Computación Blanda

Soft Computing

Analisis de datos con numpy

Autores: Juan José León Tabares, Luis Aníbal Loaiza Cardona
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: juanjose.leon@utp.edu.co Luis.loaiza@utp.edu.co

***Resumen*— Este documento tiene como finalidad fomentar la programación en Python con la librería numpy para aplicarla en problemas complejos de la computación blanda, para facilitar su implementación a futuro.**

Palabras clave*— *numpy, vector, arreglo, atributo, variable, función, datos.

***Abstract*— This document is intended to promote Python programming with the numpy library to apply it to complex problems in soft computing, to facilitate its implementation in the future.**

Keywords: number, vector, array, attribute, variable, function, data.

I. INTRODUCCIÓN

Se realiza un algoritmo junto con la librería numpy de Python para filtrar los datos de un archivo .tsv y realizar las diferentes operaciones con estos datos, además para mostrar visualmente la propagación de los datos las semanas correspondientes a su inicio.

II. ANALISIS

El primer paso es extraer los datos que tiene un archivo .tsv, principalmente se basó en la información relativa a las actividades realizadas en cada una de las horas del día. Dichas actividades están almacenada hora por hora en un archivo de testo llamado: web_traffic.tsv.

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

Con la función anterior se convierte un archivo de texto a una matriz para poder operar los datos más fácilmente, que tiene como valores de entrada el nombre del archivo y un delimitador, dentro de este archivo se pudieron encontrar dos tipos de datos, el primero son las horas transcurridas y el segundo son el número de tareas ejecutadas en su respectiva hora.

Con los siguientes arreglos se logra dividir la información de los datos:

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

En la posición cero (0) se encuentran cada una de las horas transcurridas y se almacenan en la variable X, mientras que en la posición uno (1) se encuentran las tareas ejecutadas de cada hora y se almacena en la variable Y.

Ya con los datos debidamente separados en las variables se da a la tarea de saber las dimensiones y la cantidad de datos que compone cada variable, para esto se toma la siguiente línea de código:

- Con el atributo ndim se puede retornar el número de dimensiones del vector.

  Dimensiones de los datos de las variables:
  ```
  print(x.ndim, '\n')
  print(y.ndim, '\n')
  ```

- Con el atributo shape se puede retornar la cantidad de datos del vector.

  Cantidad de datos en las variables:
  ```
  print(x.shape)
  print(y.shape)
  ```

Analizando la cantidad de datos en los vectores, se llegó a que hay unos valores nulos (como ejemplo: nan). Por lo tanto, se debe hallar alguna forma para eliminar estos valores, ya que si

no se hacen pueden afectar a la entrega de los resultados solicitados.

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

En la función anterior np.isnan que como valor de entrada recibe un vector, lo que realiza es convertir los datos nulos (nan) en valores booleanos en este caso será True para identificar que existe un valor nulo. La función np.sum lo que realiza es la suma de cada uno de los valores booleanos que son True para retornar el total de valores nulos que contenga el vector, como valor de entrada recibe la vector con valores booleanos y retorna un numero entero indicando la cantidad de datos nulos.

Al eliminar los datos nulos (nan) se presente un problema con las horas debido a que la columna de las horas queda con más datos en comparación con el número de tareas ejecutadas; para resolver este problema se tiene en cuenta el siguiente código de línea:

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

Basado en el código anterior lo primero que se realiza es eliminas las horas donde están las tareas ejecutadas nulas, y segundo eliminaría las tareas ejecutadas donde tiene valores nulos. Se le asigna a X un vector X con todas las posiciones nulas (nan) eliminadas por esta razón el comando es ~np.isnan. En tanto a la variable Y se le realiza el mismo procedimiento pero con el vector Y.

Con la función de matplotlib plt.scatter permite dibujar los puntos en el plano cartesiano con un tamaño determinado, la función plat.scatter recibe un arreglo de posiciones tanto en X como en Y, por ultimo recibe el tamaños de los puntos.

```
plt.scatter(x, y, s=10)
```

La función plt.xticks realiza la demarcación de cada semana multiplicando la cantidad de horas que tiene cada día y los días que tiene la semana, aplicándolo en un rango de diez (10). Recibe como valor de entrada cada cuanto debe de hacer la demarcación con un valor de W que funciona como un puntero recorriendo la gráfica contando los puntos, este proceso de demarcación lo realiza en un rango de diez (10) veces, además recibe una cadena en este caso SEMANAS con una variable %i que funciona como contador.

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

Para realizar la cuadricula se utiliza plt.grid que recibe como entrada un valor booleano que indica si queremos ver o no la cuadricula, como segundo valor se debe indicar el tipo se línea que se quiere utilizar, y como tercer valor entre 0 y 1 recibe la tonalidad de la línea de división y por ultimo con la función plt.show muestra la gráfica en pantalla.

```
plt.grid(True,linestyle='-',color='0.75')

plt.show()
```

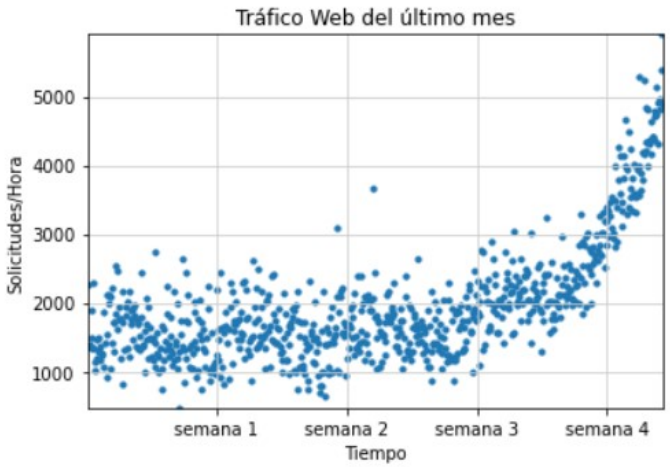

Figura 1. Grafico web

III. CONCLUSIONES

Si se tienen en cuenta todas funciones mencionadas anteriormente se puede ver su comportamiento a gran escala, esto con el fin de predecir su valor a futuro.

Asumiendo que en cualquier recopilación de datos los errores o datos nulos estarán presentes, siempre se debe tener en cuenta realizar un filtrado para manipular correctamente los datos.

Con la gráfica se puede visualizar la dispersión de los datos separados por cada semana, para su futuro análisis ya sea en una empresa, una investigación o simplemente para toma de estadísticas.

IV. BIBLIOGRAFIA

1- https://www.tutorialspoint.com/matplotlib/matplotlib_grids.htm
2- https://www.tutorialspoint.com/matplotlib/matplotlib_jupyter_notebook.htm