



Universidade do Minho
Departamento Produção e Sistemas

DEPARTAMENTO DE PRODUÇÃO E SISTEMAS
Mestrado em Engenharia de Sistemas
Sistemas Baseados em Similaridades

Conceção e implementação de modelos de Machine Learning usando Árvores de Decisão

Coordenador: Paulo Novais
Docentes da UC: Bruno Fernandes

Alunos:

Samuel Costa PG38352
Carolina Silva PG38335
Célia Figueiredo PG41022
Ana Sofia Ferreira PG38356
Márcia Costa A67672

Resumo

A modelação do fluxo de tráfego é um conhecido problema de características estocásticas, não-lineares. Tem, contudo, aparecido na literatura um conjunto de algoritmos que demonstram um potencial assinalável na previsão do fluxo de veículos.

No presente trabalho são explorados dois *datasets*, o primeiro sobre o fluxo de tráfego da cidade do Porto, fornecido pelo docente. O segundo *dataset* utilizado é sobre um abrigo de animais, no qual apenas são acolhidos cães e gatos.

Numa primeira fase do projeto foram planeadas as tarefas tendo em conta o modelo CRISP-DM. De forma a melhor perceber cada um dos *datasets*, analisou-se o seu conteúdo, fazendo uma exploração e descrição de todas as variáveis presentes. Neste fase foram elaboradas matrizes de correlação para perceber de que forma as variáveis se correlacionam.

Depois da exploração inicial é que foi possível fazer o tratamento dos dados, desde a eliminação de nulos, a substituição dos *missing values*, a conversão dos dados, a criação de novas variáveis, a agregação de dados ou a filtragem de colunas.

Finalizado o modelo foi possível obter os resultados que permitiram concluir que tanto o tratamento de dados como a escolha do algoritmo são aspectos cruciais para uma boa previsão.

Para o primeiro *dataset* conclui-se que o melhor resultado um erro de 18,952%, utilizando a medida *Information Gain*, 22 níveis, 8 nodos e 300 modelos. Quanto ao segundo *dataset* conclui-se que o menor erro, 0,131% foi encontrado com a medida *Information Gain*, com 13 níveis, 2 nodos e com 100 modelos.

Conteúdo

| | | |
|----------|---|-----------|
| 1 | Introdução | 2 |
| 1.1 | Contextualização | 2 |
| 1.2 | Objetivos | 2 |
| 1.3 | Estrutura do Relatório | 2 |
| 2 | Algoritmos de Machine Learning | 4 |
| 2.1 | Árvores de Decisão | 4 |
| 2.2 | Indução | 5 |
| 2.2.1 | Algoritmos de indução de árvores de decisão | 5 |
| 2.3 | Medidas de Seleção de Atributos | 5 |
| 2.3.1 | <i>Information Gain</i> | 5 |
| 2.3.2 | <i>Gain Ratio</i> | 6 |
| 2.3.3 | <i>Gini Index</i> | 6 |
| 2.4 | Poda | 7 |
| 2.5 | Técnicas para melhorar a precisão da classificação | 7 |
| 2.5.1 | <i>Random Forest</i> | 7 |
| 3 | Metodologia utilizada | 8 |
| 3.1 | <i>Cross Industry Standart Process for Data Mining (CRISP-DM)</i> | 8 |
| 4 | Trânsito na cidade do Porto | 10 |
| 4.1 | Descrição do <i>dataset</i> | 10 |
| 4.1.1 | Dados do <i>dataset</i> | 10 |
| 4.2 | Análise/ Exploração | 11 |
| 4.3 | Tratamento dos dados | 13 |
| 4.3.1 | Tratamento dos dados relativos à chuva | 13 |
| 4.3.2 | Tratamento dos dados relativos ao <i>timestamp</i> | 13 |
| 4.3.3 | Criação da coluna com velocidade numérica sobre variável resposta | 13 |
| 4.3.4 | Filtragem das colunas | 14 |
| 4.3.5 | Agregação de dados | 14 |
| 4.4 | Modelação | 16 |
| 4.4.1 | Preparação do modelo | 16 |
| 4.4.2 | Criação de <i>binners</i> numéricos | 19 |
| 4.4.3 | <i>Tuning</i> do modelo | 23 |
| 4.4.4 | Feature Selection | 27 |
| 4.5 | Resultados Obtidos | 28 |
| 4.5.1 | Resultados Obtidos utilizando o algoritmo <i>Decision Tree</i> | 28 |
| 4.5.2 | Resultados obtidos utilizando o algoritmo <i>Random Forest</i> | 29 |
| 5 | Animals Shelter | 30 |
| 5.1 | Descrição do <i>dataset</i> | 30 |
| 5.1.1 | Dados do <i>Dataset</i> | 30 |
| 5.2 | Análise/Exploração | 31 |
| 5.3 | Tratamento dos dados | 32 |
| 5.3.1 | Tratamento dos dados relativos ao <i>OutcomeSubType</i> | 32 |

| | | |
|----------|---|-----------|
| 5.3.2 | Tratamento dos dados relativos à cor | 33 |
| 5.3.3 | Tratamento dos dados relativos à raça | 33 |
| 5.3.4 | Tratamento dos dados relativos à idade | 33 |
| 5.3.5 | Tratamento dos dados relativos à variável <i>DateTime</i> | 33 |
| 5.3.6 | Filtragem de colunas | 33 |
| 5.4 | Modelação | 34 |
| 5.4.1 | Preparação do modelo | 34 |
| 5.4.2 | <i>Tuning</i> do modelo | 36 |
| 5.5 | Resultados Obtidos | 37 |
| 6 | Conclusões | 38 |

Listas de Figuras

| | | |
|------|---|----|
| 2.1 | Árvore de decisão que representa compra computador, fazendo a indicação se é provável que um dado cliente de uma dada empresa compre um computador. Cada nó interno representa um teste de um atributo. Cada folha representa uma classe (compra do computador = sim ou compra do computador = não) (1) | 4 |
| 3.1 | Representação da metodologia CRISP-DM (2) | 8 |
| 4.1 | Matriz de correlação | 11 |
| 4.2 | Desvio-padrão das variáveis do <i>dataset</i> | 12 |
| 4.3 | Medidas do <i>box-plot</i> referente à variável <i>Average_Time_Diff</i> | 12 |
| 4.4 | Gráfico de barras que permite identificar a importância dos dias da semana no modelo | 12 |
| 4.5 | Nodo responsável pela substituição da string "NULL" pela string "sem chuva", no KNIME | 16 |
| 4.6 | Nodo responsável pela substituição de todos os <i>missing values</i> (?) pela string "sem chuva", no KNIME | 17 |
| 4.7 | Nodo responsável pela substituição da string "chuva de intensidade pesado" pela string "chuva de intensidade pesada", no KNIME | 17 |
| 4.8 | Nodo responsável pela conversão da data do registo a uma string, no KNIME | 18 |
| 4.9 | Nodo responsável pela extração dos campos mês, dia da semana e hora da data do registo, no KNIME | 18 |
| 4.10 | Nodo responsável pela criação do biner numérico, "AVERAGE_TIME_DIFF_binned" no KNIME | 19 |
| 4.11 | Nodo responsável pela criação do biner numérico "Hour_binned" no KNIME | 19 |
| 4.12 | Nodos responsáveis pela criação do dicionário referente à designação do dia da semana, no KNIME | 20 |
| 4.13 | Nodos responsáveis pelo cálculo da variável em estudo | 20 |
| 4.14 | Fórmula responsável pelo o cálculo da distância, em média, do conjunto de ruas em estudo no KNIME | 20 |
| 4.15 | Fórmula responsável pelo o cálculo do tempo efetivo (em horas) que demora a percorrer o conjunto de ruas em estudo no KNIME | 21 |
| 4.16 | Fórmula responsável pelo o cálculo da velocidade média efetiva (km/h) de cada registo KNIME | 21 |
| 4.17 | Fórmula responsável pelo o cálculo da velocidade média efetiva (km/h) de cada registo KNIME | 21 |
| 4.18 | Colunas filtradas no KNIME | 22 |
| 4.19 | Worflow final da preparação dos dados do <i>dataset</i> em KNIME | 22 |
| 4.20 | Critérios de divisão especificados no nodo <i>Table Creator</i> | 23 |
| 4.21 | Definições de configuração do nodo <i>Parameter Optimization Loop Start</i> . | 24 |
| 4.22 | Configuração do nodo <i>X-Partitioner</i> | 24 |
| 4.23 | Caixa de diálogo de configuração do nodo <i>Parameter Optimization Loop End</i> . Otimizaram-se os parâmetros minimizando o valor na variável de fluxo chamada "Erro médio em %". | 25 |
| 4.24 | Tuning do modelo usando <i>Random Forest</i> | 25 |
| 4.25 | <i>Table Creator</i> com definição das medidas de qualidade, método de poda e uso ou não de <i>Reduced Error Pruning</i> | 26 |
| 4.26 | Parâmetros utilizados para encontrar a melhor solução | 26 |
| 4.27 | Modelo final para calcular os melhores resultados de <i>pruning</i> utilizando o algoritmo <i>Decision Tree</i> | 27 |
| 4.28 | Configuração do nodo <i>Feature Selection Loop Start</i> | 27 |
| 4.29 | Resultados obtidos com o <i>Feature Selection</i> | 28 |
| 4.30 | Modelo construído para seleção de colunas com o método <i>Foward Feature Selection</i> | 28 |
| 5.1 | Matriz de correlação | 31 |
| 5.2 | Gráfico variável <i>OutcomeType</i> | 32 |

| | | |
|------|--|----|
| 5.3 | Gráfico sobre as idades | 32 |
| 5.4 | <i>Table Creator</i> com a definição de novas cores | 34 |
| 5.5 | Substituição de valores em falta e concatenação | 34 |
| 5.6 | Substituição de <i>missing values</i> pela string "Found" | 35 |
| 5.7 | Exclusão de valores cuja idade é zero o valor em falta | 35 |
| 5.8 | Criação de uma nova variável que permitiu diminuir a quantidade de raças | 35 |
| 5.9 | Seleção das colunas relevantes para previsão | 36 |
| 5.10 | Workflow final da preparação dos dados do <i>dataset</i> , em <i>KNIME</i> | 36 |

Listas de Tabelas

| | | |
|-----|--|----|
| 4.1 | Tabela explicativa dos períodos de trânsito | 15 |
| 4.2 | Tabela explicativa dos períodos de AVERAGE_TIME_DIFF_binned | 16 |
| 4.3 | Valores utilizados para otimização de parâmetros | 23 |
| 4.4 | Resultados obtidos com o algoritmo <i>Decision Tree</i> | 28 |
| 4.5 | Resultados obtidos com o algoritmo <i>Random Forest</i> | 29 |
| 5.1 | Tabela <i>OutcomeType</i> | 30 |
| 5.2 | Tabela <i>OutcomeSubType</i> | 30 |
| 5.3 | Tabela SexuponOutcome | 31 |
| 5.4 | Tabela com as novas cores | 33 |
| 5.5 | Resultados obtidos com o algoritmo <i>Decision Tree</i> e sem tratamento dos dados | 37 |
| 5.6 | Resultados obtidos com o algoritmo <i>Random Forest</i> e sem tratamento dos dados | 37 |
| 5.7 | Resultados obtidos com o algoritmo <i>Random Forest</i> e com tratamento dos dados | 37 |

Lista de siglas

AHP *Analytic Hierarchy Process*

CART *Classification and Regression Trees*

CRISP-DM *Cross Industry Standart Process for Data Mining*

DM *Data Mining*

ID3 *Iterative Dichotomiser 3*

MDL *Minimum Description Length*

1. Introdução

1.1 Contextualização

O presente trabalho foi realizado no âmbito da Unidade Curricular de Sistemas Baseados em Similaridade do Mestrado em Engenharia de Sistemas da Universidade do Minho.

A aplicação de novas tecnologias tornaram-se fundamentais para o funcionamento de organizações dos mais variados ramos do mercado. Dada essa realidade, extrair e analisar dados passou a ser facilitado devido ao *Data Mining*.

Assim sendo, *Data Mining* permite que organizações analisem todo o conteúdo dos seus dados e compreendam o que é relevante, para posteriormente aproveitar as informações úteis e avaliar os possíveis resultados. Uma vez que os dados estão reunidos através de um processo automatizado, eles podem ser analisados e modelados para serem convertidos em estratégias de negócio.

No presente trabalho vão ser utilizados dois *datasets*, um sobre o fluxo de tráfego na cidade do Porto e outro sobre um abrigo de animais. Em geral, o tráfego numa cidade é afetado por diversos fatores, desde as condições meteorológicas até à hora do dia em que se está a medir, uma vez que há horas nas quais o fluxo é bastante mais elevado.

Relativamente ao abrigo dos animais, nele são tomadas várias decisões relacionadas com os animais que albergam. Chegam ao abrigo todo o tipo de casos, desde animais que se perderam dos donos aos que foram abandonados. Os animais tanto podem ser saudáveis, como estarem com algum problema de saúde, por isso o abrigo é também responsável por tomar as decisões relativas à saúde dos animais.

Através da plataforma KNIME será construído um modelo para cada *dataset* com o objetivo de conseguir prever tanto o fluxo de trânsito como a decisão sobre o futuro do animal aliado à minimização do erro de previsão.

1.2 Objetivos

O trabalho tem como objetivo utilizar dois *datasets* distintos para explorar, analisar e preparar os dados de cada um deles. Posteriormente, pretende-se extrair conhecimento desses dados.

Para esse efeito serão aplicados modelos de aprendizagem baseado em árvores de decisão, utilizando a plataforma KNIME. O projeto procura desenvolver um modelo capaz de prever o fluxo de tráfego num dado ponto temporal na cidade do Porto.

Para o segundo *dataset* o objetivo será criar um modelo capaz de prever a adoção de animais de um abrigo, tendo em conta as suas características.

Ambos os *datasets* têm um objetivo fundamental, que consiste em construir um modelo capaz de fazer previsões de forma a maximizar a precisão do modelo, ou seja, com o mínimo erro de previsão possível.

1.3 Estrutura do Relatório

Este projeto está dividido em seis capítulos e começa com o presente capítulo que corresponde à introdução do tema, no qual é feito um breve enquadramento e são definidos os objetivos e a metodologia de investigação utilizada.

O segundo capítulo é realizada uma breve revisão de literatura sobre certos conceitos da área de *Machine Learning* que são importantes para o desenvolvimento do projeto.

O terceiro capítulo aborda a metodologia que foi utilizada no decorrer do projeto.

O quarto capítulo aborda o processo de construção do modelo de *machine learning* utilizando árvores de decisão do *dataset* "Trânsito na Cidade do Porto" e os resultados obtidos da sua construção.

O quinto capítulo aborda o processo de construção do modelo de *machine learning* utilizando árvores de decisão do dataset "*Animals Shelter*" e os resultados obtidos.

No sexto e último capítulo apresentam-se as conclusões do projeto elaborado, assim como sugestões de trabalhos futuros.

2. Algoritmos de *Machine Learning*

O *machine learning* é um método para análise de dados que automatiza a construção de modelos analíticos. Este insere-se na área da inteligência artificial, sendo ele um ramo baseado na ideia de que os sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana.

Um dos métodos de *machine learning* mais conhecidos é o método de aprendizagem supervisionada, em que este é treinado por meio de exemplos rotulados, como uma entrada na qual a saída desejada é conhecida. Por exemplo, um equipamento poderia ter *data points* com rótulos de “F” (falha) ou “E” (executa). Algoritmos deste tipo recebem um conjunto de entradas junto com as saídas corretas correspondentes, e aprendem ao comparar a saída real com as saídas corretas para encontrar erros.

2.1 Árvores de Decisão

A Árvore de Decisão é um algoritmo que tem por base uma estrutura em árvore, baseando-se na decisão por teste, ou seja, qual teste realizar em cada nodo. A complexidade deste algoritmo aumenta de acordo com a quantidade de nós e do número de ramos da árvore. Este algoritmo de *machine learning* está envolvido nos modelos de aprendizagem e é, fundamentalmente, usado na técnica.

