Software para Validación de Modelos de Reconstrucción de Series de Tiempo: Un Avance

Release 0.0.1

Ariel Cerón González

June 11, 2020

Table of Contents

CHAPTER 1

Documentación del Módulo Principal

Este documento es un borrador de la documentación del código generado hasta el momento durante el periodo de Servicio Social.

Nota: El generador PDF rynohtype de Sphinx no aún no incluye idioma español.

class python_code.recons.cmpy2.**Reconstruir** (*data: list*, *date: list*, *name: str = 'default'*, *vec: list = [0.6, 0.2, 0.2]*)

Constructor principal de la clase.

name

El nombre de los datos a tratar

Type str

df

Variable que almcena un DataFrame de los datos y su tratamiento

Type padas.DataFrame

len

Cantidad de elementos en una columna del df

Type int

nCol

Cantidad de columnas del df

Type int

leer (*self*, *data: list*, *date: list*, *name: str = 'default'*, *vec: list = [60, 20, 20]*)

Prints the animals name and what sound it makes

añadirCol (*name: str*, *data: list*, *flag: bool = True*)

Agrega una coluna al DataFrame.
Agrega una columna de nombre name con los datos data.

contar (*nameCol: str*, *typ: str*) → list

Devuelve una lista con posiciones iniciales y cantidad de elementos.
Devuelve una lista de posiciones iniciales y cantidad de elementos continuos que le siguen a la posición inicial. Se puede contar datos ("d") o huecos en una columna ("h").

Parámetros

- **nameCol** (*str*) – Nombre de la columna a contar
- **typ** (*str*) – Indica lo que se está buscando, huecos o datos.

Devuelve Regresa un tripleta de datos, la primera posición es una lista de posiciones y tamaños, la segunda posición refleja el total acumulado y la tercera el valor máximo.

Tipo del valor devuelto

tuple

Ejemplo

namecol = "x_i" >>> r.contar(namef<col,"h") ([[0, 1]], 1, 1) >>> r.contar(namecol,"d") ([[0, 3], [4, 1]], 4, 3)

creaDF (*data: list*, *date: list*, *vec: list*)

Inicia la estructura principal del objeto.

Apoyado de los datos de entrada, se crean las primeras tres columnas del objeto DataFrame con el que va a funcionar el objeto Reconstruir.

Parámetros

- **data** (*list*) – Lista de datos numéricos a tratar
- **date** (*list*) – Lista de asocia a cada dato, un tiempo representativo para los datos
- **vec** (*list*) – Porcentaje de distribución en el conjunto de entrenamiento, prueba y verificación

describe (*nameCol: str = 'x_i'*)

Representa gráficamante la proporción de los datos.

Imprime tres gráficos:

1. Gráfico de pastel entre la relación de datos y huecos.
2. Frecuencia de los conjuntos de datos
3. Frecuencia en los conjuntos de huecos

Parámetros **nameCol** (*str, optional*) – Nombre de la columna a representar, by default «x_i»

Ejemplo

```
>>> r.describe()
```

generadorHuecos (*denH: list*, *denD: list*, *inicialTyp: str = 'd'*) → list

generadorHuecos A partir de una lista de densidades de huecos y de densidades de datos, se genera un vector de huecos y datos

Parámetros

- **nombreCol** (*str*) – Nombre de la columna a generar
- **denH** (*list*) – Densidad de huecos
- **denD** (*list*) – Densidad de datos

Devuelve Regresa una lista con unos y nan

Tipo del valor devuelto

list

guardar ( )

Actualiza el archivo csv o se crea un nuevo archivo

hist (*nameCol: str*, *typ: str*, *plot: bool = True*) → tuple

Devuelve una tupla de datos e imprime un histograma.

Devuelve una tupla de datos, el primer elemento es una lista de bins y el segundo representa las alturas para cada dato. Además imprime un histograma con estos dos datos

Parámetros
- **nameCol** (*str*) – Columna sobre la cual se realiza el histograma
- **typ** (*str*) – Se define sobre qué se desea hacer el histograma, datos o huecos
- **plot** (*bool, optional*) – Se puede negar la aparición de graficas, by default True

Devuelve Devuelve una tupla de datos. La primera posición es un vector de bins, la segunda lista son alturas para los bins.

Tipo del valor devuelto

tuple

Ejemplo

```
>>> bins,h = r.hist('x_i','h',False)
>>> bins
[0, 1]
>>> h
[0, 1]
>>> bins,h = r.hist('x_i','d',False)
>>> h
[0, 1, 0, 1]
>>> bins
[0, 1, 2, 3]
```

indices (*nameCol: str*, *typ: str*) → list

Devuelve una lista de indices de datos o huecos.

