Contenido

[Analizando las variables 2](#_Toc510469037)

[Pre-procesando los datos 3](#_Toc510469038)

[Limpieza de los datos 3](#_Toc510469039)

[Aplicando transformaciones 6](#_Toc510469040)

[Selección de variables 7](#_Toc510469041)

[Balanceo de datos 8](#_Toc510469042)

[Aplicando métodos supervisados 9](#_Toc510469043)

[Métodos de regresión 9](#_Toc510469044)

[Probando el modelo 11](#_Toc510469045)

[Métodos basados en ejemplos 13](#_Toc510469046)

[Optimizando los parámetros 13](#_Toc510469047)

[Probando el método con nuevos datos 16](#_Toc510469048)

[Máquina de soporte vectorial 17](#_Toc510469049)

[Optimizando los parámetros 17](#_Toc510469050)

[Probando el método con nuevos datos 19](#_Toc510469051)

[Arboles de decisión 19](#_Toc510469052)

[Optimizando los parámetros 20](#_Toc510469053)

[Probando el método con nuevos datos 25](#_Toc510469054)

[Bonus 26](#_Toc510469055)

[Método Bayesiano 26](#_Toc510469056)

[Probando el método con datos nuevos 29](#_Toc510469057)

[Redes neuronales 29](#_Toc510469058)

[Observaciones 35](#_Toc510469059)

[Probando el método con datos nuevos 39](#_Toc510469060)

[Conclusiones, Eligiendo a los ganadores 39](#_Toc510469061)

Autor: José Romualdo Villalobos Pérez

ID: 000294087

Lo primero que se hizo fue agregar el encabezado de WEKA al archivo drug.csv y se guardó como drug.arff

# Analizando las variables

La variable edad tiene una distribución normal, se observó que hay un individuo con una edad de 145 lo cual se puede considerar bastante improbable ya que es un dato atípico, por lo tanto, se procederá a eliminar este dato.

La variable sexo esta balanceada ya que 52% son hombres y el otro 48% son mujeres.

|  |  |
| --- | --- |
| Valor variable Sex | Porcentaje personas con el mismo valor |
| M | 52% |
| F | 48% |

La variable BP (Blood Pressure) También se encuentra balanceada:

|  |  |
| --- | --- |
| Valor variable BP | Porcentaje personas con el mismo valor |
| Low | 32% |
| Normal | 29.5% |
| high | 38.5% |

Respecto a la variable **cholesterol** se encontró un detalle, nadie tenía el colesterol en LOW por lo tanto se pensó una posible variable desbalanceada, así que se decidió consultar en portales médicos si es posible tener el colesterol bajo, se encontró lo siguiente:

*“Los médicos siguen tratando de saber más sobre la conexión entre el colesterol bajo y los riesgos para la salud. No hay consenso respecto de cómo definir un nivel muy bajo de colesterol LDL, pero el nivel de LDL se consideraría muy bajo si está por debajo de 40 miligramos por decilitro de sangre.”*

Debido a que es un tema que aun se encuentra en discusión se considera para este ejercicio que esta variable categoría se encuentra balanceada ya que solo puede tener dos valores NORMAL o HIGH.

|  |  |
| --- | --- |
| Valor variable cholesterol | Porcentaje personas con el mismo valor |
| Normal | 48.5% |
| high | 51.5% |

Por lo tanto, se concluye que esta variable se encuentra balanceada.

En la variable continua **Na** se encontró que hay un dato atípico, una persona con un nivel de sodio de -0.85, se sospecha que este valor nos es humanamente posible (al menos que se este en la condición incapacitante de la muerte) aunque no se encontró una fuente médica confiable que permitiera comprobar esta afirmación, por lo tanto, se procederá a eliminar a esta persona de los registros.

Al analizar la variable continua **K** se encontró que tiene una distribución uniforme y no se vio ningún valor anormal.

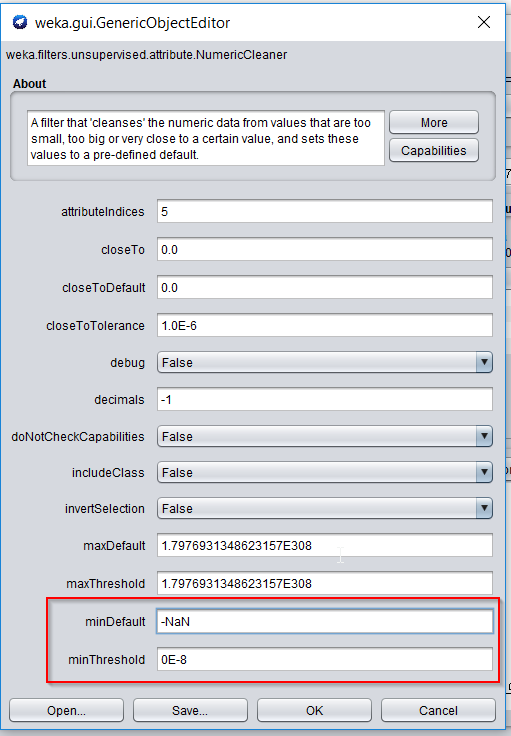
# Pre-procesando los datos

## Limpieza de los datos

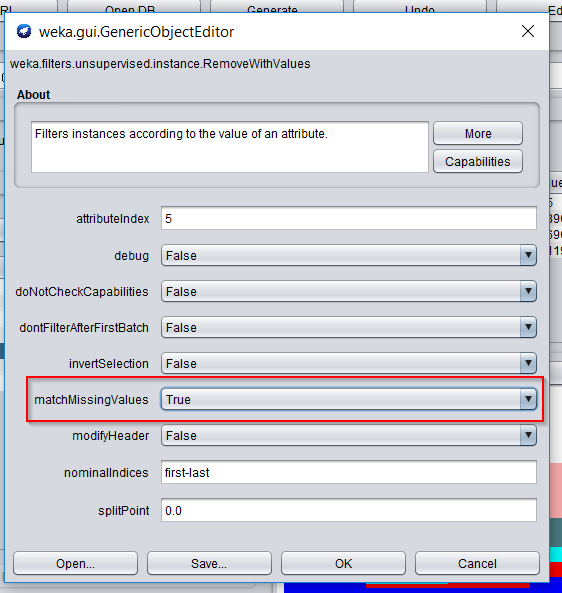
Debido a que hay poquitos pacientes (instancias) con valores atípicos se decidió eliminar dichas instancias en lugar de reemplazar el valor del atributo en cuestión.

Para lograrlo se hizo lo siguiente:

Para la variable **Na** se aplicó el filtro **NumericCleaner** y se marcaron como valores perdidos aquellos que estuvieran por debajo de 0 es decir se marcaron como NaN.



Luego se aplicó un filtro que elimina cualquier instancia que tenga un atributo con un valor perdido.



Luego de aplicar este filtro se pudo observar que el número de instancias bajo de 200 a 199, se aplicó un proceso similar para eliminar la instancia con la edad atípica, solo que en este caso se puso el maxThreshold del NumericCleaner en 110 (ya que es poco probable que algún humano pase esa edad)

Siguiendo con la limpieza de datos, no se encontró ni datos ausentes o nulos ni registros repetidos, razón por la cual se continuó al siguiente paso del preprocesamiento de los datos.

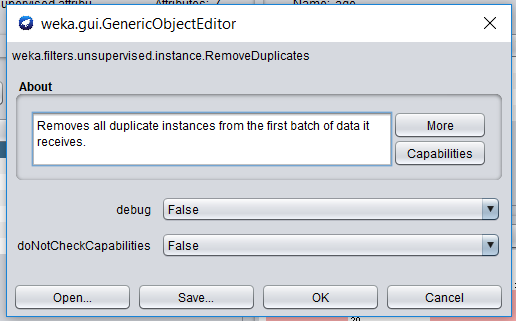
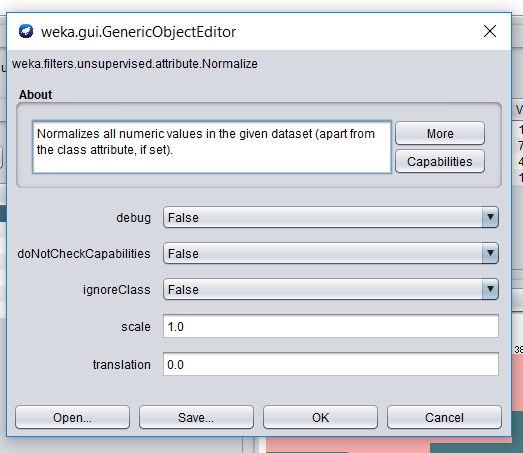


Ilustración 1 Después de aplicar un filtro que remueve los duplicados se observó que no se eliminó ninguna instancia, por tanto se concluye que no hay registros repetidos.

Volviendo a observar las variables continuas se observa todas tienen una distribución uniforme.

