Universidade do Minho
Departamento de Informática

Aprendizagem e Decisão Inteligentes

Grupo 24

June 14, 2024

Carlos Ferreira
A89509

Gerson Junior
A88000

Pedro Sousa
A100823

Pedro Viana
A100701

Sumário

1 Introdução 3
2 Dataset 1 - ILPD (Indian Liver Patient Dataset) 4
2.1 Exploração dos Dados 4
2.2 Preparação do Dataset 6
2.2.1 Column Rename e Rule Engine 6
2.2.2 Missing Value 6
2.2.3 Numeric Outliers 7
2.2.4 Column Filter 8
2.3 Aprendizagem do Dataset 9
2.3.1 Decision Tree 9
2.3.2 Logistic Regression 11
2.3.3 Cluster 12
2.3.4 Random Forest 13
2.3.5 Tree Ensemble 13
2.3.6 Naive Bayes 14
2.3.7 Gradient Boosted 14
2.3.8 Redes Neuronais 15
2.4 Conclusões sobre o dataset 17
3 Dataset 2 - Car Price Prediction Challenge 18
3.1 Exploração dos Dados 18
3.2 Preparação do Dataset 19
3.2.1 Java Snippet 19
3.2.2 String to Number 19
3.2.3 Group Outliers 19
3.2.4 Row Filter 20
3.2.5 Model to Numeric 20
3.3 Aprendizagem do Dataset 20
3.3.1 Linear Regression 20
3.3.2 Decision Tree 22
3.3.3 Random Forest 22
3.3.4 Polynomial Regression 23
3.3.5 Redes onaisrais 23
3.4 Conclusão sobre o dataset 24
4 Conclusão 26
5 Bibliografia 27

1 Introdução

Este relatório documenta o processo de desenvolvimento do trabalho prático da unidade curricular de Aprendizagem e Decisão Inteligentes, realizado durante o ano letivo 2023/2024. O trabalho prático consistiu na análise, tratamento e exploração de dois conjuntos de dados distintos, fornecidos pela equipa docente.

O primeiro conjunto de dados, denominado ILPD (Indian Liver Patient Dataset), contém informações sobre pacientes e a presença ou ausência de doença no fígado. O objetivo foi realizar o tratamento dos dados, análise exploratória e, se possível, construir modelos de previsão ou classificação relacionados à saúde hepática.

Além disso, como parte do trabalho, foi solicitado que escolhêssemos um segundo conjunto de dados para ser tratado da mesma maneira. Optamos pelo desafio de previsão de preço de carros (Car Price Prediction Challenge), o que nos permitiu aplicar as técnicas aprendidas num contexto diferente e expandir a nossa compreensão e habilidades em aprendizagem de máquinas.

Ao longo deste relatório, detalharemos os processos e metodologias empregados na análise e tratamento destes conjuntos de dados, bem como os resultados obtidos e as conclusões alcançadas.

2 Dataset 1 - ILPD (Indian Liver Patient Dataset)

Este dataset consiste em 584 registos de pacientes recolhidos na região Nordeste de Andhra Pradesh, Índia. A mortalidade, na zona, por cirrose hepática está em ascensão devido ao aumento das taxas de consumo de álcool, infeções crónicas por hepatite e doenças hepáticas relacionadas à obesidade. Embora a deteção precoce seja crucial para melhorar os resultados dos pacientes, há evidências de que as pacientes do sexo feminino podem ser marginalizadas nesse processo. O dataset contém informações sobre diversos marcadores bioquímicos, incluindo albumina e outras enzimas metabólicas, que são utilizadas para prever se um paciente sofre de doença hepática. A análise desses dados pode fornecer *insights* importantes para melhorar a deteção precoce e o tratamento das doenças hepáticas na região estudada.

2.1 Exploração dos Dados

Na exploração dos dados, inicialmente, utilizamos os nodos *Data Explorer* e *Statistics* para verificar as estatísticas descritivas e identificar possíveis valores atípicos. Para os valores categóricos, e também examinar a presença de erros ortográficos.

Columns: 16	Column Type	Column Ind...	Color Hand...	Size Handler	Shape Han...	Filter Handl...	Lower Bound	Upper Bou...	Value 0	Value 1	Value 2	Value 3	Value 4
Age	Number (integer)	0					4	90	?	?	?	?	?
birth_year	Number (integer)	1					1,933	2,019	?	?	?	?	?
birth_month	Number (integer)	2					1	11	?	?	?	?	?
birth_date	String	3					?	?	?	?	?	?	?
Gender	String	4					?	?	Female	Male	female	Masculine	male
TB	Number (double)	5					0.4	75	?	?	?	?	?
DB	Number (double)	6					0.1	19.7	?	?	?	?	?
Alkphos	Number (integer)	7					63	2,110	?	?	?	?	?
Sgpt	Number (integer)	8					10	2,000	?	?	?	?	?
Sgot	Number (integer)	9					10	4,929	?	?	?	?	?
TB (#1)	Number (double)	10					2.7	9.6	?	?	?	?	?
ALB	Number (double)	11					0.9	5.5	?	?	?	?	?
CHOL	Number (integer)	12					0	0	?	?	?	?	?
AG_Ratio	Number (double)	13					0.3	2.8	?	?	?	?	?
BILmg	Number (double)	14					0.023	4.386	?	?	?	?	?
Selector	String	15					?	?	1=liver dis...	2=no liver ...	2=without l...	?	?

Figura 1: Estatistica do Dataset

De seguida, para identificar *outliers* nos valores ordinais, empregamos o nodo *Box Plot (JavaScript)*. Este método permite uma visualização (Figura 2) eficaz das distribuições dos dados e destaca quaisquer pontos que estejam significativamente afastados da maioria dos outros valores.

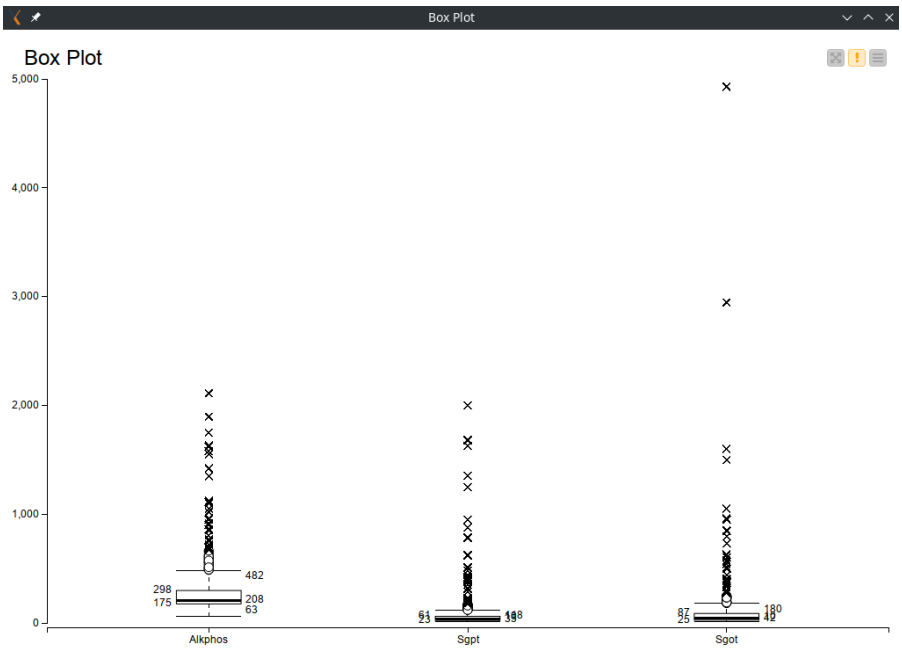

Figura 2: Box Plot do Alkphos, Sgpt e Sgot

Depois disto, para verificar a correlação entre as variáveis, utilizamos o nodo *Linear Correlation*. No entanto, não foi possível visualizar corretamente esta correlação, pois é necessário normalizar alguns valores e corrigir os dados ausentes. Portanto, antes de utilizar o nodo *Column Filter*, para filtrar colunas do dataset, foi necessário executar este pré-processamento de modo a verificar corretamente a correlação entre as variáveis.

De seguida, utilizamos o nodo *Scatter Plot (legacy)* para examinar visualmente (Figura 3) se existem padrões distintos entre as variáveis, distinguindo entre os pacientes diagnosticados com a doença hepática e os não diagnosticados. Esta abordagem gráfica é valiosa para identificar possíveis relações entre as variáveis e sua influência na presença ou ausência da doença.

