Escuela Técnica Superior de Ingenierías Informática y de Telecomunicaciones

Máster Oficial en Ingeniería Informática

Curso 2020/2021

P1 – PREPARACIÓN DE DATOS

Tratamiento Inteligente de Datos

Breve descripción

Memoria sobre lo desarrollado en la práctica

Autor

Álvaro de la Flor Bonilla

RESUMEN

El objetivo de este documento es hacer un análisis y dar respuesta a las actividades planteadas en la práctica

1 ÍNDICE

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

Como pasos previos antes de realizar cualquier tarea se ha realizado lo siguiente.

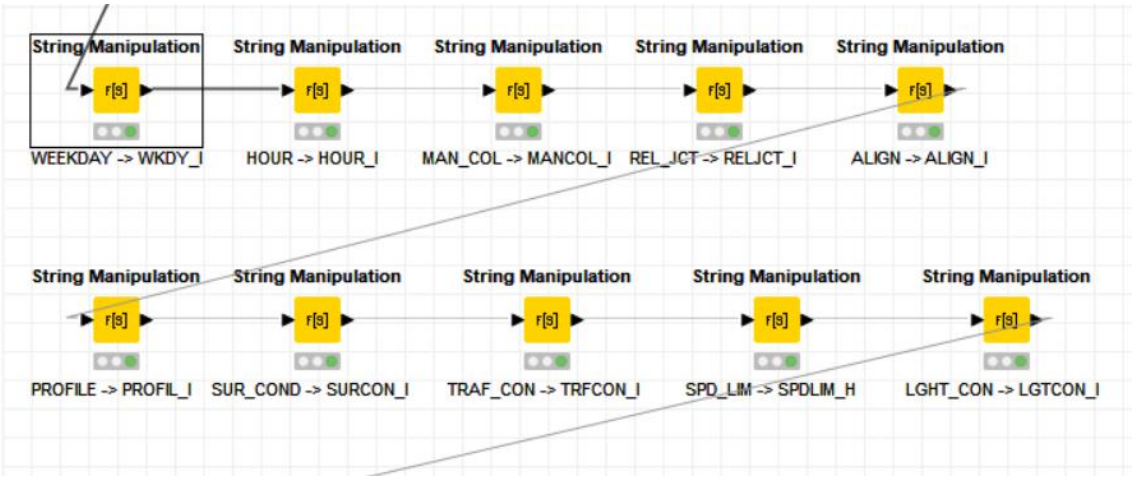


Ilustración 1 - Eliminación de columnas

En primer lugar, se han eliminado las columnas duplicadas de los valores **no imputados**. Es decir, en los casos donde se duplicaban columnas para añadir valores imputados se ha eliminado la columna inicial.

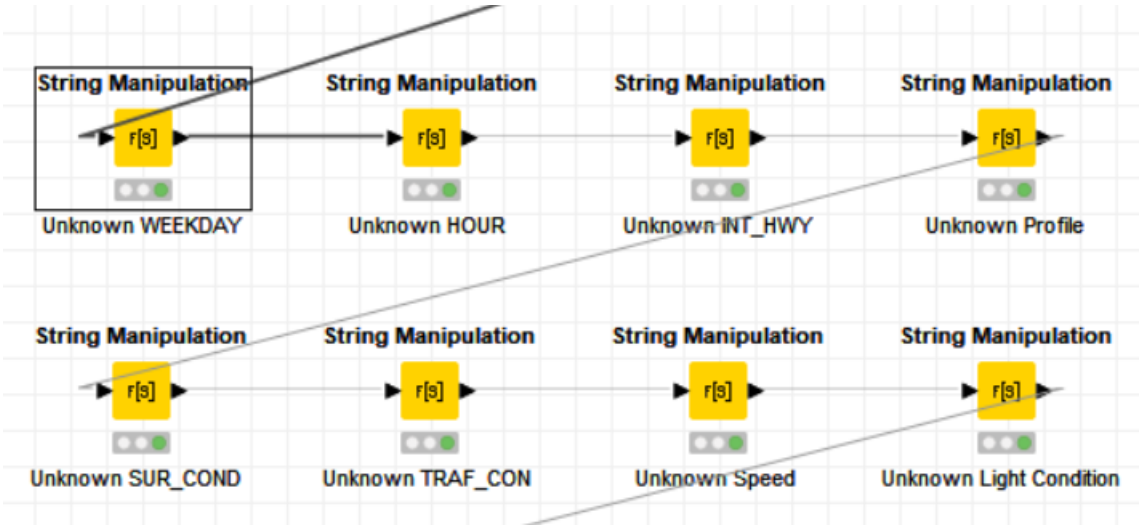


Ilustración 2 - Valores perdidos a blanco

Posteriormente, se han editado todas las columnas, de tal forma que las celdas que contenían valores desconocidos (calificadas con el valor 9 o 99 dependiendo del caso), han sido dejadas en blanco.

Por último, se ha construido la variable clase como dependencia de las variables *FATALITIES* (60), *INJURY_CRASH* (30) y *PRPTYDMG_CRASH* (10).

2 DISCRETIZACIÓN

2.1 Ejecutar el algoritmo de prueba para clasificación (C4.5) y estudiar cómo divide las características numéricas en los árboles de decisión aprendidos

El *"workflow"* utilizado ha sido el de la siguiente imagen.

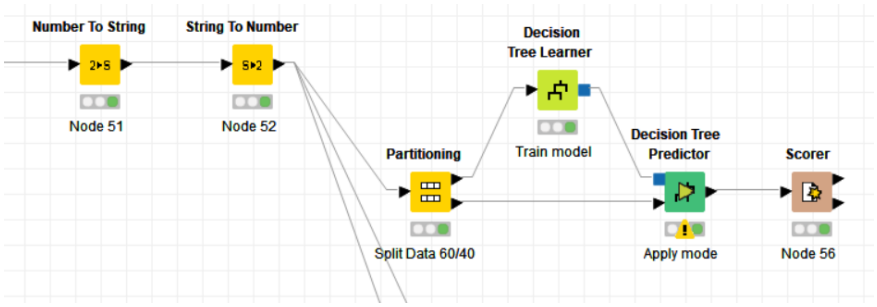


Ilustración 3 - Workflow Discretización 1

El único tratamiento realizado al *"workflow"* anterior ha sido el comentado en la sección de introducción, aparte de ello solo se ha hecho lo que aparece en la captura anterior. Como resultado del modelo anterior, se han obtenido los siguientes árboles de decisión.

