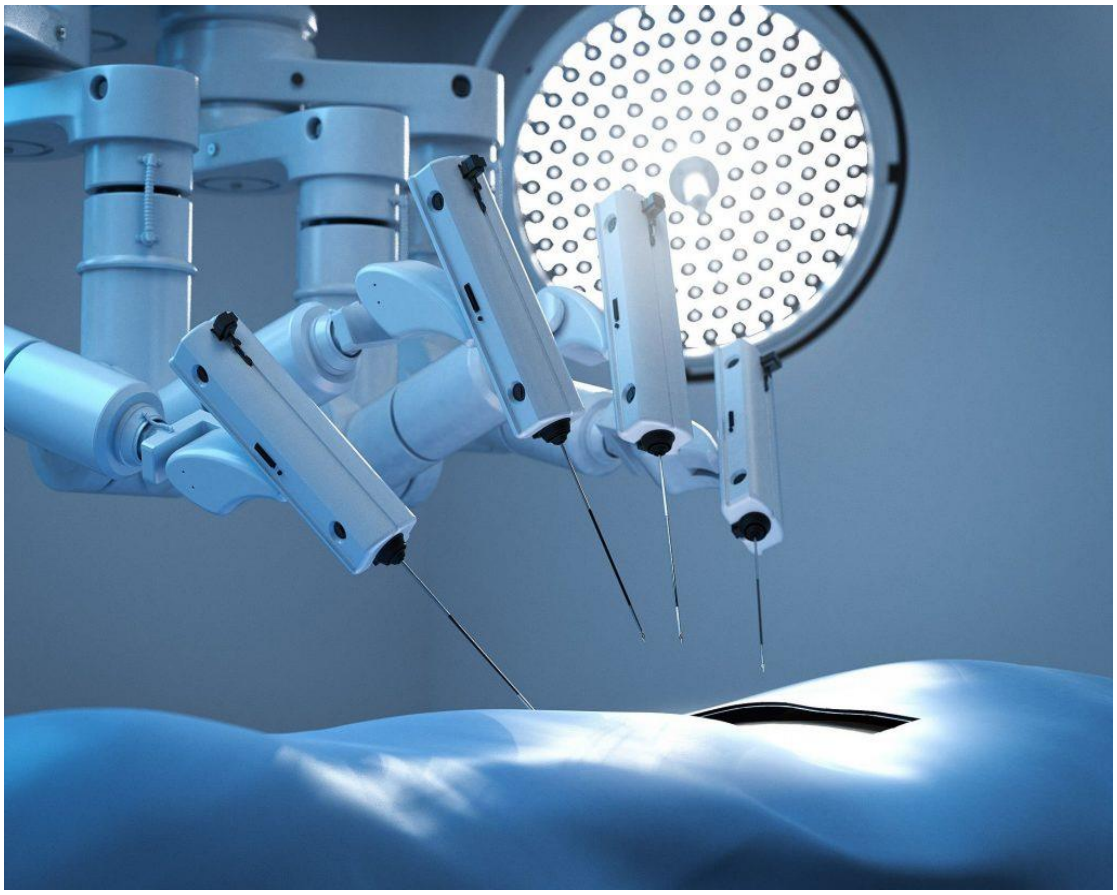
Universidade do Minho



Machine Learning

NO DIAGNÓSTICO DE DOENÇAS CARDÍACAS

Engenharia Biomédica | 2022/2023 | Inteligência Artificial

Adriana Costa, A96737; Maria Vale, A97022; Patrícia Costa, A97308; Sofia Correia, A96498

6 de janeiro de 2023

Resumo

Este trabalho foi realizado no âmbito da UC de Inteligência Artificial em Engenharia Biomédica e proposto pelos docentes César Analide e Inês Amorim.

A sua realização pretende, assim, aplicar as competências adquiridas até ao momento de aprendizagem automática para análise de dados e construção de modelos suportados por técnicas de aprendizagem (*Machine Learning*).

Deste modo, fez-se uso da plataforma KNIME para se desenvolver um sistema de aprendizagem automática, através da construção de *workflows*.

É proposto o desenvolvimento um sistema de aprendizagem automática para estimar o atributo *target* de um estudo sobre doenças cardíacas. Os dados recolhidos correspondem a quatro locais onde foram realizados exames a parâmetros que influenciam a existência de doenças cardíacas. Com isto, pretende-se fazer um diagnóstico sobre a presença ou não destas doenças. A concretização deste trabalho implica a análise e tratamento de grandes quantidades de dados para que as previsões sejam mais confiáveis.

Índice

Introdução

Machine Learning é um campo de estudo que dá a uma máquina a capacidade de aprender sem ser explicitamente programada. Permite que os computadores encontrem padrões nos dados e os usem para tomar decisões. Como tal, deve-se fazer uso de algoritmos de *Machine Learning* que vão aprender sobre os dados através de sistemas de aprendizagem para resolver um problema.

A aprendizagem automática engloba outras três aprendizagens: aprendizagem por reforço, aprendizagem com supervisão e aprendizagem sem supervisão. A primeira, é um paradigma de aprendizagem onde apesar de não se ter informações sobre os resultados pretendidos, permite efetuar uma avaliação sobre se os resultados produzidos são bons ou maus. Na aprendizagem com supervisão, também ela um paradigma de aprendizagem, os casos que se usam para aprender contêm informações sobre os resultados pretendidos, sendo possível estabelecer uma relação entre os valores pretendidos e os valores produzidos pelo sistema. Por último, a aprendizagem sem supervisão, é um paradigma de aprendizagem em que não são conhecidos resultados sobre os casos, apenas os enunciados dos problemas, tornando necessário a escolha de técnicas de aprendizagem que avaliem o funcionamento interno do sistema.

A grande maioria dos algoritmos Machine Learning usa aprendizagem com supervisão. Tal acontece porque esta aprendizagem permite que os dados de entrada e os resultados tornem possível que o algoritmo aprenda uma função de mapeamento dos dados nos resultados. Os dados têm conhecimento associado, sendo uns mais suscetíveis a ter mais informação que outros. Desta forma, um algoritmo de Machine Learning encontra esses padrões nos dados e gera um modelo. Um modelo representa o que foi aprendido pelo algoritmo, tornando-o capaz de reconhecer padrões mesmo quando lhe são apresentados novos dados.

No entanto, existem etapas que devem ser realizadas para se obter com sucesso o modelo. Primeiramente, tem-se de definir o problema a que se quer dar resposta, de modo a se proceder à identificação e recolha de dados com que se vai trabalhar. Seguidamente, deve haver uma preparação dos dados, uma vez que estes se encontram brutos e raramente se encontram na forma correta para serem processados. Concluída esta etapa, divide-se os dados em subconjuntos de treino, teste e validação, de modo a generalizarem para dados novos. O dataset de treino é usado para treinar o nosso modelo. Por sua vez, o dataset de validação é utilizado para comparar diferentes modelos e hiperparâmetros. Já o dataset de teste é para testar o modelo, fazendo uso de dados que o modelo não viu. Os subconjuntos de teste e validação permitem avaliar o desempenho de um modelo através da *accuracy* da previsão. Este processo é iterativo e termina quando se tiver um modelo suficientemente bom para responder à nossa pergunta inicial.

Existem muitos tipos diferentes de modelos de aprendizagem, cada um adequado para diversas situações. Alguns dos modelos mais comuns incluem:

- Regressão linear: o modelo de regressão linear é usado para prever uma variável contínua com base em uma ou mais variáveis explicativas;
- Árvores de decisão: o modelo é usado para fazer previsões de classificação com base em uma série de perguntas lógicas sobre os dados;
- Redes neuronais: o modelo é composto por várias camadas de neurónios interconectados que trabalham juntos para fazer previsões. As redes neuronais são capazes de lidar com conjuntos de dados complexos e não linearmente separáveis;
- Random Forest: o modelo Random Forest é um modelo de ensemble que combina várias árvores de decisão para criar um modelo mais robusto. Ele é amplamente utilizado em tarefas de classificação e regressão.

Após a criação de um modelo usando uma técnica de aprendizagem (Machine Learning) tem-se de avaliar o seu desempenho. Esta medição do desempenho é feita com dados não apresentados durante o treino. Assim, faz-se uso de métricas de qualidade para avaliar o desempenho dos modelos, nomeadamente de matrizes de confusão. A partir destas tabelas, é possível obter o valor da *accuracy*. Esta métrica faz a divisão da quantidade de previsões corretas divididas pela quantidade total de observações, pelo que quanto maior for o resultado desse cociente, melhor o desempenho do modelo.

Para além da *accuracy*, existem outras métricas de avaliação tais como:

- Precisão: proporção de previsões corretas positivas em relação ao total de previsões positivas;
- Especificidade: proporção de previsões negativas corretas em relação ao total de casos negativos;
- F1 Score: média da precisão e sensibilidade;
- Curva ROC: visualização gráfica da sensibilidade e especificidade ao longo de diferentes limiares de classificação. Reduz o patamar de classificação (*threshold*), aumentando os falsos positivos e os verdadeiros positivos;
- Curva AUC: mede quão bem as previsões são classificadas, em vez de avaliar os seus valores absolutos;
- Erro Médio Absoluto: magnitude média de erros num conjunto de previsões;
- Erro Médio Quadrado: cálculo da média das diferenças, ao quadrado, entre os erros num conjunto de previsões.

Preliminares

Em *KNIME* os nodos são elementos básicos de um *workflow*, que é um conjunto de etapas que são executadas para realizar uma tarefa específica. Cada nodo representa uma operação ou uma atividade que é executada, como ler dados de um arquivo, limpar os dados, treinar um modelo de aprendizagem ou fazer previsões.

Para a concretização deste trabalho foi necessário recorrer à utilização de diferentes funções do KNIME (nodos), sendo estas:

***CSV Reader*:**

Lê ficheiros CSV (ficheiro sem formatação em que os valores estão separados por vírgulas, delimitados por aspas e cada linha tem uma *feature* diferente)

***Number to String*:**

Converte números de uma coluna (ou um conjunto de colunas) em strings

***Missing Value Column Filter*:**

Remove todas as colunas da tabela de entrada que contém mais *missing values* que uma determinada percentagem. A filtragem é apenas aplicada às colunas na lista de entrada do painel de filtro de coluna

***Missing Value*:**

Trata os *missing values* encontrados nas células da tabela de entrada

***Partitioning*:**

A tabela de entrada é dividida em duas partições (treino e teste). As duas partições estão disponíveis nas duas portas de saída

***Normalizer*:**

Normaliza os valores de todas as colunas (numéricas), permitindo escolher, na caixa de diálogo, as colunas que se quer trabalhar

***Normalizer (Apply)*:**

Normaliza os dados de entrada de acordo com os parâmetros de normalização fornecidos na entrada do modelo (normalmente provenientes do nodo *Normalizer*). Ele aplicará uma transformação afim a todas as colunas nos dados de entrada contidos na entrada do modelo

***Scorer (JavaScript)*:**

Compara, pelos seus pares de valores atribuídos, duas colunas e mostra a matriz de confusão, ou seja, quantas linhas de um determinado atributo e a classificação correspondente

***Scorer*:**

Assim como o nodo anterior, o *Scorer* compara, pelos seus pares de valores atribuídos, duas colunas e mostra a matriz de confusão.

***RProp MLP Learner*:**

Implementa o algoritmo RProp para redes *feedfoward* multicamadas. RProp realiza uma adaptação local das atualizações de peso de acordo com o comportamento da função de erro

***MultiLayerPerceptron Predictor*:**

Baseado num modelo *MultiLayerPerceptron* treinado fornecido no modelo de entrada desse nodo, os valores de saída espetados são calculados. Se a variável de saída for nominal, a saída de cada neurónio e a classe do neurónio vencedor são produzidas. Caso contrário, o valor da regressão é calculado. É necessário filtrar os *missing values* antes deste nodo ser usado

***Decision Tree Learner*:**

Induz uma árvore de decisão de classificação na memória principal. O atributo de destino deve ser nominal. Os outros atributos usados para tomar a decisão podem ser nominais ou numéricos

***Decision Tree Predictor*:**

Utiliza uma árvore de decisão existente (transmitida pela porta modelo) para prever o valor da classe para novos padrões;

***Numeric Outliers*:**

Deteta e trata os *outliers* de cada uma das colunas selecionadas individualmente pelo intervado interquartil (IQR). Para detetar os *outliers* para uma determinada coluna, o primeiro e o terceiro quartil (Q_1 e Q_3) são calculados. Uma observação é sinalizada como atípica se estivar fora do intervalo $R = [Q_1 - k(IQR), Q_3 + k(IQR)]$ com $IQR = Q_3 - Q_1$ e $k \geq 0$. Definindo $k = 1.5$, o menor valor em R corresponde, normalmente, à extremidade inferior do bigode de um *boxplot* e o maior valor à sua extremidade superior. A observação sinalizada como atípica pode ser substituída por um outro valor ou removida/mantida a linha correspondente.

***Rule Engine*:**

Obtém uma lista de regras definidas pelo programador e tenta combiná-las com cada linha da tabela de entrada. Se uma regra corresponder, o valor resultante será adicionado a uma nova coluna. A primeira regra correspondente na ordem de definição determina o resultado. Cada regra é representada por uma linha.

***Concatenate*:**

Concatena duas tabelas com nomes iguais. Se uma tabela de entrada contém nomes de colunas que a outra não contém, as colunas podem ser preenchidas com *missing values* ou filtradas, ou seja, elas não estarão na tabela de saída.

***Column Rename*:**

Renomeia os nomes das colunas (editando o campo de texto) ou altera os seus tipos (escolhendo um dos tipos possíveis na caixa de combinação)

Descrição do Trabalho e Análise de Datasets

O trabalho tem por base um caso real, de estudos realizados em quatro localidades sobre doenças cardíacas. Deste modo, teve-se acesso a ficheiros, *datasets*, que contêm os dados do diagnóstico da existência ou não de doenças cardíacas nas pessoas a que se sujeitaram ao teste.

Antes de construirmos *workflows* na plataforma *KNIME*, para se desenvolver um sistema de aprendizagem automática capaz de prever um diagnóstico, analisamos os datasets fornecidos e os seus parâmetros de estudo.

PARÂMETROS DE ESTUDO

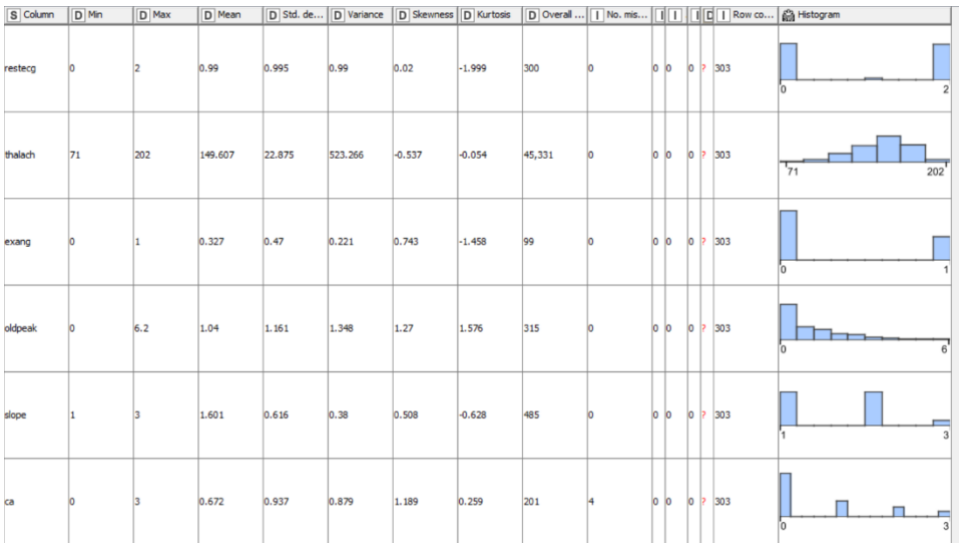
Os seguintes parâmetros estão presentes e foram avaliados nos 4 datasets fornecidos.

age: idade em anos
sex: sexo (1 = masculino; 0 = feminino)
cp: tipo de dor no peito
-- Valor 1: angina típica
-- Valor 2: angina atípica
-- Valor 3: dor não anginosa
-- Valor 4: assintomático
tretbps: pressão arterial em repouso (em mm Hg na admissão ao hospital)
col: colesterol sérico em mg/dl
fbs: (açúcar no sangue em jejum > 120 mg/dl) (1 = verdadeiro; 0 = falso)
restecg: resultados eletrocardiográficos em repouso
--Valor 0: normal
--Valor 1: tendo anormalidade da onda ST-T (inversões da onda T e/ou elevação ou depressão ST > 0,05 mV)
-- Valor 2: mostrando hipertrofia ventricular esquerda provável ou definitiva pelos critérios de Estes
talach: frequência cardíaca máxima alcançada
exang: angina induzida por exercício (1 = sim; 0 = não)
oldpeak = depressão do ST induzida pelo exercício em relação ao repouso
slope: a inclinação do pico do segmento ST do exercício
-- Valor 1: ascendente
-- Valor 2: plano
-- Valor 3: descendente
ca: número de vasos principais (0-3) coloridos por fluoroscopia
tal: 3 = normal; 6 = defeito corrigido; 7 = defeito reversível
num: diagnóstico de doença cardíaca (estado angiográfico da doença)
-- Valor 0: < 50% de estreitamento do diâmetro
-- Valor 1: > 50% de estreitamento do diâmetro

Figura 1. Parâmetros avaliados.

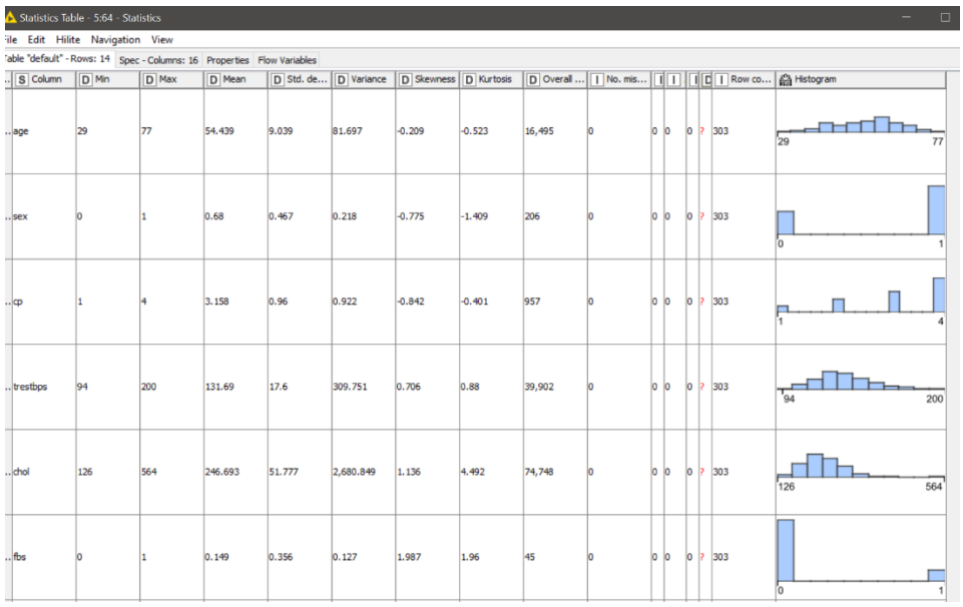
ANÁLISE

- **Cleveland.csv**
  O ficheiro "Cleveland.csv" apresenta 303 linhas de valores de 14 parâmetros, o diagnóstico feito na região apresenta um grupo de pessoas com idade média de 54 anos, onde 97 registos são de identidades femininas e 206 masculinas. A partir do nodo *statistics* ligado ao nodo *file reader* conseguimos conferir outros dados estatísticos. Neste *dataset* ainda verificamos que os parâmetros *ca* e *thal* apresentam alguns *missing values*.