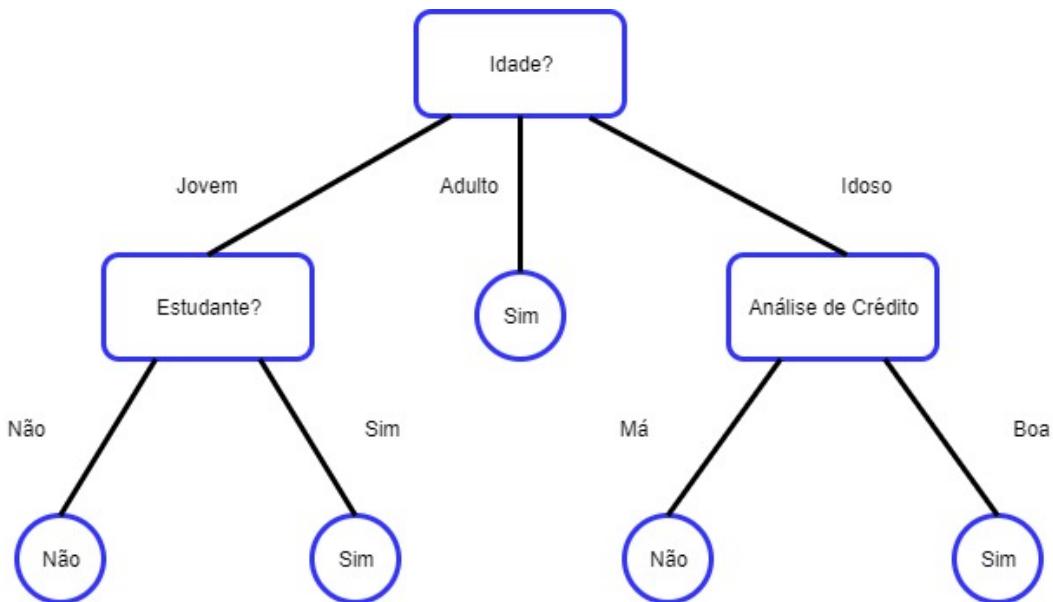


Figura 2.1: Árvore de decisão que representa compra computador, fazendo a indicação se é provável que um dado cliente de uma dada empresa compre um computador. Cada nó interno representa um teste de um atributo. Cada folha representa uma classe (compra do computador = sim ou compra do computador = não) (1)

Os modelos de Árvore de Decisão são criados utilizando duas etapas. A primeira denomina-se por Indução, na qual se procede à construção da árvore, definindo-se os limites hierárquicos de decisão com base nos dados de um *dataset*. A segunda fase é a poda que consiste no processo de remover a estrutura desnecessária da árvore. Desta forma, é reduzida a sua complexidade combatendo assim o *overfitting*, tendo ainda a vantagem de haver uma interpretação da árvore mais facilitada (*site towards data science*).

2.2 Indução

Começando do nível mais alto, a indução da árvore de decisão passa por 4 etapas principais para construir a árvore (1):

- 1. Começar com o conjunto de dados de treino, que deve ter algumas variáveis de recurso e saída de classificação ou regressão.
- 2. Determinar o recurso mais adequado no conjunto de dados para dividi-los;
- 3. Dividir os dados em subconjuntos que contêm os valores possíveis para esse melhor recurso. Essa divisão basicamente define um nó na árvore, ou seja, cada nó é um ponto de divisão com base num determinado recurso dos dados.
- 4. Gerir recursivamente novos nós da árvore usando o subconjunto de dados criado a partir da etapa 3. Continua-se a dividir até chegar a um ponto em que optimiza a precisão, minimizando o número de divisões.

2.2.1 Algoritmos de indução de árvores de decisão

2.2.1.1 Iterative Dichotomiser 3 (ID3)

Neste algoritmo, um subconjunto do conjunto de treino é escolhido aleatoriamente e uma árvore de decisão é criada a partir dele. Os restantes objetos do conjunto de treino são classificados usando a árvore. Se a árvore fornecer a resposta correta para todos esses objetos, então estará correta para todo o conjunto de treino e o processo será encerrado. Caso contrário, uma seleção dos objetos classificados incorretamente é adicionado ao subconjunto, e o processo continua. A principal limitação do ID3 é o facto de apenas lidar com atributos categóricos não-ordinais, não sendo possível apresentar a este algoritmo conjuntos de dados com atributos contínuos, por exemplo (4).

2.2.1.2 C4.5

O C4.5 é um dos algoritmos mais utilizados, por apresentar ótimos resultados em problemas de classificação.

Este é uma evolução do algoritmo ID3. Lida tanto com atributos categóricos (ordinais ou não-ordinais) como com atributos contínuos.

Este converte as árvores treinadas (isto é, a saída do algoritmo ID3) em conjuntos de regras *if-then* (1).

2.2.1.3 CART

O algoritmo *Classification and Regression Trees* (CART) consiste numa técnica não-paramétrica que induz tanto árvores de classificação como árvores de regressão, dependendo se o atributo é nominal (classificação) ou contínuo (regressão) (1).

2.3 Medidas de Seleção de Atributos

Uma medida de seleção de atributo é uma heurística para selecionar o critério de divisão que “melhor” separa uma determinada partição de dados, D, de instâncias de treino classificadas individualmente.

Se dividir D em partições menores de acordo com os resultados do critério de divisão, idealmente, cada partição seria pura, ou seja, todas as instâncias que acabam numa determinada partição pertenceriam à mesma classe (1).

As medidas de seleção de atributos também podem ser denominadas como regras de divisão porque determinam como as instâncias num dado nó devem ser divididas.

2.3.1 Information Gain

Para avaliar o quanto bom é um recurso para a divisão, é calculada a diferença na entropia (conceito usado para medir informações ou distúrbios ou para medir a pureza de um conjunto de dados) antes e depois da divisão. Ou seja, primeiro calcula-se a entropia do conjunto de dados antes da divisão e depois calcula-se a entropia para cada subconjunto após a divisão. Finalmente, a soma das entropias de saída ponderadas pelo tamanho dos subconjuntos é subtraída da entropia do conjunto de dados antes da divisão. Essa diferença mede o ganho de informação ou a

redução na entropia. Se o ganho de informação for um número positivo, isso significa que passamos de um conjunto de dados confuso para um número de subconjuntos mais puros.

$$GanhoDeInformacao = Entropia(\text{antes}) - \sum_{j=1}^K Entropia(j, \text{depois}) \quad (2.1)$$

Na equação, “antes” é o conjunto de dados antes da divisão, K é o número de subconjuntos gerados pela divisão e (j, depois) é o subconjunto j após a divisão. Em cada etapa, optar-se-ia por dividir os dados no recurso com o maior valor em ganho de informações, pois isso leva aos subconjuntos mais puros. O algoritmo que aplica essa medida é o algoritmo ID3. Este tem a desvantagem de favorecer recursos com um número maior de valores, gerando árvores de decisão maiores (3).

2.3.2 Gain Ratio

A medida *gain ratio*, usada no algoritmo C4.5, introduz o conceito *SplitInfo*. Este é definido como a soma dos pesos multiplicados pelo logaritmo dos pesos, em que os pesos são a proporção do número de pontos de dados no subconjunto atual em relação ao número de pontos de dados no conjunto de dados pai. A *gain ratio* é, então, calculada dividindo o ganho de informação do algoritmo ID3 pelo valor *SplitInfo*.

$$RacioDeGanho = \frac{GanhoDeInformacao}{SplitInfo} = \frac{Entropia(\text{antes}) - \sum_{j=1}^K Entropia(j, \text{depois})}{\sum_{j=1}^K w_j \log_2 w_j} \quad (2.2)$$

$$w_j = \frac{\text{amostraDoSubset}(j, \text{depois})}{\text{amostraDoDataSet}(\text{antes})} \quad (2.3)$$

Na equação, “antes” é o conjunto de dados antes da divisão, K é o número de subconjuntos gerados pela divisão e (j, depois) é o subconjunto j após a divisão (3).

2.3.3 Gini Index

A medida usada pelo algoritmo CART é *Gini index*, que é baseado na impureza de Gini. A impureza de Gini é definida como 1 menos a soma dos quadrados das probabilidades de classe num conjunto de dados.

$$ImpurezaDeGini = 1 - \sum_{i=1}^N p_i^2 \quad (2.4)$$

Na equação acima, p é o conjunto de dados completo, N é o número de classes e pi é a frequência da classe i no mesmo conjunto de dados.

O *Gini index* é definido como a soma ponderada da impureza de *Gini* dos diferentes subconjuntos após uma divisão, em que cada parte é ponderada pela proporção do tamanho do subconjunto em relação ao tamanho do conjunto de dados pai.

Para um conjunto de dados com duas classes, o intervalo do *Gini index* está entre 0 e 0,5: 0 se o conjunto de dados for puro e 0,5 se as duas classes forem distribuídas igualmente. Assim, o recurso com o menor *Gini index* é usado como o próximo recurso de divisão.

$$GiniIndex = w_j \sum_{j=1}^K ImpurezaDeGini(j, \text{depois}) \quad (2.5)$$

$$w_j = \frac{\text{amostraDoSubset}(j, \text{depois})}{\text{amostraDoDataSet}(\text{antes})} \quad (2.6)$$

Nas equações anteriores, K é o número de subconjuntos gerados pela divisão e (j, depois) é o subconjunto j após a divisão (3).

2.4 Poda

Quando construída uma árvore de decisão, poderão existir muitas ramificações que refletem anomalias nos dados de treino devido a ruídos ou *outliers*. Os métodos de remoção de árvores resolvem esse problema de *overfitting* dos dados.

As árvores podadas tendem a ser menores e menos complexas e, por isso, mais fáceis de interpretar. Para além disso, geralmente estas são mais rápidas e melhores na classificação correta de dados de teste independentes do que em árvores não podadas.

Existem duas abordagens relativamente à poda de uma árvore de decisão denominadas por pré-poda e pós-poda.

Na abordagem de pré-poda é realizado durante o processo de construção da árvore, em que o processo pode simplesmente parar de dividir o conjunto de elementos e transformar o nó corrente num nó folha da árvore.

A segunda, e mais comum abordagem, é a poda, que é realizada após a construção da árvore de decisão, removendo ramos completos, onde tudo que está abaixo de um nó interno é excluído e esse nó é transformado em folha, representando a classe mais frequente no ramo (1).

2.5 Técnicas para melhorar a precisão da classificação

Nesta parte, serão abordados algumas maneiras para aumentar a precisão da classificação. O foco será nos *ensemble models* (métodos de conjunto). A aprendizagem por conjuntos, em geral, é um modelo que faz previsões com base em vários modelos diferentes. Ao combinar modelos individuais, o modelo de conjunto tende a ser mais flexível e menos sensível a dados (menos variação).

Alguns dos métodos mais populares são o *bagging* que pressupõe o treino de uma série de modelos individuais de forma paralela, sendo que cada modelo é treinado por um subconjunto aleatório dos dados; existem também o *boosting* que consiste no treino de vários modelos individuais de maneira sequencial, em que cada modelo individual aprende com os erros cometidos pelo modelo anterior; e por fim as *random forests* que serão abordadas na secção posterior (1).

2.5.1 Random Forest

O *Random Forest* consiste num grande número de árvores de decisão individuais que funcionam como um conjunto. Cada árvore individual lança uma previsão de classe e a classe com mais votos torna-se a previsão do modelo.

A razão deste modelo apresentar um bom funcionamento é pelo facto deste possuir um grande número de árvores relativamente não correlacionadas que operam como um conjunto. Desta forma, o seu funcionamento irá superar qualquer um dos modelos constituintes individuais. Este método obedece a uma série de etapas, sendo elas as seguintes:

- Etapa 1: Selecionar n subconjuntos aleatórios no conjunto de treino;
- Etapa 2: Treinar n árvores de decisão
 - um subconjunto aleatório é usado para treinar uma árvore de decisão;
 - as divisões ideais para cada árvore de decisão são baseadas por subconjunto aleatório de recursos;
- Etapa 3: Cada árvore individual prevê no conjunto de treino de forma independentemente;
- Etapa 4: Fazer a previsão final.

3. Metodologia utilizada

O *data mining* não consiste em apenas consultas a bases de dados ou conjunto de algoritmos como árvores de decisão, redes neurais, métodos estatísticos, entre outros. A exploração de dados é um processo ou uma metodologia para a aquisição de conhecimento que pode envolver todas os conceitos anteriores.

3.1 CRISP-DM

A metodologia foi criada nos anos 90, devido à necessidade dos profissionais de *Data Mining*. Apesar de haver várias ferramentas capazes de ajudar esses profissionais, quando o assunto é *Big Data* e o seu grande volume de dados, essas ferramentas não são tão eficazes.

O CRISP-DM surgiu justamente para atender aos projetos que estão diretamente envolvidos com o processamento e a análise de um grande volume de dados. A metodologia CRISP-DM reúne as melhores práticas para que o *Data Mining* seja o mais eficiente possível, fazendo uma análise dos dados, para propor modelos de melhoria ou solução de problemas (2).

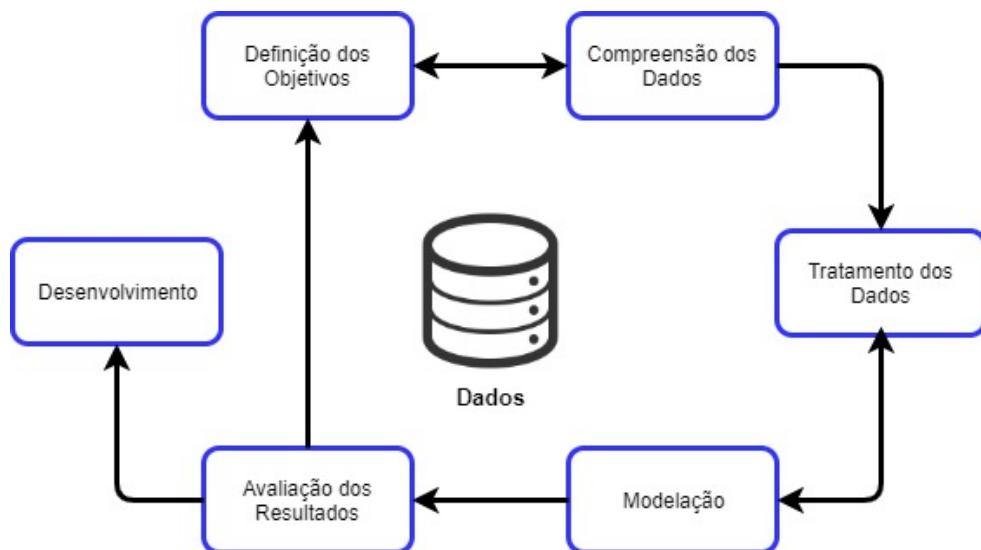


Figura 3.1: Representação da metodologia CRISP-DM (2)

Esta é composta por seis fases, cada uma das quais contém um conjunto de tarefas específicas e com resultados esperados bem definidos. As fases que compõem a metodologia são:

- **Definição dos objetivos:** Procura-se entender qual o problema a ser resolvido, tentando procurar todos os detalhes sobre o impacto dele numa dada organização e quais os objetivos em relação ao trabalho.
- **Compreensão dos dados:** É efetuada uma análise aprofundada quanto à qualidade dos dados, identificando eventuais problemas que possam prejudicar os resultados do processo.
- **Preparação dos dados:** Os dados são devidamente identificados e tratados para, posteriormente, serem submetidos aos algoritmos de *Data Mining* (DM).
- **Modelação:** São aplicadas as técnicas de DM, com base nos objetivos identificados no primeira etapa.
- **Avaliação dos resultados:** Verifica se os modelos obtidos correspondem aos objetivos inicialmente propostos para o projeto.

- **Implementação ou desenvolvimento:** Prevê que os modelos e resultados encontrados sejam adaptados à organização, quer seja produzindo novas regras de negócio ou subsidiando novas estratégicas de mercado.

4. Trânsito na cidade do Porto

4.1 Descrição do dataset

Tal como já foi mencionado anteriormente, o *dataset* utilizado na competição teve como objetivo desenvolver um modelo capaz de prever o fluxo de trânsito num dado ponto temporal na cidade do Porto. O *dataset* contém várias variáveis destacando-se, de entre elas, a variável *average-speed-diff* que indica, numa escala qualitativa (*None*, *Low*, *Medium*, *High* e *Very-High*), se existe trânsito ou não. Esta *feature* corresponde à diferença entre (1) e (2):

1. A velocidade máxima que os veículos podem atingir em cenários sem trânsito
2. A velocidade que realmente se verifica naquele ponto temporal

Assim, se a diferença de velocidade é nula, significa que a velocidade dos veículos naquele determinado ponto é a velocidade máxima que lhes é permitido atingir em cenários sem trânsito. Se, por outro lado, a diferença de velocidade é *Very-High* significa que a velocidade numa determinada rua é muito inferior à velocidade máxima que lhes é permitido atingir em cenários sem trânsito, isto é, existe trânsito.