Cabe destacar que, en un primer momento de la elaboración de la práctica, no se borraron las columnas con la que se construyó la variable de clase, por lo que se obtenían valores como el siguiente:

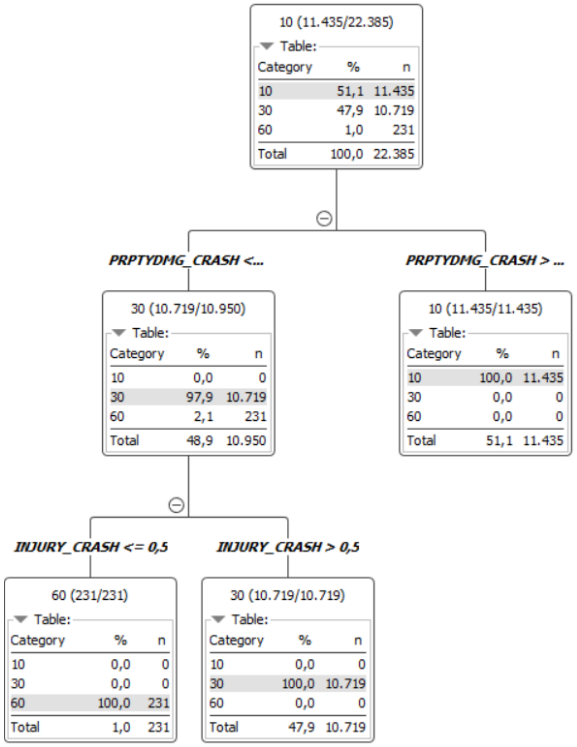


Ilustración 4 - Primer árbol de decisión

Una vez borradas estas tres columnas se volvieron a tener árboles de decisión en este caso con muchas más lógica que el mostrado anteriormente. La única diferencia fue añadir un nodo *"Colum Filter"* justo antes del particionado de los datos.

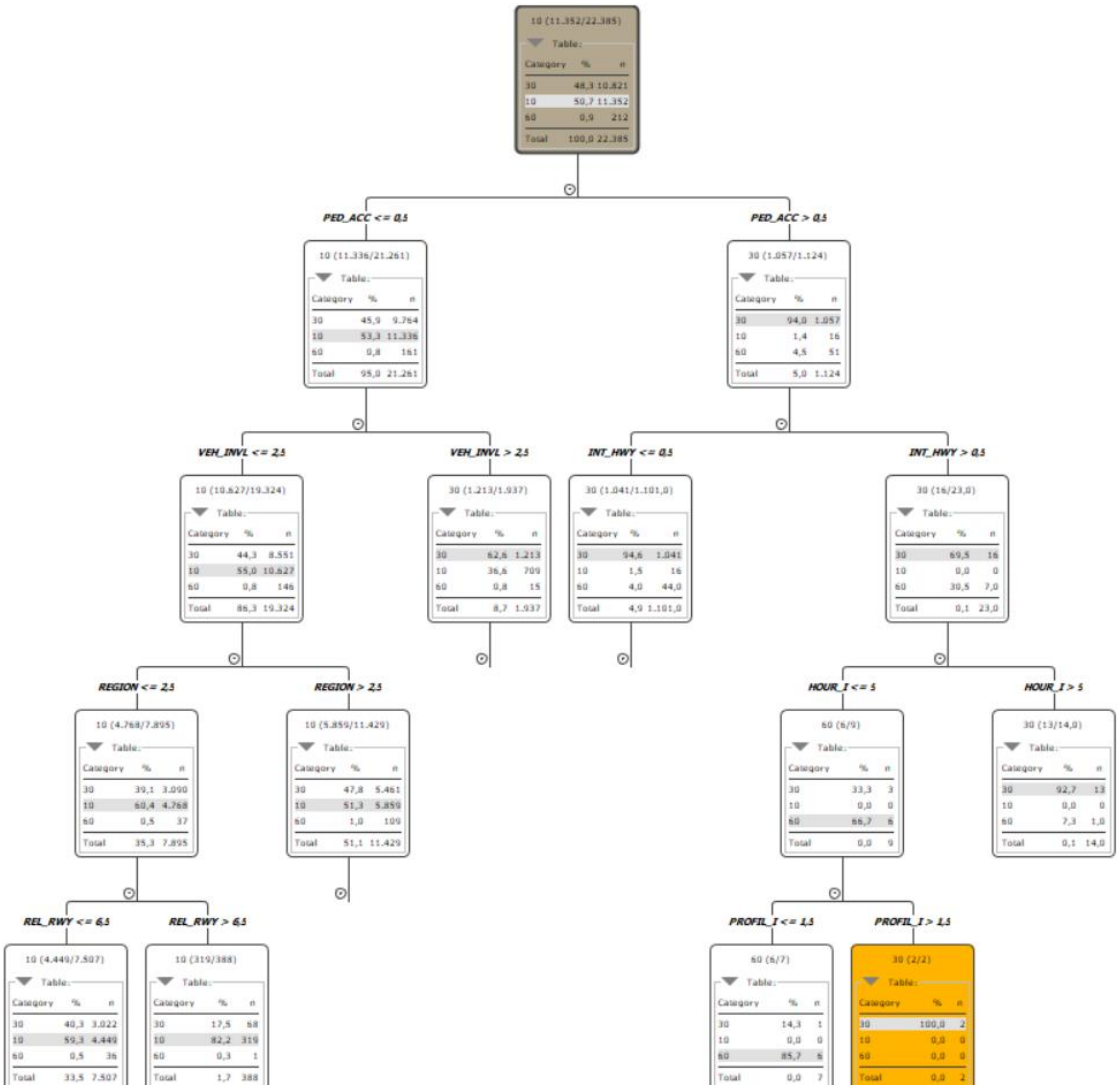


Ilustración 5 - Árbol de decisión corregido

Se obtiene un árbol de decisión enorme (en la imagen de arriba aparecen ocultados muchos nodos). Aun así, hemos sido capaz de encontrar una rama en la que tras la combinación de varios atributos el algoritmo es capaz de asegurar que se producirá un accidente con daños físicos al 100% siguiendo el esquema PED_ACC > 0.5, INT_HWY>0.5, HOUR_I > 5 y PROFIL_I > 1.5.

