **Identificación de coordenadas**. Este ejercicio lo haremos con K = 2.

Supongamos, se tiene un grupo de jugadores de fútbol en un campo y se logra obtener una fotografía desde la parte de arriba para poder identificar sus coordenadas (la variable 1 sería el eje X y variable 2 sería el eje Y). Por la altura, no se alcanza a distinguir de qué equipo es cada uno de los jugadores, por ese motivo, se pintarán todos de negro.





1. **Ubicación de centroides**. El modelo permite agrupar, a partir de la identificación de centroides. Se puede ubicar la cantidad de centroides como grupos, lo cuales se identificarán así para este ejemplo. Se sabe que hay dos equipos, entonces se utilizarán 2 centroides, el algoritmo k-medias colocará estos dos puntos centroides de forma aleatoria en el plano; se procede a calcular la distancia entre cada centro y los demás puntos del plano, y si está más cerca de un centroide, lo asigna al “uno”, de lo contrario, lo asigna al centroide “dos”.

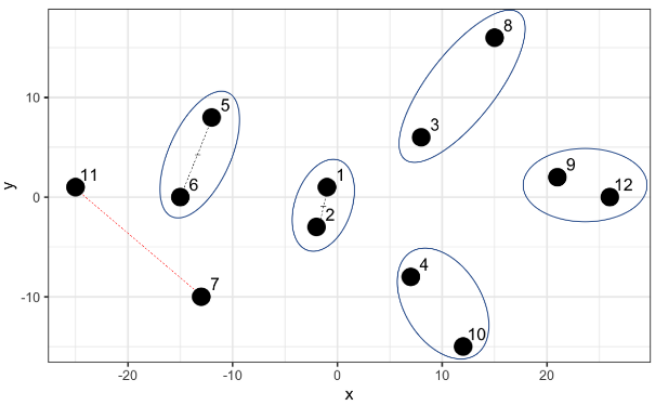


1. **Agrupamiento jerárquico**. Por otro lado, el agrupamiento jerárquico hace referencia a otro método de agrupamiento de datos; a diferencia del k-medias, esta muestra la cantidad de datos de combinaciones posibles de acuerdo con la jerarquía de las distancias entre puntos.
2. **Graficación del algoritmo**. Ahora bien, se utilizará el mismo ejemplo anterior del equipo de fútbol, con la diferencia de que esta vez se enumerará cada jugador para obtener una mejor visualización.

Su funcionamiento consiste en buscar los puntos con las distancias más cortas y agruparlos, y buscar otros dos puntos con la menor distancia y realizar la pregunta: ¿es la distancia entre estos dos grupos más corta que la distancia de los grupos creados con anterioridad?

Si la respuesta es “SÍ”, los agrupará, de lo contrario, agrega los puntos al grupo ya creado con anterioridad.

El algoritmo, gráficamente, muestra que los puntos 1 y 2 tienen la jerarquía más baja, puesto que se tiene la distancia más corta. El algoritmo sigue su búsqueda para los puntos más cercanos (el 9 y el 12) y realiza la comparación con el punto medio del 1 y el 2, por lo que se decide la creación de un nuevo grupo con una jerarquía más alta, y así lo hace sucesivamente.