Devuelve una lista de indices analizando los datos que se encuentran en nameCol y buscando huecos o datos, según typ.

Parámetros
- **nameCol** (*str*) – Nombre de la columna a analizar
- **typ** (*str*) – Tipo de dato a identificar

Devuelve Lista de indices.

Tipo del valor devuelto

list

Ejemplo

```
>>> r.indices('x_i','h')
[3]
>>> r.indices('x_i','d')
[0, 1, 2, 4]
```

reconstruyeKNN (*nameCol: str*, *h: int*, *m: int*, *tau: int*, *k: int*, *S: str = 'E'*)

reconstruyeKNN reconstruye los huecos de tamaño definido usando distancias y k primeros elementos

Parámetros
- **nameCol** (*[type]*) – Nombre de la columna con huecos
- **h** (*[type]*) – tamaño del hueco
- **m** (*[type]*) – Cantidad de elementos
- **tau** (*[type]*) – frecuencia
- **k** (*[type]*) – k primeros elementos

- **S** (*str, optional*) – conjunto que se ocupa, by default "E"

validaIndices (*vector: list*, *indices: list*)

validaIndices Valida que el vector de entrada tenga datos útiles en las posiciones de la mascara

Parámetros
- **vector** (*list*) – Datos a verificar
- **mascara** (*list*) – Lista de posiciones

vecS (*vec: list*) → list

Devuelve una lista dividida en tres grupos.

Dado un vector de porcentaje (vec) se crea una lista del tamaño del objeto, separada en tres grupos según el porcentaje, a saber ["E","P","V"]

Parámetros **vec** (*list*) – Lista de porcentajes para cada grupo, la suma de los elementos debe dar 1.

Devuelve Lista segmentada

Tipo del valor devuelto

list

Ejemplo

```
>>> v = [0.4,0.2,0.4]
>>> vp = r.vecS(v)
>>> vp
['E', 'E', 'P', 'V', 'V']
```

verificaDatos (*nameCol*, *S: str*)

verificaDatos regresa los datos pertenecientes a un conjunto definido

Parámetros
- **nameCol** (*[type]*) – Nombre de la columna a buscar
- **S** (*str*) – Tipo de dato a buscar

python_code.recons.cmpy2.**creaMascara** (*h: int*, *tau: int*, *m: int*) → list

Devuelve una lista binaria (0,1) siguiendo una frecuencia tau para una cantidad m de elementos verdaderos.

Devuelve una lista binaria, de ceros y unos, en el centro del arreglo se coloca una lista de ceros de tamaño h y a los extremos del vector se coloca una lista binaria de tamaño n, que cumple la relación

n = 2(m) - 1

La lista devuelta es de tamaño 2n + h

Parámetros
- **h** (*int*) – Cantidad de datos centrales
- **tau** (*int*) –
- **m** (*int*) –

Devuelve Lista binaria de tamaño 2n + h

Tipo del valor devuelto

list

python_code.recons.cmpy2.**generateRandom** (*cummDist: list*) → int

Genera una variable aleatoria entre [1,len(cummDist)]

cummDist: list
Valid Cummulative Distribution

ts: int
random integer between [1,len(cummDist)]

```
>>> np.add(1, 2)
3
```

python_code.recons.cmpy2.**simulateFailures** (*n: int*, *PD_1: list*, *PD_2: list*) → list
Esta funcion simula fallas en la captura de datos.

n : *int*
Length of the time series

PD_1: list
Cummulative distribution for failures (holes).

PD_2: list
Cummulative distribution for continuum data.

h : *list*
sequence of simmulated failures

```
>>> n = 5
>>> f1 = [5,5]
>>> f2 = [5,5]
>>> H = simulateFailures(n,f1,f2)
>>> H
[0.0, 0.0, 1.0, 0.0, 1.0]
```

python_code.recons.cmpy2.**toStr** (*value: int*) → str
Realiza una transformación sobre un valor numérico.
Devuelve un carácter perteneciente al conjunto S = [“E”,”P”,”V”], en función de un número incluido en el conjunto N= [0,1,2] con la siguiente definición:
Sea s en S, un carácter del conjunto S y n en N, un número en el conjunto N

f(n) = E si n = 0,
P si n = 1, V si n = 2

Parámetros **value** (*int*) – Valor numérico a transformar

Devuelve Carácter asociado al número
Tipo del valor devuelto

str

Ejemplo

```
>>> import cmpy2 as cp
>>> n = 0
>>> s = cp.toStr(n)
>>> print("f({}) = {}".format(n,s))
f(0) = E
```

p

V