Column	Min	Max	Mean	Std. de...	Variance	Skewness	Kurtosis	Overall ...	No. mis...					Row co...	Histogram
restecg	0	2	0.99	0.995	0.99	0.02	-1.999	300	0	0	0	0	?	303	0 2
thalach	71	202	149.607	22.875	523.266	-0.537	-0.054	45,331	0	0	0	0	?	303	71 202
exang	0	1	0.327	0.47	0.221	0.743	-1.458	99	0	0	0	0	?	303	0 1
oldpeak	0	6.2	1.04	1.161	1.348	1.27	1.576	315	0	0	0	0	?	303	0 6
slope	1	3	1.601	0.616	0.38	0.508	-0.628	485	0	0	0	0	?	303	1 3
ca	0	3	0.672	0.937	0.879	1.189	0.259	201	4	0	0	0	?	303	0 3

Figura 2. *Statistics* do ficheiro *Cleveland.csv* parte 1.



Statistics Table - 5:64 - Statistics

File Edit Hilite Navigation View

Table "default" - Rows: 14 Spec - Columns: 16 Properties Flow Variables

Column	Min	Max	Mean	Std. de...	Variance	Skewness	Kurtosis	Overall ...	No. mis...					Row co...	Histogram
age	29	77	54.439	9.039	81.697	-0.209	-0.523	16,495	0	0	0	0	?	303	29 77
sex	0	1	0.68	0.467	0.218	-0.775	-1.409	206	0	0	0	0	?	303	0 1
cp	1	4	3.158	0.96	0.922	-0.842	-0.401	957	0	0	0	0	?	303	1 4
trestbps	94	200	131.69	17.6	309.751	0.706	0.88	39,902	0	0	0	0	?	303	94 200
chol	126	564	246.693	51.777	2,680.849	1.136	4.492	74,748	0	0	0	0	?	303	126 564
fbs	0	1	0.149	0.356	0.127	1.987	1.96	45	0	0	0	0	?	303	0 1

Figura 3. *Statistics* do ficheiro *Cleveland.csv* parte 2.

- **Longbeach.csv**
  O ficheiro "Longbeach.csv" apresenta 200 linhas de valores de 14 parâmetros, o diagnóstico feito na região apresenta um grupo de pessoas com idade média de 39 anos, onde 6 registos são de identidades femininas e 194 masculinas. A partir do nodo *statistics* ligado ao nodo *file reader* conseguimos ainda conferir outros dados estatísticos. Neste dataset ainda verificamos que os parâmetros *trestbps*, *fbs*, *talachh*, *exang*, *oldpeak*, *slope*, *ca*, *thal e chol* apresentam *missing values*.

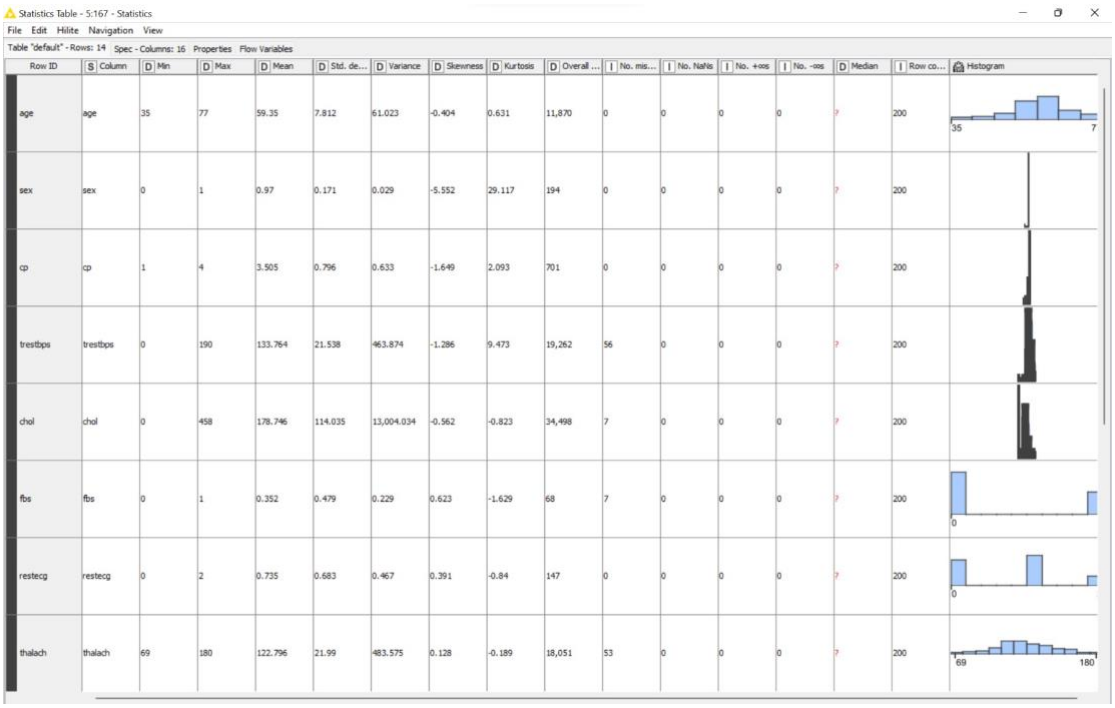


Figura 4. *Statistics* do ficheiro *Longbeach.csv* parte 1.

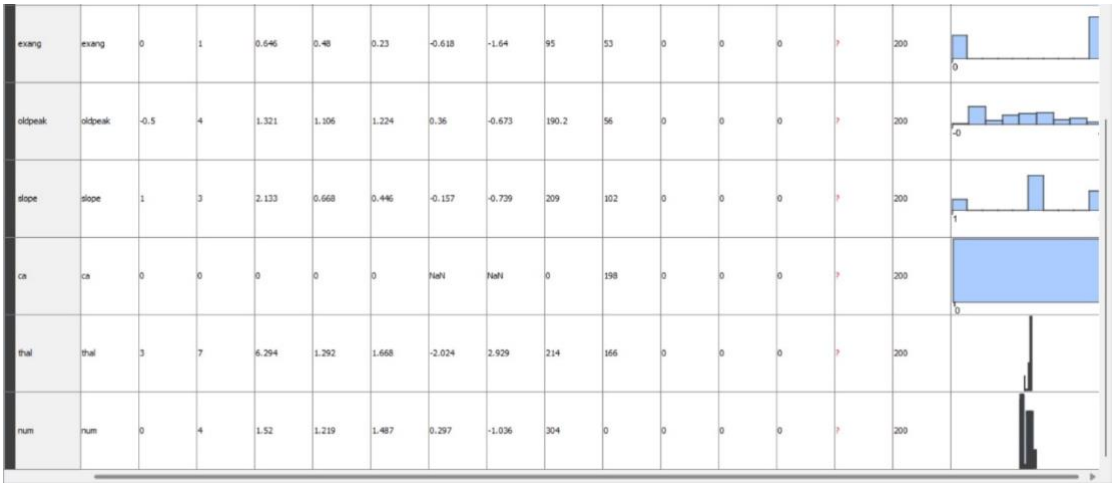


Figura 5. *Statistics* do ficheiro *Longbeach.csv* parte 2.

- **Hungarian.csv**
  O ficheiro "Longbeach.csv" apresenta 294 linhas de valores de 14 parâmetros, o diagnóstico feito na região apresenta um grupo de pessoas com idade média de 46 anos, onde 81 registos são de identidades femininas e 213 masculinas. A partir do nodo *statistics* ligado ao nodo *file reader* conseguimos ainda conferir outros dados estatísticos. Neste dataset ainda verificamos que os parâmetros *trestbps*, *fbs*, *talachh*, *exang*, *slope*, *ca*, *thal e chol* apresentam *missing values*.

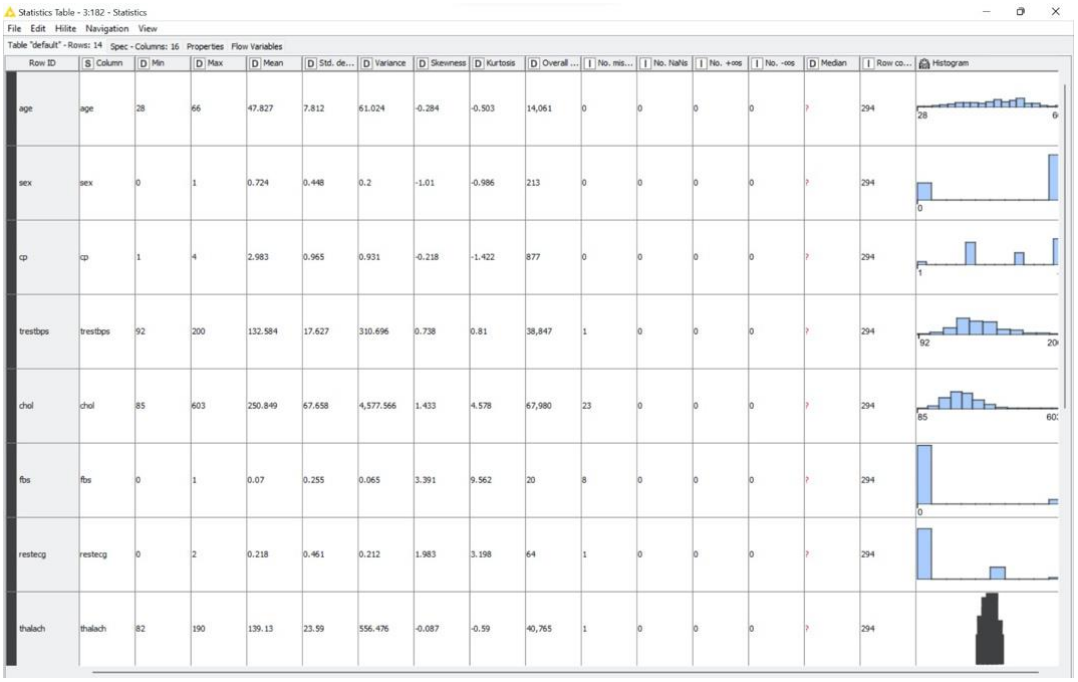


Figura 6. *Statistics* do ficheiro *Hungarian.csv* parte 1.

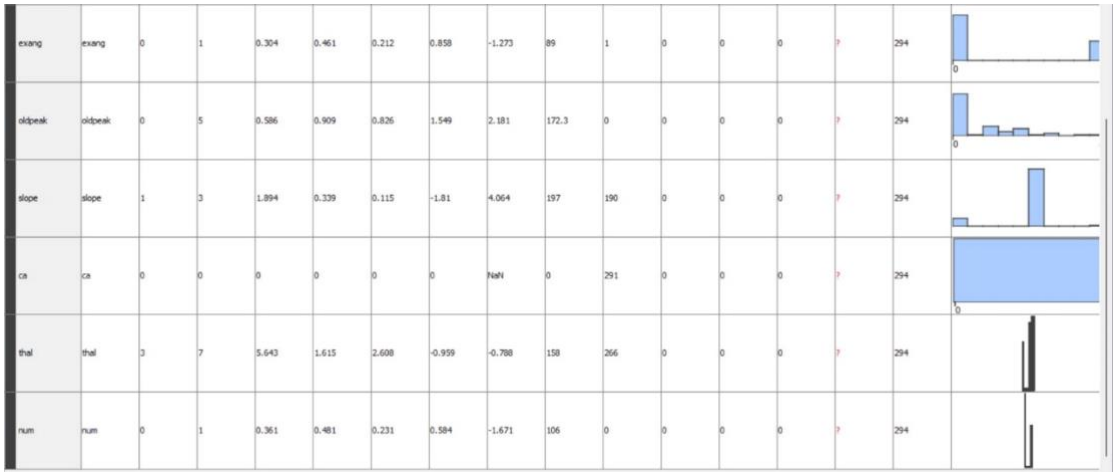


Figura 7. *Statistics* do ficheiro *Hungarian.csv* parte 2.

- **Switzerland.csv**
  O ficheiro "Longbeach.csv" apresenta 123 linhas de valores de 14 parâmetros, o diagnóstico feito na região apresenta um grupo de pessoas com idade média de 22 anos, onde 10 registos são de identidades femininas e 113 masculinas. A partir do nodo *statistics* ligado ao nodo *file reader* conseguimos ainda conferir outros dados estatísticos. Neste dataset ainda verificamos que os parâmetros *trestbps*, *fbs*, *exang*, *slope*, *ca*, *tal, chol, oldpeak e restecg* apresentam *missing values*.

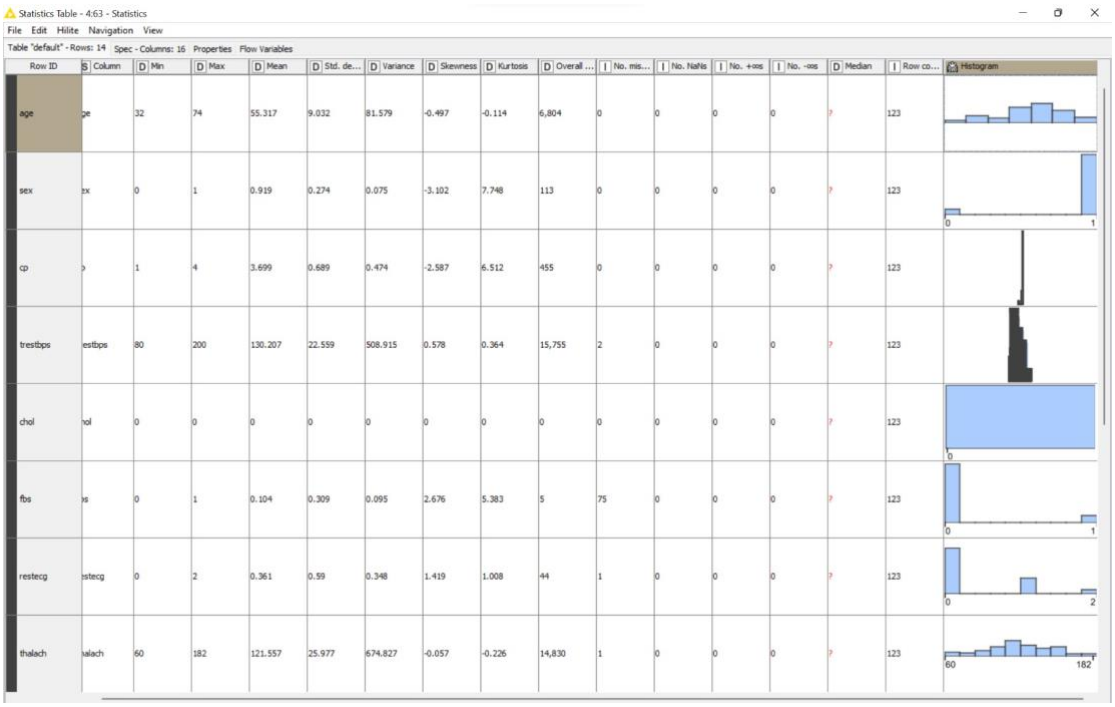


Figura 8. *Statistics* do ficheiro *Switzerland.csv* parte 1.

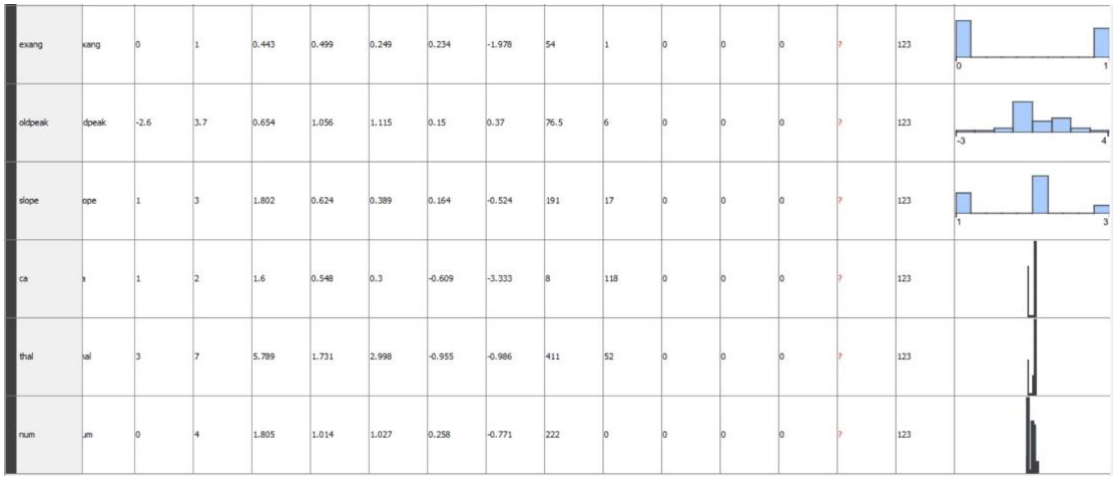


Figura 9. *Statistics* do ficheiro *Switzerland.csv* parte 2.

TRATAMENTO DE DADOS

A aprendizagem automática só poderá ter sucesso se os dados escolhidos forem os adequados e tiverem qualidade. Portanto, é necessário haver um pré-processamento de dados que inclui procedimentos de limpeza e transformação de dados, para que estes se tornem eficientes. A limpeza de dados pretende eliminar erros, tratar os casos de *missing values* ou deteção de inconsistências em registos.

Quando se observa *missing values* numa base de dados, existem várias formas de lidar com eles, nomeadamente:

- Remover as linhas ou colunas que contêm os *missing values*;
- Preencher os *missing values* com um valor específico, como, por exemplo, a média ou mediana da coluna;
- Preencher os *missing values* com a previsão de um modelo de *Machine Learning*;
- Deixar os *missing values* como estão e tratá-los como um valor separado durante o processo de tratamento de dados.

Seleciona-se qual a decisão a tomar para um atributo específico com base na percentagem de *missing values* presente na mesma. Se esta for inferior a 1%, consideramos razoável remover a linha onde os valores desconhecidos aparecem. No entanto, a partir de 10%, a quantidade de dados perdida começa a ser significativa, pelo que se escolhe a mediana ou a moda. Note-se que a média não será uma boa interpolação visto que os valores extremos poderão influenciá-la de tal forma que o valor médio do parâmetro em análise não corresponde ao esperado para uma "pessoa média" que tenha participado no estudo. Se a percentagem de valores desconhecidos é da ordem dos 60% ou superior, eliminamos o atributo. Para percentagens intermédias, fez-se experiências com o intuito de determinar qual das substituições do *missing value* se traduz num modelo melhor.

O objetivo da seleção de características é reduzir a quantidade de atributos a considerar nos registos, sem perder informação útil para a tarefa que se vai realizar.

O tratamento dos dados tem como principal objetivo transformar os *data sets* fornecidos de forma a que a informação neles contida esteja apta a ser analisada. Os dados, inicialmente fornecidos, podem:

- Ser incompletos uma vez que apresentam falta de valores em alguns atributos;
- Conter informação desnecessária, visto que identificam valores impossíveis
- conter inconsistências, na medida em que se encontram discrepâncias entre valores/nomes.

Para se proceder à preparação dos dados são efetuadas um conjunto de etapas. A primeira etapa a ser efetuada é a discretização/enumeração, que consiste numa redução do número de valores de um atributo contínuo, dividindo-o em intervalos (método mais utilizado para produzir sumarização de dados); a segunda etapa consiste numa limpeza desses mesmos dados, através do preenchimento de valores de

atributos, da remoção de valores desnecessários e impossíveis e da resolução de inconsistências; em terceiro procede-se à integração de múltiplas fontes de dados, ou seja, os dados que caracterizam o problema podem provir de fontes diversas; em quarto à transformação, ou seja, à normalização e agregação de dados; por último, recorre-se à redução, que pretende obter uma representação reduzida do volume de dados, mas produzindo os mesmos (ou quase os mesmos) resultados analíticos.

ESCOLHA DOS MODELOS

Na realização do trabalho adotou-se como modelos de aprendizagem as redes neuronais e as árvores de decisão. São dois tipos diferentes de modelos, usados para fazer previsões e tomar decisões com base em dados. Cada um deles tem as suas próprias vantagens e desvantagens e são adequados para diversas situações.