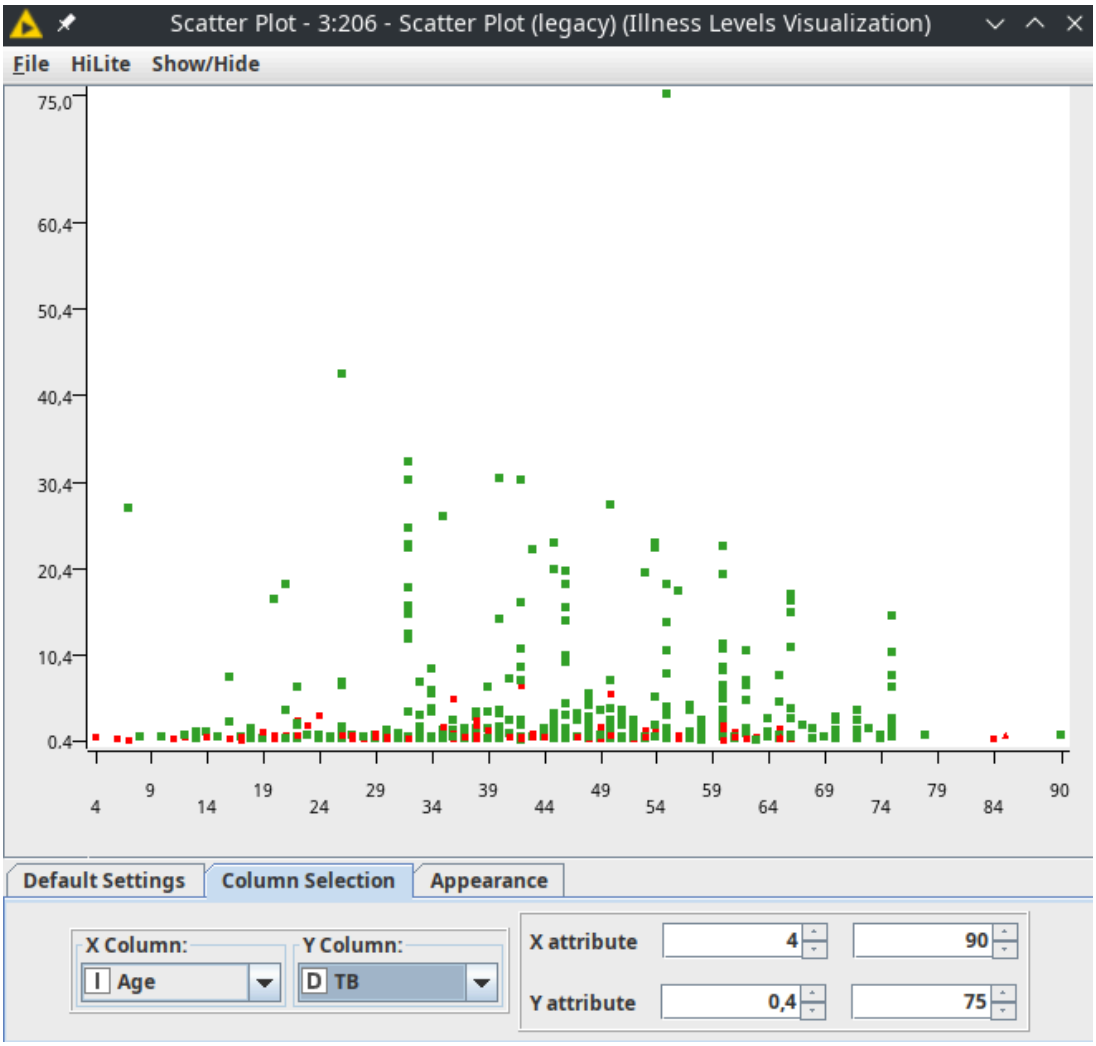

Figura 3: Exemplo de um dos Scatter Plots

2.2 Preparação do Dataset

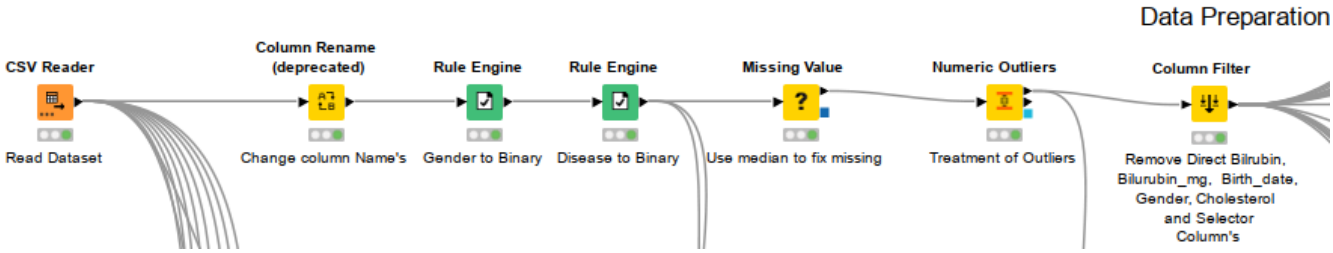

Figura 4: Preparação dos Dados

2.2.1 Column Rename e Rule Engine

Como podemos observar na Figura 4, iniciámos a preparação do *dataset* renomeando as colunas através do nodo *Column Rename*, tornando-as mais facilmente manipuláveis. De seguida, normalizámos e corrigimos os erros ortográficos das colunas do tipo *string* (como podemos ver na Figura 1) utilizando o *Rule Engine*. Desta forma, os valores foram convertidos para *booleanos*, ou seja, criámos uma coluna com **TRUE** ou **FALSE** para indicar se o paciente está doente ou não, e uma coluna chamada "male" com **1** e **0** para indicar se o paciente é do sexo masculino ou não.

2.2.2 Missing Value

Depois disto, demos início ao tratamento dos valores ausentes utilizando o nodo *Missing Value* (Figura 5). Considerando que estamos a lidar com um *dataset* relativamente pequeno, optámos por não realizar qualquer filtragem de linhas do dataset, já que isso poderia comprometer a fiabilidade dos dados. Em vez disso, criámos uma abordagem para testar qual seria o melhor método para os atributos em falta.

Para tal, implementámos uma pequena *Decision Tree* (Figura 5) para avaliar qual dos métodos resultaria numa maior precisão. Esta abordagem permitiu escolher a estratégia mais adequada para lidar com os valores ausentes, mantendo a integridade e a utilidade dos dados para análises posteriores.

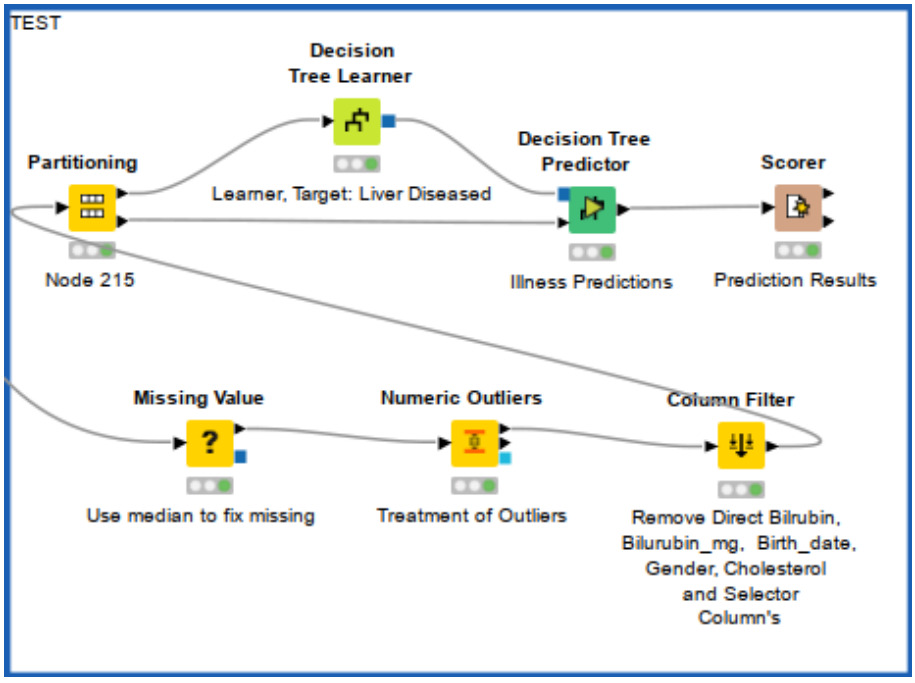

Figura 5: Zona de teste

Tipo	**Precisão**	**Cohen's kappa**
Média	79,452%	0.363%
Mediana	79,452%	0.363%
Média Arredondada	79,452%	0.363%
...		

Tabela 1: Tabela com os resultados

Pela Tabela 1, constatámos que não há diferença significativa na precisão entre os valores escolhidos para substituir os valores ausentes. Portanto, optámos por uma delas sem critérios específicos.

2.2.3 Numeric Outliers

De seguida, realizámos o tratamento dos *outliers* de valores ordinais utilizando o nodo *Numeric Outliers* (Figura 6 e Figura 7). Identificámos a presença desses *outliers* por meio de diversos *Box Plots* realizados, conforme pode ser observado na Figura 2. Seguindo a mesma lógica adotada para os valores ausentes, optámos por não filtrar as linhas do *dataset* e tratamos os valores discrepantes atribuindo-lhes o valor mais próximo permitido. Além disso, utilizámos um cálculo estatístico adequado, considerando se o paciente tem ou não a doença, para determinar os valores de substituição dos *outliers*. Este processo contribuiu para manter a integridade dos dados e evitar distorções nos resultados das análises subsequentes.

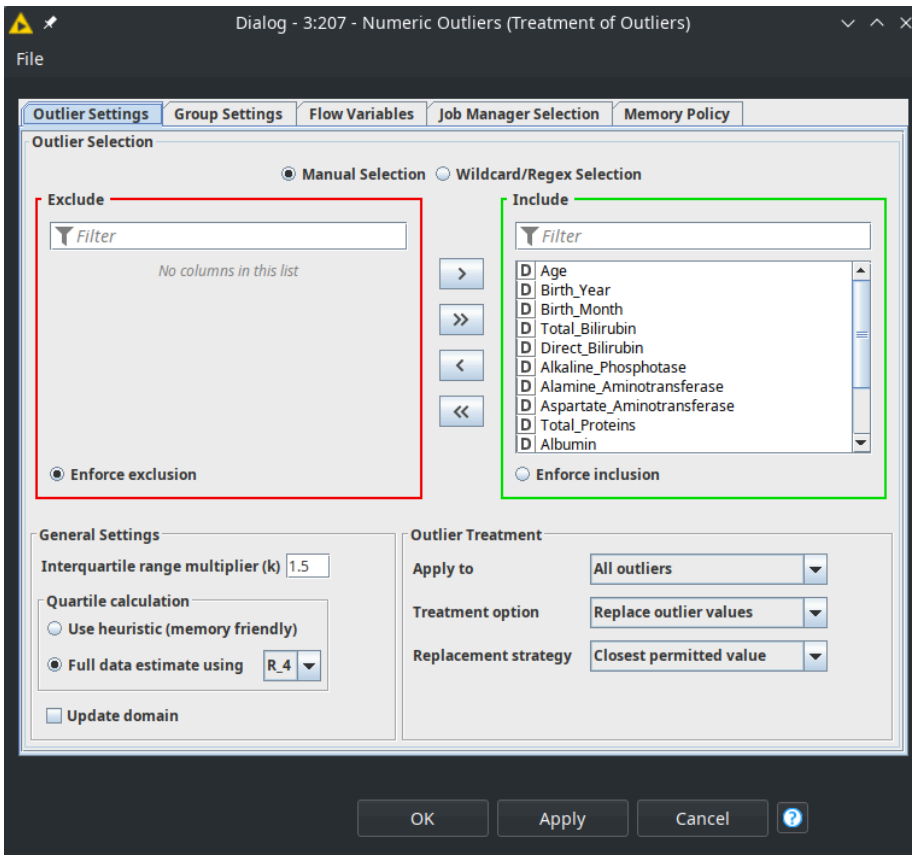

Figura 6: Configuração dos *outliers*

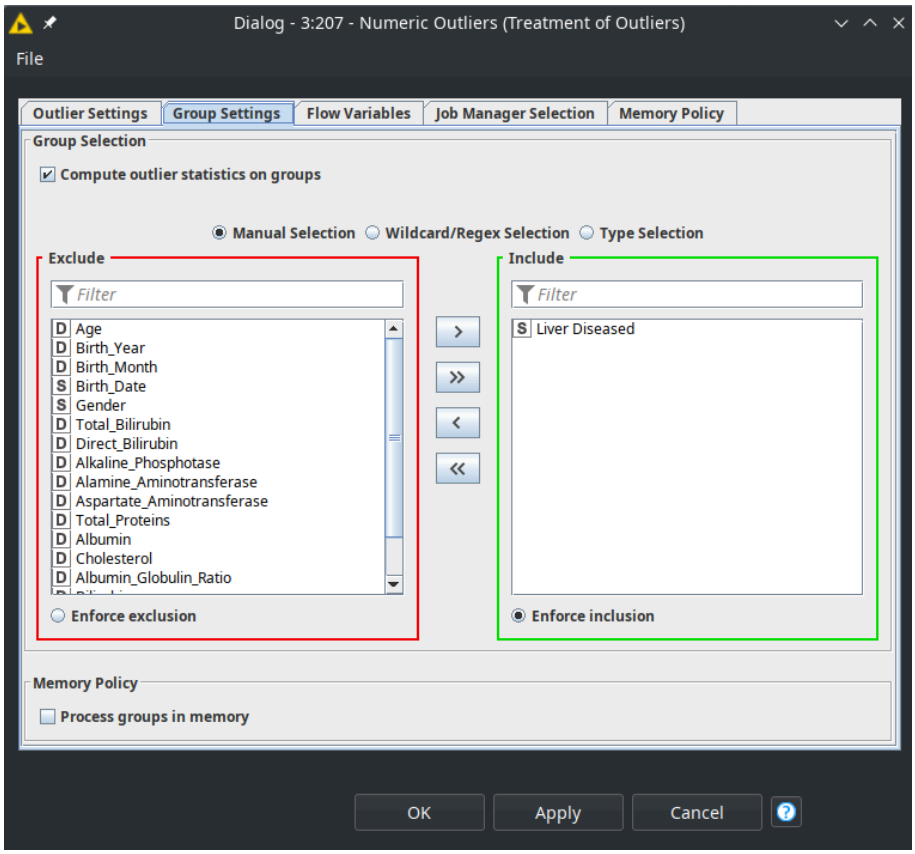

Figura 7: Configuração dos *outliers*

2.2.4 Column Filter

Como mencionado anteriormente durante a exploração dos dados, é importante verificar a correlação (Figura 8) entre as colunas do *dataset* antes de lhes aplicarmos qualquer tipo de filtro.

