

**Autor:** José Romualdo Villalobos Pérez

**ID:** 000294087

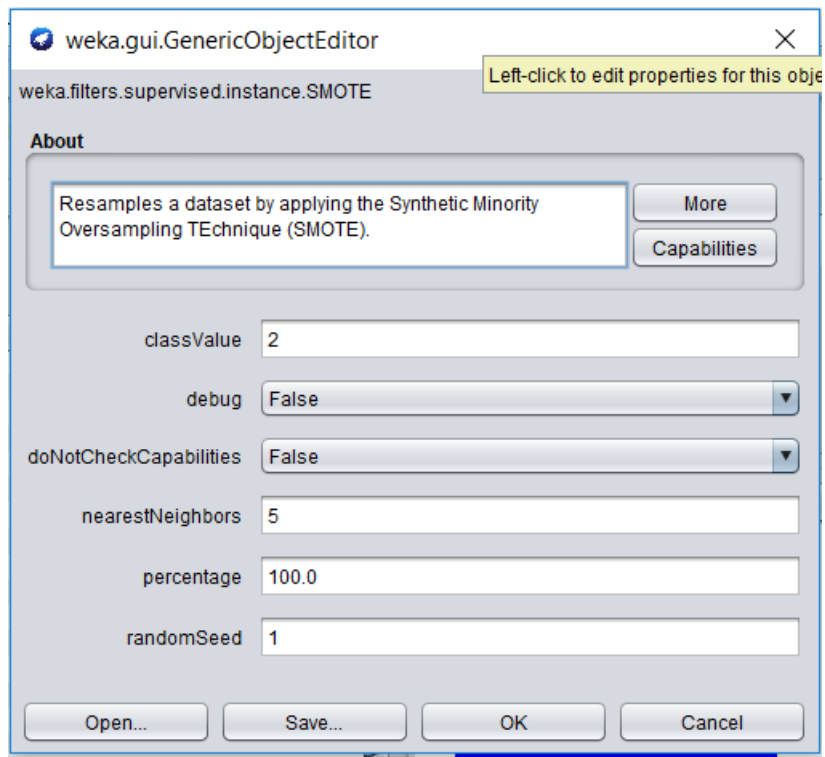
### El problema:

Un banco alemán desea establecer políticas de mercadeo según el tipo de cliente. Para esto se desean identificar tipos de clientes de una base de datos de 1000 registros con 20 atributos cada uno.

## Preparando los datos



Al observar los datos, se consideró que estos se encuentran desbalanceados, por lo tanto se procedió a balancearlos aplicando un SMOTE.



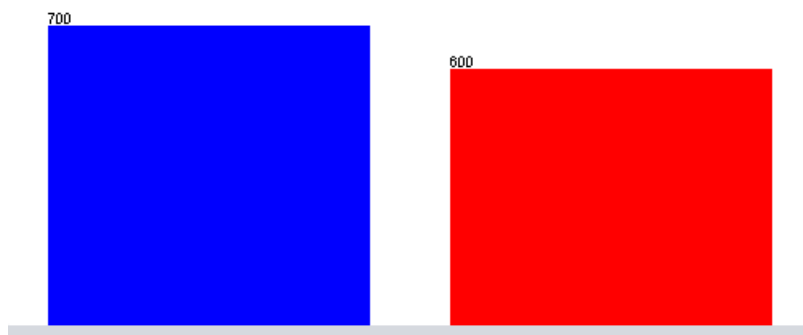
El SMOTE agregó 300 nuevas instancias al dataset dándonos un total de 1300 instancias en total.

**Selected attribute**

Name: class	Distinct: 2	Type: Nominal
Missing: 0 (0%)		Unique: 0 (0%)

No.	Label	Count	Weight
1	good	700	700.0
2	bad	600	600.0

Class: class (Nom) Visualize All



Como se puede evidenciar el dataset quedo significativamente más balanceado.

Luego, se reviso el dataset en busca de valores atípicos, pero no se identificó ninguno.

## Definiendo el número de clusters a detectar

Determinar el parámetro K es uno de los problemas de las técnicas no supervisadas, existen diversos métodos para hallar un valor optimo, como por ejemplo maximizar el Bayesian Information Criterion, sin embargo para nuestro caso se utilizó una técnica de prueba – error para determinar dicho parámetro, teniendo como criterio de selección **el primer valor de K para el cual la cantidad de centroides que pertenezcan al grupo de buenos clientes sea mayor al numero de centroides que pertenezcan al grupo de malos clientes** al aplicar el algoritmo de clustering k-means.