La precisión del algoritmo es la siguiente:

Confusion Matrix - 0:56 - Scorer

File Hilite

PREDICT \ ...	10	30	60
10	10000	7108	24
30	7339	8703	88
60	107	196	14

Correct classified: 18.717 Wrong classified: 14.862

Accuracy: 55,74 % Error: 44,26 %

Cohen's kappa (κ) 0,125

Ilustración 6 - Confusion Matrix I

2.2 Aplicar el algoritmo de discretización top-down CAIM sobre las características numéricas y comprobar el comportamiento del algoritmo de prueba.

En primer lugar, vamos a presentar el *"workflow"* utilizado.

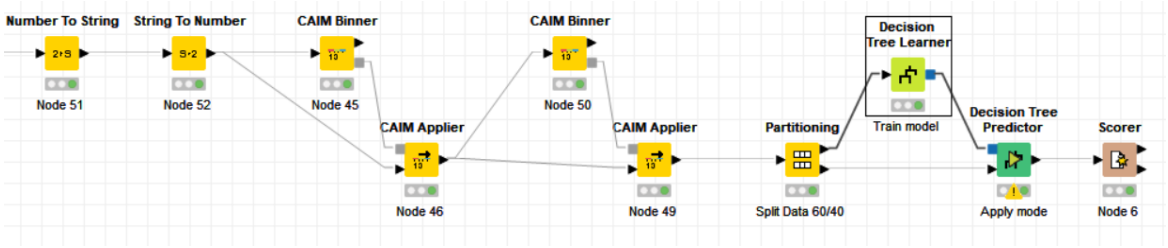


Ilustración 7 - Workflow Discretización 2

En este primer *"workflow"* para la discretización utilizada en el que una vez más, no aparecen los procesos descritos en la introducción. Tal y como se mostró en teoría, vamos a utilizar el algoritmo *"CAIM"*, el cual comienza por el intervalo total y lo va dividiendo (*"top-down"*). En la imagen anterior se muestra cómo se aplica el algoritmo de discretización solicitado en este caso.

- Algoritmo de discretización supervisado
- Necesita de un conjunto de entrenamiento
- Busca el menor número de intervalos
  - Comienza con el intervalo total y lo va dividiendo (top-down)
- Maximiza el número de ejemplos de la misma clase en el mismo intervalo

Ilustración 8 - Algoritmo CAIM

Tras realizar este proceso, obtenemos el siguiente árbol de decisión.

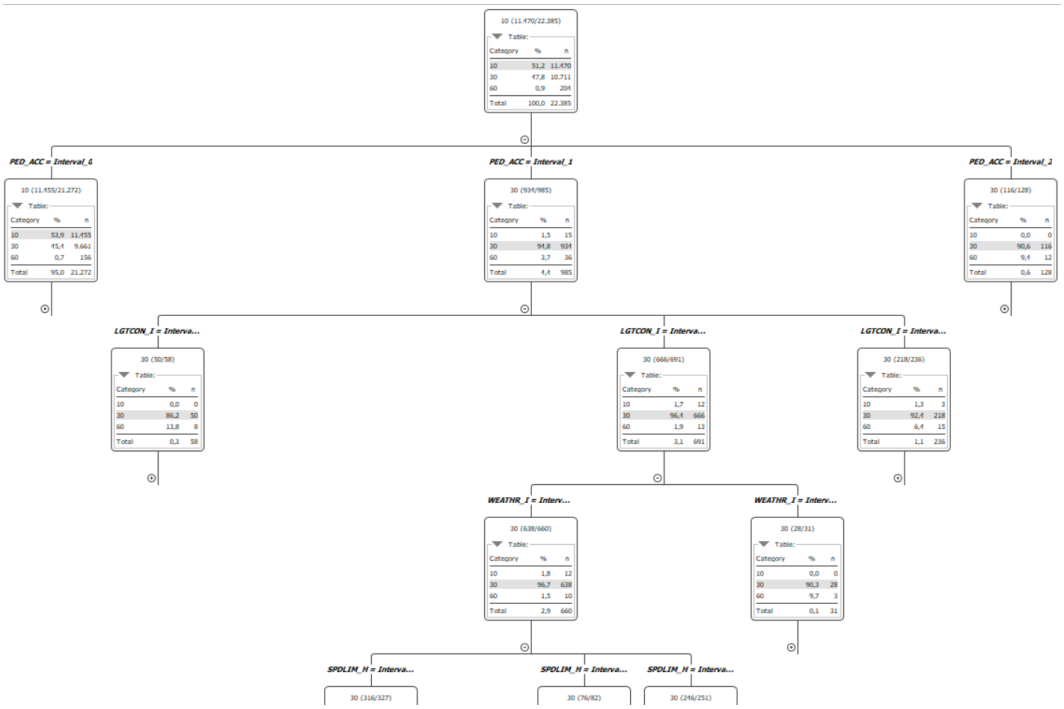


Ilustración 9 - Árbol de decisión con atributos discretizados

Como dato curioso en este árbol podemos señalar que se ha acentuado la posibilidad de que el accidente que se produzca sea del tipo 10 (solo daños materiales). De hecho, ha pasado de 48.3% al 51.2%, casi un punto. Además, todo este valor se le ha restado a los accidentes del tipo 30 (con daños físicos pero sin muertes).

Una vez más, el árbol de decisión que se muestra es capaz de llegar a un porcentaje de acierto del 100% sobre el accidente de tipo 30 con tan solo bajar 4 ramas, es decir, con tan solo utilizar 4 atributos.

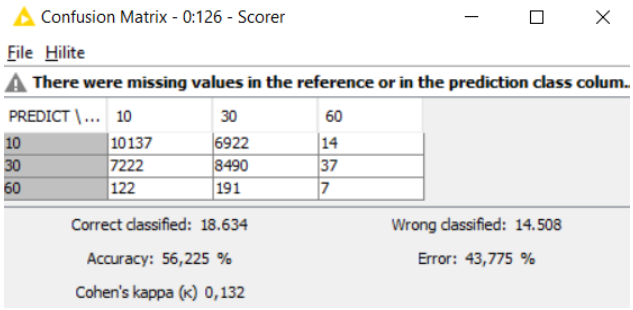


Ilustración 10 - Confusion Matrix II

En cuanto a resultado, como puede comprobar tras discretizar todos los atributos se ha conseguido llegar hasta el 56,23% de exactitud, un punto más que en el caso anterior en el que los casos no estaban tratados.