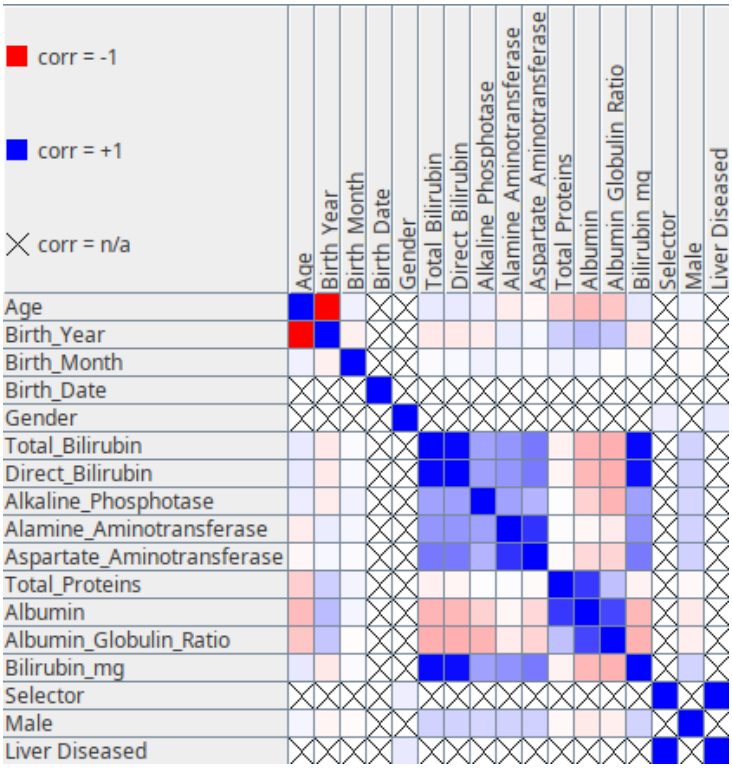

Figura 8: Correlação este as colunas

Após uma cuidadosa análise, observamos que algumas das colunas do dataset apresentam uma correlação bastante forte, ou seja, acima de 0.9. Este é o caso das colunas *Total_Bilirubin*, *Direct_Bilirubin* e *Bilirubin_mg*, indicando que estas fornecem informações muito semelhantes. Dessa forma, torna-se necessário filtrar duas destas colunas, a fim de evitar redundâncias nos dados. Optamos por manter a coluna *Total_Bilirubin*, pois esta possui a pior correlação com outras colunas, garantindo assim a preservação das informações relevantes.

Além disso, removemos as colunas *Gender* e *Selector*, uma vez que foram substituídas pelas colunas *Male* e *Liver Diseased*, respetivamente. Esta medida simplifica o *dataset*, eliminando variáveis redundantes.

Também foi filtrada a coluna *Cholesterol*, pois constatamos que esta possui apenas o valor "0", conforme pode ser observado na Figura 1. A remoção dessa coluna contribui para a limpeza e a simplificação dos dados, eliminando informações que não agregam valor ao conjunto de dados em análise.

2.3 Aprendizagem do Dataset

Antes de prosseguir com a discussão sobre cada um dos algoritmos utilizados para a aprendizagem do modelo e os seus resultados, começamos por explicar a razão pela qual optámos por utilizar os nodos *X-Partitioner* e *X-Aggregator* em vez do nodo *Partitioner* padrão. Isto deveu-se à necessidade de contar com funcionalidades mais avançadas e flexíveis na divisão e avaliação dos conjuntos de dados. Enquanto que o *Partitioner* padrão oferece apenas a capacidade básica de dividir o *dataset* em conjuntos de treino e teste, o *X-Partitioner* amplia essa funcionalidade, permitindo a criação de conjuntos de validação cruzada ("cross validation") e conjuntos de treino adicionais.

Abordaremos ainda o nodo *Scorer*. Este é utilizado para avaliar o desempenho dos modelos de aprendizagem de máquina, fornecendo uma variedade de métricas que nos permitem compreender como o modelo se está a desempenhar em termos da sua capacidade de fazer previsões precisas. Algumas das métricas mais comuns fornecidas pelo *Scorer* são:

1. **Correct classified** : Esta métrica indica o número de instâncias que foram classificadas corretamente pelo modelo. São os casos em que o modelo previu corretamente a classe verdadeira da instância.
2. **Wrong classified** : Esta métrica indica o número de instâncias que foram classificadas incorretamente pelo modelo. São os casos em que o modelo fez uma previsão errada em relação à classe verdadeira da instância.
3. **Accuracy** : A exatidão é uma medida geral do desempenho do modelo, representando a proporção de instâncias classificadas corretamente em relação ao total de instâncias. É calculada como o número de instâncias classificadas corretamente dividido pelo total de instâncias.
4. **Error** : O erro é a proporção de instâncias classificadas incorretamente em relação ao total de instâncias. É o complemento da exatidão e é calculado como 1 menos a exatidão.
5. **Cohen's Kappa (K)** : O coeficiente de Cohen's Kappa é uma medida de concordância entre duas classificações. Este leva em consideração o acordo entre as previsões do modelo e as classes verdadeiras, tendo em conta a possibilidade de concordância aleatória. É uma métrica útil, especialmente em casos de classes desequilibradas.

Sendo assim, **Accuracy** e o **Cohen's Kappa** são as métricas relevantes para verificar se um modelo é bom.

2.3.1 Decision Tree

A árvore de decisão é um algoritmo de aprendizagem supervisionada muito interpretável e fácil de entender. Este divide o conjunto de dados em subconjuntos menores com base em características es-

pecíficas, ajudando a identificar padrões e relações entre as variáveis. Optámos por experimentar este algoritmo devido à sua simplicidade e capacidade de lidar com conjuntos de dados complexos.

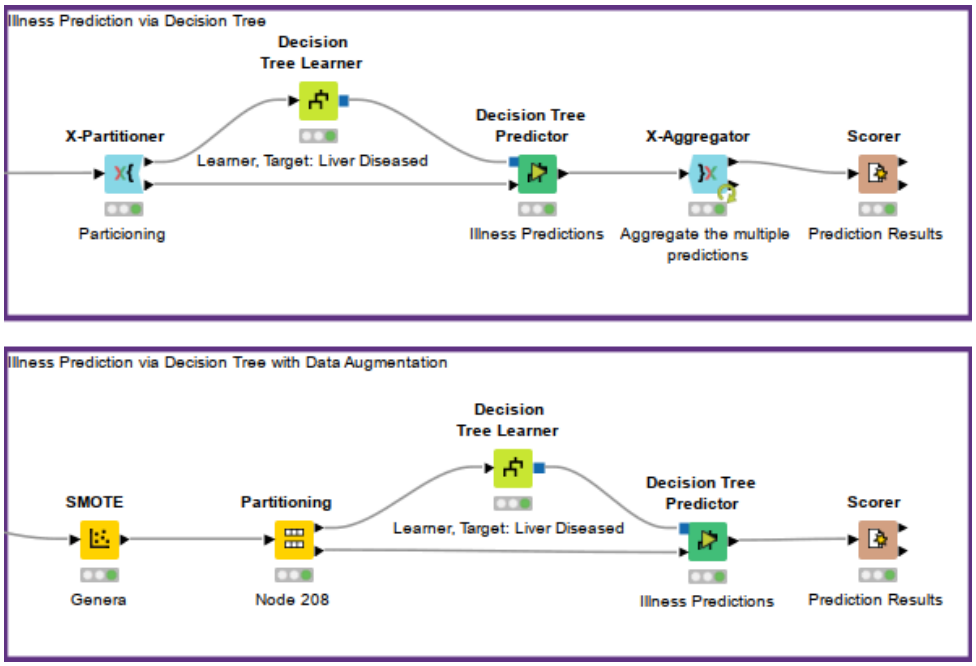

Figura 9: Decision Tree

Conforme ilustrado na Figura 9, nesta experiência, foram criados dois modelos: um utilizando um aumento dos dados fornecidos e outro sem. A seguir, apresentaremos a configuração utilizada para a árvore de decisão.

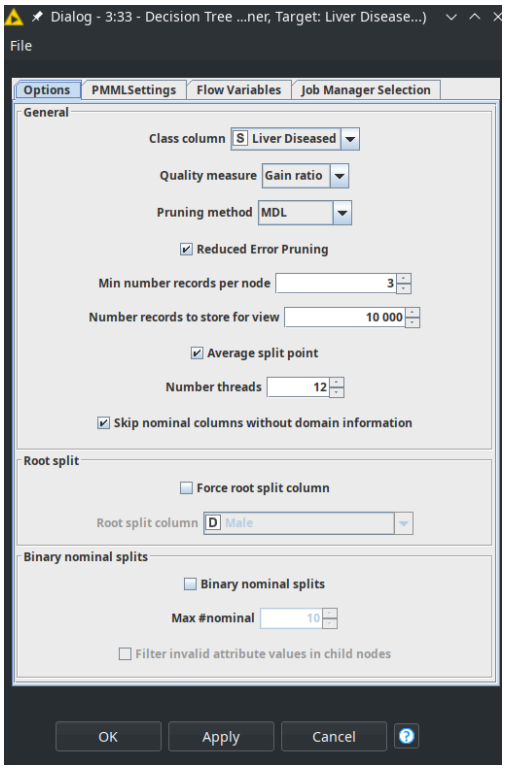

Figura 10: Configuração da Decision Tree

Com a configuração da árvore de decisão na Figura 10, sem o aumento de decisão tivemos os valores na Tabela 2:

Precisão	Cohen's kappa
79.76%	0.375%

Tabela 2: Resultados da Decision Tree

Isto significa que, em aproximadamente 80% das vezes, conseguimos prever corretamente o resultado esperado. No entanto, o resultado do coeficiente de Cohen's kappa é bastante baixo, o que indica uma concordância relativamente fraca entre os resultados apresentados.

