**Universidade do Minho**
Escola de Engenharia
Licenciatura em Engenharia Informática

# Unidade Curricular de Aprendizagem e Decisão Inteligentes

Ano Letivo de 2024/2025

# Trabalho prático - Grupo 4

Marco Soares Gonçalves (a104614)
Salvador Duarte Magalhães Barreto (a104520)
André Filipe Soares Pereira (a104275)
Leonardo Gomes Alves (a104093)

19 de maio de 2025

ADI

# Índice

# 1 Introdução

Neste projeto, no âmbito da unidade curricular de Aprendizagem e Decisão Inteligentes, foi solicitado o estudo e desenvolvimento de modelos de **Machine Learning** sobre dois conjuntos de dados distintos. O primeiro dataset, atribuído pela equipa docente por se tratar de um grupo com número par, diz respeito a empréstimos ("loans") e tem como objetivo prever o estado do empréstimo ("loan_status"). O segundo dataset foi escolhido livremente pelos elementos do nosso grupo e refere-se a dados sobre anúncios de "Airbnbs"e tem como objetivo prever o preço por noite. Todo o trabalho foi realizado utilizando a plataforma **KNIME**, que facilita a análise e permite a preparação e modelação dos dados de forma estruturada. Foram aplicadas técnicas de **classificação** e **regressão**, seguindo diferentes paradigmas de Machine Learning, com o intuito de maximizar a precisão e fiabilidade das previsões efetuadas.

# 2 Objetivos e metodologia

Como objetivos para a correta implementação temos, numa análise inicial, as 2 **variáveis alvo**:

- Dataset atribuído - "**loan_status**"(Approved/Pending/Rejected)"

- Dataset escolhido - "**price**"(Valor discreto)

Para além destes pretende-se demonstrar uma boa execução dos processos de exploração, modificação, modelação e avaliação e justificar coerentemente os passos tomados nesses processos, para isso seguimos uma metodologia e corroboramos todos os passos tomados com representações gráficas e imagens ilustrativas.

A metodologia adotada para a exploração e implementação dos modelos foi o **SEMMA** (Sample, Explore, Modify, Model and Assess). Assim, dividimos o processo, e a respetiva exposição neste relatório, em cinco etapas:

1. **S**ample / Amostragem: No caso dos datasets utilizados, não foi necessário realizar qualquer procedimento adicional nesta fase, para além da simples importação para o KNIME.

2. **E**xplore / Exploração: Exploração visual e/ou numérica das tendências presentes nos dados.

3. **M**odify / Modificação: Aplicação de todas as transformações e pré-processamentos necessários aos dados.

4. **M**odel / Modelação: Definição e aplicação das técnicas de construção de modelos de data mining, como redes neuronais artificiais, árvores de decisão, regressão linear, entre outras.

5. **A**ssess / Avaliação: Medição do desempenho dos modelos construídos com base em métricas adequadas, de acordo com o tipo de problema.

# 3 Dataset *Loans*

## 3.1 Exploração do dataset

Foram-nos fornecidos dois datasets, um **Dataset de treino** e um **Dataset de teste**,com 40000 linhas e 4500 linhas, respetivamente. Cada linha representa um pedido de empréstimo. Para além disso cada dataset possui 17 colunas:

- **date** : Data de nascimento
- **person_name** : Nome da pessoa (Nominal)
- **person_age** : Idade da Pessoa
- **person_gender** : Género da pessoa (Categórica)
- **person_education** : Nível de educação (Categórica)
- **person_income** : Salário anual
- **person_emp_exp** : Anos de experiência profissional
- **person_home_ownership** : Situação de habitação (Categórica)
- **loan_amnt** : Quantidade de empréstimo pedido
- **loan_intent** : Objetivo do empréstimo (Categórica)
- **loan_int_rate** : Taxa de juro do empréstimo
- **loan_percent_income** : Montante do empréstimo como percentagem do rendimento anual
- **cb_person_cred_hist_length** : Duração do histórico de crédito (anos)
- **credit_score** : Pontuação de crédito da pessoa
- **previous_loan_defaults_on_file** : Histórico de incumprimentos de empréstimos (Categórica)
- **tax** : Indicador fiscal do requerente (Categórica)
- **loan_status** : Estado do empréstimo (Categórica) (**Alvo**)

Relativamente às idades, o dataset apresenta a seguinte distribuição:

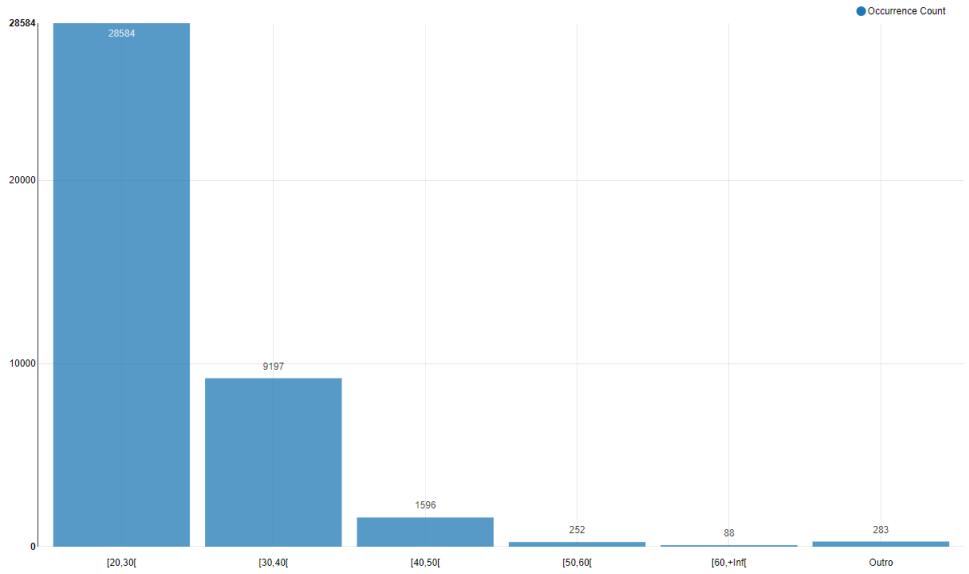



Figura 3.1: Distribuição das idades (Outro representa os valores em falta)

O gráfico da Figura 3.1 revela que a esmagadora maioria dos empréstimos é pedido por pessoas entre os 20 e 40 anos, sendo que a partir dos 60 anos a quantidade é insignificante. Para além disso podemos perceber que existem **prováveis** idades erradas, tal como 144 anos.

Quanto ao género das pessoas que solicitaram empréstimos, a distribuição encontra-se abaixo:

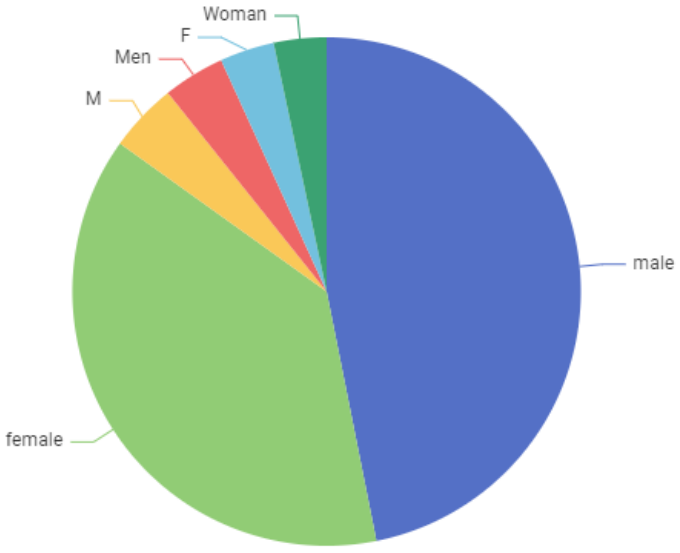



Figura 3.2: Distribuição dos géneros

Observando o gráfico da Figura 3.2, existem múltiplas formas de definir géneros idênticos, como, por exemplo "M"e "Men", que poderiam ser representados apenas por "male". Para além disso podemos verificar que a maioria dos empréstimos foi solicitada por indivíduos do **sexo masculino**.