Uma rede neuronal é um sistema computacional de base conexionista, usado para aprendizagem e inspirado no funcionamento do cérebro humano. É definida por uma estrutura interligada de unidades computacionais designadas neurónios, que são unidades de processamento com múltiplas entradas e se encontram organizadas em camadas (entrada, saída e *hidden layers*). As redes neuronais são treinadas relativamente ao modo como se interligam os neurónios de camadas adjacentes e como cada neurónio trata os sinais que recebe. Quanto mais se treinar a rede, mais precisa ela se torna na sua previsão. Os parâmetros no modelo de uma rede neuronal clássica são o *decay* para os pesos, o número de neurónios em cada camada escondida e o número de camadas de rede. Para além disso, numa rede neuronal ajustada, são comuns os hiperparâmetros como batch size, epoch, entre outros.

Em *KNIME*, para aplicar um modelo de rede neuronal, usa-se nodos como *Neural Network Learner*, para treinar o modelo, *Neural Network Predictor*, para fazer previões com um modelo já treinado, e *Neural Network Assess* para avaliar o desempenho do modelo treinado.

Por sua vez, as árvores de decisão são utilizadas para realizar a classificação e tomada de decisões. Elas funcionam através da criação de uma estrutura em forma de árvore, na qual cada nó representa uma decisão a ser tomada com base num conjunto de atributos e cada ramo representa uma possível escolha dessa decisão. As árvores de decisão são treinadas com base em exemplos de treino, em que cada exemplo é um conjunto de valores para os atributos e uma classe a ser prevista. O objetivo é criar uma árvore que seja capaz de prever a classe correta para novos exemplos baseado nas decisões tomadas ao longo da árvore. A vantagem deste modelo é a sua relativa simplicidade, onde é possível interpretar a lógica de decisão entre classes na própria estrutura criada.

Em *KNIME*, para aplicar o modelo de árvores de decisão, usa-se nodos como *Decision Tree Learner* para treinar e *Decison Tree Predictor* para fazer previsões com a árvore de decisão já treinada.

As principais diferenças entre redes neuronais e árvores de decisão baseiam-se:

- Complexidade: redes neuronais são mais complexas que árvores de decisão. Elas são compostas por várias camadas de neurónios interconectados que trabalham juntos para fazer previsões, enquanto as árvores de decisão são compostas por uma série de ramos e folhas que representam diferentes decisões e resultados;

- Separação de dados: as redes neuronais são geralmente melhores para conjuntos de dados complexos e não linearmente separáveis, enquanto árvores de decisão são mais adequadas para conjuntos de dados mais simples;
- Interpretabilidade: árvores de decisão são mais fáceis de interpretar do que redes neuronais, pois elas geram uma representação visual da lógica que está por detrás das previsões. Por outro lado, as redes neuronais são mais difíceis de interpretar pois são compostas por uma série de neurónios interconectados que trabalham juntos de maneira complexa;
- Tempo de treino: árvores de decisão demoram menos tempo a treinar do que redes neuronais. Tal acontece devido à complexidade adicional de redes neuronais, que precisam de mais tempo para processar e ajustar os pesos dos neurónios para fazer previsões precisas.

Como o objetivo neste trabalho é desenvolver um sistema de aprendizagem automática capaz de realizar o diagnóstico sobre se o paciente tem ou não uma doença cardíaca, ou seja, fazer previsões de classificação, ambos os modelos são adequados.

Assim, para cada ficheiro construiu-se um sistema de aprendizagem em árvores de decisão e em redes neuronais, procedendo posteriormente à testagem. Para se saber qual dos métodos foi o melhor, tem de se comparar as métricas de avaliação de cada modelo. Contudo, não se deve focar unicamente nestas métricas de avaliação e ter igualmente em conta fatores como o tempo de treino, a complexidade de cada modelo e a quantidade de dados disponíveis.

CLEVELAND

A criação do sistema de aprendizagem automática, através do modelo de redes neuronais apresenta a seguinte estrutura:

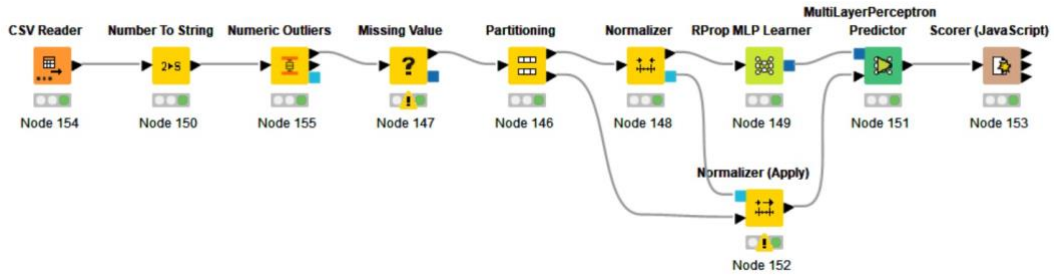


Figura 10. Representação da rede neuronal.

Este modelo inicia com a utilização do nodo *CSV Reader*, o qual recebe o ficheiro de *Exel* com a informação e procede à leitura do mesmo.

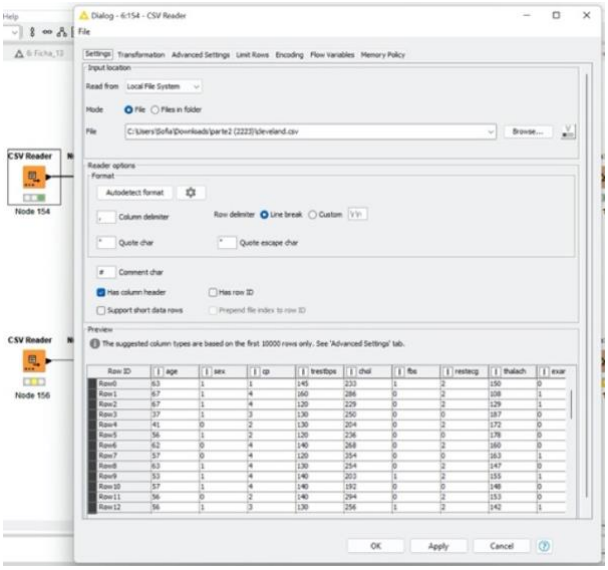


Figura 11. Aplicação do nodo *CSV Reader*.

Seguidamente, recorreu-se ao nodo *Number to String,* para converter o atributo *num* numa *string*. Esta conversão é necessária, uma vez que o nodo *Scorer* só consegue comparar duas colunas se os parâmetros a avaliar forem nominais ou *strings*.

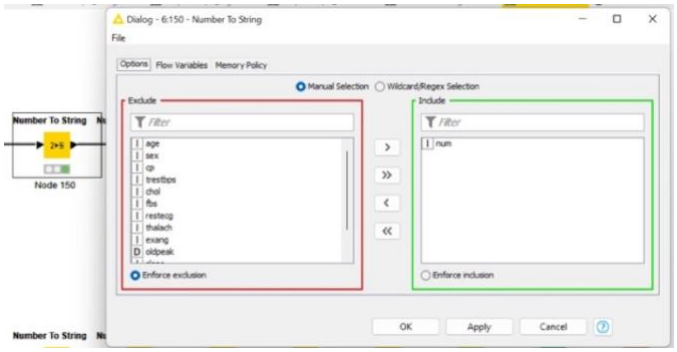


Figura 12. Aplicação do nodo *Number to String*.

Para além disso, utilizou-se o nodo *Numeric Outliers*, o qual vai detetar e tratar dos valores discrepantes numéricos das colunas *age*, *sex*, *tretbps*, *chol*, *talach*, *exang*, *oldpeak* e *thal*, considerando-os igualmente *missing values*.

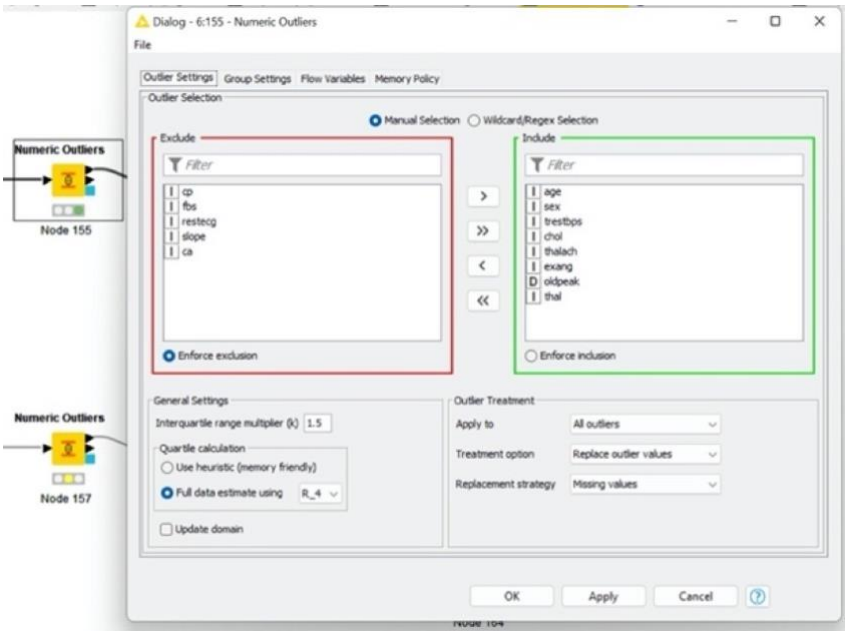


Figura 13. Aplicação do nodo *Numeric Outliers*.

A colocação do nodo *Missing Value* permitiu tratar os valores desconhecidos através da eliminação da coluna onde se encontram ou utilização da mediana. A escolha do tratamento varia dependendo da percentagem de *missing values* existentes em cada atributo.

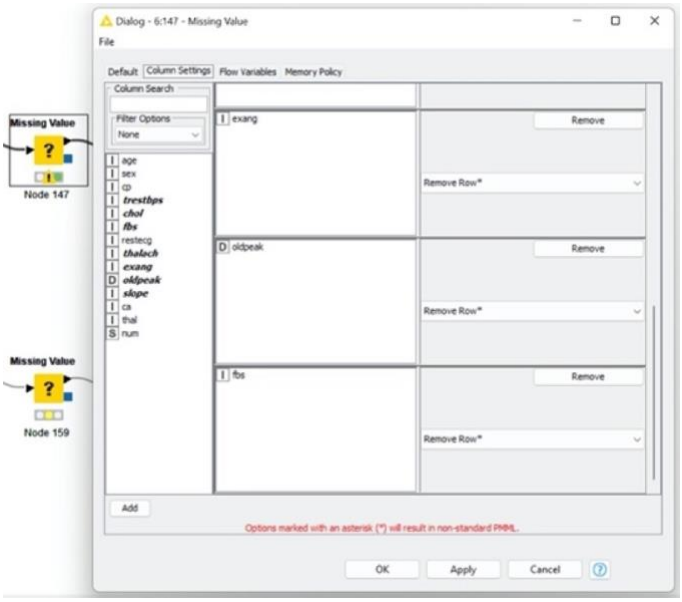


Figura 14. Aplicação do nodo *Missing Values* aos atributos *exang, oldpeak* e *fbs*.

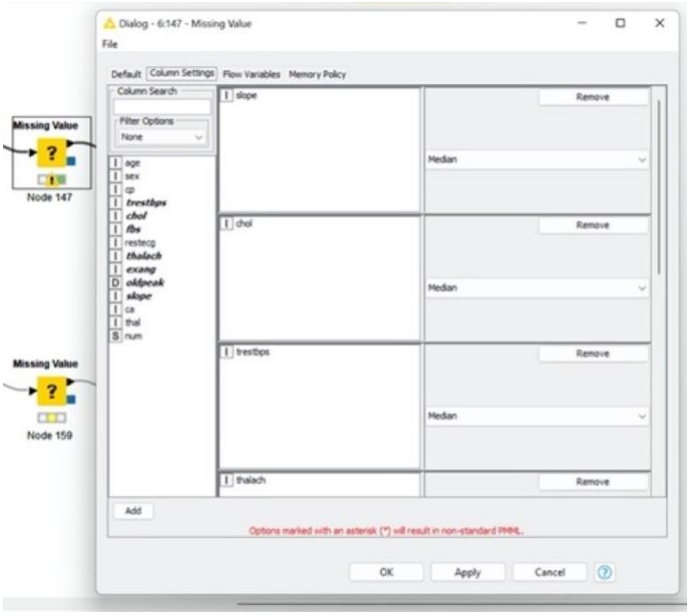


Figura 15. Aplicação do nodo *Missing Values* aos atributos *slope, chol* e *trestbps*.

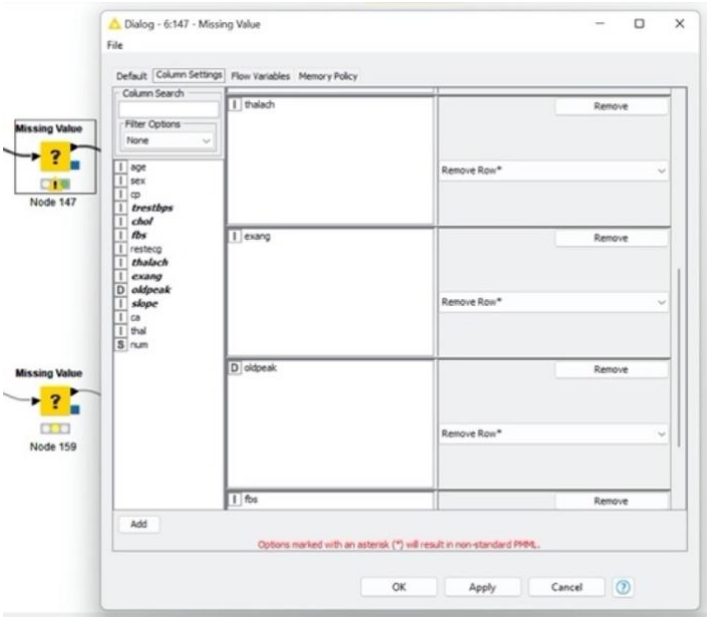


Figura 16. Aplicação do nodo *Missing Value* aos atributos *talach, exang, oldpeak* e *fbs*.

De seguida, aplicou-se o nodo *Partitioning*, com uma percentagem de partição de 70%. Utilizou-se, ainda, a opção *random seed* para garantir a repetitividade.

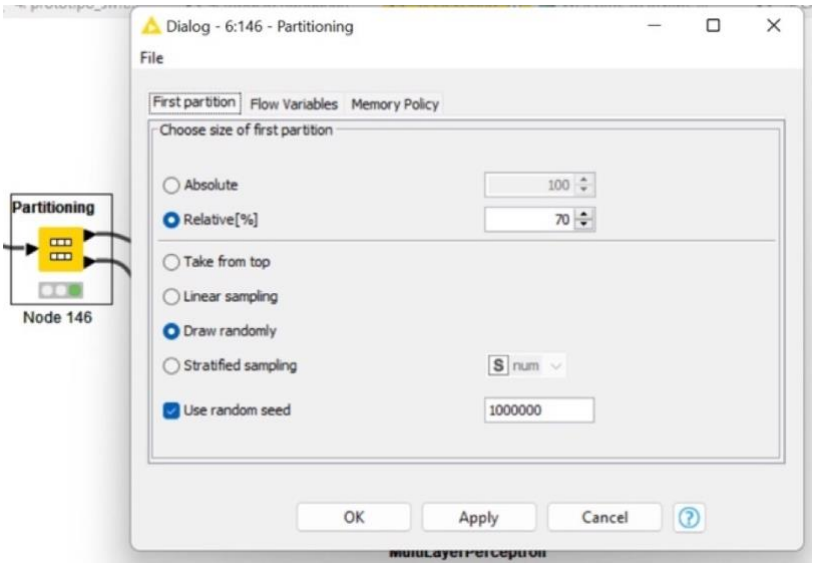


Figura 17. Aplicação do nodo *Partitioning*.

Este nodo é ligado a outros dois nodos, o *Normalizer* e o *Normalizer (Apply)*.

No *Normalizer* fez-se a normalização dos atributos para que todos tenham a mesma ordem de grandeza.

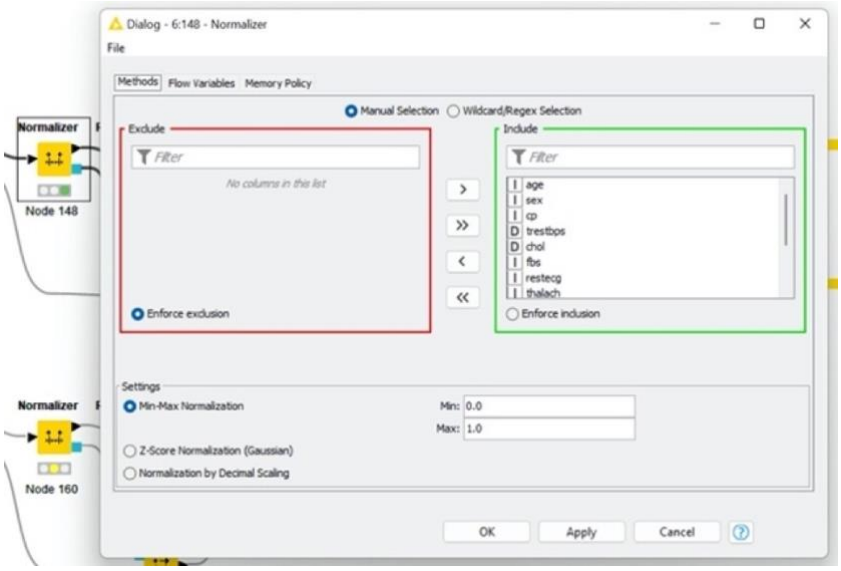


Figura 18. Aplicação do nodo *Normalizer*.

No nodo *Normalizer (Apply)* normalizou-se os dados de entrada de acordo com os parâmetros de normalização já fornecidos.

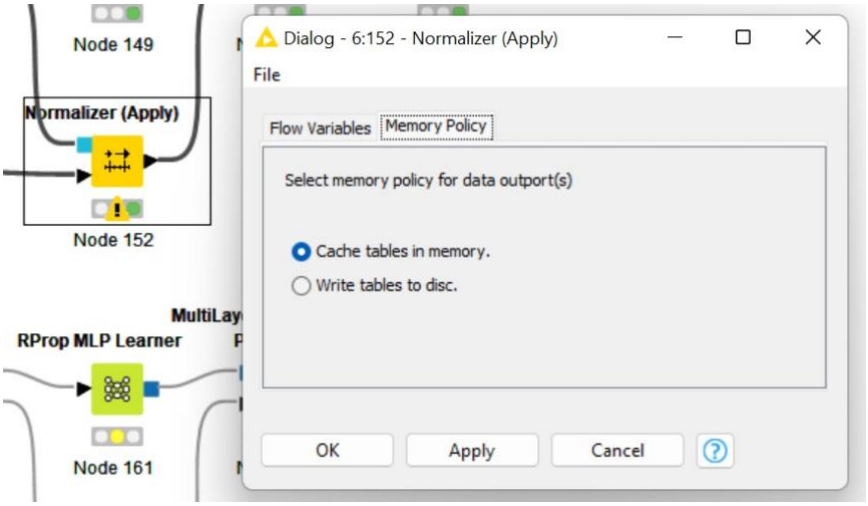


Figura 19. Aplicação do nodo *Normalizer (Apply)*.

Em seguida, utilizou-se o nodo *RProp MLP Learner* para identificar o atributo *num* como aquele que se quer prever. Utilizou-se mais uma vez a opção de *random seed*..