2.3.1.1 Data Augmentation

Neste método, empregámos a técnica de aumento dos dados (*Data Augmentation*) para lidar com o desafio de um *dataset* com poucas linhas. Utilizámos o nodo *SMOTE* para aumentar a quantidade de dados de forma responsável, focando na coluna *Liver Disease* como alvo de aumento da minoria, ou seja, os casos em que a doença hepática não está presente (classificados como "FALSE"). O objetivo foi equilibrar as classes do *dataset*, aumentando a representação da classe minoritária e melhorando a capacidade do modelo de fazer previsões precisas para ambas as classes.

Precisão	Cohen's kappa
84.135%	0.684%

Tabela 3: Resultados da Decision Tree com Aumento de Dados

Na Tabela 3 podemos observar uma clara melhoria nos resultados em comparação com a abordagem anterior, especialmente no coeficiente de Cohen's kappa, que aumentou para 0.684%. No entanto, é importante destacar que o uso do aumento de dados nem sempre é a melhor opção, pois este pode levar a um aumento da complexidade do modelo e à introdução de ruído nos dados. Por isto, acreditámos que não seja benéfico a utilização do mesmo.

2.3.2 Logistic Regression

A regressão logística (Figura 11) é um algoritmo de aprendizagem supervisionada usado para problemas de classificação binária. Este calcula a probabilidade de um determinado evento ocorrer com base nas variáveis independentes. Escolhemos este algoritmo pela sua eficácia com dados binários e por ser fácil de interpretar, o que permite entender facilmente como as variáveis influenciam a classificação.

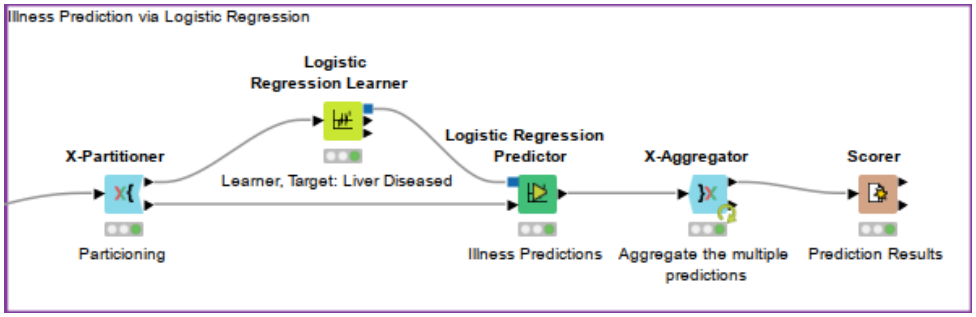

Figura 11: Logistic Regression

Precisão	Cohen's kappa
79,417%	0.41%

Tabela 4: Resultados da Logistic Regression

Pela Tabela 4 é importante notar que a precisão do modelo é relativamente alta, o que sugere que este é capaz de fazer previsões corretas na maioria das vezes. No entanto, o coeficiente de Cohen's kappa é relativamente baixo, indicando uma concordância fraca entre as previsões do modelo e as classes verdadeiras. Isso sugere que o modelo pode não estar tão bem ajustado aos dados ou que as classes estão desequilibradas, o que pode afetar a fiabilidade das previsões. Portanto, apesar da alta precisão, é essencial considerar outras métricas, como o coeficiente de Cohen's kappa, para avaliar completamente o desempenho do modelo.

2.3.3 Cluster

Os algoritmos de *clustering* (Figura 12) são usados para agrupar dados semelhantes em *clusters* distintos. Optámos por experimentar este algoritmo para explorar possíveis agrupamentos naturais nos dados e identificar padrões ou grupos de pacientes com características semelhantes.

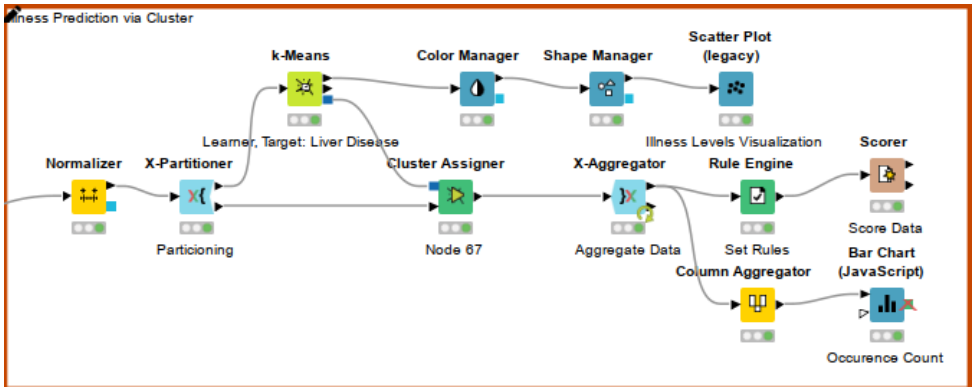

Figura 12: Cluster

Para realizar um *clustering*, removemos a coluna de *Liver Disease* e procuramos identificar padrões nos dados fornecidos para facilitar a deteção entre eles. Utilizámos o *Scatter Plot* para visualizar possíveis padrões nos dados e decidir quais *clusters* seriam atribuídos aos casos classificados como "TRUE" ou "FALSE", como evidenciado no *Bar Chart*.

Precisão	Cohen's kappa
51,286%	-0.083%

Tabela 5: Resultados do Cluster

Os resultados da Tabela 5 revelam a precisão e o coeficiente de Cohen's kappa obtidos pelo modelo de *clustering*. A precisão, que é de aproximadamente 51,286%, representa a proporção de instâncias que foram agrupadas corretamente nos seus respetivos *clusters*. Por outro lado, o coeficiente de Cohen's kappa, que é de −0.083%, é uma medida de concordância entre os *clusters* atribuídos pelo modelo e os *clusters* verdadeiros.

É preocupante notar que o coeficiente de Cohen's kappa é negativo, o que indica uma concordância muito fraca ou até mesmo uma discordância entre os *clusters* atribuídos pelo modelo e os *clusters* verdadeiros. Isso sugere que o modelo de *clustering* não conseguiu encontrar padrões significativos nos dados ou que a estrutura dos dados não é bem capturada pelos métodos de *clustering* utilizados.

Estes resultados indicam que o *clustering* pode não ser uma abordagem adequada para este conjunto de dados ou que a remoção da coluna de *Liver Disease* pode ter comprometido a capacidade do modelo de encontrar padrões relevantes nos dados. Em resumo, os maus resultados do *clustering* sugerem que os dados podem não ter uma estrutura natural que possa ser facilmente identificada e agrupada, ou que o modelo de *clustering* precisa de ser ajustado ou substituído por uma abordagem mais adequada.

2.3.4 Random Forest

O *Random Forest* (Figura 13) é um algoritmo de aprendizagem *ensemble* que cria várias árvores de decisão e combina os seus resultados para melhorar a precisão e reduzir o *overfitting*. Escolhemos este algoritmo devido à sua capacidade de lidar com grandes e complexos conjuntos de dados, além da sua robustez e flexibilidade.

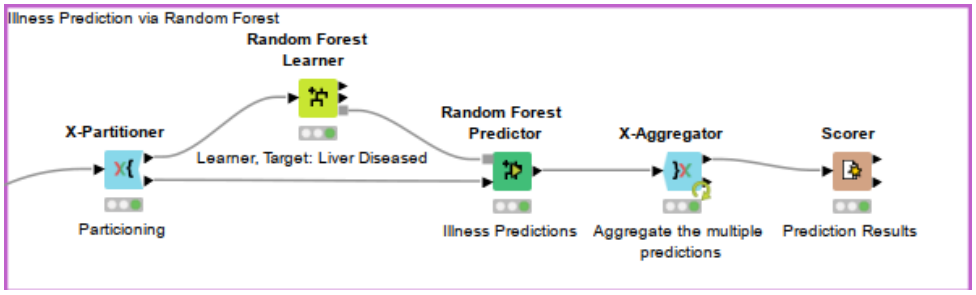

Figura 13: Random Forest

Precisão	Cohen's kappa
85,763%	0.63%

Tabela 6: Resultados do Random Forest

Os resultados da Tabela 6 são bastante encorajadores. A alta precisão sugere que o modelo *Random Forest* está a fazer previsões precisas na maioria dos casos, enquanto que o coeficiente de Cohen's kappa relativamente alto indica uma concordância razoável entre as previsões do modelo e as classes verdadeiras.

2.3.5 Tree Ensemble

Os algoritmos de *ensemble* (Figura 14), como o *Gradient Boosting*, combinam várias árvores de decisão para melhorar o desempenho preditivo. Optámos por experimentar esta técnica para aumentar a precisão das previsões e reduzir o enviesamento da modelagem associado a um único modelo.

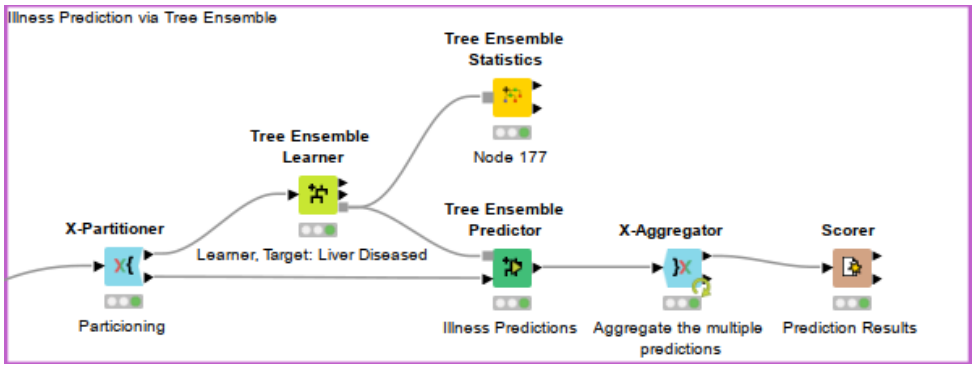

Figura 14: Tree Ensemble

Precisão	Cohen's kappa
85,763%	0.63%

Tabela 7: Resultados do Tree Ensemble

Pela tabela Tabela 7, apesar de os resultados serem relativamente inferiores ao modelo anterior, ainda são bastante satisfatórios. Estes resultados são semelhantes aos do modelo anterior, o que faz sentido, pois ambos são algoritmos de *ensemble*.

