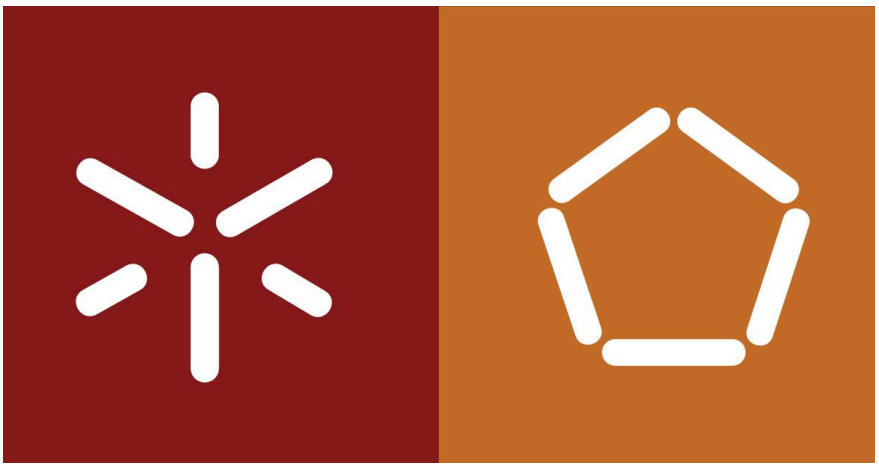


Trabalho Prático - Aprendizagem e Decisão Inteligentes

Grupo 25

Gabriel Ribeiro - a104171

Luís Cunha - a104613

Tomás Barbosa - a104532

Índice

1. Introdução

Este trabalho é realizado no âmbito da UC de Aprendizagem e Decisão Inteligentes e tem como o objetivo trabalhar dados presentes em datasets e construir modelos de aprendizagem utilizando diversas técnicas de machine learning e explorando diversos paradigmas de aprendizagem com recurso à plataforma KNIME.

2. Metodologia para análise de dados

De forma a garantir melhor planeamento e gestão do projeto e, consequentemente, uma melhor qualidade de resultados, o primeiro passo num projeto de Machine Learning é organizar o seu desenvolvimento numa série de etapas e tarefas.
Para isso, devemos escolher uma metodologia que nos permita definir um processo para seguir. Das que nos foram apresentadas, CRISP-DM e SEMMA, a que nos pareceu mais adequada e fácil de perceber e seguir foi a primeira, CRISP-DM, então é a que iremos seguir.
Esta metodologia defende uma organização hierárquica das tarefas a realizar, separando em várias camadas que se vão especializando: Fases, Tarefas Genéricas, Tarefas Especializadas e Processos.
O projeto é dividido nas seguintes fases: Estudo do Negócio, Estudo dos Dados, Preparação dos Dados, Modelação, Avaliação e Desenvolvimento. No contexto deste projeto, as 2 primeiras fases podem ser transformadas em Definição do Problema e Exploração dos Dados. A última fase também não é importante para este projeto e então pode ser ignorada.
Assim, a estrutura das fases para o projeto será então:

1. **Definição do Problema** – identificar o tema e objetivo do problema;
2. **Exploração dos Dados** – familiarização com os dados (estrutura, quantidade, qualidade, ...)
3. **Preparação dos Dados** – tratamento dos dados e possíveis anomalias
4. **Modelação** – aplicação de técnicas para a construção de um modelo
5. **Avaliação** – avaliar o desempenho do modelo

Na etapa de exploração dos dados, é importante ficar a conhecer as características de cada coluna do dataset e o seu significado e também possíveis padrões existentes entre os dados. Desde a identificação do tipo de dados existentes ao reconhecimento de problemas, esta é uma etapa essencial para as próximas que se seguem.

Na preparação de dados, faz-se tudo o que tenha a ver com seleção, limpeza, construção e formatação destes dados para que o modelo tenha uma complexidade reduzida e um desempenho melhorado.

Na modelação é importante selecionar as técnicas corretas e os algoritmos mais indicados para o contexto do problema. Também é preciso ter em atenção a parametrização destas técnicas e a geração de testes.

Por fim, na parte de avaliação é essencial escolher as métricas mais indicadas para refletir o desempenho do modelo e assim ser possível avaliar os resultados e fazer uma análise crítica.

3. Tarefa 1 - Dataset atribuído

3.1. Definição do Problema

O dataset da primeira tarefa, indicado pelos docentes, diz respeito a um contexto hospitalar. O problema retratado é a realização de um teste clínico que pode ter alguns resultados diferentes. O objetivo é, com base em alguma informação sobre os pacientes, prever qual será o resultado desse teste aplicado a cada um.

3.2. Exploração dos Dados

Os dados fornecidos possuem 20 colunas: 19 features e 1 target. O target é, precisamente, o resultado do teste clínico representado numa String que pode assumir 3 valores distintos:

- Normal
- Abnormal
- Inconclusive

No entanto, as proporções de cada valor estão bastante desequilibradas, verificando-se uma maioria de 'Normal' e uma quantidade relativamente menor de 'Inconclusive'.

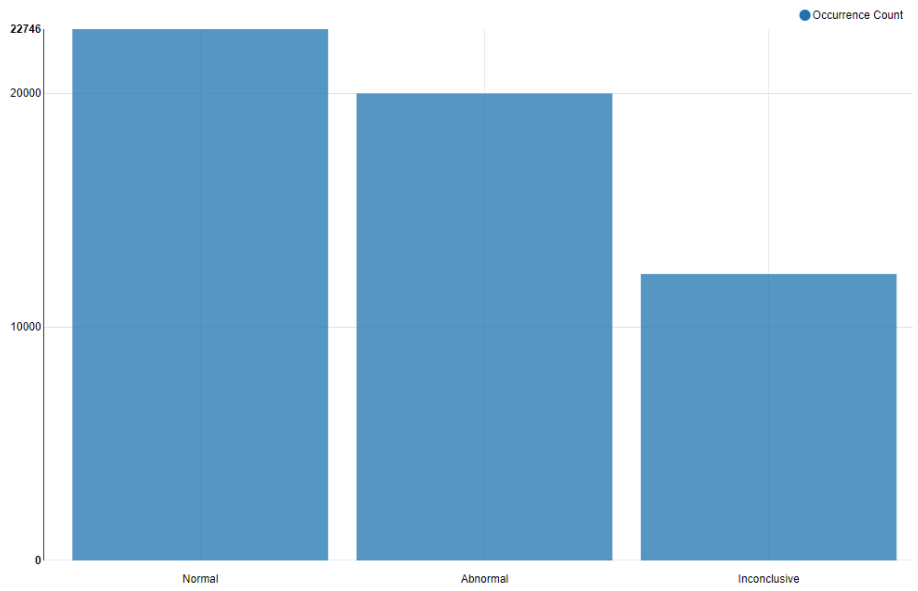


Figura 1 - Quantidades de cada valor do *target*

As features são diversas informações sobre o paciente:

- **Name** (String) - indica o nome do paciente.
- **Age** (Integer) - indica a idade do paciente na altura de admissão.
- **Year** (Integer) - indica o ano de nascimento do paciente.
- **Month** (Integer) - indica o mês de nascimento do paciente.
- **Day** (Integer) - indica o dia de nascimento do paciente.
- **Gender** (String) - indica o género do paciente. Assume os valores 'Male' ou 'Female'.

- **Blood Type** (String) - indica o tipo sanguíneo do paciente. Assume os valores comuns (B-, B+, AB-, AB+, A-, A+, O-, O+).
- **Medical Condition** (String) - indica a principal condição médica associada ao paciente.
- **Date of Admission** (Local Date) - indica a data em que o cliente foi admitido no hospital.
- **Doctor** (String) - indica o nome do médico responsável pelo paciente.
- **Hospital** (String) - indica a identificação do hospital.
- **Insurance Provider** (String) - indica a seguradora do paciente. Assume os valores 'Aetna', 'Blue Cross', 'Cigna', 'UnitedHealthcare' ou 'Medicare'
- **Billing Amount** (String) - indica a quantidade de dinheiro que o paciente pagou.
- **Room Number** (Integer) - indica o número do quarto em que o paciente ficou.
- **Floor** (Integer) - indica o piso em que o quarto se situa.
- **Admission Type** (String) - indica o tipo de admissão. Assume os valores 'Emergency', 'Elective' ou 'Urgent'.
- **Discharge Date** (String) - indica a data em que o paciente saiu do hospital.
- **Medication** (String) - indica a medicação administrada ao paciente durante a sua admissão. Assume os valores 'Aspirin', 'Ipobrufen', 'Penicillin', 'Paracetamol' ou 'Lipitor'.
- **Pill Form** (String) - indica o método de administração do medicamento.
- **Test Results** (String) - indica o resultado do teste feito ao paciente. Assume os valores 'Normal', 'Abnormal' ou 'Inconclusive'.