2.3 Estudiar las distintas características categóricas y proponer una discretización de las mismas basándose en el significado de la característica, la visualización de los datos, etc. Del mismo modo, proponer una discretización de las características numéricas basándose en su significado (por ejemplo, accidentes en hora punta o no). Estudiar los resultados sobre el algoritmo de prueba.

Cabe destacar que alguno de los algoritmos que utilizamos (como por ejemplo C4.5) tienen incorporados mecanismos para la discretización de atributos continuos que se realiza en el proceso de clasificación.

En mi opinión, basándonos en el significado de las características solo discretizaría la columna del límite de velocidad y la de la hora del accidente.

Como resultado de este, obtenemos el siguiente árbol de decisión.

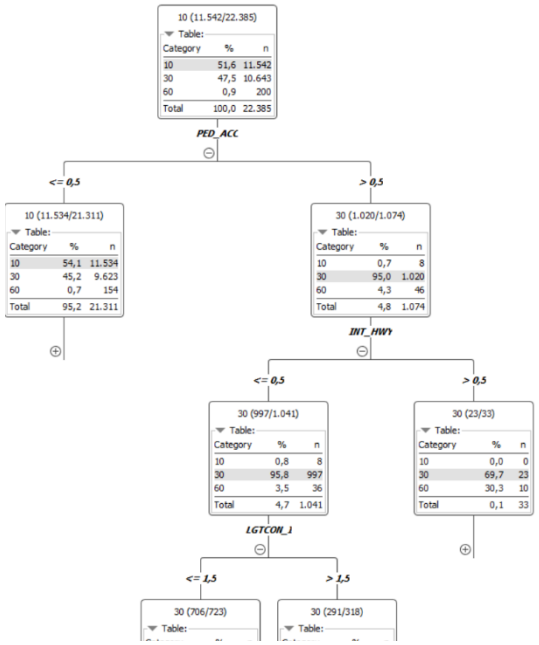


Ilustración 11 - Segundo árbol de decisión

En este se acentúa aún más la probabilidad de que el accidente sea del tipo 10 (solo daños materiales), llegando al 51.6%. Sin embargo, en esta ocasión al tener muchos menos parámetros discretizados alcanzar una solución válida certera (100%) en comparación con el apartado anterior.

Los parámetros que se le dan más importancia, es decir, que aparecen en las primeras ramas del algoritmo siguen siendo los mismos que en el caso anterior una vez más.

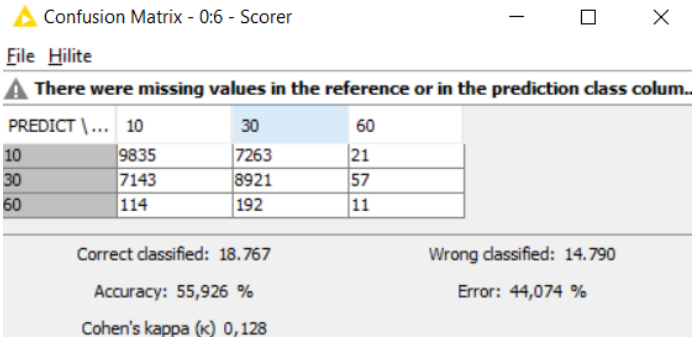


Ilustración 12 - Confusion Matrix III

En este caso los resultados son mejores que en el primer entrenamiento, pero peores que en los que se discretizaron todas las características, por tanto, como resultado podemos destacar que necesitamos ajustar un poco más las variables a discretizar.

3 VALORES PERDIDOS

3.1 Ejecutar el algoritmo de prueba para clasificación (C4.5) sobre los datos sin imputar y comprobar el comportamiento de este algoritmo para tratar implícitamente datos perdidos.

En primer lugar, para realizar este apartado hemos tenido que eliminar los datos imputados (justo lo contrario que se realizó en la parte de introducción).

Ilustración 13 - Resumen Workflow

En resumen, en la parte de arriba de nuestro *"workflow"* se utilizan variables imputadas mientras que en la parte inferior se usan las columnas no imputadas.

Como resultado, se ha obtenido el siguiente árbol de decisión.

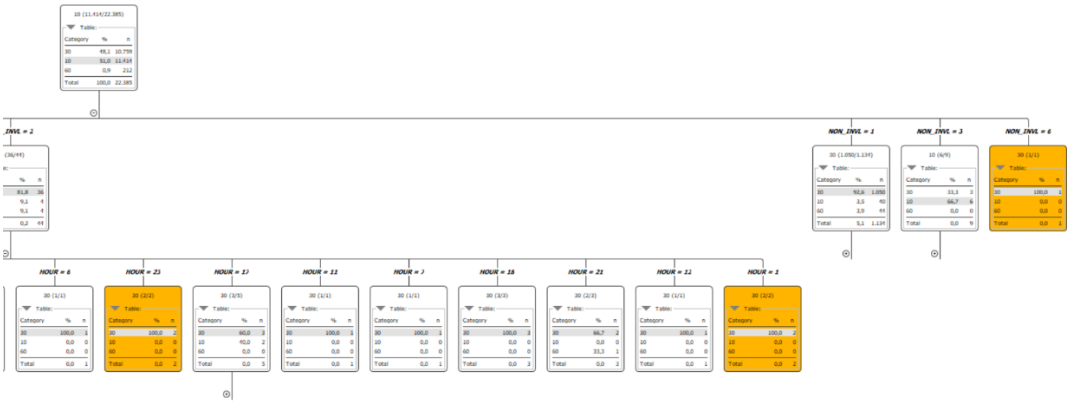


Ilustración 14 - Tercer árbol de decisión

Como puede comprobar, se han obtenido datos muy curiosos. El porcentaje de probabilidad de los accidentes sigue estando estable (10 -> 51%, 30 -> 48% y 60 -> 1%), sim embargo, como puede apreciar en la imagen de arriba con tan solo bajar en un nivel en el árbol este ya nos afirma con una rotundidad del 100% que si en el accidente están implicadas más de 6 motocicletas este tendrá daños físicos. Así como para otros muchos más atributos de una forma muy rápida.

