



TALENTOFUTURO

Clase 7: Estructuras de Datos Avanzadas

Módulo 2: Programación Avanzada en Python.

Apoyado por:

CORFO

Clase de hoy

01

Dataframes

Creación y entendimiento de dataframes.

02

Archivos y dataframes

Carga de archivos de datos en dataframes.

03

Visualización

Creación de gráficos para visualizar datos.



7A

Dataframes

Bloque A

Qué veremos en Bloque A

- Definición de Dataframe.
- Uso de Dataframes.
- Carga de archivos con Dataframes.

Dataframes (Pandas)

¿Qué es un Dataframe?

- Una “dataframe” es un arreglo bidimensional que permite aplicar operaciones sobre los datos que se encuentran almacenados en el arreglo.
- La biblioteca Pandas es una implementación Python, open-source, de una serie de operaciones habilitantes para el análisis y manejo de datos.
- Pandas define un “dataframe” como un arreglo de dos dimensiones, de tamaño mutable, y de datos potencialmente heterogéneos.
- El concepto de arreglo es usado por muchas operaciones en la matemática y en la programación, dando sentido semántico a muchos tipos de caracterizaciones: por ejemplo, un arreglo puede representar un grafo y sus conexiones.
- En el caso de un “dataframe”, el arreglo se interpreta como datos en forma tabular.

Dataframes (Pandas)

Dataframes en Pandas

- Un dataframe en Pandas es, además, una clase.
- Como tal, la clase debe ser instanciada para crear un objeto de tipo dataframe.
- Típicamente instanciaremos el objeto con los datos, sumado a algunos atributos extras que caracterizan los tipos de datos cargados (por ejemplo, si la tabla contiene una fila con etiquetas o no).
- Lo primero que debemos hacer es importar la biblioteca:

```
import pandas as pd
```

- Notar que comenzaremos a usar un alias (“as pd”).

Dataframes (Pandas)

Creando un dataframe desde un diccionario

- Si contamos con un diccionario, podemos cargarlo a un dataframe rápidamente:

```
import pandas as pd
```

```
datos = {  
    "nombres": ["Gabriela", "Juan", "Daniela", "Alberto"],  
    "edades": [70, 75, 78, 82]  
}
```

```
personas = pd.DataFrame(datos)
```

	nombres	edades
0	Gabriela	70
1	Juan	75
2	Daniela	78
3	Alberto	82

Índice

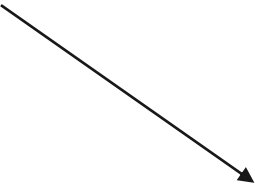
Dataframes (Pandas)

Agregando columnas

- Si ya tenemos un dataframe creado, podemos agregar columnas:

```
# al dataframe anterior agregamos columna con rut
```

```
personas["rut"] = ["44444444", "4500000", "3455888", "5444355"]
```



	nombres	edades	rut
0	Gabriela	70	44444444
1	Juan	75	4500000
2	Daniela	78	3455888
3	Alberto	82	5444355

Dataframes (Pandas)

Obtener filas

- Podemos obtener los valores de una fila haciendo referencia a la fila con el operador loc:

```
personas.loc[0] →
```

nombres	Gabriela
edades	70
rut	4444444
Name: 0, dtype: object	

- Podemos especificar un listado de filas con el mismo operador:

```
personas.loc[[0, 2]] →
```

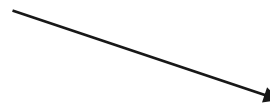
	nombres	edades	rut
0	Gabriela	70	4444444
2	Daniela	78	3455888

Dataframes (Pandas)

Obtener filas en base a una condición

- Si estamos almacenando datos comparables con algún valor, tiene sentido pensar en obtener filas cumpliendo alguna condición:

```
# obtener personas con edad mayor a 75  
personas[personas["edades"] > 75]
```



	nombres	edades	rut
2	Daniela	78	3455888
3	Alberto	82	5444355

- Otros operadores disponibles: '>', '<', '<=', '>=', '!=', '=='

Dataframes (Pandas)

Descripción general de los datos

- Cuando estamos trabajando con dataframes grandes, es más complicado tener una idea general de los datos con los que estamos trabajando.
- Podemos obtener una visión general con la operación head:

```
personas.head()
```