Cluster#	0	1	2	3	4
	(106.0)	(291.0)	(256.0)	(358.0)	(289.0)
=====					
<0	no checking	<0	<0	no checking	
20.6829	22.3849	19.8834	26.4828	17.0726	
existing paid	existing paid	existing paid	existing paid	existing paid	
used car	radio/tv	new car	new car	radio/tv	
3691.6142	3685.5808	2382.773	4797.7338	2282.6939	
<100	no known savings	<100	<100	<100	
1<=X<4	>=7	<1	1<=X<4	1<=X<4	
2.4327	3.2371	3.2691	2.8778	2.834	
female div/dep/mar	male single	female div/dep/mar	male single	female div/dep/mar	
none	none	none	none	none	
3.6315	3.1856	2.8296	2.83	2.2905	
car	car	real estate	car	real estate	
31.8896	39.4227	32.0713	35.3599	34.1934	
none	none	none	none	none	
rent	own	own	own	own	
1.2642	1.5258	1.3132	1.4131	1.3807	
skilled	skilled	skilled	skilled	skilled	
1.0283	1.1924	1.1109	1.2312	1.0923	
none	yes	none	none	none	
yes	yes	yes	yes	yes	
good	good	bad	bad	good	

Se usó este criterio porque se consideró más importante segmentar a los buenos clientes que segmentar a los malos, esto con el objetivo de intentar brindarles una experiencia más personalizada.

Luego de varias pruebas el primer valor de K que satisfizo este criterio fue **K=5**

A continuación, vamos a probar varios algoritmos no supervisados de clustering.

## K-Means

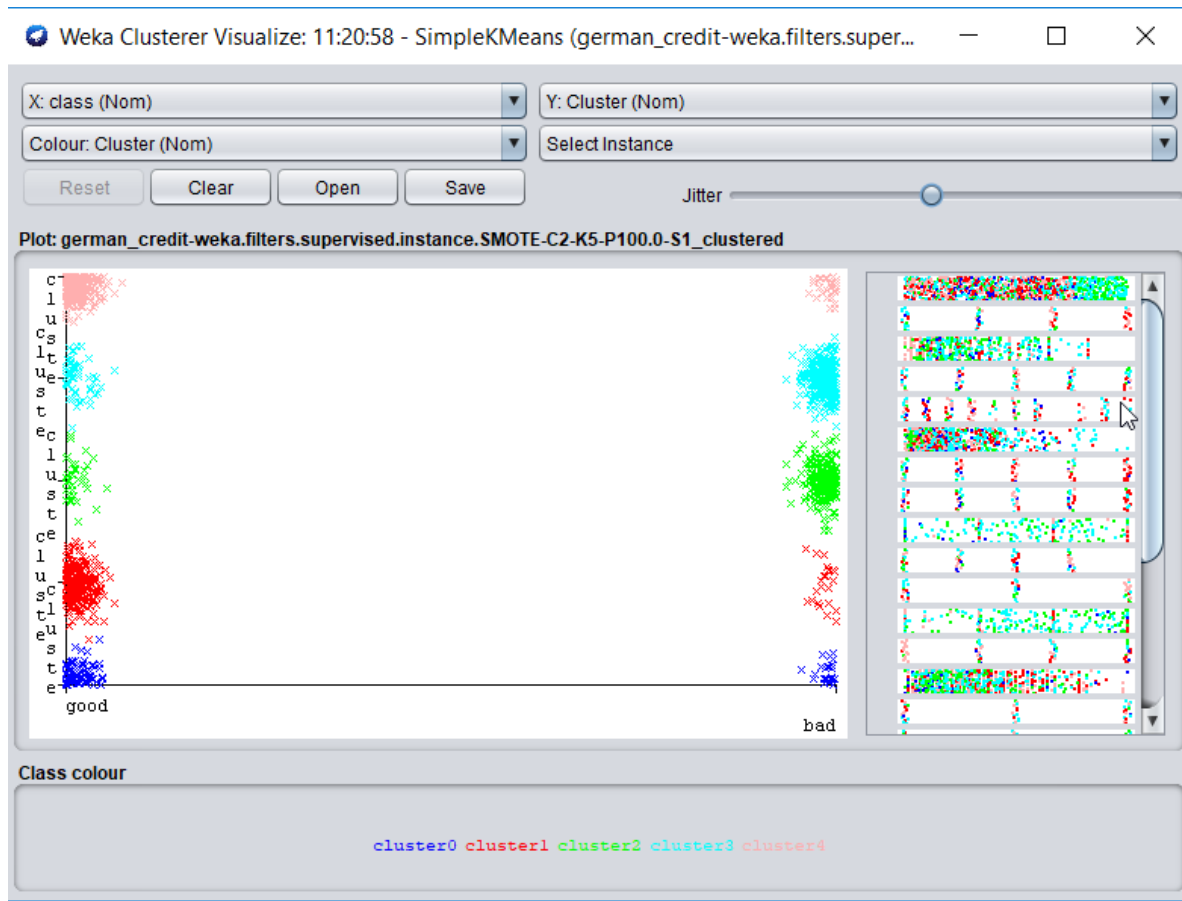
The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.clusterers.SimpleKMeans' class. The 'About' section contains a text box with 'Cluster data using the k means algorithm.' and buttons for 'More' and 'Capabilities'. Below this, various parameters are listed with their current values:

Parameter	Value
canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	5
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

At the bottom, there are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'.

Se obtuvieron los siguientes centroides para los 5 clusters:

Cluster#	0	1	2	3	4
	(106.0)	(291.0)	(256.0)	(358.0)	(289.0)
=====					
	<0	no checking	<0	<0	no checking
	20.6829	22.3849	19.8834	26.4828	17.0726
	existing paid	existing paid	existing paid	existing paid	existing paid
	used car	radio/tv	new car	new car	radio/tv
	3691.6142	3685.5808	2382.773	4797.7338	2282.6939
	<100	no known savings	<100	<100	<100
	1<=X<4	>=7	<1	1<=X<4	1<=X<4
	2.4327	3.2371	3.2691	2.8778	2.834
female div/dep/mar		male single	female div/dep/mar	male single	female div/dep/mar
	none	none	none	none	none
	3.6315	3.1856	2.8296	2.83	2.2905
	car	car	real estate	car	real estate
	31.8896	39.4227	32.0713	35.3599	34.1934
	none	none	none	none	none
	rent	own	own	own	own
	1.2642	1.5258	1.3132	1.4131	1.3807
	skilled	skilled	skilled	skilled	skilled
	1.0283	1.1924	1.1109	1.2312	1.0923
	none	yes	none	none	none
	yes	yes	yes	yes	yes
	good	good	bad	bad	good



Lamentablemente este método de clustering no separó los clientes buenos de los malos.

Ahora, vamos a nombrar los clusters:

**Cluster 0:**

**Nombre:** Independiente

**Descripción Persona:** Mujer responsable y dedicada a su carrera, no le da miedo ir a otros países a trabajar, estas mujeres le huyen a los gastos grandes, prefiere tener un auto usado que uno nuevo, prefiere rentar su vivienda que comprar una no le preocupa lo que pueda pasar en el futuro.

**Cluster 1:**

**Nombre:** Libertad

**Descripción Persona:** Dedicado a su carrera, valora mucho los espacios de ocio, por eso tiene un TV gigante, vive en su propia casa y no tiene pareja.

**Cluster 2:**

**Nombre:** Imperial

**Descripción Persona:** Tiene su propia casa, no le gusta endeudarse a largo plazo, pero a veces no calcula bien sus verdaderas capacidades, le gusta movilizarse en automóvil y prefiere sentir que es la dueña.

**Cluster 3:**

**Nombre:** Orgullo personal

**Descripción Persona:** No tiene pareja, le gusta lucir su automóvil, dedicado a su carrera, se siente cómodo en proyectos de mediano plazo, a veces no calcula bien sus verdaderas capacidades.

**Cluster 4:**

**Nombre:** Vive la vida

**Descripción Persona:** Dedicada a su carrera, valora su comodidad y los momentos de ocio, tiene su propia casa, despreocupada de lo que pueda pasar en el futuro.

**Probando el modelo con 5 usuarios nuevos:**

Para predecir a que cluster pertenecen 5 usuarios nuevos, vamos a aplicar el clasificador Support Vector Machine donde la clase de cada instancia es el cluster al que pertenece, para este caso entrenamos una SVM con un kernel lineal.