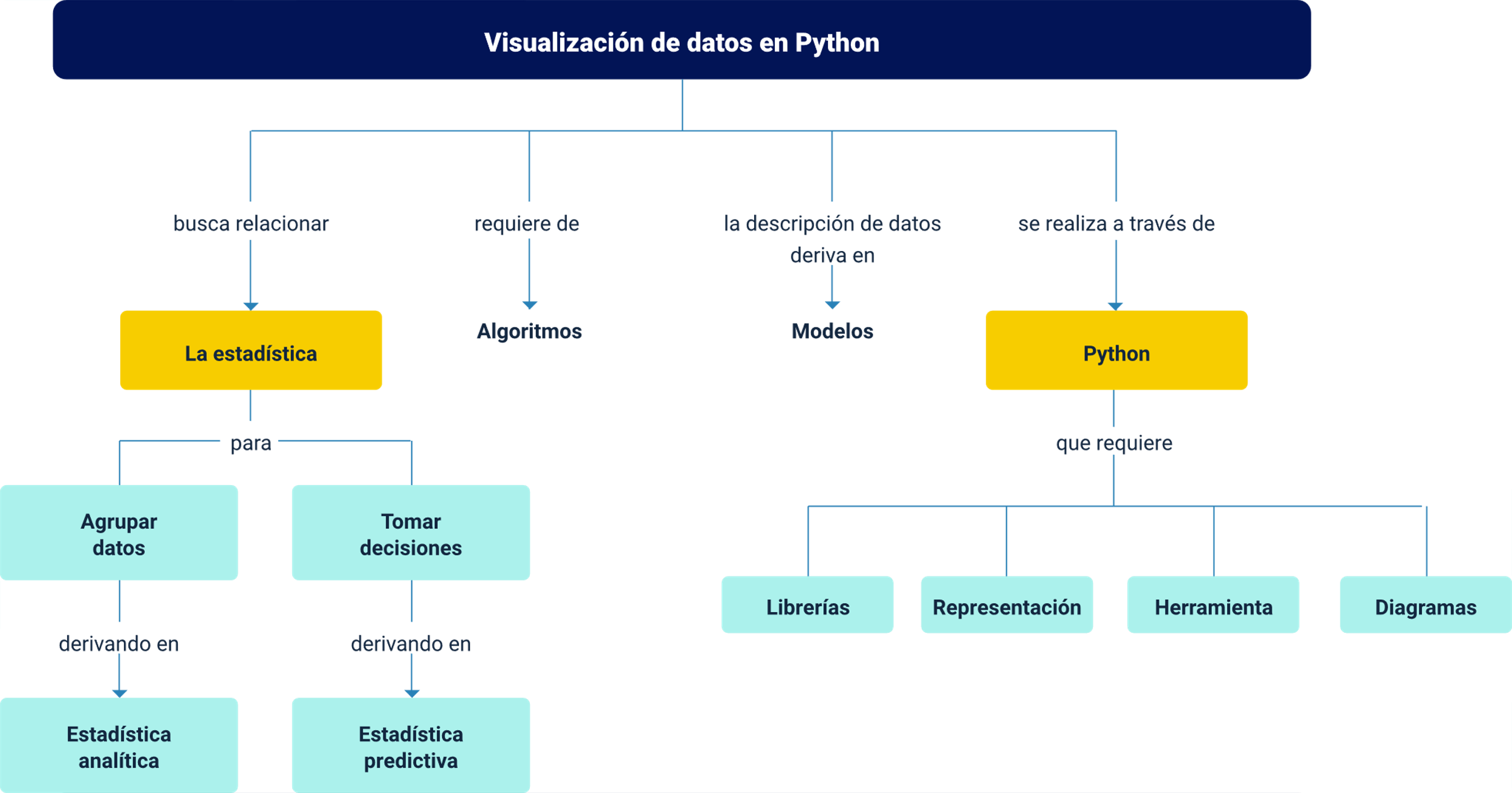


Síntesis

La visualización de datos es muy importante hoy en día, ya que, gracias a estas proyecciones, se tienen en cuenta muchos análisis predictivos para la correcta toma de decisiones. Esta necesidad de visualización, gracias a su gran importancia en el mundo informático, ha creado herramientas que ayudan a los desarrolladores a obtener un número considerable de estos datos que circulan, para poder armar patrones y generar grupos de datos que conlleven tendencias de fácil identificación.

Estas herramientas son muy prácticas e intuitivas para que las empresas puedan sustraer los resultados y generar objetivos específicos, que dirijan a un grupo de científicos de datos a orientar los caminos o direcciones a que se pretende llegar con estos análisis detallados. La ciencia de datos parte de un principio que dice “una imagen vale más que mil palabras”.

A continuación, se presenta una síntesis gráfica (mapa conceptual) de lo visto en el componente y esos aspectos claves que se debe llevar:



Material complementario

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia | Tipo de material | Enlace del recurso |
| 1. Python y Colab | Excibit Corp. (2021). Mejores Prácticas de Visualización de Datos. | Video | <https://www.youtube.com/watch?v=83ftOkX0jD8> |
| 1.1 Listas y estructuras | BitBoss. (2021). Estructuras de datos con Python en 8 minutos: Listas, Tuplas, Conjuntos y Diccionarios. | Video | <https://www.youtube.com/watch?v=v25-m1LOUiU> |
| 1.5 Visualización y analytics | Camacho, E. (2020). 1 - Gráficas simples - Visualización de datos en Python con Matplotlib. | Video | <https://www.youtube.com/watch?v=UbrSH3aJt1w&t=6s> |

Glosario

**Algoritmo**: son los algoritmos que mediante una serie de datos identifican distintas etiquetas. Por ejemplo, se pueden hacer dos grupos y diferenciar clientes que están cercanos a la compra en una página web de clientes que no lo están. Clientes muy rentables, poco rentables y no rentables.

**Centroides**: es la ubicación real o imaginaria que representa el centro del grupo. Cada punto de datos se asigna a cada uno de los grupos mediante la reducción de la suma de cuadrados en el grupo.

**“Dataset”**: no es más que un conjunto de datos tabulados en cualquier sistema de almacenamiento de datos estructurados.