Foram disponibilizados dois *datasets*. Um deles, o *dataset* de aprendizagem, foi utilizado para treinar e fazer o *tuning* do modelo de *machine learning*. O segundo foi utilizado como *dataset* de teste. Assim, para cada registo do *dataset* de teste foi previsto o trânsito verificado, utilizando a escala anteriormente referida.

4.1.1 Dados do dataset

O *dataset* apresenta 6812 regtos de treino e 1500 regtos de teste. Contém 14 variáveis tais como:

- ***City_name***: variável nominal que apresenta o nome da cidade em causa;
- ***Record_date***: variável contínua que representa o *timestamp* associado ao registo;
- ***Average_speed_diff***: variável ordinal que apresenta a diferença de velocidade correspondente à diferença entre a velocidade máxima em cenários sem trânsito e a velocidade que realmente se verifica. Quanto mais alto o valor, maior é a diferença entre o que está em movimento no momento e o que deveria estar a andar sem trânsito, isto é, valores elevados deste atributo implicam um movimento mais lento. Esta será a variável resposta, ou seja, a variável sobre a qual a previsão recairá;
- ***Average_free_flow_speed***: variável contínua que mede o valor médio da velocidade máxima que os carros podem atingir em cenários sem trânsito (em km/h);
- ***Average_time_diff***: variável contínua que apresenta o valor médio da diferença do tempo que se demora a percorrer um determinado conjunto de ruas. Quanto mais alto o valor, maior é a diferença entre o tempo que demora a percorrer as ruas e o que deveria demorar sem trânsito, isto é, valores altos implicam um tempo mais alargado a atravessar o conjunto de ruas, em segundos;
- ***Average_free_flow_time***: variável contínua que apresenta o valor médio do tempo que demora a percorrer um determinado conjunto de ruas quando não há trânsito, em segundos;
- ***Luminosity***: variável ordinal que representa o nível de luminosidade na cidade do Porto;
- ***Average_temperature***: variável discreta que representa o valor médio da temperatura para o *record_date* na cidade do Porto;

- *Average_atmosp_pressure*: variável contínua que apresenta o valor médio da pressão atmosférica para o *record_date*;
- *Average_humidity*: variável discreta que representa o valor médio da humidade para o *record_date*;
- *Average_wind_speed*: variável discreta que representa o valor médio da velocidade do vento para o *record_date*;
- *Average_cloudiness*: variável nominal que representa o valor médio da percentagem de nuvens para o *record_date*;
- *Average_precipitation*: variável discreta que representa o valor médio de precipitação para o *record_date*;
- *Average_rain*: variável nominal que representa a avaliação qualitativa da precipitação para o *record_date*.

4.2 Análise/ Exploração

Em probabilidade e estatística, correlação, dependência ou associação é qualquer relação estatística entre duas variáveis. A matriz de correlação mostra os valores de correlação de *Pearson*, que medem o grau de relação linear entre cada par de variáveis. Os valores de correlação variam entre -1 e +1, em que -1 significa uma correlação negativa entre as variáveis, ou seja, quando uma aumenta a outra diminui e 1 significa uma correlação positiva. Através da análise da matriz obtida pelo KNIME consegue-se verificar que as variáveis com mais correlação são a *Luminosity* e a *average_time_diff*.

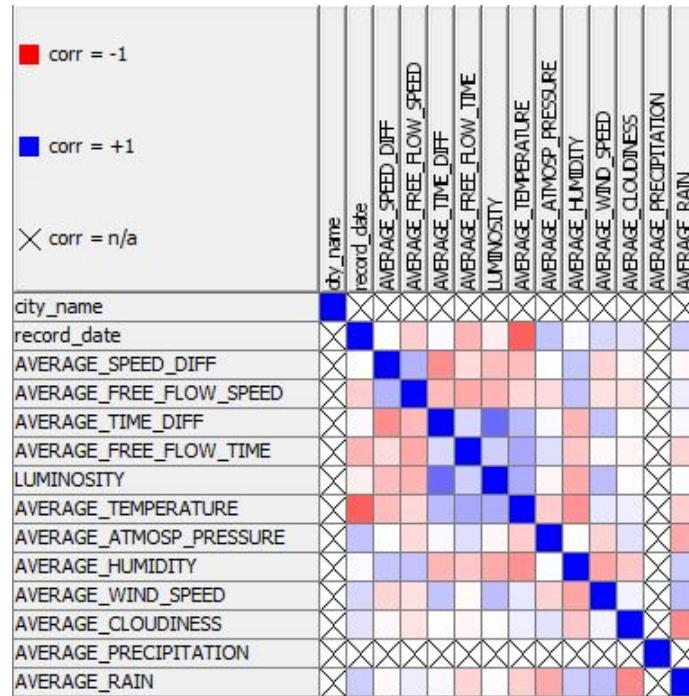


Figura 4.1: Matriz de correlação

Para além da correlação entre as variáveis considerou-se importante perceber o desvio padrão de cada uma. Assim, através da figura 4.2, é possível verificar essa medida. Pode-se considerar importante a sua análise, uma vez que variáveis com desvio padrão reduzido, variam pouco, podendo não impactar o modelo por tal razão.

| S Column | D Std. deviation |
|-------------------------|------------------|
| AVERAGE_PRECIPITATION | 0 |
| AVERAGE_WIND_SPEED | 2.138 |
| AVERAGE_FREE_FLOW_SPEED | 4.119 |
| AVERAGE_TEMPERATURE | 5.163 |
| AVERAGE_ATMOSP_PRESSURE | 5.751 |
| AVERAGE_FREE_FLOW_TIME | 8.294 |
| AVERAGE_HUMIDITY | 18.239 |
| AVERAGE_TIME_DIFF | 33.511 |

Figura 4.2: Desvio-padrão das variáveis do dataset

Uma vez analisada a tabela, percebe-se que existem variáveis com valores de desvio padrão muito reduzidos. No caso da "Average_precipitation", o valor é zero, uma vez que esta apenas toma esse valor, sendo considerada um erro. Na "Average_wind_speed" o desvio padrão também se considera reduzido e aliado à sua pouca explicação lógica no que diz respeito à previsão de trânsito, é considerada uma variável candidata a ser removida do modelo.

Por se perceber que a variável "Average_Time_Diff" se relacionava com a "Average_Speed_Diff", ponderou-se agrupar a mesma. Para definir os intervalos a agrupar criou-se um *Box-Plot* que permitiu perceber quais as medidas correspondentes aos quartis, sendo essa a base para a divisão dos intervalos, figura 4.3.

| Row ID | D AVERAGE_TIME_DIFF |
|----------------|---------------------|
| Minimum | 0 |
| Smallest | 0 |
| Lower Quartile | 2.25 |
| Median | 12.2 |
| Upper Quartile | 36.2 |
| Largest | 87.1 |
| Maximum | 296.5 |

Figura 4.3: Medidas do box-plot referente à variável Average_Time_Diff

Considerou-se o dia da semana um factor crucial na previsão do trânsito na cidade do Porto. Para tal fez-se uma análise exploratória do impacto destes no trânsito, Figura 4.4.

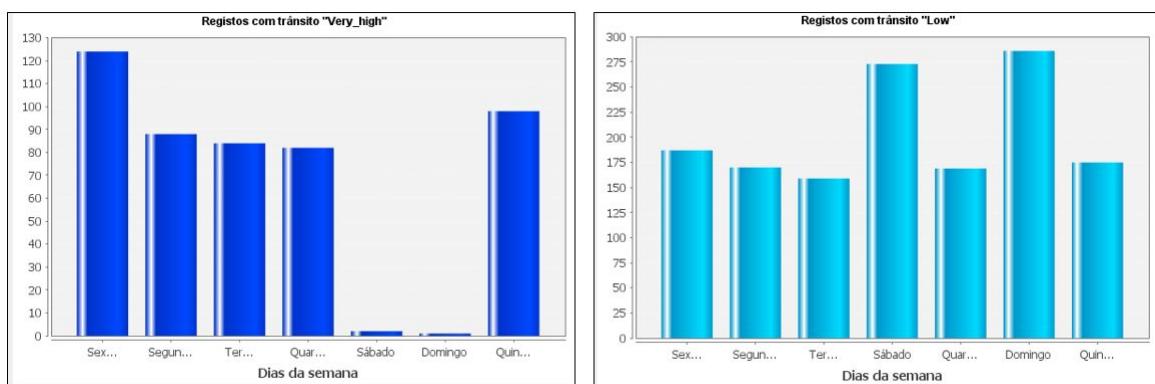


Figura 4.4: Gráfico de barras que permite identificar a importância dos dias da semana no modelo

Pela análise dos gráficos percebe-se que à semana existem muitos mais registos de trânsito designados por "Very_high", do que ao fim de semana. Já os registos de trânsito "Low" possuem uma maior frequência ao fim de semana, apesar de não tão acentuada comparativamente ao caso anterior.

4.3 Tratamento dos dados

Após a identificação de todas as variáveis do modelo, e ainda, depois de analisar cada uma delas ao detalhe, segue-se a fase responsável por tratar os dados. Esta fase é crucial para garantir que os dados submetidos posteriormente a algoritmos de *machine learning* estejam consistentes. Tendo em vista melhorar a precisão da previsão efetuada, o grupo decidiu analisar por partes o *dataset* fornecido.

4.3.1 Tratamento dos dados relativos à chuva

Da análise inicial, foi observado que o *dataset* apenas continha registos do tempo quando chovia, ou seja, quando não chovia não existia qualquer registo. Na generalidade dos casos, sabe-se que o trânsito tende a ser mais caótico com a presença deste fenómeno meteorológico e, por esta mesma razão, tornou-se quase obrigatório tratar deste aspeto. Neste sentido, ficou decidido que em todos os campos que não continham registo, isto é, "missing values (?)", foram substituídos pela string "Sem chuva".

Sempre que surgiu a string "NULL", também foi substituída pela string "Sem chuva", através do nodo "*String Replacer*".

Foi também observado que existiam campos com designações bastantes parecidas, por exemplo, existia a string "Chuva de intensidade pesado" e "chuva de intensidade pesada". Quase que não se repara, mas a primeira expressão encontra-se no masculino e a segunda no feminino. Ficou decidido passar tudo para "chuva de intensidade pesada".

4.3.2 Tratamento dos dados relativos ao *timestamp*

O trânsito prima, geralmente, pela organização, fluindo em faixas de tráfego numa direção particular e os períodos do dia têm naturalmente um grande impacto. Evidentemente que em determinadas horas existe mais trânsito do que noutras, desta forma também se tratou os dados relativos ao tempo.

1. Converteram-se as datas presentes na coluna *Record_date* para string.
2. Extraíram-se campos da data da coluna *Record_date*, nomeadamente, o ano, o mês (em número), dia da semana (em nome) e a hora.
3. Foram eliminados os valores em falta, representados no *dataset* por ?, apenas da coluna *Average_rain* uma vez que esta é considerada uma variável logicamente importante para a previsão em causa. Todos os valores em falta foram substituídos por "NULL", ou seja, considerados como ausência de chuva no *timestamp* considerado.

4.3.3 Criação da coluna com velocidade numérica sobre variável resposta

Como forma de tentar melhorar o modelo, criou-se uma variável com o intuito de obter valores que mais se assemelhassem à variável resposta, "Average_speed_diff".

Numa primeira fase percebeu-se que o *dataset* fornecia dados acerca de tempos e velocidades. Desta forma, utilizou-se uma simples fórmula de velocidade, presente na equação 4.1, como base para os cálculos.

$$V = \frac{d}{t} \quad (4.1)$$

1. Uma vez que:

- A variável "Average_free_flow_time" equivalia ao tempo que se demorava a percorrer o conjunto de ruas num cenário sem trânsito, em segundos.
- A variável "Average_free_flow_speed" correspondia à velocidade média máxima que se alcançaria num cenário sem trânsito, em km/h.

Assim, foi possível calcular a distância efetiva, em média, do conjunto de ruas em estudo, através da multiplicação do tempo pela velocidade, não sem antes converter o tempo para horas.

2. Para além disso:

- A variável "*Average_time_diff*" resulta da diferença entre o tempo que efetivamente demora a percorrer o conjunto de ruas e o tempo que demoraria na ausência de trânsito.

Em seguida, uma vez que já se possuía a distância efetiva do conjunto de ruas, passou-se a calcular o tempo efetivo que demoraria a percorrer-las por registo. Assim somou-se a "*Average_time_diff*" ao "*Average_free_flow_time*", resultando o tempo efetivo que demoraria a percorrer o conjunto de ruas, por registo.

3. Uma vez calculadas estas variáveis é possível encontrar a velocidade média efetiva, resultante do quociente entre a distância e o tempo encontrados nos dois passos anteriores.
4. Por fim, através da velocidade efetiva, é possível determinar a diferença entre a velocidade máxima num cenário sem trânsito e a velocidade anteriormente determinada, o que se considera ser, de um grosso modo, a definição do "*Average_Speed_Diff*", a variável a prever.

4.3.4 Filtragem das colunas

Nem todas as variáveis são importantes para o modelo. Assim, procedeu-se à filtragem de algumas colunas, as quais:

- **City_name**: Não se considera importante uma vez que a cidade é sempre a mesma, o Porto.
- **Record_date**: Uma vez que se extraíram os campos da data acima explicados, considerou-se que esta coluna poderia ser eliminada, por se tornar redundante.
- **Luminosity**: Não se considera importante no que diz respeito à previsão do trânsito, aliado ao facto de se relacionar, apesar de forma pouco acentuada, com outras como com a variável "*Average_time_diff*".
- **Average_atmosp_pressure**: Considerou-se que esta variável não possuí impacto na quantidade de trânsito. Para além disso possuí um desvio padrão reduzido, ou seja, não varia muito, o que torna difícil perceber o seu impacto.
- **Average_humidity**: Uma vez que, como referido anteriormente, se encontrava fortemente relacionada com a temperatura, foi eliminada. Para além disto, considerou-se que, logicamente, esta não terá grande impacto no trânsito.
- **Average_wind_speed**: Pela mesma lógica utilizada várias vezes, a velocidade do vento não terá grande impacto no trânsito rodoviário.
- **Average_cloudiness**: Eliminada uma vez que possuía muitos registos nulos e não seria possível substituir estes por não se ter conhecimento suficiente do *dataset* para tal.
- **Average_precipitation**: Uma vez que todos os registos desta variável tomavam valor zero, considerou-se não ser de qualidade para a previsão em causa.
- **Year**: Considerou-se não ser necessária pelo o facto de só existirem registos de dois anos, 2018 e 2019.

4.3.5 Agregação de dados

Com a intenção de melhorar os resultados procedeu-se à criação de *binners*. Nesta fase a ideia seria agrupar determinados intervalos de valores para filtrar, de forma mais eficiente, determinadas categorias.

| <i>Binner</i> | <i>Período de Trânsito</i> |
|---------------|------------------------------------|
| 1 | quase nulo: $]-\infty \dots 7.0[$ |
| 2 | muito elevado: $[7.0 \dots 9.0]$ |
| 3 | médio: $]9.0 \dots 12.0[$ |
| 4 | elevado: $[12.0 \dots 14.0]$ |
| 5 | médio: $]14.0 \dots 17.0[$ |
| 6 | muito elevado: $[17.0 \dots 19.0[$ |
| 7 | médio: $[19.0 \dots 21.0[$ |
| 8 | quase nulo: $[21.0 \dots +\infty[$ |

Tabela 4.1: Tabela explicativa dos períodos de trânsito

Desta forma, criou-se uma coluna nova, "Períodos de transito" utilizando as horas anteriormente extraídas da data. Esta coluna diz respeito aos períodos de trânsito, sendo estes divididos em 8 *binnners*, dando origem a 4 categorias, tal como se pode visualizar na tabela 4.1:

- "Período de trânsito quase nulo"
- "Período de trânsito muito elevado"
- "Período de trânsito elevado"
- "Período de trânsito médio"

| <i>Binner</i> | <i>AVERAGE_TIME_DIFF_binned</i> |
|---------------|---------------------------------|
| 1 |] -∞ ... 3.0[|
| 2 | [3.0 ... 12.0] |
| 3 |]12.0 ... 36.0[|
| 4 | [36.0 ... 87.0] |
| 5 |]87.0 ... +∞[|

Tabela 4.2: Tabela explicativa dos períodos de *AVERAGE_TIME_DIFF_binned*

Foi ainda criada uma coluna nova, "*AVERAGE_TIME_DIFF_binned*", utilizando os registos da coluna *Average_Time_Diff*. Foram criados 5 *binnings*, divididos de acordo com a amplitude interquartílica resultante do *Box-plot* criado, da variável em estudo. Os períodos estão expostos na tabela 4.2.

