Teoria de la Informacion

TRABAJO PRÁCTICO ESPECIAL 2021

UNCPBA – FACULTAD DE CIENCIAS EXACTAS
WAGNER JULIAN - JULIANWAGNERR10@GMAIL.COM- LU:248964
ALVAREZ MAXIMILIANO -MAXI25294@GMAIL.COM- LU:249175
TALU BERNABE- BERNI.TALU@HOTMAIL.COM-LU:248830
GRUPO: 4
AYUDANTE: IGNACIO LARRABIDE

Introducción

El mundo de las criptomonedas es muy volátil por lo que se tratará, contando con la información de las cotizaciones históricas de las dos criptomonedas con más volúmen, de relacionarlas a través de una nueva señal analizando sus precios. Estimaremos la variación de una a partir del comportamiento de la otra, y se almacenarán los datos, tanto los dados como los obtenidos, de una manera más eficiente.

En este trabajo se implementará el uso del muestreo computacional para llevar a cabo lo solicitado en determinadas situaciones, y poder corroborar la veracidad del cálculo analitico, expresado mediante este mecanismo.

Desarrollo y Análisis

1. Fuentes de información y muestreo computacional:

Matriz de pasaje para ambas monedas: Para el cálculo de las matrices de pasaje, se plantea un código en lenguaje java utilizando los archivos brindados por la cátedra, con las cotizaciones correspondientes para cada moneda. En el tiempo t=0, suponemos que el precio se mantiene con respecto a la cotización anterior, para de esta forma poder iniciar la simulación. Cada moneda registra 1000 cotizaciones.

- **Matriz de pasaje para BTC:**

□	Baja(0)	Mantiene(1)	Sube(2)
Baja(0)	0.40732265	0.45454547	0.4611111
Mantiene(1)	0.011441648	0.09090909	0.025925925
Sube(2)	0.5812357	0.45454547	0.51296294

- **Matriz de pasaje para ETH:**

□	Baja(0)	Mantiene(1)	Sube(2)
Baja(0)	0.34285715	0.21639344	0.37203166
Mantiene(1)	0.18730159	0.47868854	0.26121372
Sube(2)	0.46984127	0.30491802	0.36675462

Pseudocódigo:

```
llenarMatriz(){
    while(haya cotizaciones por recorrer) {
        if(el precio se mantiene)
            s=1
        else
            if(el precio subió)
                s=2;
            else
                if(el precio disminuye)
```

```
                    s=0;
        }

        for(todas las filas de matriz){
                for(todas las columnas de matriz)
                        transiciones[fila][col] = (transiciones[fila][col]/cant Iteraciones[col]);
        } }
```

Inciso 1.b:

Pseudocódigo:

```
        Autocorrelación():
        correlación[ ] //Arreglo solución
        Arreglo [ ] //Arreglo con los valores de eth/btc dependiendo el caso
        emisiones
                for (t1(1) hasta t2(50))
                        for (cada posición en el archivo)
                                sumar en la posición t1 del arreglo correlación, el valor del arreglo de
                                valores en la posición del segundo iterador multiplicado por el valor del
                                arreglo en esa posición+t1, y todo eso dividido por la cantidad de
                                emisiones - t1.
```