**EDA**: el análisis exploratorio de datos (EDA, por sus siglas en inglés) implica el uso de gráficos y visualizaciones para explorar y analizar un conjunto de datos. El objetivo es explorar, investigar y aprender, no confirmar hipótesis estadísticas.

**Google Colaboratory**: colaboratory, o "Colab" para abreviar, es un producto de Google Research. Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación.

**IPython**: es un “Shell” interactivo que añade funcionalidades extra al modo interactivo incluido con Python, como resaltado de líneas y errores mediante colores, una sintaxis adicional para el “Shell”, autocompletado mediante tabulador de variables, módulos y atributos.

**Jupyter**: jupyter Notebook es una aplicación cliente-servidor lanzada en 2015 por la organización sin ánimo de lucro Proyecto Jupyter. Permite crear y compartir documentos web en formato JSON que siguen un esquema versionado y una lista ordenada de celdas de entrada y de salida. Estas celdas albergan, entre otras cosas, código, texto (en formato Markdown), fórmulas matemáticas y ecuaciones, o también contenido multimedia (“Rich Media”). El programa se ejecuta desde la aplicación web cliente que funciona en cualquier navegador estándar.

**“Machine Learning”:** es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita. Sin embargo, “machine learning” no es un proceso sencillo. Conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos. Un modelo de “machine learning” es la salida de información que se genera cuando entrena su algoritmo de “machine learning” con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida.

Referencias bibliográficas

Aprende IA. (2020). ¿Qué es el Aprendizaje no Supervisado?. <https://aprendeia.com/aprendizaje-no-supervisado-machine-learning/>

Bagnato, J. (2019). Análisis Exploratorio de Datos con Pandas en Python. Aprende Machine Learning. <https://www.aprendemachinelearning.com/analisis-exploratorio-de-datos-pandas-python/>

De la Fuente, Ó. (2019). Google Colab: Python y Machine Learning en la nube. Adictos al trabajo. <https://www.adictosaltrabajo.com/2019/06/04/google-colab-python-y-machine-learning-en-la-nube/>

Kumar, A. (2020). Tensor Explained with Python NumPy Examples. Data Analytics.

López, R. (2015). Probabilidad y Estadística con Python. <https://relopezbriega.github.io/blog/2015/06/27/probabilidad-y-estadistica-con-python/>

Mariños, J. (2015). Listas en Python. DevCode. <https://devcode.la/tutoriales/listas-python/>

NumPy Developers (2022). NumPy documentation. NumPy. <https://numpy.org/doc/stable/>

Python Software Foundation. (s. f.). 5. Estructuras de datos. Python. <https://docs.python.org/es/3/tutorial/datastructures.html#more-on-lists>

Rodríguez, D. (2020). Formatos condicionales en Pandas. Analytics Lane. <https://www.analyticslane.com/2020/04/17/formatos-condicionales-en-pandas/>

Tableau Software. (2019). Guía de visualización de datos para principiantes: definición, ejemplos y recursos de aprendizaje. Salesforce. <https://www.tableau.com/es-mx/learn/articles/data-visualization>

The Carpentries. (s. f.). Análisis y visualización de datos usando Python. datacarpentry.org. <https://datacarpentry.org/python-ecology-lesson-es/07-visualization-ggplot-python.html>

Créditos

| Nombre | Cargo | Regional y Centro de Formación |
| --- | --- | --- |
| Claudia Patricia Aristizábal | Responsable del Ecosistema | Dirección General |
| Rafael Neftalí Lizcano Reyes | Responsable de Línea de Producción | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| José Luis Bastidas Pérez | Experto Temático | Regional Cauca - Centro de Teleinformática y Producción Industrial |
| Zvi Daniel Grosman Landáez | Diseñador Instruccional | Regional Distrito Capital - Centro de Gestión Industrial |
| Andrés Felipe Velandia Espitia | Asesor Metodológico | Regional Distrito Capital - Centro de Diseño y Metrología |
| Miroslava González Hernández | Diseñadora Instruccional | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Darío González | Corrector de estilo | Regional Distrito Capital – Centro de Diseño y Metrología |
| Juan Daniel Polanco | Diseñador de Contenidos Digitales | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Francisco José Lizcano Reyes | Desarrollador “Fullstack” | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Carmen Alicia Martínez Torres | Animador y Producción Audiovisual | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Emilsen Alfonso Bautista | Actividad Didáctica | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Zuleidy María Ruiz Torres | Validador de Recursos Educativos Digitales | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Luis Gabriel Urueta Álvarez | Validador de Recursos Educativos Digitales | Regional Santander - Centro Industrial del Diseño y la Manufactura |
| Daniel Ricardo Mutis Gómez | Evaluador para contenidos inclusivos y accesibles | Regional Santander - Centro Industrial del Diseño y la Manufactura |