Em relação ao nível de educação, obtemos a seguinte distribuição:

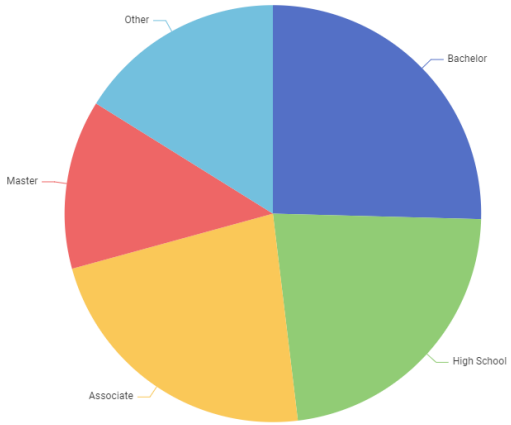



Figura 3.3: Distribuição dos níveis de educação

| | |
|---|---|
| Other | 6456 (16.14%) |
| *Bachelors* | ***591 (1.48%)*** |
| *Bachlor* | ***588 (1.47%)*** |
| *HS* | ***561 (1.40%)*** |
| *Assoc. Degree* | ***558 (1.40%)*** |
| *Associates* | ***553 (1.38%)*** |
| *BSc* | ***546 (1.36%)*** |
| *H-School* | ***524 (1.31%)*** |
| *Assoc* | ***519 (1.30%)*** |
| *Highschool* | ***518 (1.29%)*** |
| *Doctorate* | ***487 (1.22%)*** |
| *MSc* | ***335 (0.84%)*** |
| *Masters* | ***303 (0.76%)*** |
| *Mstr* | ***295 (0.74%)*** |
| *Dr.* | ***21 (0.05%)*** |
| *Doctoral* | ***20 (0.05%)*** |
| *Doctor* | ***13 (0.03%)*** |
| *PhD* | ***12 (0.03%)*** |
| *Doctrate* | ***12 (0.03%)*** |

Figura 3.4: Distribuição do Other

Podemos observar nas figuras acima que pessoas com nível de educação **"Bachelor"** pedem mais empréstimos. Para além disso, verifica-se novamente o problema de existirem formas distintas de representar a mesma informação (Ex: Bachelor e Bachelors)

No que diz respeito aos anos de experiência, a distribuição encontra-se representada na figura abaixo:

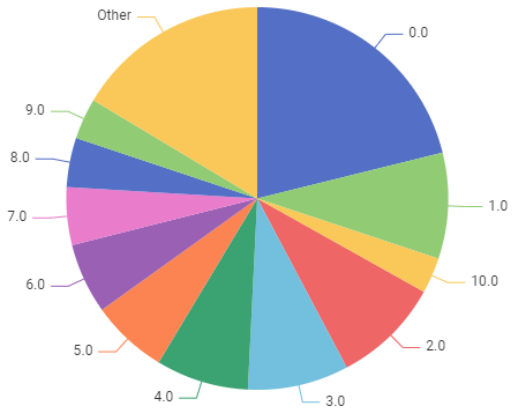



Figura 3.5: Distribuição dos anos de experiência

***Nota**: É importante mencionar que a categoria **"Other"** representa mais de 10 anos de experiência.*

A maioria dos pedidos de empréstimos é feita por pessoas **sem qualquer experiência profissional**. A frequência dos pedidos tende a diminuir à medida que os anos de experiência aumentam.

Sobre a situação de habitação dos requerentes, os dados estão organizados da seguinte forma:

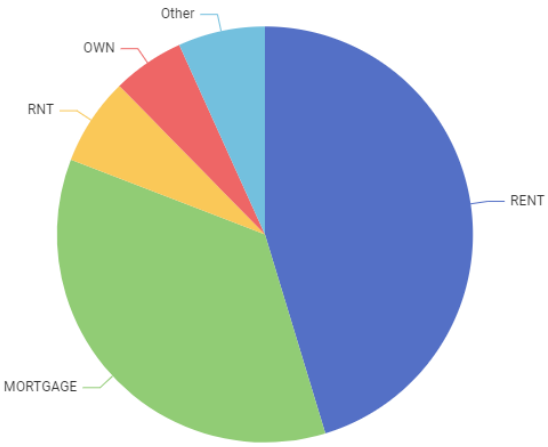



Figura 3.6: Distribuição da situação de habitação

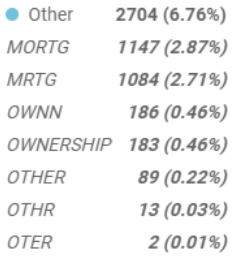



Figura 3.7: Distribuição do Other

Neste atributo encontramos novamente valores com nomes ligeiramente diferentes do campo ao qual assumimos pertencerem como "RNT"ou "OTER"que devem pertencer a "RENT"e "OTHER", respetivamente. Também podemos reparar que uma grande porção (52%) dos empréstimos foram requisitados por pessoas que pagam renda.

A seguir apresentamos a distribuição das quantias de empréstimo:

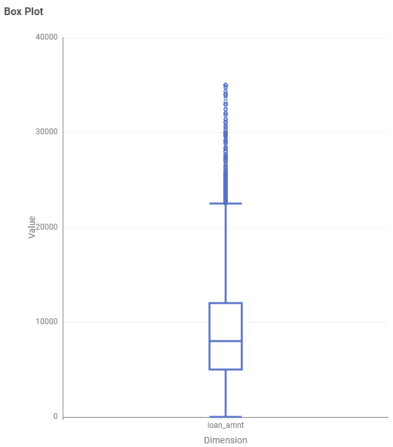



Figura 3.8: Distribuição da quantia de empréstimo

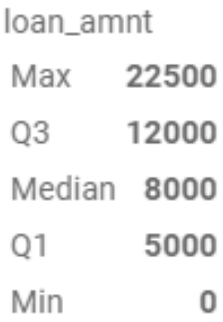



Figura 3.9: Distribuição do Other

Observamos uma grande concentração de valores no segundo quadrante, entre 5000 e 8000

unidades monetárias. No entanto, identificamos a existência de valores que ultrapassam significativamente o intervalo esperado, "**outliers**".

Relativamente à intenção dos empréstimos, a distribuição pode ser visualizada abaixo:

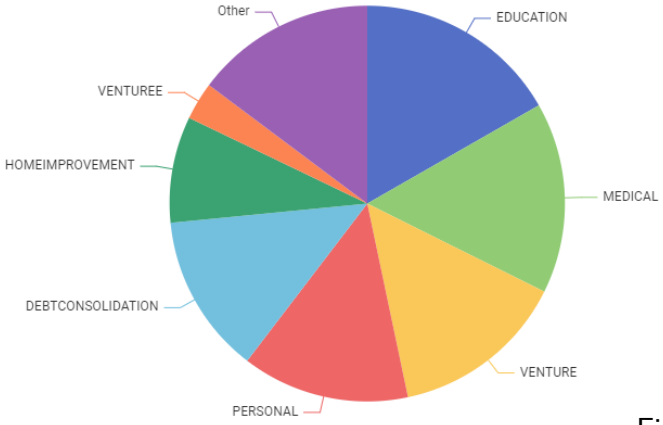



Figura 3.10: Distribuição de objetivo de empréstimo

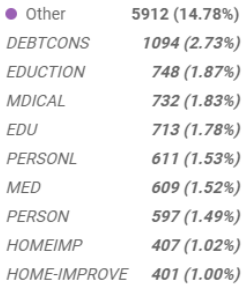



Figura 3.11: Distribuição do Other

Mais uma vez podemos verificar que a categoria "Other"contém formas distintas de representar a mesma informação ("EDUCATION"e "EDUCTION"). Para além disso podemos perceber que motivos para o pedido de empréstimo estão **bem distribuídos**.

