Machine Learning 3

Autores: Nestor Marmolejo
Agustín Salazar

IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: nestor.marmolejo@utp.edu.co & agustin.salaza@utp.edu.co

***Resumen*— Este artículo tiene como objetivo dar a conocer sobre la librería "NumPy" de Python, dar a conocer dicha librería de una manera sencilla y eficiente, explorando algunas de sus funciones aplicadas no solamente teoría o ejemplos básicos sino también hacerlo con un ejemplo más elaborado y real.**

Palabras clave*— *Funciones, NumPy, Python.

***Abstract*— This article aims to make known about the Python "NumPy" library, to publicize said library in a simple and efficient way, exploring some of its applied functions not only theory or basic examples but also doing it with a more elaborate example and real.**

***Key Word*— - Functions, NumPy, Python.**

I. INTRODUCCIÓN

NumPy es un paquete de Python que significa "Numerical Python", es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales. Estas estructuras de datos garantizan cálculos eficientes con matrices.
NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

II. METODOLOGÍA

Se importó la librería numpy y se renombró por 'np' para comodidad en el momento de hacer llamados a dicha librería de la siguiente manera:

```
import numpy as np
```

Se creó un vector de seis (6) posiciones y se denomina 'a', después se muestra en pantalla junto con la dimensión y el número de elementos de este con las instrucciones:

```
print(a)
print(a.ndim)
print(a.shape)
```

El resultado de las instrucciones anteriores imprime:

```
[0 1 2 3 4 5]
1
(6,)
```

Se cambió la estructura del vector 'a' y se guardó en un nuevo vector llamado 'b' por medio de la instrucción:

```
b = a.reshape((3,2))
```

Al ejecutar esta instrucción, se modificó la estructura del vector 'a', creando así una matriz de tres (3) filas por dos (2) columnas.

```
[[0 1]
 [2 3]
 [4 5]]
```

A partir de esta matriz obtenida, se puede modificar cualquier elemento, tanto en filas como en columnas. Para modificar un elemento de la matriz, basta con llamar a esta, darle los valores para la fila, la columna y el nuevo valor para el elemento deseado. Por ejemplo, para el caso de la matriz anterior 'b' se deseó modificar el valor dos (2) que se encuentra en la fila uno (1), columna cero (0) por el valor (77).

```
b[1][0] = 77
```

Para verificar que la modificación si se efectuó, se imprimió en pantalla y el resultado que arrojó fue:

```
[[0  1]
 [77 3]
 [4  5]]
```

Cabe aclarar que al modificar el vector 'b', se modificó también el vector 'a', ya que de este se creó el vector 'b'. Para evitar que este se viera afectado, la solución más óptima fue crear una copia del vector original agregando la instrucción `.copy()` al final del comando.

```
c = a.reshape((3,2)).copy()
```

Al comprobar que la copia realizada fue exitosa, se hizo una segunda modificación y se imprimió nuevamente en pantalla.

En los vectores también se pueden realizar operaciones aritméticas, esto es posible llamando al vector y asignándole la operación que se desea aplicar. Por ejemplo:

```
d = np.array([1, 2, 3, 4, 5])
print(d*2)
print(d**2)
```

Al ejecutar la instrucción anterior, se imprimió el vector con cada elemento multiplicado por dos (2) y elevado al cuadrado (^2).

```
[1, 4, 6, 8, 10]
[1, 4, 9, 16, 25]
```

Si se quiere realizar comparación, igualación, entre otros. Se llama el vector y se le asigna la condición. Como ejemplo, se deseó imprimir en pantalla, que valores de este vector son mayores que cuatro `print(a>4)`.

El resultado fue:

```
[False False False False True]
```

Esto también se cumple para modificar cualquier elemento de la matriz o del vector.

```
a[a==4] = 1
```

El resultado que se obtuvo fue:

```
[1 2 3 1 5]
```

Como se muestra anteriormente el numero cuatro (4) fue reemplazado por uno (1), ya que cumplió la condición que se le asignó.