A quantidade de dados é bastante elevada, totalizando 55000 registos. Quanto à qualidade, pode-se verificar a presença de missing values em várias features, a existência de erros de ortografia em alguns valores nominais e algumas colunas com informação redundante. Para além disto existem algumas features com variância elevada, no caso de números, e com imensos valores nominais.

Também podemos encontrar features como 'Billing Amount' ou 'Discharge Date' que não estão no tipo de dados mais adequado para o que representam.

Column	Exclude Column	Minimum	Maximum	Mean	Standard Deviation	Variance	Skewness	Kurtosis	Overall Sum
Age	☐	13	89	51.496	19.607	384.422	-0.003	-1.185	2569595
Year	☐	1935	2011	1972.504	19.607	384.433	0.003	-1.185	98625211
Month	☐	1	12	6.525	3.455	11.934	-0.003	-1.219	326271
Day	☐	1	31	16.039	8.939	79.900	-0.010	-1.202	801938
Room Number	☐	101	500	301.031	115.281	13289.602	-0.011	-1.196	15051572
Floor	☐	1	5	2.517	1.119	1.253	-0.006	-1.329	125626

Showing 1 to 6 of 6 entries

Figura 2 - Características dos dados numéricos (nodo Data Explorer)

Column	Exclude Column	No. missings	Unique values	All nominal values
Name	☐	0	>1000	AnGEL MiTChelL, naThaN wileY, AsHley WaRnER, JODI martiNeZ, JONathAn yaTeS, [...], CHrISTINA caldeROn, eLIZAbeTh MiLLEr, MIChAeL contreraS, JOSe LopEz, chAd crOss
Gender	☐	0	6	Male, Female, Masculine, Girl, Boy, Feminine
Blood Type	☐	28	8	A+, A-, B+, O+, AB+, AB-, B-, O-
Medical Condition	☐	0	8	Arthritis, Hypertension, Obesity, Diabetes, Cancer, Asthma, Obesitty, Arthritys
Doctor	☐	265	>999	John Smith, James Smith, Jennifer Johnson, Daniel Smith, Matthew Smith, [...],
Hospital	☐	0	>1000	LLC Smith, Ltd Smith, Johnson PLC, Smith Ltd, Smith LLC, [...], Sons Horn and, Mitchell-Navarro, Hicks-Perez, Payne Parker, Murphy and, Hernandez-Pearson
Insurance Provider	☐	162	5	Medicare, Cigna, UnitedHealthcare, Blue Cross, Aetna
Billing Amount	☐	0	>1000	48995.9805916571, 24011.7022492557, 8451.00175847895, 3899.86364076517, 10521.4719302488, [...], 25163.8224838312, 30334.9890660505, 49078.4690110566, 3963.12112831519, 26660.4482046587
Admission Type	☐	0	4	Emergency, Elective, Urgent, Immediate
Discharge Date	☐	0	>1000	15-3-2020, 29-4-2023, 13-12-2021, 2-12-2020, 25-2-2022, [...],

Medication	☐	101	8	Aspirin, Lipitor, Ibuprofen, Paracetamol, Penicillin, Peniciline, Penicilline, Lipidor
Pill form	☐	0	1	Tablet
Test Results	☐	0	3	Normal, Abnormal, Inconclusive

Figura 3 - Características dos dados nominais (nodo Data Explorer)

Dada a presença de datas, também verificamos se existiam datas inválidas. Para isso separamos o dia, mês e ano da 'Date of Admission' e pusemos num Scatter Plot para identificar valores de dias e meses errados. No caso da 'Discharge Date', devido a estar em String, recorremos a uma expressão regex para identificar padrões inválidos e armazenamos o valor numa nova coluna 'validDate' com *true* se é uma data válida e *false* caso contrário. Como se pode ver nas imagens abaixo, não haviam datas inválidas.

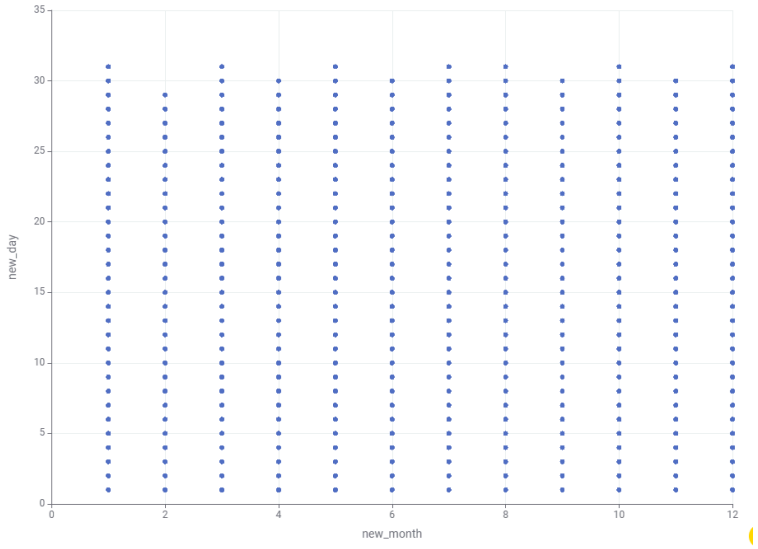


Figura 4 - Scatter Plot do dia e mês da Date of Admission

Figura 5 - Bar Chart da coluna 'validDate', com tudo a 'true'

Para verificar a existência de outliers nos valores numéricos, recorremos a uma Box Plot. Assim, verificamos que não existiam outliers.

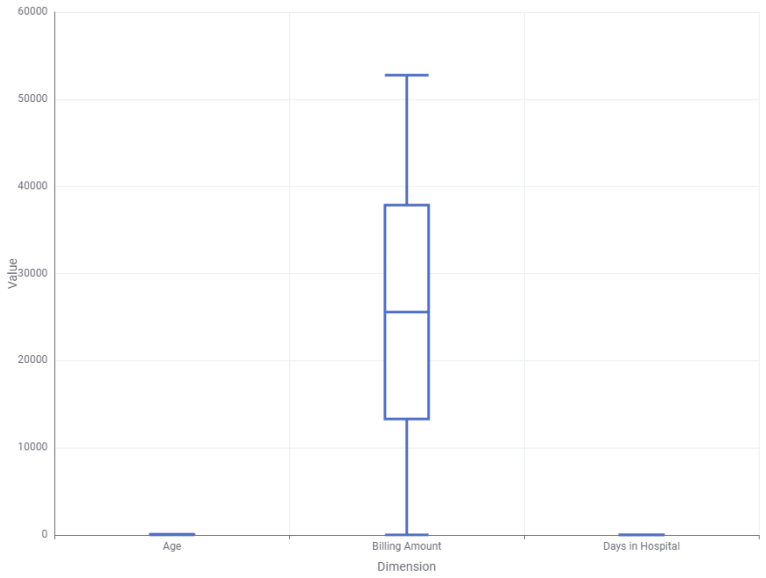


Figura 6 - Box Plot das colunas Age, Billing Amount e Days in Hospital

Todos estes problemas identificados, entre outros, têm de ser tratados antes de passar os dados aos modelos.

3.3. Preparação dos Dados

3.3.1 Preparação geral

Após a exploração dos dados, agora é necessário corrigir os problemas encontrados.