A proporção entre o montante do empréstimo e o rendimento anual encontra-se a seguir:

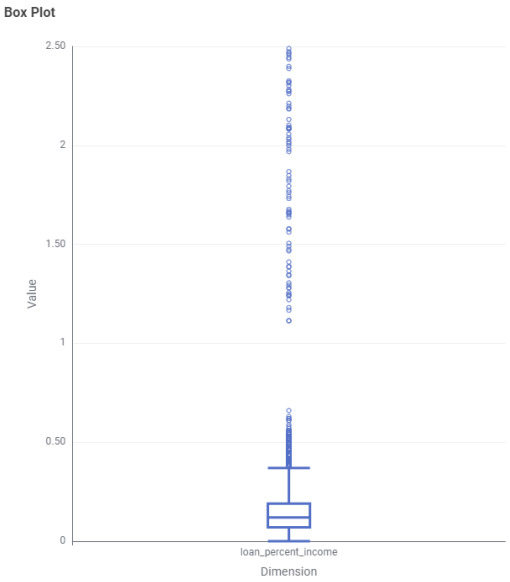



Figura 3.12: Proporção entre o montante do empréstimo e o rendimento anual

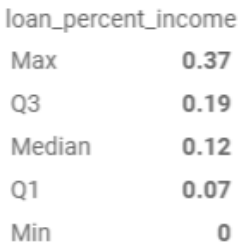



Figura 3.13: Distribuição do Other

Constatamos que existe um grande número de "**outliers**", muitos são muito superiores à esmagadora maioria dos valores no dataset.

No que toca ao credit_score e ao cred_hist_length verificamos que pessoas que solicitam mais créditos tendem a ter um **credit score mais elevado**, embora a tendência não seja muito acentuada, tal como demonstrado na figura 3.1:

relacao$_c$$reditos_c$$redit_s$$core.png$

Figura 3.14: Credit Scrore por Credit History Length

O histórico de incumprimentos de empréstimos apresenta a seguinte distribuição:

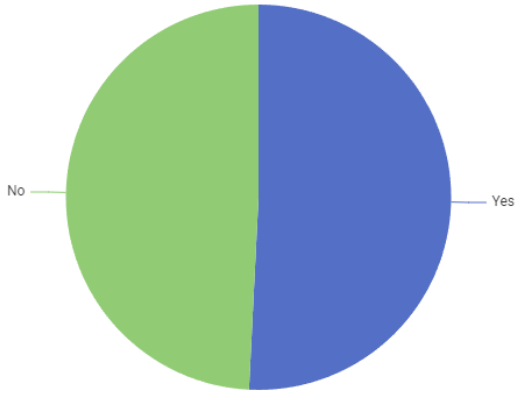



Figura 3.15: Histórico de incumprimento de empréstimos

O incumprimento dos empréstimos apresenta uma distribuição **balanceada** entre valores afirmativos (Yes) e negativos (No).

Por fim, a análise do estado dos empréstimos revela o seguinte:

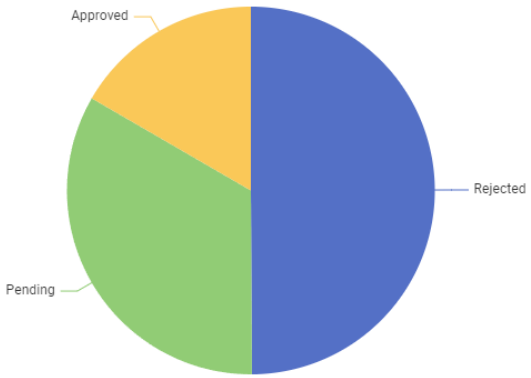



Figura 3.16: Loan Status

Verificamos um número significativamente superior de empréstimos "**Rejected**"em comparação com os "**Approved**", o que poderá indicar um critério de aprovação exigente ou pedidos com perfis de risco elevado.

## 3.2 Modificação

Começamos por aplicar alguns nodos "Rule Engine"a 4 dos atributos do dataset, sendo estes o "person_gender", "person_education", "home_ownership"e "loan_intent"; todos eles apresentavam vários valores sinónimos pelo que resolvemos condensa-los da seguinte forma:

1. **person_gender** — uniformizado para "F"e "M"
2. **person_education** — reduzido a "Master", "Doctor", "Bachelor", "Associate"e "High School"
3. **home_ownership** — reduzido a "RENT", "OWN"e "MORTGAGE"
4. **loan_intent** — reduzido a "PERSONAL", "EDUCATION", "MEDICAL", "VENTURE", "DEBTCONSOLIDATION"e "HOMEIMPROVEMENT"

A tabela seguinte demonstra os valores originais que foram "normalizados"e o novo valor que o substitui:

| Campo | Valor Original | Valor Normalizado |
|---|---|---|
| **person_gender** | Woman, female<br>Men | F<br>M |
| **person_education** | Highschool, H-School, HS<br>Assoc. Degree, Associates, Assoc<br>BSc, Bachlor, Bachelors<br>Mstr, Masters, MSc<br>Doctorate, PhD, Doctoral, Dr., Doctorate | High School<br>Associate<br>Bachelor<br>Master<br>Doctor |
| **home_ownership** | RNT<br>OWNN, OWNERSHIP<br>MORTG, MRTG<br>OTER, OTHR | RENT<br>OWN<br>MORTGAGE<br>OTHER |
| **loan_intent** | EDUCTION, EDU<br>MED, MDICAL<br>VENTUREE<br>DEBTCONS<br>HOME-IMPROVE, HOMEIMP<br>PERSON, PERSONL | EDUCATION<br>MEDICAL<br>VENTURE<br>DEBTCONSOLIDATION<br>HOMEIMPROVEMENT<br>PERSONAL |

Tabela 3.1: Normalização de valores com Rule Engine

Este pré-processamento garantiu maior consistência e fiabilidade nas etapas seguintes de análise e modelação. Realizamos também um filtro inicial de colunas que consideramos não ter potencial de melhorar os nossos modelos:

- **person_name** : O nome de uma pessoa não é relevante para prever o estado do empréstimo;
- **person_genre** : o género da pessoa teorizamos ser irrelevante e confirmamos com uma matriz de correlação(figura 3.17).
- **tax** - Este atributo e unitário tendo sempre o valor de "1", sendo absolutamente irrelevante, na matriz mencionada previamente verificamos que o tax não tem sequer possibilidade de apresentar uma correlação com outro atributo;