## Aplicando transformaciones

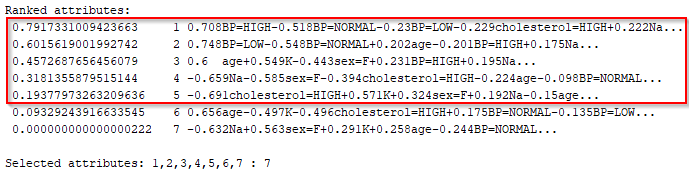
Se procederá a continuación a normalizar todas las variables continuas para asegurar que los entrenamientos no sean muy sensibles a las diferencias de escala entre los valores de los atributos. Por ejemplo: Si observamos los valores de los atributos **Age** y **Na** veremos que el primero está en el rango [15, 74] y el segundo en el rango [0.5, 0.896]



No se discretizarán los atributos con valores continuos en este punto.

## Selección de variables

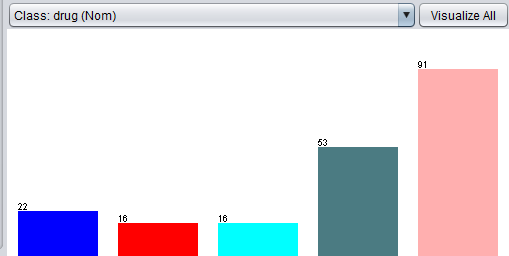
Se procederá ahora a buscar los atributos con mayor importancia usando Principal Component Analisys este algoritmo puede crear atributos nuevos combinando los atributos originales.



Se encontró un problema subjetivo al aplicar este método. Los atributos se vuelven mucho más difíciles de entender por un humano, lo cual hace difícil usar el sistema con nuevas instancias de ejemplo.

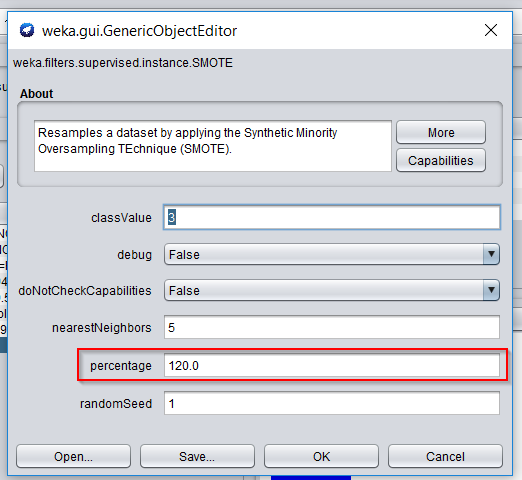
Además, debido a la relativamente baja cantidad de atributos se considero que no hace falta eliminar atributos.

## Balanceo de datos

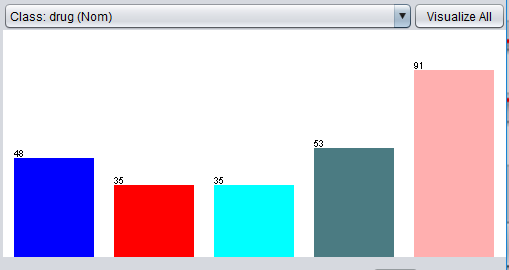


Al observar la **class** se observa que esta desbalanceada, lo cual nos expone a un alto riesgo de overfitting, razón por la cual se procedió a balancearlos “adicionando instancias” usando la técnica **SMOTE (Synthetic Minority Over-sampling Technique)**.

Se aplicó el **SMOTE** a los valores drugA, drugB y drugC de la **class**, para ello se usaron los siguientes parámetros de configuración:



Se incremento en un 120% la cantidad de instancias cuyas clases tienen valores drugA, drugB y drugC. El resultado obtenido fue el siguiente:

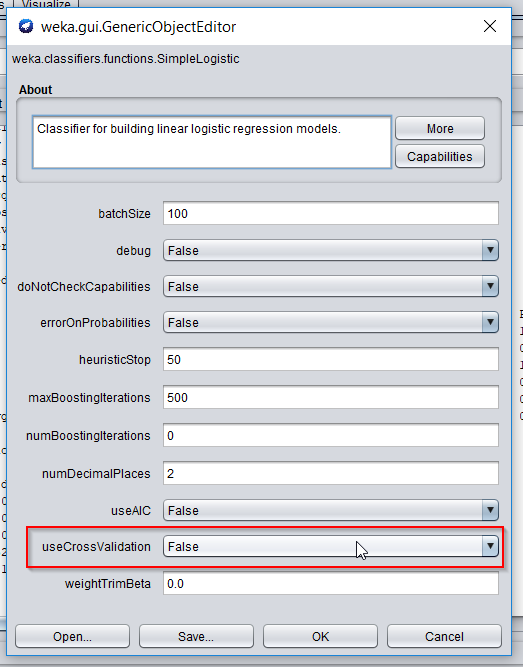


Cabe mencionar que este proceso incremento el numero de instancias de 198 a 262.

# Aplicando métodos supervisados

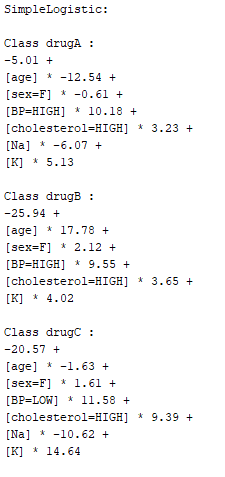
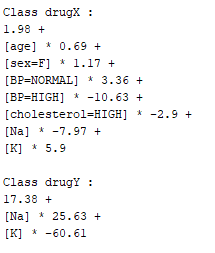
## Métodos de regresión

No se usará el método de regresión lineal porque el atributo a predecir es categórico en su lugar se usará el método de regresión logística el cual se usa cuando el atributo a predecir es categórico.



Se desactivo la opción que viene incorporada en el método SimpleLogistic de realizar validación cruzada porque para esta labor se usará la opción que viene incorporada por defecto en weka.

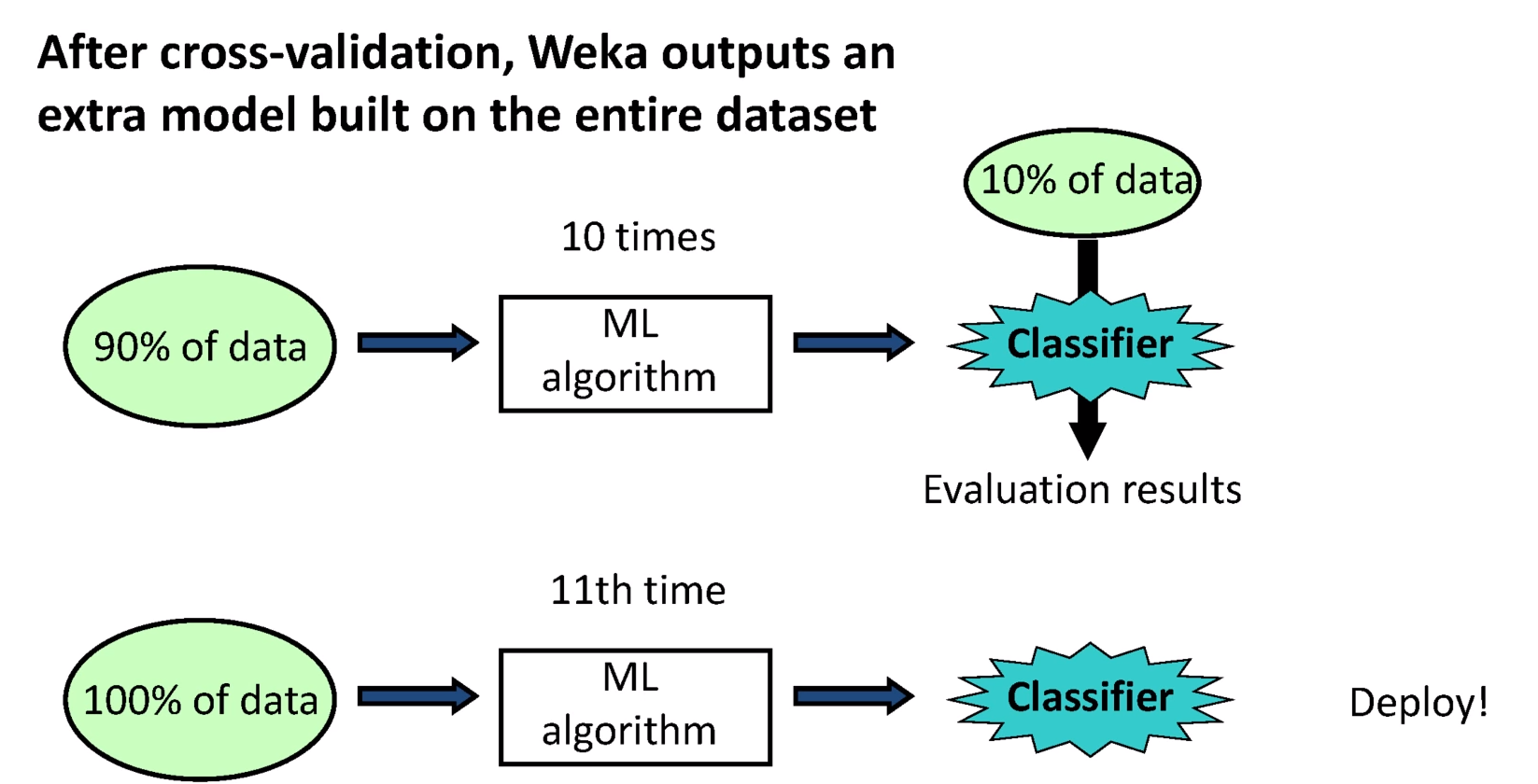
Debido a que este es un problema de regresión con múltiples clases nuestro modelo tendrá **n** regresiones donde n es el número de clases diferentes, a continuación, se muestra el modelo obtenido:

Para una instancia de ejemplo determinada nosotros escogeremos la clase cuyo output sea el más grande (la de mayor probabilidad).

