UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
Centro de e-Learning

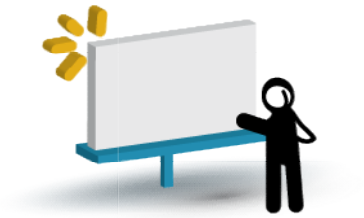
UNIDAD DIDÁCTICA II

MACHINE LEARNING

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo I – Bases y herramientas

Unidad II – Adquisición de datos.



Presentación:

Como primer paso en nuestro trabajo, tenemos que poder tomar los datos provenientes de diferentes formatos, para esto haremos uso de la librería pandas, la cual es una biblioteca de código abierto con licencia BSD que proporciona estructuras de datos de alto rendimiento y fáciles de usar, y herramientas de análisis de datos para el lenguaje de programación Python. pandas es un proyecto patrocinado por NumFOCUS.

En el transcurso de esta unidad veremos cómo realizar y formatear la información proveniente de diversas fuentes de uso cotidiano.

Objetivos:

Que los participantes:

Puedan obtener los datos de diferentes fuentes de procedencias.

Bloques temáticos:

1.- Tomar datos con pandas.

Formato CSV.

Desde Excel.

Formato json.

Formato html.

Base de datos – SQLite3

XML

Consignas para el aprendizaje colaborativo

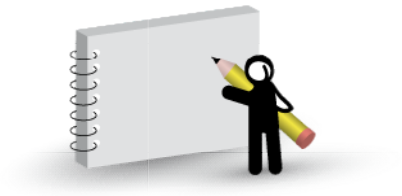
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Tomar datos con pandas

Importar datos con pandas es muy simple, a continuación analizaremos diferentes fuentes de procedencia:

Formato CSV

Para tomar datos de un archivo csv, podemos partir de una hoja de Excel que posea una tabla como la siguiente:

id	Nombre	Sexo
1	Juan	m
2	Pedro	m
3	Anna	f
4	Julieta	f
5	Pablo	m
6	Ramón	m
7	Gaby	f

Es necesario que dejemos solo una hoja en el libro y lo guardemos con formato '**csv delimitado por comas**'

Nota: Cuando tenemos configurada la maquina en español la separación entre datos es tomada por punto y como (;) en lugar de coma (,). En este caso podemos utilizar algún editor de texto como notepadd++ y sustituir todos los punto y comas (;) por comas (,).

Si ahora abrimos una hoja de jupyter podemos importar el archivo mediante el método read_csv de pandas.

```
import pandas as pd
dat_csv = pd.read_csv('empleados.csv', encoding = "ISO-8859-1")
dat_csv
```

La salida nos retorna:

Out[8]:

	id	Nombre	Sexo	Sueldo
0	1	Juan	m	10000
1	2	Ramón	m	20000
2	3	Anna	f	10500
3	4	Julieta	f	13000
4	5	Pablo	m	30000
5	6	Pedro	m	10500
6	7	Gaby	f	11000
7	8	Cecilia	f	27000

Si queremos que en lugar de todos los datos, se restrinja la salida a las primeras cinco filas, podemos utilizar el método **head()**.

```
dat_csv.head()
```

Out[9]:

	id	Nombre	Sexo	Sueldo
0	1	Juan	m	10000
1	2	Ramón	m	20000
2	3	Anna	f	10500
3	4	Julieta	f	13000
4	5	Pablo	m	30000

En los dos casos anteriores, aparecen los elementos de la primera fila como los nombres de las columnas de datos, si no quisiéramos este comportamiento podemos agregar el atributo header = None, de la siguiente manera:

```
dat_csv = pd.read_csv('empleados.csv', encoding = "ISO-8859-1", header = None)
dat_csv
```

La salida ahora ya no restringe los datos:

	0	1	2	3
0	id	Nombre	Sexo	Sueldo
1	1	Juan	m	10000
2	2	Ramón	m	20000
3	3	Anna	f	10500
4	4	Julieta	f	13000
5	5	Pablo	m	30000
6	6	Pedro	m	10500
7	7	Gaby	f	11000
8	8	Cecilia	f	27000