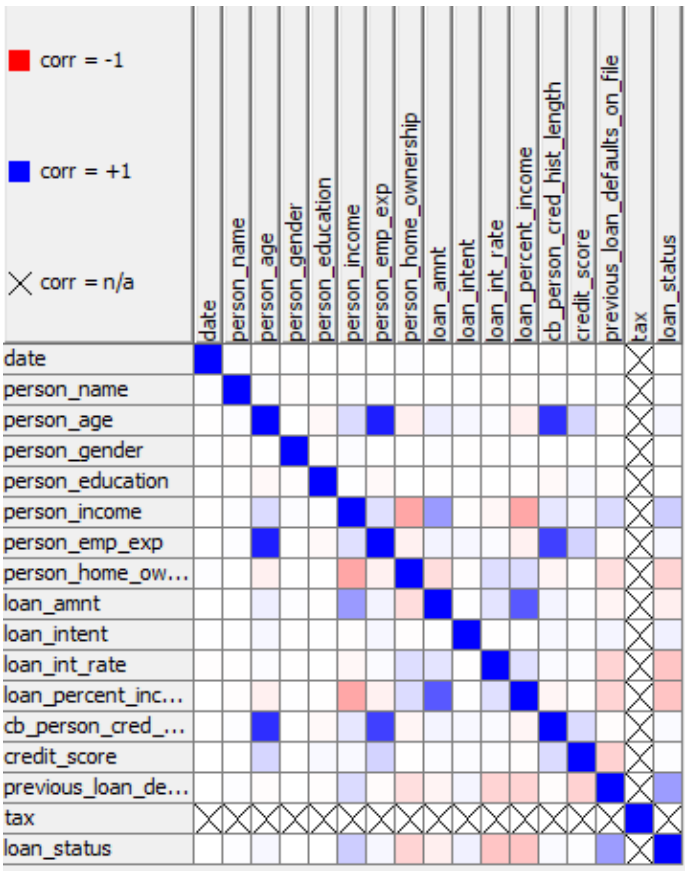



Figura 3.17: Matriz de correlação

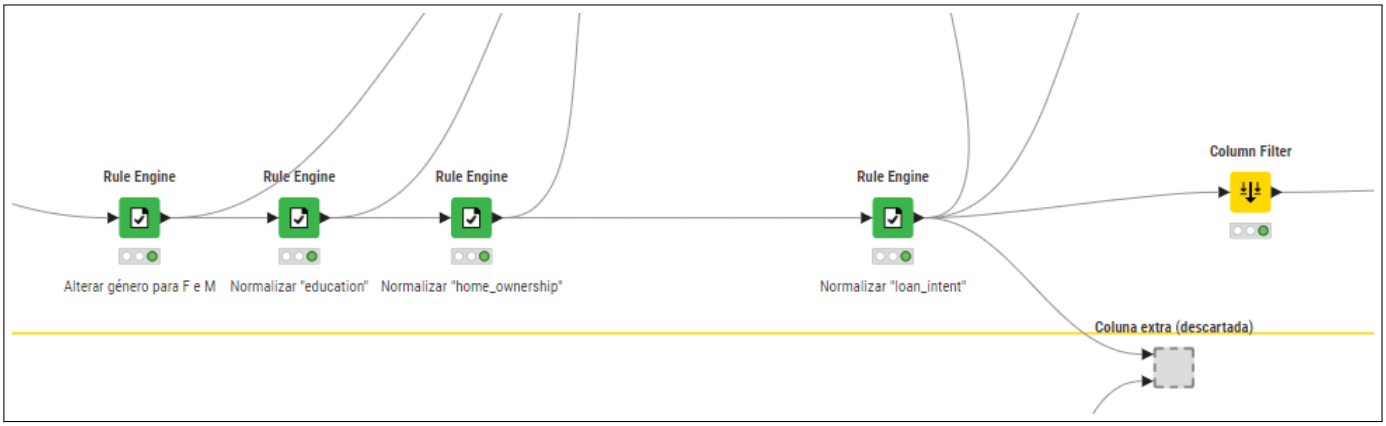



Figura 3.18: Rule engines / column filter no workflow

Na imagem anterior pode-se identificar um "*Metanode*"com o nome "Coluna extra (descartada)"aqui encontra-se um javasnippet que implementamos com o propósito de introduzir uma nova coluna com o número de empréstimos associados a cada pessoa. Infelizmente esta tentativa de criar um atributo que se correlacionasse ao nosso alvo caiu por terra ao analisarmos os resultados; quase nenhuma das pessoas tinha mais do que 1 empréstimo associado tornado os dados estatisticamente irrelevantes.

Para o resto das modificações de dados implementamos os seguintes nodos:

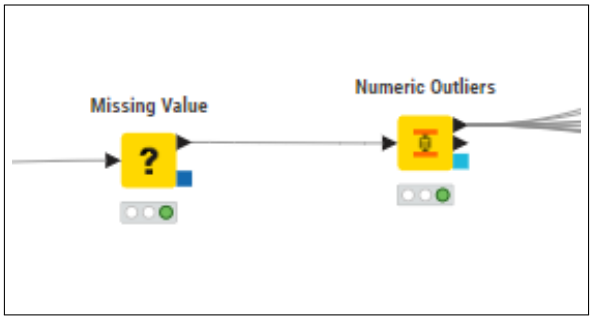



Figura 3.19: Nodos adicionais

Introduzimos um processamento de "***Missing Values***"e de "***Numeric Outliers***", uma vez que identificamos a falta de uma quantidade considerável de valores (figuras 3.1; tabela 4.1) e valores numéricos muito superiores ao expectável (figuras 3.8; 3.12; 3.1).

| **Atributo** | **Numero de "*Missing Values*"** |
|---|---|
| person_income | 393 |
| person_emp_exp | 336 |
| loan_percent_income | 360 |
| loan_int_rate | 252 |
| loan_amnt | 373 |
| credit_score | 328 |
| cb_person_cred_hist_length | 282 |
| **Total de linhas com 1 ou mais "*Missing Values*"** | 2279 |

Tabela 3.2: "*Missing values*"por coluna (dataset de treino)

Durante os nossos testes explorámos diversas estratégias de tratamentos de dados, algumas como normalização de valores, que se revelaram ser inadequadas para a melhoria dos modelos e, como tal, foram descartadas. Assim mantemos apenas os nodos mencionados:

1. Rule-engine
2. Column filter
3. Missing Value
4. Numeric Outliers

Algo que também implementamos foi o **"Binning"** em 3 variáveis (credit_score, loan_amnt e person_income). Um dos casos que consideramos ser mais relevantes foi o **credit_score**. Com este nó procuramos criar diferentes "classes"económicas na esperança que conseguíssemos melhorar o desempenho do modelo. Também adicionamos ao projeto um processo de **"Oversampling"** com recurso ao nodo "**SMOTE**", isto porque um problema do dataset é a discrepância entre a quantidade de entradas com o atributo alvo igual a "pending"e as outras

duas opções como é evidente na figura 3.16. (Para mais detalhes dos parâmetros utilizados recorrer à Folha "Definições genéricas"nos testes realizados:6.0.1)

Para a melhoria de resultados restou-nos a otimização das definições de parâmetros dos nodos de filtro (2-4). Conforme demonstrado no anexo de testes (6.0.1) o processo de otimização foi extenso e garantiu-nos uma melhoria significativa dos resultados iniciais. O processo levou a 4 alterações da implementação inicial:

1. Ignorar *missing values* ao invés de os remover
2. Incluir "person_emp_exp"no nodo de remoção de *Outliers*
3. Ajustar metodo do nodo de remoção de *Outliers*
4. Incluir coluna "person_gender"

(Detalhes sobre implementação inicial e resultados das alterações na folha de testes: 6.0.1)

Os testes dependeram dos modelos descritos em seguida para a avaliação e comparação de resultados.

## 3.3 Modelação

Como se trata de um problema de classificação decidimos testar os seguintes algoritmos de aprendizagem:

- **Random Forest** : Baseia-se em árvores de decisão, robusto a *overfitting* e eficaz em conjuntos de dados com muitas variáveis.
- **Gradient Boosted Trees** : Combina várias árvores de forma sequencial, focando-se na minimização do erro residual.
- **Decision Trees** : Modelo simples e fácil de interpretar.

Estas foram as abordagens escolhidas, pois podemos comparar como a nossa previsão se comporta com modelos mais simples e interpretáveis ("Decision trees"), com modelos mais avançados como o ("Random Forest Tree").

Houve algumas tentativas de modificação dos parâmetros dos nodos de aprendizagem associados a cada métodos, mas, dado que todas as alterações, com 1 exceção, resultavam em piores métricas de avaliação, mantivemos a alteração singular. Esta alteração foi a ativação de pruning nas Decision Trees para "MDL"e com "Reduced Error Pruning"que parece evitar algum *overfitting* e garantir melhores resultados.

(**Nota** : a seed utilizada foi consistente entre nodos e igual a "1498869038995572890")

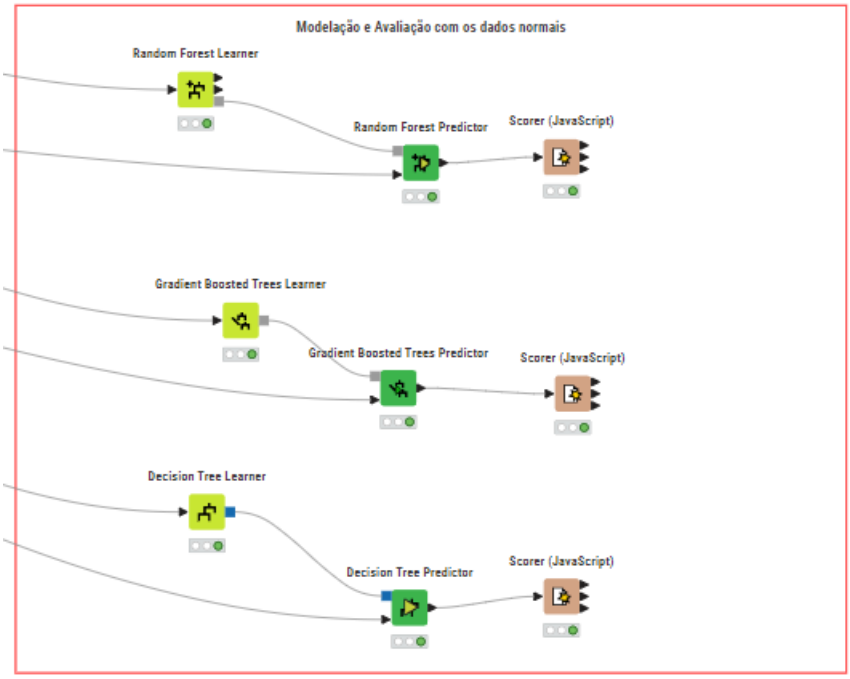
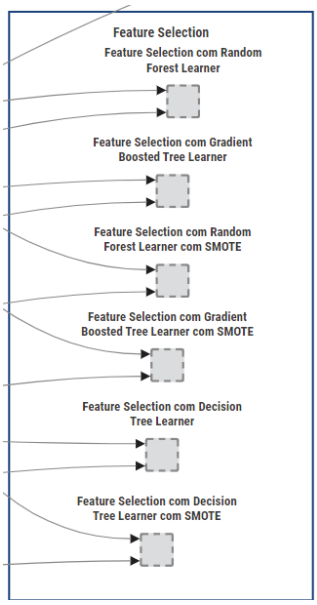