### Probando el modelo

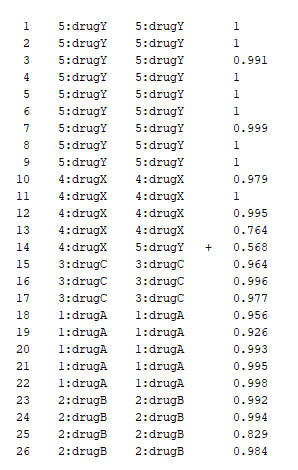
Debido a que el dataset es relativamente pequeño se decidió probar el modelo usando el método de cross validation



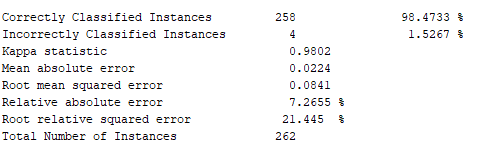
Se realizo un **stratified-cross-validation** con un **fold** de 10, nótese que por defecto Weka realiza una validación cruzada estratificada lo cual asegura que cuando se realizan los **folds** los valores de las clases tengan una proporción similar a los del **dataset** original.

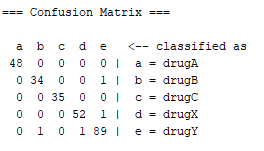
A continuación, se muestra los resultados después de probar el modelo obtenido con 26 instancias que no se usaron para entrenar el modelo.

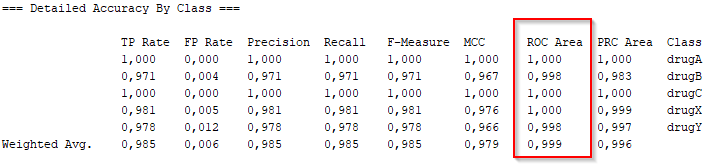
Instance | Actual | Predicted | Error | Probability



La efectividad:







Al observar el área bajo la curva ROC de las distintas clases podemos concluir que el modelo tiene una muy buena capacidad de clasificar correctamente.

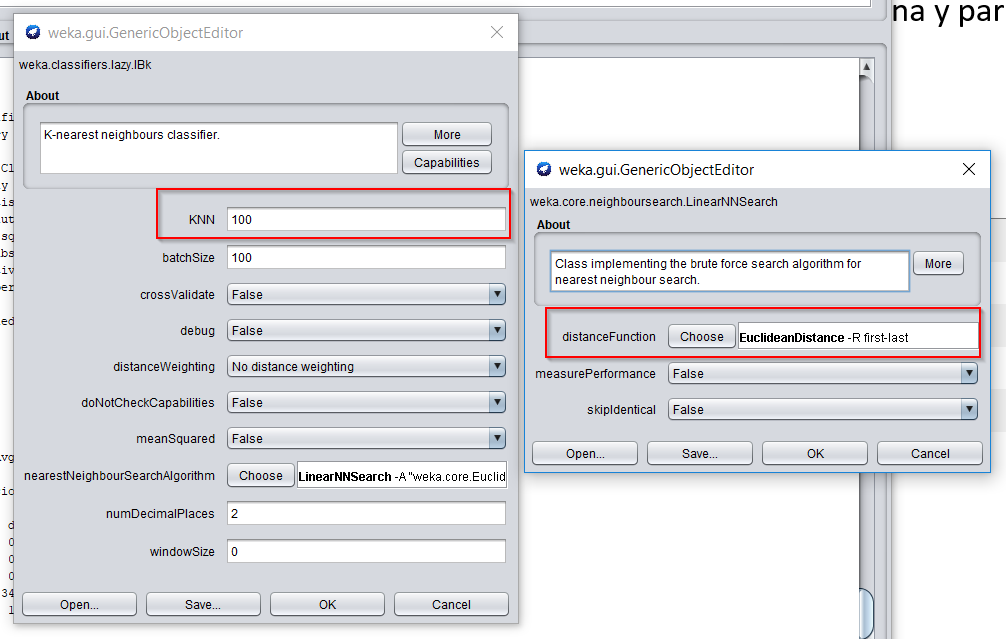
## Métodos basados en ejemplos

Se usará el método conocido como k-nearest-neighborh también conocido como **instance-based** learning, aplicaremos el método IBk (Instance Based with K argument) de weka.

### Optimizando los parámetros

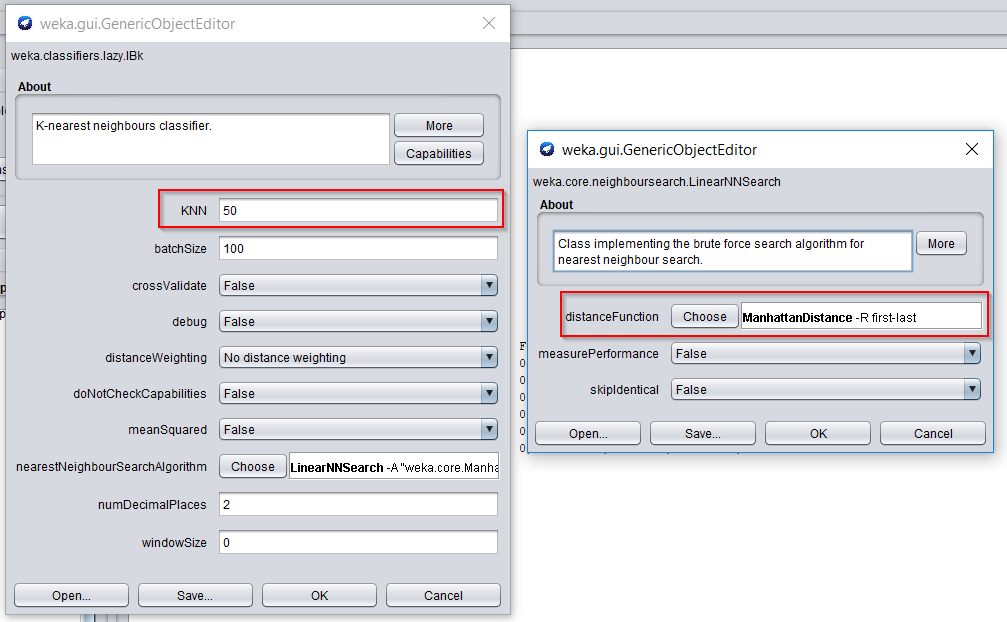
Para obtener mejores resultados vamos a probar el efecto que tiene el parámetro k en la efectividad de este método, haremos 12 pruebas con distintos valores para k y las probaremos con un 10-fold cross validation. Además, observaremos el efecto de la función de distancia en la efectividad, para las 6 primeras pruebas se usará la función de distancia euclidiana y para las restantes 6 se usará la función de distancia manhattan.

Distancia Euclidiana:



|  |  |
| --- | --- |
| Valor para K | Efectividad |
| 1 | 90.83% |
| 5 | 84.35% |
| 20 | 73.66% |
| 50 | 77.48% |
| 100 | 62.97% |
| 200 | 34.73% |

Distancia Manhatan:



|  |  |
| --- | --- |
| Valor para k | efectividad |
| 1 | 90.83% |
| 5 | 84.73% |
| 20 | 78.62% |
| 50 | 75.57% |
| 100 | 63.03% |
| 200 | 34.73% |

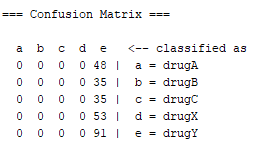
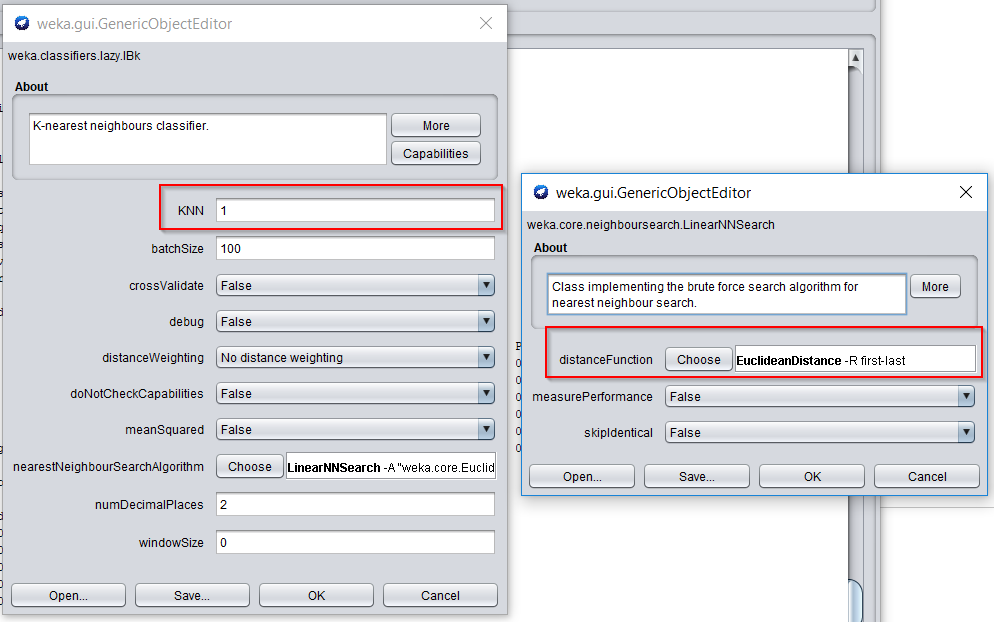