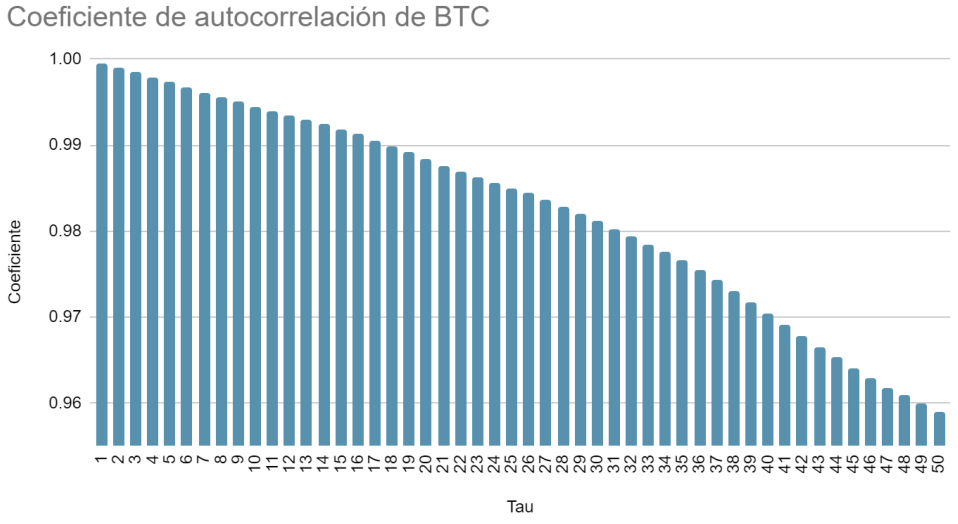
Inciso 1.c:

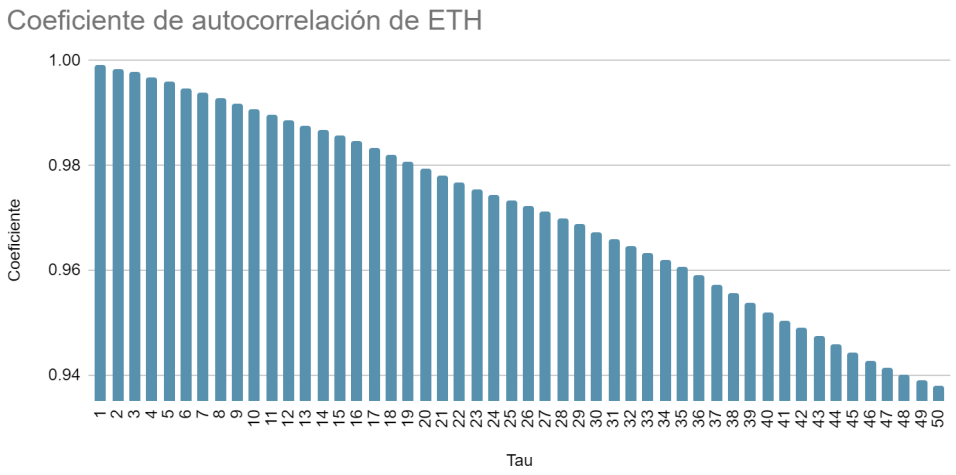
Pseudocódigo:

```
correlacionCruzada(arregloA,arregloB,tiempo){
        correlación[] //se define un arreglo que representa la solución de correlación para cada tiempo
        determinado
        for(0 hasta 999-tiempo)
                correlación[índice] = (arregloA en iterador multiplicado por arregloB en
                iterador+tiempo)/(cantEmisiones - tiempo)
        Se retorna correlacion.
}
```

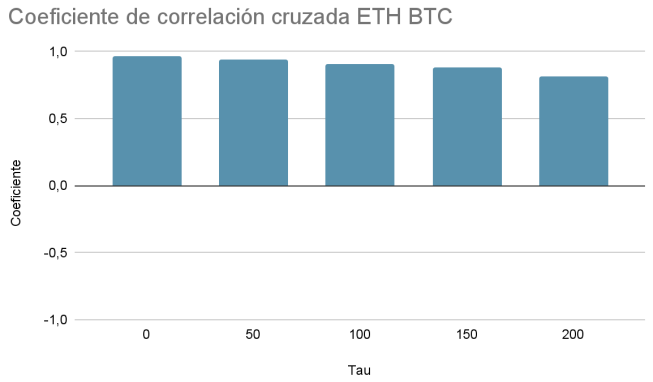
Inciso 1.d:

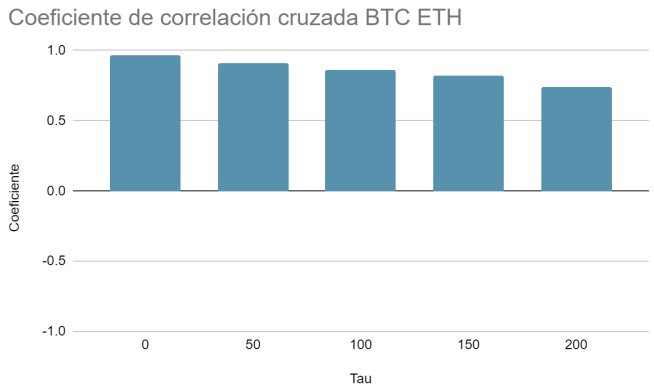
Para un mejor análisis de la autocorrelación y la correlación cruzada, calculamos los coeficientes de los mismos y de esta forma es posible ver su comportamiento.





Como se puede observar en ambos gráficos, para un tau menor hay mayor coeficiente de autocorrelación, por lo tanto indica la relación que tienen entre valores contiguos. Tiene mucho sentido debido a que el crecimiento de las cotizaciones es paulatino, de ese modo tomando un tau = 1, si comparamos de uno en uno, su comportamiento con el siguiente va a ser semejante y la autocorrelación va a ser muy alta. En cambio con tau = 50, podemos ver que el coeficiente de autocorrelación es menor, por lo tanto llegamos a la conclusión de que la emisión de símbolos en t1 con la emisión en t1+50 va a tener una menor relación respecto a los demás tau.





Para el caso de la correlación cruzada ocurre algo muy similar a lo explicado anteriormente, pero con un menor grado de coeficiente de correlación cruzada para los tau más grandes, ya que la emisión de un valor en un tiempo t+0 con una en el tiempo t+150/200 no tiene tanta relación como la tendría con un tau=0.

2. Codificación y compresión

Inciso 2.a:

```
//Se define una estructura que contiene la cotización con su probabilidad.
for(i =0 hasta cantidad de cotizaciones){
        contador = 0
        if(cotización no existe en la estructura){
                for(j = 0 hasta cantidad de cotizaciones){
                        if(cotizaciones[j] == cotización actual)
                            contador++
                }
                probabilidad = contador/cantidad de cotizaciones
                se agrega cotización con su probabilidad a la estructura
        }
}
```

De esta forma tendremos la cantidad de veces que se repite cada valor dentro del archivo sobre el total de emisiones.

Inciso 2.b:

Para formar las cadenas de símbolos se utiliza un algoritmo recursivo, siguiendo las ramas izquierdas se agrega un 0 a la cadena y siguiendo la rama derecha un 1.

```
Codificar(Nodo raíz, string cadena)
        if(raiz.izq == null & raiz.der == null)
                imprimir(raiz.precio + cadena)
codificar(raiz.izq,cadena + '0')
codificar(raiz.der, cadena + '1')
```

Para recorrer y formar el árbol primero se crean las hojas con los valores y luego se ingresan dentro de una cola de prioridad ordenados por frecuencia, luego;

```
while (cola.size() > 1){
        nodo x = cola.recuperarPrimero()
        nodo y = cola.recuperarPrimero()
        nodo nuevo;
        nuevo.setProba(x.prob()+y.prob())
        nuevo.setPrecio(-1)
        nuevo.setIzq(x)
        nuevo.setDer(y)
        raiz = nuevo;
        cola.agregar(nuevo) }
```

Inciso 2.c:

Para llevar a cabo lo requerido en el enunciado, representamos mediante muestreo computacional, el método RLC sin pérdida de información.Como los archivos de texto para ambas monedas no presenta muchos casos en los que una cotizacion se repite muchas veces seguidas, la salida de nuestro método nos muestra un código levemente comprimido.

