INTRODUCCIÓN A PYTHON

CLASE 11

Contenido

“

Este espacio curricular permite fortalecer el espíritu crítico y la actitud creativa del futuro graduado. Lo alienta a integrar los conocimientos previos, recreando la práctica en el campo real, al desarrollar un proyecto propio.

”

Programación II

Comenzamos a prepararnos para utilizar Python, para analizar conjunto de datos. Python es un lenguaje que ofrece un ecosistema muy rico de librerías para potenciar el campo del análisis. En esta oportunidad, veremos cómo usar la librería numPy y pandas

Introducción

Hemos visto hasta aquí, que Python es un lenguaje muy amigable para programar. Pero, también, hemos notado que es un lenguaje muy extensible gracias a la gran variedad de librerías que ofrece la comunidad. Para sumar a esto, Python también es un lenguaje de código abierto muy utilizado en el análisis de datos.

Este lenguaje, nos facilita el análisis de datos y es utilizado por muchas empresas. Se pueden usar las funciones analíticas de Python para explorar datos, para programar funciones personalizadas, manipular datos y optimizar el flujo de trabajo.

En conclusión, crear nuestras funciones en Python es muy sencillo. Pero, antes debemos conocer las librerías que constituyen este ecosistema.

librería NumPy

Numpy (numerical Python) es una biblioteca que se utiliza para trabajar datos numéricos. Se utiliza en conjunto con otras librerías como pandas, o sciPy.

Esta librería nos permite trabajar con matrices (estructura multdimensional). A diferencia de las listas, que antes hemos visto, las matrices son mas compactas y utilizan menos memoria. Razón por la que terminan siendo mas rápidas y por ella mas utilizadas en estos ámbitos.

Lo primero que debemos hacer es instalar la librería, en el caso que no la tengamos.

Cómo instalar e importar Numpy

Primero debemos asegurarnos tener la librería instalada, en caso de no contar con ella ejecutamos el siguiente código en la terminar (CTRL+ñ)

pip install numpy

Completado con éxito este proceso, importamos la librería dentro de nuestro archivo .py. Notaran en este caso, que hemos renombrado la librería como **np**

```
import numpy as np
```

Creando una matriz

La implementación es muy sencilla, usamos el método array. LA sintaxis para crear una matriz es:

Variable= np.array([lista de valores])

Todo lo que tiene que hacer es pasarle una lista al método. El ejemplo podría ser:

```
mat= np.array([1,2,3,4,5])
```

Analizando la matriz

Se pueden conocer datos de la matriz, como la cantidad de elementos o la cantidad de dimensiones que esta tiene. A continuación, veremos una seria de métodos nuevos que brindan información de la matriz.

type

¿Cómo saber si es una matriz?

Utilizamos la función type, tal como hacíamos antes con las variables

```
print (type(mat))
```

En la terminal veremos esto:

<class 'numpy.ndarray'>

ndim

¿Cómo conocer la cantidad de dimensiones que tiene la matriz?

```
print (mat.ndim)
```

en este caso, el resultado sería:

1

shape

¿Cómo saber la cantidad de elementos en cada dimensión?

```
print (mat.shape)
```

esta matriz tiene solo una dimensión, y la cantidad de elementos es 5

(5,)

Creando matrices automáticas

zeros y ones

Se utiliza para crear una matriz y completar los elementos con ceros

```
mat= np.zeros(5)
```

En el ejemplo anterior, creamos una matriz de una dimensión con 5 elementos iguales a cero. También nos permite crear matrices de más de una dimensión. Algo similar es cuando queremos llenarla con 1.

```
import numpy as np
mat= np.ones(5)
print (mat)
```

empty

Se puede crear una matriz vacía. En realidad, crea una matriz con valores aleatorios, y esos valores depende del estado de entorno, pero digamos que es una matriz vacía 😊

Se logra usando el siguiente código:

```
mat= np.empty(2)
```

no arroja el siguiente resultado:

[1.72159068e+279 -7.07346108e-109]

dtype

Como lo habíamos comentado, es necesario que la matriz sea homogénea. Es decir, tenga los mismos tipos de datos. En NumPy podemos crear matrices especificando el tipo de dato. Es algo que contradice lo que tanto destacamos de Python, pero bueno… en el mundo de desarrollo esto pasa con bastante frecuencia.

```
mat= np.array(2,dtype=np.int32)
```

Agregando, eliminando datos de la matriz

append

Nos permite agrega un elemento al final de la matriz. Hay que tener en cuenta, que la función append crea una nueva matriz

```
import numpy as np

mat= np.array([2,100], dtype=np.int32)
mat=np.append(mat,[200],axis=0)
print (mat)
```

- **arr** es la matriz original a la que está agregando
- **los valores** se agregan a la copia de arr.
- **axis** es el eje a lo largo del cual se añaden los valores. Este es un valor opcional que es Ninguno por defecto.

delete

Retorna una nueva estructura con los datos eliminados.

```
import numpy as np

mat= np.array([2,100], dtype=np.int32)
mat=np.append(mat,[200],axis=0)

mat=np.delete(mat,[0])
print (mat)
```

En este caso, borra el elemento en la posición 0 (cero). Se pueden borrar dos elementos, de la siguiente manera:

```
mat=np.delete(mat,[0,1])
```

Operaciones aritméticas con matrices

Operaciones (+ - * /)

Se pueden sumar dos matrices, como si fuesen variables independientes de un solo tipo de dato

```
mat1=np.array([1,2,3,4])
mat2=np.array([1,2,3,4])
mat3= mat1 + mat2
print (mat3)
```

Estas operaciones básicas como restar, multiplicar y dividir, se pueden hacer de la misma manera. También se puede hacer conta un escalar.