Valores Inválidos

O primeiro passo foi corrigir os erros de escrita em várias colunas com recurso ao nodo Rule Engine:

```
1 $Gender$ = "Male" => "Male"
2 $Gender$ = "Masculine" => "Male"
3 $Gender$ = "Boy" => "Male"
4 $Gender$ = "Female" => "Female"
5 $Gender$ = "Feminine" => "Female"
6 $Gender$ = "Girl" => "Female"
```

Figura 7 - Regras para a coluna Gender

```
$Medical Condition$ = "Arthritis" => "Arthritis"
$Medical Condition$ = "Arthritys" => "Arthritis"
$Medical Condition$ = "Hypertension" => "Hypertension"
$Medical Condition$ = "Obesity" => "Obesity"
$Medical Condition$ = "Obesitty" => "Obesity"
$Medical Condition$ = "Diabetes" => "Diabetes"
$Medical Condition$ = "Cancer" => "Cancer"
$Medical Condition$ = "Asthma" => "Asthma"
```

Figura 8 - Regras para a coluna Medical Condition

```
$Medication$ = "Aspirin" => "Aspirin"
$Medication$ = "Lipitor" => "Lipitor"
$Medication$ = "Lipidor" => "Lipitor"
$Medication$ = "Ibuprofen" => "Ibuprofen"
$Medication$ = "Paracetamol" => "Paracetamol"
$Medication$ = "Penicillin" => "Penicillin"
$Medication$ = "Peniciline" => "Penicillin"
$Medication$ = "Penicilline" => "Penicillin"
```

Figura 9 - Regras para a coluna Medication

Também identificamos um valor na coluna Admission Type 'Immediate' que não constava na descrição dos campos. Como estava em muito pouca representação (cerca de 1072 em 55000 registos) e dado que o seu significado é similar ao do valor 'Urgent', decidimos trocar esses valores para estes último para reduzir a complexidade do modelo e mantivemos pois mostrou trazer algumas melhorias.

```
$Admission Type$ = "Emergency" => "Emergency"
$Admission Type$ = "Elective" => "Elective"
$Admission Type$ = "Urgent" => "Urgent"
$Admission Type$ = "Immediate" => "Urgent"
```

Figura 10 - Regras para a coluna Admission Type

Novo atributo

Para conseguir tirar alguma informação das colunas Date of Admission e Discharge Date, decidimos criar um novo atributo 'Days in Hospital' que consiste na diferença entre estas 2 datas e indica o número de dias que o paciente passou no hospital.

Tratamento Billing Amount (Binning)

Para tratar da coluna Billing Amount foram necessários alguns passos. Primeiro, eliminou-se as entradas inválidas 'NaN'. Depois, passamos o tipo de String para Number (Double) e por fim, filtramos as entradas que tinham valores com significado inválido que eram os negativos.

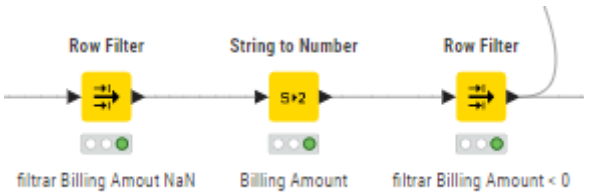


Figura 11 - Tratamento da coluna Billing Amount

Como a coluna Billing Amount agora é do tipo Double e tinha uma alta variância, achamos relevante fazer o Binning dos valores. Optamos por 6 bins de igual largura, pois dividiam os dados de forma bastante uniforme e variar nestes valores não mostrou trazer grandes ganhos mais tarde no desempenho mais tarde. Esta parte dos tratamentos não vai ser feita para o modelo da rede neuronal, como será referido mais à frente.

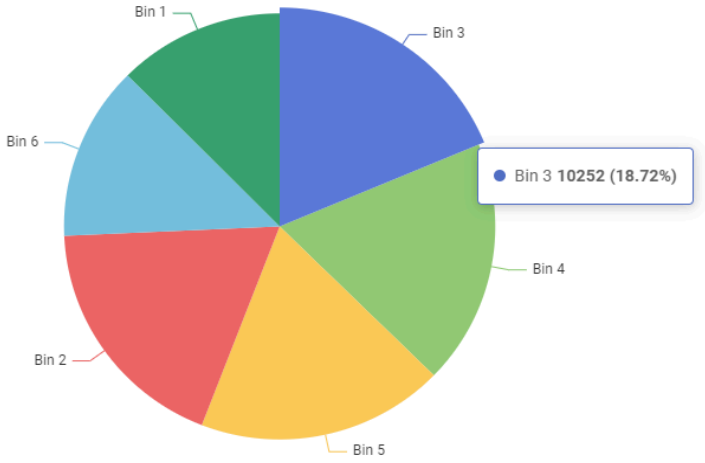


Figura 12 - Contagem de entradas por bin

Filtragem de Colunas

Depois, filtramos as colunas que eram redundantes, ou seja, que não ofereciam nenhuma informação útil que outra coluna já não oferecesse ou que tinham baixa ou alta variância, logo não permitem identificar padrões nos dados, e que por isso não contribuem para o desenvolvimento do modelo, mas sim apenas para a sua complexidade:

- **Year** - redundante com **Age**
- **Month** - redundante com **Age**
- **Day** - redundante com **Age**
- **Date of Admission** - redundante com **Days in Hospital**
- **Discharge Date** - redundante com **Days in Hospital**
- **Name** - Alta Variância
- **Doctor** - Alta Variância
- **Hospital** - Alta Variância
- **Room Number** - Alta Variância
- **Pill Form** - Baixa Variância

- **Floor** - Baixa Variância

Missing Values

Para tratar os missing values, como não são muitos comparados com o tamanho total do dataset, optamos por remover as linhas que continham missing values, exceto quando fossem Strings, onde optamos por preencher com o valor mais frequente, por mostrar um desempenho ligeiramente melhor mais à frente.

3.3.2 Preparação específica (SMOTE / Encoding)

Para além desta preparação de dados mais geral, decidimos fazer mais algumas variações para ver se teriam melhor ou pior desempenho no final.

Primeiro, temos 2 variantes da preparação: 1 em que mantivemos os dados separados como eles vieram, em treino e teste, e outra em que juntamos os dados e depois fazemos uma partição aleatória, com 90% dos dados para treino e 10% para simular a separação original que já foi dada. Também experimentamos outras proporções que não mostraram grandes melhorias.

Outro aspeto já referido é que o dataset é um pouco desequilibrado nas quantidades de cada valor da coluna 'Test Results'. Por isso aplicamos o nodo SMOTE nas classes minoritárias para testar mais tarde se resulta num modelo mais equilibrado. O SMOTE (Synthetic Minority Over-sampling Technique), resumidamente, cria registos sintéticos com base nos restantes dados do dataset (neste caso para as classes em minoria, resultando num dataset mais equilibrado). É importante referir que aplicamos esta técnica apenas nos dados de treino do modelo, pois, caso se aplique aos dados todos, verifica-se um overfitting dos dados de teste, pois estão a ser criados mais com base nos que já existiam. Como é uma operação computacionalmente, apenas selecionamos alguns dos algoritmos que se estavam a mostrar mais promissores.

Para o teste das redes neuronais e da transformação do problema em regressão, foi necessário recorrer a uma preparação de dados especial. Para tornar os dados compatíveis com esta técnica precisamos de transformar as features que eram Strings em Números. Como os dados em questão não apresentam nenhuma ordem entre as suas categorias, optamos por usar a técnica One-Hot Encoding que cria colunas novas para cada categoria na coluna e coloca valor 1 se essa for a categoria que estava no registo, caso contrário, 0. Depois, normalizamos os outros dados que já eram números com o método Z-score (que apresentou ligeiramente melhores resultados, devido à grande amplitude de valores), sempre com o mesmo modelo de normalização tanto para dados de treino como de teste.

No caso das redes neuronais, como o target também está em String, precisamos de usar a técnica One-Hot Encoding. Nos testes que fizemos, vimos que o modelo estava a prever quase só com a categoria 'Normal', então decidimos aplicar o SMOTE aqui também da mesma forma que descrito acima.

Já no caso da regressão numérica, apenas convertemos as categorias do target usando Label Encoding para um valor numérico (1, 2, 3). Aqui mantivemos a coluna do target

original para conseguirmos perceber a associação número-categoria que tinha sido criada, no caso: 1 - Normal, 2 - Abnormal, 3 - Inconclusive.

Para garantir a uniformidade dos testes de desempenho, todas as seeds de aleatoriedade nesta etapa e em todas as outras são fixadas num valor aleatório.

3.4. Modelação

Na parte de modelação, o objetivo foi testar diversos algoritmos para perceber quais teriam melhor desempenho neste problema.

3.4.1 Modelos de Classificação

Para este problema de classificação, selecionamos vários algoritmos:

- Decision Tree
- Random Forest
- Naive Bayes
- Tree Ensemble
- Gradient Boosted Trees

Todos estão são adequados para este tipo de problemas. Quanto aos seus parâmetros, não fizemos nenhuma alteração impactante no seu desempenho, exceto as seeds de aleatoriedade, as quais fixamos num valor aleatório.

Os algoritmos que foram selecionados para a preparação especial com SMOTE (e mais à frente para o Cross Validation) foram os que se mostraram mais promissores desde o início, ou seja que tiveram não só melhores resultados, como mais equilibrados:

- Random Forest
- Tree Ensemble
- Decision Trees

Para a Logistic Regression, também experimentamos usar com os dados transformados em números e normalizados, como para a rede neuronal.

