Estadística Descriptiva y Pandas

El objetivo del siguiente notebook es, además de repasar conceptos de estadística, que sigas aprendiendo a operar con Numpy y que, además, incorpores Pandas a tu caja de herramientas.

1. Estadística Descriptiva

La Estadística Descriptiva nos sirve para comenzar a analizar y entender un conjunto de datos. En el caso de datos numéricos, lo hace obteniendo *valores estadísticos* que, de alguna forma, reemplazan a nuestros datos. Por ejemplo, es muy difícil leer y *entender* la edad de 1000 personas. Pero con un grupo reducido de valores estadísticos (mínimo, máximo, media y desviación estándar, etc.) podemos aproximarnos a ese conjunto de una manera mucho más comprensible. Veamos dos medidas muy importantes:

Promedio

Dados n números $x_1, x_2, \ldots, x_n$, el promedio o media es

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

Desviación Estándar

La varianza y la desviación estándar nos dan una idea de cuán "dispersos" están los valores con respecto a su promedio.

$$Var = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}$$

La desviación estándar es la raiz cuadrada de la varianza. En general se usa la letra griega σ para representarla o las siglas SD:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}$$

$$SD = \sqrt{Var}$$

Comentarios:

1. Dado un conjunto de números, el promedio suele ser considerado el número más representativo de ese conjunto. Esto no siempre es así. Pensá o googleá por qué.

2. Al conjunto de números $x_1, \ldots, x_n$ los pueden encontrar por el nombre de *población* o *muestra* (¡Ojo que no estamos diciendo que *población* y *muestra* sean lo mismo!).

Challenge:

Vamos a utilizar de excusa la estadística descriptiva para hacer un desafío de programación:

Dadas la siguiente lista de números, escribir una rutina que calcule su promedio, su varianza y desviación estándar. **Pistas:**

- Probablemente te sea muy útil usar lo que hiciste para ejercicios anteriores.
- Para calcular la varianza y la desviación estándar, usa el resultado que obtuviste al calcular el promedio.

```
x_s = [1,2,3,1,2,2,3,4,1,2,3,4,1,2,4]
# COMPLETAR
n=len(x_s)
suma=0
for numero in x_s:
  suma=suma+numero
promedio=suma/n
print('el promedio es',promedio)
suma=0
for numero in x_s:
  suma=suma+(numero-promedio)**2
var=suma/(n-1)
print('la varianza es', var)
desviacion=var**(1/2)
print('la disviación estandar es', desviacion)
```

```
el promedio es 2.3333333333333335
la varianza es 1.2380952380952384
la disviación estandar es 1.1126972805283737
```

```
# COMPLETAR
```

¿Cómo te fue con el Challenge? Si no pudiste resolverlo, no te preocupes. ¡NumPy tiene funciones ya incorporadas que calcula algunos estadísticos sobre un arreglo!

1.2 Estadística con NumPy

Veamos cómo se calculan, en NumPy, el promedio, varianza y desviación estándar sobre un arreglo.

```
import numpy as np

x_s = np.array([1,2,3,1,2,2,3,4,1,2,3,4,1,2,4])

# Promedio
print(x_s.mean())

# Varianza
print(x_s.var(ddof = 1))

# Desviación estándar
print(x_s.std(ddof = 1))
```

Para investigar: ¿qué es el parámetro `ddof` de esa función?¿Qué pasa si no lo usas? Esta pregunta es **difícil** y requiere cierto conocimiento previo. Pero intenta, de todas formas, averiguarlo.

NumPy también puede calcular percentilos (¡googlear!), cuantilos, mínimos y máximos:

```
print(np.percentile(x_s,75))
print(np.quantile(x_s,0.5))
print(np.min(x_s))
print(np.max(x_s))
```

Para investigar: ¿Cuál es la diferencia entre `np.percentile()` y `np.quantile()` ?¿Cómo obtendrías los cuartiles a partir de ellos?

