



**CURSO 2020-2021**

**MÁSTER EN BUSINESS INTELLIGENCE Y DATA SCIENCE**

**MODULO:**

**Customer Analytics**

**Nombre estudiante: Marc Faravelli Rodríguez**

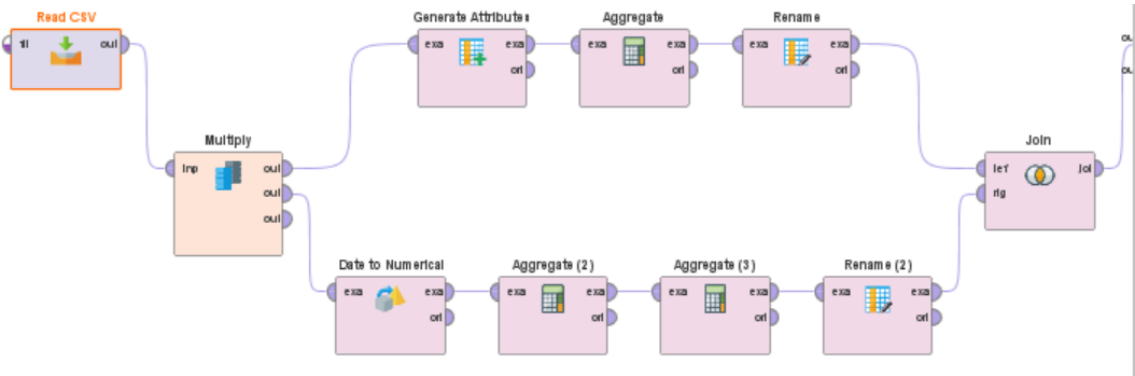
**No.Matricula: 2047104**

**E-mail: marcfaravelli@gmail.com**

Empezaremos con la herramienta “Read CSV” donde procederemos con la carga de los datos. Una vez comprobada la ausencia de valores “missings”, podemos observar que el dataset está compuesto por datos enteros, fecha y polinomiales

▼ orderid	Integer	0	Min 1	Max 1000
▼ clientid	Integer	0	Min 1	Max 300
▼ product	Polynomial	0	Least c (188)	Most a (879)
▼ gender	Polynomial	0	Least male (2140)	Most female (2665)
▼ orderdate	Date	0	Earliest date Jan 2, 2017	Latest date Apr 11, 2017

Después de hacer la comprobación de todo el dataset, empezaremos el calculo de la recencia y de la frecuencia. Primeramente, haremos una copia de nuestro flujo de datos mediante el operador “multiply” para obtener 2 ramas: una para la recencia y el otro para la frecuencia.



En la rama superior, correspondiente a la recencia, con el operador “General attribute” calcularemos dos nuevos atributos: diferencia\_fecha y gasto. Ahora sabremos el tiempo que ha transcurrido desde la ultima vez que se ha comprado un producto y el gasto que ha tenido.

attribute name	function expressions
gasto	Price*Quantity
diferencia_fecha	date_diff(date_now(),orderdate)/(1000*60*60*24)

En el siguiente paso, mediante el operador “Aggregate”, agruparemos los dos atributos generados anteriormente en un único valor. Para esto seleccionaremos el valor mínimo de “diferencia\_fecha” y la suma del “gasto”:

aggregation attribute	aggregation functions
diferencia_fecha	minimum
gasto	sum

También lo agruparíamos por “cliente”:

**Attributes**  
 ✕  


# diferencia\_fecha  
# gasto  
gender  
orderdate  
# orderId  
# Price  
product  
# Quantity

**Selected Attributes**  
 + ✕  

# clientId

En cuanto a la rama de frecuencia, el operador “Multiply” lo conectaremos con el operador “Date to numerical” por tal de convertir el numero real en decimales y que el atributo “ordendate” esté en unidades mes y que el mes esté referenciado en años.