2.3.6 Naive Bayes

O classificador *Naive Bayes* (Figura 15) é um algoritmo simples de aprendizagem supervisionada baseado no teorema de Bayes. Este assume independência entre as variáveis e é eficaz para problemas de classificação com muitas características. Escolhemos este algoritmo devido à sua simplicidade e eficiência, especialmente em conjuntos de dados com muitas variáveis.

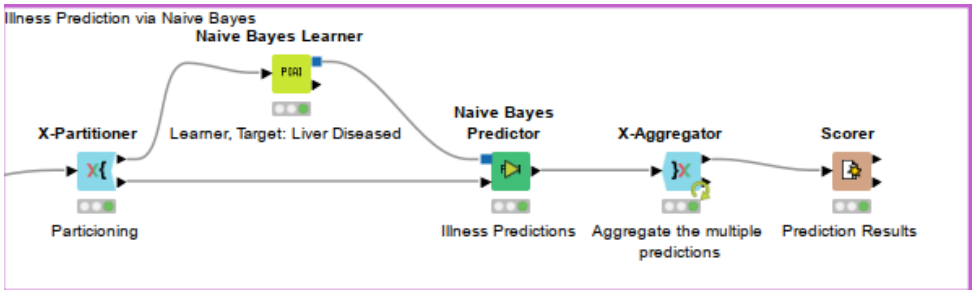

Figura 15: Naive Bayes

Precisão	Cohen's kappa
79,931%	0.379%

Tabela 8: Resultados do Naive Bayes

Os resultados da Tabela 8 são moderadamente satisfatórios. A precisão indica que o modelo *Naive Bayes* consegue fazer previsões corretas na maioria dos casos, enquanto que o coeficiente de Cohen's kappa sugere uma concordância razoável entre as previsões do modelo e as classes verdadeiras.

2.3.7 Gradient Boosted

O *Gradient Boosting* (Figura 16) é uma técnica de *ensemble* que combina várias árvores de decisão fracas para criar um modelo forte. Optámos por experimentar este algoritmo devido à sua capacidade de lidar com dados complexos e produzir modelos de alta precisão, mesmo em conjuntos de dados desequilibrados ou com ruído.

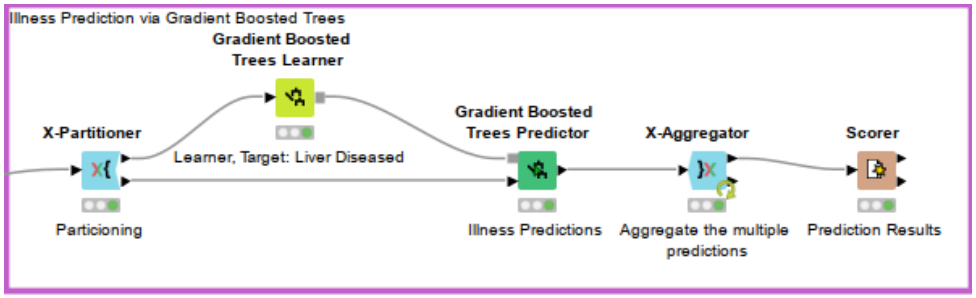

Figura 16: Gradient Boosted

Precisão	Cohen's kappa
83,533%	0.573%

Tabela 9: Resultados do Gradient Boosted

Os resultados na Tabela 9 são bastante promissores. A alta precisão sugere que o modelo *Gradient Boosted* faz previsões precisas na maioria dos casos, e o coeficiente de Cohen's kappa relativamente alto indica uma concordância robusta entre as previsões do modelo e as classes verdadeiras.

Em resumo, os resultados obtidos pelo modelo Gradient Boosted indicam que este é capaz de fazer previsões precisas e confiáveis para este conjunto de dados. Isso sugere que o *Gradient Boosted* pode ser uma escolha eficaz para tarefas de classificação em conjuntos de dados semelhantes.

2.3.8 Redes Neuronais

As redes neuronais (Figura 17) são modelos de aprendizagem de máquina fundamentados na estrutura e no funcionamento do sistema nervoso humano. Estes consistem em várias camadas de "neurónios" interconectados, cada uma realizando operações matemáticas em dados de entrada para produzir previsões ou classificações. A aplicação de redes neuronais é amplamente reconhecida.

Além disso, as redes neuronais frequentemente demonstram um desempenho notável em termos de precisão e generalização, especialmente em conjuntos de dados grandes e complexos. A sua capacidade de se adaptar a diferentes problemas e conjuntos de dados é uma vantagem adicional, permitindo ajustes na arquitetura da rede, na função de ativação e nos parâmetros para otimizar o desempenho do modelo.

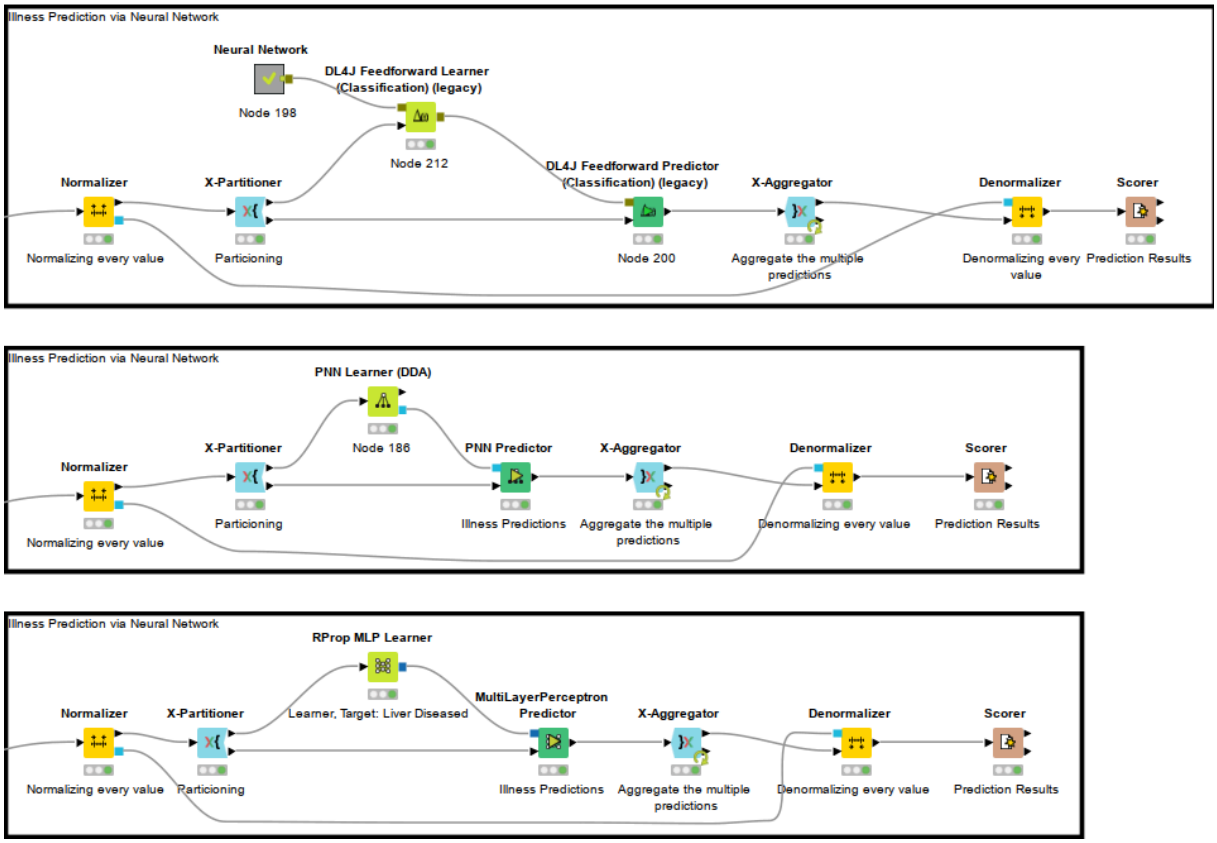

Figura 17: Redes Neuronais

2.3.8.1 RProp MLP

O nodo *RPorp MLP* (Figura 18) implementa uma rede neuronal artificial (RNA) com múltiplas camadas de neurónios, utilizando o algoritmo de retro propagação resiliente (*RProp*) para treino. O *RProp* é uma técnica de otimização de gradiente que ajusta os pesos da rede de forma adaptativa, baseando-se apenas na direção do gradiente, sem depender da sua magnitude. Isto torna-o particularmente adequado para treinar redes neuronais profundas, onde o problema do desaparecimento ou explosão do gradiente pode ocorrer. A utilização do *RProp MLP* pode ser vantajosa devido à sua capacidade de lidar com con-

juntos de dados complexos e de alta dimensão, bem como a sua eficiência em termos de convergência durante o treino.

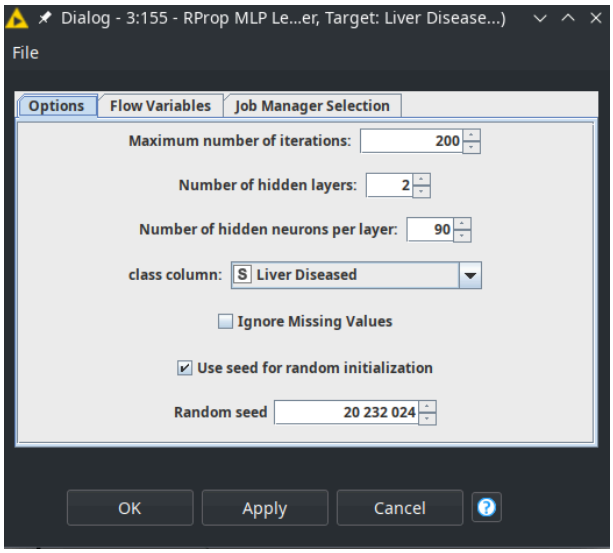

Figura 18: Configuração do RProp

Nesta configuração da rede neuronal, são definidas duas camadas ocultas, cada uma composta por 90 neurónios. Além disso, o treino da rede é conduzido por até 200 iterações.

Precisão	Cohen's kappa
78,988%	0.448%

Tabela 10: Resultados do RProp MLP

Os resultados da Tabela 10 indicam um desempenho relativamente bom do modelo *RProp MLP* na tarefa de classificação. A precisão em torno de 79% sugere que o modelo é capaz de fazer previsões corretas na maioria dos casos. Além disso, o coeficiente de Cohen's kappa, embora modesto, ainda mostra uma concordância significativa entre as previsões do modelo e as classes verdadeiras.

2.3.8.2 DL4J Feedforward

Este nodo permite a construção de uma rede neuronal muito semelhante à rede *RProp MLP*, no entanto, a diferença está na capacidade de configurar a rede neuronal de forma mais dinâmica. Ou seja, que é possível escolher exatamente o número de neurónios em cada camada, ajustar os hiperparâmetros e personalizar a arquitetura da rede conforme seja necessário. Esta flexibilidade oferece mais controlo sobre a construção da rede neuronal, permitindo adaptá-la de maneira mais precisa às características e necessidades do problema em questão e às nuances dos dados disponíveis.