1.3 Generación de muestras al azar

Una cosa sumamente útil que podemos hacer con NumPy es generar muestras al azar. Esto no permite simular situaciones. Por ejemplo, las tiradas de dados que aparecen en el GIF de la bitácora. Estas funciones las encontramos dentro del paquete `random` de NumPy, cuya documentación pueden encontrar aquí. Veamos cómo lo podemos hacer:

```
muestras_dado = np.random.randint(1,7, size = 15)
```

```
print(muestras_dado)

### También se puede
muestras_dado = np.random.choice([1,2,3,4,5,6], size = 15)
print(muestras_dado)
```

Ejercitación

Ejercicio 1: ¿Cuál será el promedio de los valores obtenidos al tirar muchas veces un dado?¿Te animás a averiguar - o calcular - cuánto *debería dar* antes de hacerlo? Vamos a tratar de responder esta pregunta **simulando** un dado. Para ello:

- Obtener muestras al azar de un dado usando lo que vimos anteriormente.
- Calcular su promedio y desviación estándar.

¿A partir de qué cantidad de muestras el promedio se "estabiliza"?

```
# COMPLETAR
```

Ejercicio 2: Simular un dado cargado para favorecer un valor de su elección. Por ejemplo, el seis. Para ello, consultar la ayuda de la función `np.random.choice`. ¿Cómo se modifica el promedio y la desviación estándar?

```
# COMPLETAR
```

2. Pandas

Pandas es la librería más conocida de Python para manipular y analizar datos. Está montada sobre NumPy, por lo cual muchas funcionalidades son similares. Utilizaremos Pandas para trabajar con datasets estructurados (y bueno, ¡bastante más!).

Así como NumPy nos proveé de los *arreglos* y con ellos accedemos a muchas nuevas funcionalidades, Pandas nos provee de los *Data Frames* y las *Series*. Por lejos, el objeto más utilizados es el primero, los Data Frames.

En esta sección empezaremos a:

1. Familiarizarnos con los Data Frames de Pandas, manipular sus funciones básicas y entender la lógica de las mismas (¡para después googlearlas!).

2. Empezar a trabajar con Datasets.

¡Manos a la obra!

2.1 Primeros pasos

Importamos la librería.

```
import pandas as pd
```

Vamos a crear nuestro propio dataset. Es decir, agarrar a mano los datos poblacionales de http://www.ign.gob.ar/nuestrasactividades/geografia/datosargentina/divisionpolitica y guardarlos en una variable `data_dic`.¿Qué tipo de variable es, desde el punto de vista de la programación?

Nota: la población está en número de habitantes y la superficie en km2.

```
data_dic = {"Jurisdiccion":["CABA","Buenos Aires","Catamarca","Chaco","Chubut","Córdo
                            "Santa Fe"],"Poblacion":[2890151,15625084,367828,1055259,5
                                                     1101593,638645,273964,3194537],"Su
           [200,307521,102606,99633,509108,165321,53219,148827,29801,203013,243943,13
```

Así como podemos crear arreglos a partir de listas, podemos crear Data Frames a partir de diccionarios.

```
# Creamos el DataFrame
data_pandas = pd.DataFrame(data_dic)
data_pandas
```

Ejercicio 1: investigar las funciones que se implementan en la próxima celda. ¿Qué hacen? ¿Para qué piensan que pueden ser útiles?

```
# data_pandas.head()
# data_pandas.tail()
# data_pandas.count()
# data_pandas.shape
```

Ejercicio 2: agregar al Dataset la información correspondiente a alguna jurisdicción faltante. Recuerden que, al tratarse de una nueva instancia, corresponde a una fila. Pista:

googlear "add row to pandas dataframe" o similar. No hay una única forma de hacerlo.

```
# COMPLETAR
```

Ejercicio 3: Investigar las funciones columns e index. ¿Qué hacen? ¿Qué tipo de dato es su salida?¿A qué tipo de dato conocido se parecen?

```
# data_pandas.columns
# data_pandas.index
```

Ejercicio 4: ¿Qué hacen las siguientes operaciones?

```
# data_pandas['Jurisdiccion']
# data_pandas[['Jurisdiccion','Poblacion']]
# data_pandas.Jurisdiccion
# 'Poblacion' in data_pandas
```

Ejercicio 5: Agregar una columna al dataframe que corresponda a la densidad de cada jurisdicción. Usar la información que **ya está** en el dataset.

```
# COMPLETAR
```

2.2 Filtrado por máscara.

Lo que veremos a continuación es **muy importante**, ya que es una operación que haremos muchas veces. Su implementación es muy parecida tanto en NumPy como en Pandas, por lo que veremos cómo hacerlo primero en NumPy luego en Pandas.