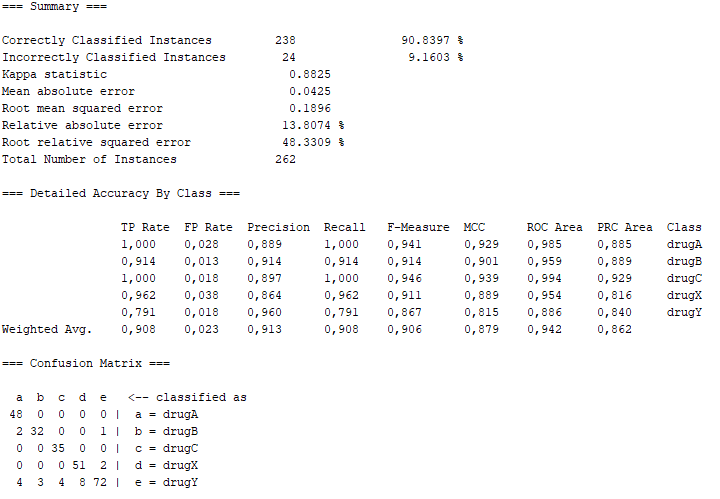
Ilustración 2 Matriz de confusión con k=200 y manhattan distance

Después de realizar las pruebas anteriores es posible sacar las siguientes conclusiones:

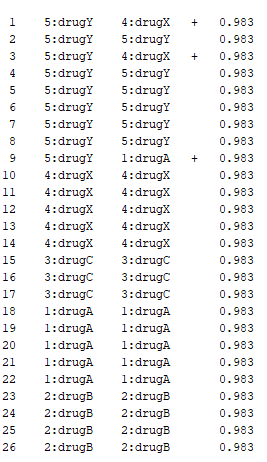
1. El método IBk funciona mejor cuando la K=1 es decir, cuando la clase de la instancia de ejemplo es la misma que la clase del vecino más cercano.
2. La función de distancia que se utilice implicará cambios menores en la efectividad, que podrían ser despreciables.
3. Cuando k se hace grande respecto al número total de instancias el modelo tiende a predecir que todas las instancias pertenecen a la clase con más instancias en el dataset, esto tiene sentido si estudia la implementación de este método.

Ahora se puede decir con mayor seguridad que los siguientes serian parámetros óptimos para el IBk:





### Probando el método con nuevos datos

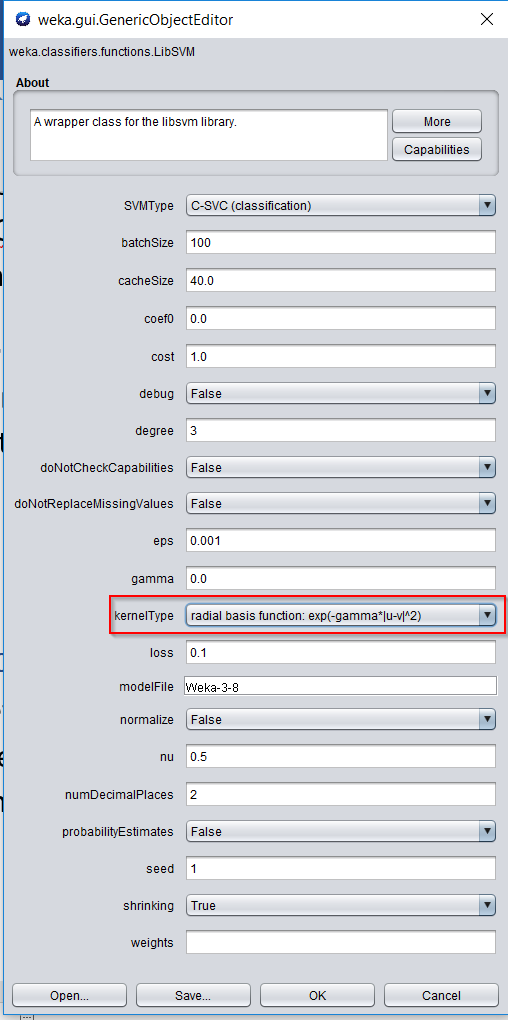


## Máquina de soporte vectorial

Este método es bastante resistente al sobre ajuste, lamentablemente en **Weka** el método functions.SMO está restringido a datasets de 2 clases y debido a que nuestro dataset tiene 5 clases se decidió usar el libSVM porque este último si soporta datasets con más de 2 clases.

### Optimizando los parámetros

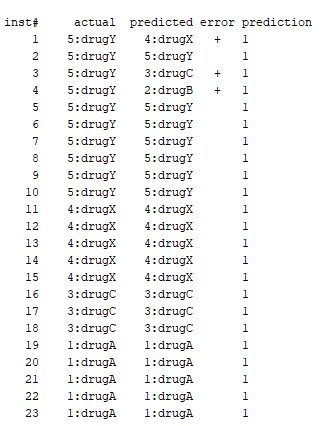
Para optimizar los parámetros de esto método se va a cambiar el tipo del kernel y se relacionará con el porcentaje de instancias correctamente clasificadas.



|  |  |
| --- | --- |
| Tipo del Kernel | % Instancias correctamente clasificadas |
| Lineal | 95.03% |
| Polinomial | 41.98% |
| Radial | 85.49% |
| Sigmoid | 71.37% |

Después de las pruebas anteriores se puede considerar que los datos de nuestro dataset son linealmente separables debido a que fue este tipo de kernel el que mejor logro clasificar las instancias de ejemplo.

### Probando el método con nuevos datos



## Arboles de decisión

Este es un método bueno para problemas donde la clase es nominal, razón por la cual se usará, cabe decir que el output de este algoritmo tiene la gran ventaja de que es muy fácil de entender por las personas lo que facilita su aplicación para futuras instancias de ejemplo.

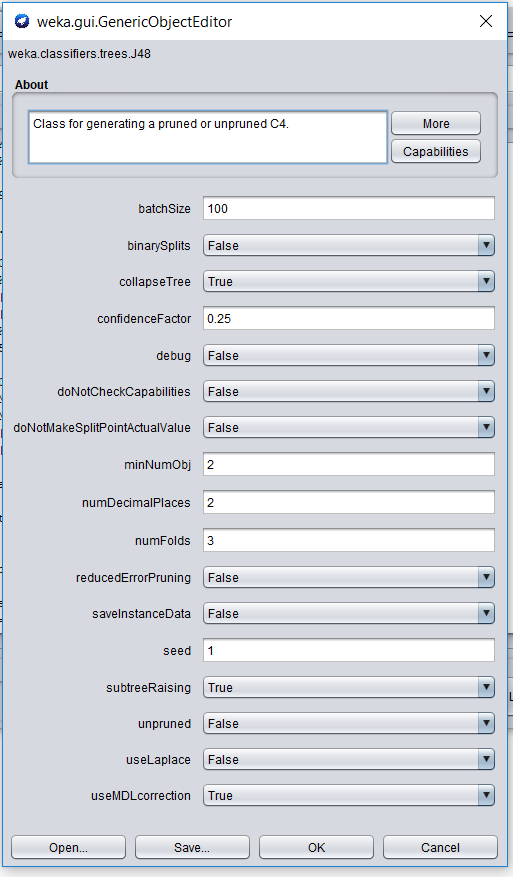
Este modelo tiene el riesgo de quedar sobreajustado si las ramas son muy profundas.

Vamos a podar el árbol, cabe decir que esto va a significar una menor efectividad con los datos de entrenamiento y prueba, pero nos permitirá obtener mejores resultados para casos desconocidos. Es decir, obtendremos un modelo más general que nos permitirá predecir casos futuros con más precisión.