sum

suma todos los elementos de una matriz

```
import numpy as np
```

```
mat1=np.array([1,2,3,4])

print (mat1.sum())
```

cuando tiene más de una dimensión:

```
import numpy as np

mat1=np.array(([1,2,3,4],[1,2,3,4]))

print (mat1.sum())
```

Operaciones avanzadas

max

Devuelve el valor máximo de los elementos

```
mat1=np.array(([1,2,3,40],[-1,2,3,4]))

print (mat1.max())
```

min

Devuelve el mínimo de los elementos

```
mat1=np.array(([1,2,3,4],[-1,2,3,4]))

print (mat1.min())
```

Unique

Devuelve los valores únicos de la matriz

```
mat=np.random.randint(5, size=(2, 4))
print (mat)
mat_unique= np.unique(mat)
print(mat_unique)
```

#valores de mat

[[4 2 4 2]

[2 3 0 2]]

#Valores de mat_unique

[0 2 3 4]

Flip

Invierte la matriz

```
print(mat_unique)
print(np.flip(mat_unique))
```

[0 1 2 3 4]

[4 3 2 1 0]

flatten

convierte la matriz en una matriz de una dimensión

```
mat=np.random.randint(5, size=(2, 4))
print (mat)
print ("flatten")
mat2=mat.flatten()
print (mat2)
```

se observa:

```
[[3 0 4 1]
 [0 0 3 3]]
flatten
[3 0 4 1 0 0 3 3]
```

random

para crear una matriz con valores aleatorios

```
mat=np.random.random_integers(5, size=(2, 4))
print(mat)
```

aunque esta función esta deprecada y se reemplaza por

```
mat=np.random.randint(5, size=(2, 4))
```

Numpy con fórmulas matemáticas

Numpy nos da una facilidad para implementar formulas matemáticas, algo muy apreciable cuando estamos implementando algún algoritmo de machine learning.

Por ejemplo, consideremos la formula del error cuadrático medio que se implementa en la documentación de la librería. Para representar "una fórmula central utilizada en los modelos de aprendizaje automático supervisado que se ocupan de la regresión"

$$RMSE = (1/n)\sum_{i=1}^{n}(Y_{prediccion_i} - Y_i)^2$$

Implementarlo con NumPy es muy sencillo, para ellos creamos dos matrices que deben ser del mismo tamaño

```
real=np.array([3, -0.5, 2, 7])
estiamdo=np.array([2.5, 0.0, 2, 8])
```

completando la formula con el siguiente código:

```
real=np.array([3, -0.5, 2, 7])
estiamdo=np.array([2.5, 0.0, 2, 8])
n=4
RMSE= (1/n) * np.sum(np.square(real-estiamdo))
```

librería Pandas

Es una librería que se especializa en el manejo y análisis de estructuras de datos.

Una de las características es que puede definir nuevas estructuras basadas en las matrices de la librería NumPy. Además, permite leer fácilmente datos de archivos CSV, Excel o base de datos como SQL.

Sumando a lo anterior, Pandas, adiciona algunos tipos de datos. Esos datos son, las series, los dataFrame y los paneles.

Series

Son estructuras similares a los arrays, pero deben ser homogéneas y son inmutables. Al igual que las matrices, usan el índice para identificar al elemento.

Creando series

La sintaxis es muy similar a la experiencia que tenias con matrices al usar Numpy. Para crear series con Pandas, debemos usar el siguiente código.

```
import numpy as np
import pandas as pd

serie=pd.Series([1,4,54],dtype="int32")
print(type(serie))
print (serie)
```

luego de ejecutar veríamos lo siguiente:

```
<class 'pandas.core.series.Series'>
0     1
1     4
2    54
dtype: int32
```

Creando series desde otras estructuras

Se pueden crear series, tomando otras estructuras como fuente de origen. Por ejemplo, podemos usar listas, matrices o diccionario entre otras cosas.

A continuación, veremos cómo crear listas

Creando series desde un diccionario

```
serie=pd.Series({"Pan":100.5,"Arroz":123.6,"Agua":145.78})
```

Creando serias desde una lista

```
serie=pd.Series([100,45,34,56])
```

Creando series desde una matriz o array

```
import numpy as np
import pandas as pd

matriz_una_dimension=np.array([1,2,3,4])
series=pd.Series(matriz_una_dimension)
print (series)
```

Manipulando series

Accediendo a los datos

Se puede acceder a los elementos de las series, usando el índice como posición. Podemos hacerlo de dos maneras diferentes:

1- series[posición] se indica el índice del valor según su ubicación dentro de la serie
2- series[lista] se puede usar una lista, para indicar mas de un valor para acceder

Funciones adicionales

Rápidamente, haremos un resumen de algunas funciones que podemos usar con las series que tengan valores números:

sum

Devuelve la suma acumulada cuando los datos son numéricos

value_counts

Recuento de valores únicos

min

devuelve el mínimo

max

devuelve el máximo

mean

devuelve la media

Bibliografía

Miller, Curtis. (2016). Hands-On: Data Analysis with Numpy and pandas. (1ed).Packt>. Birmingham

John Hunter, Darren Dale, Eric Firing, Michael Droettboom .(2012). Quick start guide.

https://numpy.org/doc/stable/user/absolute_beginners.html

(2022). numpy.org.

Heras Martinez, jose. (2020). Error cuadrático medio para regresión. Relevado de https://www.iartificial.net/error-cuadratico-medio-para-regresion/#:~:text=El%20Error%20Cuadr%C3%A1tico%20Medio%20es,podremos%20indicar%20el%20resultado%20correcto. (2022). IAartificial.net