Para além disto, também decidimos explorar as redes neuronais e transformar o contexto para um problema de regressão de forma a ser possível aplicar outras técnicas.

3.4.2 Rede Neuronal

Ao explorar as redes neuronais, como já foi referido, foi necessário uma preparação especial dos dados que consistiu em passar os dados categóricos para números e fazer a sua normalização. No KNIME, para usar uma rede neuronal optamos pela integração do Keras com um ambiente Anaconda.

A rede neuronal foi construída da seguinte maneira:

- 1 camada de input a receber 32 features de input
- 3 camadas ocultas com a função de ativação sigmóide e output de tamanho 32

- 1 camadas de output com a função de ativação sigmóide e output de tamanho 3
- Keras Network Learner com Categorical cross entropy como função de perda, por ter mostrado bons resultados (juntamente com outras)
- Keras Network Executor que dá output de 3 colunas de previsão, cada uma delas diz respeito à coluna de cada classe do target que foi gerada no One-Hot Encoding

Depois desta configuração, recorremos a um Rule Engine para criar uma coluna de previsão com os nomes originais das categorias do target, dependendo da coluna que tiver o maior valor de previsão.

```
5 $Output_1/Relu:0_0$ > $Output_1/Relu:0_1$ AND $Output_1/Relu:0_0$>$Output_1/Relu:0_2$ => "Normal"
6 $Output_1/Relu:0_1$>$Output_1/Relu:0_0$ AND $Output_1/Relu:0_1$ >$Output_1/Relu:0_2$ => "Abnormal"
7 $Output_1/Relu:0_2$>$Output_1/Relu:0_0$ AND $Output_1/Relu:0_2$ > $Output_1/Relu:0_1$ => "Inconclusive"
```

Figura 13 - Regras para a coluna de previsão

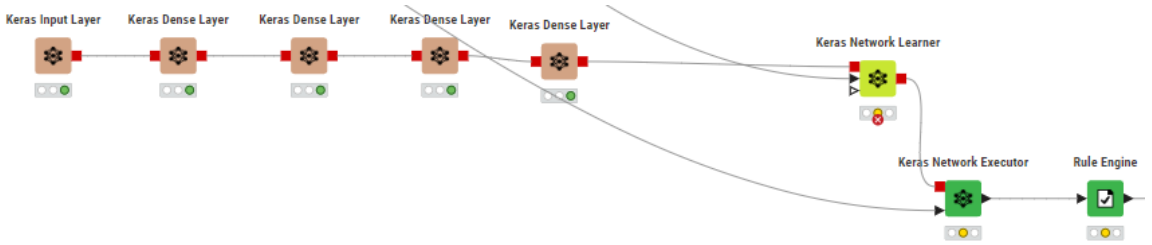


Figura 14 - Configuração da rede neuronal

3.4.3 Modelos de Regressão

Para experimentar algumas técnicas de regressão também selecionamos alguns algoritmos. Desde já selecionamos o Logistic Regression para abordar a regressão logística. Quanto à regressão numérica, selecionamos os seguintes algoritmos:

- Random Forest (Regression)
- Simple Regression Tree
- Linear Regression

Para aplicar estas técnicas, como já foi referido, foi necessário transformar features que eram categorias em numéricas e no caso da regressão numérica, também o target.

Aqui também não houve nenhuma alteração nos parâmetros dos nodos que refletissem alterações significativas no desempenho, exceto as seeds de aleatoriedade e o Solver do nodo Logistic Regression, no qual selecionamos o Stochastic average gradient, que será melhor em tabelas de dados grandes, como é o caso.

3.4.4 Cross Validation

Para além do partitioning normal dos dados (Hold-out Validation), também exploramos a técnica de cross validation. Tentámos fazer com que a quantidade de dados para teste fosse semelhante à divisão que já havia nos datasets, então consideramos *k*, o número de folds, 5 e 10 e a seleção dos dados aleatória. Não fomos para um valor de *k* mais alto pois,

apesar de em alguns testes apresentarem valores mais elevados nas métricas, poderia significar que o modelo apenas se estava a acostumar demasiado aos dados de teste.

Ainda chegamos a considerar o Leave-one-out cross validation, mas devido ao elevado número de dados, isso mostrou-se impossível computacionalmente, como suspeitávamos.

Não aplicamos esta técnica a todos os algoritmos, tendo selecionado apenas aqueles que mostraram melhor desempenho em testes anteriores, já que os restantes não mostraram grandes melhorias

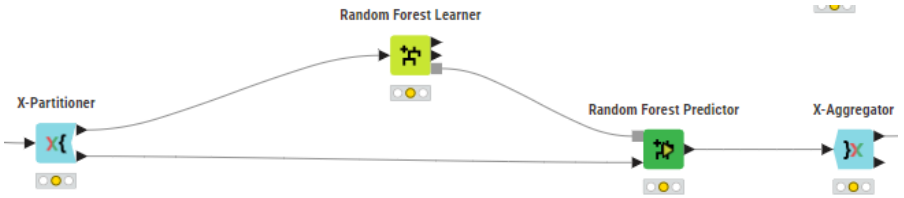


Figura 15 - Exemplo de Cross Validation no KNIME com X-Partitioner e X-Aggregator

3.5. Avaliação

Para avaliar o desempenho dos modelos construídos recorremos aos nodos Scorer e Numeric Scorer para avaliar as seguintes métricas:

- Accuracy global, principalmente
- Recall, equivalente à accuracy por categoria que permite ver o equilíbrio das previsões
- Precision
- R^2 (coeficiente de determinação), quando aplicável na regressão
- MAE (erro médio absoluto), quando aplicável na regressão
- MSE (erro médio quadrado), quando aplicável na regressão

Também se considerou consultar as curvas ROC, mas não se mostraram muito úteis neste caso de estudo, seria melhor num problema de classificação binário, então não as incluímos aqui embora reconheçamos a sua utilidade e importância na análise de desempenho de modelos. Quanto ao RMSE (raiz quadrada do erro médio quadrado), os valores a prever não são muito grandes, então não será muito importante considerar na análise.

Nas tabelas abaixo (A) corresponde à categoria do target 'Abnormal', (I) 'Inconclusive' e (N) 'Normal'.

3.5.1 Avaliação dos Modelos de Classificação

Primeiro a avaliação dos modelos de classificação com a separação do dataset feita pelo partitioning:

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (A)		32,48	44,87

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (I)	44,3	5,09	26,05
Random Forest (N)		76,32	45,23
Tree Ensemble (A)	44,39	31,93	44,76
Tree Ensemble (I)		5,83	29,34
Tree Ensemble (N)		76,63	45,21
Gradient Boosted (A)	39,94	16,09	34,87
Gradient Boosted (I)		0	0
Gradient Boosted (N)		83,2	40,99
Naive Bayes (A)	40,96	3,42	39,88
Naive Bayes (I)		0	0
Naive Bayes (N)		97,14	41
Decision Tree (A)	41,12	40,72	41,49
Decision Tree (I)		17,09	23,56
Decision Tree (N)		54,27	46,67

Tabela 1 - Tabela dos modelos de classificação para o dataset atribuído dividido com partitioning

Podemos ver que os que são mais promissores são Random Forest, Tree Ensemble e Decision Tree, não só porque têm as maiores Accuracy, mas também porque tem as métricas individuais mais equilibradas, ao contrário de outros que tentam adivinhar quase só 1 ou 2 categorias, que são as que têm maior representação no dataset.

Agora, o dataset já dividido originalmente:

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (A)	44,37	32,47	44,77
Random Forest (I)		4,85	22,41
Random Forest (N)		76,17	45,76
Tree Ensemble (A)	43,95	31,86	44,55
Tree Ensemble (I)		3,77	19,44
Tree Ensemble (N)		76,26	45,25
Gradient Boosted (A)	41,58	17,20	38,73
Gradient Boosted (I)		0	0

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Gradient Boosted (N)		85,44	42,16
Naive Bayes (A)	41,24	2,77	37,59
Naive Bayes (I)		0	0
Naive Bayes (N)		97,28	41,34
Decision Tree (A)	38,72	38,5	38,57
Decision Tree (I)		14,14	20,85
Decision Tree (N)		52,07	44,36

Tabela 2 - Tabela dos modelos de classificação para o dataset atribuído dividido originalmente

Como esperado, já que esta divisão também estava com a diversidade de valores bem equilibrada, não se nota grandes oscilações nestes valores, apenas algumas ligeiras subidas e descidas.