4.4 Modelação

4.4.1 Preparação do modelo

Depois de explorar o *dataset* fornecido, bem como as suas características, passou-se a tratar os dados com o objetivo de aumentar a precisão do modelo. Para tal, utilizou-se o *software KNIME*, utilizando as funcionalidades necessárias que permitiriam um tratamento de dados de encontro ao referido na secção 4.3.

4.4.1.1 Tratamento da string "NULL", respeitante à chuva

Em termos práticos, utilizou-se um nodo "*String Replacer*" para efetuar a substituição da string "NULL" pela string "sem chuva", figura 4.5.

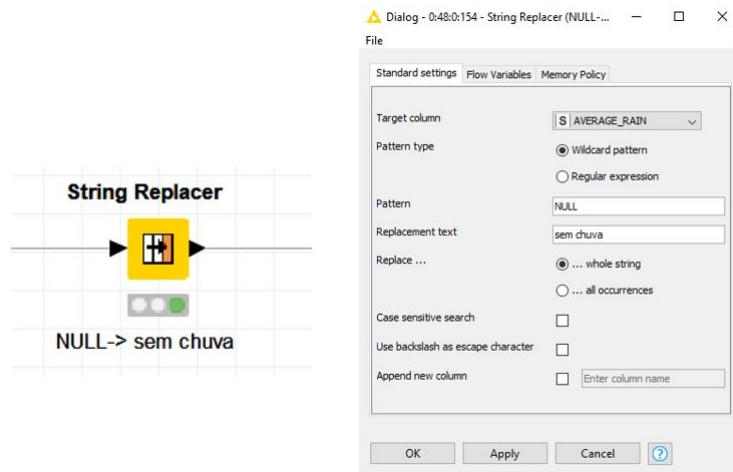


Figura 4.5: Nodo responsável pela substituição da string "NULL" pela string "sem chuva", no KNIME.

4.4.1.2 Tratamento dos *missing values*, respeitantes à chuva

Nesta fase do tratamento utilizou-se um nodo "Missing Value" para efetuar a substituição dos *missing values*, (?), pela string "sem chuva", figura 4.6.

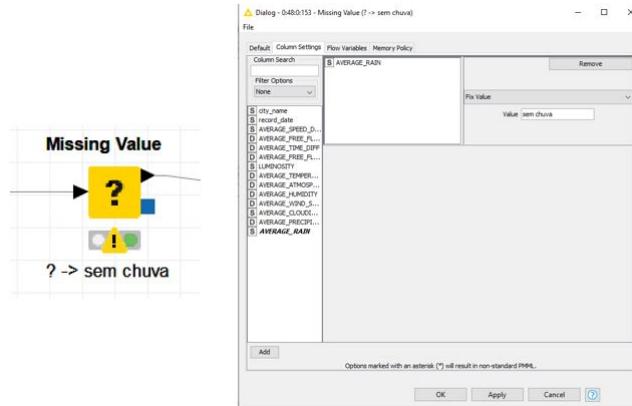


Figura 4.6: Nodo responsável pela substituição de todos os *missing values* (?) pela string "sem chuva", no KNIME

4.4.1.3 Tratamento da string "chuva de intensidade pesado"

Como já referido, uma vez ter sido encontrado no *dataset* um erro na designação da chuva, utilizou-se novamente um nodo *String Replacer* para substituir, de uma forma lógica, a mesma pela string "chuva de intensidade pesada", substituição esta evidenciada na figura 4.7

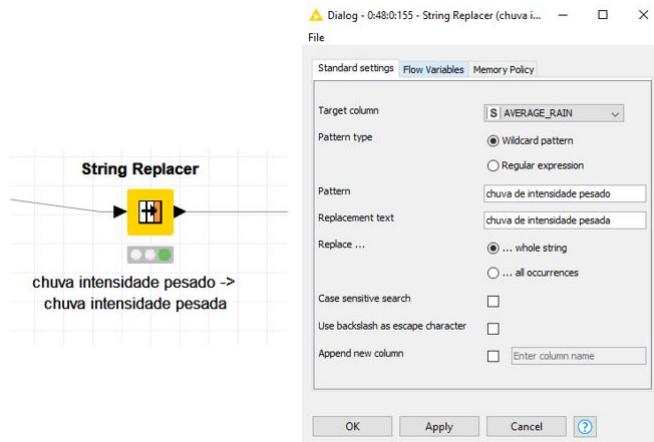


Figura 4.7: Nodo responsável pela substituição da string "chuva de intensidade pesado" pela string "chuva de intensidade pesada", no KNIME

4.4.1.4 Extrair dados da variável *timestamp*

Converter o *timestamp* para um formato de data

Para que possam ser extraídos da data os campos considerados relevantes para o modelo é necessário converter, previamente, esta, que originalmente está definida com uma *string*, para um formato de data. Para tal utilizou-se um nodo "*String to Date & Time*", como se pode verificar na figura 4.8.

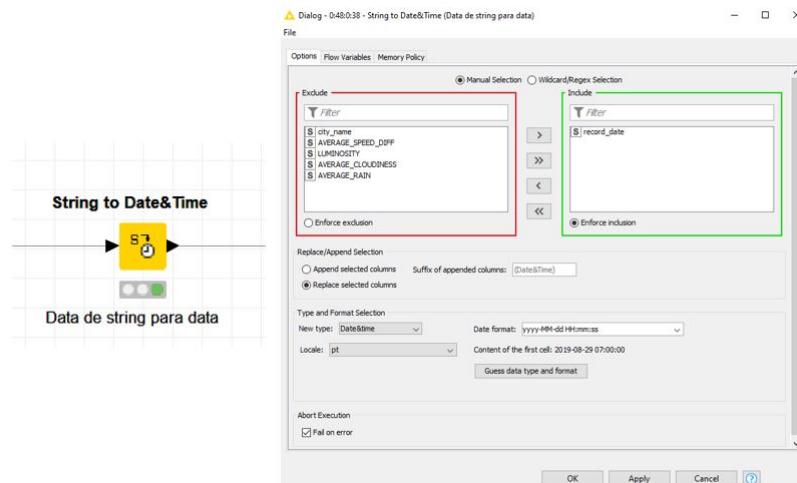


Figura 4.8: Nodo responsável pela conversão da data do registo a uma *string*, no KNIME

De referir que o *timestamp* estava definido como o equivalente a um Date&time com formato yyyy-MM-dd HH:mm:ss apresentado.

Extrair campos mês, dia da semana e hora da data

Como já referido, considera-se que hora, o dia da semana e o mês têm um impacto significativo no trânsito. Desta forma, estes foram os campos extraídos da data, através do nodo "*String to Date&Time*", como é possível observar através da figura 4.9. Assim, foram criadas mais três colunas/variáveis, correspondentes aos campos referidos.

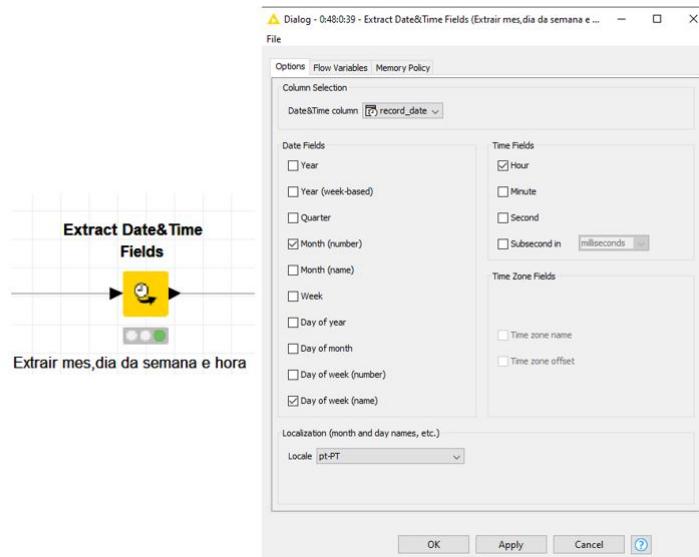


Figura 4.9: Nodo responsável pela extração dos campos mês, dia da semana e hora da data do registo, no KNIME

4.4.2 Criação de *binners* numéricos

Agrupar a variável "Average_time_diff"

Depois de feita a análise necessária, e como já referido anteriormente, numa tentativa de melhorar a precisão do modelo, procedeu-se ao agrupamento da variável "Average_time_diff". Assim, utilizando o nodo "Numeric Binner", agrupou-se esta variável em cinco intervalos, criando uma nova variável no modelo, "AVERAGE_TIME_DIFF_binned", Figura 4.10.

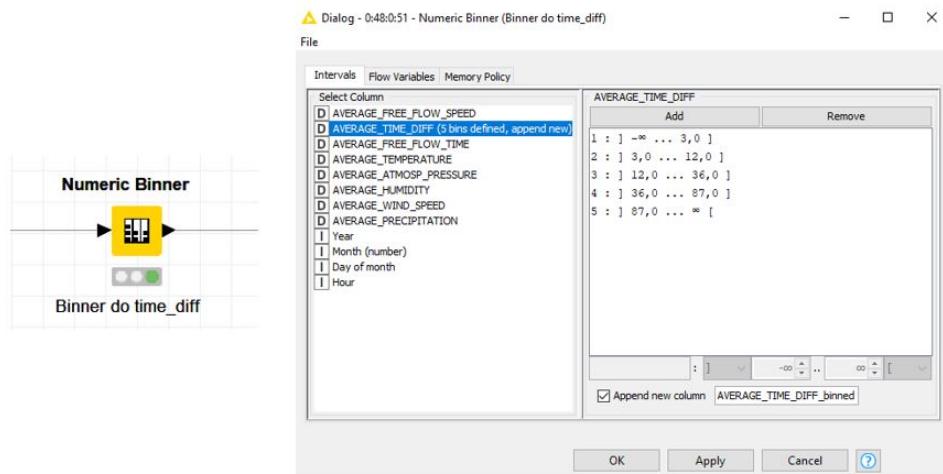


Figura 4.10: Nodo responsável pela criação do biner numérico, "AVERAGE_TIME_DIFF_binned" no KNIME

Agrupar a variável Hour

Sendo a hora uma variável importante no que diz respeito à previsão do trânsito na cidade do porto, agruparam-se estas, utilizando também o nodo "Numeric Binner", dando origem a uma nova variável no modelo, "Hour_binned". Foram criadas quatro categorias, mas uma vez tratar-se de uma variável contínua, foi necessária a criação de 8 bins, figura 4.11.

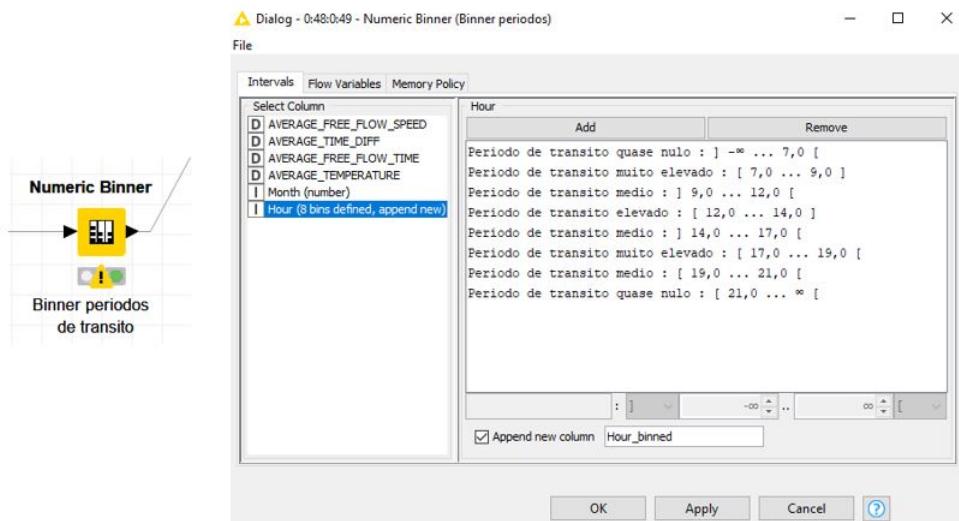


Figura 4.11: Nodo responsável pela criação do biner numérico "Hour_binned" no KNIME

4.4.2.1 Criação de *binnings* nominais, com a utilização de um dicionário

Numa tentativa de melhorar o modelo procedeu-se à criação de um dicionário, atribuindo a cada dia da semana uma "designação", ou seja, se se tratava de um dia da semana regular ou fim de semana, designado de "Semana" e "Fim de semana", respetivamente. Para tal associação utilizou-se um nodo "Table Creator", associado a um nodo "Binner (Dictionary)", como apresentado na figura 4.12.

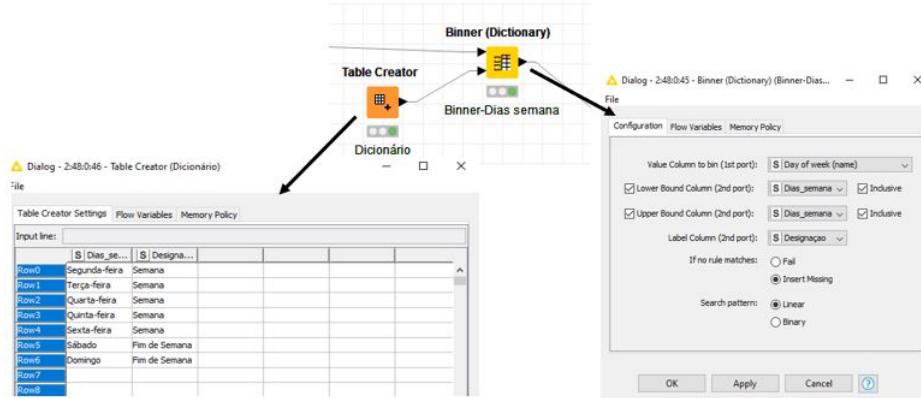


Figura 4.12: Nodos responsáveis pela criação do dicionário referente à designação do dia da semana, no *KNIME*

4.4.2.2 Criação de uma variável auxiliar respeitante à velocidade efetiva, média, por registo

Foi criada uma variável auxiliar através de cálculos efetuados com as variáveis já existentes, através da sucessão de vários nodos *Math Formula*, como representado na figura 4.13.

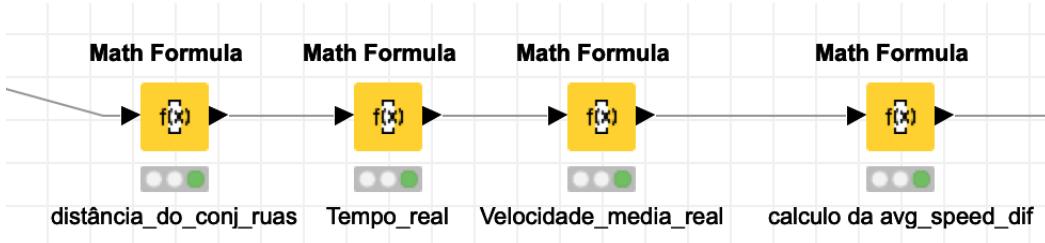


Figura 4.13: Nodos responsáveis pelo cálculo da variável em estudo

1. Cômputo da distância, em média, do conjunto de ruas em estudo

Numa primeira abordagem calculou-se a distância (em quilómetros), em média, do conjunto de ruas em estudo. Foi criada uma nova variável no modelo, designada por "distância_do_conj_ruas". Esta resulta da multiplicação entre a variável "*AVERAGE_FREE_FLOW_SPEED*" pela variável "*AVERAGE_FREE_FLOW_TIME*", ou seja, a velocidade em média (km/h) que um carro atinge a percorrer uma rua vazia multiplicada pelo tempo (em segundos convertidos a horas) que o carro demora a percorrer efetivamente esse conjunto de ruas, como se pode observar na figura 4.14.