Si quisiéramos que tomar una determinada línea como cabecera, debemos asignarle al parámetro header el número de fila correspondiente, en el siguiente ejemplo se toma la línea 3 como cabecera:

```
dat_csv = pd.read_csv('empleados.csv', encoding = "ISO-8859-1", header = 3)
dat_csv
```

Out[11]:

	3	Anna	f	10500
0	4	Julieta	f	13000
1	5	Pablo	m	30000
2	6	Pedro	m	10500
3	7	Gaby	f	11000
4	8	Cecilia	f	27000

Por defecto, **read_csv** asigna un índice numérico predeterminado que comienza con cero al leer los datos. Sin embargo, es posible alterar este comportamiento pasando el nombre de la columna que utilizaremos como índice. A continuación se dejara la columna **id** como índice de la tabla:

```
dat_csv = pd.read_csv('empleados.csv', encoding = "ISO-8859-1", index_col='id')
dat_csv
```

Out[13]:

id	Nombre	Sexo	Sueldo
1	Juan	m	10000
2	Ramón	m	20000
3	Anna	f	10500
4	Julieta	f	13000
5	Pablo	m	30000

En el caso de que queramos restringir la tabla a algunas columnas específicas, podemos realizarlo con el parámetro **usecols** indicando en una lista las columnas seleccionadas.

```
dat_csv = pd.read_csv('empleados.csv', encoding = "ISO-8859-1", usecols=['Nombre', 'Sueldo'])
dat_csv.head()
```

Out[15]:

	Nombre	Sueldo
0	Juan	10000
1	Ramón	20000
2	Anna	10500
3	Julieta	13000
4	Pablo	30000

Nota: Si alguna de las líneas se encontraran en blanco, podemos eliminar dichas líneas mediante el atributo **skip_blank_lines=False**

Si queremos eliminar una fila en particular, podemos utilizar el parámetro **skiprows**, en el siguiente caso se eliminan las filas 1 y 4:

```
dat_csv  = pd.read_csv('empleados.csv', encoding = "ISO-8859-1", skiprows = [1,4])
dat_csv
```

Out[16]:

	id	Nombre	Sexo	Sueldo
0	2	Ramón	m	20000
1	3	Anna	f	10500
2	5	Pablo	m	30000
3	6	Pedro	m	10500
4	7	Gaby	f	11000
5	8	Cecilia	f	27000

Para presentar un número dado de filas desde el inicio, podemos utilizar el parámetro **nrows**, en el siguiente ejemplo se presentan las primeras tres filas de datos.

```
dat_csv  = pd.read_csv('empleados.csv', encoding = "ISO-8859-1",  nrows=3)
dat_csv
```

Out[19]:

	id	Nombre	Sexo	Sueldo
0	1	Juan	m	10000
1	2	Ramón	m	20000
2	3	Anna	f	10500

Para seleccionar un rango específico de filas podemos utilizar el método query especificando los límites aplicables sobre una determinada columna:

```
dat_csv  = pd.read_csv('empleados.csv', encoding = "ISO-8859-1")
dat_csv.query('2 < id < 6')
```

Out[21]:

	id	Nombre	Sexo	Sueldo
2	3	Anna	f	10500
3	4	Julieta	f	13000
4	5	Pablo	m	30000

Para tratar con los datos de una columna, podemos recuperarlos dentro de una lista mediante el uso de un bucle for, y luego si es necesario convertir la lista en una array como se muestra a continuación:

```
import pandas as pd
import numpy as np
dat_csv  = pd.read_csv('datos.csv', encoding = "ISO-8859-1")
datos_x = dat_csv.x
datos_y = dat_csv.y
x = []
y = []
for i in dat_csv.x:
    x.append(i)
for j in dat_csv.y:
    y.append(j)
print(x)
print(y)

x_array = np.array(x)
print(type(x_array))
print(x_array)
```

La salida nos retorna:

[10000, 20000, 10500, 13000, 30000, 10500, 11000, 27000]

Desde Excel

En el caso de trabajar directamente con un archivo de Excel, podemos utilizar el método read_excel() de forma análoga a como lo hicimos anteriormente con read_csv()

```
import pandas as pd
dat_excel  = pd.read_excel('empleados.xlsx')
dat_excel.head()
```

Nota: El parámetro **sheetname** nos permite seleccionar el número de hoja, las cuales se comienzan a numerar desde cero.

