Prácticas Inteligencia de Negocio

Sergio Quijano Rey - 72103503k
sergioquijano@correo.ugr.es
5º Doble Grado Ingeniería Informática y Matemáticas
Grupo de prácticas 1

5 de noviembre de 2021

Índice

Índice de figuras

1. Introducción

En esta sección hablaremos de cada uno de los problemas abordados, tratando las particularidades de cada caso y algunas consideraciones en base a estas peculiaridades tratadas.

1.1. Información general

En todas las partes en las que necesitemos usar números aleatorios, usaremos la semilla 123456789 para poder reproducir bajo las mismas condiciones los experimentos y para que las comparaciones entre los distintos algoritmos sean lo más justas posible. En los siguientes subapartados, introduciremos los distintos problemas a tratar, sus características y problemas que puedan plantear.

En todos los *datasets* hemos usado la misma estructura jerárquica apoyándonos en los metanodos de `KNIME`. Esta estructura busca una mayor limpieza en el *"código"*. Dicha estructura se va a ir vislumbrando a lo largo de las secciones de estas memorias, donde incluiremos capturas de pantalla de los distintos *workflows* desarrollados.

1.2. *Heart Failure Prediction*

En primer lugar, la fuente original del *dataset* se puede encontrar en [1]. Aunque podemos estar trabajando con un *dataset* ligeramente modificado por los profesores de la asignatura, al igual que con el resto de *datasets* que estudiaremos en esta práctica.

En la propia página de la que se obtiene el dataset [1], se especifica que la tarea a resolver para este *dataset* es *"Create a model to assess the likelihood of a possible heart disease event. This can be used to help hospitals in assessing the severity of patients with cardiovascular diseases"*. Es decir, nuestra tarea es generar un modelo de clasificación para predecir, con los datos de entrada dados, si un paciente puede desarrollar algún problema de tipo cardiaco.

En el siguiente apartado, pasamos a comentar las particularidades de este *dataset*, información que hemos extraído con el análisis exploratorio hecho en `KNIME`:

1.2.1. Análisis Exploratorio de los Datos

Realizamos un pequeño *Análisis Exploratorio Inicial* (*EDA*) de los datos usando las herramientas que nos proporciona `KNIME`.

Como ya comentábamos en *"1.2.* Heart Failure Prediction*"*, buscamos predecir si una persona tendrá problemas de tipo cardiaco. Por tanto, la variable de salida con la que vamos a trabajar es `HeartDisease`. Consideraremos, por su mayor relevancia, como clase positiva, a la clase 1.

Lo primero que vemos es que tenemos 12 variables (11 variables de entrada más la variable de salida) y 918 filas (y por tanto, 918 ejemplos).

El *workflow* de mayor nivel, para este primer *dataset*, se muestra en la siguiente figura:

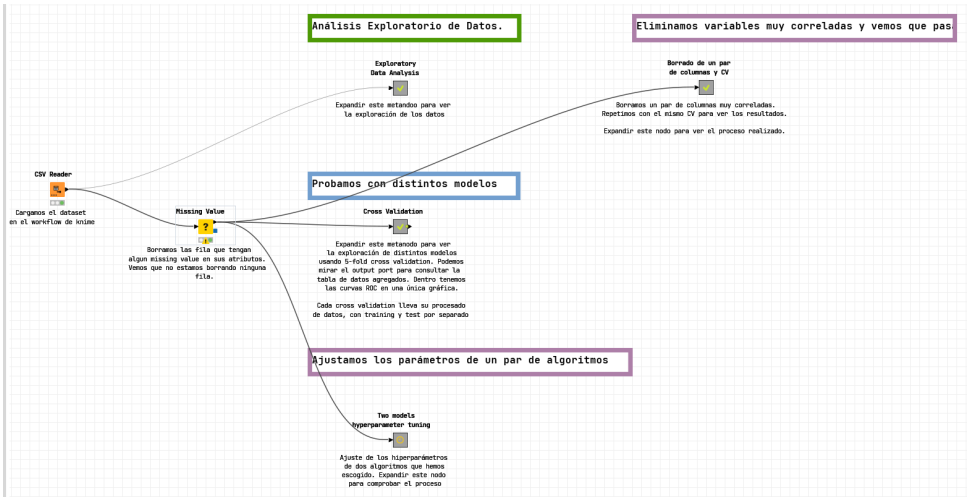

Figura 1: *Workflow* de más alto nivel para el primer *dataset*

La parte que ahora nos interesa es la de Análisis Exploratorio de Datos, que mostramos en la siguiente figura:

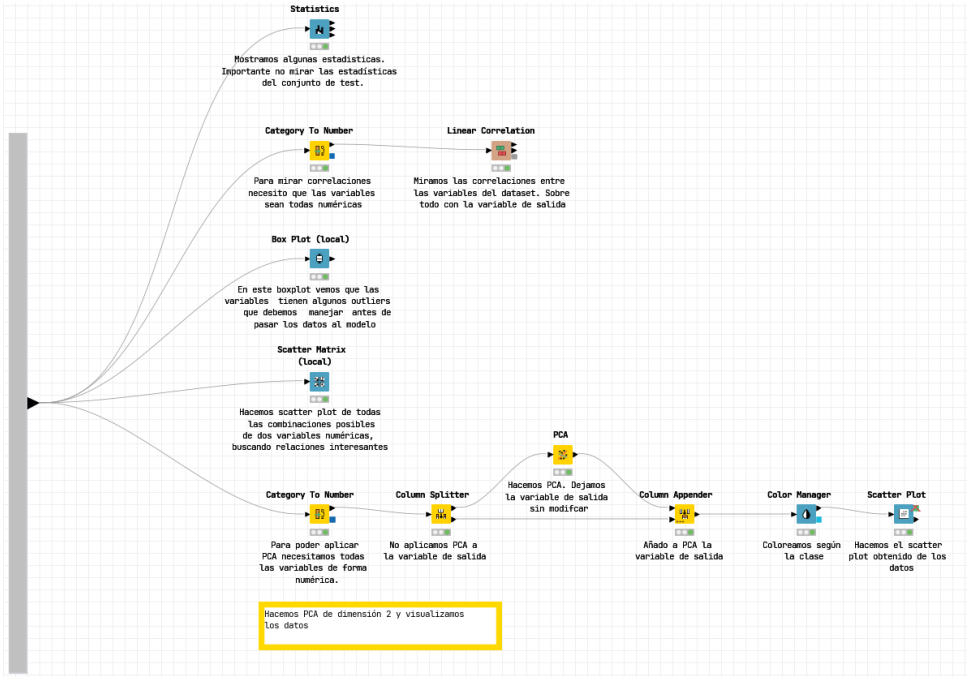

Figura 2: *Workflow* de Análisis Exploratorio de Datos para el primer *dataset*

Empezamos con el nodo de estadísticas, que nos muestra que tenemos 410 ejemplos para la clase de salida 0 y 508 para la clase 1. Por tanto tenemos un ligero desbalanceo (44.66 % para la clase 0 y 55.34 % para la clase 1), pero en este *dataset* no vamos a tratar dicho desbalanceo. En futuros *datasets* nos vamos a encontrar con clases mucho más desbalanceadas.