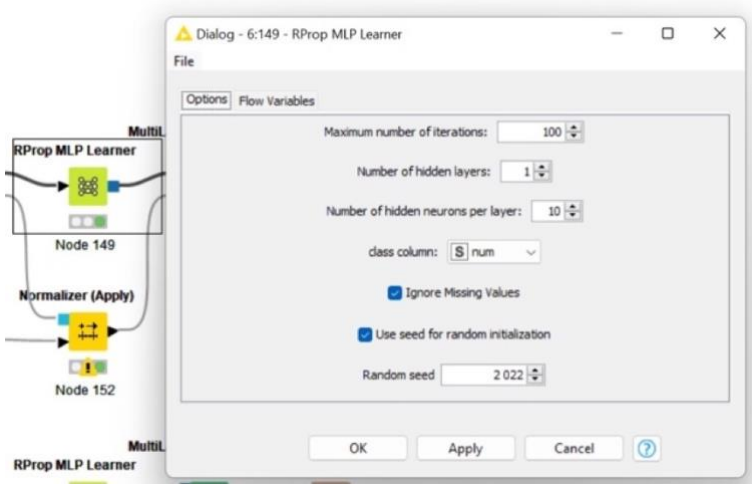


Figura 20. Aplicação do nodo *RProp MLP Learner*.

De seguida, vem o nodo *Multilayer Perceptron Predictor*, que calcula os valores de saída da rede neuronal.

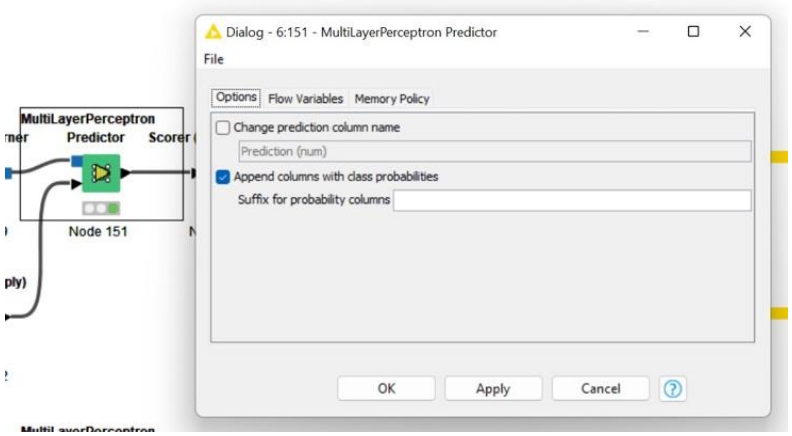


Figura 21. Aplicação do nodo *Multilayer Perceptron Predictor*.

Por último, recorreu-se ao nodo *Scorer (JAVA Script)*, onde é possível observar os seguintes valores da matriz de confusão:

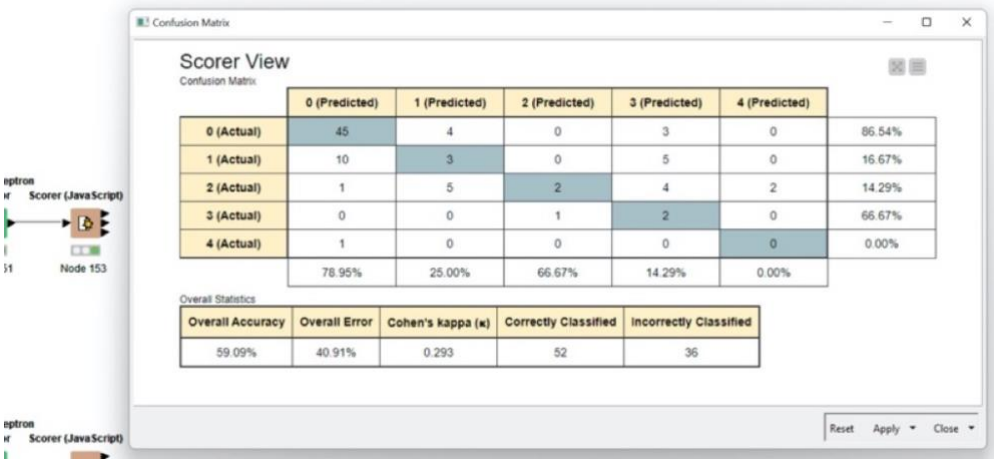


Figura 22. Visualização da matriz de confusão.

Tendo em conta o valor obtido para a accuracy na matriz de confusão, percebemos que o modelo não é muito confiável. Tal poderá dever-se à distribuição dos dados na database. De facto, cerca de metade dos doentes eram saudáveis, o que implica que na metade restante estejam compilados todos os outros graus de doença, resultando num número reduzido de dados para os valores 1, 2, 3 e 4 do target. Possivelmente, ao substituir os *missing values* por medianas, pôs-se muitos atributos a assumir um valor correspondente a alguém saudável, pelo que o modelo tem mais tendência a consider o target 0.

HUNGARIAN

O primeiro passo para construir o workflow é a preparação de dados.

O aspeto mais evidente e que salta imediatamente à vista neste dataset é o facto de a coluna "num", correspondente ao target, não estar dividida em 5 valores (0 se o paciente é saudável e 1, 2, 3 e 4 caso contrário, dependendo do número de vasos sanguíneos obstruídos), mas apenas em dois: 0, se apresenta doença cardíaca e 1, se não o faz.

Para além disso, percebemos que os atributos "slope", "ca" e "thal" possuem um número exacerbante de *missing values*.

Uma vez mais, numa primeira fase, construiu-se o sistema de aprendizagem automática através do modelo de redes neuronais.

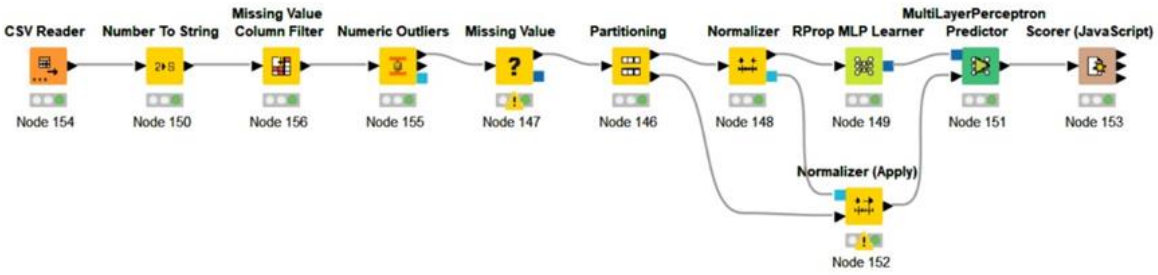


Figura 23. Representação da rede neuronal.

Deste modo, procede-se à inserção do nodo *CSV Reader*, para que ele receba o ficheiro de *Exel* com a informação e proceda à leitura do mesmo.

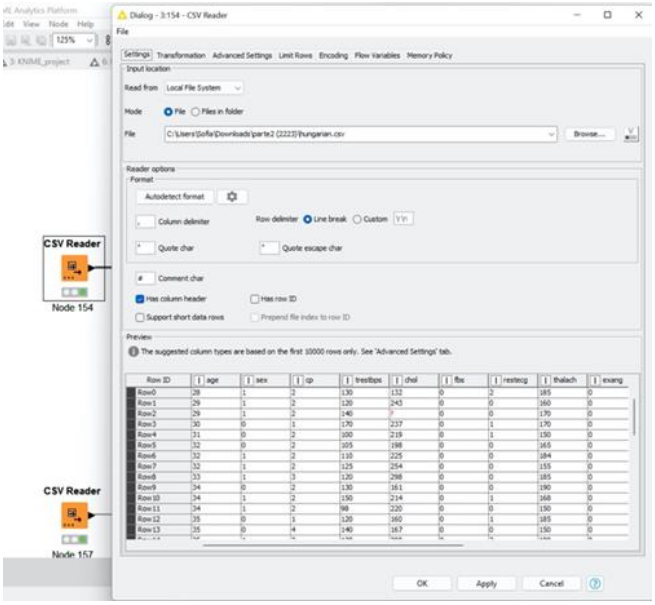


Figura 24. Aplicação do nodo *CSV Reader*.

De seguida, utilizou-se o nodo *Number to String,* para converter o atributo *target* numa *string*. Tal é necessário pelo facto do *Scorer* apenas conseguir avaliar a qualidade do modelo criado se a *label* é uma *string*.

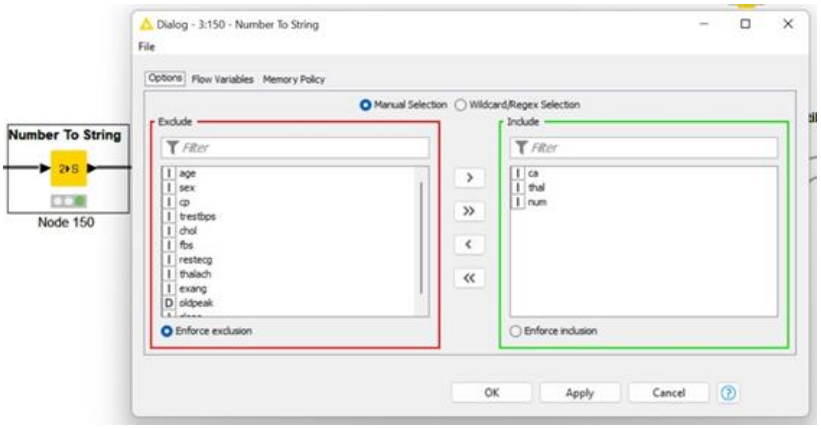


Figura 25. Aplicação do nodo *Number to String*.

Posteriormente, procedeu-se à colocação do nodo *Missing Value Column Filter* para eliminar por inteiro as colunas dos atributos *slope*, *ca* e *thal*, pois possuem um número exacerbante de *missing values*.

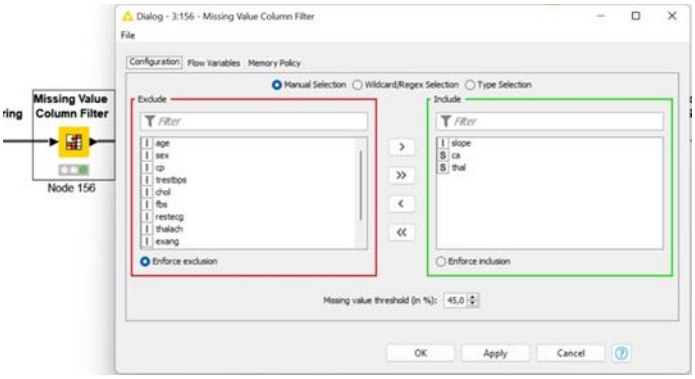


Figura 26. Aplicação do nodo *Missing Value Column Filter*.

Para além disso, utilizou-se o nodo *Numeric Outliers*, o qual vai detetar e tratar dos valores discrepantes numéricos das restantes colunas, considerando-os igualmente *missing values*. Esta etapa é importante para que depois se possa efetuar o tratamento desses valores.

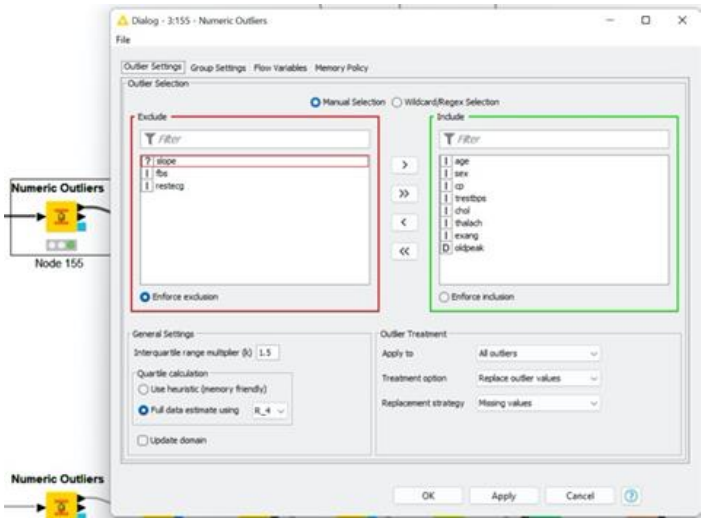


Figura 27. Aplicação do nodo *Numeric Outliers*.

O nodo *Missing Values* permitiu tratar os *missing values* através da utilização da mediana ou eliminado a coluna onde eles se encontram. A escolha do tratamento varia dependendo da percentagem de *missing values* existentes em cada atributo.

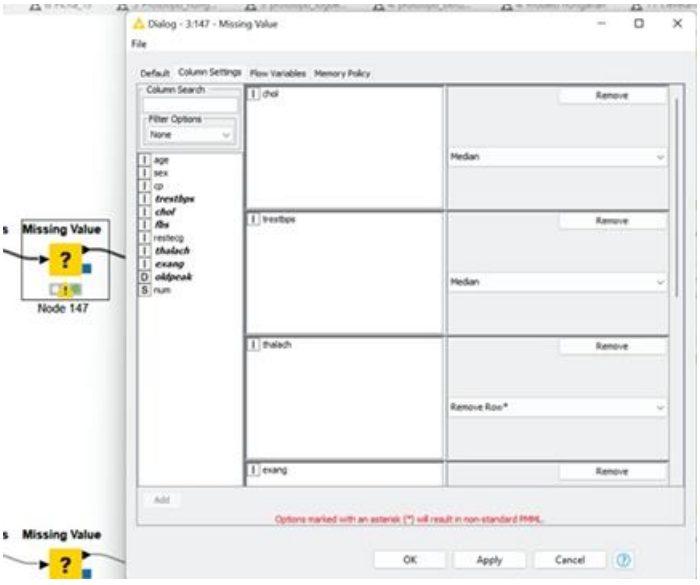


Figura 28. Aplicação do nodo *Missing Values* aos atributos *col*, *tretbps* e *talach*.

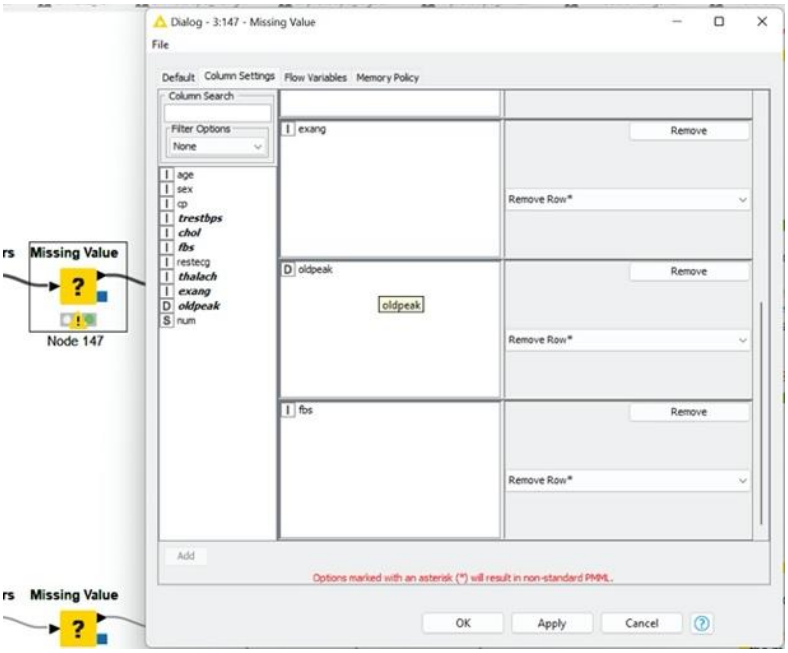


Figura 29. Aplicação do nodo *Missing Values* aos atributos *exang*, *oldpeak* e *fbs*.

Houve ainda a necessidade de recorrer ao nodo *Partitioning*, ao qual se aplicou uma partição de 76%. Utilizou-se, ainda, a opção *random seed* para garantir a repetitividade.

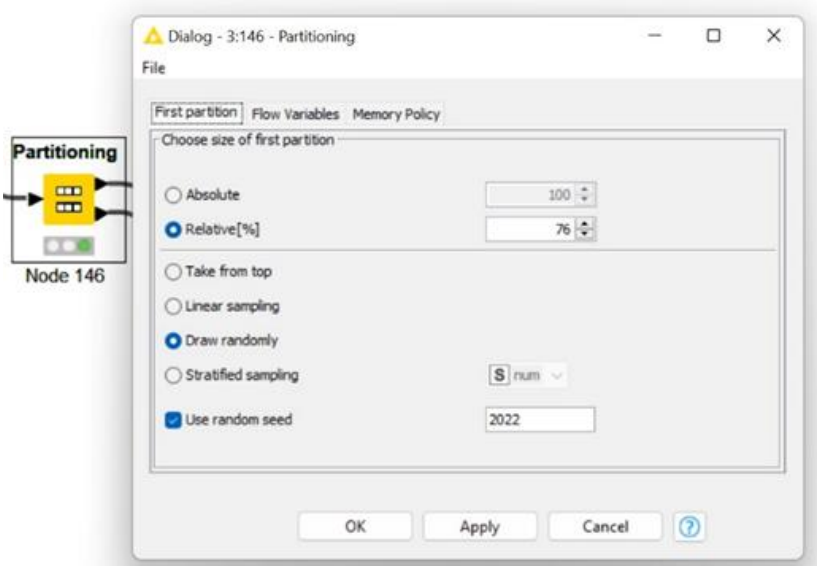


Figura 30. Aplicação do nodo *Partitioning*.

Por sua vez, no nodo *Normalizer* fez-se a normalização dos atributos para que todos tenham a mesma ordem de grandeza.

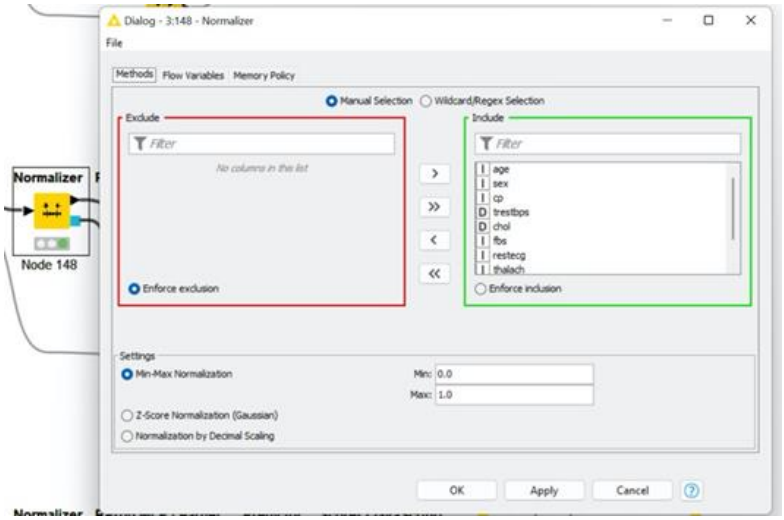


Figura 31. Aplicação do nodo *Normalizer*.

No nodo *Normalizer (Apply)* normalizou-se os dados de entrada de acordo com os parâmetros de normalização já fornecidos.

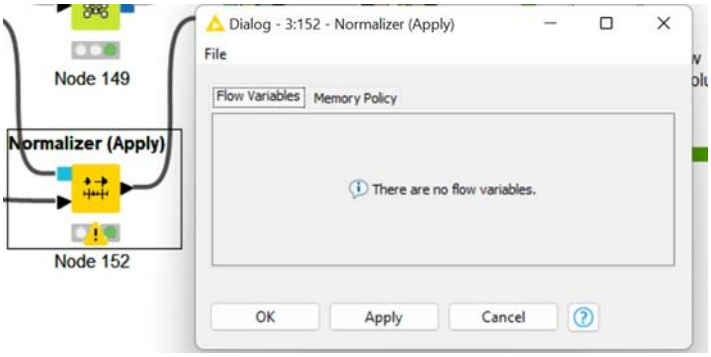


Figura 32. Aplicação do nodo *Normalizer (Apply)*.

De seguida, utilizou-se o nodo *RProp MLP Learner* para identificar o atributo *num* como aquele que se quer prever. Utilizou-se mais uma vez a opção de *random seed*.