- Por defecto, la operación head retorna las primeras 5 filas.
- Si la cantidad de columnas es muy grande, Pandas (por defecto) omitirá algunas de ellas. Podemos modificar el valor de la siguiente variable global:

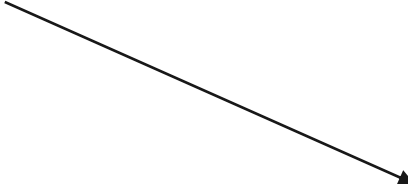
```
pd.set_option('display.max_columns', None)
```

Dataframes (Pandas)

Descripción general de los datos

- También podemos obtener una descripción de los tipos de datos con la operación `info`:

```
personas.info()
```



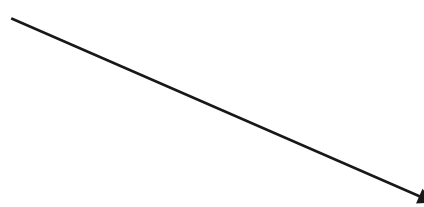
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0  nombres    4 non-null      object
 1  edades     4 non-null      int64
 2  rut        4 non-null      object
dtypes: int64(1), object(2)
memory usage: 228.0+ bytes
```

Dataframes (Pandas)

Descripción general de los datos

- Finalmente, también podemos obtener una descripción estadística general de aquellos tipos de datos que son numéricos:

```
personas.describe()
```



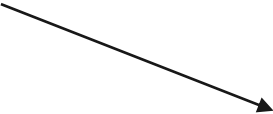
	edades
count	4.000000
mean	76.250000
std	5.057997
min	70.000000
25%	73.750000
50%	76.500000
75%	79.000000
max	82.000000

Dataframes (Pandas)

Nueva columna en base a datos de otras columnas

- Además de agregar columnas en forma estática como ya vimos, también podemos agregar columnas cuyos datos son derivados desde otras columnas.
- Una manera muy sencilla es especificar una operación entre dos columnas:
- Ejemplo:

```
personas["rut_estandar"] = "0" + personas["rut"]
```



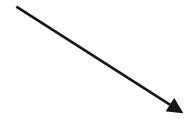
	nombres	edades	rut	rut_estandar
0	Gabriela	70	4444444	04444444
1	Juan	75	4500000	04500000
2	Daniela	78	3455888	03455888
3	Alberto	82	5444355	05444355

Dataframes (Pandas)

Nueva columna en base a datos de otras columnas

- Y ¡qué mejor!... ¡También podemos aplicar una lambda para crear una nueva columna!
- Ejemplo:

```
personas["categoria"] = personas.apply(  
    lambda x: "mayor de 75" if  
    (x["edades"]>75) else  
    "menor de 75", axis=1  
)
```



	nombres	edades	rut	categoria
0	Gabriela	70	4444444	menor de 75
1	Juan	75	4500000	menor de 75
2	Daniela	78	3455888	mayor de 75
3	Alberto	82	5444355	mayor de 75



Trabajo grupal – Bloque A

Paralelo 2

G1	G2	G3	G4
Nicolas Mardones	Víctor Meza Herrera	Daniela Méndez Gándara	Claudia Blanco
Manuel Denis	Estefania Manriquez	Álvaro Pérez	Ariel Inostroza
Bryan Castillo	Patricio Vera	Pedro Nahum	Héctor Aguayo
GERALDY SUAREZ	Oscar Torres	Javier Gajardo	Ruben Sanhueza Ramirez
Scarlett Espinoza	Braulio Quiroz	Angela Proboste Neira	Félix González
Ulises Campodónico	Yerko Gallardo	Nicolás Guzmán	Ariel Mora
Carol Leiva	Rodrigo Araya		
G5	G6	G7	G8
Fabian Díaz	Camila Oyarzún	Mayerlyn Rodriguez	Daniela Porto
Natalia Rivera	Stefanya Pulgar	Sebastian Vega	Cristian Chavez Jara
Juan Salinas	Carlos Emilio Azócar Riquelme	Efrain Duarte Campos	Juan Rodrigo Vega
Rodrigo Pastén Cortés	Nicolas Rojas	Bianel Bianchini	Rodolfo Cantillana
Flavio Jara R.	luis.paillan.cnc@gmail.com	Bastián Gamboa Labbé	Abraham Ruiz
Daniel García	Cristóbal Gajardo	Pablo Uribe	Rodrigo Álvarez

Trabajo grupal – Ejercicio #1

Creando un dataframe

- Instale la biblioteca pandas e impórtela en su proyecto Python.
- Cree un dataframe con un diccionario, tal como se vio en los ejemplos, con las siguientes columnas, y con 5 datos:
 - Nombre
 - Rut
 - Edad
 - Altura
 - Peso
- Describa y discuta los datos almacenados en el dataframe (almacene los ruts como strings).

Trabajo grupal – Ejercicio #2

Creando columnas en un dataframe

- Cree las siguientes columnas:
 - Rut estandarizado
 - Dígito verificador
 - Índice de masa corporal (IMC)
 - Clasificación de edad (mayor de 30 años, menor de 30 años, mayor de 60 años).

A photograph of three women in a professional setting, overlaid with a magenta tint. The woman on the left has dark hair and is looking down. The woman in the center has short blonde hair and is smiling while looking at a laptop. The woman on the right has dark hair and is also smiling. They are sitting at a desk with a laptop and some papers. A large blue rectangle with the word 'Break!' in white text is positioned in the lower-left foreground.

Break!



7B

Carga de archivos y gráficos

Bloque B

Qué veremos en Bloque B

- Cargar archivos a un dataframe.
- Crear algunos gráficos básicos.

Dataframes (Pandas)

Carga de archivos en un dataframe

- Hasta ahora hemos revisado conceptualmente los aspectos básicos de un dataframe.
- En la práctica, un dataframe tiene su mayor utilidad en el procesamiento de grandes cantidades de datos (con limitaciones dadas por la capacidad de la máquina).
- Es aquí donde la carga de archivos tiene su mayor utilidad: poder analizar y procesar datos que provienen de alguna fuente que los genere.
- Casi todas las aplicaciones pueden volcar datos en formatos como CSV (comma-separated values) o XLSX (Microsoft Excel).

Dataframes (Pandas)

Carga de un archivo CSV

- La forma más básica de cargar un archivo CSV es usando:

```
dataframe = pd.read_csv('data.csv')
```

- Por defecto, la operación `read_csv` intentará inferir la cabecera del archivo.
- De encontrar nombres, procederá a usar estos como identificadores para las columnas.
- Un parámetro interesante de mencionar es “sep”, que se puede usar para indicar un separador específico de los datos. Por defecto se tiene:

```
sep = “,”
```


Dataframes (Pandas)

Carga de un archivo XLSX

- La forma más básica de cargar un archivo XLSX es usando:

```
dataframe = pd.read_excel('tmp.xlsx')
```

- Por defecto, la operación `read_excel` trabaja con un `header = 0`, es decir, asume que la fila 0 contiene la cabecera del archivo.
- Con `header = None` se descarta la cabecera y todos los datos se consideran parte del dataframe.
- En un archivo Excel, típicamente tenemos “hojas”. Estas se pueden especificar con el parámetro `sheet_name = “nombre de hoja”`.

Gráficos

Visualizando datos

- Aunque no es una parte de Pandas propiamente tal, la visualización de datos siempre va de la mano con los dataframes.
- La explicación de esto viene desde la práctica: si estamos usando dataframes, es porque estamos analizando datos.
- El análisis de datos siempre va acompañado de una debida visualización de los datos.
- En esta clase revisaremos cómo crear dos gráficos usando la biblioteca seaborn que está construida sobre una biblioteca conocida como matplotlib.

Gráficos

Visualizando datos con seaborn

- Para trabajar con seaborn necesitamos importar la biblioteca. Esto se realiza utilizando típicamente el alias sns:

```
import seaborn as sns
```

- Como siempre, debemos instalar la biblioteca seaborn.
- Dado que seaborn está construida sobre una biblioteca llamada matplotlib, en ocasiones conviene utilizar esta biblioteca para manejar el canvas (i.e., donde el gráfico se despliega).