No entanto, devido a essa flexibilidade e ao tempo necessário para processamento computacional, não foi possível realizar a otimização ideal do modelo. Porém, na Tabela 11, podemos ver os resultados do modelo:

Precisão	Cohen's kappa
73,07%	0.333%

Tabela 11: Resultados do DL4J Feedforward

Apesar de não serem os valores otimizados, verificamos que o modelo consegue prever a maior parte dos casos, porém, o coeficiente de Cohen's Kappa indica uma concordância relativamente fraca entre as previsões do modelo e as classes verdadeiras.

2.4 Conclusões sobre o dataset

Podemos concluir, com base nos resultados obtidos, que os algoritmos de *ensemble*, como o *Decision Tree*, o *Random Forest*, o *Tree Ensemble* e o *Gradient Boosted*, são os que oferecem os melhores resultados em termos de precisão e coeficiente de Cohen's kappa. Notavelmente, o *Random Forest* destaca-se como o melhor algoritmo para este conjunto de dados. Isto sugere que os dados tem uma estrutura que favorece a modelagem por meio de técnicas de *ensemble*, onde múltiplos modelos são combinados para produzir previsões mais precisas e estáveis.

Além disso, podemos observar o comportamento dos dados por meio do nodo *Scatter Plot Matrix* (Figura 19), onde podemos visualizar a forma como as variáveis se relacionam entre si e como são distribuídas em relação às diferentes métricas. Esta visualização ajuda a entender melhor a natureza dos dados e a identificar possíveis padrões ou *clusters*, contribuindo para uma análise mais completa e uma melhor interpretação dos resultados obtidos.

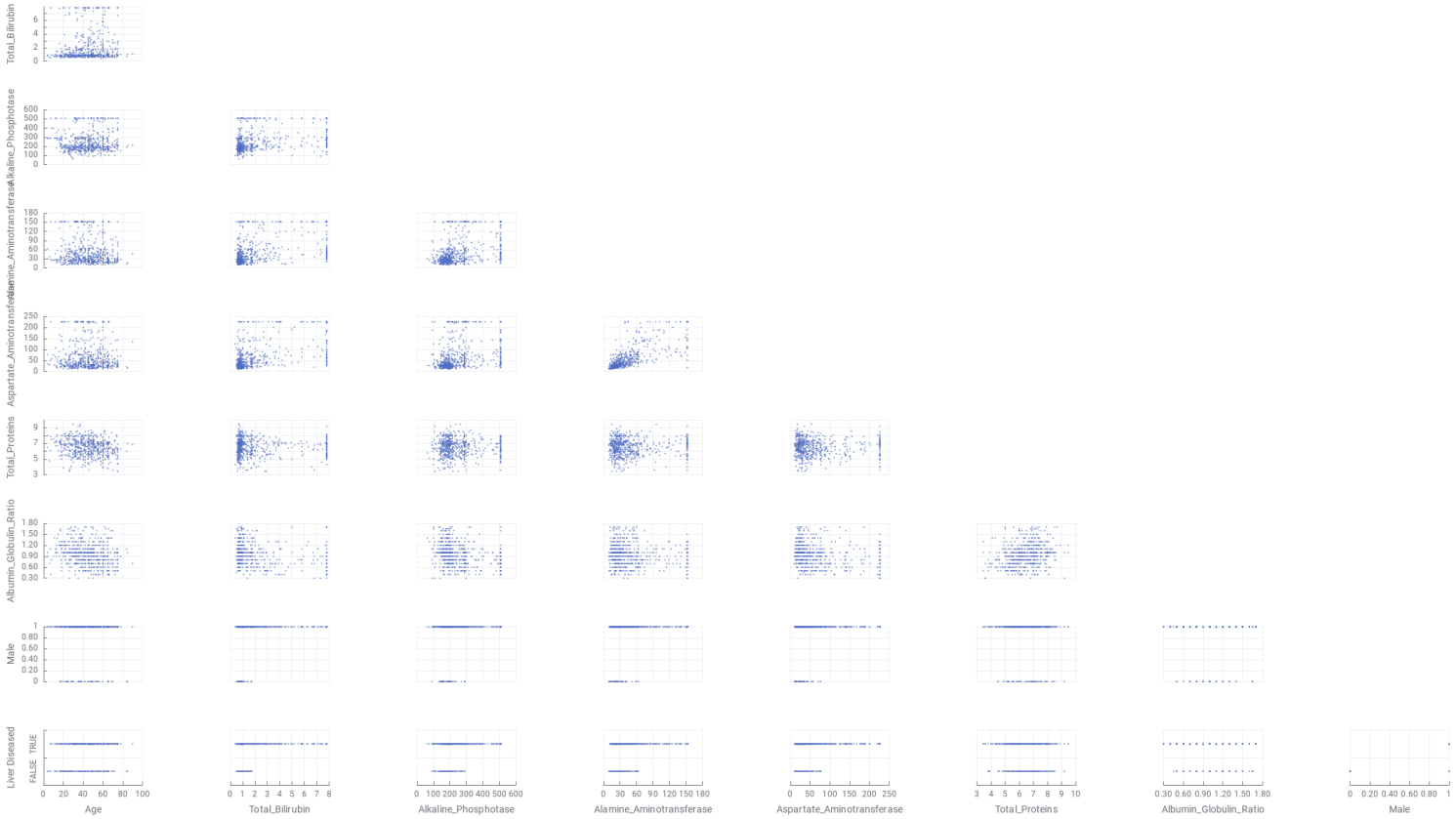

Figura 19: Comportamento dos Dados

3 Dataset 2 - Car Price Prediction Challenge

O *dataset* em questão é um conjunto de dados relacionados a carros, contendo informações detalhadas sobre várias características de veículos, como modelo, ano, quilometragem, tipo de combustível, entre outros. O objetivo deste *dataset* é fornecer uma base para a previsão do preço de carros com base nas suas caracterísi Este foi obtido em https://www.kaggle.com/datasets/deepcontractor/car-price-prediction-challenge.cas.

A previsão do preço de um carro é uma tarefa crucial para compradores, vendedores e profissionais do setor automóvel. Ao compreender as relações entre as diferentes características de um veículo e o seu preço final, é possível oferecer uma estimativa precisa e informada do valor de mercado de um carro.

Assim, este *dataset* oferece uma oportunidade valiosa para explorar e aplicar técnicas de aprendizagem de máquina na previsão de preços de carros, contribuindo para uma tomada de decisão mais informada e eficaz no mercado automóvel.

3.1 Exploração dos Dados

File

Table "default" - Rows: 19237 | Spec - Columns: 17 | Properties | Flow Variables

Columns: 17	Column Type	Column Ind...	Color Hand...	Size Handler	Shape Han...	Filter Handl...	Lower Bound	Upper Bou...	Value 0	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6	Value 7	Value 8	Value 9	Value 10	Value 11	Value 12	Value 13	Value 14	Value 15
Price	Number (integer)	0					1	26,307,500	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Levy	String	1					?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Manufacturer	String	2					?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Model	String	3					?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Prod. year	Number (integer)	4					1,939	2,020	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Category	String	5					?	?	Jeep	Hatchback	Sedan	Microbus	Goods wag...	Universal	Coupe	Minivan	Cabriolet	Limousine	Pickup	?	?	?	?	?
Leather interior	String	6					?	?	Yes	No	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Fuel type	String	7					?	?	Hybrid	Petrol	Diesel	CNG	Plug-in Hyb...	LPG	Hydrogen	?	?	?	?	?	?	?	?	?
Engine volume	String	8					?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Mileage	String	9					?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Cylinders	Number (double)	10					1	16	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Gear box type	String	11					?	?	Automatic	Tiptronic	Variator	Manual	?	?	?	?	?	?	?	?	?	?	?	?
Drive wheels	String	12					?	?	4x4	Front	Rear	?	?	?	?	?	?	?	?	?	?	?	?	?
Doors	String	13					?	?	04-May	02-Mar	>5	?	?	?	?	?	?	?	?	?	?	?	?	?
Wheel	String	14					?	?	Left wheel	Right-hand ...	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Color	String	15					?	?	Silver	Black	White	Grey	Blue	Green	Red	Sky blue	Orange	Yellow	Brown	Golden	Beige	Carnelian r...	Purple	Pink
Airbags	Number (integer)	16					0	16	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Figura 20: Estatisticas do dataset

Utilizámos os nodos *Statistics*, *Linear Correlation* e *Box Plot* com os mesmos objetivos que no conjunto de dados anterior, para examinar as estatísticas gerais, correlações entre as métricas e identificar *outliers*. No entanto, para detetar *outliers* em valores nominais, empregámos o nodo *Bar Chart (JavaScript)*.

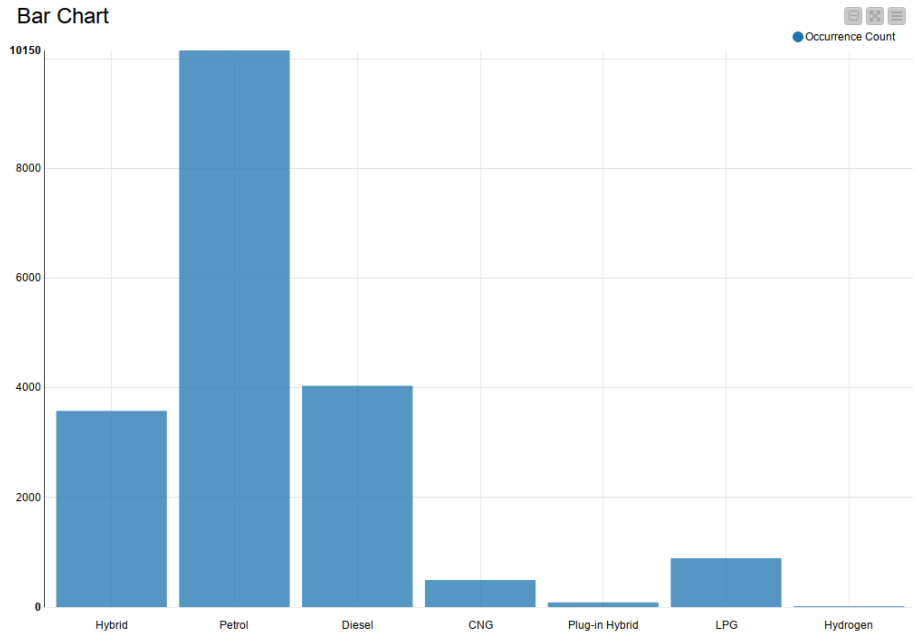

Figura 21: Ocorrência dos tipos de combustível

Por exemplo, neste gráfico de barras (Figura 21), podemos observar que o tipo de combustível "Hydrogen" é um *outlier*, pois possui apenas uma ocorrência em todo o conjunto de dados. Também utilizámos

este tipo de nodo para identificar outros tipos de *outliers*, como a marca do carro em relação à média de preços.