Dentro del anterior nodo vemos las distribuciones de las otras variables con las que trabajamos, sin llegar a conclusiones de gran relevancia, salvo razonamientos del tipo hay una característica que predomina en un valor sobre el otro valor en la población, o la población tiene una distribución de la variable normal o con cierta asimetría hacia un lado. Ejemplo de característica que predomina se muestra en la siguiente figura:

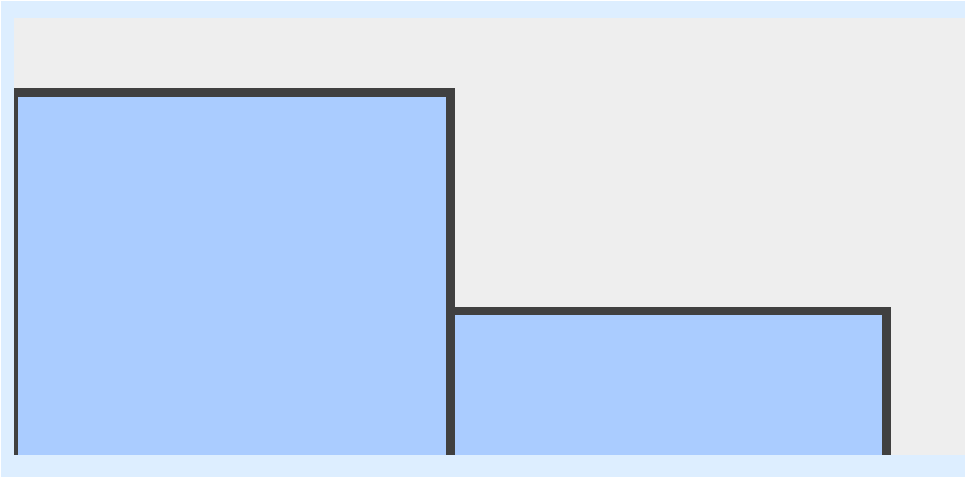

Figura 3: Distribución de la variable sexo. A izquierda, los hombres. A la derecha, las mujeres. Es claro que tenemos muchos más hombres que mujeres en nuestra población.

Esta información puede ser utilizada de forma experta en nuestros sistemas automáticos.

Sin embargo, por una cuestión de tiempo, no realizamos un ajuste tan a fondo de los modelos que vamos a generar (sobre todo teniendo en cuenta que practicar esto para los cuatro *datasets* es inviable. En otros *datasets* realizaremos una selección de variables usando otras técnicas).

Ejemplo de una distribución con cierta asimetría se muestra a continuación:

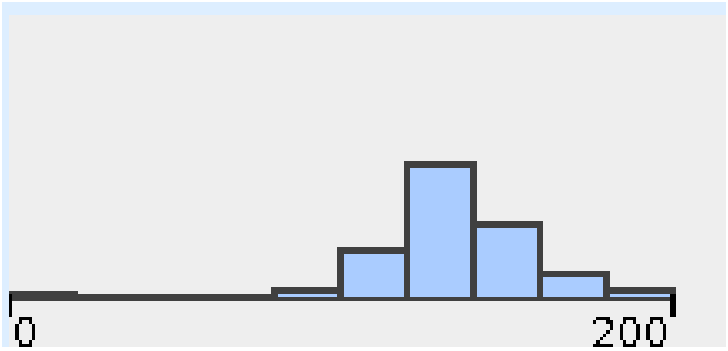

Figura 4: Distribución de la variable `RestingBP`. Podemos ver claramente cómo se acumula más hacia la derecha.

De nuevo, esta información es interesante para plantear los modelos de aprendizaje automático de una forma mucho más concienzuda, pero por la extensión de la práctica en técnicas y aspectos a tener en cuenta, no entramos en detalle en este aspecto.

Lo siguiente que hacemos en el Análisis Exploratorio de Datos es mostrar la matriz de correlaciones lineales, que presentamos a continuación:

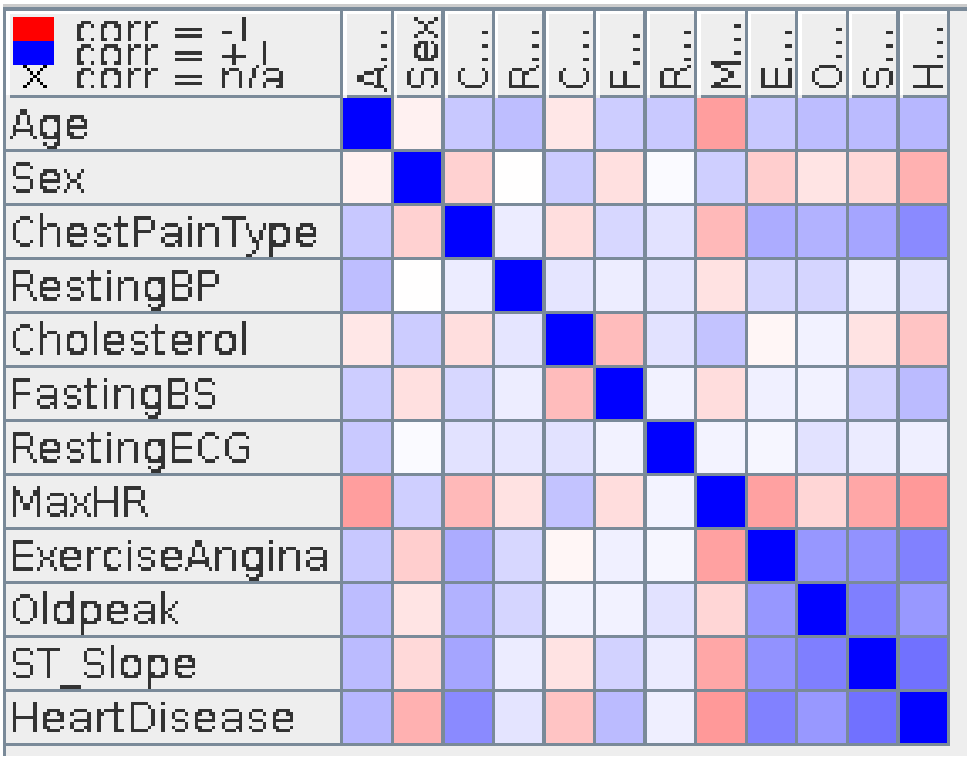

Figura 5: Matriz de correlaciones lineales. Un color azul significa correlación lineal positiva, y un color rojo significa correlación lineal negativa.

