Reporte 2

Algoritmos para facilitar la evaluación de modelos de reconstrucción

Elaborado por Ariel Cerón González

En este reporte se describe como se utilizan la clase del módulo desarrollado el cual contiene una estructura de datos, y funciones que hacen uso de esta estructura para el análisis descriptivo de valores faltantes de series de tiempo, preparación y organización de los datos para su posterior análisis de desempeño de algoritmos de reconstrucción.

La estructura de datos consiste en una tabla, donde una medicion de la serie de tiempo se encuentra en una fila en una determinada columna. Además, la tabla contiene una serie de columnas que guardan una serie de vectores con valores en el tiempo útiles para la evaluacion de modelos de reconstrucción.

Las columnas de esta tabla son:

date: Tiempo

x_i: Variable dependiente a estudiar.

S: conjunto de datos (entrenamiento, prueba o validación)

Y^{null}: Huecos generados artificialmente

Y': Reconstrucción

[Poner los nombres de las columnas correctos, Cual mas ?]

Para utilizar las funciones de visualización y reconstrucción (el cual incluye las funciones para la simulación de fallas) es necesario primero cargar los paquetes Pandas, datetime y el módulo desarrollado *cmpyw2.py*.

```
#Se importa la libreria pandas para observar el archivo de tiempo
#Libreria para manipular variables de tiempo
#Importación de la libreria desarrollada
#Libreria que permite crear series de datos con caos
import pandas as pd
import datetime
import cmpy2
from lorenz import *
```

Una vez cargados los paquetes y módulos para utilizarlos primero es necesario tener una lista de datos que será nuestra serie de tiempo. A manera de ejemplo construimos una serie de tiempo sintética de pocos datos con valores nulos.

```
#Lista de datos con valores nulos
data = [1,2,3,np.nan,np.nan,4,np.nan,5,6,7,8,9]
```

Después generamos un vector de datos con el mismo tamaño del vector data los cuales se utilizaran como índices para acceder de maneara unica a cada fila.

```
# Lista de fechas
date = ['01/01/1986 01:00:00', '01/01/1986 02:00:00', '01/01/1986 03:00:00',
'01/01/1986 04:00:00', '01/01/1986 05:00:00', '01/01/1986 06:00:00', '01/01/1986
07:00:00', '01/01/1986 08:00:00', '01/01/1986 09:00:00', '01/01/1986 10:00:00',
'01/01/1986 11:00:00', '01/01/1986 12:00:00']
```

Generamos un vector de tres elementos los cuales indican que proporcion de los valores vamos a designar a conjunto de entrenamiento, prueba y validación.

```
#Vector de clasificación v = [0.6,0.2,0.2], la suma debe ser uno.
clasificacion = [0.6,0.2,0.2]
```

Una vez que tenemos los vectores requeridos procedemos a inicializar el objeto cmpy2 con la función reconstrucción.

```
#Creación del objeto se alimenta con el vector de datos, el vector de fechas y el vector de clasificación (opcional)
prueba = cmpy2.Reconstruir(data,date,clasificacion)
Datos agregados
Datos agregados
Datos agregados
Datos importados
```