3.2 Preparação do Dataset

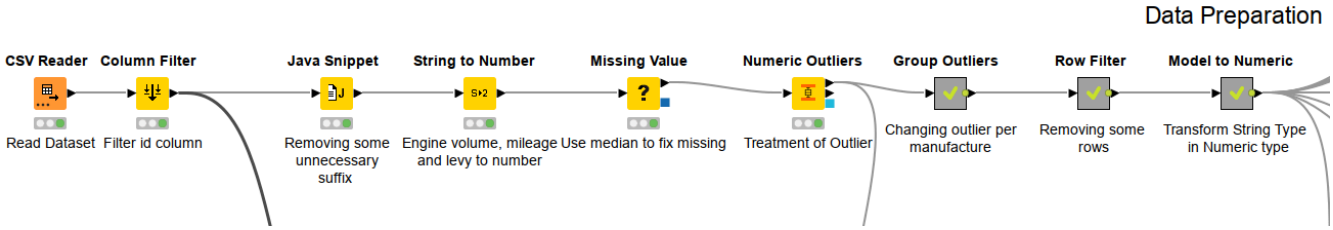

Figura 22: Preparação dos Dados

Os nodos *Missing Value* e *Numeric Outliers* (Figura 22) desempenham a mesma função que no *dataset* anterior, tratando valores ausentes e *outliers* "comuns".

3.2.1 Java Snippet

Utilizámos este nodo para remover alguns sufixos que não eram úteis, por exemplo, o sufixo "km" na coluna *Mileage*, para poder converter a string em número.

3.2.2 String to Number

A utilização deste nodo serve para transformar as colunas que continham sufixos não úteis em números, facilitando assim a manipulação das mesmas métricas.

3.2.3 Group Outliers

Inicialmente, não utilizávamos este método para remover *outliers*. No entanto, após analisar os dados, constatámos que existiam muitos *outliers*, principalmente ao examinar os preços, quilometragem, impostos e ano de fabricação por marca de carro. Como podemos observar no Figura 23:

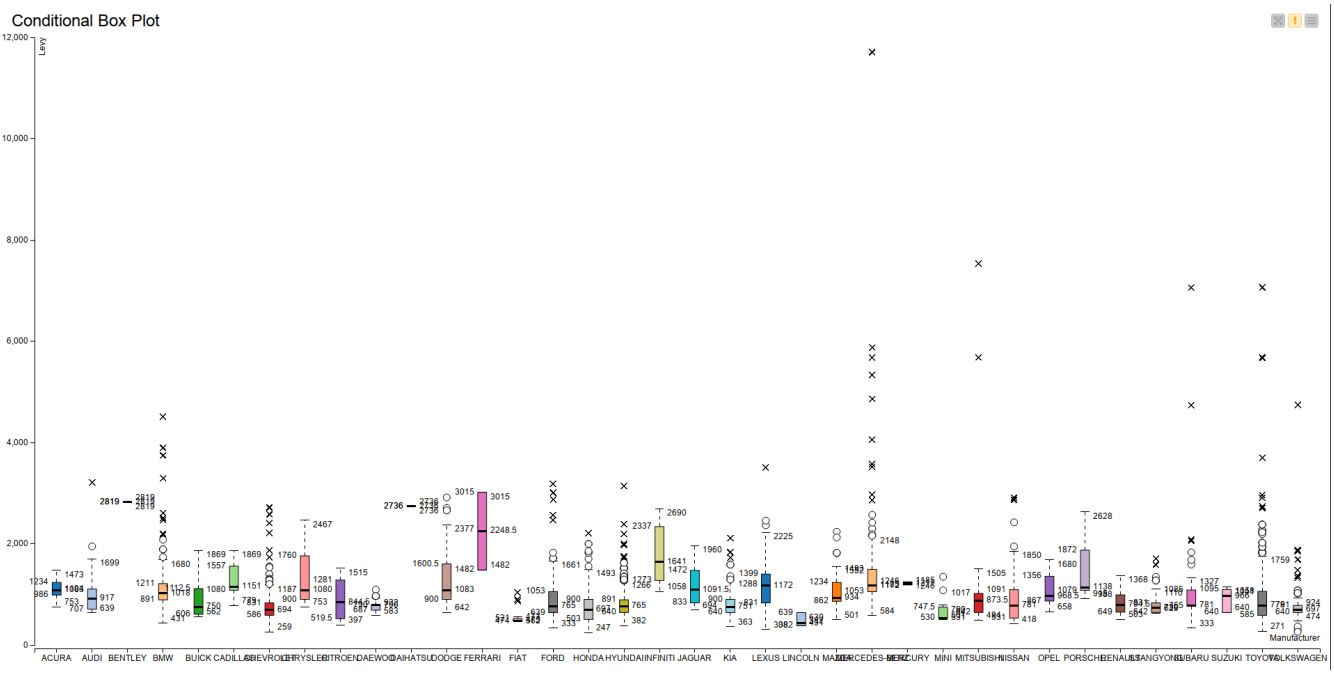

Figura 23: Exemplo dos outliers do imposto por marcas

Para resolver isto fizemos um "loop" (Figura 24) sobre cada uma das marcas de carro e retiramos estes *outliers*, sem perda de informação relevante.

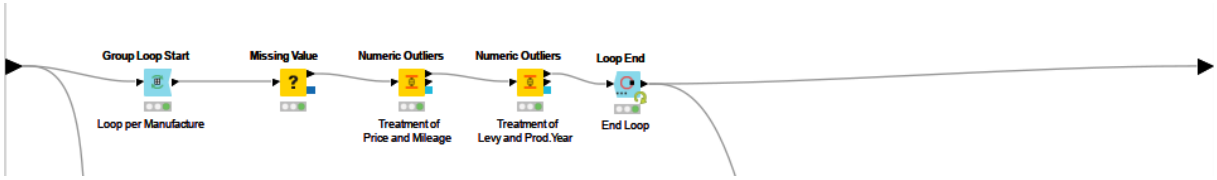

Figura 24: Group Outliers

3.2.4 Row Filter

Este conjunto de nodos serve para eliminar informações não realistas, por exemplo, carros a custar 1 euro, carros sem impostos, etc.

3.2.5 Model to Numeric

Como existiam vários tipos de modelos de carros, já que cada marca de carro possui "n" números de modelos diferentes, decidimos converter todos os modelos de carro para dados numéricos. Isto torna mais fácil para os algoritmos entenderem esta coluna.

3.3 Aprendizagem do Dataset

Continuo utilizando o *X-Partitioner* em vez do *Partitioner*, pelos mesmos motivos do conjunto de dados anterior. No entanto, como este é um problema de regressão, não utilizo o *Scorer*, mas sim o *Numeric Scorer*.

O nodo *Numeric Scorer* fornece diversas métricas para avaliar o desempenho de um modelo de regressão. Algumas das métricas comuns incluem:

1. R^2 **(R-squared)** : Uma medida da proporção da variância na variável dependente que é explicada pela variável independente no modelo. Quanto mais próximo de 1, melhor é o ajuste do modelo aos dados.
2. **Mean Absolute Error (MAE)** : A média dos valores absolutos das diferenças entre as previsões do modelo e os valores reais.
3. **Mean Squared Error (MSE)** : A média dos quadrados das diferenças entre as previsões do modelo e os valores reais.
4. **Root Mean Squared Error (RMSE)** : A raiz quadrada do MSE, proporcionando uma medida da dispersão dos erros.
5. **Mean Signed Difference** : A média das diferenças entre as previsões do modelo e os valores reais, levando em conta o sinal das diferenças.
6. **Mean Absolute Percentage Error (MAPE)** : A média das percentagens absolutas das diferenças entre as previsões do modelo e os valores reais, expressas como uma percentagem do valor real.
7. **Adjusted** R^2 : Uma versão ajustada do R^2 que leva em consideração o número de variáveis independentes no modelo. Essa métrica é útil para modelos com múltiplas variáveis independentes.

3.3.1 Linear Regression

A regressão linear (Figura 25) é um dos algoritmos de aprendizagem de máquina mais simples e amplamente utilizados. Este assume uma relação linear entre a variável dependente (a ser prevista) e uma ou mais variáveis independentes (usadas para fazer previsões). O modelo de regressão linear tenta encontrar a "melhor" linha reta que se ajusta aos dados, minimizando a soma dos quadrados das diferenças entre as previsões do modelo e os valores reais.

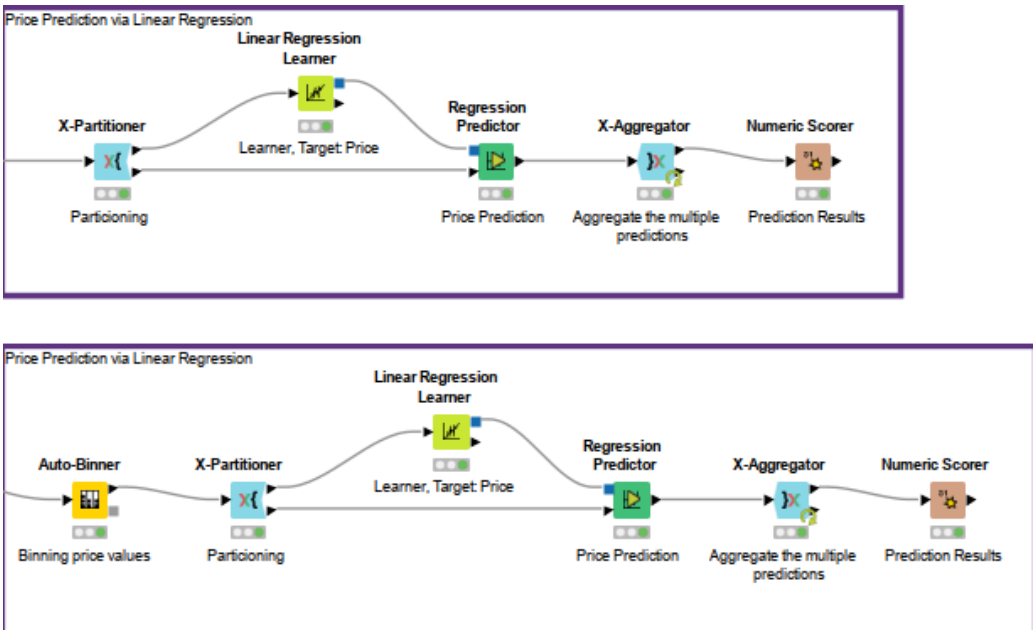

Figura 25: Linear Regression

R^2	Mean absolute error	Mean absolute percentage error
0,586	6562,116	1,436

Tabela 12: Resultados do Linear Regression

Pela Tabela 12, o coeficiente de determinação (R^2) é de aproximadamente $0,586$, indicando que o modelo é capaz de explicar cerca de $58,6\%$ da variabilidade nos preços dos carros com base nas características consideradas.