Somos conscientes de que, para este caso, es un método poco conveniente en relación a las pocas secuencias de símbolos repetidos, pero asegura no perder información.
Pseudocódigo RLC:

```
j=inicio del arreglo;
datosRLC //arraylist para representar la salida.
for(i=1 hasta 999){
    if(arrCotizaciones[i] == arrCotizaciones[j])
            repeticiones++
    else
            se agregan a datosRLC la cotización y sus repeticiones seguidas.
            repeticiones = 1;
            j = i;
}
se retorna datosRLC.
```

Inciso 2.d

Partiendo de que a cada valor de los archivos principales, cabecera de Huffman, etc. se los considerará con un tamaño de 2 bytes.

HUFFMAN SEMI-ESTÁTICO

Calculo de tasa de compresión para archivo ETH:

Tamaño original: 1000 cotizaciones * 2 Bytes = 2000 Bytes
Huffman: [Cabecera][Codificación en bits del mensaje] = [297*16 bits]+[6703 bits]/8 = 1.431,875 Bytes
Tasa de compresión = Tamaño original / Tamaño comprimido
Tasa de compresión ETH: 1.396769969445657 → 1,39 : 1

Calculo de tasa de compresión para archivo BTC:

Tamaño original: 1000 cotizaciones * 2 Bytes = 2000 Bytes
Huffman: [Cabecera][Codificación en bits del mensaje] = [1657*16 bits]+[9591 bits]/8 = 4.512,875 Bytes
Tasa de compresión = Tamaño original / Tamaño comprimido
Tasa de compresión BTC: 0.4431764673295848 → 0,44 : 1

Se observa que realizando el algoritmo de Huffman para ETH hay una mejoría debido a que hay muchos valores repetidos, esto hace que la cabecera sea más pequeña y también afecta a que a la hora de hacer el árbol haya menos ramas, por ende las cotizaciones codificadas a nivel bit sean más pequeñas en longitud y como conclusión ocupen menos espacio.

Para el BTC la tasa es muy chica, esto se debe correspondiendo con la justificación anterior a que no hay tantos valores repetidos como en el ETH, esto hace que la cabecera ocupe mucho más espacio (cada valor no repetido con su probabilidad) y se incrementa la cantidad de ramas (también por cada valor no repetido), lo que aumenta significativamente la extensión en bits.

RLC

Se considera 2 Bytes para cotización, tanto para código original como código comprimido, y 2 Bytes para repeticiones de cotización para código comprimido.

Calculo de tasa de compresión para archivo ETH:

Tamaño original: 1000 cotizaciones * 2 bytes = 2000 Bytes
Tamaño comprimido: 22.240 bits → 2780 Bytes.
Tasa de compresión = Tamaño original / Tamaño comprimido
Tasa de compresión ETH: 0,7194244604316546 → 0,71 : 1

Calculo de tasa de compresión para archivo BTC:

Tamaño original: 1000 cotizaciones * 2 bytes = 2000 Bytes
Tamaño comprimido: 31.372 bits → 3921 Bytes.
Tasa de compresión = Tamaño original / Tamaño comprimido
Tasa de compresión BTC: 0.5112474437627812 → 0,51 : 1

Por lo tanto, queda demostrado que el tamaño del código comprimido es mayor al tamaño del código original ya que la tasa obtenida es inferior a 1, lo cual quiere decir que la compresión no retornó un código reducido sino que al contrario, se expande.

Inciso 2.e:

Como los archivos relacionados a cada moneda no contienen un número grande de repeticiones, el rendimiento de RLC se podría mejorar estableciendo un número mínimo de repeticiones para aplicar el método y anteponer un bit flag para indicar si lo que sigue en el código es un par codificado como C-R (Cotización-Repeticiones) o si es una cotización sin codificar. Se podría definir a ese bit con los valores de 1 si se utiliza RLC o 0 si la cotización no se codifica.

Este procedimiento se realiza ya que al no presentarse muchas repeticiones seguidas de cotizaciones, la salida del método en estos casos, generalmente retorna un tamaño de código comprimido mayor al tamaño de código original, el cual es el caso que se presenta en el inciso c).

Entonces, si aplicamos la mejora para un número de repeticiones mínimo de 5, el resultado de la tasa de compresión es de 1,04616, y para un mínimo de 2 se encuentra el mayor valor de compresión, el cual es 1,1155.

Inciso 2.f:

El valor de la entropía para ETH es 6.669716829526338, realizando de la misma manera con Huffman, da un valor de 6.702999999999991. Por lo tanto, para observar el rendimiento,

$$\frac{H}{L} = 0,995034585935603$$

Para el caso del Btc, la entropía es 9.585063625675984 y para Huffman es de 9.590999999999864

$$\frac{H}{L} = 0,999381046918922$$

Como conclusión se puede observar que el rendimiento es muy bueno debido a que en el caso del ETH no solamente se pudo comprimir de forma eficiente el archivo, sino que la longitud calculada es similar a la entropía. En el caso de BTC tiene sentido de que el rendimiento sea de esta forma porque a pesar de que no se pueda comprimir el archivo (la cabecera es totalmente ineficiente), calculando la entropía se iba a alcanzar un número alto por la gran cantidad de cotizaciones diferentes que se tienen, pero, observándose el rendimiento, utilizando Huffman se alcanza casi al mismo valor, es decir que la longitud de codificación a nivel bit de cada cotización diferente de BTC es muy similar al cálculo analítico con su logaritmo en base 2 de cada moneda.

El rendimiento podría mejorarse utilizando en vez de el modelo semi-estático de Huffman, el modelo dinámico (FGK) que si bien es mayor el costo computacional debido a que son algoritmos más complejos, tiene dos grandes beneficios. El primero es que se requiere una única pasada por los datos mientras que en el semi-estático se necesitan dos pasadas (una para armar la tabla y otra para codificar), y la otra mejora es que la distribución de probabilidades se va ajustando a los datos y no se transmite la tabla. Si lo que se busca es mejorar en rendimiento, el modelo dinámico es la solución, si lo que se quiere es que se realice en un menor tiempo o que requiera menos poder computacional, se seguirá con el semi-estático.

3. Canales

Dos fuentes ternarias(tres entradas asociadas a BTC y tres salidas asociadas a ETH) unidas por un canal.

Variables estocásticas

- **X**: tomamos a X como variable estocástica para representar las entradas del canal correspondientes a BTC.
- **Y**: tomamos a Y como variable estocástica para representar las salidas del canal correspondientes a ETH.

Como primera medida lo que hicimos fue calcular la matriz de distribución de probabilidades conjuntas, relacionando ambas monedas para terminar de completar los datos y así definir con exactitud el canal en cuestión. Esto lo logramos gracias a lo implementado en el Ejercicio 1, donde generamos una nueva señal haciendo referencia a si las monedas bajan, mantienen o suben su precio con respecto a la cotización anterior.

El cálculo de esta matriz lo realizamos mediante muestreo computacional donde nuestro método recorre comparando una a una las cotizaciones de ambas monedas para cada tiempo determinado.

Matriz conjunta de probabilidades

P(X,Y)	Baja BTC	Mantiene BTC	Sube BTC	P(Y)
Baja ETH	0.263	0.004	0.048	0.315
Mantiene ETH	0.129	0.014	0.142	0.285
Sube ETH	0.045	0.004	0.331	0.38
P(X)	0.437	0.022	0.521	

Luego, como se conoce la matriz conjunta, se pueden obtener todos los valores de probabilidades necesarios para el resto de las operaciones.