Figura 3.20: Modelação

### 3.3.1 Feature selection

Utilizamos também o **Feature Selection** com o objetivo de selecionar diferentes "features"que pudessem trazer melhorias de resultados em relação às escolhidas pela equipa. Para tal, aplicamo-lo aos diferentes tipos de tratamento de dados, nomeadamente ao SMOTE e ao tratamento genérico, e verificamos que os resultados obtidos eram piores que os nossos. Para além disso, uma vez que a utilização de *binning* piorou os resultados do modelo (tanto com tratamento de dados genérico como com SMOTE), optámos por não utilizar **Feature Selection** com esse tratamento, uma vez que a sua performance iria manter-se baixa.



## 3.4 Avaliação de resultados

A avaliação completa do nosso *workflow*, com os vários testes efetuados (binning, smote, etc..) pode ser verificada no Anexo 6.0.1, sendo o melhor valor da *accuracy* que obtivemos igual a **62.22%** e um valor de Cohen's Kappa de 0.334 , utilizando o **Decision Tree Predictor** (sem SMOTE e/ou Binning).

Este comportamento da *accuracy* é essencialmente influenciado pelo facto do modelo não conseguir prever corretamente as entradas cujo *target* tem o valor de **Pending**, tendo esse *target* um valor de 0.020 de F-score.

Para confirmar este comportamento recorrermos à utilização de **clusters**, de modo a organizar os dados em 3 clusters:

- Approved
- Rejected
- Pending

Olhando para a tabela abaixo, conseguimos então confirmar que no **cluster_0** existe uma quantidade de pedidos **Approved** maior do que os restantes, indicando que este consegue selecionar os pedidos com Approved. Já o **cluster_1 e cluster_2** tentam ambos organizar pedidos do tipo **Rejected**, uma vez que este é o que tem a maior quantidade, no entanto, tem imensa dificuldade a distinguir os pedidos **Rejected** dos **Pending**.

Esta dificuldade dos algoritmos de clustering em separar corretamente os dados entre as diferentes categorias de loan_status deve-se a que o nosso problema é supervisionado, enquanto o clustering é uma abordagem a problemas não supervisionados. Por este motivo, como os clusters não têm acesso à informação supervisionada, não é possível agrupar os grupos pelo atributo loan_status corretamente.

| Cluster<br>*String* | Sum(Rejected)<br>*Number (integer)* | Sum(Pending)<br>*Number (integer)* | Sum(Approved)<br>*Number (integer)* |
|---|---|---|---|
| cluster_0 | 1453 | 1239 | 1706 |
| cluster_1 | 2013 | 1570 | 857 |
| cluster_2 | 1228 | 829 | 304 |

Figura 3.21: Clusters do target

Por outro lado, as entradas cujo *target* tem valores de **Approved** e **Rejected**, apresentam valores relativamente superiores (comparativamente ao **Pending**) de F-score, 0.708 e 0.750, respetivamente.

Assim, podemos que concluir que as entradas com o *target* de valor **Pending**, por se tratar de um estado transitório, não é possível ser distinguido de forma consistente dos dois outros tipos de loan_status.

| | Approved (Predicted) | Pending (Predicted) | Rejected (Predicted) | |
|---|---|---|---|---|
| **Approved (Actual)** | 514 | 5 | 166 | 75.04% |
| **Pending (Actual)** | 205 | 12 | 981 | 1.00% |
| **Rejected (Actual)** | 48 | 14 | 1811 | 96.69% |
| | 67.01% | 38.71% | 61.22% | |

Overall Statistics

| Overall Accuracy | Overall Error | Cohen's kappa (κ) | Correctly Classified | Incorrectly Classified |
|---|---|---|---|---|
| 62.22% | 37.78% | 0.334 | 2337 | 1419 |

Figura 3.22: Resultados

Nos testes foi notável o destaque do algoritmo ***Decision Tree*** como superior em quase todas as métricas e parâmetros dos nodos na modificação de dados testados para este dataset.

A inclusão do nodo **SMOTE** e/ou de **Binning** provou-se fútil e piorou consistentemente os resultados obtidos de forma generalizada.

Finalmente vale destacar que a otimização dos nodos de modificação realizada resultou num aumento de 61.63% para 62.22% de *accuracy*; não foi significativa (aumento relativo de aprx. 1%) e diminuiu até o resultado de algumas métricas.

# 4 Dataset Escolhido ("*Airbnb*")

## 4.1 Exploração do dataset

Decidimos trabalhar sobre o dataset "*Airbnb NYC Price Prediction*"visto que este contém uma grande variedade de dados, que inclui **missing values**, **valores numéricos** e **strings**, o que nos permite utilizar diversas técnicas de pré-processamento de dados. Para além disso, o problema em questão é de regressão, o que cria um contraste com o dataset anterior, no qual o problema era de classificação. Deste modo, conseguimos abordar ambos os tipos de problemas lecionados.

Com este dataset, temos como atributo alvo **descobrir o preço por noite**, ou seja, um valor discreto, associado a determinado anuncio no site Airbnb. Cada linha representa um anúncio e o dataset engloba 48895 entradas, com cerca de 10089 *missing value*. Cada entrada possui 16 colunas:

- **id** : Identificador único do anúncio
- **name** : Nome do anúncio (Nominal)
- **host_id** : Identificador do anfitrião
- **host_name** : Nome do anfitrião (Nominal)
- **neighbourhood_group** : Zona principal da cidade (Categórica)
- **neighbourhood** : Bairro específico (Categórica)
- **latitude** : Latitude da localização
- **longitude** : Longitude da localização
- **room_type** : Tipo de alojamento (Categórica)
- **price** : Preço por noite (**Alvo**)
- **minimum_nights** : Número mínimo de noites por reserva
- **number_of_reviews** : Número total de avaliações
- **last_review** : Data da última avaliação

- **reviews_per_month** : Número médio de avaliações por mês
- **calculated_host_listings_count** : Número de anúncios ativos do anfitrião
- **availability_365** : Número de dias disponíveis por ano

Relativamente às zonas principais da cidade, o dataset apresenta a seguinte distribuição:

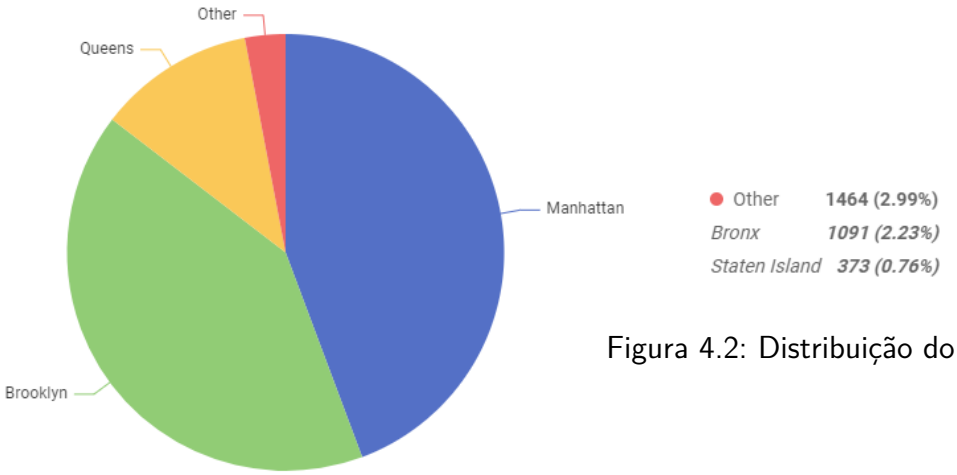



Figura 4.2: Distribuição do Other

Figura 4.1: Distribuição das vizinhanças

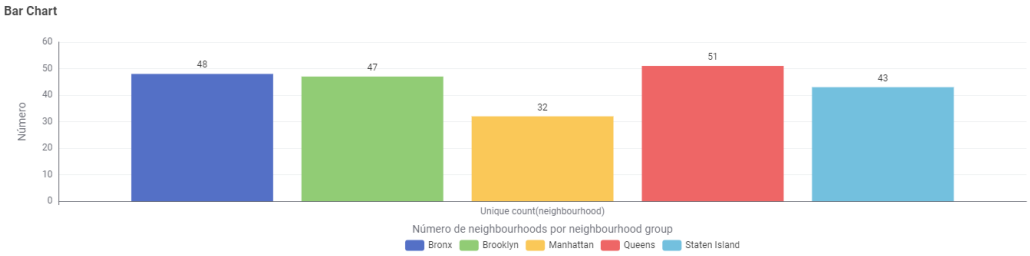



Figura 4.3: Número de neighbourhoods por neighbourhood_group

Como podemos ver no pie chart da Figura 4.1, a maioria dos anúncios pertencem à zona de **"Manhattan"**. No entanto, o gráfico de barras da Figura 4.3, mostra que **"Queens"** é a zona com maior diversidade de bairros representados.

Relativamente ao tipo de quarto, temos a seguinte distribuição:

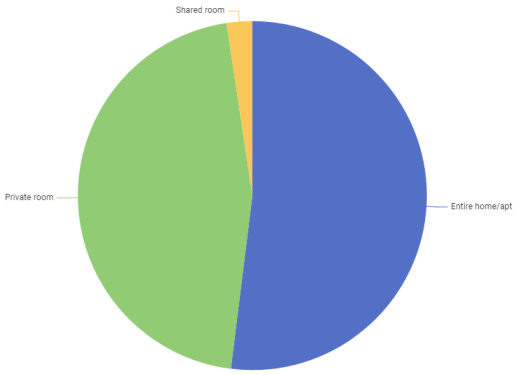



Figura 4.4: Tipo de quarto

Podemos observar que os anúncios são maioritariamente de casas/apartamentos inteiros.

No que toca a preços, o dataset tem a seguinte distribuição:

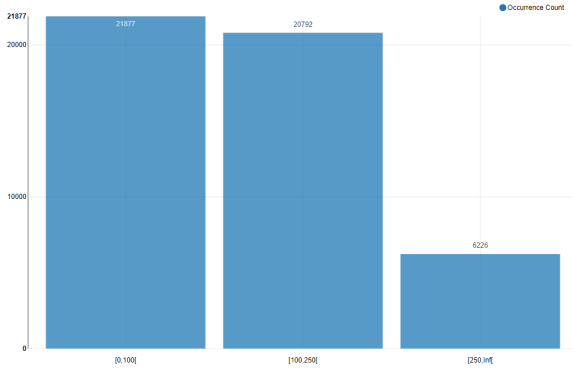



Figura 4.5: Preço por noite divido em bins

Para facilitar a analisar, os preços por noite foram agrupados em bins de [0,100[, [100,250[ e [250,inf[. Verificamos que o dataset está bem distribuído entre os dois primeiros bins, embora contenha também alguns anúncios com preços mais elevados.

Sobre o preço por dia em cada vizinhança, o dataset segue a seguinte distribuição:

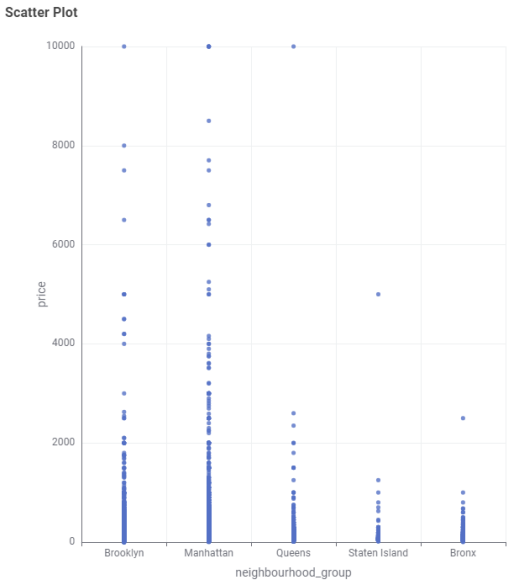



Figura 4.6: Preço por noite por zona da cidade

Na Figura 4.6, podemos observar que **"Brooklyn** e **"Manhattan** apresentam uma maior concentração de preços elevados, o que sugere que são as zonas mais caras. **"Queens"**, **"Staten Island"** e **"Bronx"** tendem a ter preços mais baixos, com a maioria dos preços abaixo de 3000 dolares, apesar de apresentarem **outliers** com preços bastante elevados em relação à média de preços da zona.

No que toca à distribuição das avaliações por preço por noite, esta é a seguinte:

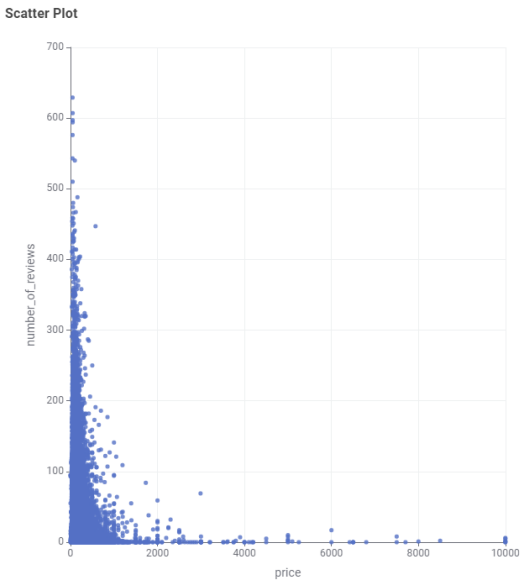



Figura 4.7: Avaliações por preço por noite

Da Figura 4.7, observamos que o número de avaliações diminui à medida que o preço por noite aumenta, o que pode traduzir uma maior procura por Airbnbs com preços por noite mais baixos.

Sobre o preço por tipo de quarto, o dataset tem a seguinte distribuição:

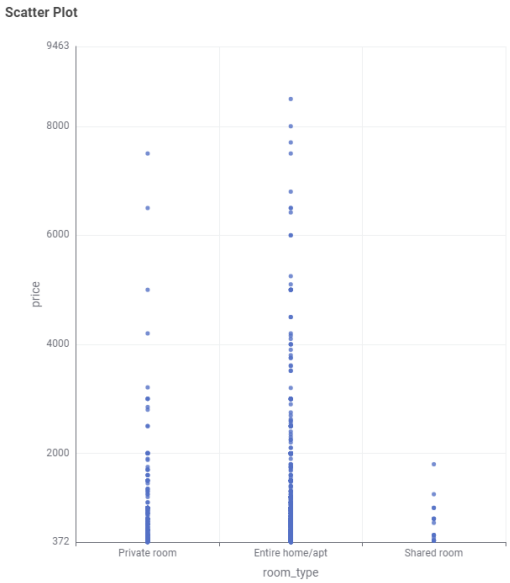



Figura 4.8: Preço por tipo de quarto

Observando a Figura 4.8, os "apartamentos/casas inteiros"são os mais caros, apresentando uma maior concentração de preços elevados, incluindo os outliers maiores. Por outro lado, os "quartos partilhados"tendem a ser as opções mais económicas, com valores mais baixos e menos dispersados.

## 4.2 Modificação

Começamos por remover linhas que não consideramos válidas para os nossos modelos, ou seja, linhas cujo "host_name"e "name"são valores nulos e linhas cujo valor do preço por noite é igual a 0. Para tal, utilizamos os nodes "Rule-based Row Filter"e "Row Filter"respetivamente.