Vemos algunas variables correladas. Las más interesantes son el grupo que forman las variables `MaxHR` `ExerciseAngina`, `Oldpeak`, `ST_Slope` y la variable de salida `HeartDisease`. Sabiendo que estas variables están muy correladas, y que una de ellas es la variable de salida, podríamos probar a construir los modelos predictivos en base a este grupo de variables. Por tanto, estamos justificando el interés de que, más adelante, probemos a eliminar ciertas filas empleando esta matriz de correlaciones, y ver cómo afecta esto al rendimiento de nuestros modelos.

Mostramos ahora los *boxplots* de algunas variables, para ver que tenemos ciertos valores *outliers*:

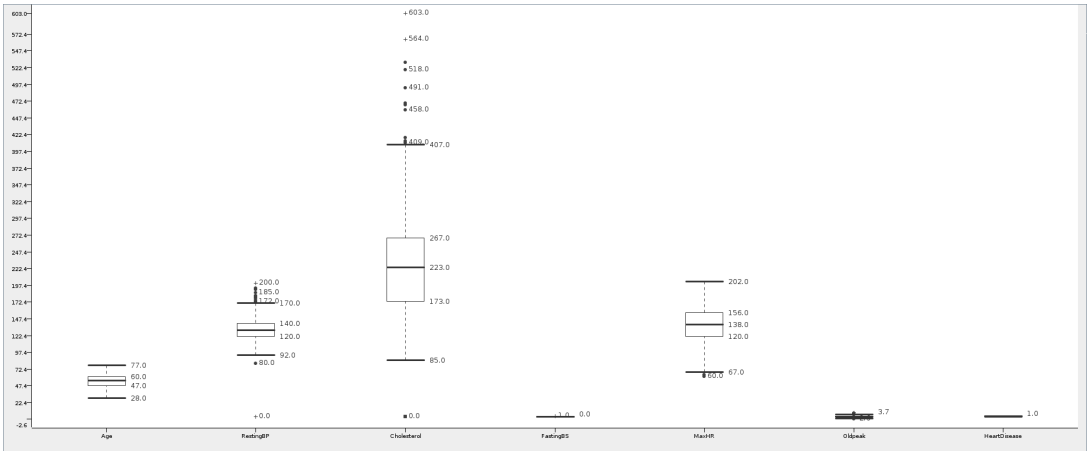

Figura 6: *Boxplots* de algunas de las variables con las que trabajamos. Queda claro que tenemos *outliers* (sabiendo que están alejados de la media más de 3 veces la desviación típica). Por tanto, tendremos que tratar de alguna forma estos *outliers*

También hacemos *plots* de todas las combinaciones posibles entre dos variables, obteniendo la siguiente gráfica:

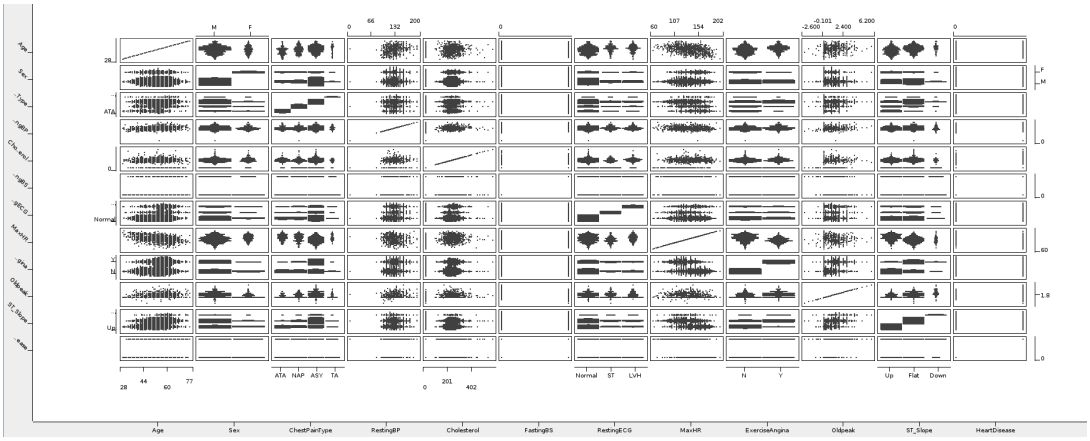

Figura 7: Gráficas de todas las posibles combinaciones de dos variables de nuestro dataset

De nuevo, como ya comentábamos para *"3. Distribución de la variable sexo. A izquierda, los hombres. A la derecha, las mujeres. Es claro que tenemos muchos más hombres que mujeres en nuestra población."*, podemos extraer de esta gráfica algunas conclusiones sobre la distribución de las variables y algunas relaciones que, por simpleza del problema y por falta de tiempo, no vamos a incluir en nuestro modelo.

En último lugar, probamos a aplicar *PCA* al *dataset* para obtener solo dos variables de entrada junto con la variable de salida, buscando sacar alguna información de alto nivel del problema. Esta técnica busca transformar el conjunto de datos con nuevas variables, de modo que estas variables no estén correlacionadas entre sí manteniendo el máximo posible de la varianza del conjunto de datos original. Este procedimiento se apoya en la descomposición en valores propios de la matriz de covarianzas [2]. Mostramos las dos variables obtenidas usando *PCA*, coloreando cada punto del plano $2D$ según a la clase a la que pertenecen:

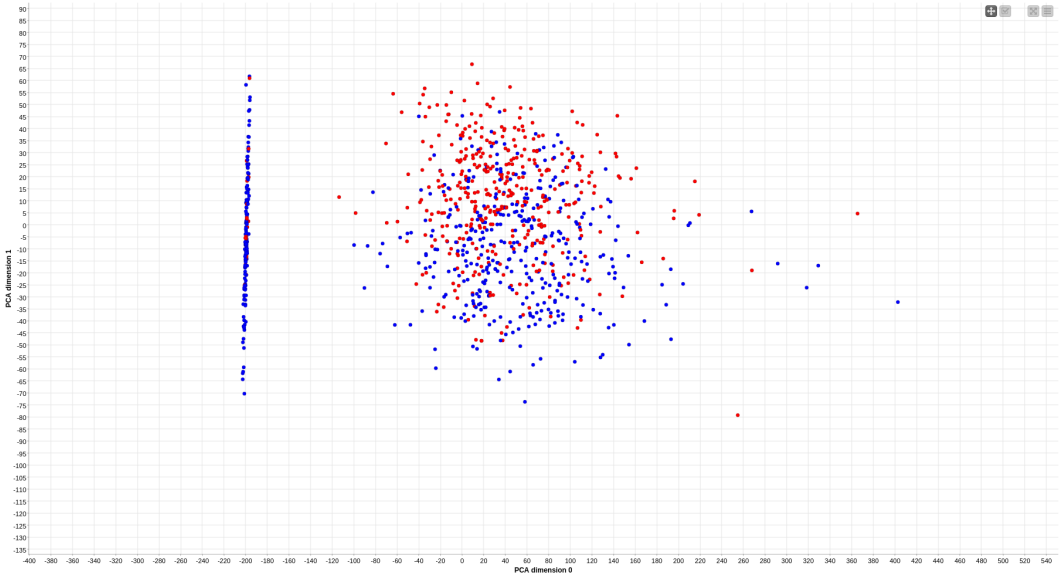

Figura 8: Resultado de aplicar *PCA* en dos dimensiones, coloreando cada punto según la clase a la que pertenecen