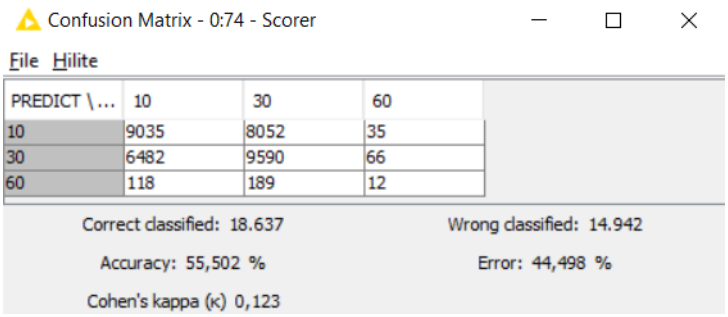


Ilustración 15 - Confusion Matrix IV

En cuanto a resultados, puede ver el porcentaje de acierto en función de la imagen de arriba.

3.2 Ejecutar el algoritmo de prueba con los datos imputados disponibles y comprobar su comportamiento.

Los datos mostrados en el apartado anterior (el 2.1) fueron realizados utilizando los datos imputados, por lo que se considera que los resultados que se van a obtener en este apartado son exactamente los mismos.

3.3 Imputar valores perdidos con la media o moda, según proceda, y comprobar el comportamiento del algoritmo de prueba.

Para rellenar valores vacíos simplemente se ha utilizado la funcionalidad de KNIME específicamente diseñada para ello, justo antes de aplicar el algoritmo.

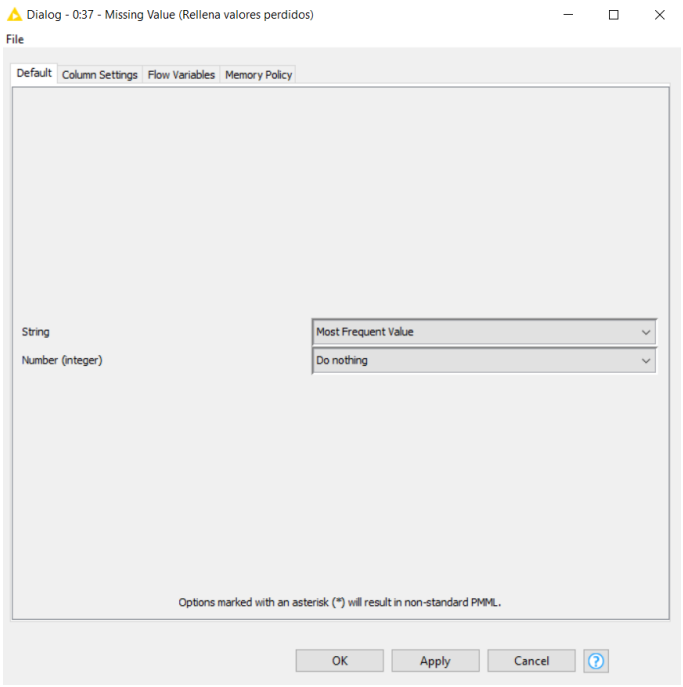


Ilustración 16 - Insertar valores perdidos

Como resultado, se ha obtenido el siguiente árbol de decisión.

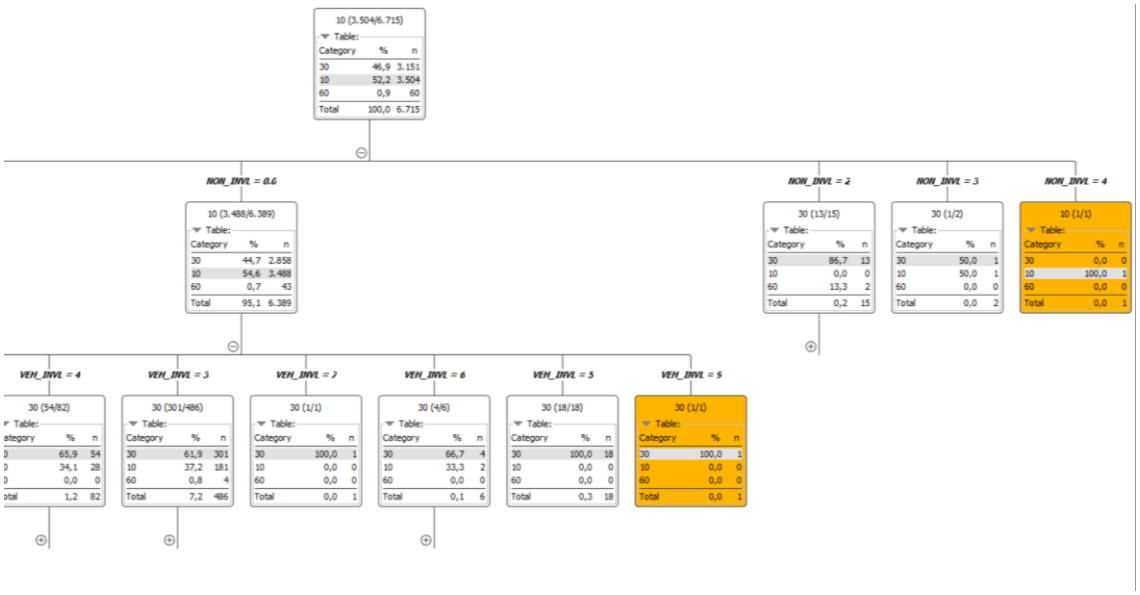


Ilustración 17 - árbol con tratamiento de valores perdidos

En la imagen superior puede ver la actualización del árbol de decisión en este caso podemos destacar que obtenemos una solución con el 100% de probabilidad en la primera rama. En este caso se confirma que si en el accidente intervienen únicamente 4 vehículos que no son motos con el 100% de posibilidad el accidente solo tendrá daños materiales. Por otro lado, si no intervienen motos, pero si exactamente 5 coches el accidente tendrá daños físicos (pero no habrá muertes).

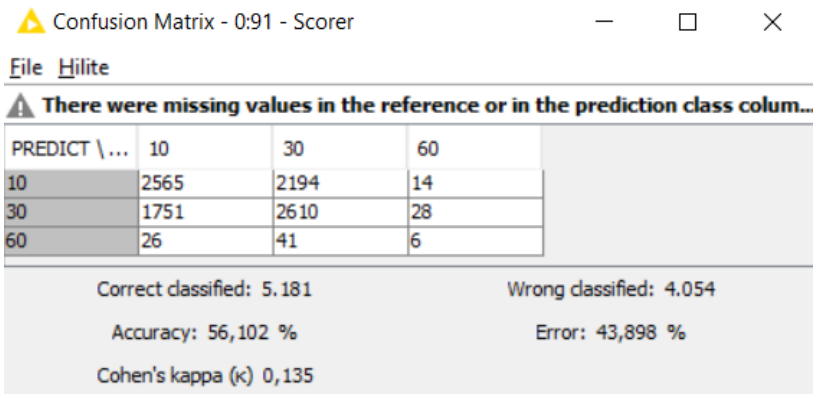


Ilustración 18 - Confusion Matrix V

Como puede ver, el porcentaje de acierto ha subido, lo cual significa que la estrategia para resolver los valores perdidos ha sido un éxito.