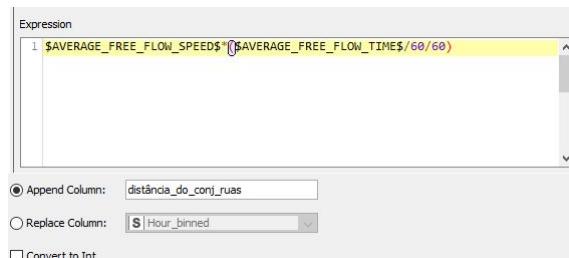


Figura 4.14: Fórmula responsável pelo o cálculo da distância, em média, do conjunto de ruas em estudo no *KNIME*

2. Cômputo do tempo efetivo que demora a percorrer o conjunto de ruas em estudo

Em seguida calculou-se o tempo efetivo que demora (em horas) a percorrer o conjunto de ruas em estudo. Assim, foi criada uma nova variável no modelo, designada "Tempo_real". Esta resulta da soma entre a variável "AVERAGE_TIME_DIFF" com a variável "AVERAGE_FREE_FLOW_TIME". Uma vez que a primeira diz respeito à diferença entre o tempo efetivo que o carro demora a percorrer e o tempo que este demoraria num cenário sem trânsito, somada à segunda, o tempo que demora na ausência de trânsito, é possível calcular o tempo efetivo que este demora a percorrer o conjunto de ruas, figura 4.15.

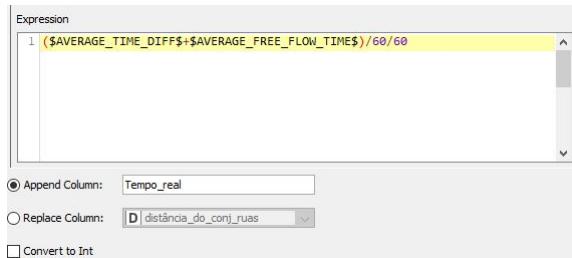


Figura 4.15: Fórmula responsável pelo o cálculo do tempo efetivo (em horas) que demora a percorrer o conjunto de ruas em estudo no KNIME

3. Cômputo da velocidade média efetiva

Uma vez já calculada a distância do conjunto de ruas e a velocidade média efetiva por registo, é possível calcular a velocidade efetiva de cada registo. Desta forma, criou-se uma nova variável no modelo, "Velocidade_media_real", resultante do quociente entre a "distância_do_conj_ruas" e o "Tempo_real", figura 4.16.

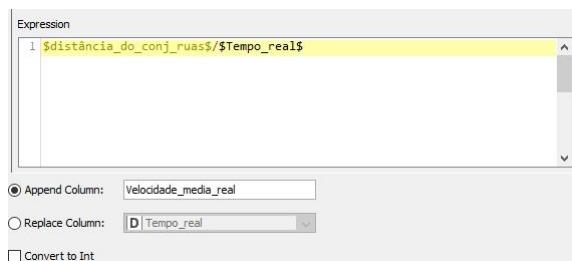


Figura 4.16: Fórmula responsável pelo o cálculo da velocidade média efetiva (km/h) de cada registo KNIME

4. Cômputo da diferença entre a velocidade num cenário sem trânsito e a velocidade efetiva

Numa tentativa de criar uma variável que se aproxima-se à variável alvo de previsão, "AVERAGE_SPEED_DIFF", criou-se uma nova variável no modelo, "Max-real(previsão)". Esta resulta da diferença entre a média da velocidade num cenário sem trânsito, "AVERAGE_FREE_FLOW_SPEED" e a velocidade efetiva, anteriormente calculada, "Velocidade_media_real", figura 4.17.

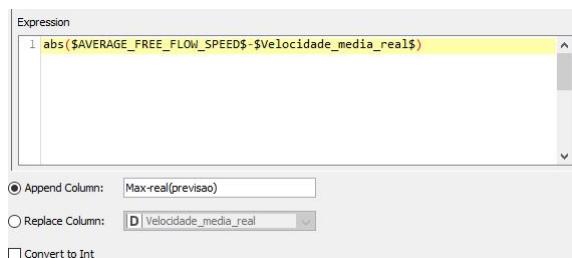


Figura 4.17: Fórmula responsável pelo o cálculo da velocidade média efetiva (km/h) de cada registo KNIME

4.4.2.3 Filtragem de colunas final

Depois de todas as transformações consideradas importantes serem realizadas, passou-se à filtragem de algumas colunas consideradas não relevantes para o modelo, através do nodo *Column Filter*, figura 4.18.

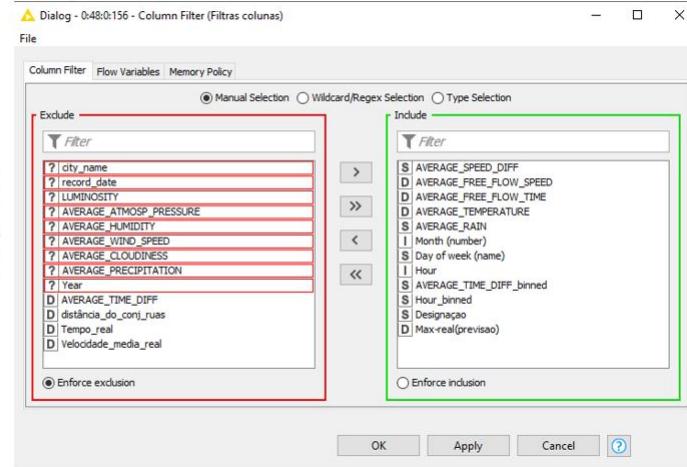


Figura 4.18: Colunas filtradas no KNIME

As colunas intermédias que serviram de cálculo auxiliar à nova variável ("distância_do_conj_ruas", "Tempo_real" e "Velocidade_media_real"), "Max-real(previsão)", são excluídas do modelo por se considerar que apenas servem de auxílio, sendo o foco na variável final, a não excluída.

4.4.2.4 Workflow completo da preparação de dados do dataset

Depois de todas as transformações acima referidas, obteve-se um workflow sequencial respeitante ao tratamento de dados do dataset, figura 4.19

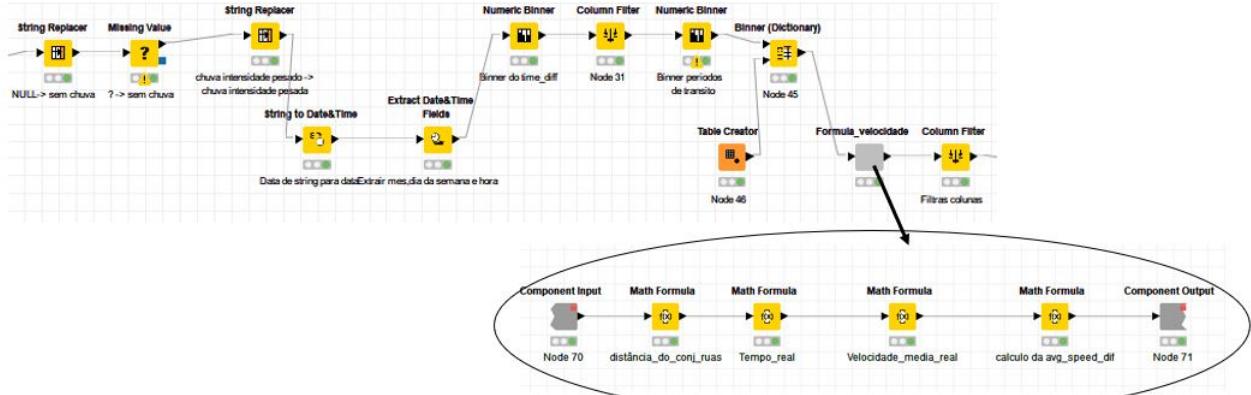


Figura 4.19: Worflow final da preparação dos dados do dataset em KNIME

4.4.3 Tuning do modelo

Quando se treina um modelo, é necessário fazer algumas escolhas dos valores de parâmetros. E como não há certeza de quais parâmetros escolher para obter um melhor modelo é necessário fazer testes com diferentes valores. É nesta situação que o *loop* de otimização é útil, pois refina essa escolha inicial. Em vez de escolher apenas um valor único, pode-se definir um intervalo de valores e, eventualmente, atribuir o melhor em relação à função de custo definida através do *loop* de otimização. Decidiu-se demonstrar as configurações em primeiro lugar do modelo que usa *Random Forest*, capítulo 4.4.3.1, pois com este algoritmo foi possível obter resultados com erros menores do que utilizando *Decision Tree* presente no capítulo 4.4.3.2. As configurações para estes dois modelos são ligeiramente diferentes, para o segundo modelo apresentado apenas serão descritas as diferenças em relação ao primeiro.

4.4.3.1 Random Forest

Para o algoritmo *Random Forest*, começou-se por definir, através de um *Table Creator*, os critérios de divisão usados pelo nodo *Random Forest Learner*, apresentados na figura 4.20. Sendo eles o *Information Gain*, *Information Gain Ratio* e o *Gini Index*.

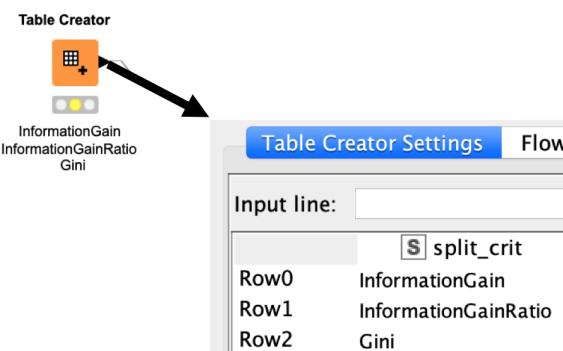


Figura 4.20: Critérios de divisão especificados no nodo *Table Creator*

Foi necessário acrescentar, de seguida, um nodo denominado por *Table Row to Variable Loop Start*. Este nó usa cada linha da tabela de dados para definir novos valores de variáveis para cada iteração de *loop*. O nome da variável é definido pelo nome dado na coluna.

De seguida, introduziu-se o nodo *Parameter Optimization Loop Start*, este *loop* executa uma otimização de parâmetros. Na implementação atual da otimização de parâmetros para o modelo *Random Forest*, pretende-se otimizar os seguintes parâmetros para obter o melhor desempenho de classificação:

- A profundidade da árvore (prof);
- O tamanho mínimo da folha (node);
- O número ideal de árvores na floresta (nr_models).

Para cada um dos parâmetros mencionados acima, correu-se o modelo alternando subconjuntos dos valores presentes na tabela 4.3. Decidiu-se correr o modelo com valores dentro dos apresentados na tabela 4.3, mas nunca tudo de uma vez, pois o modelo demorava demasiado tempo a completar todas as iterações.

Tabela 4.3: Valores utilizados para otimização de parâmetros

| | Min | Max | Salto |
|---------------------|-----|-----|-------|
| Profundidade | 1 | 25 | 1 |
| Nodos | 1 | 15 | 1 |
| Nr Árvores | 100 | 600 | 100 |

Para cada um desses parâmetros, os valores de início e fim, bem como o tamanho do salto, são definidos na caixa de diálogo de configuração do nodo, exemplificada na figura 4.21. Os diferentes parâmetros são emitidos como variáveis de fluxo pelo nodo *Parameter Optimization Loop Start*, e as configurações dos parâmetros para o modelo

Random Forest são substituídas por estas variáveis. Usou-se a estratégia de pesquisa *BruteForce*, que explora todas as combinações possíveis dos valores dos parâmetros e escolhe a melhor.

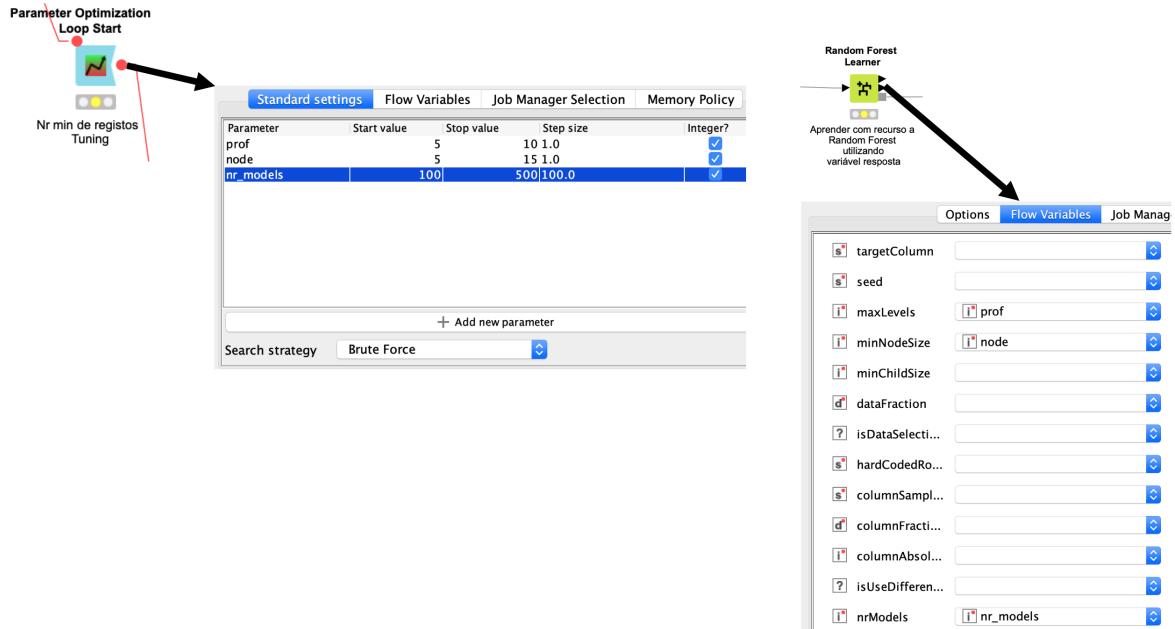


Figura 4.21: Definições de configuração do nodo *Parameter Optimization Loop Start*.

Posteriormente, acrescentou-se o nodo *X-Partitioner*, para definir o número de iterações de validação cruzada que deveriam ser executadas. Nesta configuração, o *learner* e o *predictor* são re-executados repetidamente, para este caso em estudo definiu-se que seriam dez repetições. No final do *loop*, é necessária a existência de um *X-Aggregator* para recolher os resultados de cada iteração. Todos os nodos entre estes dois são executados quantas vezes as iterações definidas no nodo *X-Partitioner*. As configurações utilizadas estão presentes na figura 4.22.

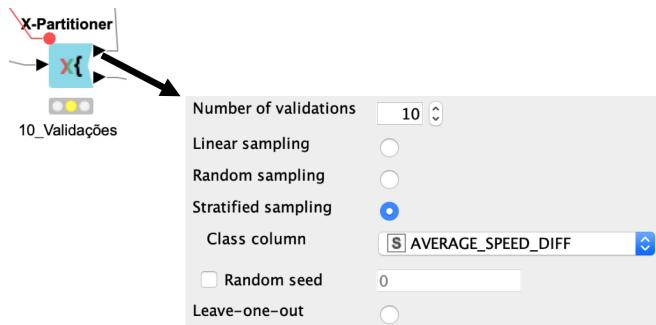


Figura 4.22: Configuração do nodo *X-Partitioner*

Foi também necessário acrescentar o nodo *Parameter Optimization Loop End*, para otimizar os parâmetros que minimizam o erro. Na janela de configuração do nodo *Parameter Optimization Loop End*, foi selecionado o critério de otimização usado para determinar quais os melhores valores de parâmetro para o modelo, de modo a minimizar a percentagem de erro, as configurações usadas estão evidenciadas na figura 4.23.

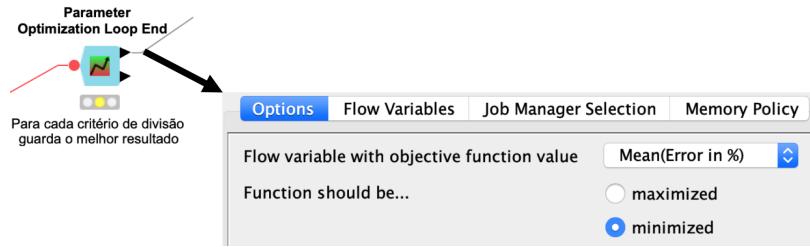


Figura 4.23: Caixa de diálogo de configuração do nodo *Parameter Optimization Loop End*. Otimizaram-se os parâmetros minimizando o valor na variável de fluxo chamada “Erro médio em %”.