Além disso, a média do erro absoluto (*Mean absolute error*) é de $6562,116$ unidades monetárias, o que significa que, em média, as previsões do modelo estão erradas nesse valor em relação aos preços reais dos carros. Já o erro absoluto percentual médio (*Mean absolute percentage error*) é de 1.436%, indicando que, em média, as previsões do modelo estão erradas em cerca de $1,436\%$ em relação aos preços reais dos carros.

Esses resultados fornecem uma avaliação inicial do desempenho do modelo de regressão linear na previsão de preços de carros. Embora o modelo seja capaz de explicar uma parte significativa da variabilidade nos preços dos carros, ainda há espaço para melhorias, especialmente em relação à redução dos erros de previsão.

3.3.1.1 Com Binning

Nesta versão do modelo de regressão linear com *binning* dos preços, é gerada uma nova coluna que representa faixas de preço mais significativas. Isso é feito para mitigar a ampla variação observada nos preços no modelo sem *binning*, que vai de 559 euros a 47 mil euros. Ao aplicar *binning*, os preços são agrupados em intervalos mais amplos, o que pode fornecer uma representação mais precisa da relação entre o preço e outras variáveis independentes no modelo.

R^2	Mean absolute error	Mean absolute percentage error
0,989	1177,961	0,228

Tabela 13: Resultados do Linear Regression com Binning

Os resultados na Tabela 13 demonstram uma melhoria significativa no desempenho do modelo de regressão linear com *binning* em comparação com o modelo de regressão linear tradicional. O aumento substancial no coeficiente de determinação, juntamente com a redução considerável nos erros de previsão, sugere que a técnica de *binning* foi eficaz na captura de padrões nos dados e na melhoria da precisão das previsões de preços de carros.

3.3.2 Decision Tree

Neste modelo mudáo objetivo pe *do problema. Em vez de tentar resolver o problema quanto ao preço exato do carro, tentámos descobrir eque intervalo de preço o carro se encontra através de um_Decision Tree* (Figura 26) d.a.

Figura 26: Decision Tree

)

Accuracy	Cohen's Kappa
58,853%	0.529%

Tabela 14: Resultados do Decision Tree

)Os ses resultana Tabela 14 dos indicam um desempenho moderado do modelo de árvore de decisão na tarefa de classificação das faixas de preço dos carros. Embora a precisão seja relativamente baixa, o coeficiente de Cohen's Kappa mostra uma concordância razoável entre as previsões do modelo e as classes verdadeiras, considerando o equilíbrio entre as classes.

3.3.3 Random Forest

Figura 27: Random Forest

)

Accuracy	Cohen's Kappa
64,268%	0.59%

Tabela 15: Resultados do Random Forest

Os resultados na Tabela 15 demonstram um desempenho satisfatório do modelo de *Random Forest* na classificação das faixas de preço dos carros. Com uma precisão de cerca de 64,268%, o modelo é capaz de fazer previsões consistentes em relação às faixas de preço. Além disso, o coeficiente de Cohen's kappa mostra uma concordância razoável entre as previsões do modelo e as classes verdadeiras, indicando uma boa capacidade de generalização do modelo.)

3.3.4 Polynomial Regression

Neste modenovamente,qalterarámos ucobjetivocopo do probe se concendontra em determinar ue intervaloaixa de preço o carro se encaix recorrendo a *Polynomial Regression* (Figura 28)a ia.

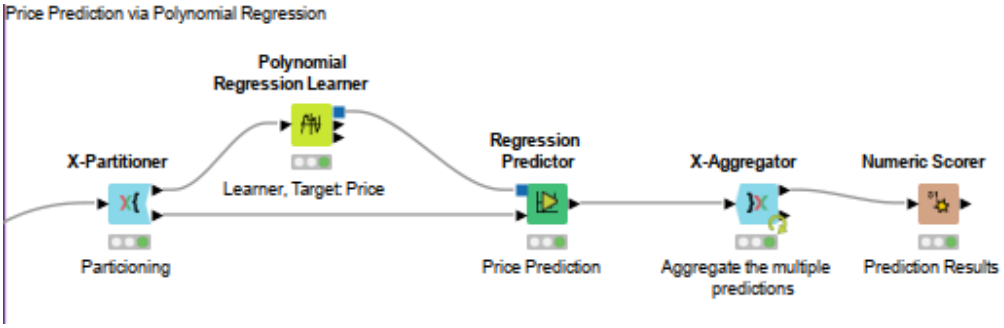

Figura 28: Polynomial Regression

)

R^2	Mean absolute error	Mean absolute percentage error
0,581	6500,985	1.488

Tabela 16: Resultados do Polynomial Regression

)Os ses resultana Tabela 16 dos fornecem uma avaliação do desempenho do modelo de regressão polinomial na previsão de preços de carros. Embora o modelo seja capaz de explicar uma parte significativa da variabilidade nos preços dos carros, os erros de previsão ainda são relativamente altost Isso sugere que o modelo pode não ser o mais adequado para capturar a relação entre as características dos carros e seus preços de forma precisa.

3.3.5 Redes onaisrais

3.3.5.1 RProp MLP

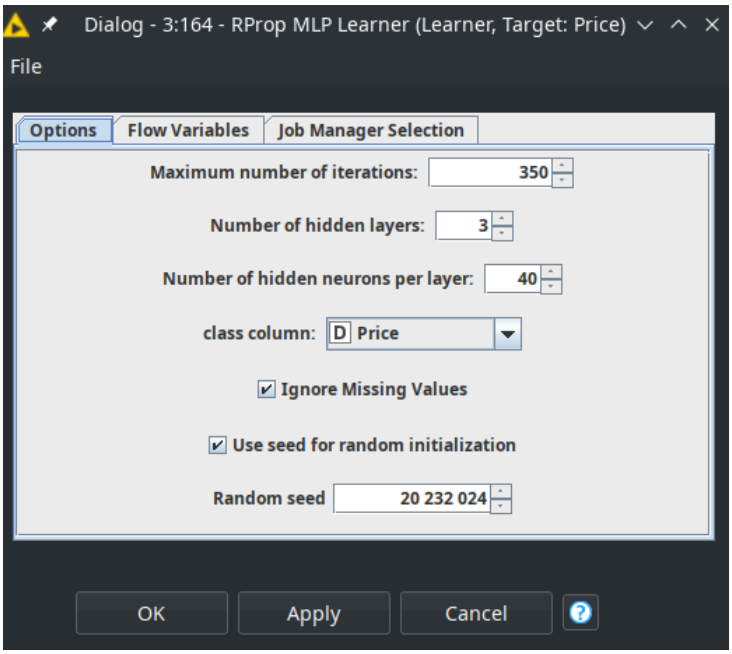

Figura 29: Configurações do RProp MLP

)

Nesta configuração da rede neur (Figura 29)onal, são definidas três camadas ocultas, cada uma composta por 40 neurónios. Além disso, o treino da rede é conduzido por até 350 iterações.

R^2	Mean absolute error	Mean absolute percentage error
0,79	4300,641	0.841

Tabela 17: Resultados do RProp MLP

Os resultados na Tabela 17 demonstram um desempenho bastante promissor do modelo RProp MLP na tarefa de previsão de preços de carros. O coeficiente de determinação próximo a 0,79 indica uma boa capacidade do modelo em explicar a variabilidade nos preços dos carros. Além disso, os erros de previsão são relativamente baixos, o que sugere uma precisão satisfatória nas estimativas de preços de carros. Essa análise indica que o modelo RProp MLP pode ser uma opção viável e eficaz para a previsão de preços de carros com bnase em suas características.

3.4 Conclusão sobre o dataset

Podemos observar a partir dos resultados obtidos através dos diversos algoritmos executados que as reneuronaisrais, em particular o modelo RProp MLP, demonstram ser a melhor solução para este problema de classificação das faixas de preço dos carros. Essa superioridade sugere que os dados presentes no *dataset* possuem padrões não convencionais ou complexos, que são melhor capturados e interpretados por modelos de reneuronaisrais.

A eficácia das reneuronaisrais nesse contexto pode ser atribuída à sua capacidade de aprender e representar relações complexas entre as variáveis de entrada e as classes de saída. Ao contrário de modelos mais simples, as reneuronaisrais são capazes de identificar padrões subtis e não lineares nos dados, adaptando-se de forma flexível a diferentes tipos de distribuições e relações.

Portanto, os resultados sugerem que os padrões presentes nos dados do *dataset* são mais bem compreendidos e explorados por modelos de reneuronaisrais, destacando a importânciatdessa abordagem para resolver problemas com características não convencionais ou altamente complexas. Esta conclu-

reforçaalta a importância de escolher adequadamente o modelo de *machine learning* de acordo com a natureza dos dados e a complexidade do problema em questão.

4 Conclusão

A análise, exploração e aplicação de algoritmos de *machine learning* nos *datasets* permitiram obter *insights* valiosos e resultados promissores. Conseguimos explorar a complexidade e diversidade dos dados em questão ao utilizar várias técnicas desde modelos simples a algoritmos de *ensemble*.

Podemos concluir a partir dos resultados obtidos, que de um modo g,eral os algoritmos de *ensemble* se destacam dos demais em múltiplos casos, pois conseguem oferecer boa precisão e estabilidade na previsão de resultados.

Ressaltamos a importância da preparação dos *datasets*, desde tratamento de valores em falta e *outliers* até à transformação de dados para garantir a qualidade dos modelos.

Perceb,eque em partic,ular que há espaço para aprimorar, por exe,mplo ao incluir a aplicação de técnicas mais avançadas de exploração de dados. Além disso, reconhecemos a necessidade de aprofundar o nosso conhecimento relativamente a algoritmos de regressão além dos que foram estudados neste trabalho.

Também reconhecemos que houve desafios na conclusão das reneuronaisrais do tiLoJDJ4L, bem como a oportunidade de explorar outras técnicas como a aplicação de expressões regulares para identificar e categorizar adequadamente os diferentes modelos de carros presentes nos datasets.

Em resumo, o grreconheceualta a importância da escolha adequada de algoritmos, técnicas de processamento dos *datasets* e avaliação dos modelos de forma a alcançar resultados com significado em problemas de *machine learning*.

5 Bibliografia

1. https://archive.ics.uci.edu/dataset/225/ilpd+indian+liver+patient+dataset [ILPD (Indian Liver Patient Dataset)]
2. https://medium.com/@pouyahallaj/data-augmentation-benefits-and-disadvantages-38d8201aead [Data Augmentation: Benefits and Disadvantages]
3. https://medium.com/turing-talks/turing-talks-24-modelos-de-predi%C3%A7%C3%A3o-ensemble-learning-aa02ce01afda [Modelos de Predição | Ensemble Learning]
4. https://d-nb.info/1101582936/34 [Constructive Training of Probabilistic Neural Networks]
5. https://www.kaggle.com/datasets/deepcontractor/car-price-prediction-challenge [Car Price Prediction Challenge]
6. https://en.wikipedia.org/wiki/Car_classification Car classification