**Clase 5: Pandas**

Pandas es una **colección de estructuras de datos y funciones** que facilitan el trabajo con **datos estructurados**. **Pan**el **Da**ta **S**ystem. Está **construido en base a Numpy.** Permite trabajar con **datos de diferentes tipos**. Se pueden identificar las filas y columnas con **etiquetas** en lugar de sólo números. Mix de **alta performance de cálculo numérico de Numpy + flexibilidad en manipulación de datos**. Tiene **funcionalizades de indexación avanzadas**.  **Operaciones de agrupación por columnas, filtros y resúmenes.** Es muy útil para la **limpieza y procesamiento de datos.**

**Estructuras de datos:**

* **Series:** Objeto similar a un vector unidimensional. Contiene un **array de valores** y un **array asociado de etiquetas** (índice). En caso de no especificarle índice, toma por default los valores desde 0 hasta N-1, siendp N la cantidad de valores de la serie. Todos los valores de una serie deben ser del mismo tipo.

Similar a un diccionario de tamaño fijo y con sus claves ordenadas. Al igual que Numpy, permite pasar una lista con valores de índices para seleccionar un subconjunto de valores.

A partir de una lista: Data = pd.Series(Nombredelista).

A partir de un escalar: Data = pd.Series(5, index = [100, 200, 300].

A partir de un diccionario: Data = pd.Series({2:’a’, 1:’b’, 3:’c’})

Data.values da los valores del array.

Data.index da los valores del índice (etiquetas de cada valor).

Data[#]; Data[1:3]. Se pueden obtener valores de la serie en forma análoga a Numpy.

El índice explícito de la Serie puede tener valores repetidos en diferentes posiciones.

Data.iloc[#] sirve para dar como input la posición.

Data.loc[etiqueta] sirve para dar como input la etiqueta.

‘b’ in Data. Devuelve True si Data tiene un valor indizado con ‘b’

List(data.items()) devuelve una lista de tuplas con los índices y valores en cada tupla de la misma.

Una vez definidos los Index, no pueden modificarse.

Es posible utilizar operciones sobre conjuntos con las instancias de Index:

IndA & IndB devuelve aquellos valores de índices que tienen en común IndA e IndB (Intersección).

IndA | IndB devuelve aquellos valores de índices que están en IndA + los que están en IndB, sin repetirlos (Unión).

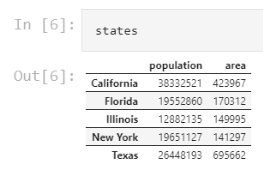
IndA ˆ IndB devuelve aquellos valores de índices que están en IndA o en IndB pero no están en ambos (Diferencia Simétrica).

IndA - IndB devuelve aquellos valores de índices que están en IndA pero no están en IndB (Diferencia).

* **DataFrame:** **Estructura de datos tabular**; contiene una **colección de columnas**, cada una con un tipo de datos determinado.Cuenta con **índice de filas y de columnas.** Sirve muchísimo para la Limpieza de Datos.

Una vez definidos los Index, no pueden modificarse.

Ejemplo práctico: Genera una serie de áreas de estados de USA a partir de un diccionario. Área = pd.Series(área\_dict). Esto hace que automáticamente tome los key values como index y los valores del diccionario como valores. Luego, con dos listas, crea otra serie, pero esta vez de poblaciones de dichos estados: population = pd.Series (lista\_valores\_población, index = Lista\_de\_estados). Por último, genera un DataFrame a partir de estas dos series: states = pd.DataFrame(‘population’ = population, ‘area’ = Área).



States.Index da los nombres con los cuales están indizadas las filas (California, Florida,…,Texas).

States.Index[2] da el valor del índice en la 3era fila (‘Illinois’).

States.Columns da los nombres con los cuales están indizadas las columnas (population, área).

Es posible cambiar el nombre de los index y columnas:

States.Index = [Nuevonombre1, Nuevonombre2, …, Nuevonombren]

States.Column = [Nuevonombre1, Nuevonombre2, …, Nuevonombren]

States.head(2): Muestra los primeros 2 elementos del DataFrame.

States.tail(3): Muestra los últimos 3 elementos del DataFrame.

States.shape da la forma que tiene el DataFrame. Sirve para conocer cuántas filas y columnas tiene.

States.sample(2): Muestra 2 elementos aleatorios del DataFrame.

States[‘área’] o states.area selecciona sólo la columna área del DataFrame.

States[‘área’].dtype da el tipo de datos que tiene la columna ‘area’.

Data[‘nivel educativo’].value\_counts() Si tengo una columna de strings, me agrupa los strings y me indica cuántas filas hay para cada valor único de string (es como una tabla dinámica que cuenta valores).

Data.sort\_values(by=’edad’, ascending = False) Ordena las filas de un DataFrame por la columna ‘edad’ en orden Descendente.

States.iloc[:3, :1] llama todas las filas desde el principio hasta la 4ta no inclusive y la primer columna. (California a Illinois, sólo la población). En este caso está usando los **índices implícitos**, análogos a los que usa Numpy.

States.loc[:’Illinois’, :’population’]. llama todas las filas desde el principio hasta Illinois inclusive y las columnas desde el principio hasta ‘population’ inclusive. (California a Illinois, sólo la población). En este caso está usando los **índices explícitos**. Los resultados de estas dos sentencias son los mismos, pero se puede hacer la consulta de las dos maneras.

También es posible **indexar un DataFrame usando loc y máscaras booleanas**:

States.loc[states.area > 423000, [‘population’,’area’]]

En este caso llamará aquellas filas con área mayor que 423000; y mostrará las columnas population y área.

Se pueden modificar valores de un DataFrame:

States[‘density’] = states[‘population’]/states[‘area’]

Data.iloc[0,2] = 90 En este caso reemplaza el valor del DataFrame en La primer fila y 3er columna por 90.

Es posible utilizar operciones sobre conjuntos con las instancias de Index:

IndA & IndB devuelve aquellos valores de índices que tienen en común IndA e IndB (Intersección).

IndA | IndB devuelve aquellos valores de índices que están en IndA + los que están en IndB, sin repetirlos (Unión).

IndA ˆ IndB devuelve aquellos valores de índices que están en IndA o en IndB pero no están en ambos (Diferencia Simétrica).

IndA - IndB devuelve aquellos valores de índices que están en IndA pero no están en IndB (Diferencia).

**Importación de un archivo en Pandas**:

data\_location = "../Data/data\_filt.csv"

data = pd.read\_csv(data\_location, sep=",", encoding="latin1")