Supongamos que hacemos 50 tiradas de un dado, como hicimos en la sección anterior, pero queremos seleccionar solamente aquellas tiradas que fueron menores que cuatro. ¿Cómo podemos hacerlo?

```
muestras_dado = np.random.randint(1,7, size = 50)
print(muestras_dado)
```

Lo que podemos hacer es crear una máscara:

```
mascara = muestras_dado < 4
print(mascara)
print(type(mascara))
```

Notar que `mascara` es un arreglo de booleanos, con `True` en los valores que cumplen la condición y `False` donde no. Una vez que creamos la máscara, podemos usarla para seleccionar de nuestro arreglo aquellos elementos que queríamos:

```
print(muestras_dado[mascara])
```

Notar que con `mascara.sum()` podemos contar cuántas veces se cumple la condición que pedimos.

```
print(mascara.sum())
```

A veces, podemos hacerlo en una sola línea. Supongamos que queremos aquellas tiradas donde salió seis:

```
print(muestras_dado[muestras_dado == 6])
```

En Pandas

Supongamos que queremos seleccionar aquellas jurisdicciones cuya población sea mayor a un millón de habitantes. Podemos hacerlo de la siguiente forma:

```
data_pandas[data_pandas.Poblacion > 1000000]
```

¿Y si queremos seleccionar aquellas jurisdicciones cuya población sea mayor a un millón de habitantes **y** su superficie menor a cien mil km2?

```
mascara = np.logical_and(data_pandas.Poblacion > 1000000, data_pandas.Superficie < 10
data_pandas[mascara]

### Es equivalente
# data_pandas[(data_pandas.Poblacion > 1000000) & (data_pandas.Superficie < 100000)]
```

Ejercicio: seleccionar aquellas jurisdicciones cuya población sea menor a 500 mil habitantes **o** su superficie mayor a cien mil km2.

```
# COMPLETAR
```

2.3 Iris dataset

¿Pero siempre vamos a tener que crear un diccionario y luego pasarlo a un Data Frame? Evidentemente, esta opción no parece muy cómoda, en particular para conjuntos de datos de gran volumen. Veamos cómo trabajamos con un conjunto de datos preexistente.

Aviso: Esta sección es, en realidad, un ejercicio. Para hacerlo, debes ir googleando y consultando la documentación que consideres apropiada. Obviamente, también puedes consultar a tu mentor/a.

Vamos a trabajar con el Iris Dataset, probablemente uno de los conjuntos de datos más famosos, ya que muchos ejemplos se realizan con él. Es un dataset sencillo pero ilustrativo.

1. Abrir con Pandas el archivo 'DS_Bitácora_04_iris.csv' (¿Qué tipo de archivo es?) e imprimir sus primeros cinco elementos. Pista: `pd.read...()`.

```
# COMPLETAR
```

2. ¿Cuántas columnas (features) tiene?¿Cuáles son sus nombres?¿Y cuántas filas (instancias)? Pistas: `shape, columns`.

```
# COMPLETAR
```

3. Obtener el valor medio y desviación estándar de cada columna. ¿Hay alguna función de Pandas que nos dé aún más estadísticos? Pistas: `describe`.

```
# COMPLETAR
```

4. ¿Creen que todas las columnas tienen información? *Tirar* la columna que crean que está demás. Dependiendo de la función que uses - hay más de una opción -, tal vez tengas que prestar **mucha** atención al argumento `inplace`. Pista: `drop`, `del`.

```
# COMPLETAR
```

5. ¿Para qué sirven `loc` e `iloc`? Crea algunos ejemplos.

✓ 0 s completado a las 18:24