=== Summary ===

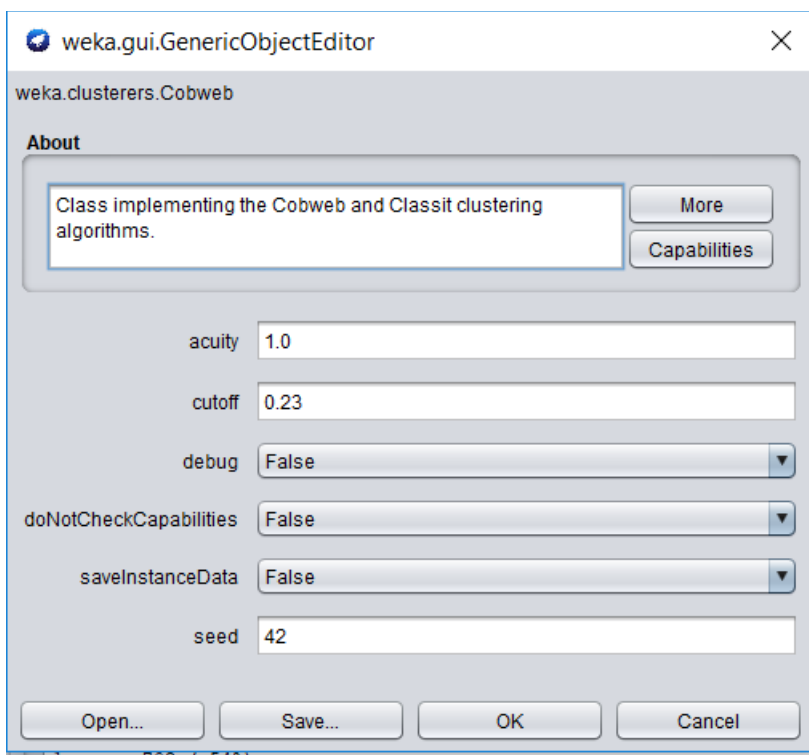
Correctly Classified Instances	355
Incorrectly Classified Instances	87
Kappa statistic	0.7459
Mean absolute error	0.0787
Root mean squared error	0.2806
Relative absolute error	25.2636 %
Root relative squared error	71.1517 %
Total Number of Instances	442

80.3167 %
19.6833 %

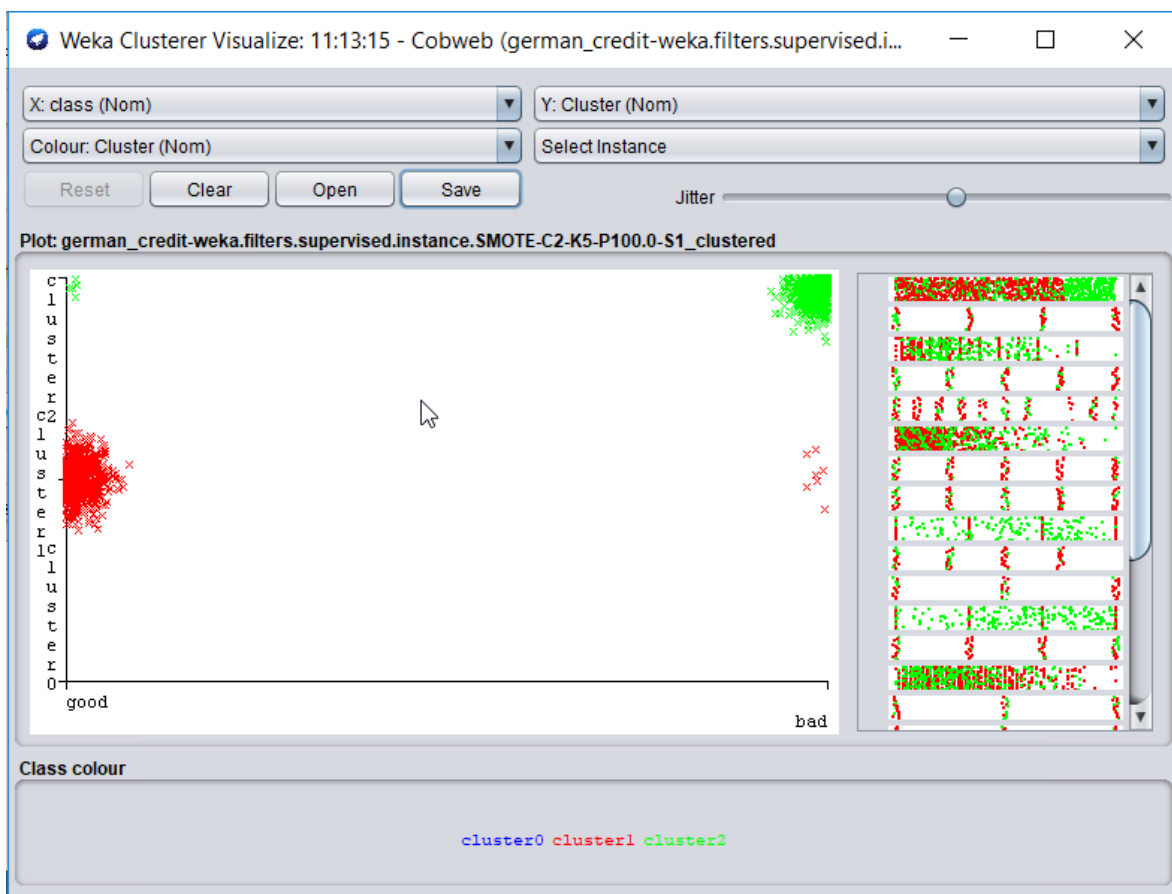
=== Evaluation on test set ===  
Clustered Instances