Los resultados obtenidos no nos invitan a usar *PCA* para usarlo como base a modelos de aprendizaje automático. Sin embargo es interesante que se ha obtenido una línea a la izquierda con clara predominancia de la clase representada por el color azul, mientras que a la derecha se obtiene un *clúster* ligeramente circular donde los datos de las dos clases están mezclados sin un patrón claro (quizás la clase roja tiene mayor predominancia en la parte superior). Estamos seguros de que aplicando *PCA* con dimensiones mayores obtendríamos un buen *performance* en los modelos de aprendizaje automático. Pero como para otros *datasets* es más interesante aplicar esta técnica, lo dejamos para más adelante.

En último lugar, tratamos los ***missing values***. En el *workflow* de más alto nivel, hacemos una primera aproximación al tratamiento de los *missing values*. Tratamos de borrar todas las filas que contengan *missing values*, pero no borramos ninguna fila. En un principio pensamos que no tenemos *missing values* en este *dataset*. Sin embargo, explorando la colección de datos comprobamos el siguiente hecho: la variable `Cholesterol` contiene *missing values* codificados con el valor 0. Para tratar estos *missing values*, en *Cross Validation* los marcaremos como tal, y los trataremos convenientemente en cada nodo de validación (para evitar hacer *data snooping*). Entraremos en detalles más adelante.

El nodo que usamos para realizar esta comprobación sobre los *missing values* se muestra en el *workflow* general que aparece en *"1.* Workflow *de más alto nivel para el primer* dataset*"*.

2. Resultados Obtenidos

2.1. Consideraciones iniciales

Para esta práctica, hemos considerado 7 modelos de aprendizaje automático distintos, que puedan ser interesantes para los cuatro problemas a los que nos vamos a enfrentar. Algunos de estos modelos pueden tener muy poco sentido para algún problema concreto. Sin embargo, elegimos los mismo 7 modelos para los 4 datasets, porque esto nos permitirá realizar una comparación entre ellos en distintos ambientes. De hecho, el que algunos modelos no tengan sentido en ciertas situaciones será una buena conclusión a extraer en el análisis posterior.

Los modelos considerados son:

1. Árbol de decisión construido con *AdaBoost*
2. Red Neuronal Simple
3. *Support Vector Machine*
4. *Support Vector Machine* con normalización
5. *K-NN*
6. *Naive Bayes*
7. *Random Forest*

Pensamos que con estos modelos tenemos una variedad suficiente para comparar en situaciones diferentes. Además, todos los los modelos tienen sentido para al menos uno de los *datasets* con los que vamos a trabajar.

En este proceso, usaremos un nodo al que llamamos *Custom Cross Validation*. Este nodo tiene la misma base que el nodo de `KNIME` *Cross Validation*. Sin embargo, añadimos dos nodos, uno para preprocesar el *fold* de entrenamiento y otro para aplicar dicho preprocesado al *test*, sin hacer *data snooping*. Es decir, las operaciones se calculan y aplican con *training*, y se aplican en *test* **usando el cálculo realizado sobre *training***.

2.2. Heart Failure Prediction

En primer lugar, tenemos un *metanodo* para encapsular todo el trabajo con los 7 algoritmos que hemos considerado. Este *metanodo* se mostraba en el *workflow* más general en *"1. Workflow de más alto nivel para el primer* dataset*"*. Dentro de este nodo para *Cross Validation* tenemos la siguiente estructura:

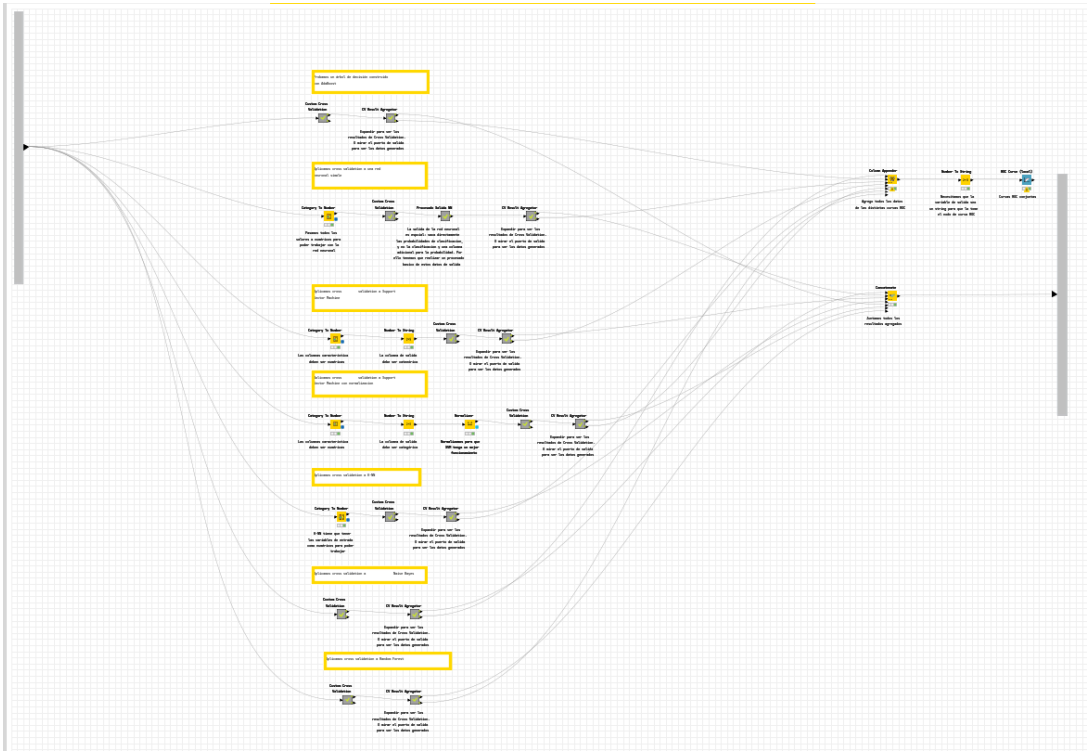

Figura 9: *Workflow* en el que realizamos todos los *Cross Validation* de los algoritmos considerados