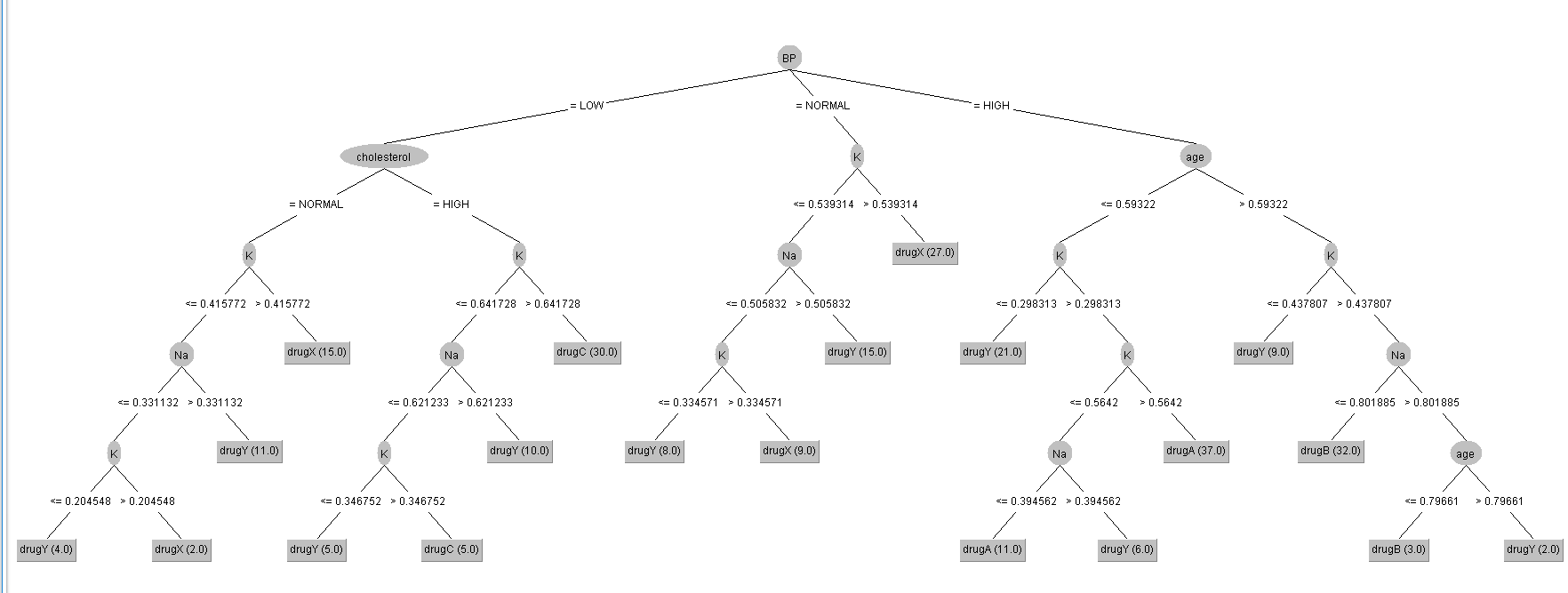
### Optimizando los parámetros

En este caso se observará el árbol y la efectividad del árbol si lo podamos o no lo podamos además se realizará una validación cruzada con un fold de 10.

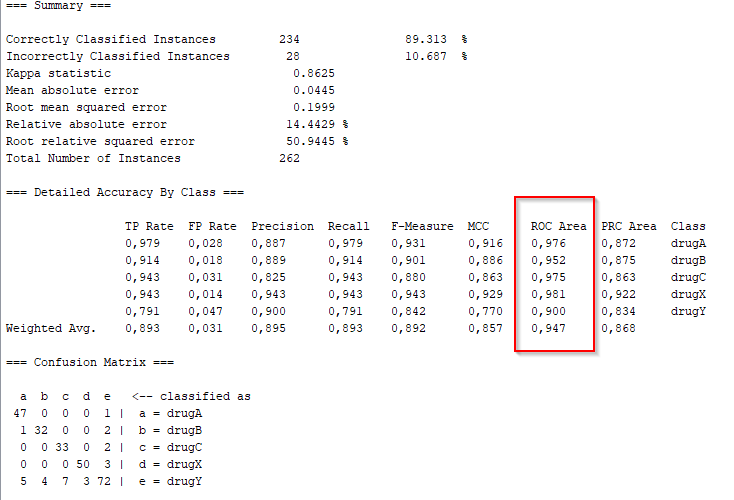
Árbol Podado:



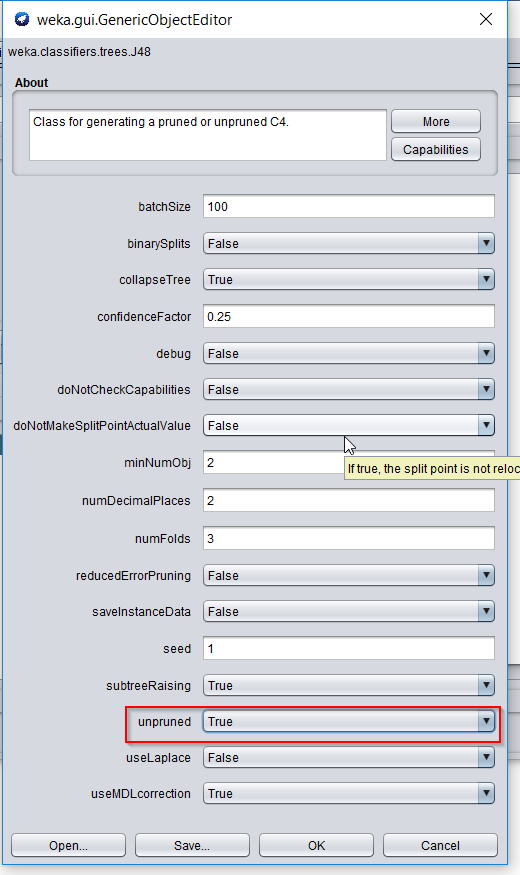
Obtenemos un árbol con 20 nodos hoja y 38 nodos en total:



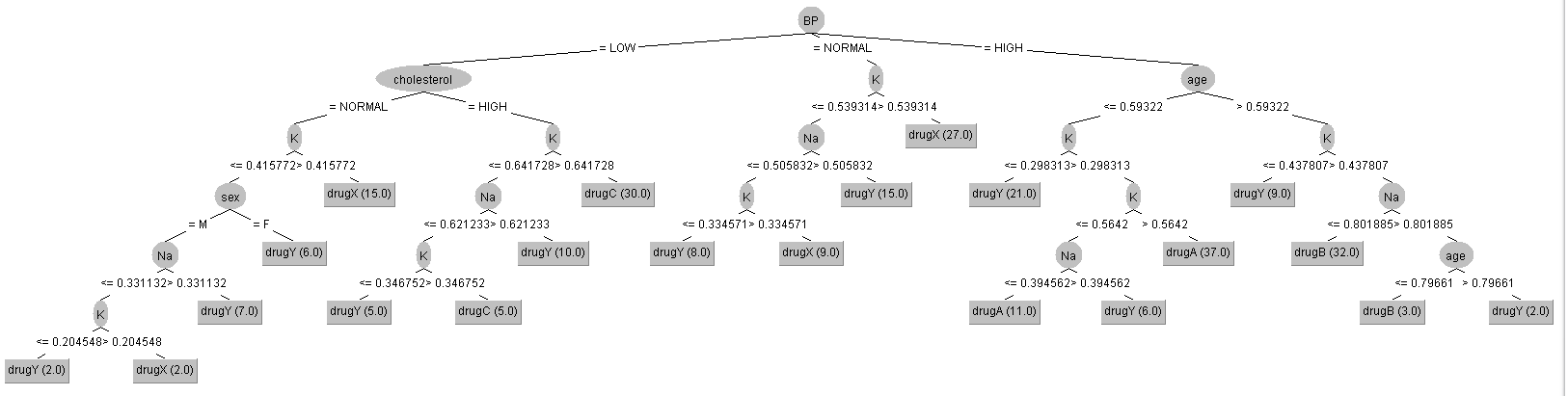
Obtenemos la siguiente efectividad:

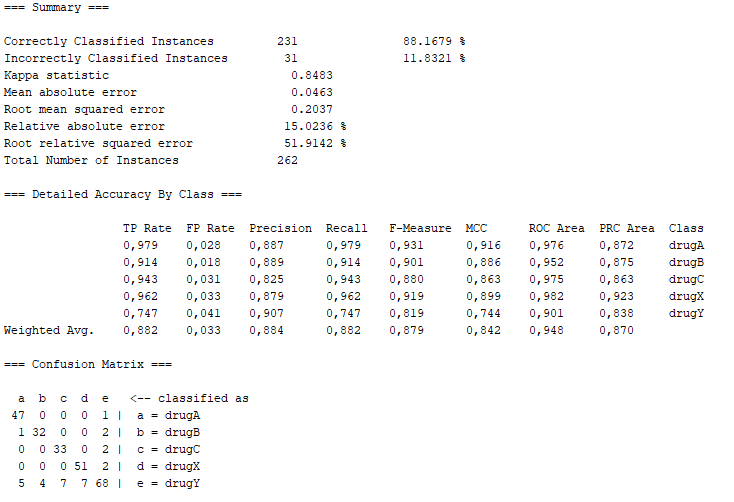


Árbol no podado



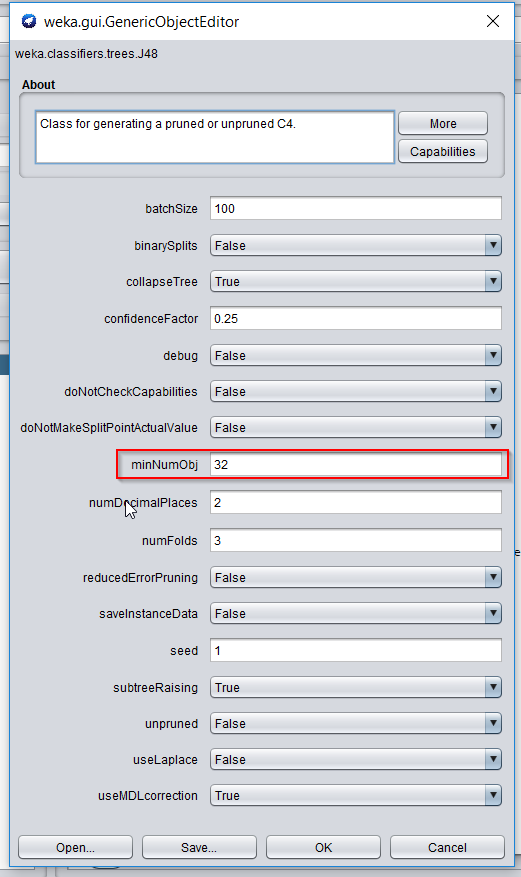
Se obtiene un árbol con 21 nodos hojas y 40 nodos en total (2 más que el árbol podado)





Con los cambios realizados anteriormente no se encontró una mejoría significativa en la efectividad del modelo, sin embargo, se llegó a un árbol 2 nodos más simples.

Debido a este poco cambio se decidió probar variaciones en el parámetro minNumObj el cual determina el número mínimo de instancias que puede haber en un nodo hoja, aumentar este valor disminuirá el número de nodos hojas en el árbol y por lo tanto ayuda a simplificarlo más.



|  |  |  |  |
| --- | --- | --- | --- |
| minNumObj | Total nodes | Leaf nodes | Correctly Clasified Instances |
| 1 | 38 | 20 | 90.83% |
| 2 | 38 | 20 | 89.31% |
| 4 | 30 | 16 | 91.22% |
| 8 | 26 | 14 | 85.87% |
| 16 | 18 | 10 | 78.24% |
| 24 | 14 | 8 | 71.75% |
| 32 | 10 | 6 | 72.90% |

Se puede observar como esta variable obliga al árbol a tener menos nodos, incluso se logró mejorar el porcentaje de instancias clasificadas correctamente poniendo el valor en 4 lo cual resulto en un árbol 1. Mas simple y 2. Con más porcentaje de predicciones exitosas.

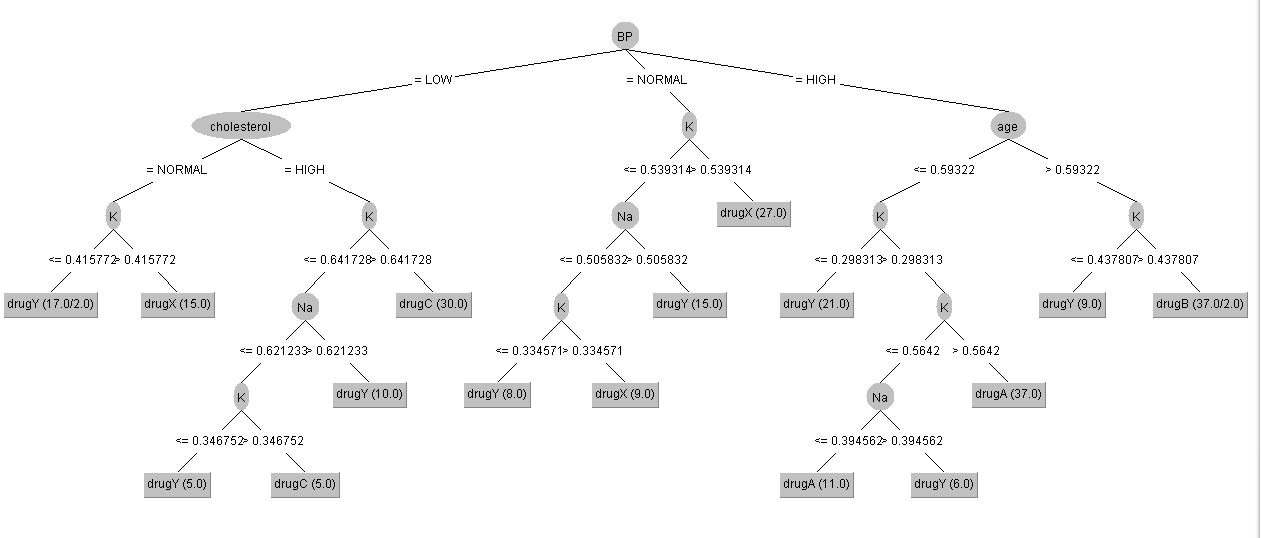
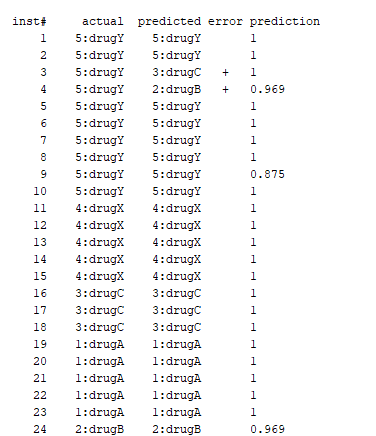


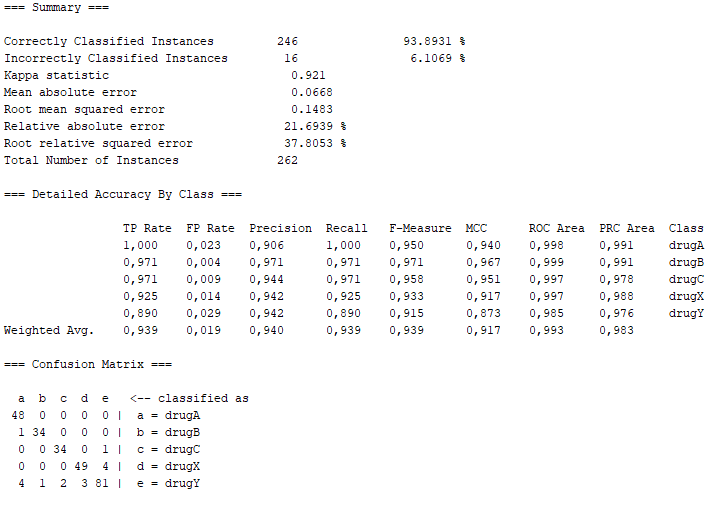
Ilustración 3 Árbol de decisión calculado con minNumObj = 4, se obtuvo 16 hojas

### Probando el método con nuevos datos



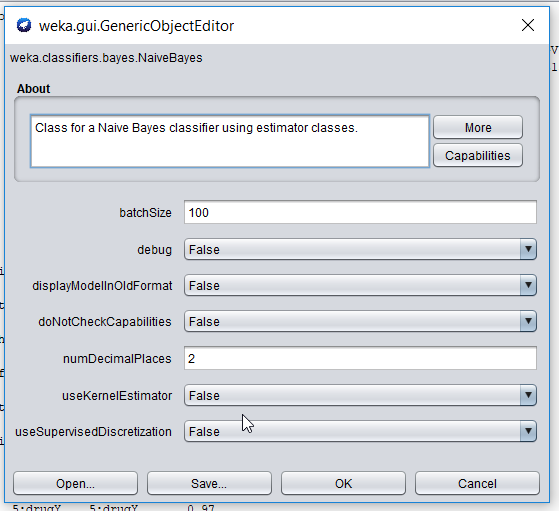
### Bonus

Se probo rápidamente el método Random Forest para probar su efectividad.



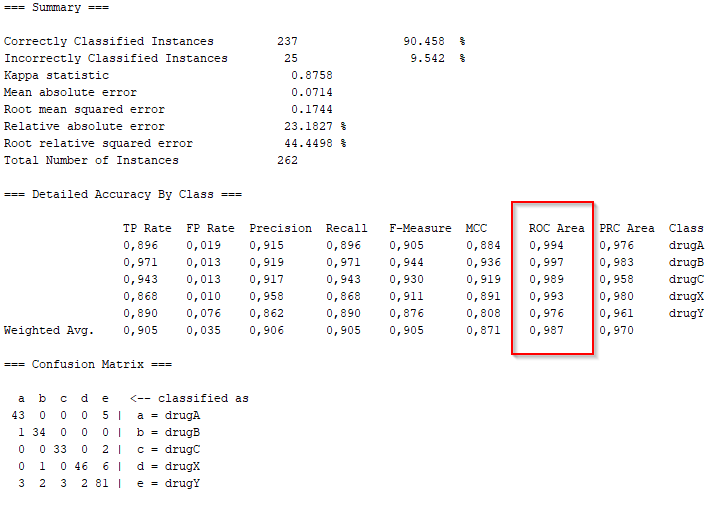
## Método Bayesiano

Se usará a continuación el método bayesiano ingenuo el cual es un clasificador probabilístico que se fundamenta en el teorema de Bayes de probabilidad condicional.



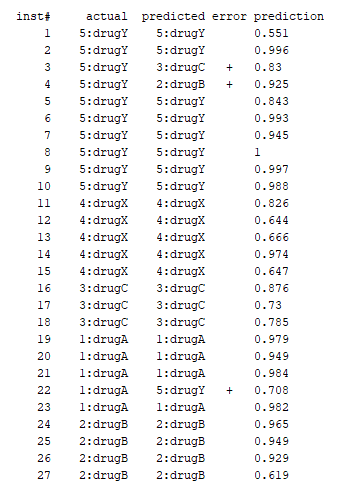
Debido a que este método se fundamenta casi que completamente en el teorema de Bayes no hay muchos parámetros que podamos configurar y optimizar.