Podemos verificar la tabla construida con prueba.df

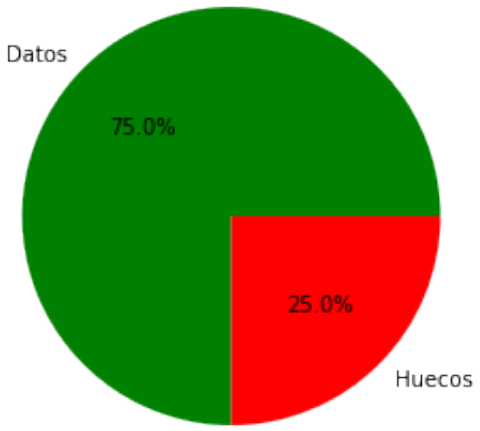
```
prueba.df
```

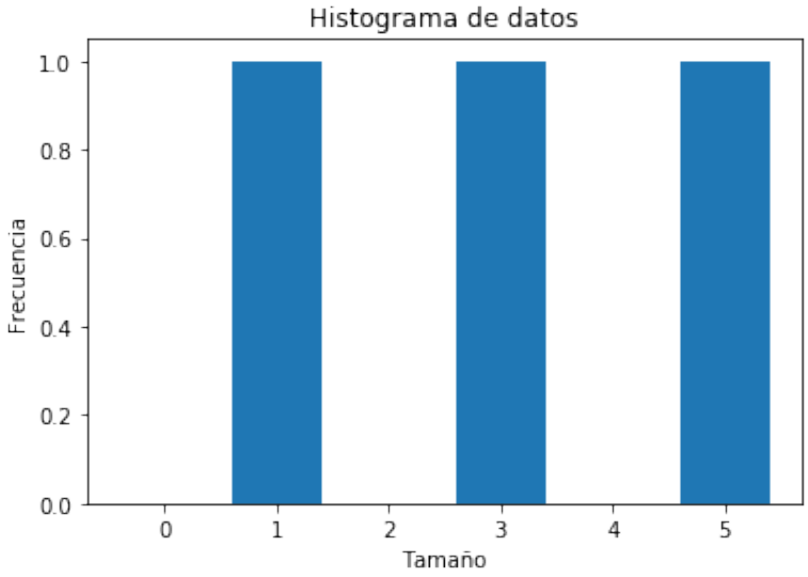
La cual tendra la siguiente información

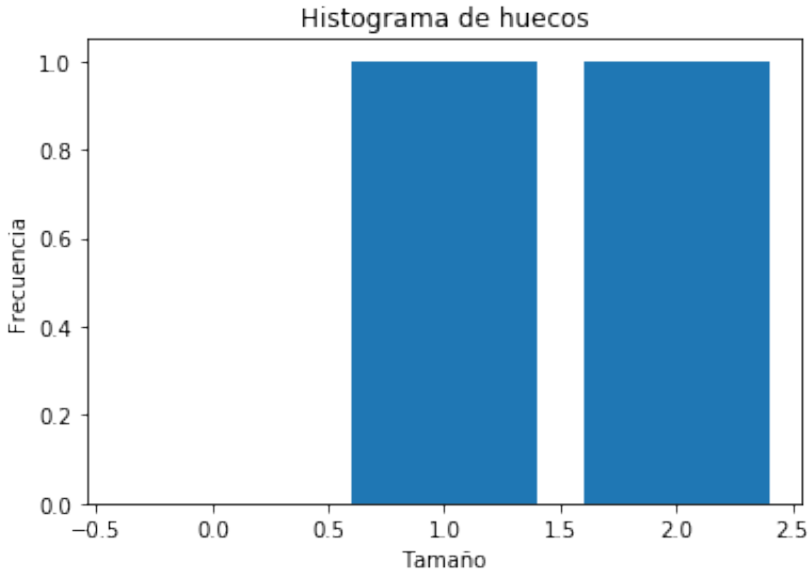
```
date  x_i S
0 1986-01-01 01:00:00 1.0 E
1 1986-01-01 02:00:00 2.0 E
2 1986-01-01 03:00:00 3.0 E
3 1986-01-01 04:00:00 NaN E
4 1986-01-01 05:00:00 NaN E
5 1986-01-01 06:00:00 4.0 E
6 1986-01-01 07:00:00 NaN E
7 1986-01-01 08:00:00 5.0 E
8 1986-01-01 09:00:00 6.0 P
9 1986-01-01 10:00:00 7.0 P
10  1986-01-01 11:00:00 8.0 P
11  1986-01-01 12:00:00 9.0 V
```

Para poder visualizar los datos como la proporción y distribución de los datos faltantes se utiliza la función describe().

```
prueba.describe()
```


Datos
75.0%
25.0%
Huecos


Histograma de datos
Frecuencia
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
Tamaño



La primera gráfica nos indica la proporción en general de datos faltantes con respecto al total de datos.

La segunda gráfica es un histograma que describe el número de secuencias de algún tamaño sin datos faltantes. Por ejemplo, hay una secuencia de tamaño 1, una secuencia de tamaño 3 y una de tamanño 5.

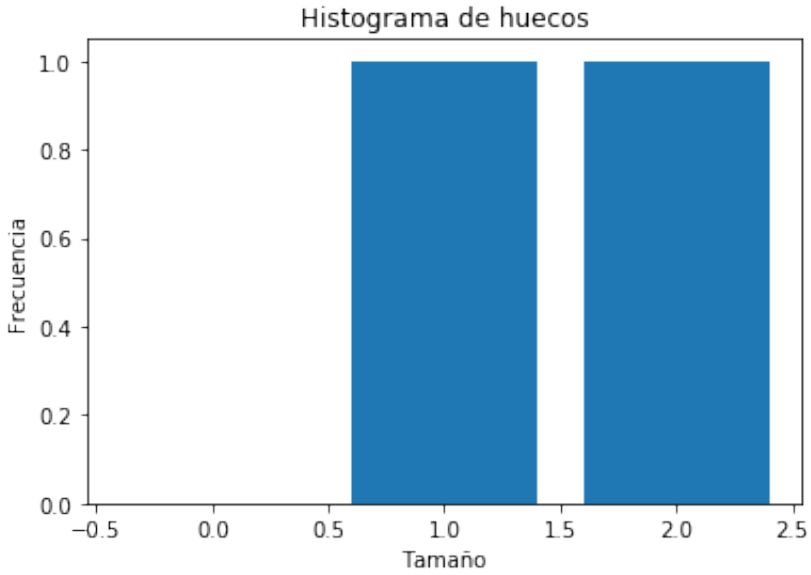
La tercera gráfica es un histograma que describe la frecuencia con la que aparece secuencias de datos faltantes de algún tamaño. La manera de leer los datos es igual a la segunda gráfica.