A continuación, mediante el operador “Aggregate” modificaremos el atributo “orderId” con un recuento de las veces que un cliente ha comprado un producto:


**Edit Parameter List: aggregation attributes**  
The attributes which should be aggregated.

aggregation attribute	aggregation functions
orderId	count

**Attributes**  
 ✕  

gender  
# orderdate  
# Price  
product  
# Quantity

**Selected Attributes**  
 + ✕  

# clientId  
# orderId

Con el mismo operador calcularemos también la media de veces que un producto ha sido comprado y obtendríamos la frecuencia de compra:

aggregation attribute	aggregation functions
count(orderId)	average

**Attributes**  
 ✕  

# count(orderId)  
# orderId

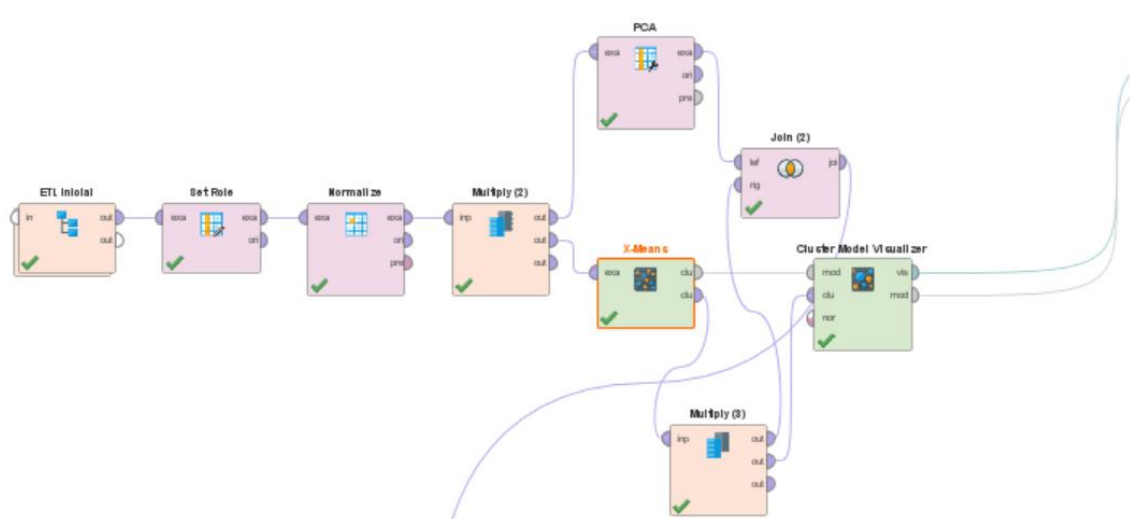
**Selected Attributes**  
 + ✕  

# clientId

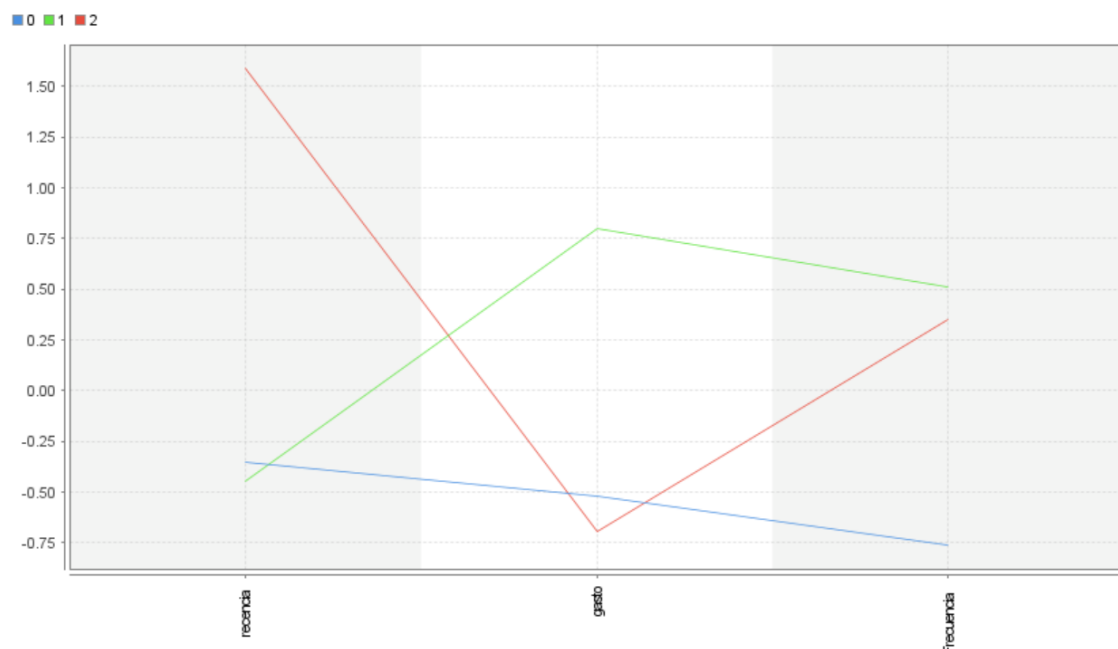
Para acabar el ETL inicial, nos faltará renombrar con el operador “rename” los atributos de recencia y frecuencia y unirlos mediante el operador “join” con el atributo común “clientId”.