3.4 Eliminar las instancias que contienen algún valor perdido y comprobar el comportamiento del algoritmo de prueba.

Entiendo por eliminar instancias a eliminar las filas que contienen valores perdidos.

Para ello se ha utilizado el elemento *"Missing Value"*, con la opción configurada *"Remove Row"*.

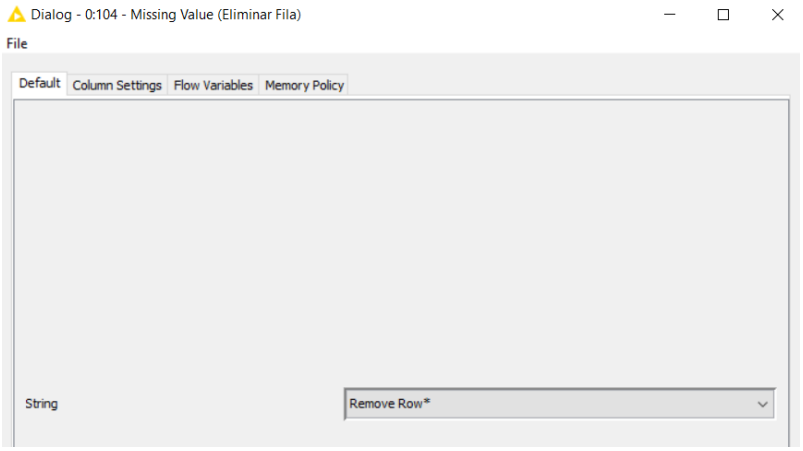


Ilustración 19 - Eliminar instancias con valor perdido

Con esta opción, junto con la parte de pretratamiento que explicamos en la parte de introducción (dejar en blanco los valores desconocidos como 9 o 99 en otros casos) somos capaces de eliminar las filas donde desconocemos algún valor.

Como resultado, obtenemos el siguiente árbol de decisión.

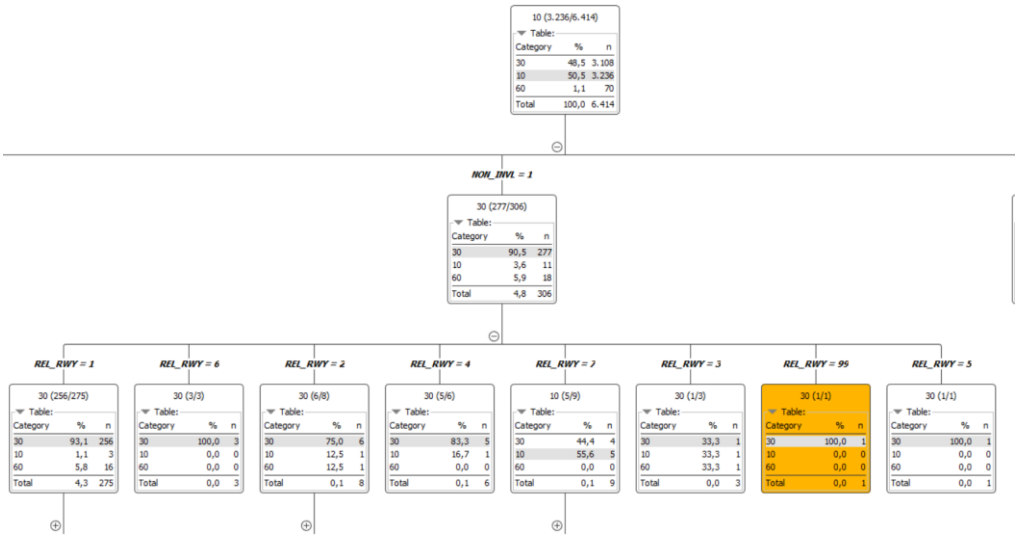


Ilustración 20 - Eliminación de filas con datos perdidos

Como puede comprobar el árbol ha variado mucho al caso anterior. Una de las modificaciones más llamativas es que la segunda característica más importante ya no es *"VEH_INVL"* sino *"REL_RWY"* en esta ocasión. Seguimos obteniendo el primer resultado con 100% de probabilidad en el segundo nivel del árbol.

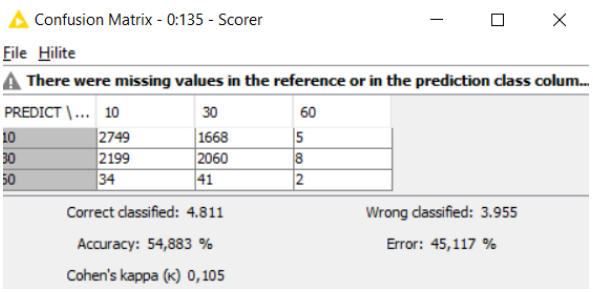


Ilustración 21 - Confusion Matrix VI

En este caso el porcentaje de acierto con la evaluación del algoritmo ha bajado, por lo que podemos garantizar que la eliminación de filas que contienen valores perdidos es una mala elección para su tratamiento en este caso.

3.5 Eliminar las características con valores perdidos y comprobar el comportamiento del algoritmo de prueba.

Por eliminar características entendemos la eliminación completa de las columnas que contienen valores perdidos, para ello podemos utilizar la opción *"Missing Value Column Filter"* que realiza una función muy parecida el subpunto anterior.