Es importante señalar que describe genera dos tablas, la tabla de secuencia datos faltantes (huecos) y secuencia de datos no faltantes.

Estas tablas tienen la forma... [falta añadir la descripción de las tablas de los huecos]

Es posible utilizar la función `histograma` la cual nos produce un histograma tal como se muestra enseguida.

```
n, bins = prueba.hist('x_i','h')
print("Posición: ",n)
print("Peso:", bins)
```



Funciones de Reconstrucción

Para probar los algoritmos de reconstrucción es necesario tener datos con los cuales poder comparar. Por lo tanto, una estrategia es simular datos faltantes en la serie de tiempo los cuales deben ser reconstruidos y al final comparar la reconstrucción (o estimación) de algún algoritmo con el la medición real.

Para simular datos faltantes en la serie de tiempo, se utiliza una columna especial para no alterar los datos originales. El generador de datos faltantes funciona con distribuciones de probabilidad. Estas pueden ser teóricas o empíricas. Por ejemplo para generar una secuencia aleatoria de número primero necesitamos definir histogramas de frecuencias de aparición de secuencias de algún tamaño ya sea de datos faltantes o datos no faltantes.

```
# Frecuencia de tamaño de secuencia de datos faltantes
denH = [1,3,5,7,42,1]

# Frecuencia de tamaño de secuencia de datos faltantes
denD = [4,2,1,4,1,2]
```

El generador de datos faltantes se encarga de convertirlos en una función de distribución de probabilidad cada uno, simular los datos faltantes y agregarlos en una nueva columna. La columna creada toma el nombre **HGV#**, donde # es sustituido por algún número? variando en función de la posición de la columna que tome.

```
#Nueva columna
nuevaCol = prueba.generadorHuecos(denH,denD)
```

La tabla queda de la siguiente forma

```
  date  x_i S HGV3
0 1986-01-01 01:00:00 1.0 E 1.0
1 1986-01-01 02:00:00 2.0 E 1.0
2 1986-01-01 03:00:00 3.0 E 1.0
3 1986-01-01 04:00:00 NaN E 1.0
4 1986-01-01 05:00:00 NaN E NaN
```

Una vez que se tiene esta tabla, es posible utilizarla para probar algún algoritmo de reconstrucción.

Particularidades de la serie de tiempo de PM 2.5

En esta sección se aplica la clase desarrollada para describir la calidad de la serie de tiempo de acuerdo a sus datos faltantes.

La serie de de PM 2.5 fué capturada en la estación [cual?] la zona metropolitana de Ciudad de México. Los datos son registrados cada 60 minutos, desde el 1 de enero de 1986. hasta el 30 de Diciembre del 2018 . Esta serie de tiempo es típica de los sistemas de monitoreo los cuales no están libres de errores o fallas técnicas.

[Incluir una grafica con plt.plot(y)]

[Hacer el ejercicio anterior pero con estos datos]

Notas

Avances en la documentación del codigo se encuentra en el documento adjunto reconstrucción.pdf

También se encuentra en la siguiente liga en github.

Anexo (esto a mi parecer es mucho detalle para lo que queremos comunicar. Puedes usar esto para complementar los comentarios del código del programa)

Para reconstruir los huecos del documento se construyeron diferentes métodos que permiten crear datos que permitan realizar pruebas estadísticas, nuevos huecos siguiendo una distribución definida o agregar columnas a los datos.

1. **añadirCol**: Agrega una columna a la matriz.
2. **vecS**: Agrega una columna a la matriz que representa la pertenencia a valor del conjunto S.
3. **contar**: Devuelve una matriz de datos que contiene la posición inicial y la cantidad de elementos del tipo hueco o dato.
4. **indices**: Devuelve una matriz de datos que contiene la posición inicial y la cantidad de elementos del tipo hueco o dato.
5. **generadorHuecos**: Agrega una columna de huecos (o datos) definida por una distribución.
6. **validaIndices**: Verifica que se cumpla una condición de existencia en función de una mascara.
7. **verificaDatos**: Verifica que se cumpla una condición de pertenencia a un elemento del conjunto S.

Reconstrucción usando la estructura (Esto lo dejamos para el último reporte que incluye el algoritmo de reconstrucción)

La mejor forma de aprovechar la estructura es creando una función de reconstrucción que tome como entrada los parámetros *objeto* y *nombreCol* (además de los que el modelo de reconstrucción necesite). El primero representa a la estructura de datos y el segundo representa la columna en la que se almacenan los datos que se desean construir o visualizar.

Para ejemplificar el uso de la estructura se desarrollo una función **vecinosCercanos** que intenta reconstruir los datos faltantes de una serie de tiempo considerando el valor de los elementos circundantes a un punto y la métrica euclidiana.

Para su implementación se utilizó unicamente métodos de la matriz y trabajando con las reglas del álgebra (la estructura puede ser trabajada con las reglas del álgebra vectorial).

Con los métodos de la estructura es fácil añadir los valores creados por la función en una nueva columna de datos.