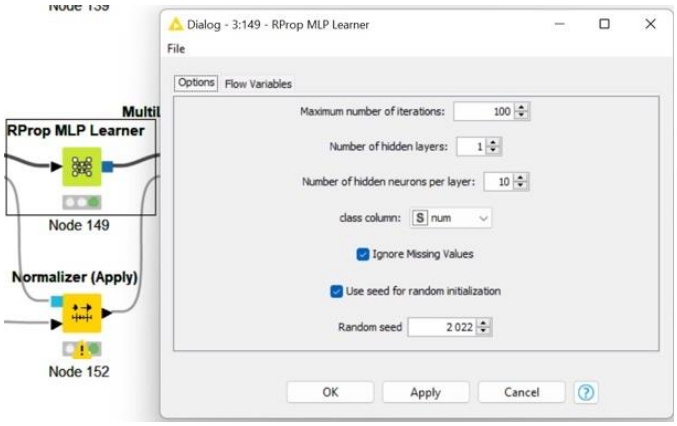


Figura 33. Aplicação do nodo *RProp MLP Learner*.

No nodo *Multilayer Perceptron Predictor* calculou-se os valores de saída da rede neuronal.

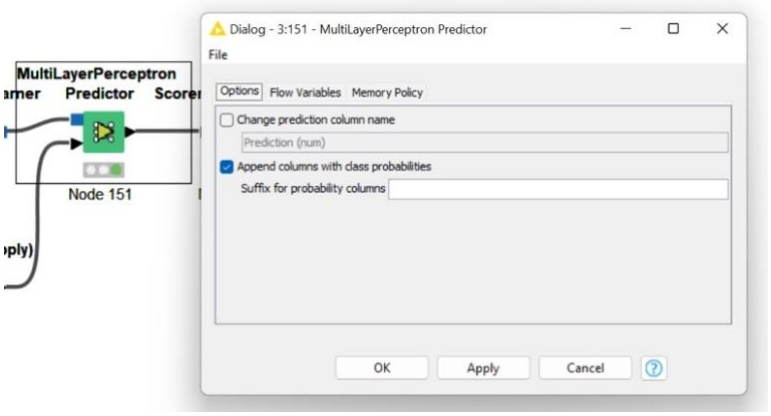


Figura 34. Aplicação do nodo *Multilayer Perceptron Predictor*.

Por último, recorreu-se ao nodo *Scorer (Java Script)*, no qual foi possível observar os seguintes valores da matriz de confusão:

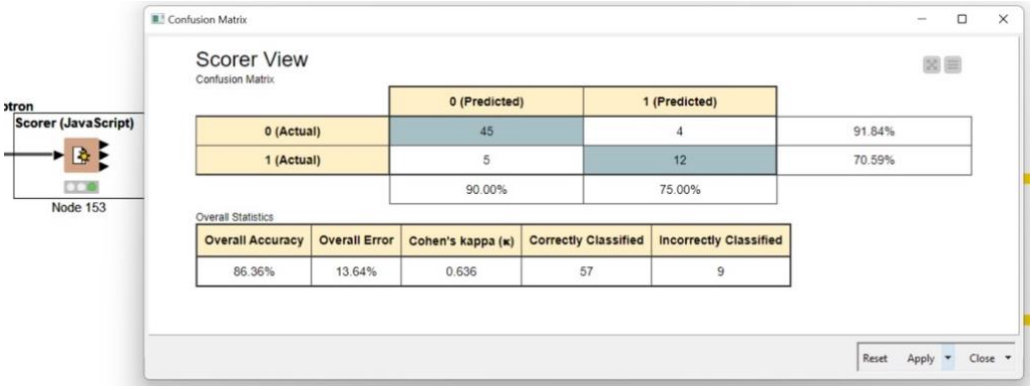


Figura 35. Visualização da matriz de confusão.

Analisando o valor de *accuracy* obtido, concluímos que o modelo conseguido é satisfatoriamente confiável, tendo em conta que o tratamento de *missing values* foi feito de forma que estes fossem interpolados da melhor forma possível e, simultaneamente, estatisticamente credível. No entanto, seria de esperar que, comparativamente aos outros *databases*, a *accuracy* do modelo obtido pelo estudo feito na Hungria fosse mais elevada, uma vez que as previsões só classificam os pacientes como doentes ou saudáveis, sem especificar o estado da doença.

LONG BEACH

A partir do raciocínio aplicado aos outros dois ficheiros, também aqui se iniciou com o modelo de redes neuronais para a construção do sistema de aprendizagem automática. Para este ficheiro efetuou-se dois testes, um ao qual foi aplicado um modelo de redes neuronais binarizado e outro ao qual foi aplicado um modelo de redes neuronais não binarizado. Um modelo diz-se binarizado quando os seus valores são binários, ou seja, variam entre 0 e 1. Deste modo, o nosso modelo de redes neuronais binarizado deixa de prever os valores entre 0, 1, 2, 3 e 4 para apenas 0 e 1. A diferença entre os dois modelos reside na introdução de mais um nodo *Rule Engine* no que é binarizado.

O modelo de redes neuronais não binarizado apresenta a seguinte configuração:

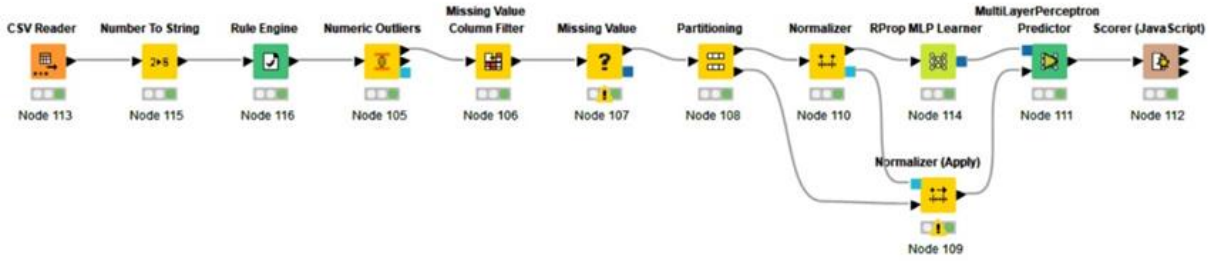


Figura 36. Representação da rede neuronal não binarizada.

Assim, inseriu-se o nodo *CSV Reader*, para que ele receba o ficheiro de *Exel* com a informação e proceda à leitura do mesmo.

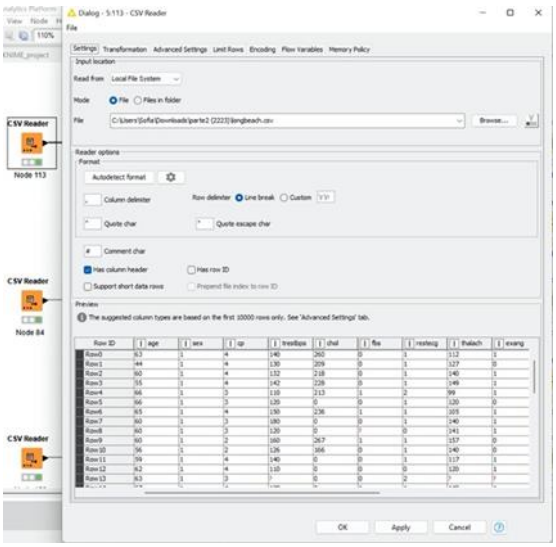


Figura 37. Aplicação do nodo *CSV Reader*.

Posteriormente, recorreu-se ao nodo *Number to String,* para converter o atributo target numa *string*. Esta etapa é necessária para que, o último nodo, *Scorer*, seja capaz de comparar a coluna *num* com a previsão de *num*, isto apenas acontecerá se *num*, o nosso *target*, for um atributo de tipo nominal ou string.

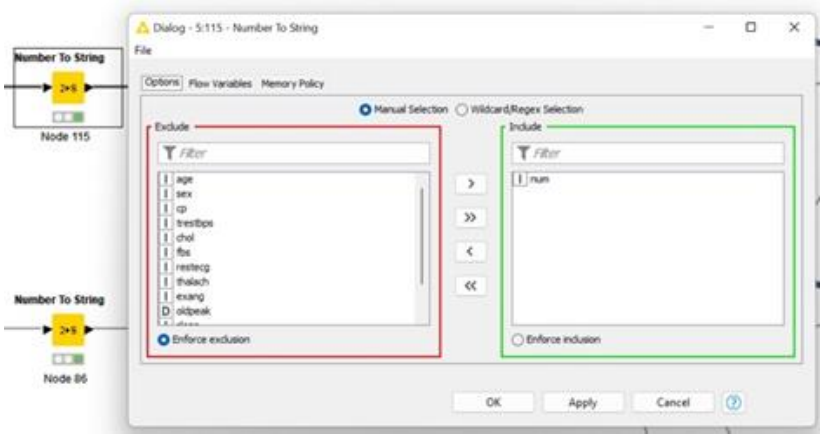


Figura 38. Aplicação do nodo *Number to String*.

O nodo *Rule Engine* permitiu passar o atributo *chol* para *missing values*. Isto porque a coluna desse atributo apresentava a número bastante elevado de zeros para o valor do colesterol, algo erradamente medido.

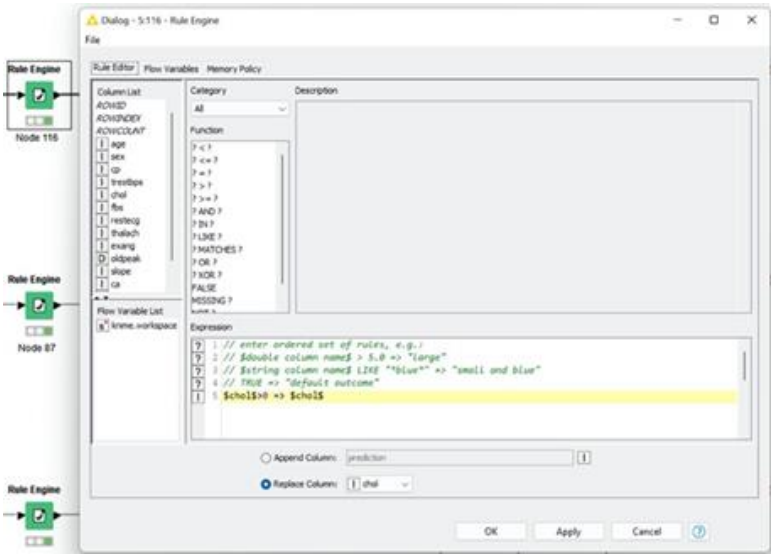


Figura 39. Aplicação do nodo *Rule Engine*.

Para além disso, utilizou-se o nodo *Numeric Outliers*, o qual vai detetar e tratar dos valores discrepantes numéricos presentes nas colunas *tretbps*, *chol*, *restecg*, *thalach*, *oldpeak*, *slope* e *ca*. Esses valores serão considerados *missing values*.

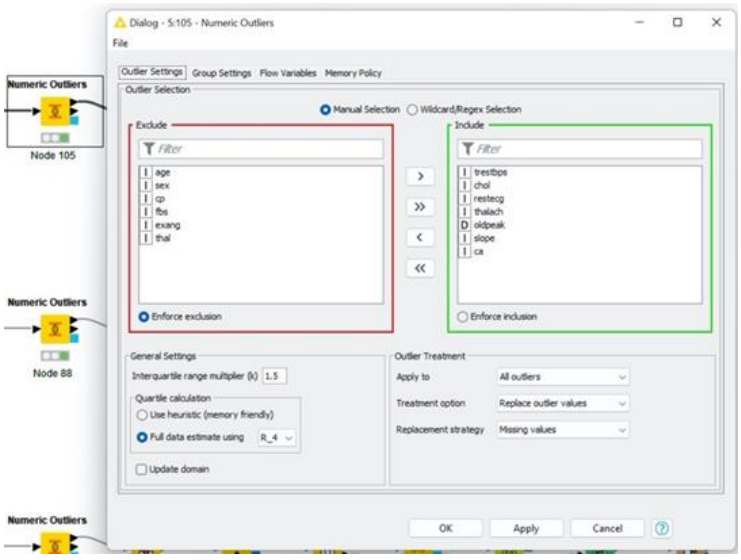


Figura 40. Aplicação do nodo *Numeric Outliers*.

De seguida, procedeu-se à colocação do nodo *Missing Value Column Filter* para eliminar as colunas dos atributos *slope*, *ca* e *tal*, pois possuem um número considerável de *missing values*.

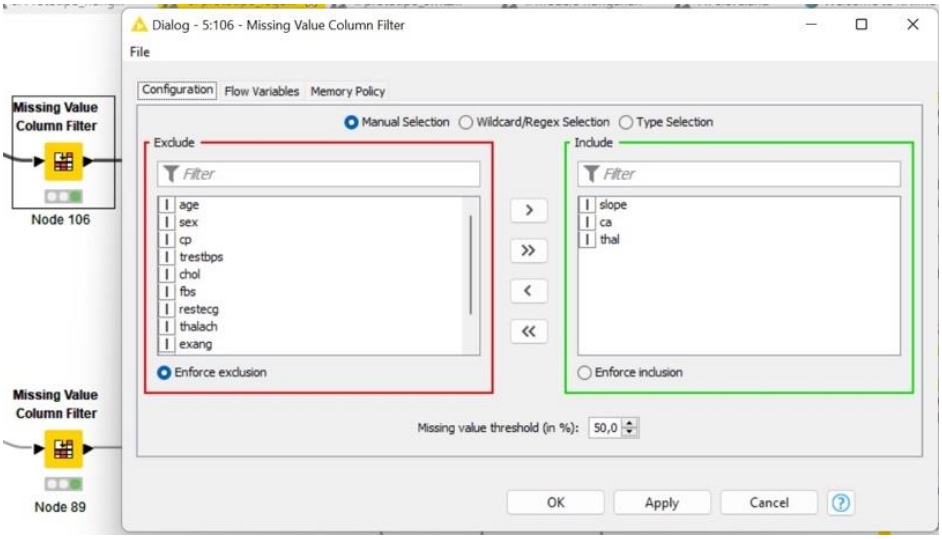


Figura 41. Aplicação do nodo *Missing Value Column Filter*.

O nodo *Missing Value* permitiu tratar os *missing values* através da utilização da mediana ou eliminado a coluna onde eles se encontram. A escolha do tratamento varia dependendo da percentagem de *missing values* existentes em cada atributo.

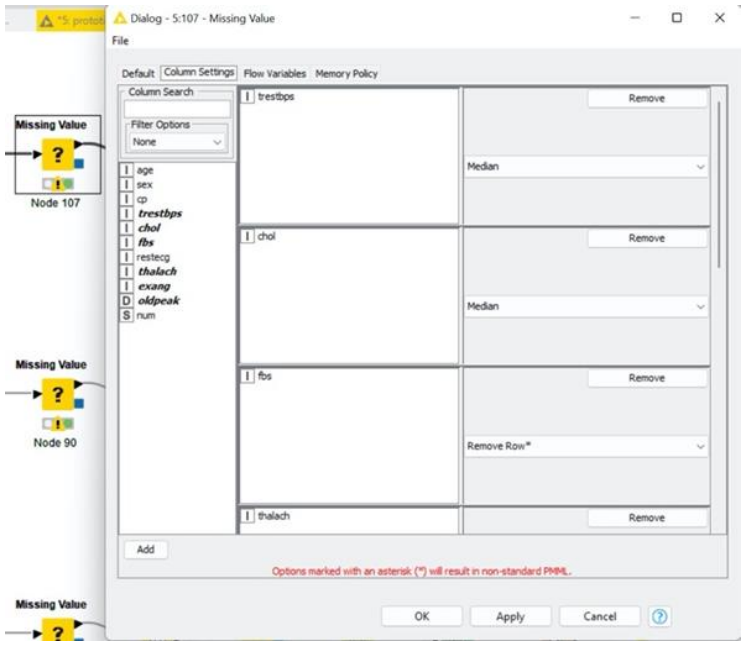


Figura 42. Aplicação do nodo *Missing Value* aos atributos *tretbps*, *chol* e *fbs*.

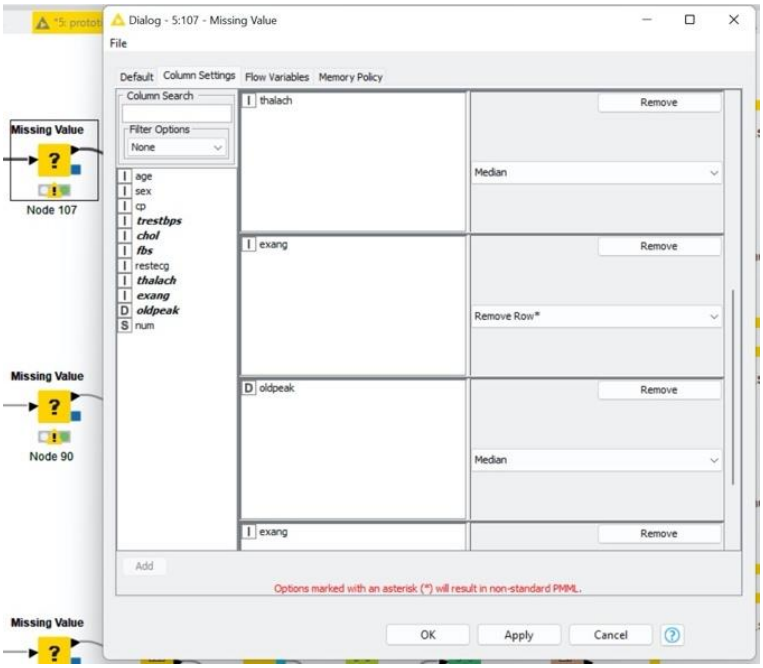


Figura 43. Aplicação do nodo *Missing Value* aos atributos *talach*, *exang* e *oldpeak*.

Inseriu-se também o nodo *Partitioning*, ao qual se aplicou uma partição de 70%. Utilizou-se, ainda, a opção *random seed* para garantir a repetitividade.

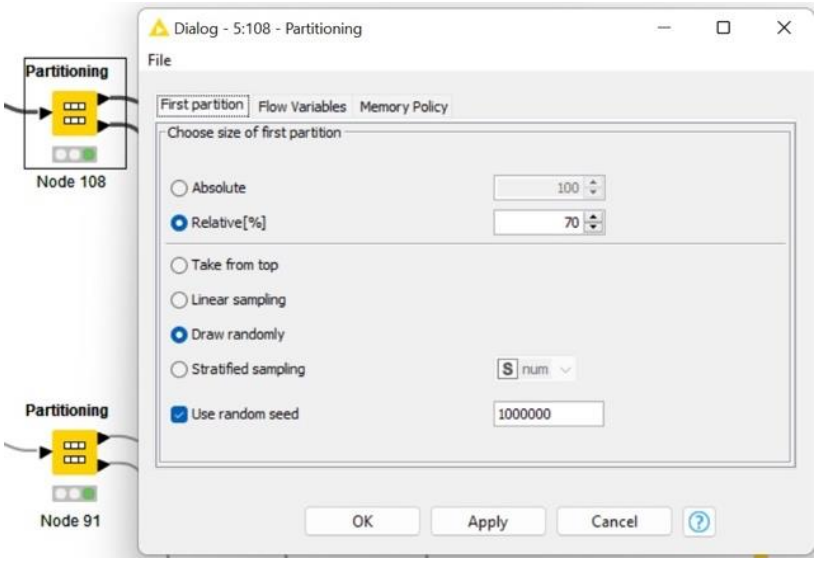


Figura 44. Aplicação do nodo *Partitioning*.

Por sua vez, no nodo *Normalizer* fez-se a normalização dos atributos para que todos tenham a mesma ordem de grandeza.