Una vez tengamos el ETL, usaremos el operador “Set Role” para eliminar el atributo cliente ya que no lo necesitamos.

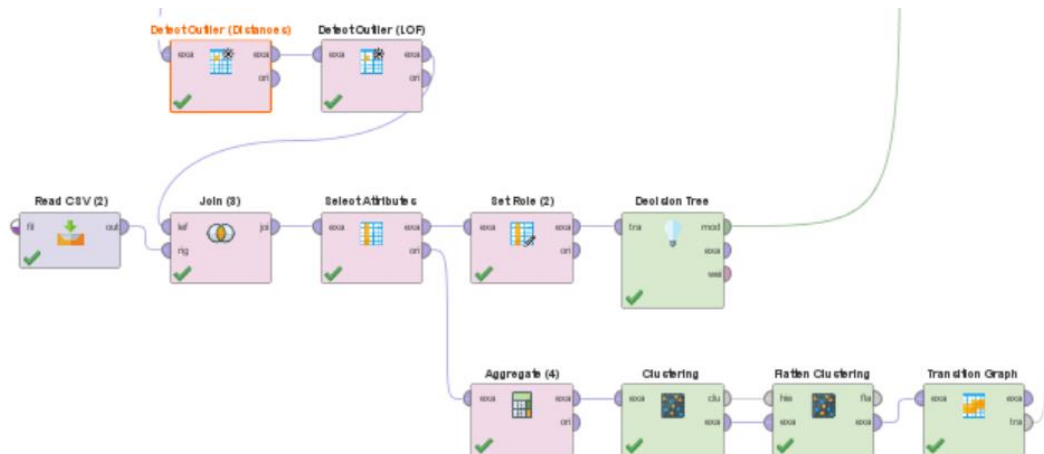
Acto seguido y antes de hacer la PCA, tendremos que normalizar los datos y evitar que el valor absoluto de los atributos determine la importancia numérica en la consecución de la PCA. Mediante el operador “Multiply” conectaremos la PCA y X-Means con el operador “Normalize”, por una parte, y por otra al operador “Join”:



Así nos quedaría el grafico de X-Means:



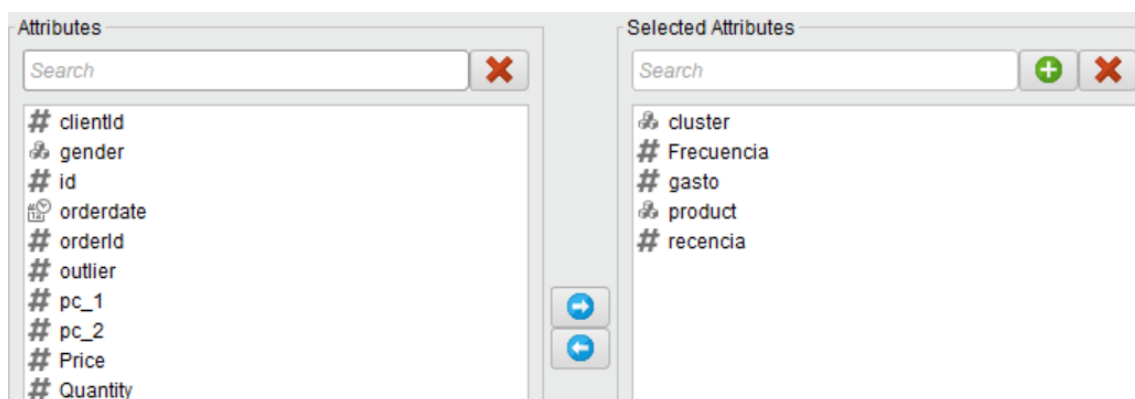
El flujo creado con la PCA y X-Means lo tendremos que conectar con un detector de outliers, basado en la distancia euclidiana. A este le conectaremos otro detector de outliers LOF que no identificará valores atípicos dentro de un conjunto de ejemplos dado en función de factores locales atípicos.