1	3 ( 60%)
2	1 ( 20%)
4	1 ( 20%)

## Cobweb



El valor de cutoff es 0.23 para evitar que el árbol crezca mucho, el resultado obtenido se muestra a continuación:



Afortunadamente para el banco, este método de clustering agrupo a los clientes buenos y a los clientes malos en clusters diferentes.



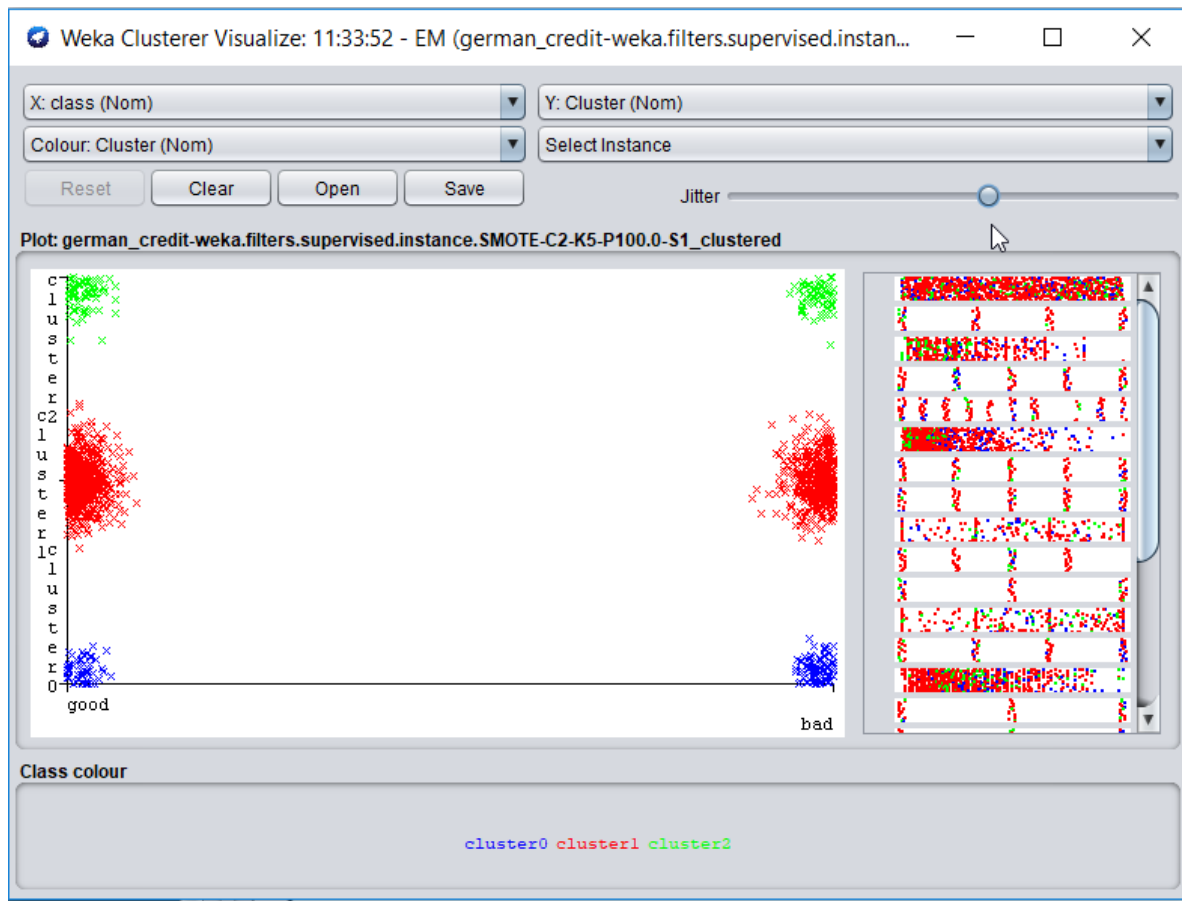
## Expectation Maximization

The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.clusterers.EM' class. The 'About' tab is active, displaying the description 'Simple EM (expectation maximisation) class.' and buttons for 'More' and 'Capabilities'. Below this, various configuration options are listed:

- debug: False
- displayModelInOldFormat: False
- doNotCheckCapabilities: False
- maxIterations: 100
- maximumNumberOfClusters: -1
- minLogLikelihoodImprovementCV: 1.0E-6
- minLogLikelihoodImprovementIterating: 1.0E-6
- minStdDev: 1.0E-6
- numClusters: -1** (highlighted with a red box)
- numExecutionSlots: 1
- numFolds: 10
- numKMeansRuns: 10
- seed: 100

At the bottom, there are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'.

Cuando numClusters = -1 el algoritmo selecciona un valor automáticamente para K basado en la cross validation.



No separó a los clientes buenos de los clientes malos.

## Self-Organizing Maps

Debido a que Weka no tiene una clase para aplicar este algoritmo, se implementará en Python y en R únicamente.

*"Self-organizing map is not in weka catalogue the only neural networks do not compress features to viewable 2d map only regular nn's named multilayerperceptron and rbfnetwork which have different (opposite) activation functions"*

- Harry M.T. Saarikoski, Weka mail list

## Association, A priori

weka.gui.GenericObjectEditor

weka.associations.Apriori

**About**

Class implementing an Apriori-type algorithm. [More](#) [Capabilities](#)