Formato json

El formato JSON es un lenguaje de intercambios de datos como XML pero mucho más simple. Es un lenguaje muy utilizado en el envío de datos desde aplicaciones web y móviles realizadas con javascipt. Por su similitud con el tipo de datos de objetos de javascript a adquirido un papel muy importante en desarrolladores de esta comunidad. También es muy utilizado por los desarrolladores de videojuegos de y de animaciones para describir las partes del cuerpo de un avatar o darle formato a las texturas de un objeto.

El formato JSON para los que trabajamos con python sería muy similar asemejarlo a una lista de diccionarios, o sea que podemos pensar en la siguiente estructura:

```
[
  {
    " " : " ",
    " " : " ",
    " " : " ",

  },
    {
    " " : " ",
    " " : " ",
    " " : " ",
  },
]
```

De hecho cuando transformamos un json a python es entendido exactamente así, como una lista de diccionarios. Consideremos el siguiente ejemplo:

json1.json

```
{
   "nombre": "Ana",
   "edad": 33,
   "estado_civil": true,
   "esposo": "Pablo",
   "hijos": ["Cecilia","Luis"],
   "autos": [
      {
         "modelo": "Ford",
         "color": "rojo"
      },
      {
         "modelo": "Chevrolet",
         "color": "azul"
      }
   ]
}
```

Pandas utiliza el método read_json() para importar un archivo con este formato como se muestra a continuación:

```
import pandas as pd
movies_json = pd.read_json('json1.json')
movies_json.head()
```

Out[2]:

	nombre	edad	estado_civil	esposo	hijos	autos
0	Ana	33	True	Pablo	Cecilia	{'modelo': 'Ford', 'color': 'rojo'}
1	Ana	33	True	Pablo	Luis	{'modelo': 'Chevrolet', 'color': 'azul'}

Formato html

Con pandas podemos importar un formato html desde la web o desde un archivo local, los datos los debemos guardar en tablas. Los datos tomados son los correspondientes a las filas <tr></tr> según se muestra a continuación:

index.html

```
1   <!Doctype html>
2   <html>
3      <head>
4         <title>
5            Mi sitio
6         </title>
7      </head>
8      <body>
9         <h1>Esta es una tabla</h1>
10        <div>
11           <table width="100%" border=1 cellpadding=1 cellspacing=2
12  style="background-color: #ffffff;">
13              <th valign=top>
14                 <td style="border-style: inset;">
15                 <p><span class=rvts1>id</span></p>
16                 </td>
17                 <td style="border-style: inset;">
18                 <p><span class=rvts1>Nombre</span></p>
19                 </td>
20                 <td style="border-style: inset;">
21                 <p><span class=rvts1>Sexo</span></p>
22                 </td>
23                 <td style="border-style: inset;">
24                 <p><span class=rvts1>Sueldo</span></p>
25                 </td>
26              </th>
27              <tr valign=top>
28                 <td style="border-style: inset;">
29                 <p><span class=rvts1>1</span></p>
30                 </td>
31                 <td style="border-style: inset;">
32                 <p><span class=rvts1>Anna</span></p>
33                 </td>
34                 <td style="border-style: inset;">
35                 <p><span class=rvts1>f</span></p>
```

```
36                  </td>
37                  <td style="border-style: inset;">
38                  <p><span class=rvts1>10000</span></p>
39                  </td>
40                </tr>
41              </table>
42            </div>
43        </body>
44    </html>
```

```
import pandas as pd
pd.read_html('index.html')
```

```
Out[12]: [   0     1  2      3
          0  1  Anna  f  10000]
```

Base de datos – SQLite3

También podemos trabajar con SQLite 3, el cual es un tipo de base de datos relacional como MySQL o MariaDB. Para trabajar con este tipo de bases de datos podemos descargar SQLITE Studio, el cual es un administrador gráfico de este tipo de bases de datos del siguiente sitio: ***https://sqlitestudio.pl/index.rvt***