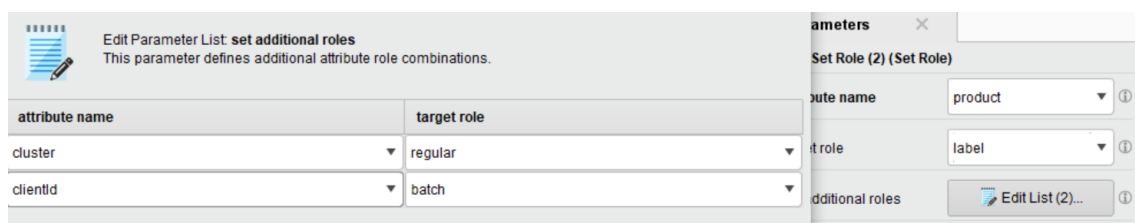
Todo este conjunto, lo uniremos mediante “Join” al documento CSV original con lo atributos “clientId” en común:

Row No.	id	clientId	cluster	outlier	pc_1	pc_2	recencia	gasto	Frecuencia	orderid
1	1	1	cluster_0	1.060	-0.815	-0.251	0.534	-0.211	-0.849	131
2	1	1	cluster_0	1.060	-0.815	-0.251	0.534	-0.211	-0.849	131
3	1	1	cluster_0	1.060	-0.815	-0.251	0.534	-0.211	-0.849	518
4	1	1	cluster_0	1.060	-0.815	-0.251	0.534	-0.211	-0.849	518

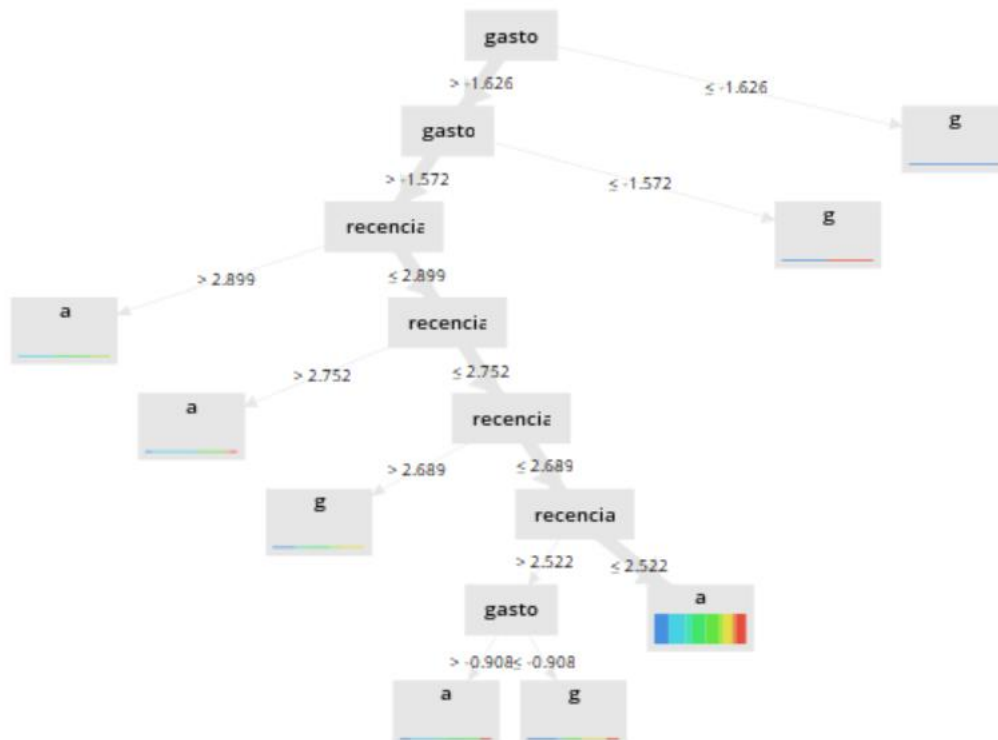
A continuación, mediante el operador “Select attributes”, crearemos un subgrupo con los atributos de cluster, frecuencia, gasto, producto y recencia:



Seguidamente, lo conectamos al operador “Set Role” donde el objetivo será: label = producto y la etiqueta de tipo cluster sea de tipo regular:



Mediante el árbol de decisión podremos observar las relaciones de gasto que hay entre los productos:



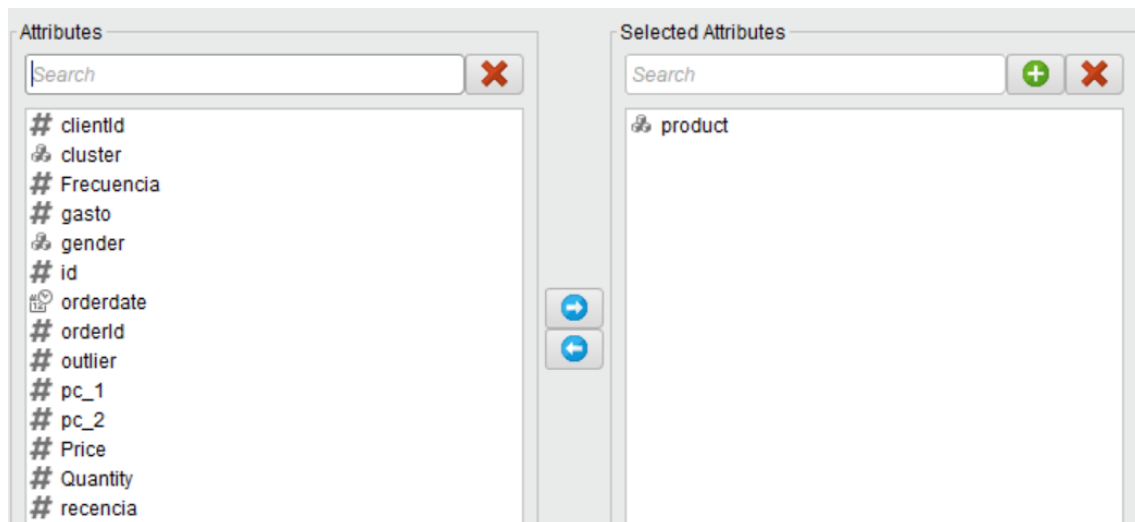
```

gasto > -1.626
|   gasto > -1.572
|   |   recencia > 2.899: a {g=0, a=2, b=0, h=1, d=1, i=1, f=0, c=0, e=0}
|   |   recencia ≤ 2.899
|   |   |   recencia > 2.752: a {g=1, a=7, b=0, h=1, d=3, i=0, f=0, c=1, e=1}
|   |   |   recencia ≤ 2.752
|   |   |   |   recencia > 2.689: g {g=2, a=0, b=1, h=2, d=0, i=1, f=2, c=0, e=0}
|   |   |   |   recencia ≤ 2.689
|   |   |   |   |   recencia > 2.522
|   |   |   |   |   |   gasto > -0.908: a {g=1, a=2, b=1, h=2, d=1, i=0, f=0, c=0, e=1}
|   |   |   |   |   |   gasto ≤ -0.908: g {g=5, a=1, b=0, h=0, d=3, i=0, f=4, c=0, e=2}
|   |   |   |   |   |   recencia ≤ 2.522: a {g=735, a=867, b=381, h=678, d=669, i=280, f=423, c=187, e=531}
|   |   |   |   |   |   gasto ≤ -1.572: g {g=1, a=0, b=0, h=0, d=0, i=0, f=0, c=0, e=1}
|   |   |   |   |   |   gasto ≤ -1.626: g {g=2, a=0, b=0, h=0, d=0, i=0, f=0, c=0, e=0}

```

A partir del operador “Select attributes” y de los datos originales, añadiremos el “Aggregate” donde añadiremos la media de gasto agrupándolo por el atributo producto:

aggregation attribute	aggregation functions
gasto ▼	average ▼



Esto lo conectaremos al operador “Clustering” que nos permitirá hacer una agrupación aglomerativa. De este operador le conectaremos el “Flatten Clustering” obteniendo la pertenencia a los clusters:

Row No.	id	cluster	product	average(gas...
1	1	cluster_2	a	0.548
2	2	cluster_2	b	0.567
3	3	cluster_0	c	0.472
4	4	cluster_0	d	0.407
5	5	cluster_1	e	0.661
6	6	cluster_2	f	0.534
7	7	cluster_0	g	0.414
8	8	cluster_2	h	0.560
9	9	cluster_2	i	0.536

Finalmente, añadiremos el operador “Transition Graph” que nos crea una representación gráfica de todas las transiciones desde el inicio hasta el final:

## Transition Graph

Transition Graph

Source Attribute: cluster

Target Attribute: product

Strength Attribute:

Type Attribute:

