

# Práctica 2 SATD

Pedro Allué Tamargo (758267)      Juan José Tambo Tambo (755742)  
Jesús Villacampa Sagaste (755739)

15 de octubre de 2020

## Índice

1. Ejercicio 1	1
2. Ejercicio 2	1
3. Ejercicio 3	1
4. Ejercicio 4	2
5. Ejercicio 5	3

## 1. Ejercicio 1

Para diferenciar entre los diferentes nodos de tipo *Row Filter*, se han creado dos tablas tal y como se indica en el gui3n.

- *Row Filter*: Permite filtrar elementos de la tabla a partir de una cadena o una expresi3n regular.
- *Nominal Row Filter*: Se puede filtrar manualmente a partir de los atributos de la tabla o mediante expresiones regulares.
- *Reference Row Filter*: Permite filtrar el contenido de una tabla a partir de una segunda tabla. Para ello, se selecciona la columna deseada de cada tabla para realizar una operaci3n “*join*” entre ambas. Tambi3n permite realizar un “*join*” con la condici3n inversa (*exclude*).

## 2. Ejercicio 2

Para verificar que la columna *R3nking* es de tipo *String*, se selecciona esa columna desde la pre visualizaci3n del fichero y se escoge el tipo de dato correspondiente. Desde ah3 tambi3n se puede modificar el nombre de la columna.

Para poder eliminar los comentarios, se deben de modificar las columnas correspondientes a3nadiendo ‘//’ al principio de las mismas para que sean de tipo comentario *Java*. Desde la previsualizaci3n de *Knime*, se selecciona la casilla *Java-Style comments* para poder ignorar los mismos.

La columna *class* se puede eliminar a3nadiendo un elemento de *Column Filter* con el cual se introduce esta columna en el apartado de *exclude*.

Si se desea escribir un nuevo fichero CSV, se debe a3nadir un nuevo componente *CSV writer*. En configuraci3n del m3dulo se selecciona el tipo de separador, la ruta donde debe escribir el archivo y que se incluya la cabecera con la opci3n *write column header*.

## 3. Ejercicio 3

En este ejercicio, la modificaci3n del nombre de la columna “*marcas*” se ha realizado a3nadiendo un nodo *column rename* a continuaci3n de *CSV reader* (mediante el cual se lee el archivo “*data1Nuevo.csv*”).

Si se desea filtrar las filas con el campo *comments* = “.average”, se debe a3nadir un nuevo nodo *Nominal Value Row Filter*. Se selecciona la columna *comments* y en la secci3n de *include* se busca el valor *average*. La columna *ranking* se elimina con el nodo *column filter*.

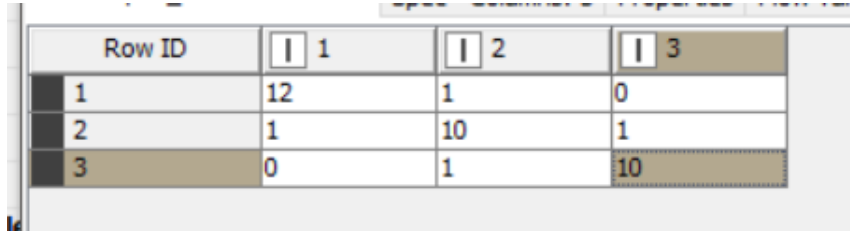
Para la escritura final, se introduce un nodo *CSV writer* y se configura con el modo *append* (se debe indicar que el archivo ya existe), y tabulador como car3cter delimitador.

## 4. Ejercicio 4

Basándonos en el ejemplo de árbol de decisiones de *Knime*, se ha creado un *Workflow* de árbol de decisión para clasificar las muestras de vinos. Para leer el archivo se utiliza el nodo *File Reader*. Se añade un nodo *Partitioning* con el que se selecciona la cantidad de datos de entrenamiento (el 80 %).

Seguidamente se añade un nodo *Decision Tree Learner*. Para evitar errores con este nodo, se debe modificar la columna *class* desde el nodo lector de archivos e indicar que es de tipo *String*. Posteriormente se incorpora un nodo *Decision Tree Predictor* el cual “predice” los valores.

Por último, se añade el nodo *Scorer* para poder observar las predicciones realizadas. Los resultados obtenidos son los siguientes:



Row ID	1	2	3
1	12	1	0
2	1	10	1
3	0	1	10

Figura 1: Matriz de confusión del árbol de decisión entrenado con el 80 % de los datos.

El siguiente cuadro (1) muestra los resultados de ejecutar el *workflow* con un volumen de datos al 80 % con 10 iteraciones:

Clase	Positivos confirmados	Falsos negativos
1	0.91221	0.08778
2	0.93633	0.06366
3	0.94682	0.05317

Cuadro 1: Cuadro de datos comparativos para el caso de entrenamiento con el 50 % de los datos. Los resultados están en tanto por 1.

Si se entrena el árbol de decisión con un volumen de datos inferior al 80 % se obtienen los siguientes resultados (cuadros 2 y 3):

Clase	Positivos confirmados	Falsos negativos
1	0.92442	0.07557
2	0.85028	0.14971
3	0.89869	0.10130

Cuadro 2: Cuadro de datos comparativos para el caso de entrenamiento con el 50 % de los datos. Los resultados están en tanto por 1.

Clase	Positivos confirmados	Falsos negativos
1	0.94393	0.05606
2	0.81067	0.18932
3	0.79701	0.20298

Cuadro 3: Cuadro de datos comparativos para el caso de entrenamiento con el 20 % de los datos. Los resultados están en tanto por 1.

Por lo tanto, al compararse estos datos (cuadros 2 y 3) con los obtenidos en una ejecución entrenada con el 80 % de los datos se llega a la conclusión de que con el 80 % de los datos se obtienen mejores resultados, por encima del 90 % en identificación de positivos y con un fallo inferior al 10 % en falsos negativos.

## 5. Ejercicio 5

Como se pide en el enunciado, a partir del ejercicio 4, se modifica para esta vez entrenar la muestra con un perceptrón multicapa. Por tanto, se cambia el nodo *Decision Tree Learner* por *RProp MLP Learner*, así como el nodo *Decision Tree Predictor* por *MultiLayerPerceptron Predictor*. Además, se añade un nodo *Normalizer* antes de hacer el particionado de los datos. Los resultados han sido los siguientes:

Clase	Positivos confirmados	Falsos negativos
1	1	0
2	0,94432	0,05567
3	0,96087	0,03912

Cuadro 4: Cuadro de datos comparativos para el caso de entrenamiento con el 80 % de los datos (10 ejecuciones) con un perceptrón multicapa. Los resultados están en tanto por 1.

Clase	Positivos confirmados	Falsos negativos
1	0,98833	0,01166
2	0,92275	0,07724
3	0,99019	0,00980

Cuadro 5: Cuadro de datos comparativos para el caso de entrenamiento con el 50 % de los datos (10 ejecuciones) con un perceptrón multicapa. Los resultados están en tanto por 1.

Clase	Positivos confirmados	Falsos negativos
1	0,95236	0,04763
2	0,88767	0,11232
3	0,95010	0,04989

Cuadro 6: Cuadro de datos comparativos para el caso de entrenamiento con el 20 % de los datos (10 ejecuciones) con un perceptrón multicapa. Los resultados están en tanto por 1.

Por lo tanto, al compararse estos datos se llega a la conclusión de que con el 80 % de los datos se obtienen mejores resultados, por encima del 94 % en identificación de positivos y con un fallo inferior al 6 % en falsos negativos.

Con el fin de obtener un rendimiento superior, se parte de la base del entrenamiento con el 80 % de los datos, ya que ha sido el que mejor rendimiento ha dado y se modifican diversos parámetros. Los parámetros que se pueden modificar son el número de iteraciones, el número de capas ocultas y el número de neuronas por capa. En el entrenamiento inicial los valores eran los siguientes:

- Número de iteraciones: 100
- Capas ocultas: 1
- Neuronas por capa: 10

Se han realizado diversas pruebas para ajustar estos parámetros y obtener el mejor resultado.

- Número de iteraciones: 1000
- Capas ocultas: 3
- Neuronas por capa: 10

Con estos valores los resultados han mejorado considerablemente, como se observa en el siguiente cuadro.

Clase	Positivos confirmados	Falsos negativos
1	0,9772	0,0227
2	0,9565	0,0435
3	1	0

Cuadro 7: Cuadro de datos comparativos para el caso de entrenamiento con el 80 % de los datos (1000 ejecuciones) con un perceptrón multicapa con tres capas ocultas. Los resultados están en tanto por 1.

- Número de iteraciones: 5000
- Capas ocultas: 5
- Neuronas por capa: 10

Clase	Positivos confirmados	Falsos negativos
1	0,963636364	0,036363636
2	0,953947368	0,046052632
3	0,96	0,04

Cuadro 8: Cuadro de datos comparativos para el caso de entrenamiento con el 80 % de los datos (5000 ejecuciones) con un perceptrón multicapa con 5 capas ocultas y 10 neuronas por capa. Los resultados están en tanto por 1.

Con estos valores los resultados han mejorado con respecto al inicial pero han empeorado con respecto a los anteriores valores.

- Número de iteraciones: 1000
- Capas ocultas: 3
- Neuronas por capa: 100

Clase	Positivos confirmados	Falsos negativos
1	1	0
2	0,975	0,025
3	1	0

Cuadro 9: Cuadro de datos comparativos para el caso de entrenamiento con el 80 % de los datos (1000 ejecuciones) con un perceptrón multicapa con tres capas ocultas y 100 neuronas por capa. Los resultados están en tanto por 1.

Con estos valores se han obtenido los mejores resultados, obteniendo casi una efectividad del 100 %, excepto para los elementos de la clase 2. Sin embargo, el modelo tarda más tiempo en entrenar.

En comparación con el método *Tree Learner*, se puede considerar que el *Perceptrón Multicapa* obtiene mejores resultados para cualquier porcentaje de datos entrenados, como se puede observar en los cuadros anteriores. El porcentaje de positivos confirmados supera el 90 % salvo en una ocasión, a diferencia de los resultados obtenidos con el *Tree Learner*, los cuales rondan el 80 e incluso 70 % de aciertos en algunos casos.

Para obtener mayor precisión en los resultados, el programa *Knime* recomendaba el uso de un nodo *Normalizer* junto al nodo *MLP Learner* para normalizar los datos de entrada mediante el método *Z-Scorer Normalization*.