En una matriz o vector que se le esté asignado valores sea por un sensor o por cualquier cosa, hay una gran posibilidad de que en un momento dado exista un tipo de dato erróneo o un valor que no es prudente ante la información que se está recolectando. Para poder controlar los dichos valores erróneos se aplica el identificador np.NAN, ayudando a la veracidad de los datos. Para dar un ejemplo se aplica el siguiente comando:

```
c = np.array([1, 2, np.NAN, 3, 4])
```

Si se visualiza esta instrucción en pantalla, se puede ver que el vector creado es `[1 2 NAN 3 4]`.

Se quiso evidenciar la existencia de los valores erróneos de dicho vector y se imprimió en pantalla el vector definiendo cada uno de sus elementos como ‘True’ o ‘False’.

```
print(np.isNAN(c))
```

Se obtuvo:

```
[False False True False False]
```

Determinando que verdaderamente en la posición tres (3) del vector existía un valor erróneo.

Se extrajeron los valores que no eran erróneos o no tipo nan del vector y formando por así decirlo, un nuevo vector solamente con los datos verdaderos, es decir, los valores no erróneos. Esto fue posible con la instrucción:

```
print(c[~np.isNAN(c)])
```

El vector que se visualizó fue:

```
[1.2.3.4.]
```

Además de esto, se calculó el promedio de los valores que no eran erróneos con el comando:

```
print(np.mean(c[~np.isNAN(c)]))
```

El valor obtenido fue el esperado, que fue: `2.5`.

III. APLICACIÓN

Una empresa vende el servicio de proporcionar algoritmos de aprendizaje automático a través de HTTP.
Con el éxito creciente de la empresa, aumenta la demanda de una mejor infraestructura para atender todas las solicitudes web entrantes. No se quiere asignar demasiados recursos, ya que sería demasiado costoso. Por otro lado, se perderá dinero si no ha reservado suficientes recursos para atender todas las solicitudes entrantes.
Ahora, la pregunta es, ¿cuándo se alcanzará el límite de la infraestructura actual, que se estima en 100.000 solicitudes por hora?
Nos gustaría saberlo de antemano cuando tenemos que solicitar servidores adicionales en la nube para atender todas las solicitudes con éxito sin pagar por las no utilizadas.

Se desarrolló un programa básico de Machine Learning para poder darle solución a dicha problemática. Para realizarlo, se entrega el paquete ‘`web_traffic.tsv`’, el cual incluye los datos a procesar.

Se utilizó la siguiente función para extraer los registros del archivo y se imprimió igualmente los diez (10) primeros valores en pantalla.

```
data = np.genfromtxt("web_traffic.tsv", delimiter = "\t")

print(data[:10], '\n')
```

```
[[1.000e+00 2.272e+03]
 [2.000e+00       nan]
 [3.000e+00 1.386e+03]
 [4.000e+00 1.365e+03]
 [5.000e+00 1.488e+03]
 [6.000e+00 1.337e+03]
 [7.000e+00 1.883e+03]
 [8.000e+00 2.283e+03]
 [9.000e+00 1.335e+03]
 [1.000e+01 1.025e+03]]
```

Figura 1. Algunos datos del archivo.

En la imagen anterior se puede apreciar los primeros diez (10) de setecientos cuarenta y tres (743) registros del archivo. La primera columna representa el numero de horas y la columna dos (2), el número de tareas ejecutadas.

Se extrajeron del archivo dos columnas de registros, dividiéndolas cada una en vectores diferentes, llamados ‘x’ y ‘y’, para manejarlos de manera individual con ayuda de las funciones:

```
x: data[:,0]
y: data[:,1]
```

Para reafirmar que si se extrajeron y se guardaron correctamente todos los registros se aplicaron dos (2) funciones:

```
print(x.ndim, '\n')
print(y.ndim, '\n')
```