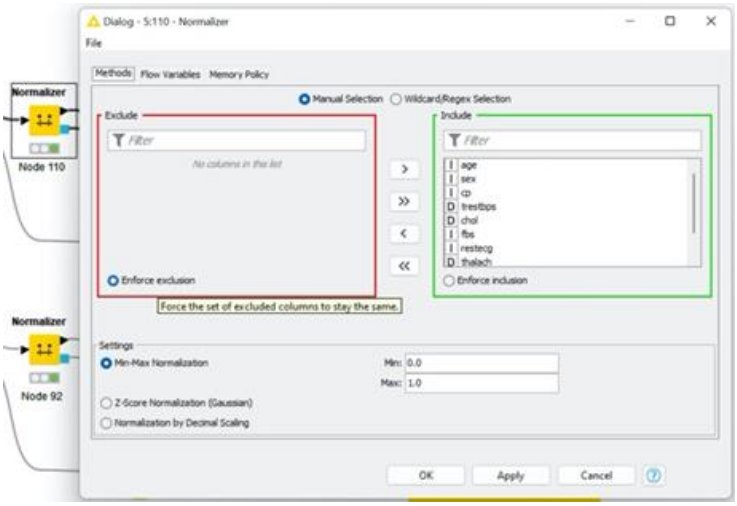


Figura 45. Aplicação do nodo *Normalizer*.

Já no nodo *Normalizer (Apply)* realizou-se a normalização dos dados de entrada de acordo com os parâmetros de normalização já fornecidos.

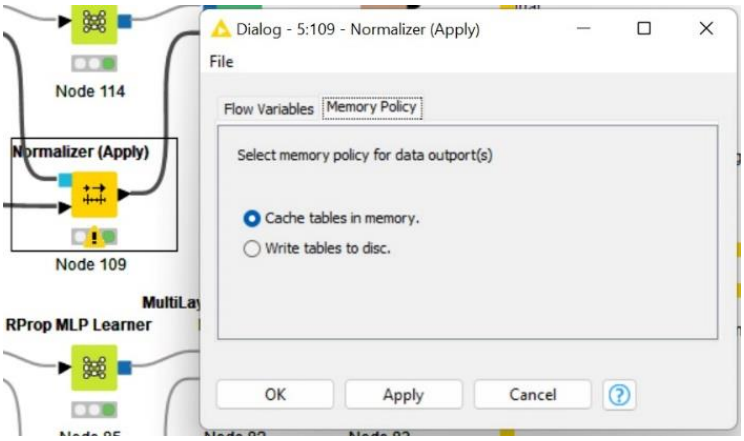


Figura 46. Aplicação do nodo *Normalizer (Apply)*.

De seguida, utilizou-se o nodo *RProp MLP Learner* para identificar o atributo *num* como aquele que se quer prever. Utilizou-se mais uma vez a opção de *random seed*.

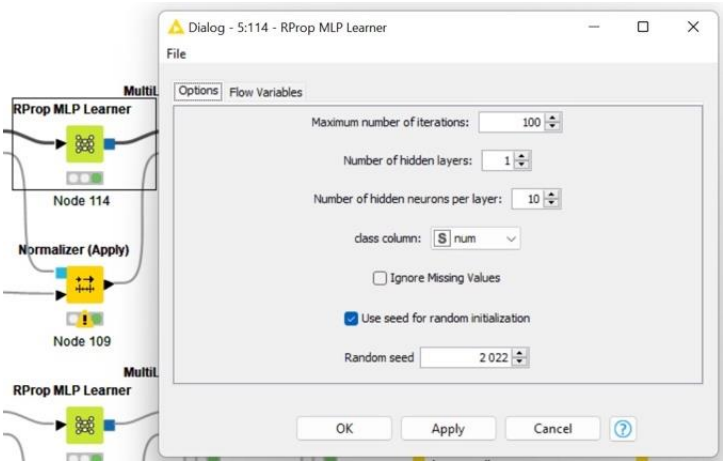


Figura 47. Aplicação do nodo *RProp MLP Learner*.

No nodo *Multilayer Perceptron Predictor*, calculou-se os valores de saída da rede neuronal.

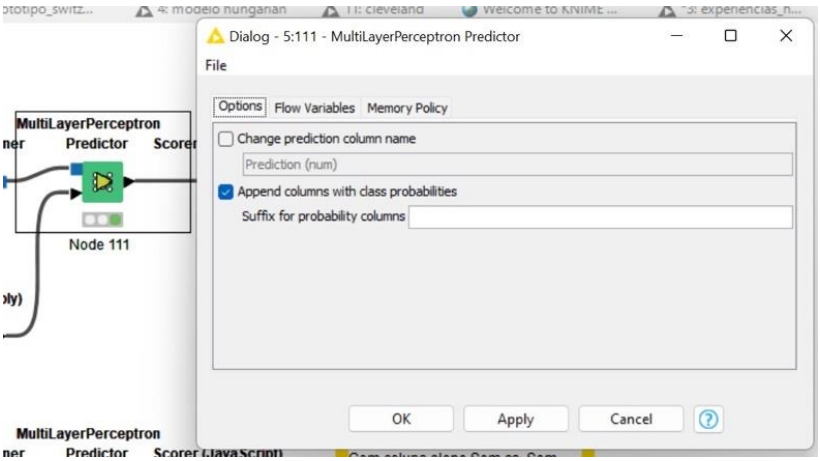


Figura 48. Aplicação do nodo *Multilayer Perceptron Predictor*.

Finalmente, recorreu-se ao nodo *Scorer (JAVA Script)*, no qual foi possível observar os seguintes valores da matriz de confusão:

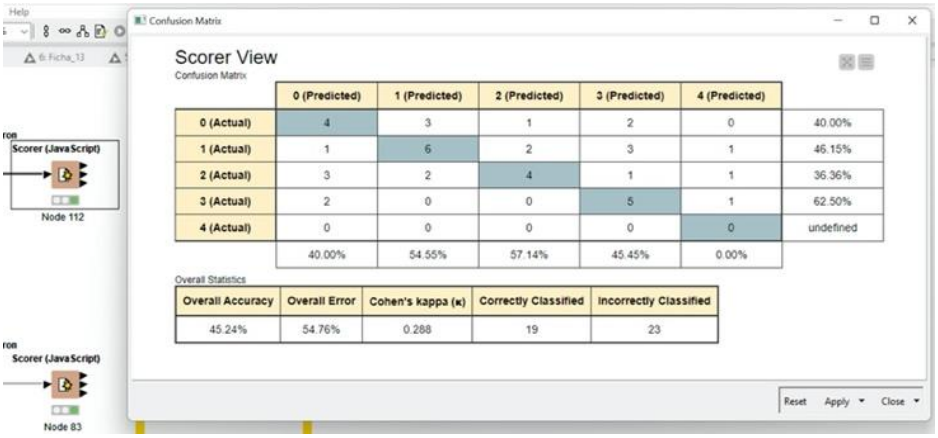


Figura 49. Aplicação do nodo *Scorer (JAVA Script)*.

A matriz de confusão mostra que o modelo é de baixa qualidade e possui uma *accuracy*. Nesta base de dados, muitos atributos tinham uma percentagem de *missing values* muito significativa, o que fez com que alguns não pudessem sequer ter sido utilizados, e outros fossem substituídos por valores como a mediana, resultando num modelo sem utilidade. Ora, de forma a conseguir uma resposta a um problema útil, ainda que não o original, decidiu-se tornar o target, mais uma vez, binário. Efetivamente, sabemos que o facto de o modelo conseguir identificar se o paciente apresenta ou não doença cardíaca é relevante, uma vez que o hospital da Hungria recolheu os dados dessa forma, sem nunca ter ambicionado concluir relativamente ao nº de vasos sanguíneos obtidos.

Posto isto, testou-se então o modelo de redes neuronais binarizado. A estrutura é semelhante, variando apenas num nodo *Rule Engine* que é acrescentado neste modelo. Assim, o modelo apresenta a seguinte esquemática:

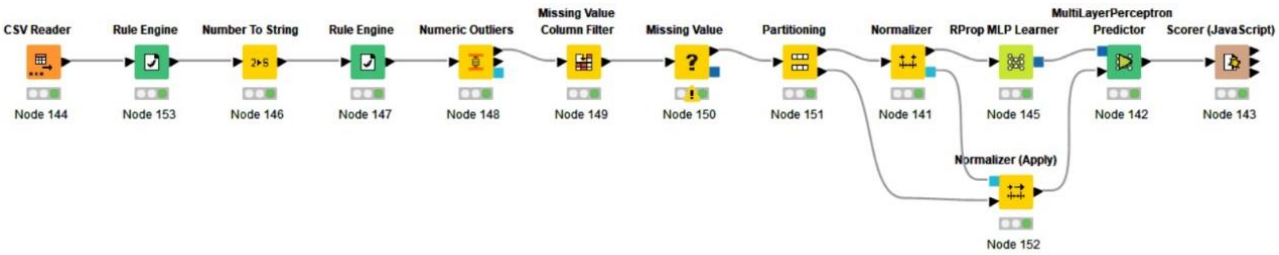


Figura 50. Representação da rede neuronal binarizada.

O nodo *Rule Engine,* que se adicionou neste modelo, permitiu passar o atributo *num* para binário. Assim, os valores que antes variavam entre 0, 1, 2, 3 e 4, ficam limitados a 0 e 1 para representar, respetivamente, a existência de doença cardíaca e a ausência de doença cardíaca.

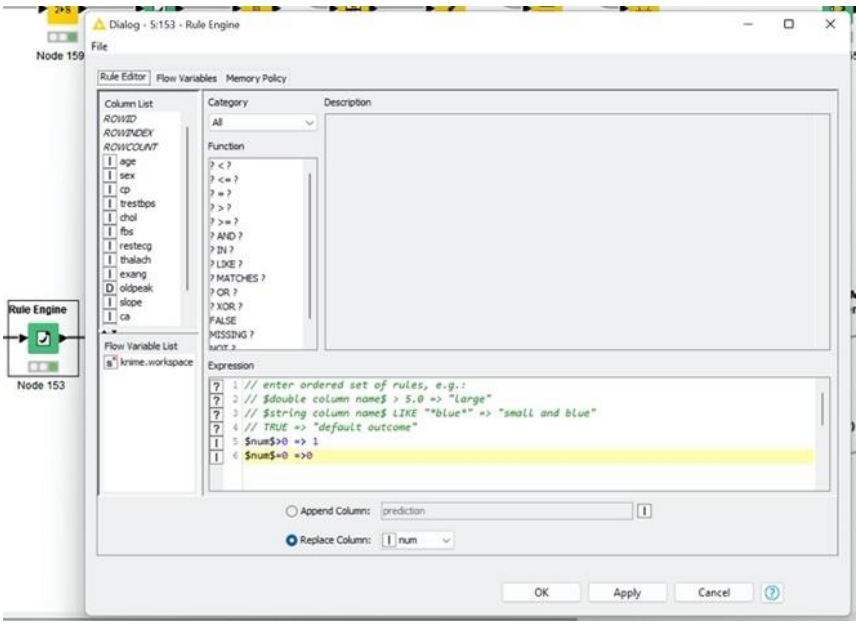


Figura 51. Aplicação do nodo *Rule Engine*.

Com este acréscimo, a matriz de confusão obtida a partir do nodo *Scorer (JAVA Script)* apresentou diferenças relativamente à matriz de confusão do modelo de rede neuronal não binarizado.

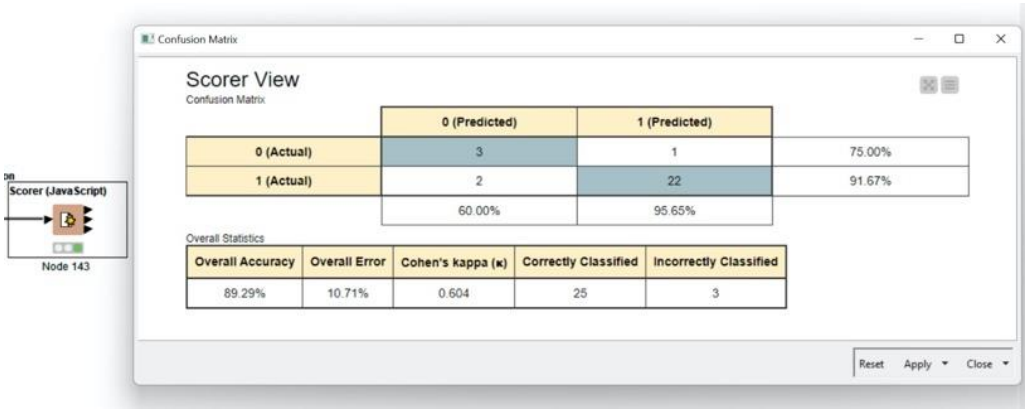


Figura 52. Aplicação do nodo *Scorer (JAVA Script)*.

Tendo em conta o valor obtido, obtivemos agora um modelo satisfatório que, embora tenha limitações, consegue dar respostas confiáveis.

SWITZERLAND

A criação do sistema de aprendizagem automática, através do modelo de redes neuronais apresenta a seguinte estrutura:

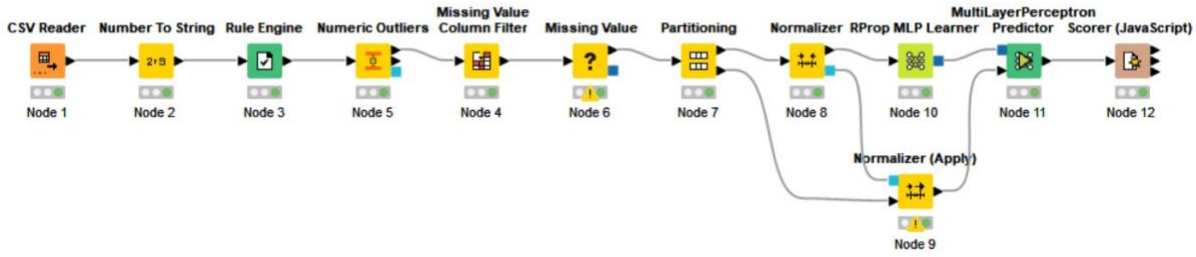


Figura 53. Representação da rede neuronal.

O modelo inicia com a utilização do nodo *CSV Reader*, o qual recebe o ficheiro de *Exel* com a informação e procede à leitura do mesmo.

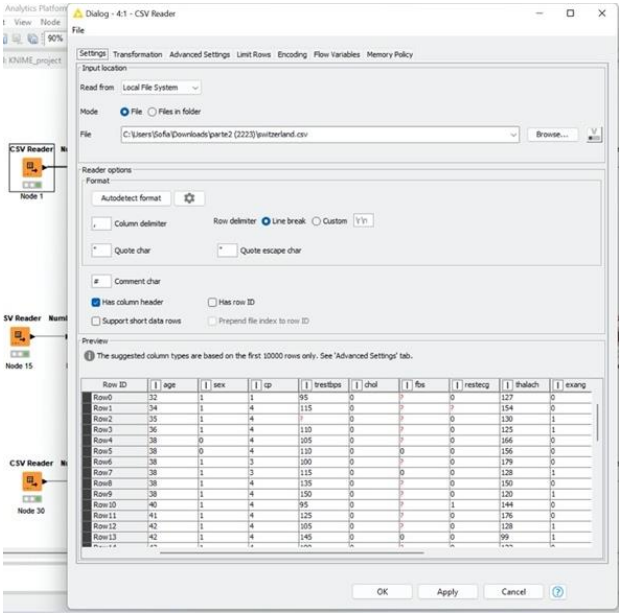


Figura 54. Aplicação do nodo *CSV Reader*.

Seguidamente, recorreu-se ao nodo *Number to String,* para converter o atributo target numa *string*. Tal é necessário para que, o último nodo, *Scorer*, seja capaz de comparar

a coluna *num* com a previsão de *num*. Isto apenas acontecerá se *num*, o nosso *target*, for um atributo de tipo nominal ou string.

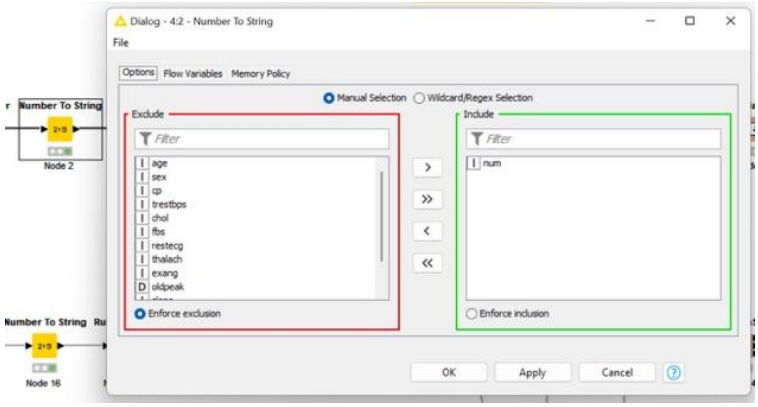


Figura 55. Aplicação do nodo *Number to String*.

O nodo *Rule Engine* permitiu passar o atributo *chol* para binário.

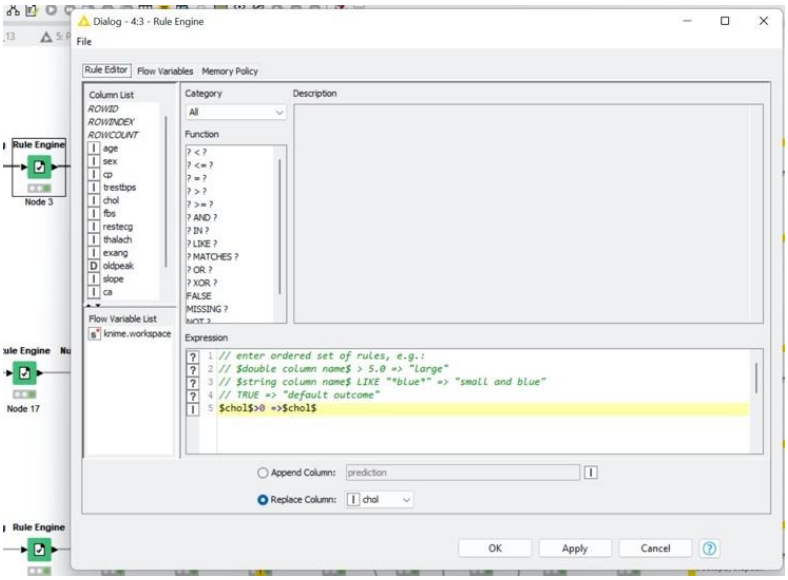


Figura 56. Aplicação do nodo *Rule Engine*.

Para além disso, utilizou-se o nodo *Numeric Outliers*, o qual vai detetar e tratar dos valores discrepantes numéricos presentes nas colunas *tretbps*, *chol*, *restecg*, *thalach*, *oldpeak*, *ca* e *tal.* Esses valores serão considerados *missing values*.

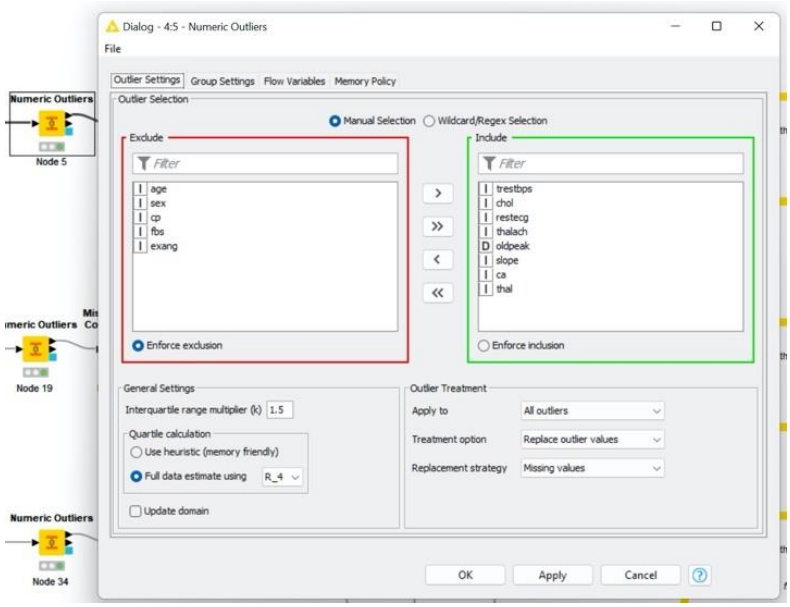


Figura 57. Aplicação do nodo *Numeric Outliers*.