De modo a facilitar a compreensão da modelação efetuada, expõe-se na figura 4.24 o modelo completo que possibilitou testar e encontrar os parâmetros em que se obteve um erro de previsão menor.

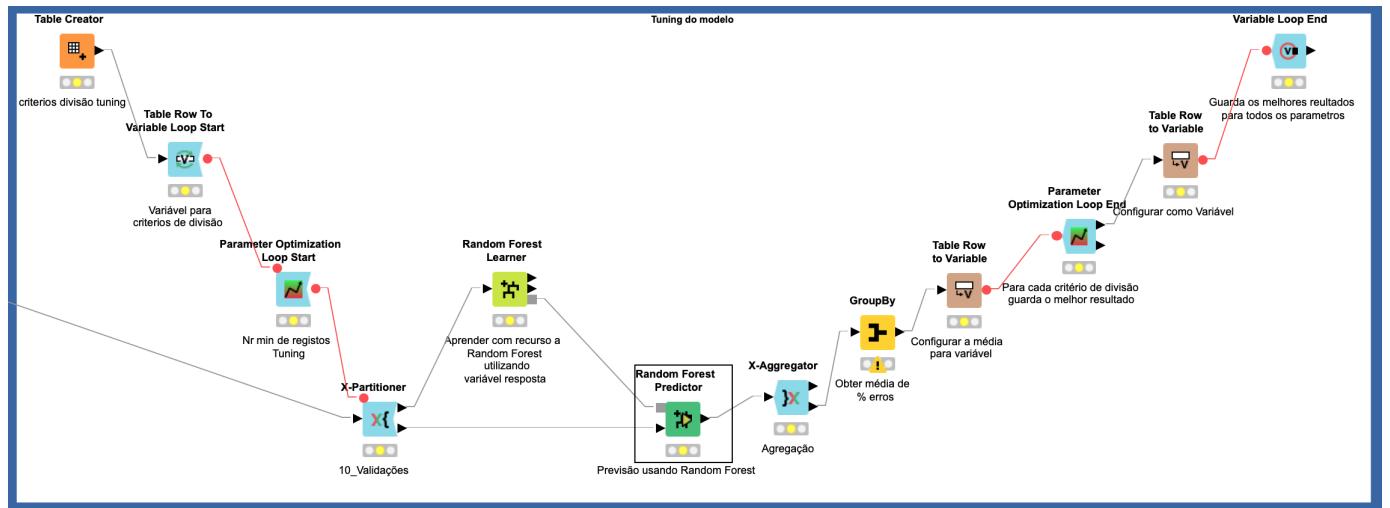


Figura 4.24: Tuning do modelo usando *Random Forest*

4.4.3.2 Decision Tree

De modo a fazer *pruning* do modelo utilizando o algoritmo *decision tree* foi necessário usar um *Table Creator* com os parâmetros a iterar no nodo *Decision Tree Learner*. Na imagem 4.25 é possível verificar quais os parâmetros usados para a otimização do modelo. Os valores permitidos como medida de qualidade são o *Gain Ratio* e *Gini Index*, o método de *pruning* existente é o *No pruning* ou *Minimum Description Length* (MDL). A opção *Reduced Error Pruning*, se marcada (*true*), aplica um simples processo de poda. Neste caso, os resultados com ou sem esta funcionalidade não apresentaram diferenças significativas.

The screenshot shows the Table Creator interface with a title bar "Table Creator". Below it is a toolbar with icons for creating, deleting, and saving tables. A tooltip "Parâmetros a iterar Decision Tree" points to the "Create" icon. The main area displays a table titled "Table Creator Settings" with the following data:

| | <input type="checkbox"/> metho... | <input type="checkbox"/> pruning | <input type="checkbox"/> Reduced_p |
|------|-----------------------------------|----------------------------------|------------------------------------|
| Row0 | Gain ratio | No pruning | true |
| Row1 | Gain ratio | MDL | true |
| Row2 | Gini index | No pruning | true |
| Row3 | Gini index | MDL | true |
| Row4 | Gain ratio | No pruning | false |
| Row5 | Gain ratio | MDL | false |
| Row6 | Gini index | No pruning | false |
| Row7 | Gini index | MDL | false |

Figura 4.25: Table Creator com definição das medidas de qualidade, método de poda e uso ou não de *Reduced Error Pruning*

Um outro critério a ter em conta é o critério de paragem, pois poderá ajudar a evitar o *overfitting*. Serve para definir o número mínimo de registo por nó. A ramificação interromperá o crescimento quando um nó contiver um número menor ou igual de amostras ao número mínimo de registo por nó. Portanto, um valor mais alto leva a árvores mais rasas, enquanto um número menor leva a árvores mais profundas.

Como explicado no modelo anterior, os diferentes parâmetros criados são emitidos como variáveis de fluxo, e as configurações do nodo *Decision Tree Learner* são substituídas por estas variáveis. As configurações do nodos estão presentes na figura 4.26.

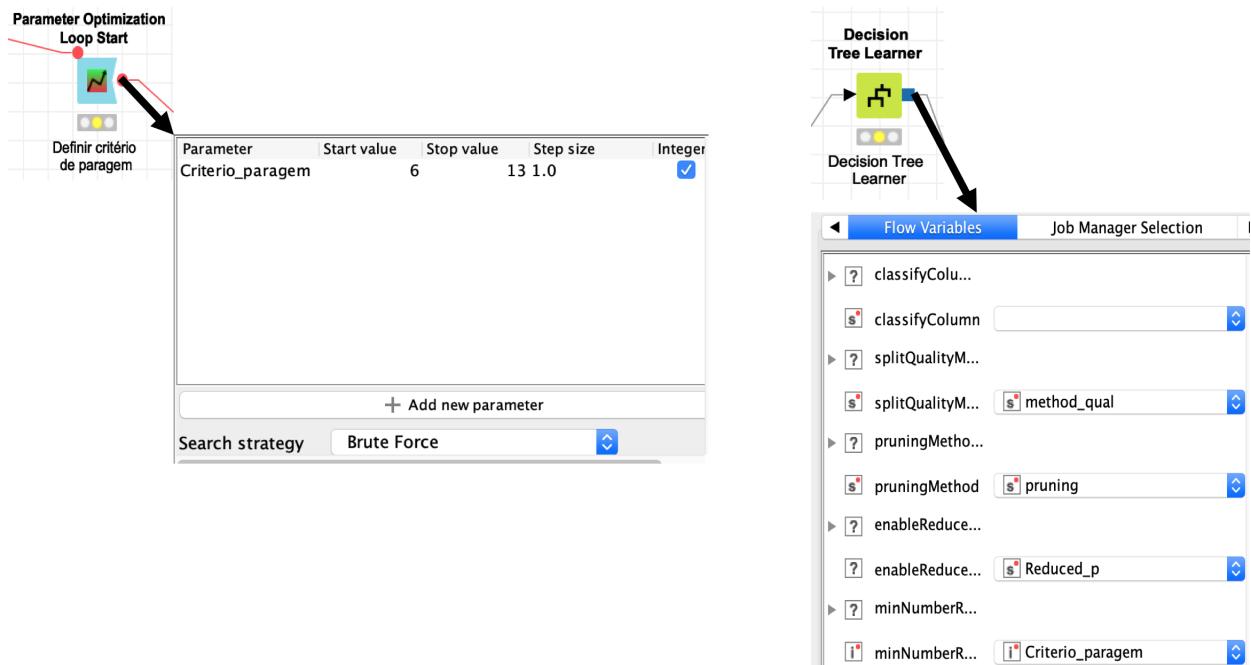


Figura 4.26: Parâmetros utilizados para encontrar a melhor solução

De um modo geral, as alterações efetuadas acima são as mais relevantes em relação ao modelo *Random Forest*. Exibe-se de seguida na figura 4.27 o modelo completo do *tuning* para este algoritmo.

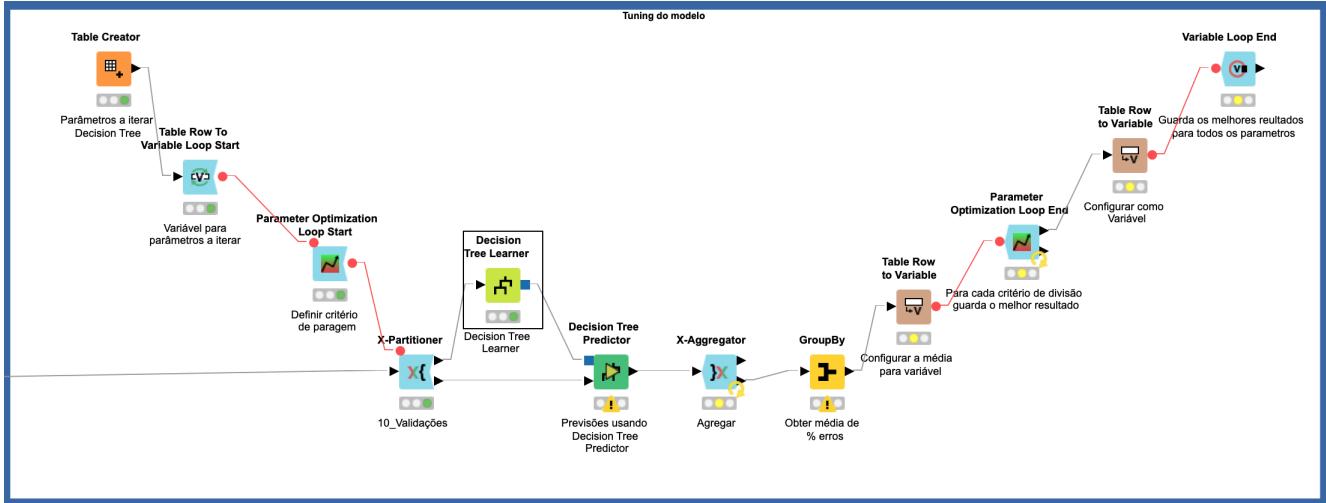


Figura 4.27: Modelo final para calcular os melhores resultados de *pruning* utilizando o algoritmo *Decision Tree*

4.4.4 Feature Selection

O nodo *Feature Selection Loop Start* é o início do *loop* de seleção de recursos. Este, permite selecionar, de entre todos os recursos do conjunto de dados de entrada, o subconjunto de recursos que é melhor para a construção do modelo. Com este nodo, é possível determinar quais as colunas que devem ser mantidas no processo de seleção. Um *loop Forward Feature Construction* cria vários classificadores pré-selecionados usando um número incremental de recursos de entrada. Este *loop* inicia a partir de 1 recurso e adiciona um recurso de cada vez nas iterações subsequentes.

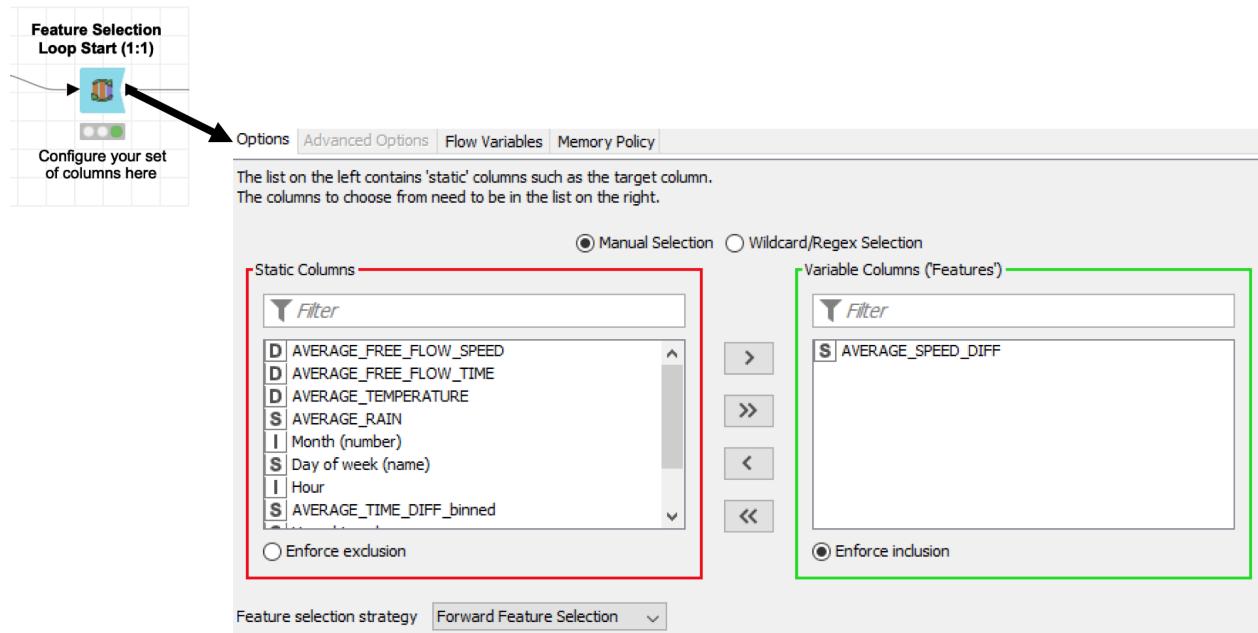


Figura 4.28: Configuração do nodo *Feature Selection Loop Start*

O número de recursos é um parâmetro fixo e é definido no nodo *Feature Selection Loop End*. A tabela de dados de saída do nodo inclui as colunas de dados selecionadas e a precisão de classificação correspondente como é possível observar na imagem 4.29.

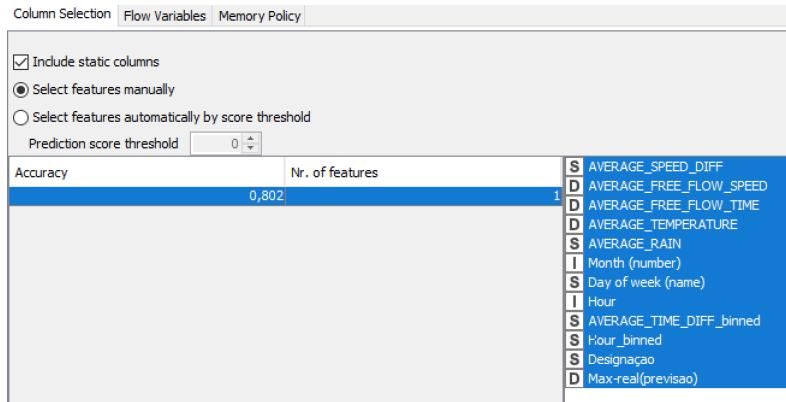


Figura 4.29: Resultados obtidos com o *Feature Selection*

E é possível concluir que todas as colunas dadas como entrada são importantes para obter um melhor resultado. Na imagem 4.30 está demonstrado o modelo final que permitiu a obtenção de resultados.

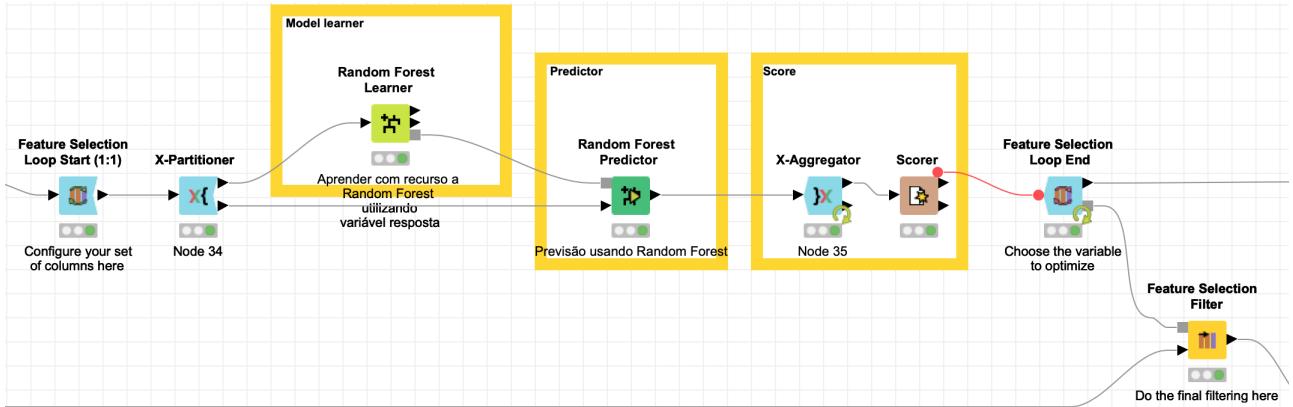


Figura 4.30: Modelo construído para seleção de colunas com o método *Forward Feature Selection*

4.5 Resultados Obtidos

Neste capítulo serão apresentados os resultados obtidos através dos dois algoritmos estudados neste projeto.