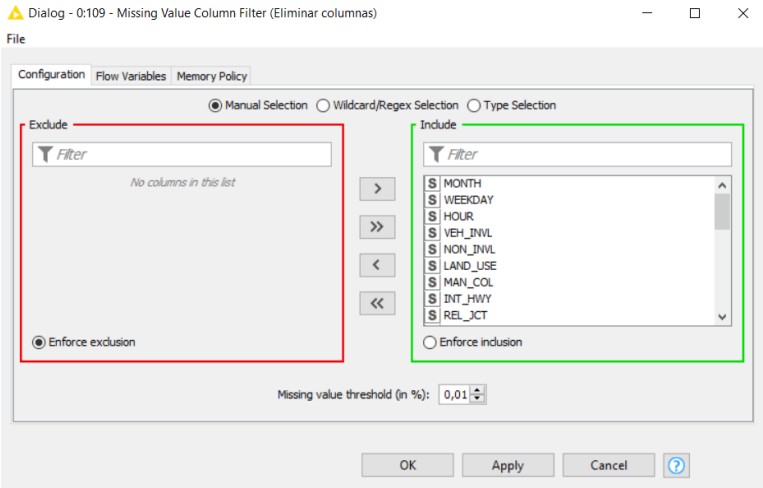


Ilustración 22 - Eliminación de filas

Incluimos todos los puntos y simplemente damos un valor muy bajo al filtro, para que en el momento que haya algún valor perdido se produzca la eliminación.

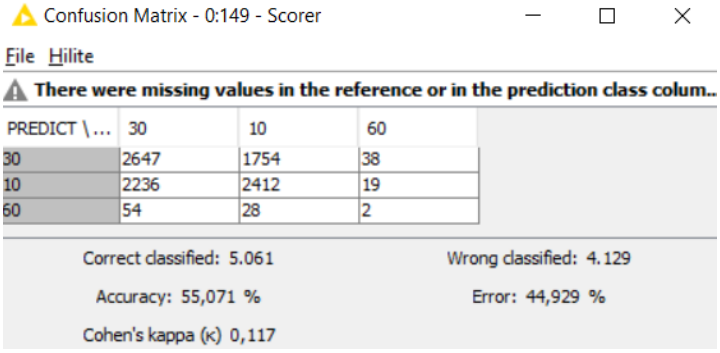


Ilustración 23 - Confusion Matrix VII

Como análisis de este tratamiento para los datos perdidos podemos garantizar que es mejor que la eliminación que las filas, sin embargo, tampoco llega a ser un procedimiento acertado ya que en este caso ha mostrado ser el segundo peor valor obtenido hasta el momento.

3.6 Emplear un algoritmo de predicción (clasificación o regresión, según la naturaleza de la variable) para imputar valores perdidos y comprobar su comportamiento con el algoritmo de prueba.

4 SELECCIÓN DE CARACTERÍSTICAS

4.1 Ejecutar el algoritmo de prueba para clasificación (C4.5) sobre el conjunto de datos completo. El modelo generado por este algoritmo es posible que no emplee todas las características disponibles, por lo que ya estará realizando una selección de ellas de forma implícita.

El *"workflow"* que se ha utilizado es el siguiente.

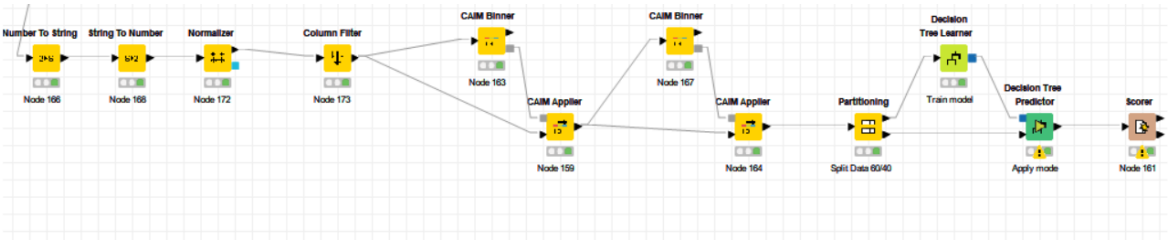


Ilustración 24 - Workflow de sección de características

En este apartado se han realizado dos actividades importantes.

En primer lugar, tal y como indica el enunciado se han normalizado los datos, para mejorar la eficiencia computacional. Para ello simplemente hemos agregado el nodo *"Normalizer"*.

Por otro lado, hemos seleccionado las características que tras diversos estudios hemos comprobado que son las más relevantes para la construcción de los árboles de decisión.

Confusion Matrix - 0:161 - Scorer

File Hilite

There were missing values in the reference or in the prediction class colum..

PREDICT \ ...	10	30	60
10	12413	4642	0
30	9302	6885	11
60	156	162	3

Correct classified: 19.301 Wrong classified: 14.273

Accuracy: 57,488 % Error: 42,512 %

Cohen's kappa (κ) 0,152

Ilustración 25 - Confusion Matrix VIII

Como resultado al procedimiento anterior hemos logrado conseguir la mejor relación de exactitud obtenida hasta el momento, el 57,49% de exactitud.

4.2 Aplicar una selección de características envolvente hacia atrás (backward). También se puede decidir eliminar o forzar a conservar, según el caso, algunas características basándose en algún criterio (por ejemplo, tras una visualización de datos) con el objetivo de reducir el coste computacional.

Se ha aplicado el nodo *"Backward Feature Elimination"* y se ha obtenido como resultado la eliminación de todas las características, salvo *"NON_INVL"*.

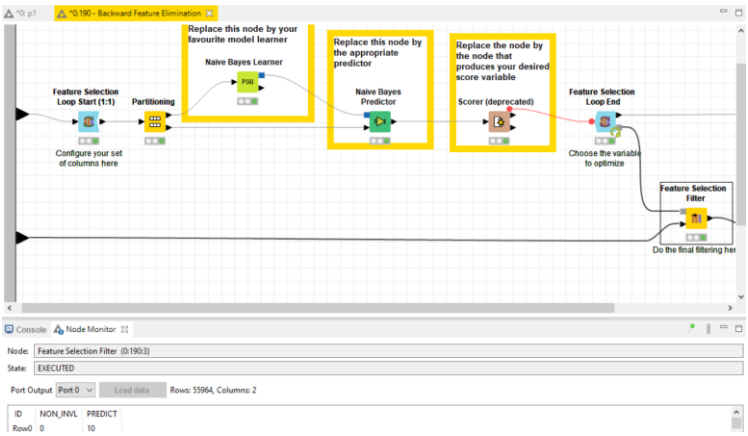


Ilustración 26 - Nodo Backward Feature Elimination

5 SELECCIÓN DE INSTANCIAS

5.1 Aplicar técnicas de muestreo y comprobar su comportamiento en el algoritmo de prueba (C4.5).

Para reducir el número de instancias se ha utilizado el nodo *"Partitioning"* tal y como se muestra en el siguiente *"workflow"*.