car True

classIndex -1

delta 0.05

doNotCheckCapabilities False

lowerBoundMinSupport 0.1

metricType Confidence

minMetric 0.9

numRules 10

outputItemSets False

removeAllMissingCols False

Remove columns with all missing values

treatZeroAsMissing False

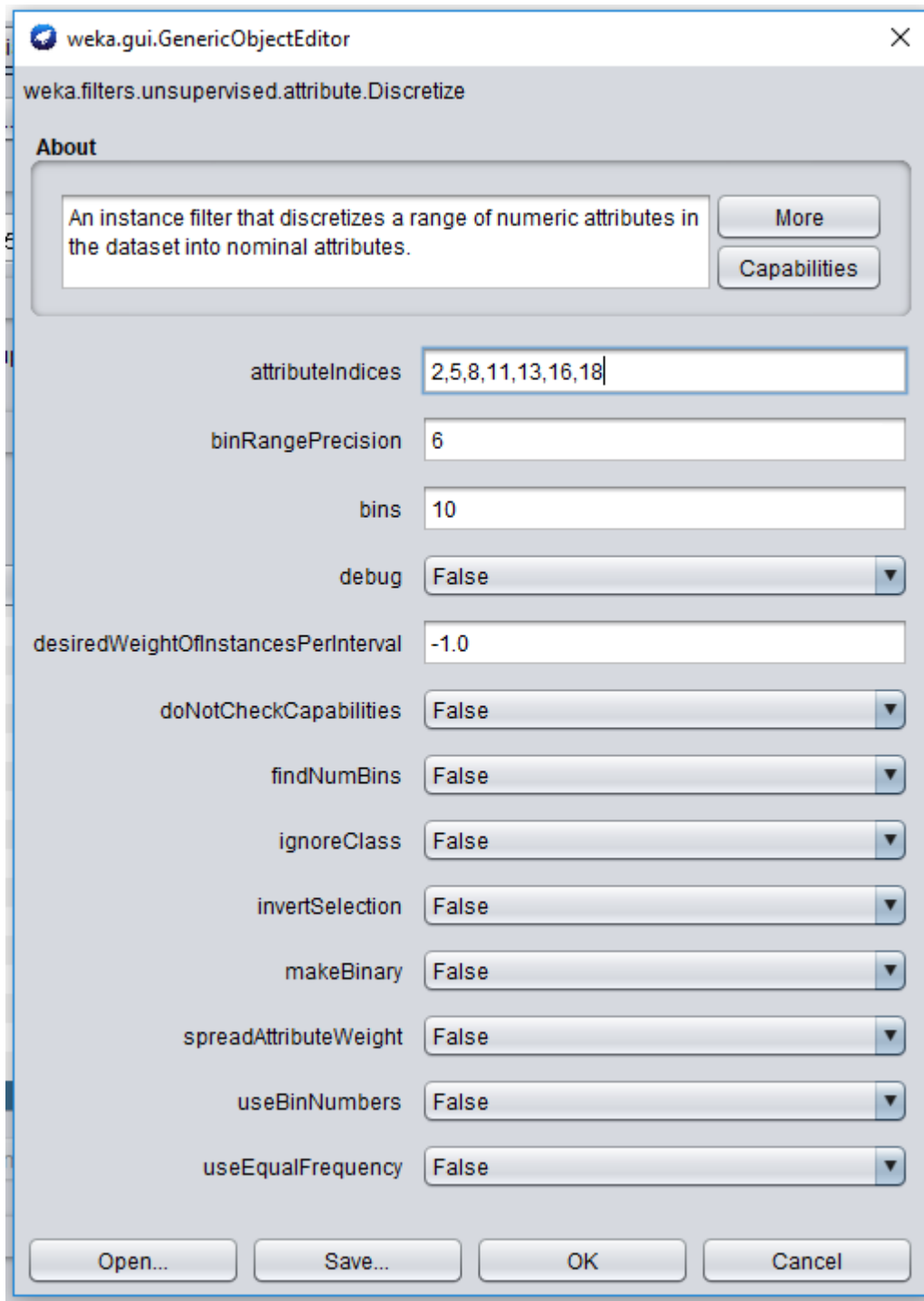
upperBoundMinSupport 1.0

verbose False

Open... Save... OK Cancel

**Car = true** porque las instancias tienen un atributo clase, y **classIndex=-1** porque la clase se encuentra en la última columna.

Debido a que el método apriori solo funciona con variables categóricas, vamos a proceder discretizando los atributos numéricos.



The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.filters.unsupervised.attribute.Discretize' filter. The 'About' section describes it as an instance filter that discretizes a range of numeric attributes into nominal attributes. The configuration options are as follows:

Property	Value
attributeIndices	2,5,8,11,13,16,18
binRangePrecision	6
bins	10
debug	False
desiredWeightOfInstancesPerInterval	-1.0
doNotCheckCapabilities	False
findNumBins	False
ignoreClass	False
invertSelection	False
makeBinary	False
spreadAttributeWeight	False
useBinNumbers	False
useEqualFrequency	False

At the bottom, there are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'.