A colocação do nodo *Missing Value Column Filter* permitiu eliminar as colunas de atributos que possuíssem um número considerável de *missing values*.

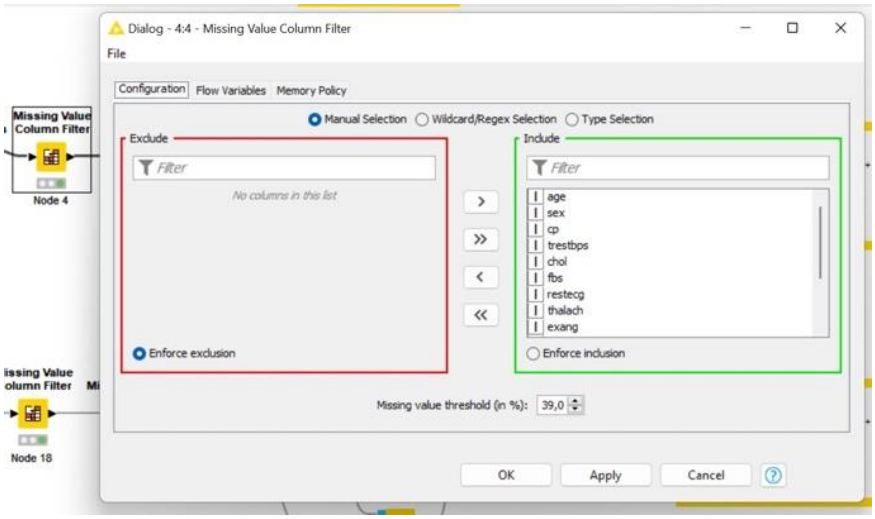


Figura 58. Aplicação do nodo *Missing Value Column Filter*.

Para além disso, procedeu-se à colocação do nodo *Missing Value*, para tratar os *missing values* através da utilização da mediana ou eliminado a coluna onde eles se encontram. A escolha do tratamento varia dependendo da percentagem de *missing values* existentes em cada atributo.

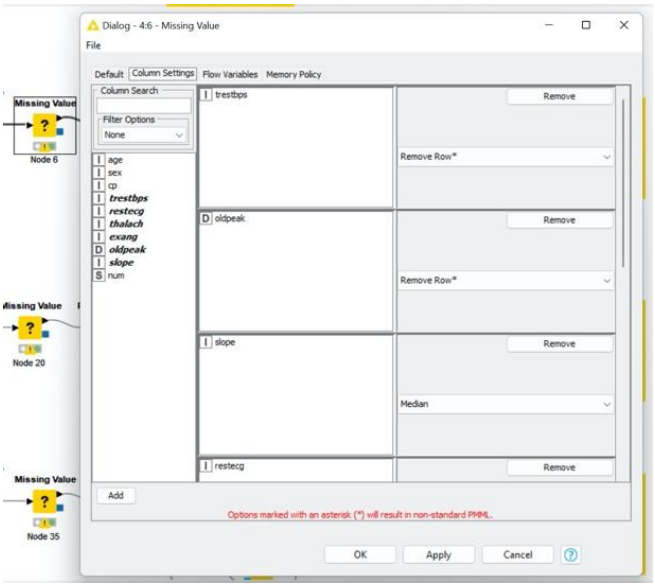


Figura 59. Aplicação do nodo *Missing Value* aos atributos *tretbps*, *oldpeak* e *slope*.

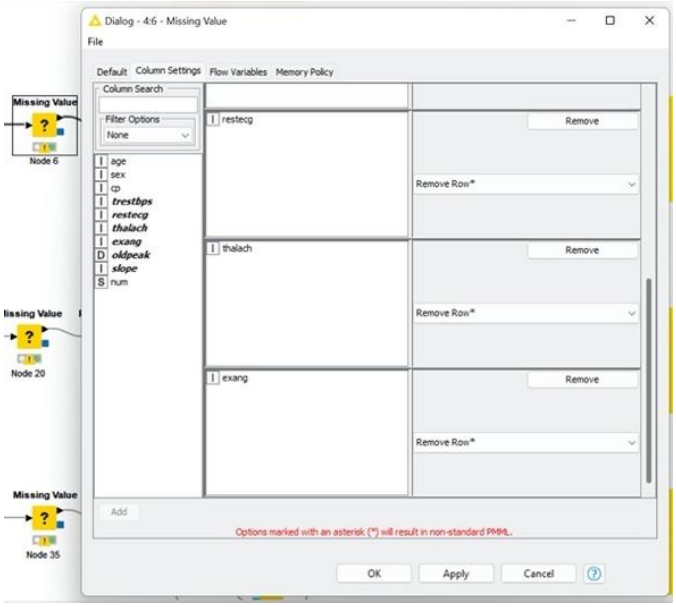


Figura 60. Aplicação do nodo *Missing Value* aos atributos *restecg*, *talach*, *exang*.

De seguida, aplicou-se o nodo *Partitioning,* ao qual se aplicou uma partição de 70%. Utilizou-se, ainda, a opção *random seed* para garantir a repetitividade.

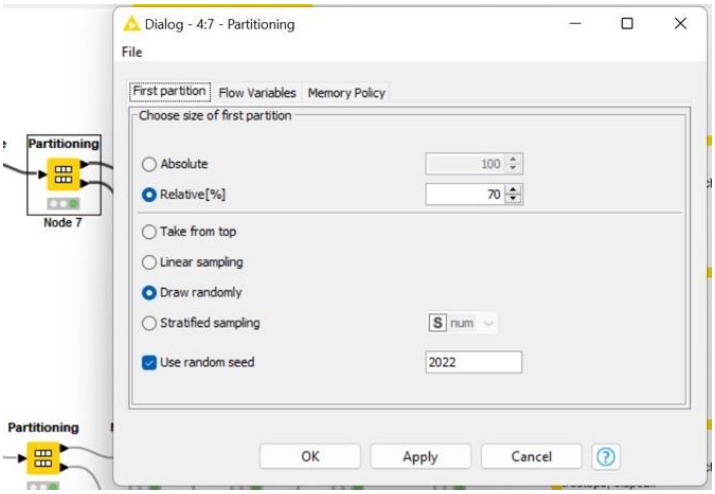


Figura 61. Aplicação do nodo *Partitioning.*

No nodo *Normalizer* fez-se a normalização dos atributos para que todos tenham a mesma ordem de grandeza.

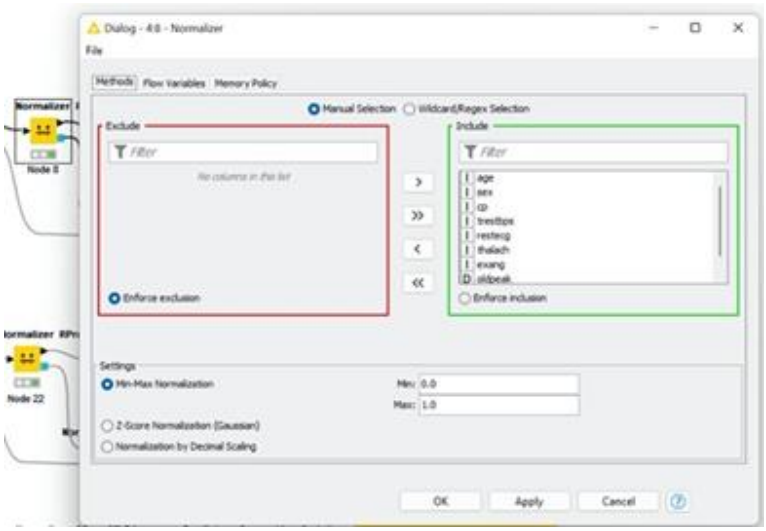


Figura 62. Aplicação do nodo *Normalizer.*

No nodo *Normalizer (Apply)* realizaram-se a normalização dos dados de entrada de acordo com os parâmetros de normalização já fornecidos.

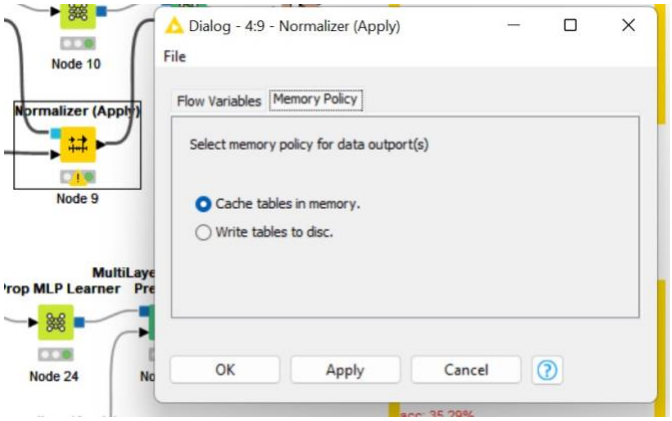


Figura 63. Aplicação do nodo *Normalizer (Apply)*.

De seguida, utilizou-se o nodo *RProp MLP Learner* para identificar o atributo *num* como aquele que se quer prever. Utilizou-se mais uma vez a opção de *random seed*.

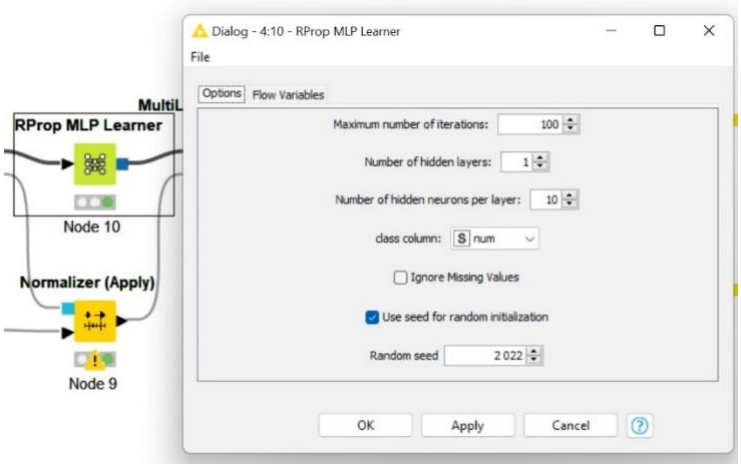


Figura 64. Aplicação do nodo *RProp MLP Learner*.

No nodo *Multilayer Perceptron Predictor*, calculou-se os valores de saída da rede neuronal.

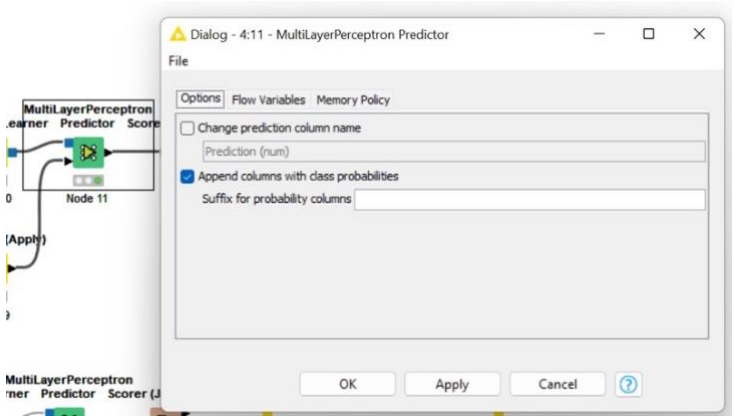


Figura 65. Aplicação do nodo *Multilayer Perceptron Predictor*.

Por último, recorreu-se ao nodo *Scorer (JAVA Script)*, para se observar os valores da matriz de confusão:

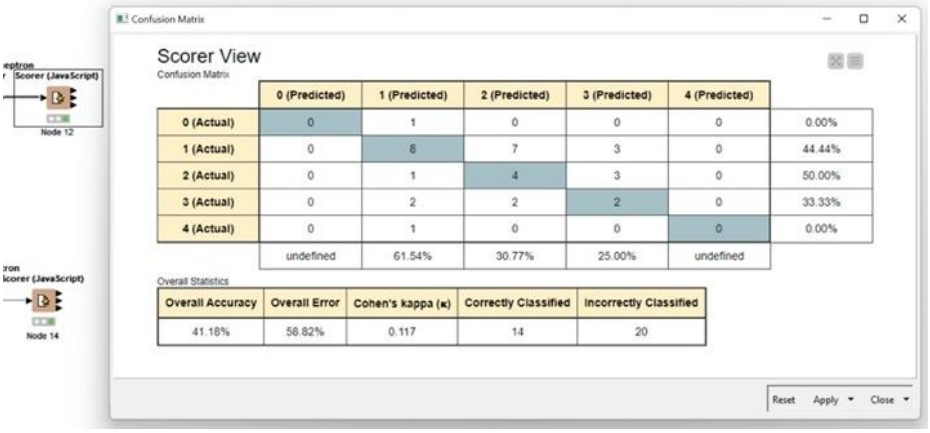


Figura 66. Visualização da matriz de confusão.

Analisando o valor obtido, concluímos mais uma vez que o modelo não terá utilidade prática, uma vez que a *accuracy* é muito reduzida. Neste contexto, ponderou-se proceder à binarização do target, da mesma forma que foi feito para Long Beach. No entanto, tal não faria sentido visto que apenas cerca de 6,5% dos pacientes considerados para o estudo eram saudáveis.

HEART DISEASE

Neste ficheiro procedeu-se à construção de um sistema de aprendizagem, usando para isso um modelo de redes neuronais. Contrariamente aos outros ficheiros, neste sistema de aprendizagem utilizou-se a informação dos 4 locais simultaneamente. O objetivo era testar outras abordagens possíveis para o problema, tentando assim criar uma base de dados mais abrangente.

Dado que num ficheiro temos o diagnóstico da existência ou não de doença numerado com 1 ou 0, respetivamente, enquanto nos outros três a existência de doença é numerada de 1 a 4, o foco deste sistema de aprendizagem é diagnosticar se existe ou não uma doença cardíaca e não o grau em que a doença se encontra, nos casos em que esta existe.

Deste modo, obteve-se o seguinte *workflow*:

Figura 67. Representação da rede neuronal.

Iniciou-se o sistema com quatro nodos *File Reader*, que recebem os ficheiros sobre Cleveland, Hungarian, Long Beach e Switzerland.

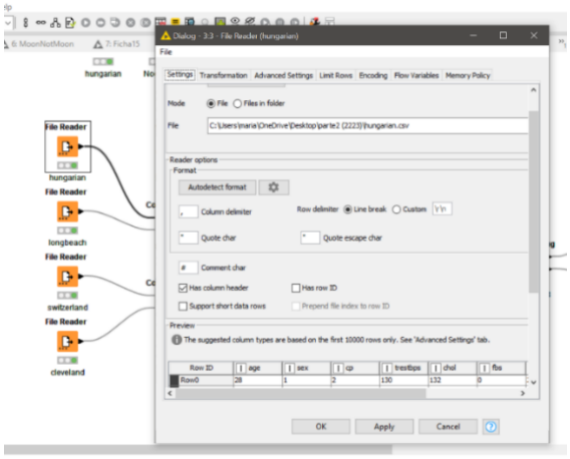


Figura 68. Aplicação do nodo *File Reader*.

Seguidamente, aplicou-se três nodos *Concatenate*, para reunir a informação numa única tabela.

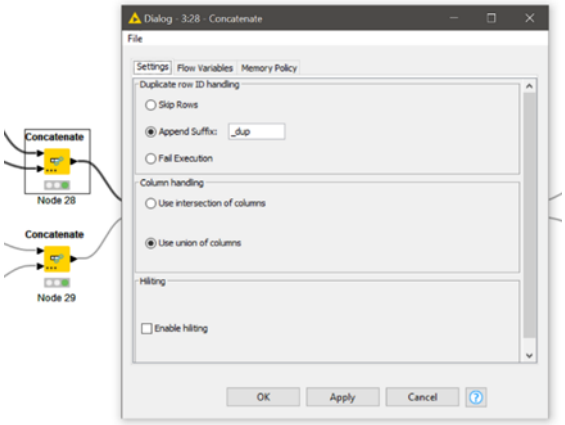


Figura 69. Aplicação do nodo *Concatenate*.

O nodo Rule Engine permitiu passar o atributo target para binário. Esta necessidade adveio do facto de os ficheiros não terem os dados iguais, pelo que para se proceder ao seu tratamento converteu-se tudo para binário. Com isto, considerou-se que era mais importante efetuar um diagnóstico capaz de dizer se existe ou não doença, do que um diagnóstico que atribui um grau para a doença.

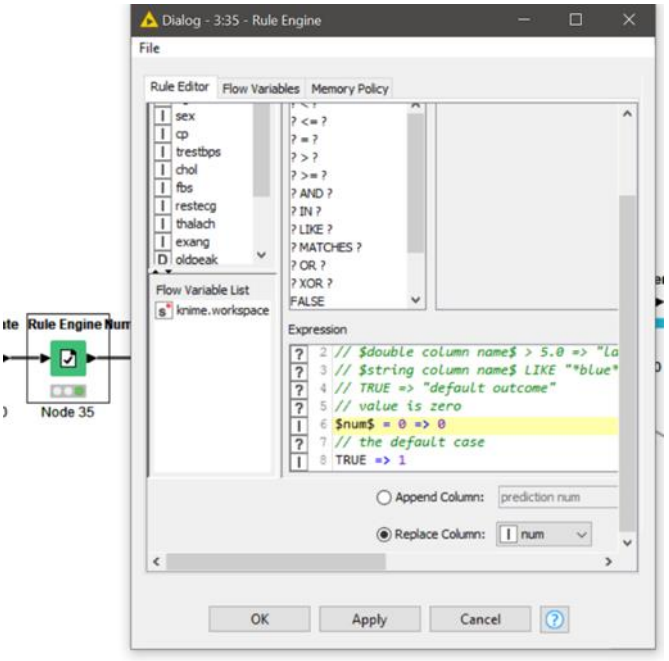


Figura 70. Aplicação do nodo *Rule Engine*.

Por outro lado, inseriu-se o nodo *Number to String*, no qual convertemos o atributo *num* para *string*. Esta etapa é necessária para que, o último nodo, *Scorer*, seja capaz de comparar a coluna *num* com a previsão de *num*, isto apenas acontecerá se *num*, o nosso *target*, for um atributo de tipo nominal ou string.

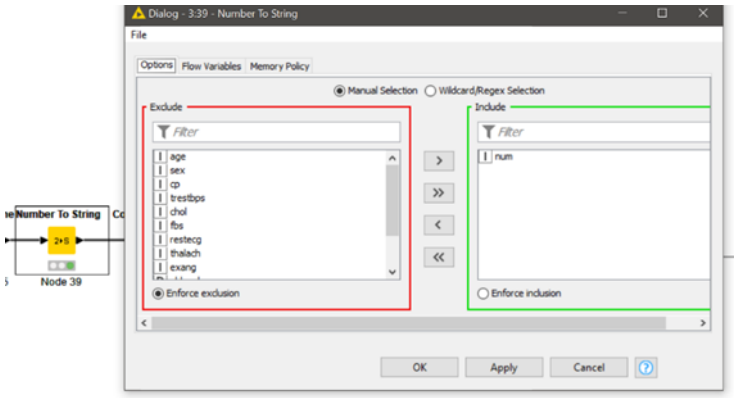


Figura 71. Aplicação do nodo *Number to String*.

Por sua vez, no nodo *Column Rename* efetuou-se a mudança de nome do atributo *num* para *target*, mantendo-o como *string*. As restantes variáveis são mantidas a inteiras, exceto o atributo *oldpeak* que é double uma vez que possui valores decimais.