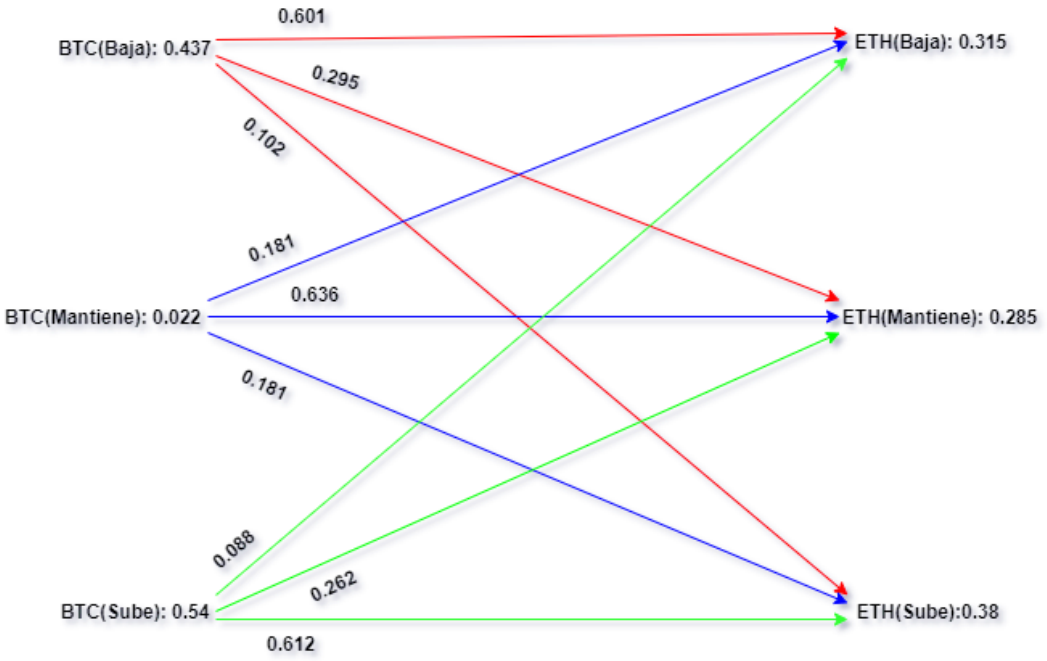
Se calculan P(Y/X) y P(X/Y) para obtener los valores de ruido y perdida:
P(Y/X) = P(X,Y)/ P(X)

P(Y/X) □	Baja BTC	Mantiene BTC	Sube BTC
Baja ETH	0.601	0.181	0.088
Mantiene ETH	0.295	0.636	0.262
Sube ETH	0.102	0.181	0.612
r(x)	1.295	1.307	1.247

P(X/Y) = se llena mediante Bayes → P(X/Y) = (P(X)*P(Y/X))/ P(Y)

P(X/Y) □	Baja BTC	Mantiene BTC	Sube BTC	**π(y)**
Baja ETH	0.833	0.012	0.145	0.700
Mantiene ETH	0.452	0.049	0.478	1.240
Sube ETH	0.117	0.010	0.839	0.641

Por lo tanto, el canal asociado queda de la siguiente manera:



Ruido y Perdida

Para obtener el ruido total del canal, lo que hicimos fue realizar el calculo analitico, utilizando la fórmula de la entropía condicional H(Y/X), calculando previamente el ruido generado en cada entrada del canal, el cual está representado como $r(x)$.

$$H(Y/X) = \Sigma\, p(x).r(x)$$

Por lo tanto, el ruido total del canal es:

$$H(Y/X) = 0.437 * 1.295 + 0.022 * 1.307 + 0.54 * 1.247$$
$$H(Y/X) = 1.268$$

Nuestro resultado nos dio que, el número mínimo de preguntas binarias que se deben realizar, en promedio, para obtener la salida y_i dada la entrada x_j, es aproximadamente una. Viendo el resultado analitico obtenido, y como sabemos que el ruido es la incertidumbre sobre la salida conocida la entrada, tiene sentido que el valor de la entropía no sea un valor muy alto, ya que observando los valores de probabilidades del canal asociado nos muestra que por lo general siempre que se entra por una entrada determinada, la salida es directa, es decir, no se generan muchos casos de cruce. Es evidente que este canal va a generar ruido en las entradas ya que si existen casos de cruce pero el mismo no va a ser un valor elevado por cuestiones de probabilidades como fue mencionado previamente.