Se probo el modelo mediante una validación cruzada con un **fold** de 10



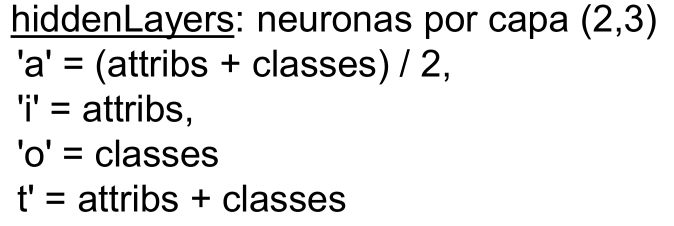
El área debajo de la curva ROC y el porcentaje instancias correctamente clasificados muestran que en términos generales este método funciona bien.

### Probando el método con datos nuevos

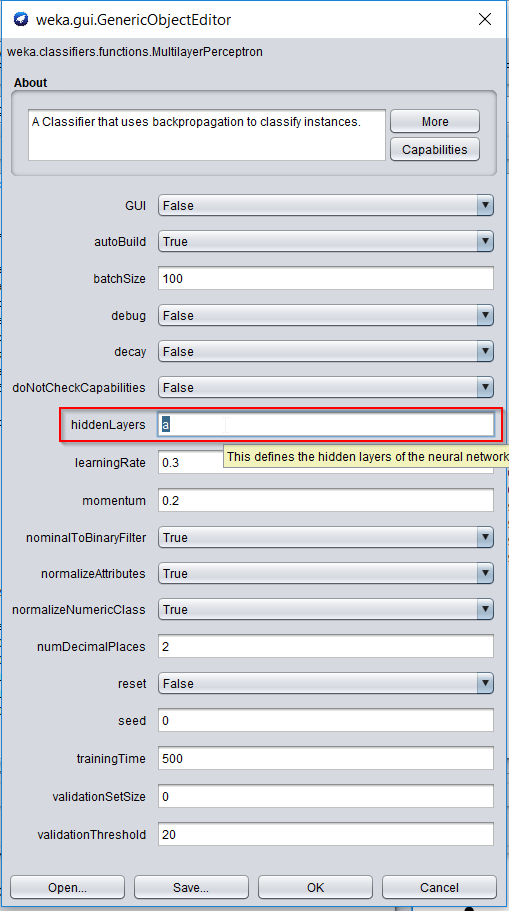


## Redes neuronales

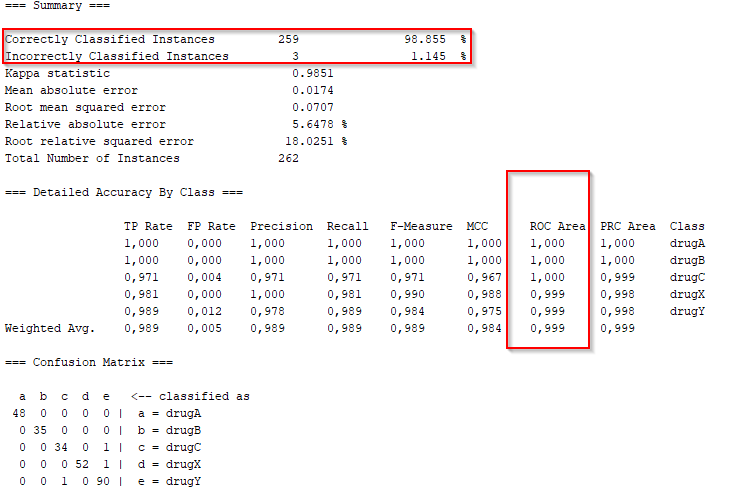
Debido a que en el problema de clasificación que estamos trabajando hay más de 2 clases se procederá a usar un perceptrón multicapa dado que el perceptrón simple tan solo discrimina entre dos clases linealmente separables.



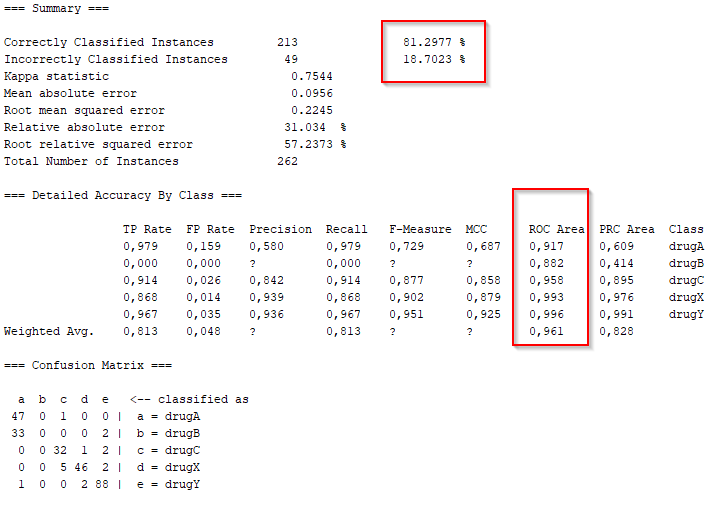
A diferencia del método discutido anteriormente este si tiene varios parámetros para modificar y probar, para probar el modelo usaremos una validación cruzada con 10 folds, lo primero que haremos será probar varios valores para el numero de capaz internas y tamaño de estas.



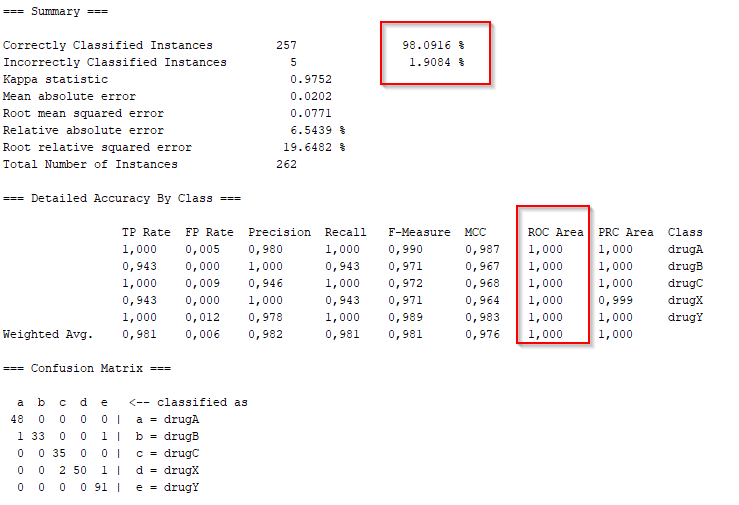
hiddenLayers = ‘a’