Mostramos una captura con mayor *zoom* para que se vea claramente la estructura generada, dejando sin mostrar alguna de las filas correspondientes a algunos de los 7 modelos estudiados:

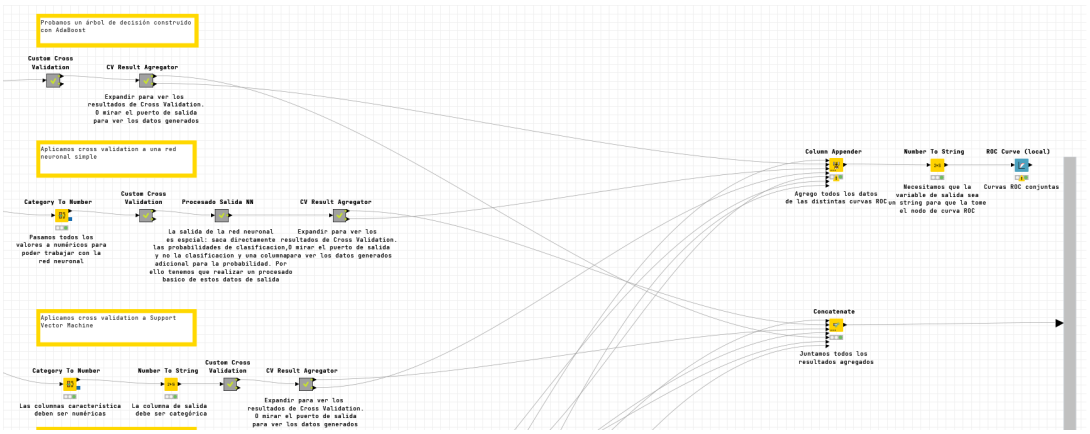

Figura 10: *Workflow* en el que realizamos todos los *Cross Validation* de los algoritmos considerados, con algo de *zoom* para que se aprecie mejor la estructura desarrollada

Mostramos el contenido del nodo *Custom Cross Validation* para *AdaBoost*. Este nodo *Custom Cross Validation* tiene ciertas diferencias como ya hemos mencionado en *"2.1. Consideraciones iniciales"*:

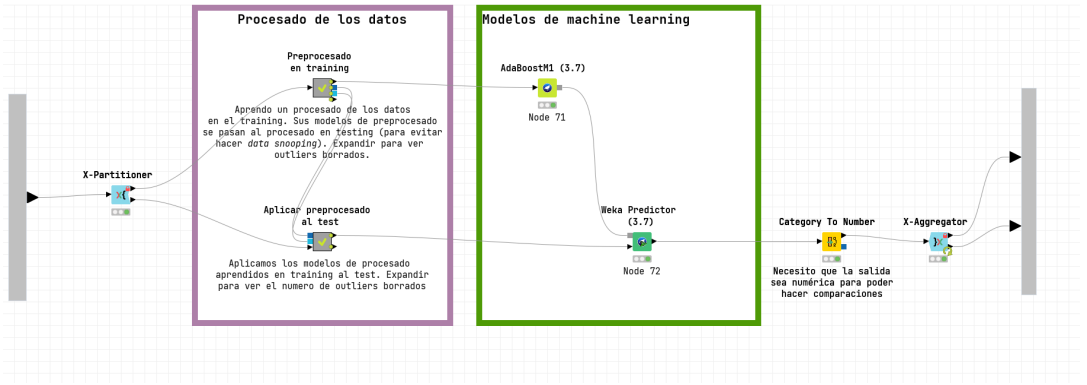

Figura 11: Nodo de *Cross Validation* para *AdaBoost*

Mostramos ahora el nodo para preprocesamiento en *training*:

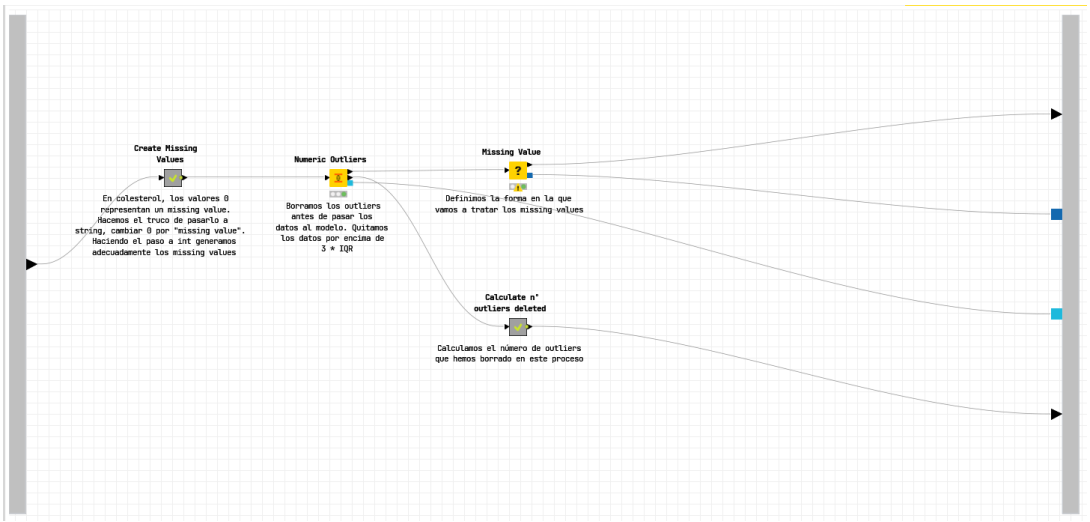

Figura 12: Preprocesamiento de los datos del *fold* de *training*

Mostramos ahora el nodo para preprocesamiento en *test*:

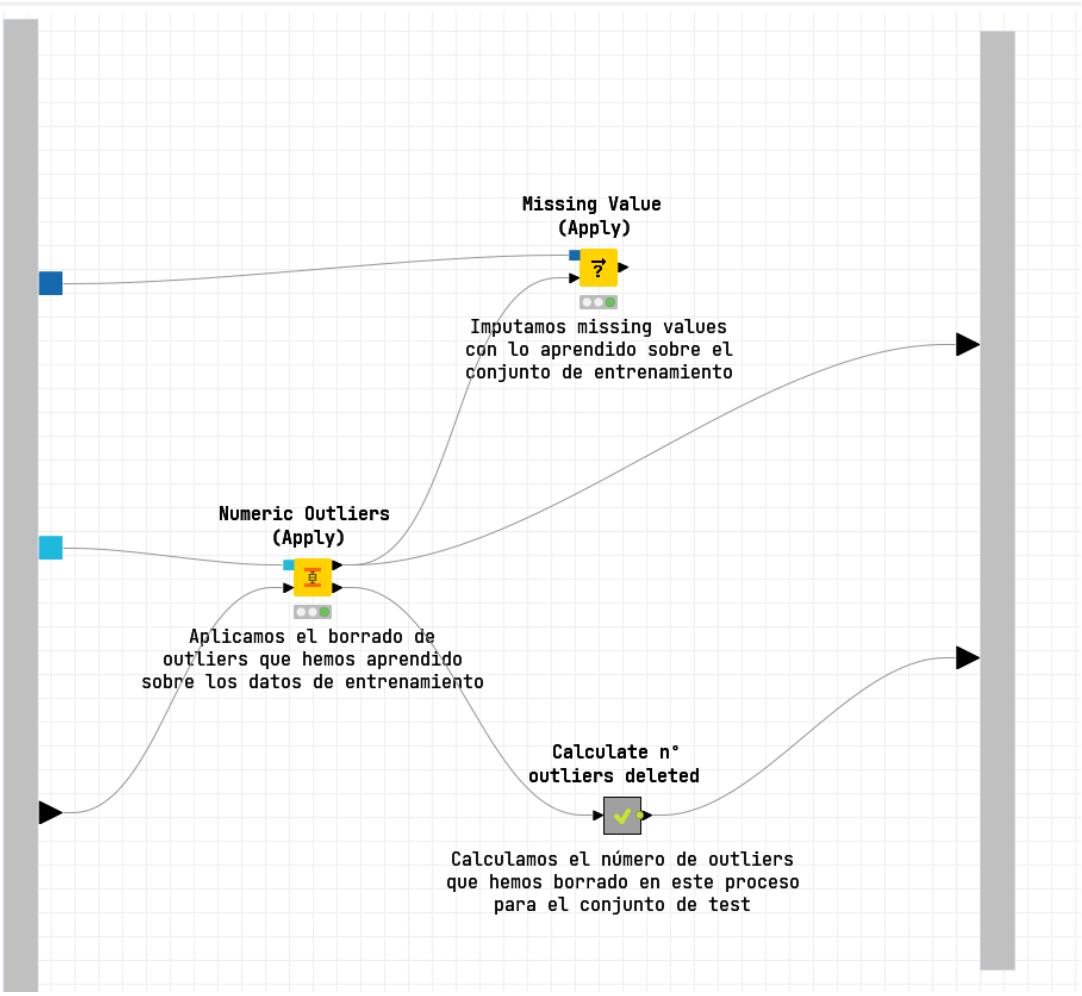

Figura 13: Preprocesamiento de los datos del *fold* de *test*

En *"13. Preprocesamiento de los datos del* fold *de* test*"* queda claro lo que ya comentábamos, preprocesamos usando los parámetros de procesamiento calculados sobre el *fold* de *training*. Esto es fundamental para **evitar hacer *data snooping***.

Además, de ahora en adelante, salvo que se diga lo contrario, estos dos nodos de preprocesamiento serán exactamente los mismos en todos los nodos de *Custom Cross Validation*, en todos los *dataset*, salvo que se indique lo contrario.

Como se muestra en *"10.* Workflow *en el que realizamos todos los* Cross Validation *de los algoritmos considerados, con algo de* zoom *para que se aprecie mejor la estructura desarrollada"*, estamos pasando la salida de *Custom Cross Validation* a un nodo de procesamiento, que mostramos a continuación:

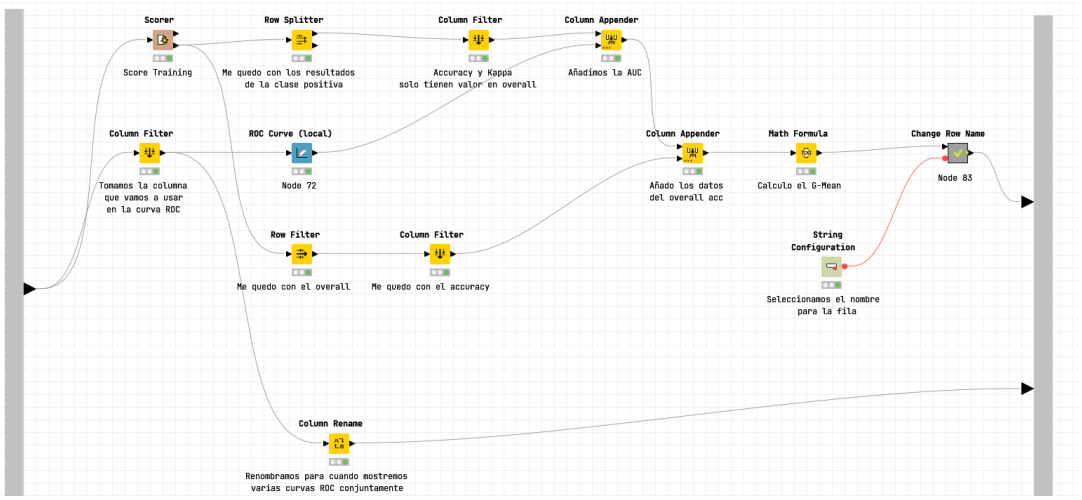

Figura 14: Nodo de procesamiento de la salida de *Custom Cross Validation*

En este nodo, tomamos la salida del *scorer* y la procesamos, tomando las métricas para la clase positiva y añadiendo el área debajo de la curva *ROC*. Además, calculamos la fórmula de la métrica *G-Mean*. También devolvemos las probabilidades en bruto para poder calcular las curvas *ROC* globales.

Se muestra también que estamos usando una `flow variable` para establecer el nombre del modelo cuyos resultados estamos procesando. Esto es útil para que en las tablas globales, cada fila esté etiquetada por el nombre del modelo que estamos tratando (esto es lo que consigue el nodo *Change Row Name*).

De nuevo, salvo que se diga lo contrario, todos los nodos de este tipo son idénticos.

Mostramos ahora el *workflow* para redes neuronales simples. Realizamos un preprocesado general básico antes de usar nuestro nodo *Custom Cross Validation* que mostramos en la siguiente figura:

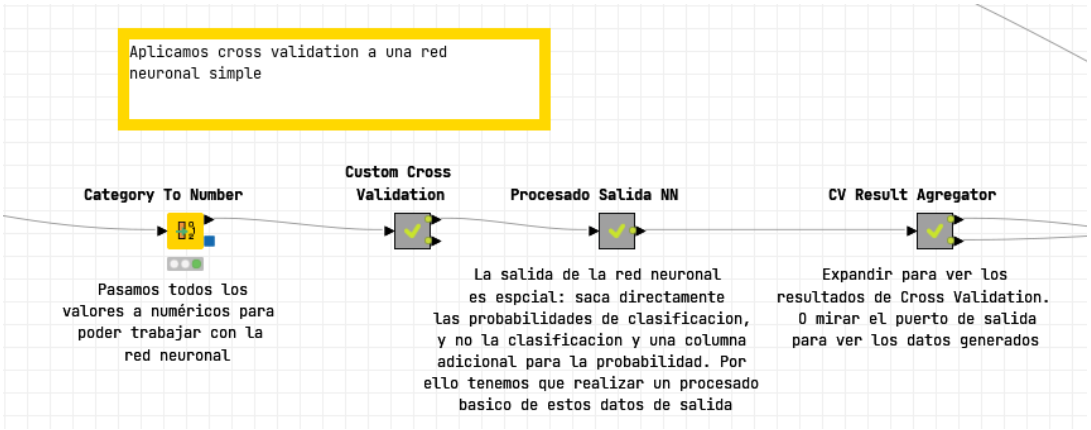

Figura 15: Preprocesado general previo a probar Redes Neuronales simples

Con esto queda claro que lo único que estamos realizando es pasar todas las variables categóricas a numéricas, pues esto es un requerimiento de las redes neuronales para que puedan funcionar. Mostramos el nodo de *Custom Cross Validation* para este caso:

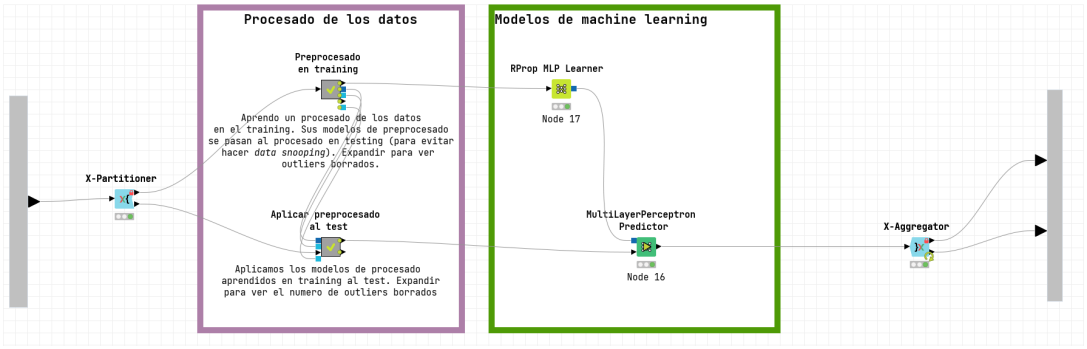

Figura 16: Nodo de *Cross Validation* para *Neural Net*

En este caso no estamos normalizando las variables, lo cual puede generar resultados no óptimos (pues las redes neuronales se benefician mucho de que las variables de entrada estén normalizadas. De hecho en *deep learning* se usa *batch normalization* para normalizar las salidas entre capas ocultas). Por simpleza del modelo no hacemos esta normalización para el *dataset*, pero en siguientes *datasets* sí que haremos la normalización.

La salida de las redes neuronales suponen un caso concreto que tratar. La red neuronal no nos da la clasificación directamente, sino un valor $x \in [0, 1]$. Tenemos que tratar la salida a mano. Para ello duplicamos la salida de este valor x que representa la probabilidad de clasificación que usaremos para las curvas *ROC*. En una de las dos ramas, colapsamos ese valor al conjunto $\{0, 1\}$ redondeando, y consideramos eso la salida de clasificación. Mostramos ese proceso, que se hace en el nodo de *Procesado Salida NN*, en la siguiente figura:

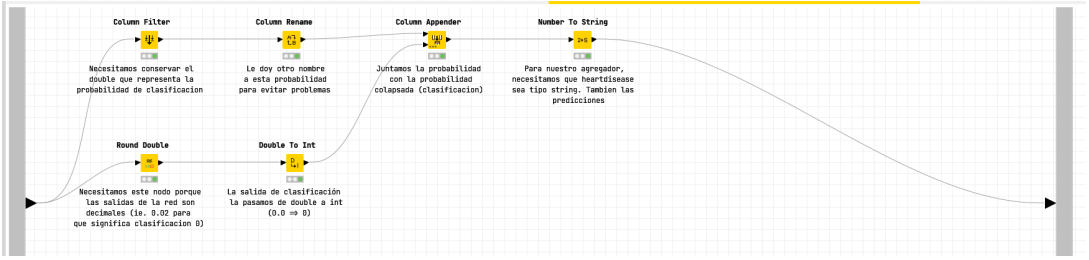

Figura 17: Procesado de la salida de *Neural Net*

Con esto, usamos el nodo que procesa la salida de *Cross Validation* que ya hemos mostrado previamente, y que no volvemos a mostrar para evitar ser repetitivos.

Mostramos ahora el preprocesado general para *Support Vector Machine*. Consiste en pasar a variables numéricas todas las variables menos la de salida, que debe ser de tipo categórica:

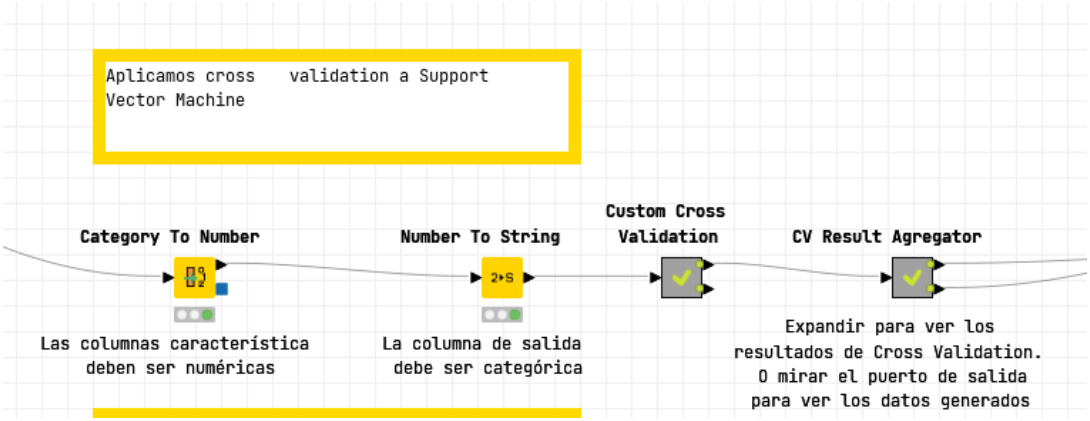

Figura 18: Preprocesado general previo a probar *Support Vector Machine*

Mostramos ahora el nodo *Custom Cross Validation* para este modelo:

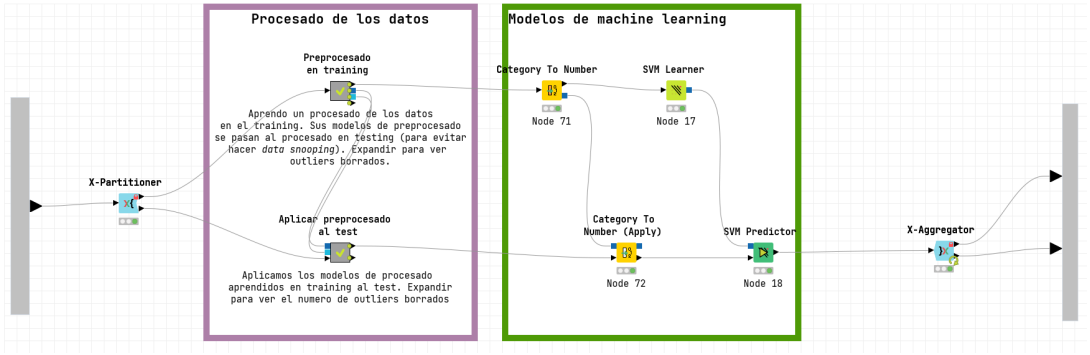

Figura 19: Nodo de *Cross Validation* para *Support Vector Machine*

En este caso, nos aseguramos de nuevo que estamos usando variables numéricas para las variables de entrada. En este *dataset*, usamos los nodos de KNIME para *SVM*. En otros *datasets* donde la eficiencia sea clave, pasaremos a usar los nodos de *Weka* que aparentemente funcionan de forma más rápida.

En el caso de *Supoort Vector Machine* con normalización, lo único que cambiamos es el preprocesado general que aplicamos antes de *Custom Cross Validation*. Mostramos esto a continuación:

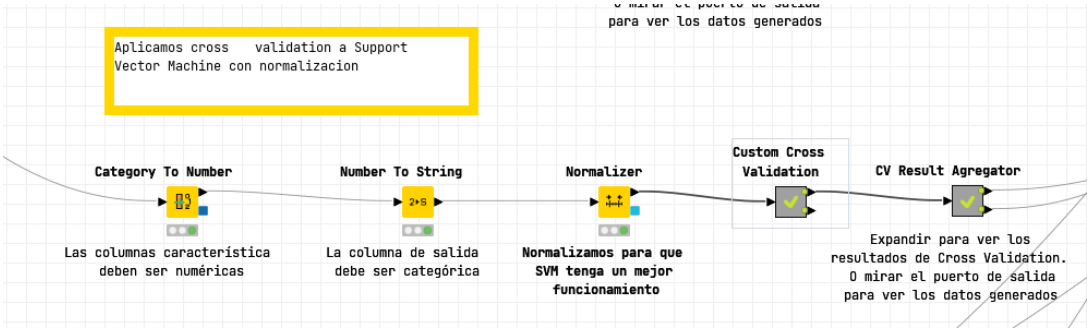

Figura 20: Preprocesado general previo a probar *Support Vector Machine* con normalización. Esto es lo único que cambia respecto al modelo anterior: *SVM* sin normalización

Mostramos ahora el preprocesado general para *K-NN*. *K-NN* tiene que trabajar con todas las variables de entrada en formato numérico, y esto es lo que hacemos:

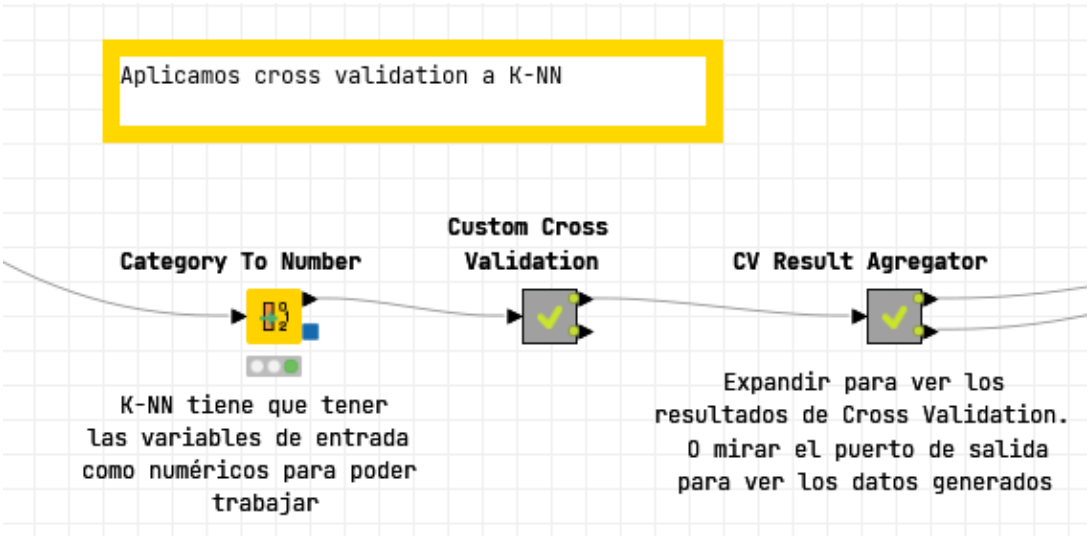

Figura 21: Preprocesado general previo a probar *K-NN*

Dentro del nodo de *Custom Cross Validation* lo que hacemos es, de nuevo asegurar que tenemos las variables en el formato correcto, y formatear la salida de *K-NN* para que el nodo *X-Aggregator* pueda funcionar correctamente. Mostramos esto a continuación:

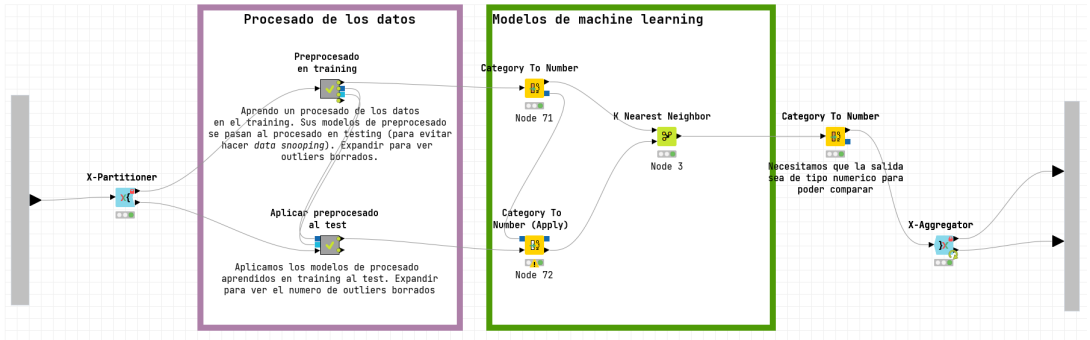

Figura 22: Nodo de *Cross Validation* para *K-NN*

3. Referencias

[1] "Heart failure prediction dataset — kaggle." https://www.kaggle.com/fedesoriano/heart-failure-prediction. (Accessed on 11/05/2021).

[2] "A step-by-step explanation of principal component analysis (pca) — built in." https://builtin.com/data-science/step-step-explanation-principal-component-analysis. (Accessed on 11/05/2021).