Como resultado, ahora todos los atributos son de tipo nominal, ahora si podemos aplicar las reglas de asociación.

Best rules found:

1. checking\_status=no checking other\_parties=none other\_payment\_plans=none job=skilled 218 ==> clas
2. checking\_status=no checking other\_parties=none other\_payment\_plans=none job=skilled foreign\_work
3. checking\_status=no checking other\_payment\_plans=none job=skilled 231 ==> class=good 213 conf:
4. checking\_status=no checking other\_payment\_plans=none job=skilled foreign\_worker=yes 224 ==> clas
5. checking\_status=no checking other\_payment\_plans=none housing=own num\_dependents='(-inf-1.1]' 223
6. checking\_status=no checking other\_parties=none other\_payment\_plans=none num\_dependents='(-inf-1.
7. checking\_status=no checking other\_parties=none other\_payment\_plans=none housing=own 251 ==> clas
8. checking\_status=no checking other\_payment\_plans=none housing=own num\_dependents='(-inf-1.1]' for
9. checking\_status=no checking other\_parties=none other\_payment\_plans=none num\_dependents='(-inf-1.
10. checking\_status=no checking other\_parties=none other\_payment\_plans=none 320 ==> class=good 290

Podemos concluir que este algoritmo brinda al banco información muy valiosa, que podrá permitir sacar mas valor de sus clientes.