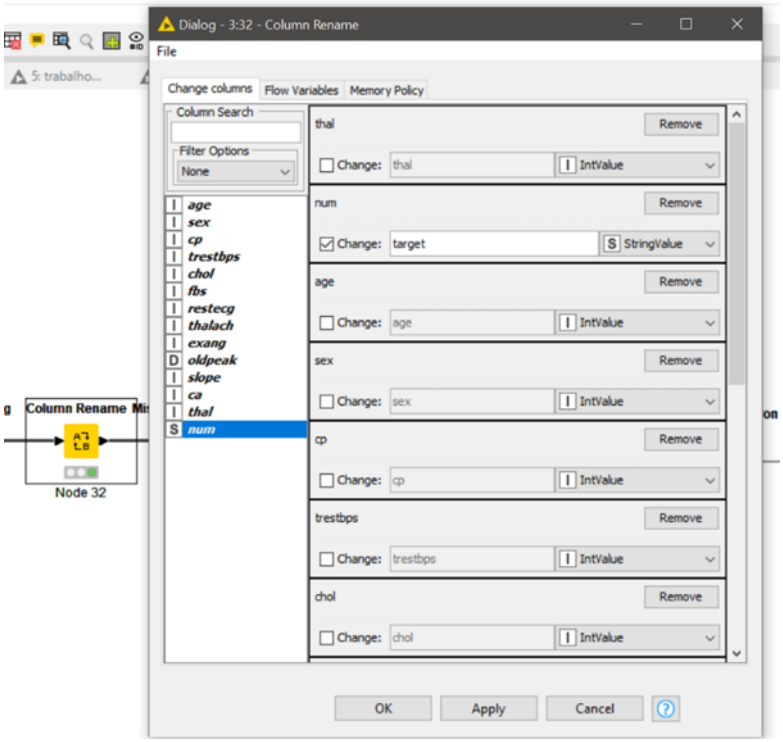


Figura 72. Aplicação do nodo *Column Rename*.

O nodo *Missing Values*, procedeu-se ao tratamento dos *missing values* através da utilização de um valor específico, nomeadamente a média inteira. Optou-se por usar antes esta média ao invés da média, dado que os valores de diagnóstico de doença são números inteiros.

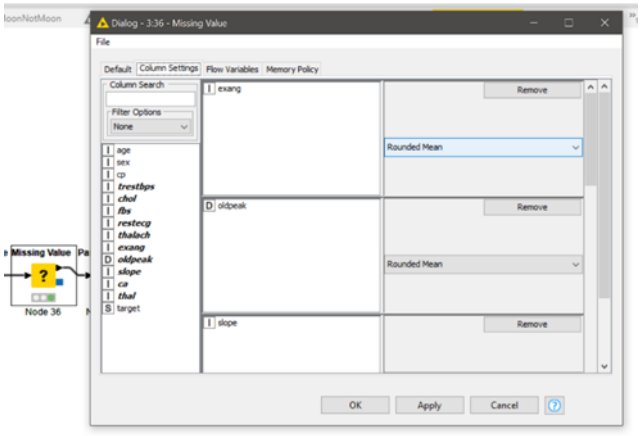


Figura 73. Aplicação do nodo *Missing Values*.

De seguida, recorreu-se ao nodo *Partitioning*, ao qual se aplicou uma partição de 75% e utilizou-se a opção *random seed*. O valor atribuído é um valor aleatório e quando o definimos ele garante a repetitividade.

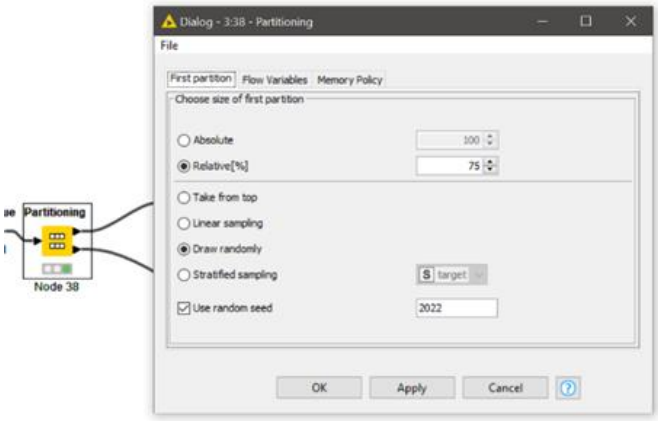


Figura 74. Aplicação do nodo *Partitioning*.

Aplicou-se o nodo *Normalizer* para que este normaliza-se os valores do atributo *oldpeak* pois este possui valores reais, contrariamente aos restantes atributos, e pretende-se que todos os atributos tenham a mesma ordem de grandeza.

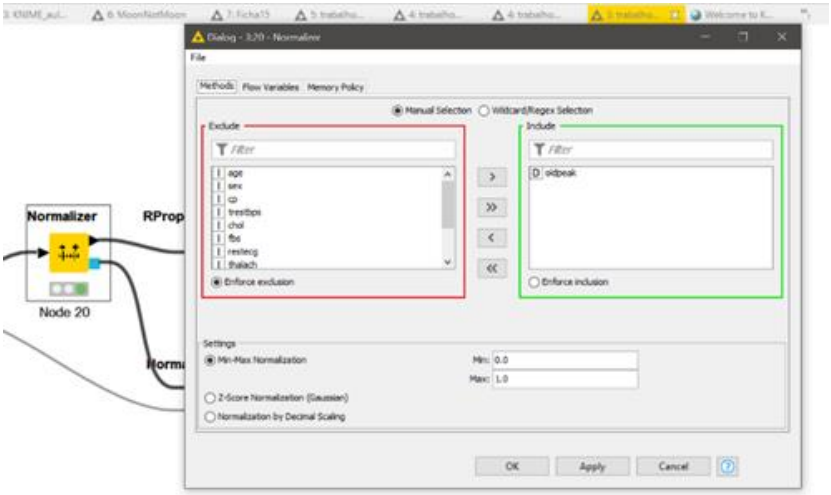


Figura 75. Aplicação do atributo *Normalizer*.

Houve ainda a necessidade de aplicar o nodo *Normalizer (Apply)* para normalizar os dados de entrada de acordo com os parâmetros de normalização já fornecidos.

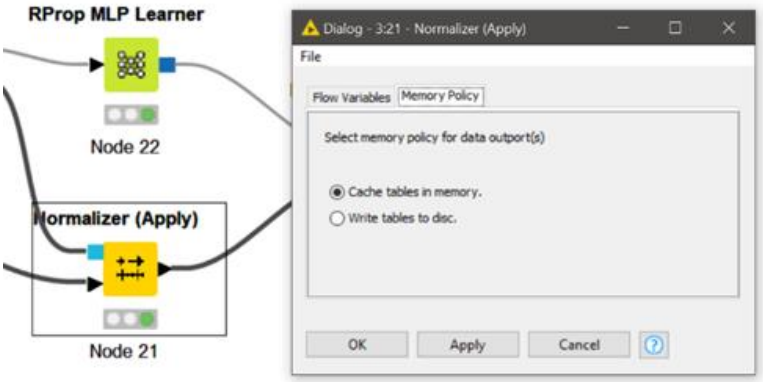


Figura 76. Aplicação do nodo *Normalizer (Apply)*.

Para além disso, utilizou-se o nodo *RProp MLP Learner* para identificar o atributo *target* como aquele que se quer prever. Utilizou-se mais uma vez a opção de *random seed*.

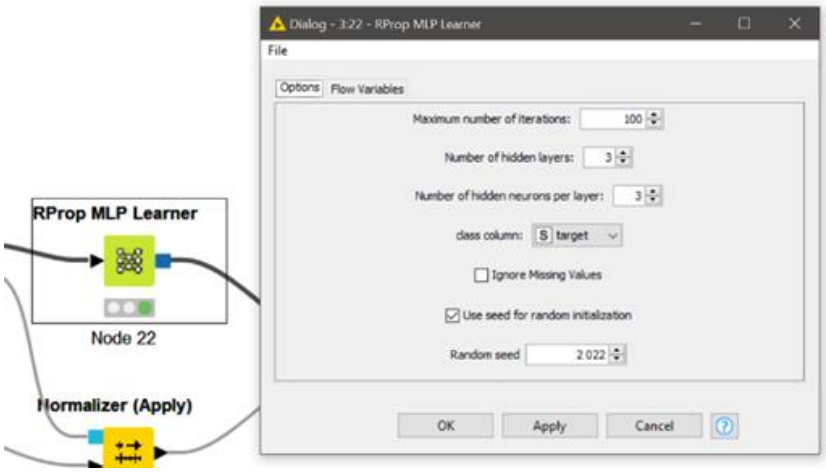


Figura 77. Aplicação do nodo *RProp MLP Learner*.

A utilização do nodo *Multilayer Perceptron Predictor* foi necessária uma vez que é um dos nodos associados ao modelo de redes neuronais. Ele calcula os valores de saída.

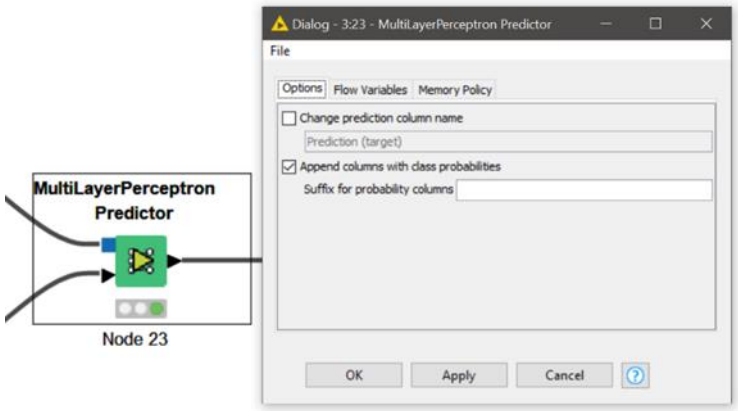


Figura 78. Aplicação do nodo *Multilayer Perceptron Predictor*.

Por último, inseriu-se o nodo *Scorer* para efetuar a comparação entre o atributo target e o atributo *Prediction (target)*, que são as previsões feitas através do modelo de aprendizagem automática.

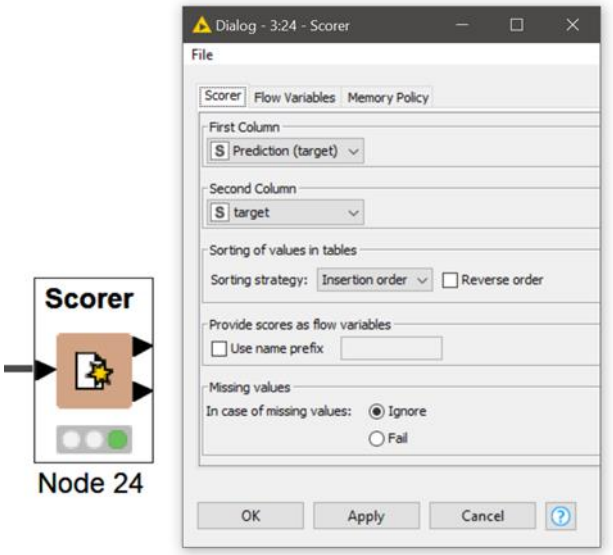


Figura 79. Aplicação do nodo *Scorer*.

Após a aplicação do modelo, foi-se observar o valor obtido para a *accuracy*.

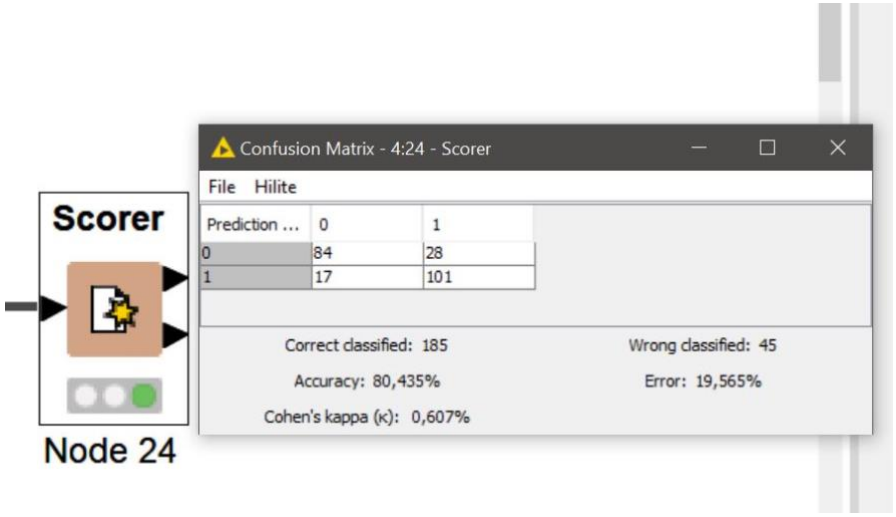


Figura 80. Visualização do valor de *accuracy*.

Posto isto, foi-se testar o ficheiro, mas aplicando um modelo de árvore de decisão. O objetivo é poder comparar no final qual dos modelos permitiu obter melhores resultados.

Assim, o modelo de aprendizagem automática através de árvores de decisão apresenta a seguinte configuração:

Figura 81. Representação da árvore de decisão.

Sendo que a única diferença entre este modelo e o de redes neuronais reside nos nodos *Decision Tree Learner* e *Decision Tree Predictor* que substituem, respetivamente, os nodos *RProp MLP Learner* e *Multilayer Perceptron Predictor* no modelo de redes neuronais, efetuou-se apenas a análise destes dois nodos.

Assim, no nodo *Decision Tree Learner* selecionou-se o atributo *target* para que se possa proceder à construção de uma árvore de decisão de classificação.

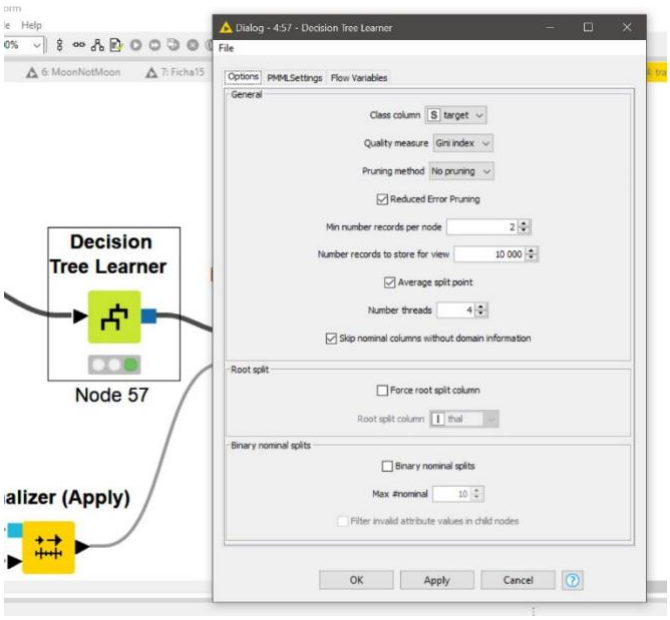


Figura 82. Aplicação do nodo *Decision Tree Learner*.

Já a utilização do nodo *Decision Tree Predictor* foi necessária para se prever o valor da classe, ou seja, do atributo *target* para novos padrões.

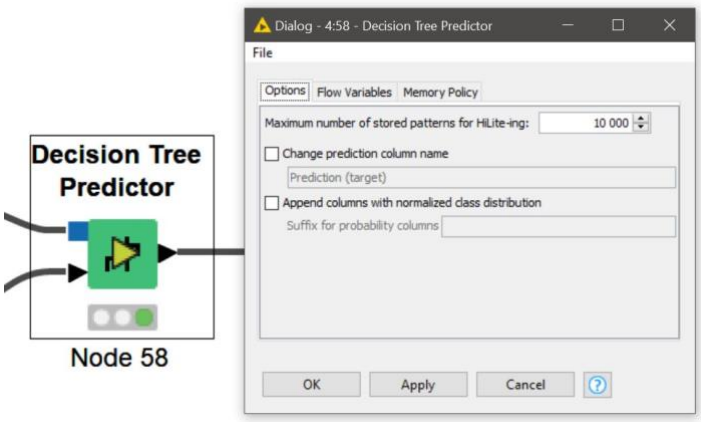


Figura 83. Aplicação do nodo *Decision Tree Predictor*.

Deste modo, observa-se o seguinte valor de *accuracy*:

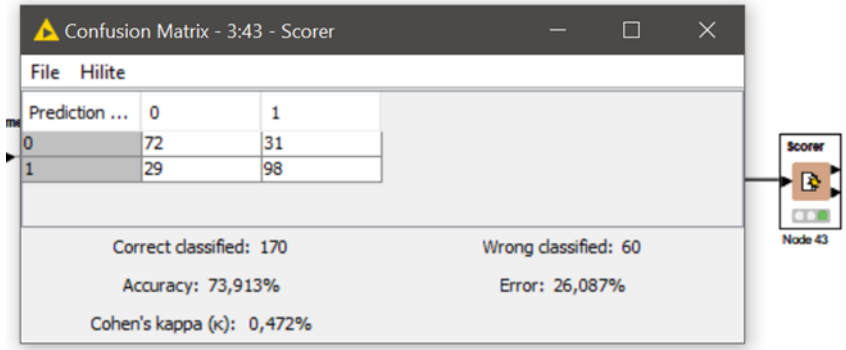


Figura 84. Visualização do valor de *accuracy*.

Conclusões e Sugestões

A quantidade e complexidade da informação implícita e distribuída em diferentes segmentos de dados, tornam complicada a extração de conhecimento que permita apoiar a tomada de decisão. Desta forma, são necessários mecanismos de tratamento automático dos dados, que permitam obter resultados corretos e em tempo útil.

Este trabalho pretendia assim que, a partir de vários ficheiros com dados sobre estudos de doenças cardíacas, se fizesse um tratamento de dados, utilizando vários modelos de aprendizagem, de forma a pôr em prática o conhecimento aprendido num caso real.

Como tal, ao longo do trabalho foram implementados modelos de aprendizagem como redes neuronais e árvores de decisão, bem como diferentes tratamentos de dados, com o intuito de responder aos objetivos solicitados. As redes neuronais possuem características que não se encontram em outros modelos, entre as quais a aprendizagem/generalização, isto é, consegue descrever o todo a partir de algumas partes, constituindo-se como forma de aprendizagem e armazenamento de conhecimento; o processamento maciçamente paralelo, que permite a realização de tarefas completas num curto espaço de tempo; a transparência e a não linearidade. Por outro lado, as árvores de decisão seguem o paradigma *Bottom-Up*, ou seja, toda a informação sobre cada um dos dados está definida numa coleção finita de atributos. Neste modelo, para se saber qual o melhor atributo para ser a raiz da árvore de decisão é necessário ter em consideração a entropia, ou seja, a medida de incerteza associada a um conjunto de objetos. Quanto maior for a capacidade de reduzir a entropia, maior será o ganho de informação.

A utilização da plataforma *KNIME* foi fundamental para se desenvolver o sistema de aprendizagem automática capaz de realizar o diagnóstico ilustrado no estudo.

Fazendo uma avaliação global do trabalho, alguns aspetos poderiam ser melhorados, nomeadamente, as formas de tratar os valores desconhecidos quando estes apareciam em larga escala. Poderia ter-se recorrido à construção de um modelo de *Machine Learning* que previsse como seria o comportamento das *features* amplamente desconhecidas.

Em suma, foi possível aprofundar os conhecimentos associados à aprendizagem automática para análise de dados e à construção de modelos suportados por técnicas de Machine Learning.

Referências

[Saias, J.; Maia, M.; Rato, L.; Gonçalves, T.]

"Machine Learning: um estudo sobre conceitos, tarefas e algoritmos relacionados com predição e recomendação"

Universidade de Évora, 2018

[Pyle, D.]

"Data Preparation for Data Mining"

San Francisco, USA, 1999

[Han, J.; Kamber, M.]

"Data Mining: Concepts and Techniques"

Waltham, USA, 2012

[Quinlan. R.; Kaufmann, M.]

"C4.5 Programs for Machine Learning "

Johns Hopkins University, Baltimore, 1994