De seguida fizemos o tratamento de "Missing Values":

| **Atributo** | **Numero de "*Missing Values*"** |
|---|---|
| last_review | 10052 |
| host_name | 21 |
| name | 16 |
| reviews_per_month | 10052 |
| **Total de linhas com 1 ou mais "*Missing Values*"** | 10036 |

Tabela 4.1: "*Missing values*"por coluna (dataset de treino)

Devido ao tratamento de *missing values* acima, a coluna *reviews_per_month* ficou com várias entradas de valor -1.0 de forma a ser possível distinguir as entradas cuja essa *feature* é inválida. Para corrigir esses valores, usamos um node *Java Snippet* que, quando o valor é -1.0 e a availability_365 é maior que 0, multiplica o number_of_reviews por 30 e divide pela availability_365, se não, coloca a 0.

Este comportamento passa por tentar calcular o número médio de *reviews* por mês, no entanto, reparou-se mais à frente que este tratamento tornou-se inútil, uma vez que entradas cuja feature *reviews_per_month* seja nula (*missing value*), é precisamente por não haver nenhuma avaliação, ou seja, corresponde a um novo AirBnB.

De seguida, realizamos um filtro de colunas que consideramos não ter potencial para melhorar os nossos modelos:

- **id**, **name** e **host_name**: o identificador do anúncio, o nome do anúncio e o nome do inquilino não são relevantes para prever o preço por noite do AirBnB;
- **neighbourhood**: como havia uma grande diversidade de bairros, decidimos remover por considerar que podia afetar negativamente o trainer;
- **last_review**: por ser uma data removemos por considerar que não seria relevante para o modelo.

É importante mencionar que todo este tratamento foi posteriormente confirmado através da avaliação.

Quanto ao tratamento de *outliers*, utilizámos as seguintes definições do nodo *Numeric Outliers*, excluindo trivialmente as colunas *host_id*, *latitude* e *longitude*, uma vez que tratá-las como *outliers* não faz sentido dado o seu significado.

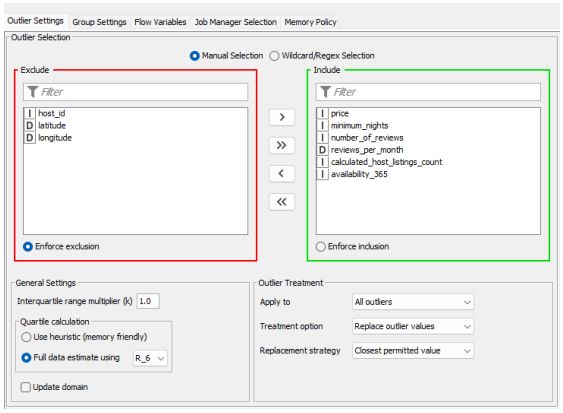



Figura 4.9: Tratamento de outliers

Realizamos também **Binning** em 3 bins de igual frequência com as colunas que consideramos relevantes(minimum_nights, number_of_reviews, reviews_per_month, calculated_host_listings_count, availability_365) para a modelação e avaliação com binning. Após os vários testes realizados, chegámos à conclusão que o número de bins que nos dava melhores resultados era 11.

## 4.3 Modelação

Como se trata de um problema de regressão, decidimos testar os seguintes algoritmos de aprendizagem:

- Linear Regression Learner: Baseia-se em regressão linear;
- Simple Regression Tree Learner: Utiliza uma única árvore de regressão, seguindo o algoritmo **Classification and Regression Trees**.

Escolhemos estes modelos uma vez que são os ideais face ao nosso problema, um problema de regressão.

Quanto à configuração dos nodos dos *Learners*, foram utilizados os valores *default* do Knime, uma vez que foram estes que apresentaram melhores resultados.

Utilizámos ainda os nodos *X-Partitioner* e *X-Aggregator*, aplicando assim *cross-validation*, de forma a reduzir o *overfitting*. Estes nodos foram configurados com um número de validações de 10, *Stratified Sampling*, utilizando a *Random Seed* escolhida pelo nosso grupo.

(**Nota**: a seed utilizada foi consistente entre nodos e igual a "438929562")

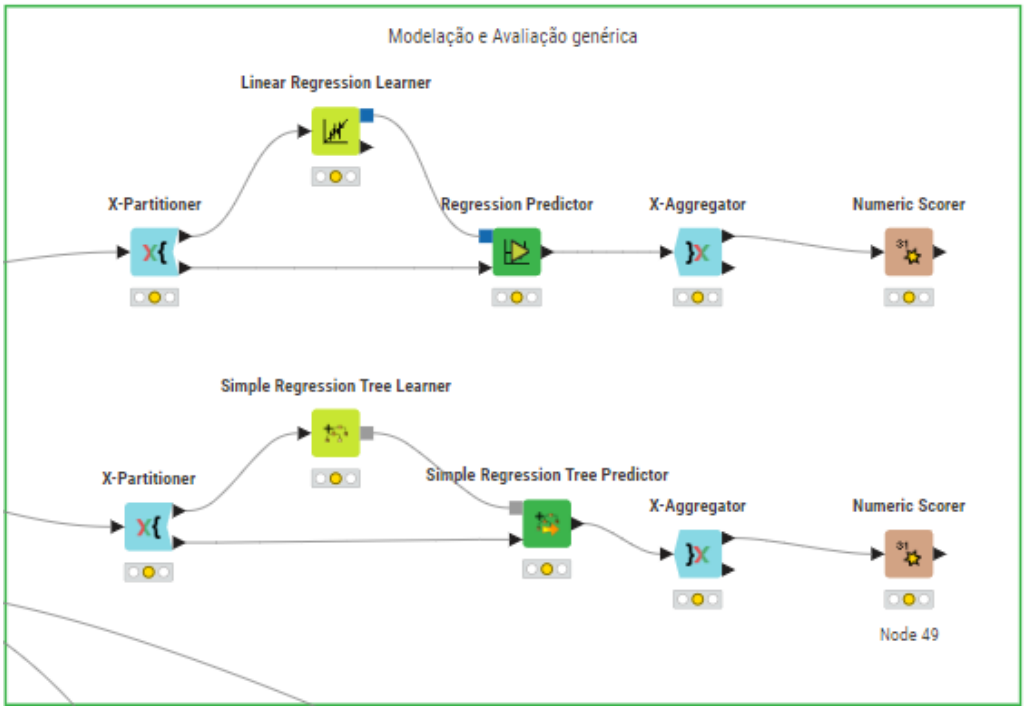



Figura 4.10: Modelação

### 4.3.1 Feature Selection

De forma idêntica ao *Feature Selection* do **dataset par**, aplicámos também este aos dois diferentes tipos de modificações, com e sem Binning, mas agora com o objetivo de maximizar o atributo $R^2$.

Os resultados obtidos destes foram que efetivamente, assim como no primeiro dataset, as *features* que tínhamos escolhido inicialmente foram as que apresentavam melhores resultados.

### 4.3.2 Redes Neuronais

De forma a implementar as redes neuronais, uma vez que estas apenas suportam trabalhar com valores numéricos, tivemos que fazer a correta normalização dos valores.

Uma vez que utilizámos redes neuronais tanto para a modelação com e sem Binning, foi necessário fazer diferentes modificações em cada um deles:

- **Sem** Binning - Aplicar *One Hot Encoding* nas *features* neighbourhood_group e room_type, e normalizar os restantes valores;
- **Com** Binning - Aplicar *One Hot Encoding* nas *features* neighbourhood_group, room_type e nas features de Binning, e normalizar os restantes valores.

Em ambos os tratamentos de dados, testámos normalizar também a feature host_id, no entanto, os melhores resultados foram obtidos quando esta era filtrada.

Relativamente ao nodo learner das Redes Neuronais, **RProp MLP Learner**, a configuração que obtém os melhores resultados é a seguinte:

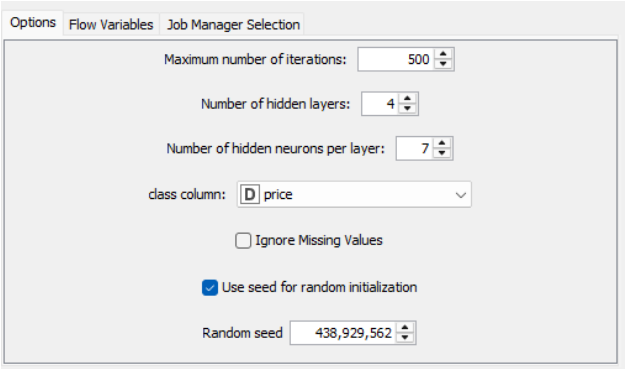



Figura 4.11: Configurações do RProp MLP Learner

A seguir, consta o nosso workflow genérico das redes neuronais.