Para el caso de la pérdida, hicimos algo similar que para el caso del ruido, donde utilizamos la fórmula de la entropía condicional H(X/Y), calculando previamente la pérdida generada en cada salida del canal$(\pi(y))$.

$$H(X/Y) = \Sigma\, p(y).\pi(y)$$

Entonces, la pérdida total del canal es:

$$H(X/Y) = 0.315 * 0.700 + 0.285 * 1.240 + 0.38 * 0.641$$
$$H(X/Y) = 0.817$$

En este caso, el número de preguntas que se deben realizar, en promedio y como mínimo, para saber por cual entrada viene el dato, dado que se sabe cual es la salida, es un valor cercano a una. Claro está que al ver el canal asociado, este presenta pérdidas en las salidas debido a que en cada salida puede llegar cualquiera de los tres diferentes valores situados en las entradas, pero con la particularidad de que no va a ser muy grande la cantidad de información que se pierde, ya que en la mayoría de los casos si un valor ingresa por x entrada, la salida es directa sin cruzarse, igual a como ocurría en el caso del ruido.

Utilidad del canal

Se calcula la información mutua de manera analitica entre ambas variables estocásticas, es decir, cuanto puedo saber de una conociendo la otra. La manera es la siguiente:

$$H(X,Y) = -0.263 * log(0.263) - 0.004 * log(0.004) - 0.048 * log(0.048) - 0.129 * log(0.129) - 0.014 * log(0.014) - 0.142 * log(0.142) - 0.045 * log(0.045) - 0.004 * log(0.004) - 0.331 * log(0.331)$$
$$H(X,Y) = 2.3773$$

$$H(X) = -0.437 * log(0.437) - 0.022 * log(0.022) - 0.521 * log(0.521) = 1.1331$$
$$H(Y) = -0.315 * log(0.315) - 0.285 * log(0.285) - 0.38 * log(0.38) = 1.5715$$
$$H(X) + H(Y) = 1.1331 + 1.5715 = 2.7046$$

$$I(X,Y) = H(X) + H(Y) - H(X,Y) = 0.3273$$

Entonces, se puede ver que el valor de la suma entre las entropías de cada variable estocástica es mayor, pero no muy superior, que el valor de la entropía conjunta entre ellas. Que ocurra esto es lógico ya que existe un acople entre cada entrada con cada salida y por lo tanto en ningún caso ninguna va a ser una variable independiente, lo cual quiere decir que nunca el valor entrópico entre ellas va a ser un valor igual a cero .
El resultado obtenido al calcular la información mutua entre ellas, quiere decir que como máximo uno se puede ahorrar 0.3273 preguntas al momento de querer conocer el par entrada y salida.

Conclusión

La realización del trabajo nos demostró la posibilidad de analizar fuentes de información desde un punto más simple con la creación de nuevas señales a partir de las dadas para una fácil lectura y comprobar su relación a lo largo del tiempo. Pudimos observar el crecimiento paulatino de las señales principales y la diferencia si se compara entre tiempos cercanos y tiempos lejanos. Nos ayudó a comprimir de forma eficiente dichas fuentes llegando en algunos casos a una mejoría de un 40%. Se pudo confirmar que dependiendo el archivo y la gama de valores que se tienen es preferible usar uno u otro método de compresión.

Finalmente, pudimos notar que al trabajar con un canal con tres entradas y tres salidas y con una gran cantidad de acople entre ellas, el mismo va a presentar casos de ruido y pérdida, más allá de que las probabilidades de cruce no sean muy elevadas.Por lo tanto se pudo ver que el canal en cuestión no es extremadamente malo y que, en relación a los cálculos realizados para obtener la información mutua, tampoco es un canal extremadamente útil.