Passando ao exercício com o SMOTE, onde só usamos os algoritmos mais promissores, obtivemos os seguintes resultados no partitioning manual:

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (A)	42,2	31,49	42,68
Random Forest (N)		11,17	22,37
Random Forest (I)		68,77	45,56
Tree Ensemble (A)	42,51	31,24	42,09
Tree Ensemble (N)		12,15	24,92
Tree Ensemble (I)		69,21	45,79
Decision Tree (A)	38,8	33,94	41,23
Decision Tree (I)		27,91	23,46
Decision Tree (N)		49,08	46,32

Tabela 3 - Tabela dos modelos de classificação para o dataset atribuído dividido com partitioning (SMOTE)

Aqui já verificamos uma maior diferença nos valores. Apesar de existir uma descida na Overall Accuracy, temos percentagens de acerto um pouco mais equilibradas nas várias classes, o que se pode considerar um ponto positivo pois o modelo é mais equilibrado e consegue responder melhor a situações mais genéricas, que podem ter quaisquer categorias do target.

E na separação dos dados original:

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (A)	43,67	34,13	44,01
Random Forest (N)		12,67	24,91
Random Forest (I)		68,79	47,03
Tree Ensemble (A)	43,85	34,79	43,92
Tree Ensemble (N)		13,03	24,7
Tree Ensemble (I)		68,45	47,6
Decision Tree (A)	37,2	34	39,05
Decision Tree (I)		26,17	22,53
Decision Tree (N)		45,95	44,72

Tabela 4 - Tabela dos modelos de classificação para o dataset atribuído dividido originalmente (SMOTE)

Novamente, não há grande variação, mas foi possível obter melhores Accuracy em 2 dos modelos.

Quanto à aplicação do Cross Validation usamos 2 valores para a quantidade de folds, 5 e 10. O desempenho com $k = 5$ foi:

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (A)	43,73	31,88	43,04
Random Forest (N)		4,58	23,03
Random Forest (I)		75,29	45,34
Tree Ensemble (A)	43,74	32,02	43,15
Tree Ensemble (N)		4,45	22,54
Tree Ensemble (I)		75,25	45,33
Decision Tree (A)	39,53	38,65	39,24
Decision Tree (I)		16,35	23,17
Decision Tree (N)		52,8	45,06

Tabela 5 - Tabela dos modelos de classificação para o dataset atribuído dividido com partitioning: Cross Validation (k=5)

Quando aumentamos o valor de *k*, verificamos:

%	Accuracy (Overall)	Accuracy (Recall)	Precision
Random Forest (A)	44,49	33,73	44,57
Random Forest (I)		4,56	23,17
Random Forest (N)		75,74	45,97
Tree Ensemble (A)	44,5	33,34	44,17
Tree Ensemble (I)		4,71	23,25
Tree Ensemble (N)		75,8	46,05
Decision Tree (A)	40,22	39,72	40,01
Decision Tree (I)		16,35	23,59
Decision Tree (N)		53,49	45,66

Tabela 6 - Tabela dos modelos de classificação para o dataset atribuído dividido com partitioning: Cross Validation (k=10)

Já podemos ver uma melhoria em relação a um número inferior de folds, sendo também uma melhoria em relação ao partitioning normal, que é precisamente o objetivo do Cross Validation: aproveitar ainda mais os dados para teste e treino e obter melhores resultados.

3.5.2 Avaliação da Rede Neuronal

Para avaliar o desempenho da rede neuronal, utilizamos as mesmas métricas acima. Aqui, o Scorer compara a coluna target original com a nova da previsão que foi gerada pelo Rule Engine. Assim, obtivemos os seguintes resultados:

	Accuracy (Overall)	Accuracy (Recall)	Precision
Abnormal	32,32	74,67	36,08
Inconclusive		23,36	20,98
Normal		0	0

Tabela 7 - Tabela dos Rede Neuronal para o dataset atribuído dividido originalmente

Seria esperado de uma rede neuronal, uma ferramenta muito versátil no mundo de Machine Learning, que tivesse um bom desempenho. Ora, este ficou aquém das expectativas, muito provavelmente devido à preparação especial dos dados que foi necessário fazer, já que os dados não eram os mais apropriados para fornecer a uma rede neuronal. Realizamos mais alguns testes, com várias combinações de algumas configurações da rede neuronal, mas não houve nenhuma que saísse deste padrão e se destacasse.

3.5.3 Avaliação dos Modelos de Regressão

Aqui, tentamos avaliar com base em todas as métricas, tanto as percentagens já usadas acima, como as métricas numéricas.

Para a regressão logística, foi feito uma avaliação semelhante às anteriores, apenas com o Scorer:

	Accuracy (Overall)	Accuracy (Recall)	Precision
Abnormal	41,02	2,28	38,98
Inconclusive		0	0
Normal		98,30	41,06

Tabela 8 - Tabela dos modelos de regressão dataset atribuído dividido originalmente (Regressão Logística)

Verifica-se uma situação semelhante a alguns modelos de classificação que já vimos, onde quase só tentou adivinhar uma categoria que é a que tem mais entradas no dataset.

Para os restantes algoritmos, também tentamos explorar o R^2, MAE e MSE. De forma a conseguirmos medir a Accuracy e restantes métricas, necessitamos de arredondar a previsão feita para um dos valores válidos das classes do target (1, 2 e 3). Para isso recorremos ao Rule Engine e depois convertemos esse número arredondado em String para depois ser comparado pelo Scorer.

```
5 $Prediction (Test Results)$ <= 1.5 => 1
6 $Prediction (Test Results)$ > 1.5 AND $Prediction (Test Results)$ < 2.5 => 2
7 $Prediction (Test Results)$ >= 2.5 => 3
```

Figura 16 - Rule Engine para arredondar os valores para inteiros

	Accuracy (Overall)	Accuracy (Recall)	Precision	R^2	MAE	MSE
Random Forest Regression (A/2)	37,86	99,06	37,24	0,019	0,65	0,585
Random Forest Regression (I/3)		0,08	100			
Random Forest Regression (N/1)		3,17	69,61			
Simple Regression Tree (A/2)	38,42	38,66	40,11	-0,918	0,792	1,144
Simple Regression Tree (I/3)		22,66	22,17			
Simple Regression Tree (N/1)		46,78	45,84			
Linear Regression (A/2)		100	36,89			

	Accuracy (Overall)	Accuracy (Recall)	Precision	R^2	MAE	MSE
Linear Regression (I/3)	36,89	0	0	-0,019	0,684	0,608
Linear Regression (N/1)		0	0			

Tabela 9 - Tabela dos modelos de regressão dataset atribuído dividido originalmente (Resto dos Modelos)

Podemos ver que estes modelos não são os mais indicados para este caso de estudo. O Linear Regression apresenta métricas muito más, onde podemos ver que apenas tentou acertar valores que representam uma única classe. As suas métricas de erro também são elevadas para a escala do problema.

O Random Forest (Regression) também tem uma distribuição muito desequilibrada e com erros altos. O Simple Regression Tree tem os erros mais elevados, mas ao mesmo tempo, tem a maior proporção de acertos e bastante equilibrados.

Estes erros elevados e os maus valores de R^2 poderão ser explicados pelo facto de os modelos tentarem prever números decimais entre 1 e 3, mantendo uma certa ordem de hierarquia entre os valores e depois estes são comparados com os valores discretos 1, 2 ou 3, logo como o modelo não tenta prever exclusivamente os valores discretos, o erro vai ser bastante elevado.

3.6. Análise Final

Nesta tarefa identificamos a maior parte dos problemas do dataset e arranjamos ferramentas e técnicas para os corrigir. Também construímos vários modelos de classificação e regressão para perceber o que funcionava melhor nesta situação. Dada a qualidade ou natureza dos dados e alguma limitação nossa, as percentagens de acerto mais elevadas que conseguimos foram na casa dos ~44% com uma distribuição equilibrada pelas classes.

Consideramos que o melhor modelo a que chegamos é o Tree Ensemble com o SMOTE aplicado na divisão original dos dados, pois é equilibrado e tem uma percentagem relativamente alta de acerto.

4. Tarefa 2 - Dataset Escolhido

4.1. Definição do Problema

O dataset escolhido para a segunda tarefa foi o Melbourne Housing Market.csv e contém informação relativa à habitação na cidade de Melbourne. Escolhemos este dataset pois tem um número adequado de registos, um bom equilíbrio entre colunas numéricas e nominais e também alguns erros e inconsistências nos dados para tratar. Outra razão da escolha foi porque achamos interessante a proposta do problema: prever o preço de uma habitação com base nas suas características, sendo assim um problema de regressão.