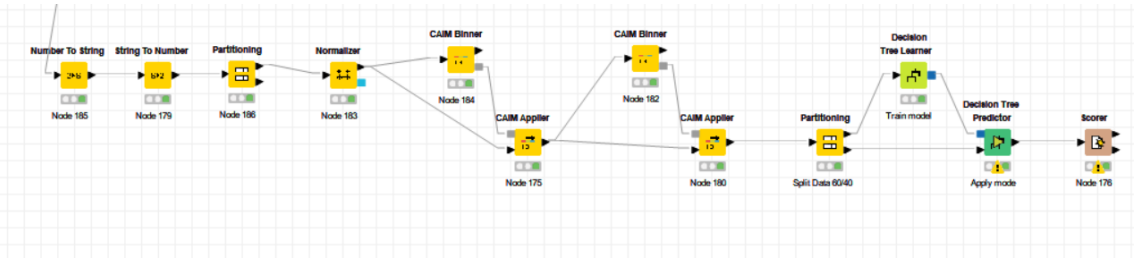
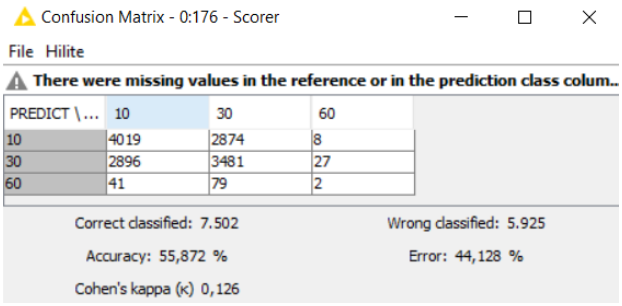


Ilustración 27 - Workflow con el uso de partitioning

En concreto se ha usado una configuración aleatoria el 40% total del que se mostraban en los datos iniciales.


Confusion Matrix - 0:176 - Scorer

File Hilite

There were missing values in the reference or in the prediction class colum..

PREDICT \ ...	10	30	60
10	4019	2874	8
30	2896	3481	27
60	41	79	2

Correct classified: 7.502 Wrong classified: 5.925

Accuracy: 55,872 % Error: 44,128 %

Cohen's kappa (κ) 0,126

Ilustración 28 - Confusion Matrix IX

Como resultado el porcentaje de exactitud es del 55,88% tras realizar este procedimiento.

5.2 El conjunto de datos contiene una categoría de la clase mucho más infrecuente que el resto, lo que hace que los datos no estén balanceados. Analizar esta situación, realizar una reducción de datos mediante muestreo aleatorio que equilibre la frecuencia de la clase y comprobar el efecto en el algoritmo de prueba. Para este análisis se puede reducir el problema a una clase binaria (por ejemplo, clase minoritaria frente al resto) y estudiar la matriz de confusión obtenida para los modelos aprendidos con y sin datos balanceados.

Para realizar este apartado hemos añadido al apartado anterior 3 nodos, de tal forma que hemos transformado el resultado de la columna final de predicción en un resultado binario: 1 en el caso de que se produzca una muerte y 0 en caso contrario.

El *"workflow"* aplicado ha sido el siguiente:

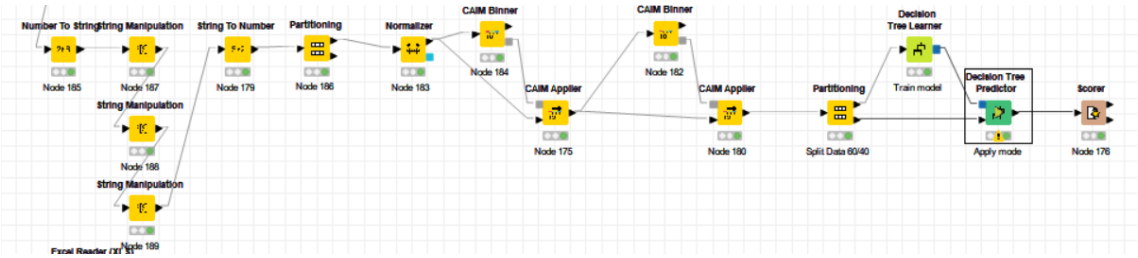


Ilustración 29 - Predicción binaria

Los tres nodos corresponden a la transformación binaria de la última columna.

PREDICT \ ...	0	1
0	13267	40
1	122	2

Correct classified: 13.269 Wrong classified: 162

Accuracy: 98,794 % Error: 1,206 %

Cohen's kappa (κ) 0,02

Ilustración 30 - Confusion Matrix X

Los resultados obtenidos en este caso han sido extraordinarios, como se puede apreciar se ha alcanzado el 98,8% de exactitud. En concreto, este algoritmo funciona muy bien para predecir las muertes ya que de las casi 13.500 instancias evaluadas solo ha cometido dos errores prediciendo muertes (40 aciertos).

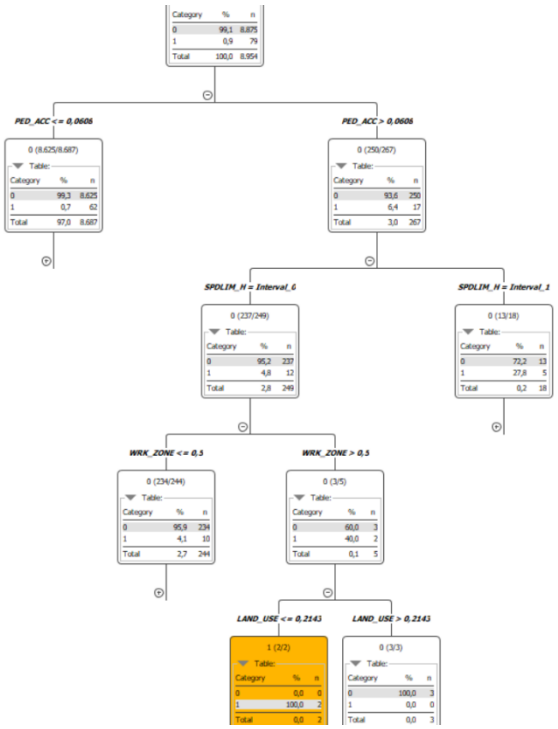


Ilustración 31 - Árbol de decisión final

La imagen de arriba representa el árbol de decisión que indica si el accidente que se ha producido tendrá víctimas mortales (1) o no (0). Es mucho más grande, pero se han ocultados sus ramas.