La conexión requiere:

- Importar el módulo para trabajar con esta base de datos
- Declarar en el método **connect()** el nombre de la base y la ruta a la misma.
- Introducir la sentencia de búsqueda en lenguaje sql mediante el método **read_sql_query()**

```
 import pandas as pd
import sqlite3
conn = sqlite3.connect("mibase.sqlite")
df = pd.read_sql_query("SELECT * FROM producto;", conn)
df.head()
```

Out[9]:

	id	nombre	sexo
0	1	Pedro	m
1	2	Anna	f
2	3	Celeste	f

Lenguaje xml

Xml es un lenguaje de etiquetas, es utilizado como lenguaje de transferencia de datos e incluso en algunas plataformas como **android** es utilizado para crear las vistas de las pantallas. Para ver como importar los datos partiremos del siguiente ejemplo:

datos.xml

```
<?xml version="1.0"?>
<datos>
   <cliente nombre="Juan" >
      <email>juan@gmail.com</email>
      <telefono>011-1111111111</telefono>
      <direccion>
         <calle>Tacuarí 342</calle>
      </direccion>
   </cliente>
   <cliente nombre="Anna" >
      <email>anna@gmail.com</email>
   </cliente>
   <cliente nombre="Pedro" >
      <email>pedro@gmail.com</email>
      <telefono>011-2222222222</telefono>
      <direccion>
         <calle>Río de Janeiro 215</calle>
      </direccion>
   </cliente>
   <cliente nombre="Gabriela" >
      <email>gabriela@gmail.com</email>
      <telefono>011-3333333333</telefono>
      <direccion>
         <calle>Rivadavia 8345</calle>
      </direccion>
```

```
    </cliente>
</datos>
```

Para trabajar con los datos podemos utilizar la API XML de ElementTree, que es un procesador XML simple y ligero. Para leer y analizar un archivo XML, todo lo que necesitamos es llamar al método de análisis "**parse**" :

```
import xml.etree.cElementTree as et
parsed_xml = et.parse("datos.xml")
```

El código anterior retorna un objeto ElementTree, y podemos usar el método "**iter ()**" para generar un iterador (para elementos XML específicos) o "**getroot ()**" para obtener el elemento raíz de este árbol, y luego iterar todos los elementos.

En el siguiente ejemplo de código se utiliza la función main para recorrer los valores del documento y darle formato

```
import xml.etree.cElementTree as et
import pandas as pd

def obtenerValorDeNodo(node):
    return node.text if node is not None else None


def main():
    parsed_xml = et.parse("datos.xml")
    dfcols = ['nombre', 'email', 'telefono', 'calle']
    df_xml = pd.DataFrame(columns=dfcols)

    for node in parsed_xml.getroot():
        nombre = node.attrib.get('nombre')
        email = node.find('email')
        telefono = node.find('telefono')
        calle = node.find('direccion/calle')

        df_xml = df_xml.append( pd.Series([nombre, obtenerValorDeNodo(email), obtenerValorDeNodo(telefono), obtenerValorDeNodo(calle)], index=dfcols), ignore_index=True)
```

```
    print(df_xml)

main()
```

La salida nos queda:

	nombre	email	telefono	calle
0	Juan	juan@gmail.com	011-1111111111	Tacuarí 342
1	Anna	anna@gmail.com	None	None
2	Pedro	pedro@gmail.com	011-2222222222	Río de Janeiro 215
3	Gabriela	gabriela@gmail.com	011-3333333333	Rivadavia 8345

Bibliografía utilizada y sugerida

Libros

Programming Python 5th Edition – Mark Lutz – O'Reilly 2013

Mastering Exploratory Analysis with pandas - By Harish Garg - September 2018

Manual online

(2019, 01). Obtenido 01, 2019, de https://pandas.pydata.org/

Lo que vimos

En esta unidad hemos visto como tomar datos de diferentes tipos de procedencia para su posterior análisis.

Lo que viene:

En la siguiente unidad veremos cómo graficar los datos obtenidos utilizando Matplotlib.