4.2. Exploração dos Dados

O dataset possui 21 colunas. Alguma informação pode ser encontrada aqui:

- **Suburb** (String) - indica o subúrbio onde a habitação se localiza
- **Address** (String) - indica a morada da habitação
- **Rooms** (Integer) - indica o número de divisões da habitação
- **Type** (String) - indica o tipo da habitação. Assume os valores 'h' (House), 'u' (Unit) ou 't' (Townhouse)
- **Price** (Double) - indica o preço da habitação em milhões
- **Method** (String) - indica o método da venda da habitação. Assume os valores:
    - S - property sold;
    - SP - property sold prior;
    - PI - property passed in;
    - VB - vendor bid;
    - SA - sold after auction;
- **SellerG** (String) - indica a identificação do agente da venda
- **Date** (String) - indica a data da venda
- **Distance** (Double) - indica a distância ao centro de Melbourne
- **Postcode** (Double) - indica o código postal
- **Bedroom2** (Double) - indica o número de quartos
- **Bathroom** (Double) - indica o número de casas de banho
- **Car** (Double) - indica o número de lugares para estacionamento de carros
- **Landsize** (Double) - indica o tamanho do terreno
- **BuildingArea** (Double) - indica a área de construção
- **YearBuilt** (Double) - indica o ano de construção
- **CouncilArea** (String) - indica o governo federal da área
- **Latitude** (Double) - indica a latitude da localização
- **Longitude** (Double) - indica a longitude da localização
- **Regionname** (String) - indica o nome da região
- **Propertycount** (Double) - indica o número de propriedades que existem no subúrbio

A quantidade de dados é adequada (13580) e podemos identificar logo alguns valores em falta em certas colunas. Dada a natureza dos dados, também há colunas com grande amplitude de valores e com alta ou baixa variância. Nos valores temos especialmente o Address com um grande universo de valores. Também podemos identificar valores inválidos nas colunas Landsize e BuildingArea que logicamente têm de ser positivos. Os valores da Date são todos válidos, no entanto, há registos com um ano de construção mais recente que o ano de venda.

Para perceber algumas relações entre os dados e tentar identificar padrões, verificamos a evolução do preço médio das habitações segundo alguns parâmetros, montamos 3 gráficos de barras recorrendo aos nodos GroupBy, para agrupar os parâmetros com a média do preço e Bar Chart.

Figura 17 - Agrupamento dos dados para construção do gráfico

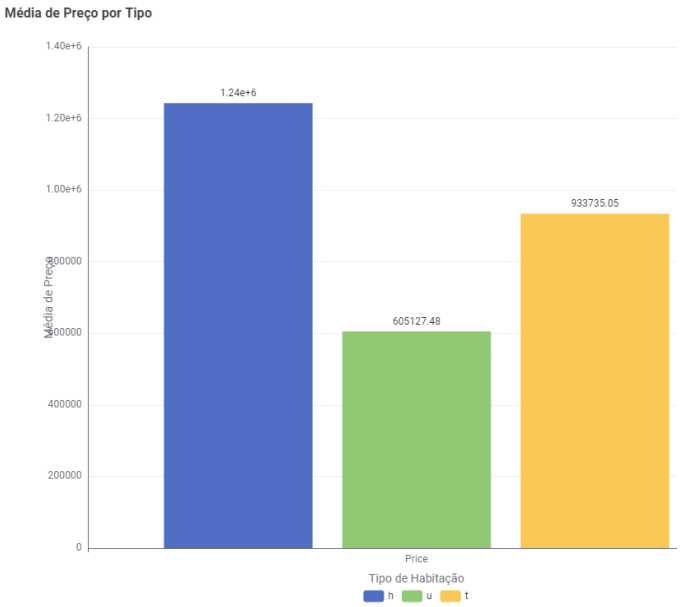
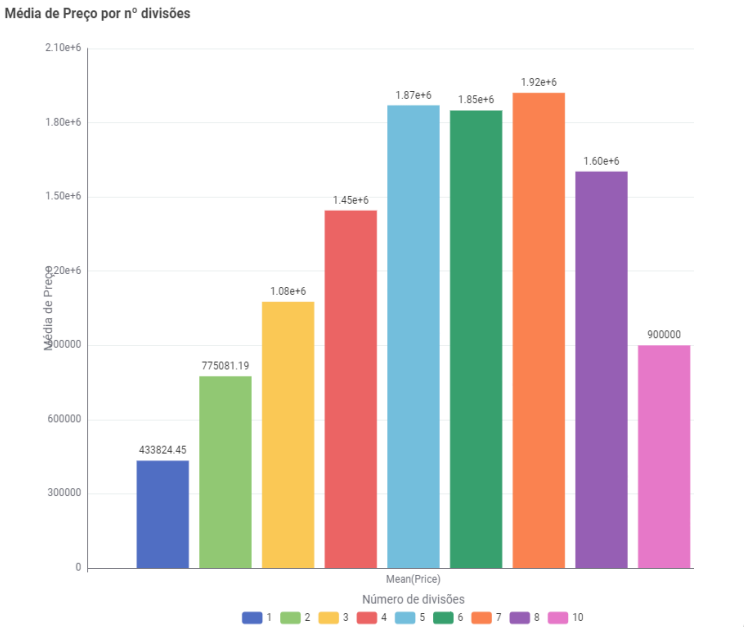


Figura 18 - Preço médio das habitações por tipo de habitação



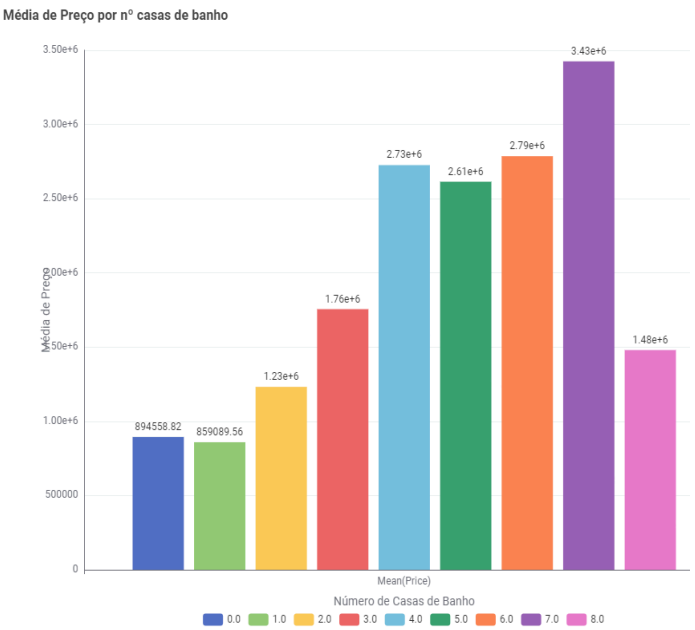


Figura 19 - Preço médio das habitações por nº de divisões **Figura 20** - Preço médio das habitações por nº de casas de banho

Por análise dos gráficos vemos que, no geral, existe uma tendência crescente do preço das habitações com o aumento do número de divisões e de casas de banho. Também vemos que tipo de habitação mais caro é o House, seguido do Townhouse e do Unit.