```
import matplotlib.pyplot as plt
```

Gráficos

Visualizando datos con seaborn

- El primer gráfico que veremos es el gráfico de barras. Este gráfico nos permite observar la frecuencia absoluta o relativa de elementos que cumplen con cierta condición
- Usaremos un archivo llamado `cvd_cleaned` (cardiovascular disease) que resume el estado de salud cardiovascular de personas atendidas en un hospital de EE.UU.
- Debemos importar pandas, seaborn y matplotlib y cargar el archivo:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
cvd_df = pd.read_csv("cvd_cleaned.csv")
```

Gráficos

Visualizando datos con seaborn

- Luego procedemos a explorar rápidamente los datos:

```
pd.set_option('display.max_columns', None)
print(cvd_df.head())
print(cvd_df.info())
```

Gráficos

Visualizando datos con seaborn

- Luego procedemos a preparar nuestro dataset para graficar:

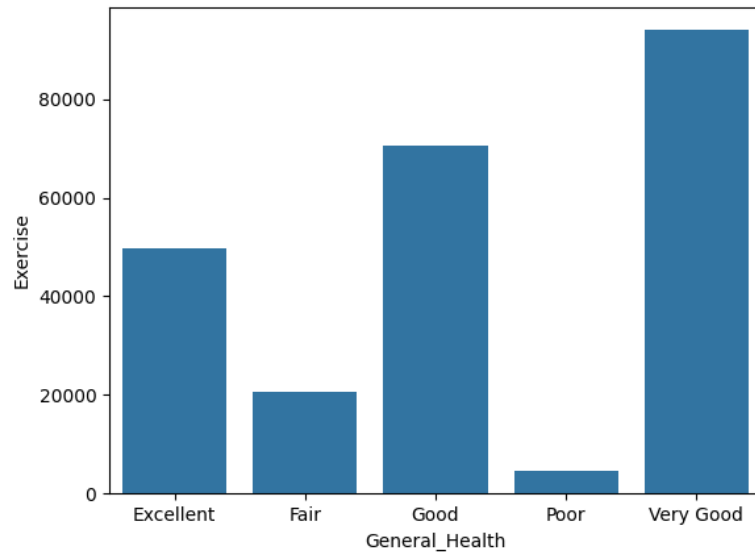
```
hacen_ejercicio = (cvd_df[cvd_df["Exercise"]=="Yes"])  
                  [["General_Health", "Exercise"]].  
                  groupby(by=["General_Health"]).  
                  count().reset_index()
```

- Y luego procedemos a graficar (en eje x dejamos las categorías y en eje y dejamos la cuenta de personas que realizan ejercicio):

```
sns.barplot(data=hacen_ejercicio, x="General_Health", y="Exercise")  
  
plt.show()
```

Gráficos

Visualizando datos con seaborn



Gráficos

Visualizando datos con seaborn

- El segundo gráfico que veremos se conoce como boxplot o diagrama de cajas.
- También se le conoce como diagrama de bigotes por la forma en que se expresa.
- Usando el mismo archivo, queremos explorar la distribución de los datos del índice de masa corporal según estado general de salud.
- Teniendo lo anterior listo, comenzamos a graficar:

```
sns.boxplot(data=cvd_df, x="BMI", y="General_Health")  
plt.show()
```


Gráficos

Visualizando datos con seaborn

- Introduciremos algunas opciones para mejorar nuestros gráficos:

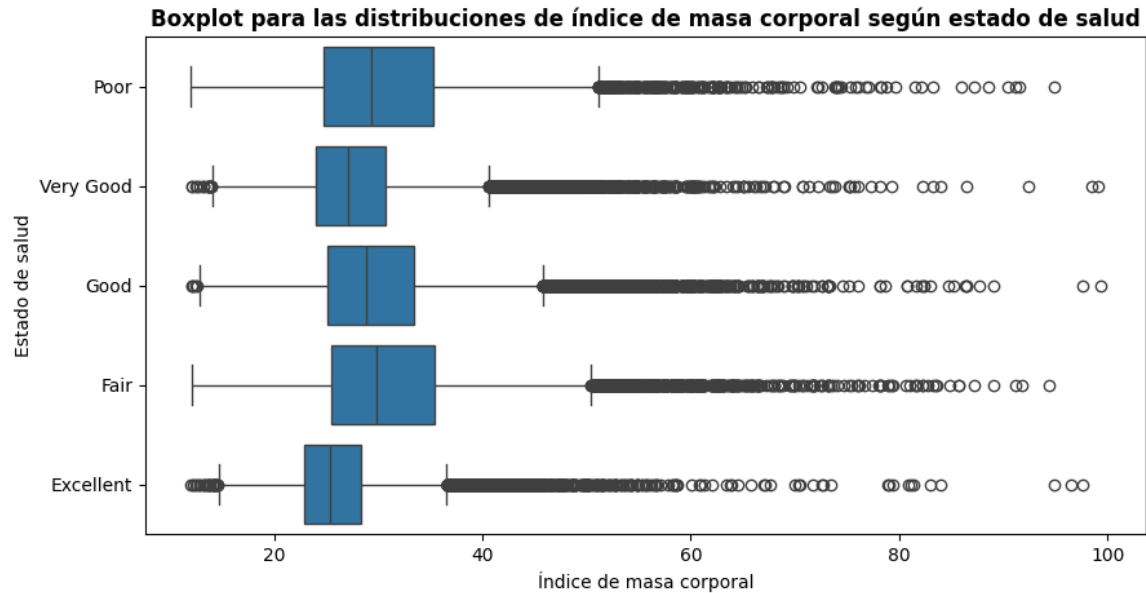
```
figure = plt.figure(figsize=(10,5))
sns.boxplot(data=cvd_df, x="BMI", y="General_Health")

plt.xlabel("Índice de masa corporal")
plt.ylabel("Estado de salud")
plt.title("Boxplot para las distribuciones de índice de masa
corporal según estado de salud", fontweight='bold')

plt.show()
```

Gráficos

Visualizando datos con seaborn





Trabajo grupal – Bloque B

Paralelo 2

G1	G2	G3	G4
Nicolas Mardones	Víctor Meza Herrera	Daniela Méndez Gándara	Claudia Blanco
Manuel Denis	Estefania Manriquez	Álvaro Pérez	Ariel Inostroza
Bryan Castillo	Patricio Vera	Pedro Nahum	Héctor Aguayo
GERALDY SUAREZ	Oscar Torres	Javier Gajardo	Ruben Sanhueza Ramirez
Scarlett Espinoza	Braulio Quiroz	Angela Proboste Neira	Félix González
Ulises Campodónico	Yerko Gallardo	Nicolás Guzmán	Ariel Mora
Carol Leiva	Rodrigo Araya		
G5	G6	G7	G8
Fabian Díaz	Camila Oyarzún	Mayerlyn Rodriguez	Daniela Porto
Natalia Rivera	Stefanya Pulgar	Sebastian Vega	Cristian Chavez Jara
Juan Salinas	Carlos Emilio Azócar Riquelme	Efrain Duarte Campos	Juan Rodrigo Vega
Rodrigo Pastén Cortés	Nicolas Rojas	Bianel Bianchini	Rodolfo Cantillana
Flavio Jara R.	luis.paillan.cnc@gmail.com	Bastián Gamboa Labbé	Abraham Ruiz
Daniel García	Cristóbal Gajardo	Pablo Uribe	Rodrigo Álvarez

Trabajo grupal – Ejercicio #1

Cargando un archivo a un dataframe

- Cargue el archivo titanic.csv en un dataframe.
- Describa y discuta los metadatos del dataframe.
- Lo que usted está realizando ahora es un análisis preliminar y exploratorio de los datos, es decir, entendiendo los tipos de datos, la relación entre estos, y, muy importante, el significado de estos.

Trabajo grupal – Ejercicio #2

Creando gráficos

- Discuta con su equipo qué aspectos son visualizables con barras y boxplots.
- Cree dos gráficos, haciendo uso de todas las herramientas vistas hasta ahora.
- Sus gráficos deben quedar "presentables", es decir, de una manera adecuada de tal forma que puedan ser incorporados en alguna presentación ejecutiva.

¿Preguntas?

¡Hemos llegado al final de la clase!