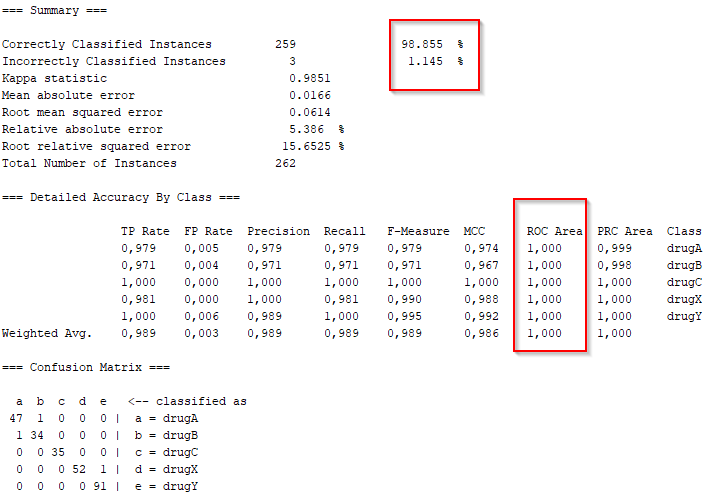
hiddenLayers = 2,5



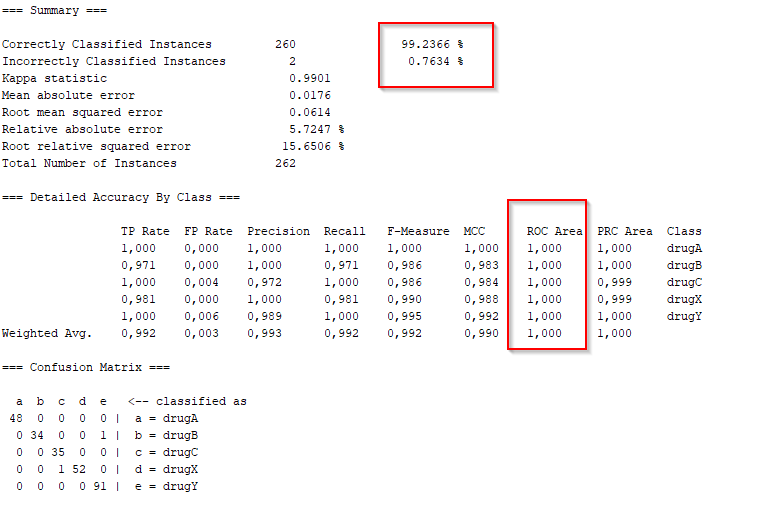
hiddenLayers = 5, 7



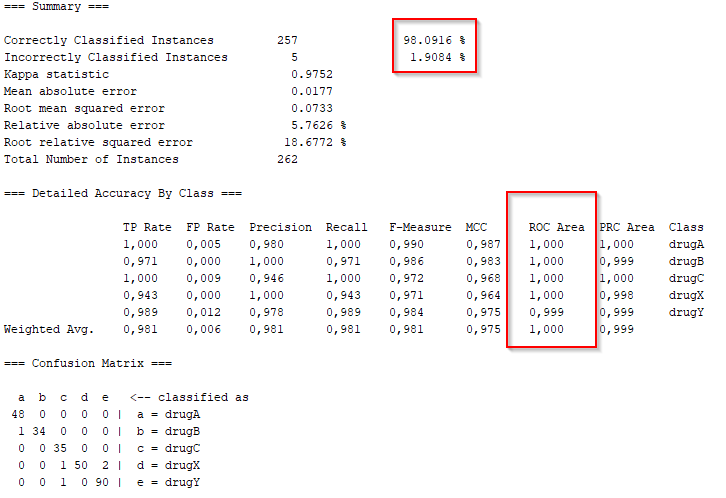
hiddenLayers = 5,5



hiddenLayers = 7,5

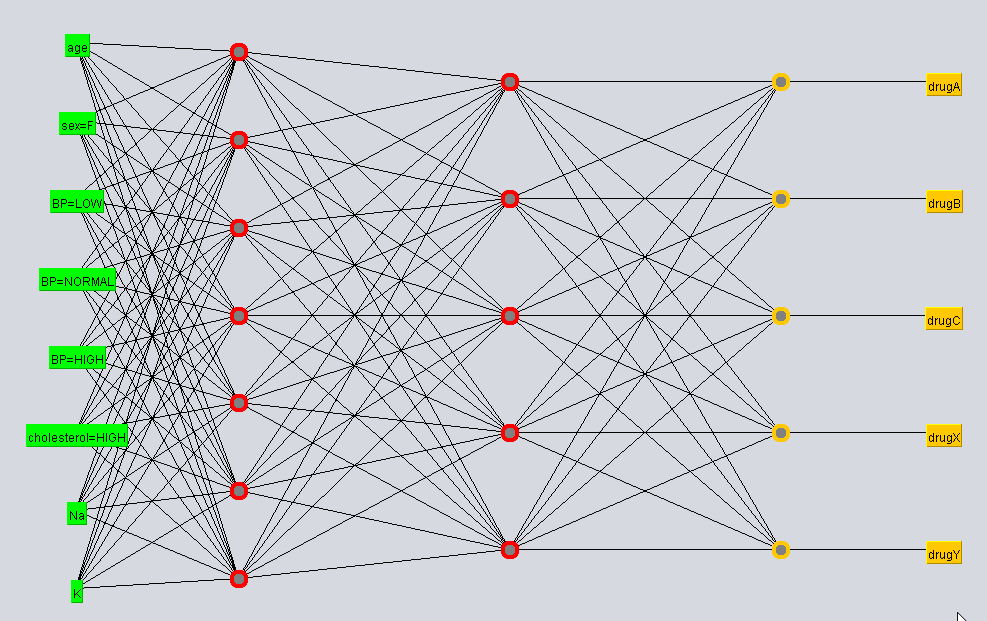


hiddenLayers = 7,7



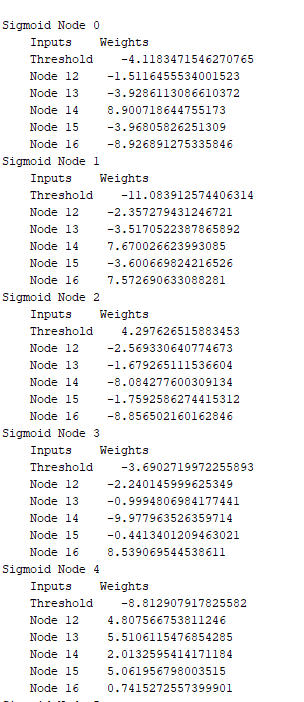
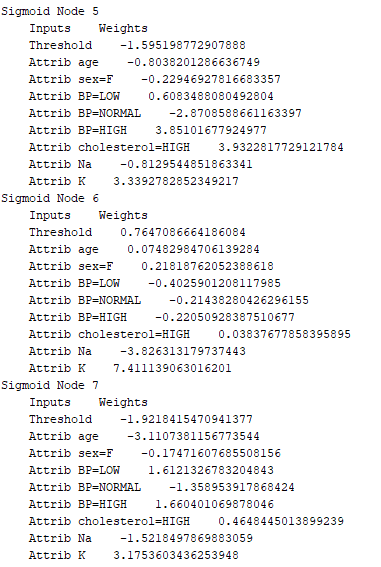
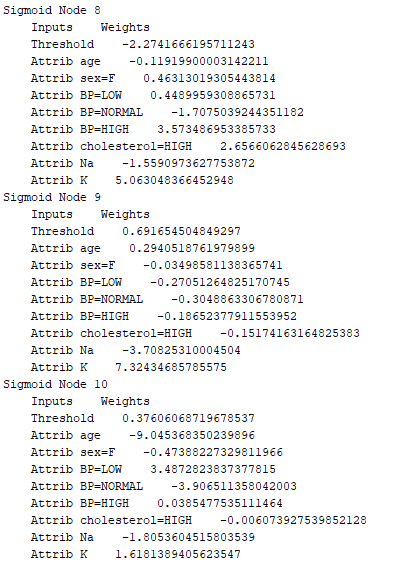
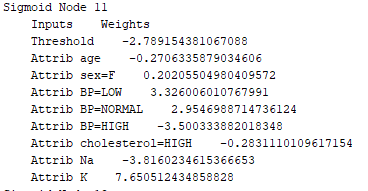
### Observaciones

Luego de probar con varios tamaños para una red neuronal de 2 capas ocultas se encontró un tamaño para las capas que logro una efectividad del 99%

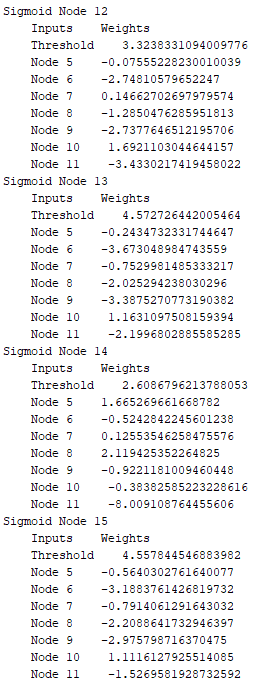
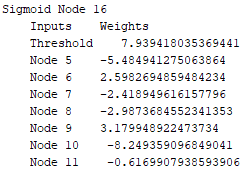


El tamaño de la primera capa es igual al número de atributos de cada instancia (incluyendo el atributo clase) y el tamaño de la segunda capa es el número de valores que tiene la clase.

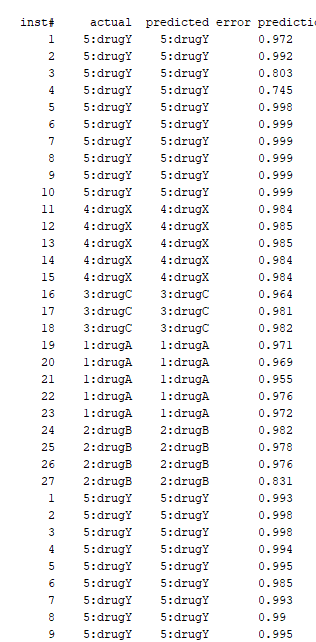
Se observó además que en la capa de entrada no están todos los atributos de las instancias, a continuación, se detallan los pesos calculados para cada nodo.

**PESOS NODOS CAPA DE SALIDA**

**** ****

### Probando el método con datos nuevos



# Conclusiones, Eligiendo a los ganadores

A continuación, se compacta en una tabla el porcentaje de instancias correctamente clasificadas por cada método de aprendizaje.

|  |  |
| --- | --- |
| Método Aprendizaje | % Instancias correctamente clasificadas |
| Regresión logística | 98.00% |
| K-Neighboors | 90.83% |
| Máquina de soporte vectorial | 95.03% |
| Árbol de decisión | 90.83% |
| Método Bayesiano | 90.45% |
| Red Neuronal | 99.23% |

Antes de elegir a un ganador es importante aclarar que no es conveniente basarse únicamente en la efectividad clasificando las instancias de prueba, también se debe tener en cuenta las instancias futuras que se desean predecir, por lo tanto, la posibilidad de sobreajuste es un factor que hay que tener muy en cuenta. Además, también se deben considerar factores subjetivos en el modelo resultante como por ejemplo la facilidad de ser entendido por un humano de tal forma que sea intuitivo para un humano sacar conocimiento valioso de forma eficaz.

Dicho lo anterior la medalla de oro se la lleva la máquina de soporte vectorial por su relativamente alta efectividad y poca posibilidad de sobreajuste debido a que se basa en pocas instancias del dataset para hallar el modelo.

La medalla de plata se la lleva el árbol de decisión porque el modelo resultante tiene una buena efectividad y es muy fácil de entender por humanos, lo cual lo hace ideal para sacar conocimiento intuitivo de forma rápida y eficaz.

La medalla de bronce se la llevan la regresión logística y la red neuronal por su alta efectividad, sin embargo de la red neuronal quedamos con el temor de que el modelo este sobreajustado, la única forma de averiguarlo es con datos de prueba nuevos.