Numpy

Biblioteca de Python comúnmente usada en la ciencias de datos y aprendizaje automático (Machine Learning). Proporciona una estructura de datos de matriz que tiene diversos beneficios sobre las listas regulares.

Importar la biblioteca:

import numpy as np

Crear arreglo unidimensional:

my\_array = np.array([1, 2, 3, 4, 5])

Resultado: array([1, 2, 3, 4, 5])

Crear arreglo bidimensional:

np.array( [[‘x’, ‘y’, ‘z’], [‘a’, ‘c’, ‘e’]])

Resultado:

[[‘x’ ‘y’ ‘z’]

[‘a’ ‘c’ ‘e’]]

Mostrar el número de elementos del arreglo:

len(my\_array)

Sumar todos los elementos de un arreglo unidimensional:

np.sum(my\_array)

Obtener el número máximo de los elementos del arreglo unidimensional

np.max(my\_array)

Crear un arreglo de una dimensión con el número 0:

np.zeros(5)

Resultado: array([0., 0., 0., 0., 0.])

Crear un arreglo de una dimensión con el número 1:

np.ones(5)

Resultado: array([1., 1., 1., 1., 1.])

Comando de Python para conocer el tipo del dato:

type(variable)

Ordenar un arreglo:

np.order(x)

Ordenar un arreglo por su llave:

np.sort(arreglo, order = ‘llave’)

Crear un arreglo de 0 a N elementos:

np.arange(n)

Ej.

np.arange(25)

Resultado:

array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,

17, 18, 19, 20, 21, 22, 23, 24])

Crear un arreglo de N a M elementos:

np.arange(n, m)

Ej.

np.arange(5, 30)

Resultado:

array([ 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,

22, 23, 24, 25, 26, 27, 28, 29])

Crear un arreglo de N a M elementos con un espacio de cada X valores:

np.arange(n, m, x)

Ej.

np.arange(5, 50, 5)

Resultado:

array([ 5, 10, 15, 20, 25, 30, 35, 40, 45])

Crear arreglo de NxM:

len(my\_array)

Número de elementos del arreglo:

len(my\_array)

## **Pandas**

Pandas es una herramienta de manipulación de datos de alto nivel, es construido con la biblioteca de Numpy. Su estructura de datos más importante y clave en la manipulación de la información es DataFrame, el cuál nos va a permitir almacenar y manejar datos tabulados observaciones (filas) y variables (columnas).

Importar la biblioteca:

import pandas as pd

Generar una serie con Pandas:

pd.Series([5, 10, 15, 20, 25])

Resultado:

0 5

1 10

2 15

3 20

4 25

Crear un Dataframe:

lst = [‘Hola’, ‘mundo’, ‘robótico’]

df = pd.DataFrame(lst)

Resultado:

0

0 Hola

1 mundo

2 robótico

Crear un Dataframe con llave y dato:

data = {‘Nombre’:[‘Juan’, ‘Ana’, ‘Toño’, ‘Arturo’],

‘Edad’:[25, 18, 23, 17],

‘Pais’: [‘MX’, ‘CO’, ‘BR’, ‘MX’] }

df = pd.DataFrame(data)

Resultado:



Leer archivo CSV:

pd.read\_csv(“archivo.csv”)

Mostrar cabecera:

data.head(n)

Mostrar columna del archivo leído:

data.columna

Mostrar los últimos elementos:

data.tail()

Mostrar tamaño del archivo leído:

data.shape

Mostrar columnas:

data.columns

Describe una columna:

data[‘columna’].describe()

Ordenar datos del archivo leído:

data.sort\_index(axis = 0, ascending = False)

## **Scikit Learn**

Scikit Learn es una biblioteca de Python que está conformada por algoritmos de clasificación, regresión, reducción de la dimensionalidad y clustering. Es una biblioteca clave en la aplicación de algoritmos de Machine Learning, tiene los métodos básicos para llamar un algoritmo, dividir los datos en entrenamiento y prueba, entrenarlo, predecir y ponerlo a prueba.

Importar biblioteca:

from sklearn import [modulo]

División del conjunto de datos para entrenamiento y pruebas:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

Entrenar modelo:

[modelo].fit(X\_train, y\_train)

Predicción del modelo:

Y\_pred = [modelo].predict(X\_test)

Matriz de confusión:

metrics.confusion\_matrix(y\_test, y\_pred)

Calcular la exactitud:

metrics.accuracy\_score(y\_test, y\_pred)