Com recurso a Box Plot, conseguimos identificar a existência de outliers em algumas colunas:

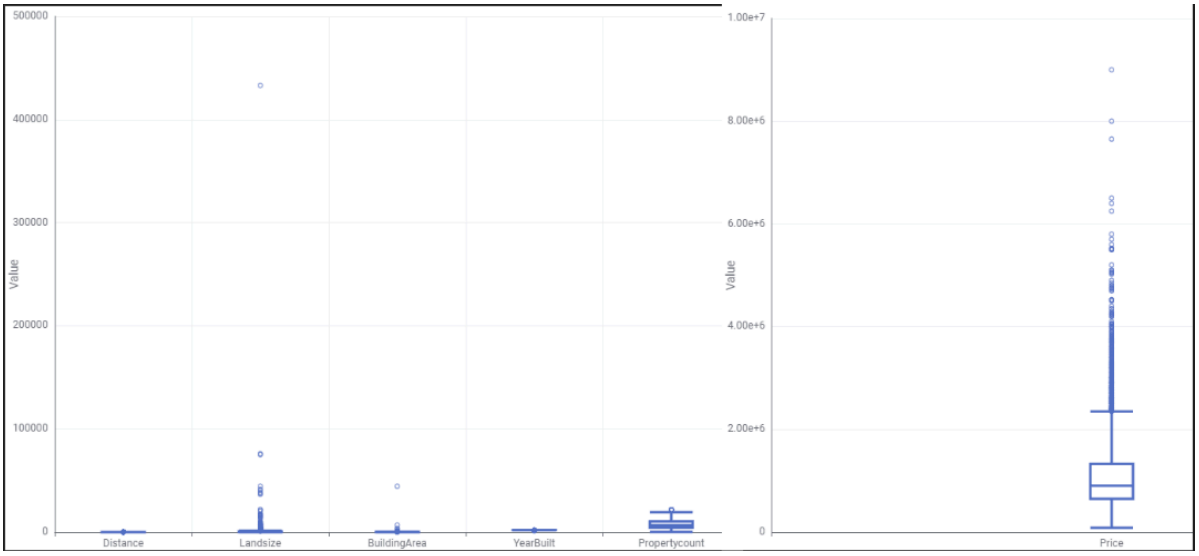


Figura 21 - Box Plot de algumas colunas relevantes para identificar outliers

Com recurso a uma matriz de correlação, também pudemos identificar alguns aspetos interessantes:

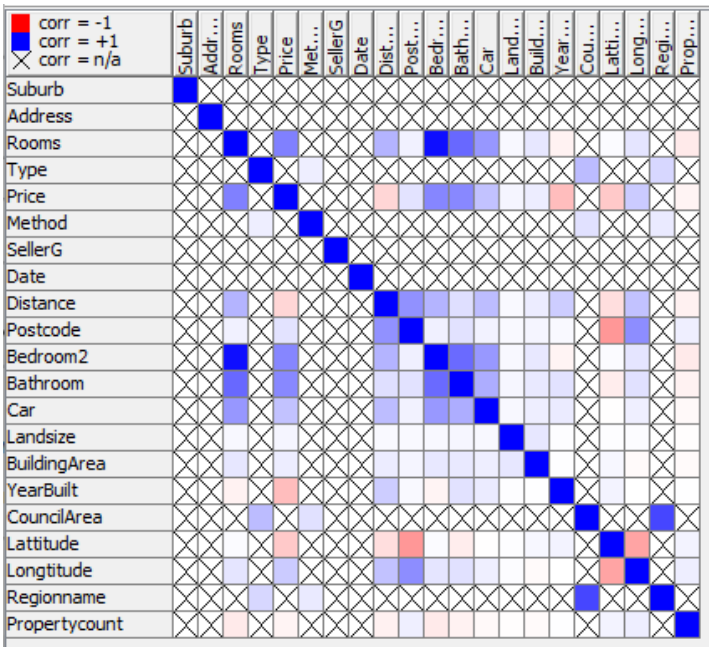


Figura 22 - Matriz de correlação

Vemos que o atributo Rooms tem uma alta correlação com o Bedroom2 e também alguma com o Bathroom, como seria esperado e o mesmo acontece entre o Regionname e o CouncilArea. Também podemos ver que o Rooms tem alta correlação com o Price (e os outros que têm alta correlação com Room), logo será importante para a previsão do preço. Por outro lado o YearBuild e a Lattitude têm alguma correlação negativa com o Price, pelo que também serão importantes.

4.3. Preparação dos Dados

4.3.1 Preparação Geral

Missing Values

O primeiro passo foi tratar dos missing values. Desta vez, optamos por não retirar as linhas, mas preencher com outros valores. Não existem Integers com missing values, no caso das Strings preenchemos com o valor mais frequente e Doubles colocamos a mediana dos valores.

Valores inválidos

Como identificamos valores inválidos no Landsize e BuildingArea (iguais a 0) e nas datas (ano de construção mais recente do que o ano de venda), o passo seguinte foi filtrar estes registos inválidos.

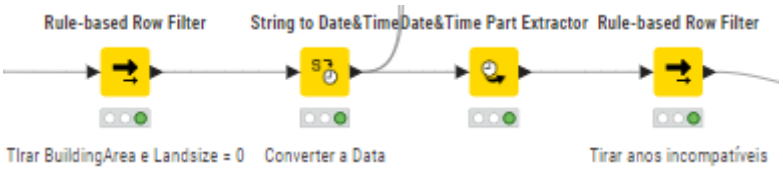


Figura 23 - Filtros de registos inválidos

Filtragem de Colunas

De seguida filtramos as colunas que consideramos desnecessárias. Primeiro foi a coluna DateYear que foi gerada na etapa de preparação passada. Depois, filtramos o Address, pois tem demasiados valores, e o SellerG pois também tem bastantes valores diferentes e achámos mais irrelevante para o problema. Por último, tiramos a coluna Bedroom2 por apresentar elevada correlação com a Room, reduzindo assim a complexidade do modelo.

Tratamento de Outliers

Aqui decidimos tratar apenas os outliers mais graves do Price, pois eram os mais expressivos, já que os outros apesar de serem considerados outliers na análise anterior, estavam muito próximos entre si, indicando que são grupos de dados com características especiais e mantivemos.

Binning da Distância

Achamos apropriado fazer binning da coluna Distance, pois é um atributo que contém dados contínuos. Os valores também mostraram ter um pouco de desvio (skeewness), então decidimos experimentar o binning por frequência aqui.

Encoding

Aplicamos novamente One-Hot Encoding nas colunas Type, Method e Regionname para transformar estes valores categóricos em numéricos, já que eles não tem nenhuma ordem

entre os seus valores e os modelos usados mais à frente são mais recetivos a valores numéricos.

Normalização dos dados

Normalizamos alguns dados que continham escalas distintas nas colunas Rooms, Postcode, Landsize, BuildingArea, YearBuilt e Propertycount. O método de normalização escolhido foi o Z-Score para lidar melhor com os outliers e valores mais afastados que foram mantidos anteriormente.

Também pensamos em fazer a previsão não dos valores de preço normal, mas sim do seu logaritmo para verificar se haveriam melhores previsões, mas percebemos que era também era indicado manter os valores normais.

4.3.2 Preparação Específica

Para tentar ter melhor desempenho a aplicar um paradigma de aprendizagem não supervisionada, excluímos algumas etapas da preparação de dados, nomeadamente o binning da distância e o tratamento de alguns outliers, pois com esse preparamento a identificação de padrões estava mais difícil e a aplicação da Segmentação estava a mostrar ligeiramente piores resultados.

Na Segmentação, também foi necessário agrupar o preço em classes. Definimos 3 categorias, com base num Rule Engine:

- **B** (Barato) - Preço <= 1000000
- **C** (Caro) - 1000000 < Preço <= 3000000
- **MC** (Muito Caro) - Preço >= 3000000

```
5 $Price$ <= 1000000 => "B"
6 $Price$ > 1000000 AND $Price$ <= 3000000 => "C"
7 $Price$ > 3000000 => "MC"
```

Figura 24 - Rule engine para organizar os preços

Assim, a categoria de Barato ficou com a maioria das habitações, seguido do Caro e um grupo mais raro que é o mais caro.

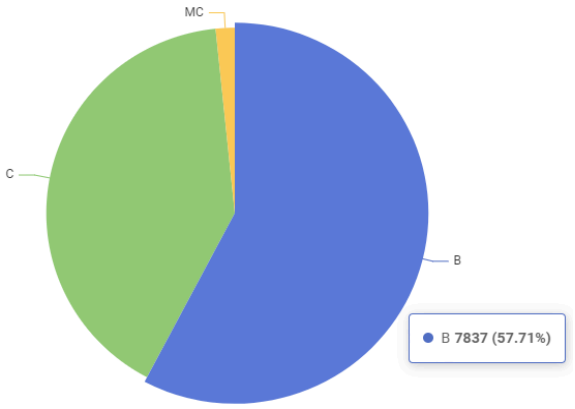


Figura 24 - Distribuição dos dados pelas categorias

Para os restantes modelos, como é necessário dividir os nossos dados em teste e treino, fizemos uma partição aleatória em que 80% dos registos são para treino e os restantes 20% para teste.

Para a rede neuronal construída, como descrita abaixo, foi necessário uma normalização de todos os dados, desta vez com o método Min-max, para manter o Price (target) entre 0 e 1 e depois uma filtragem de todas as colunas que não fossem números, pois não podiam seguir para o learner.

4.4. Modelação

Nesta tarefa o objetivo foi a aplicação de técnicas de regressão e explorar o paradigma de aprendizagem não supervisionada, que funcionaram pior ou não foram explorados na tarefa passada.