4.5.1 Resultados Obtidos utilizando o algoritmo *Decision Tree*

Numa frase incipiente do projeto foram obtidos resultados através do *tunning* do modelo utilizando árvores de decisão, mas em contrapartida, não foi utilizado o *cross validation* mas sim um nodo *partitioning* simples, no qual se fez uma divisão de 80% para treino e 20% para teste. Assim, não foi considerando um bom método, pelo que foram tidos em conta esses resultados, mas sim os obtidos utilizando *cross validation*.

Desta forma, na tabela 4.4 estão presentes os resultados obtidos com os dados tratados e com o mínimo número de colunas usadas, utilizando o *cross validation*.

Tabela 4.4: Resultados obtidos com o algoritmo *Decision Tree*

| Critério de paragem | Obj. Value | Split Criteria | Pruning |
|---------------------|------------|----------------|------------|
| 15 | 21.123 | Gain ratio | MDL |
| 13 | 21.286 | Gini index | MDL |
| 13 | 22.651 | Gain ratio | No pruning |
| 15 | 22.607 | Gini index | No pruning |

Depois de feita a preparação percebe-se que, utilizando o algoritmo *Decision tree*, o melhor critério de separação será o *Gain ratio* aliado ao MDL, como *Pruning*.

4.5.2 Resultados obtidos utilizando o algoritmo *Random Forest*

Para obter os resultados deste modelo, foram usados os dados tratados como demonstrado na secção 4.3. É de salientar que se testou a variação de erros quando se acrescentava ou diminuía o número de colunas usadas para o treino do modelo, mas não se notaram melhorias. Pode-se concluir que os melhores resultados foram atingidos quando se utilizou as colunas referenciadas no capítulo 4.3. Assim sendo, os melhores resultados estão presentes na tabela 4.5.

Tabela 4.5: Resultados obtidos com o algoritmo *Random Forest*

| | Information Gain | Information Gain Ratio | Gini index |
|-------------------------|------------------|------------------------|------------|
| Limit number of levels | 22 | 17 | 23 |
| Minimum child node size | 8 | 11 | 10 |
| Number of models | 300 | 600 | 200 |
| Objective value | 18.952 | 19.128 | 19.172 |

Pode-se concluir que, efetivamente, este modelo ainda poderá melhorar, talvez se se fizer uma abordagem ligeiramente diferente de tratamento de dados. Pois verificou-se ao longo do projeto que um bom tratamento de dados melhora bastante o número de acertos do modelo. Por este motivo, talvez fosse possível melhorá-lo se se analisasse mais aprofundadamente as variáveis que apresentavam uma menor correlação com a variável resposta, e preparar a sua integração no modelo. Para além disto escolher um abordagem diferentes para as consideradas importantes na variável resposta, uma vez que, existem centenas de abordagens possíveis, muitas das quais ainda não abordadas.

5. Animals Shelter

5.1 Descrição do dataset

O segundo *dataset* utilizado é sobre um abrigo de animais, no qual apenas são acolhidos cães e gatos. Tal como no modelo do *dataset* anterior, também neste o objetivo é construir um modelo capaz de fazer previsões, neste caso sobre o futuro dos animais no abrigo.

Das nove variáveis presentes no *dataset*, é importante destacar a variável *OutcomeType* visto ser a variável que se pretende que o modelo preveja. Esta é uma variável qualitativa (*adoption, retorno to owner, transfer, euthanasia, died*) que indica o destino do animal.

Outra variável importante é o *OutcomeSubType*, que ajuda explicar a variável *OutcomeType* e indica a causa do resultado da decisão do futuro do animal.

5.1.1 Dados do Dataset

O *dataset* apresenta 26729 registos, sendo que foram usados para treino do modelo 80% (21351 registos) e 20% (5338 registos) para teste. Em seguida estão descritas todas as variáveis:

- **Name**: variável nominal que apresenta o nome dos animais no abrigo em estudo;
- **DateTime**: variável contínua que representa a data e a hora em que foi decidida a situação final do animal, ou seja, em que se decidiu o resultado apresentado pela variável *OutcomeType*;
- **OutcomeType**: variável nominal que representa o tipo de resultado decidido para o animal, tendo cinco hipóteses possíveis, apresentadas na tabela 5.1;

Tabela 5.1: Tabela *OutcomeType*

| <i>OutcomeType</i> |
|--------------------|
| Adoption |
| Return to owner |
| Transfer |
| Euthanasia |
| Died |

- **OutcomeSubType**: variável nominal que representa a razão para a decisão sobre o que fazer com os animais, ou seja, ajuda a explicar a variável *OutputType*. Na tabela 5.2 estão presentes as dezasseis razões;

Tabela 5.2: Tabela *OutcomeSubType*

| <i>OutcomeSubType</i> | | | |
|-----------------------|-------------|-----------|------------|
| Partner | Foster | Suffering | Aggressive |
| SCRCP | Offsite | Behavior | Medical |
| In Kennel | Rabies Risk | In Foster | Enroute |
| Court/ Investigation | At vet | Barn | In Surgery |

- **AnimalType**: variável nominal que representa o tipo de animal presente no abrigo. Existem apenas dois tipos de animais, cães e gatos;

- **SexuponOutcome:** variável nominal que apresenta se o animal foi castrado, esterilizado ou se não sofreu nenhuma intervenção. Na tabela 5.3 estão presentes as opções possíveis para esta variável;

Tabela 5.3: Tabela SexuponOutcome

| SexuponOutcome |
|----------------|
| Neutered Male |
| Spayed Female |
| Intact Male |
| Intact Female |
| Unknown |

- **AgeuponOutcome:** variável contínua que indica a idade do animal. Apresenta as idades tanto em dias, como em semanas, meses ou anos;
- **Breed:** variável nominal que indica a raça do animal, sendo que existem um grande número de raças;
- **Color:** variável nominal que indica a cor de cada animal.

5.2 Análise/Exploração

Através da matriz de correlação criada no *Knime*, presente na figura 5.1, é possível perceber que as variáveis *OutcomeType* e *OutcomeSubType* são altamente correlacionadas (0,7543). Esta forte correlação já foi explicada anteriormente e prende-se com o facto da variável *OutcomeSubType* ser utilizada para explicar a *OutcomeType*. As restantes variáveis não têm correlações significativas.

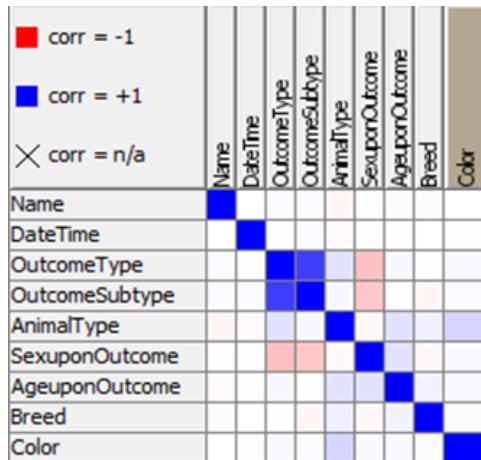


Figura 5.1: Matriz de correlação

Depois de perceber esta correlação, fez-se uma análise mais detalhada das duas variáveis. Assim, constatou-se que a maior parte dos registo com *missing values* na variável *OutcomeSubType*, tinham, ao mesmo tempo, a categoria *adoption* na variável *OutcomeType*. Como não era possível descobrir o porquê desses animais serem adotados, decidiu-se atribuir a *string unknown* a todos os valores em falta.

Na figura 5.2 está presente um gráfico que permite visualizar quais das cinco decisões possíveis acerca dos animais foram as que mais aconteceram.

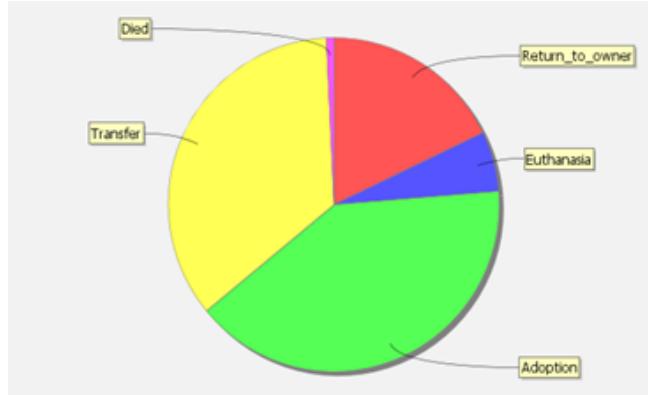


Figura 5.2: Gráfico variável *OutcomeType*

Pela análise do gráfico, percebe-se que as categorias com mais registos são a *transfer* e a *adoption*. Foi então decidido explorar as restantes variáveis de forma a perceber quais as razões que levam as pessoas a adotar.

Relativamente à raça, é de esperar que as pessoas prefiram animais de raça pura, isto é, não têm cruzamento com outras raças, em vez de animais que são fruto do cruzamento de raças.

Outra das características que são valorizadas na escolha do animal para adotar é o facto de este ser castrado/esterilizado, pois estas intervenções são dispendiosas e caso já estejam feitas pouparam dinheiro ao novo dono.

Na figura 5.3 está presente o gráfico com as idades dos animais que foram adotados.

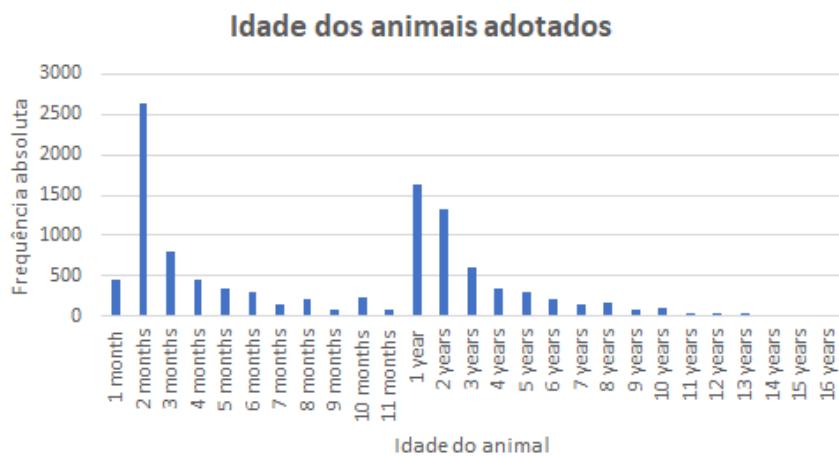


Figura 5.3: Gráfico sobre as idades

Pela análise do gráfico é perceptível que as pessoas preferem animais com apenas 2 meses, ou com 1 ou 2 anos. Isto acontece porque é mais fácil treinar um animal ainda jovem, pois podem ensinar-lhe a comportar-se como a pessoa quer.

5.3 Tratamento dos dados

Tal como foi feito no tratamento de dados do primeiro *dataset*, depois de identificadas e analisadas todas as variáveis, segue-se a fase de tratar os dados de forma a torná-los consistentes.

5.3.1 Tratamento dos dados relativos ao *OutcomeSubType*

Na fase inicial, observou-se que o *dataset* não continha alguns registos na variável que explica a decisão tomada acerca do animal. Em todos os casos em que na variável *OutcomeType* aparece o registo *return to owner*, isto é, o animal foi devolvido ao dono e não existe valor na variável *OutcomeSubType*, decidiu-se colocar a string "*Found*", uma vez que o animal foi efetivamente encontrado.

Em todos os casos em que a variável *OutcomeSubType* não continha registo e na *OutcomeType* estavam presentes umas das restantes quatro opções (*adoption, transfer, euthanasia, died*), decidiu-se colocar a *string Unknown*, visto que não se conseguia saber o porquê da decisão tomada. Um dos registos que mais nulos continha era a categoria *Adoption* e como o grupo não tinha como saber a razão dos animais serem adotados, decidiu-se colocar o sub-tipo como desconhecido.

5.3.2 Tratamento dos dados relativos à cor

A variável *color* retrata a cor dos animais do abrigo. Esta variável continha imensos registo diferentes, pelo que foi necessário criar uma nova variável "*new color*". Esta nova variável ficou apenas com 12 registo de cores diferentes, presentes na tabela 5.4.

Tabela 5.4: Tabela com as novas cores

| New Color | | | |
|-----------|------------|-------|---------|
| Cream | Two colors | Black | Brindle |
| Gray | Tricolor | Merle | Brown |
| White | Siamese | Tabby | Orange |

5.3.3 Tratamento dos dados relativos à raça

A variável *breed* permite saber a raça dos animais. Tal como com a cor, também esta variável continha imensos registo diferentes. De forma a simplificar a análise, decidiu-se criar uma nova variável em que apenas constasse se o animal é de raça pura ou não. Assim, criaram-se apenas dois tipos *breed* ou *mixed breed*.

5.3.4 Tratamento dos dados relativos à idade

A variável que indica a idade dos animais continha alguns *missing values*, assim como animais com idade zero. De forma a tornar os dados consistentes, foram eliminados todos os registo em que as duas situações descritas anteriormente aconteciam.

Depois de excluir a idade zero e os valores em falta, foi decidido transformar a idade para representá-la em dias, meses ou anos, dependendo do caso.

5.3.5 Tratamento dos dados relativos à variável *DateTime*

A variável *datetime* representa a dia e a hora a que foi decidida a situação futura do animal, ou seja, quando se decidiu em qual das categorias da variável *OutcomeType* o animal se insere. Assim, os dados foram transformados de *string* para *date & time* de forma a obter a data por dia da semana, mês e ano e a hora exata a que ocorreu a decisão.

5.3.6 Filtragem de colunas

Nem todas as colunas são importantes para o modelo, daí ser preciso filtrar algumas de forma a tornar os dados mais consistentes.

- *Name*: a coluna com os nomes dos animais foi retirada uma vez que não se considerava relevante para o modelo a ser construído;
- *Color*: como foi criada uma variável nova para diminuir a quantidade de registo, decidiu-se excluir da coluna original do *dataset*.
- *Ano*: Foi retirado no final da preparação dos dados uma vez que da data é mais importante saber o dia e mês, sendo que o ano por si só não se considera relevante.

5.4 Modelação

5.4.1 Preparação do modelo

Após a exploração dos dados, é necessário preparar os dados de modo a corrigir as inconsistências destes. Assim, foram realizadas as transformações em seguida referidas.

5.4.1.1 Criação de novas cores

Como já referido anteriormente, devido ao facto da existência de demasiados registos de cores diferentes, muitos dos quais muito semelhantes e não apresentavam significância no modelo, criou-se uma variável adicional, "New_color", utilizando o nodo "Table creator" para a criação do dicionário de associação juntamente com um nodo *Binner (Dictionary)*, que permite a associação correta a todo o dataset, figura 5.4.

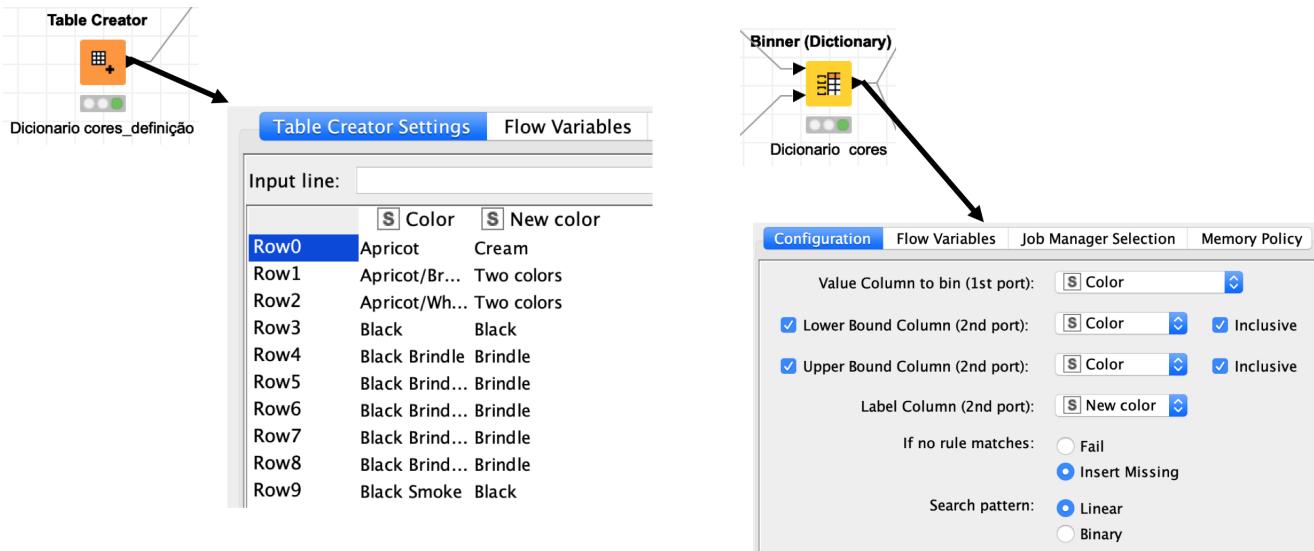


Figura 5.4: Table Creator com a definição de novas cores

5.4.1.2 Missing values

Como já referido anteriormente, o dataset em questão possuía vários *missing values* (?). Para tratar destes valores utilizou-se um conjunto de nodos representados em seguida, figura 5.5.