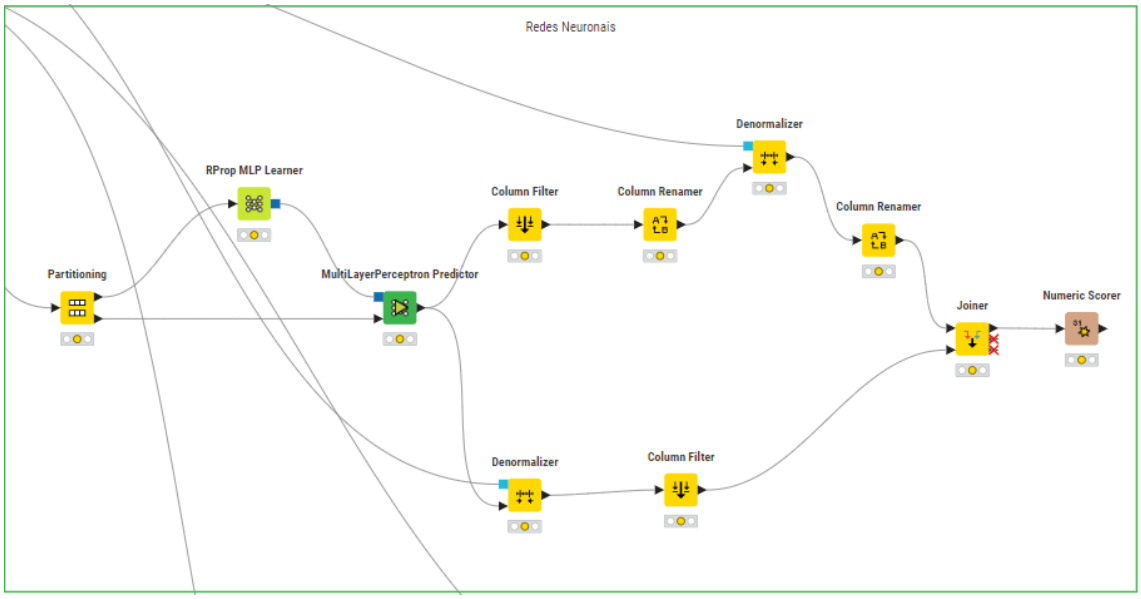
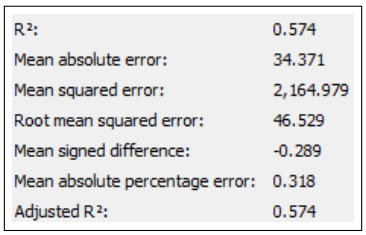



Figura 4.12: Workflow das redes neuronais

## 4.4 Avaliação

A avaliação completa do nosso *workflow*, com os vários testes efetuados (sem pré-processamento/binning) pode ser verificada no Anexo 6.0.1, sendo o melhor valor do $R^2$ igual a **0.574** e um valor de **MAE** de 34,371 , utilizando **Redes neuronais** (sem processamento extra).



| | |
|---|---|
| R²: | 0.574 |
| Mean absolute error: | 34.371 |
| Mean squared error: | 2,164.979 |
| Root mean squared error: | 46.529 |
| Mean signed difference: | -0.289 |
| Mean absolute percentage error: | 0.318 |
| Adjusted R²: | 0.574 |

Figura 4.13: Melhores resultados

Numa fase final dos testes foi notado um lapso na falta da inclusão da latitude e longitude no modelo baseado em **Redes neuronais**, mas, mesmo antes desse ajuste, foi possível verificar um aumento do $R^2$ de um valor inicial de **0.509** para **0.521**, com a otimização dos nodos de tratamento de dados. Trate-se de um aumento relativo de aprx. **2%** o que é mais significativo do que os resultados com o dataset atribuído mas, ainda assim, de baixo impacto nos resultados.

A inclusão de **binning** revelou ser útil para todos os modelos exceto aquele que apresentou os melhores resultados. O modelo baseado em regressão linear apresentou um valor de $R^2$ igual **0.510** sem binning e **0.517** com binning nas nossas iterações finais dos testes.

Dados os resultados redes neuronais destacaram-se como o método de treino ideal para este dataset (dos 3 testados) seguido da regressão linear. A regressão simples revelou ser extremamente fraca com um resultado de $R^2$ máximo igual a **0.250**, menos de metade daquilo que foi possível com os outros 2 métodos.

# 5 Conclusão

Ao longo deste projeto, explorámos e analisámos dois conjuntos de dados distintos, aplicando metodologias de *Machine Learning* com o objetivo de desenvolver modelos de previsão fiáveis. Utilizámos a plataforma **KNIME** como ferramenta principal, que se revelou extremamente útil ao proporcionar uma abordagem modular e visual ao processo de exploração, modificação, modelação e avaliação dos dados.

Foram realizados testes extensivos com diferentes abordagens de pré-processamento, algoritmos de aprendizagem e técnicas auxiliares como **SMOTE**, **Binning** e **Feature Selection**. Estes testes permitiram-nos perceber que, por mais estruturado que seja o processo, é frequentemente imprevisível determinar antecipadamente o que irá ou não otimizar um modelo. Muitas vezes, alterações intuitivamente benéficas acabaram por degradar a performance, enquanto outras, aparentemente insignificantes, trazem melhorias marginais.

Apesar das **limitações dos datasets** (como desbalanceamento de classes no caso dos empréstimos ou elevada variabilidade no preço do Airbnb), conseguimos obter **resultados satisfatórios**. No dataset dos empréstimos, destacaram-se os Decision Trees, com uma accuracy de **62.22%**. No caso do Airbnb, as redes neuronais mostraram-se mais eficazes, atingindo um valor de $R^2$ de **0.574**.

No final, a **principal conclusão** que tiramos deste projeto é que o desenvolvimento de modelos eficazes depende fortemente da experimentação e iteração constante e, acima de tudo, da **qualidade do dataset**, com um dataset medíocre é inviável criar um modelo eficaz. Não existe um modelo único superior a todos os outros, cada dataset requer uma abordagem única. O uso do **KNIME** facilitou este processo iterativo, permitindo-nos testar rapidamente diferentes **pipelines** e tirar conclusões baseadas em dados concretos.

Como **sugestões finais** para possíveis melhorias dos modelos, para além de alterações aos próprios datasets ou aos processos de recolha de dados dos mesmos, é possível:

1. **Testes mais aprofundados nos nodos de aprendizagem**: Embora tenhamos utilizado configurações default na maioria dos algoritmos, uma linha clara de melhoria seria a realização de uma panóplia extensa de testes com alterações dos hiperparâmetros, especialmente em modelos como **Random Forest** e **Gradient Boosted Trees**, onde opções como o número de árvores, profundidade máxima ou taxa de aprendizagem têm impacto direto na performance.

2. **Combinações variadas de Binning e SMOTE**: Embora os testes realizados tenham mostrado que nem sempre o **SMOTE** ou o **Binning** trazem melhorias, a realidade é que as combinações testadas foram limitadas. Existe espaço para explorar outras formas de

Oversampling, ou aplicar SMOTE apenas a determinados clusters previamente identificados, o que pode permitir um reequilíbrio mais eficaz da classe **Pending**, por exemplo. Da mesma forma, o Binning pode ser aplicado com diferentes critérios (frequência, largura fixa, binning adaptativo) e sobre outras variáveis ainda não exploradas.

3. **Pipelines personalizados**: A criação de pipelines distintos para diferentes segmentos do dataset pode melhorar a performance global. Por exemplo, aplicar tratamentos de outliers e missing values distintos consoante os atributos associado a cada empréstimo, em vez de aplicar transformações globais.

# 6 Referências

**Dataset escolhido** - https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data/data

### 6.0.1 Folha de testes

- Ficheiro "**ADI-Grupo4-2025-Testes-de-Modelos**"(incluído na pasta do projeto)