4.4.1 Modelos de Regressão

Para os modelos de regressão, selecionamos os seguintes algoritmos:

- Linear Regression
- Polynomial Regression
- Random Forest (Regression)
- Simple Regression Tree
- Gradient Boosted Trees (Regression)

Muitos destes são os que usamos na tarefa anterior que queremos ver a diferença quando aplicados a um problema adequado.

Quanto à parametrização, apenas é relevante mencionar que variamos o grau máximo do polinómio para ver qual se saía melhor.

4.4.2 Segmentação

Para abordar o paradigma de aprendizagem não supervisionada iremos explorar a segmentação. Assim, iremos usar os seguintes algoritmos:

- K-Means
- K-Medoids

O objetivo aqui será agrupar nos registos em 3 classes de preço que foram mencionadas antes (B, C, MC).

Sendo assim, nos parâmetros nodos do k-Means e do k-Medoids, configuramos o número de clusters para 3. No k-Medoids, o cálculo das distâncias que optamos foi a distância Euclidiana.

Para além disso tentamos realizar segmentação hierarquizada mas reparamos que era um processo lento que não lida bem com a escalabilidade, ou seja, para dados grandes este tipo de segmentação não é eficiente, nem a níveis de memória, como a níveis temporais.

Depois da Segmentação estar concluída, é necessário identificar quais clusters correspondem a que categoria e depois juntar novamente a categoria de preço verdadeira para ser possível avaliar o resultado.

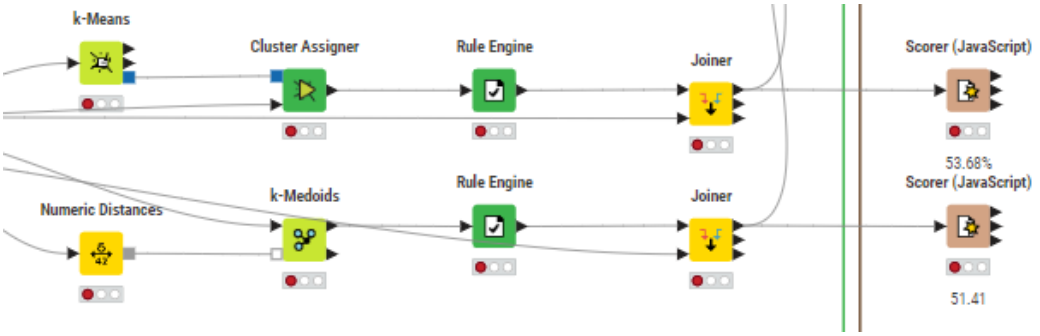


Figura 26 - Processamento dos Clusters

4.4.3 Rede Neuronal

Decidimos testar novamente a rede neuronal, mas desta vez usamos o nodo já nativo do KNIME RProp MLP Learner. Após alguma preparação de dados especial, aplicamos os nodos:

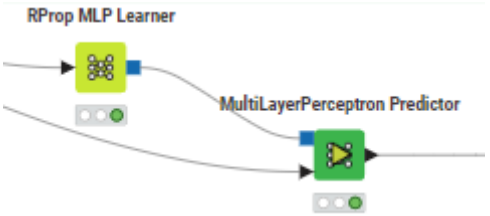


Figura 27 - Nodos da Rede Neuronal

Para a configuração dos parâmetros da rede, escolhemos alguns valores um pouco aleatórios, mas que nos pareceram adequados:

- Número máximo de iterações: 100
- Número de camadas ocultas: 4
- Número de neurónios por camada: 10
- Random seed fixa

4.5. Avaliação

4.5.1 Avaliação dos Modelos de Regressão

Para avaliação dos modelos de regressão atentamos ao R^2, MAE e RMSE, este último agora mais útil pela dimensão dos valores a prever. Dada a essa mesma grande dimensão, o

MSE apresenta valores muito elevados, pelo que por questões práticas não incluímos na análise.

	R^2	MAE	RMSE
Linear Regression	0,63	258070,83	344146,506
Polynomial Regression (grau 2)	0,646	251004,537	336938,907
Polynomial Regression (grau 5)	0,656	247889,754	331801,263
Polynomial Regression (grau 7)	0,664	245812,636	328293,621
Random Forest (Regression)	0,827	155460,323	235301,68
Simple Regression Tree	0,678	203911,731	320973,775
Gradient Boosted Trees (Regression)	0,831	155475,775	232490,874

Tabela 10 - Tabela dos modelos de regressão para o segundo dataset

Podemos ver que os modelos com melhor desempenho são claramente o Random Forest e Gradient Boosted Trees, com valores de R^2 muito bons e erros mais baixos que os outros. Interessante ver que a regressão polinomial consegue-se adaptar melhor ao problema do que a regressão linear, especialmente com graus maiores.

Também decidimos testar o desempenho com Cross Validation com *k*=10 folds em 3 dos melhores modelos para ver como se saíam:

	R^2	MAE	RMSE
Linear Regression	0,555	251134,903	372438,943
Random Forest (Regression)	0,823	155843,425	235034,934
Gradient Boosted Trees (Regression)	0,822	157140,56	235586,097

Tabela 11 - Tabela dos modelos de regressão para o segundo dataset (Cross Validation)

Continuaram a produzir bons resultados, mas piores do que os anteriores.

4.5.2 Análise da Segmentação

Neste caso, as métricas de avaliação já voltam a ser outras, a Accuracy, o Recall e a Precision.

%	Accuracy (Overall)	Accuracy (Recall)	Precision
K-Means (B)	53,68	52,64	59,12
K-Means (C)		57,07	55,89
K-Means (MC)		0	0
K-Medoids (B)	51,41	49,13	58,75
K-Medoids (C)		56,09	56,09
K-Medoids (MC)		1,38	0,24

Tabela 12 - Tabela dos valores com Segmentação aplicada no segundo dataset

Podemos dizer que tiveram uma prestação bastante sólida com boas percentagens de acerto. Os 2 algoritmos foram muito similares, com o K-Means a ter melhor prestação globalmente, mas o K-Medoids a prever um pouco mais da categoria MC. Nesta categoria, ambos tiveram muita dificuldade a prever, pois é a pior representada nos dados, logo seria expectável este resultado.

Aqui, normalmente é útil visualizar as previsões num Scatter Plot com cores atribuídas às várias classes, mas devido à grande quantidade de dados, ou diminuindo a quantidade que são apresentados nos gráficos, não é possível retirar boas informações da sua visualização.

4.5.3 Análise da Rede Neuronal

Desta vez, a rede neuronal foi melhor sucedida do que na tarefa passada, embora não se possa comparar diretamente, pois foi construída de forma diferente.

R^2	MAE	MSE	RMSE
0,734	0,062	0,008	0,088

Tabela 13 - Tabela dos valores com Rede Neuronal aplicada no segundo dataset

Aqui, não podemos comparar os erros MAE, MSE e RMSE com os outros, pois aqui os dados estão todos normalizados entre 0 e 1.

Podemos considerar na mesma um resultado bom lendo os erros na escala em que estamos e olhando em especial para o valor de R^2 que está muito bom e percebemos que o modelo teve uma ótima prestação

4.6. Análise Final

A segunda tarefa teve resultados muito satisfatórios, já que conseguimos aplicar técnicas que na primeira tarefa não tinha sido possível ou que correram mal.

Em primeiro lugar, os modelos de regressão funcionaram muito melhor agora que aplicados a um problema cujo target é um valor contínuo. Consideramos o melhor modelo a ser entre o Random Forest (Regression) ou o Gradient Boosted Trees (Regression) com R^2 na casa dos 0,82, um ótimo valor para 2 técnicas bastante poderosas.

A segmentação também teve um desempenho aceitável que podia ser muito melhor não fosse a dificuldade dos modelos a trabalhar com a categoria em grande minoria.

A rede neuronal segue o mesmo caminho que os modelos de regressão tendo uma performance melhor que a da tarefa passada, sendo bastante simples de implementar e obtendo resultados bastante bons.

Na exploração de dados também conseguimos explorar conceitos novos, nomeadamente o reconhecimento de padrões, a correlação entre features e o tratamento de outliers, coisas que não tinham sido muito trabalhadas na primeira tarefa.

Globalmente, estamos muito satisfeitos com o trabalho desenvolvido em ambas as tarefas em todas as fases, desde a exploração dos dados à descoberta dos algoritmos e construção dos modelos, apesar de haver espaço para melhorias e outras técnicas de Machine Learning possíveis de aplicar.