En pantalla se visualizó:

```
1
1
```

Y,

```
print(x.shape, '\n')
print(y.shape)
```

En pantalla se visualizó:

```
(743,)
(743,)
```

Se siguió avanzando con el proceso y para esto se investigó cuantos valores erróneos o tipo ‘nan’ había el vector ‘y’ con la función:

```
print(np.sum(np.isnan(y)))
```

la respuesta fue:

```
8
```

Para eliminar estos registros erróneos en ‘x’ y ‘y’.

```
x = x[~np.isnan(y)]
y = y[~np.isnan(y)]
```

En el vector ‘x’ para eliminar estos valores erróneos hay que hacerlo con los de ‘y’, ya que los valores erróneos o ‘nan’ se encuentran son en ‘y’ y no en ‘x’. Por lo tanto, si se le aplicara la función anterior al vector ‘x’, saldría una diferencia gradual en el número total de registros de los vectores, ya que ‘x’ no contiene valores erróneos.

Se importó la siguiente librería para graficar dichos registros, esto con el objetivo de exponer la información de una forma más sencilla.

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

La gráfica se tituló “Tráfico Web del último mes”, y con la función `plt.scatter(x, y, s=10)` que sirve para hacer gráficos de dispersión.

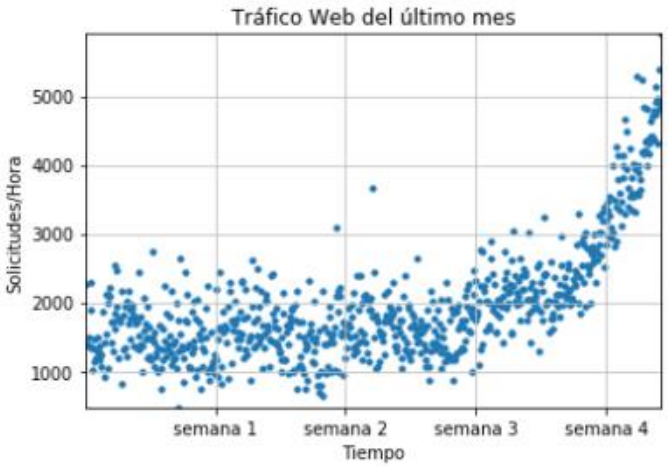

Figura 2. Grafico de dispersion de los registros.

```
plt.xticks([w*7*24 for w in range(10)],
['semana %i' % w for w in range(10)])
```

Esta función grafica los puntos que representan las solicitudes por hora en cada semana del mes.

IV. CONCLUSIÓN

- La librería NumPy es de gran ayuda para el procesamiento y aplicación de vectores y matrices, ya que ofrece una serie de métodos y funciones que facilitan al programador encontrar soluciones prontas a tan extensos registros (si es el caso).

- Siempre hay que tener en cuenta que cuando se recolectan datos habrán o existirán datos erróneos que no son acorde a la recolección que se esta haciendo, para controlar esto hay que tener una solución pronta porque por estos desfalcos pueden ocurrir cosas catastróficas ya sea en una investigación o en una empresa.
- Las gráficas son indispensables para ilustrar la recolección o manejo de los datos, ya que de esta manera se pueden visualizar de una forma más fácil y poder tomar decisiones relevantes.

V. BIOGRAFÍA

Marmolejo Nestor. Nacido en Pereira-Risaralda Colombia el 27 de Agosto de 1999, bachiller del Instituto Técnico Superior, Técnico en implementación y mantenimiento de equipos electrónicos industriales y estudiante de ingeniería en sistemas y computación en la Universidad Tecnológica de Pereira, cursando actualmente 8avo semestre, becado por el programa de la alcaldía de Pereira llamado 'Universidad para cuba'.

Salazar Agustín. Nació en Pereira, Colombia en 1998. Recibió el título de bachiller técnico en el año 2015 del Instituto Técnico Superior de Pereira. Actualmente se encuentra cursando su pregrado Ingeniería en Sistemas y Computación en la Universidad Tecnológica de Pereira.

VI. REFERENCIAS

[1] https://ligdigonzalez.com/introduccion-a-numpy-python-1/
[2] https://es.wikipedia.org/wiki/NumPy