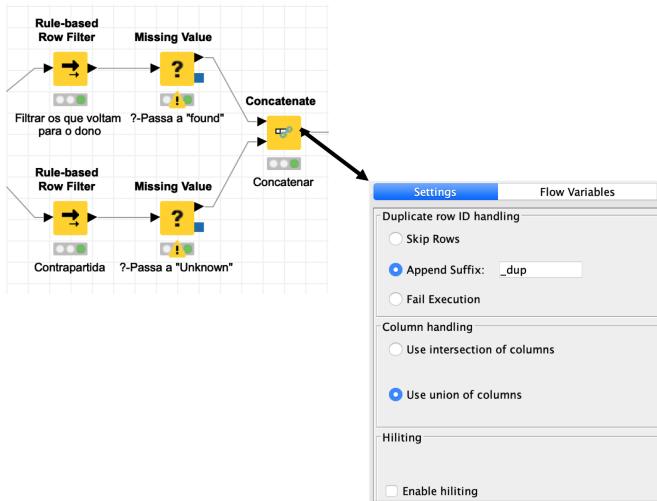


Figura 5.5: Substituição de valores em falta e concatenação

Uma vez que em todos os registos da *OutcomeType* que aparece a string *return to owner*, a variável *OutcomeSubType* apenas possui *missing values*, assim, através do nodo "Rule-based Row filter", filtraram-se todos esses

registos, substituindo os *missing values* pela string "Found", com os respetivos *outputs* representados na figura 5.6.

The diagram illustrates the replacement of missing values with the string "Found". It shows two tables side-by-side. The left table has 15 rows, all with the value "?" in the "OutcomeSubtype" column. An arrow points from the right side of this table to the left side of the second table. The second table also has 15 rows, but all the "?" values have been replaced by the word "Found" in the "OutcomeSubtype" column.

| | OutcomeType | OutcomeSubtype |
|-----------------|-------------|----------------|
| Return_to_owner | ? | ? |

| | OutcomeType | OutcomeSubtype |
|-----------------|-------------|----------------|
| Return_to_owner | Found | Found |

Figura 5.6: Substituição de *missing values* pela string "Found"

Para além destes registos, ainda existiam *missing values* na variável *OutcomeSubType*, que, através do mesmo processo, tomaram como registo a string "Unknown".

5.4.1.3 Tratamento dos dados relativos à idade

Uma vez decidido que todos os animais com idade igual a "0 years" e com idade desconhecida, (?), iriam ser eliminados do *dataset*, utilizou-se um nodo "Rule-based Row filter" para os eliminar, como evidenciado na figura 5.7.

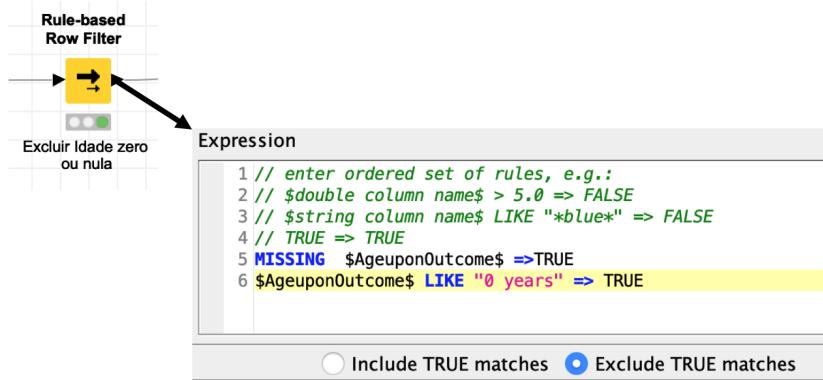


Figura 5.7: Exclusão de valores cuja idade é zero o valor em falta

5.4.1.4 Tratamento dos dados relativos à raça

Uma vez existirem mais de mil registos de raças diferentes no *dataset*, criou-se um dicionário de raças através do nodo "Table creator", criando assim, com a ajuda do nodo "Binner (Dictionary)", uma nova variável no modelo designada "nova raça", figura 5.8.

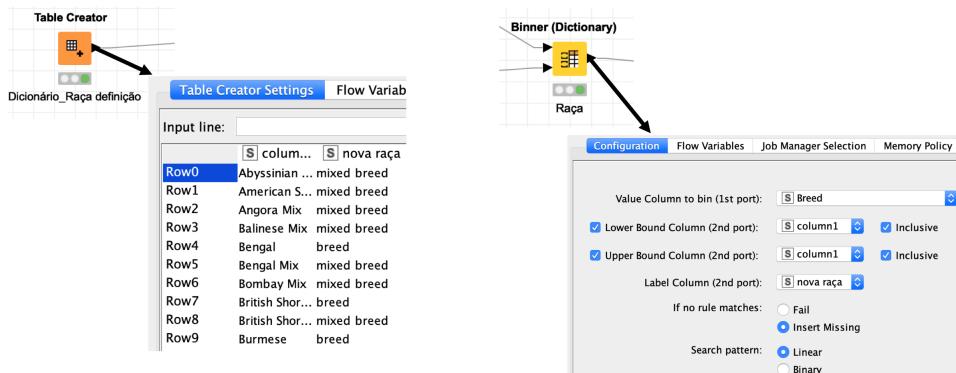


Figura 5.8: Criação de uma nova variável que permitiu diminuir a quantidade de raças

Para a criação do dicionário em questão foi necessário filtrar as linhas de acordo com regras pré-estabelecidas pelo grupo, uma vez serem registos de diferentes raças de elevado volume. Para tal, dividiu-se os registos de acordo com as seguintes regras:

- Sempre que a variável "Breed" apresentava duas raças na designação, ou seja, apresentava o símbolo "/", era designado "mixed breed", ou seja, o animal não era de raça pura.
- Sempre que a variável "Breed" apresentava "Mix" na descrição, era designado, também, como "mixed breed".
- Os restantes registos, que diziam respeito aos animais de apenas uma raça, ou seja, raça pura, foram designados de "breed".

5.4.1.5 Seleção de colunas

Uma vez realizada toda a transformação considerada necessária, procedeu-se à eliminação das colunas que não se consideravam relevantes para o modelo, figura 5.10

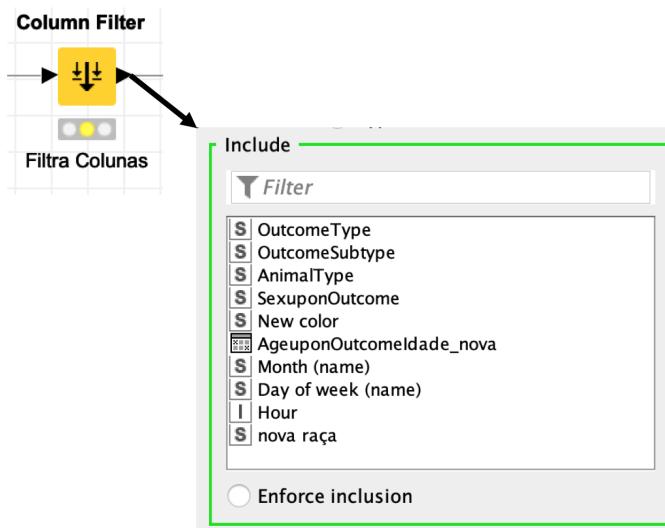


Figura 5.9: Seleção das colunas relevantes para previsão

5.4.1.6 Workflow completo da preparação de dados do dataset

Depois de todas as transformações acima referidas, obteve-se um *workflow* sequencial respeitante ao tratamento de dados do *dataset*, figura 5.10

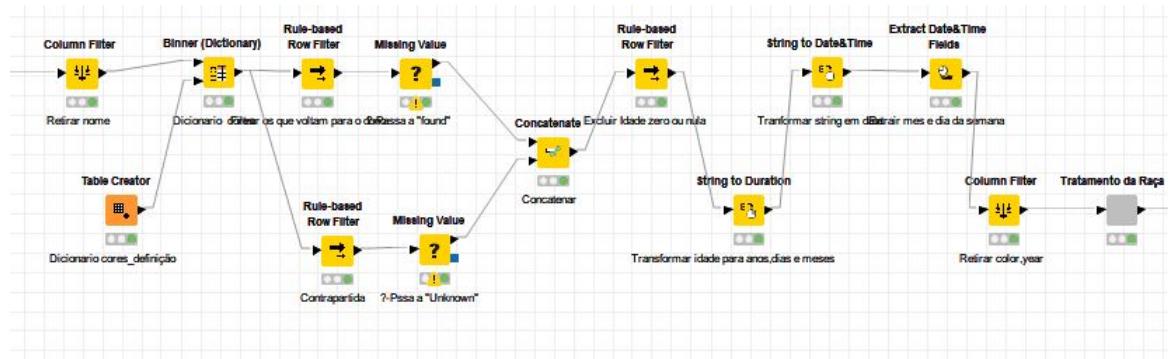


Figura 5.10: Worflow final da preparação dos dados do dataset, em KNIME

5.4.2 Tuning do modelo

O *tuning* utilizado para este conjunto de dados foi o mesmo usado na secção 4.4.3.1 referente à utilização do algoritmo *Random Forest*. Também para o algoritmo *Decision Tree* foram utilizadas as mesmas configurações

presentes na secção 4.4.3.2. Estes dois algoritmos têm como finalidade encontrar os parâmetros que permitem treinar o modelo e obter um menor erro na previsão. De referir que, neste caso, os dados para treino e para teste foram divididos utilizando um nodo *partitioning* dos quais 80% foram dados de treino e 20% dados de teste.

5.5 Resultados Obtidos

Após o tratamento de dados e o *tuning* efetuado, foi possível registar valores de erro nas previsões.

Em primeiro lugar, optou-se por treinar o modelo sem tratamento de dados e utilizou-se o algoritmo *Decision Tree*. Pela tabela 5.5 observamos que o valor do erro de previsão ronda os 22%, para todos os parâmetros iterados.

Tabela 5.5: Resultados obtidos com o algoritmo *Decision Tree* e sem tratamento dos dados

| Critério de paragem | Obj. Value | Split Criteria | Pruning |
|---------------------|------------|----------------|------------|
| 8 | 22.293 | Gini index | MDL |
| 11 | 22.34 | Gini index | No pruning |
| 8 | 22.579 | Gain ratio | No pruning |
| 10 | 22.649 | Gain ratio | MDL |

Em segundo lugar, optou-se por treinar o modelo sem tratamento de dados, mas desta vez usando o algoritmo *Random Forest*, e como é possível verificar na tabela 5.6, o erro diminuiu no melhor caso, cerca de 9%, comparando com o algoritmo *Decision Tree*.

Tabela 5.6: Resultados obtidos com o algoritmo *Random Forest* e sem tratamento dos dados

| | Information Gain | Information Gain | Ratio |
|--------------------------------|------------------|------------------|--------|
| Limit number of levels | 17 | 19 | 19 |
| Minimum child node size | 5 | 4 | 3 |
| Number of models | 100 | 200 | 100 |
| Objective value | 13.581 | 15.166 | 14.586 |

Por fim, treinou-se o modelo com os dados tratados como mencionado no capítulo 5.3. De acordo com os dados presentes na tabela 5.7, é possível verificar que se atingiram erros extremamente baixos, em particular o mais baixo é de cerca de 0,1%.

Tabela 5.7: Resultados obtidos com o algoritmo *Random Forest* e com tratamento dos dados

| | Information Gain | Information Gain | Ratio |
|--------------------------------|------------------|------------------|-------|
| Limit number of levels | 13 | 20 | 19 |
| Minimum child node size | 2 | 3 | 2 |
| Number of models | 100 | 100 | 300 |
| Objective value | 0,131 | 0,201 | 1.522 |

Para concluir, os valores de erros obtidos utilizando o algoritmo *Random Forest* e com tratamento de dados foram efetivamente bastante baixos, provavelmente devido ao facto de no tratamento dos dados ter-se eliminado todas as lacunas dos dados e também pelo facto de o *dataset* ter um elevado número de entradas de dados, em comparação com a quantidade de dados fornecida no primeiro *dataset* relativo à previsão de trânsito na cidade do Porto.

6. Conclusões

Tendo em conta os resultados obtidos para ambos os *datasets*, a utilização de árvores de decisão é capaz de solucionar uma série de problemas, uma vez que apresenta inúmeras vantagens, nomeadamente, a apresentação de um processamento relativamente rápido e com uma precisão bastante favorável.

Foi também interessante entender que a utilização de árvores de decisão e de *Random Forest* culminam em resultados bastante distintos. Em virtude dos factos mencionados, da análise das várias tentativas e também dos correspondentes resultados, utilizando o mesmo método mas com dados previamente preparados, origina resultados diferentes. Desta forma, entende-se que tanto o tratamento de dados como a escolha do algoritmo são aspectos cruciais para uma boa previsão. Por outro lado, o conteúdo do *dataset* original tem influência na precisão do modelo, uma vez que este conter dados que por si só pouca preparação necessitam.

O *dataset* referente ao trânsito da cidade do Porto apresentava vários desafios, alguns dos quais foram ultrapassados com a ajuda de uma cuidada preparação de dados, que trouxeram melhorias na precisão do modelo. Apesar de não existir nenhuma variável com um coeficiente de correlação extremamente elevado com a variável resposta, existiam algumas que logicamente foram alvo de maior cuidado por serem consideradas um fator de peso na previsão do trânsito na cidade do Porto. Assim, estas variáveis, nomeadamente a "*average_rain*", o "*average_time_diff*", o dia da semana bem com a hora foram alvo de um processamento cuidado para uma possível melhoria do modelo.

Relativamente aos resultados da previsão obtidos no *dataset* sobre o abrigo de animais, há uma elevada precisão dos resultados, provavelmente devido à variável *OutcomeSubType* que está altamente correlacionada com a variável resposta e também ao elevado número de dados utilizados para treino. Como muitos registos em que, simultaneamente, a variável *OutcomeSubType* não tinha valores e na *OutcomeType* correspondia a adoção de animais, foram substituídos pela string *unknown*, fez com que todos os registos associados a esta categoria necessitassem de maior cuidado no que diz respeito, nomeadamente, à idade, à raça bem como à cor e textura do pelo do animal.

Outro fator que ajuda a precisão do modelo é o facto de existirem um elevado número de dados utilizados para o treino do modelo (cerca de 21 mil).

No futuro, e tendo em vista uma melhoria de ambos os modelos, considera-se interessante aplicar técnicas de *data argumentation* para perceber melhor os *datasets* em estudo, bem como fazerem um tratamento mais cuidado. Para além disto seria crucial realizar mais testes ao modelo, com novas funcionalidades bem como novos tratamentos de dados.

Bibliografia

- [1] Han, J., Kamber, M., Pei, J.: Data Mining Concepts and Techniques. 23rd edition. Morgan Kaufmann, (2011)
- [2] Wirth, R., Hipp, J.: Towards a Standard Process Model for Data Mining. (2000)
- [3] From a Single Decision Tree to a Random Forest, <https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147>. Last accessed 22 Nov 2019
- [4] Quilan, J.R.: Induction of Decision Trees. Kluwer Academic Publishers. 81-106, (1986).