Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Aprendizagem e Decisão Inteligentes

Ano Letivo de 2021/2022

Relatório do Trabalho Prático

Grupo 40

A75481 José Mendes
A93189 Bernardo Saraiva
A93204 José Gonçalves
A93232 Rui Moreira

8 de maio de 2022

AD I

Índice

1 Introdução

O presente projeto enquadra-se na unidade curricular de Aprendizagem e Decisão Inteligentes, na qual foi proposta a conceção e otimização de modelos de *Machine Learning*, através da plataforma KNIME.

O desafio apresentado visa desenvolver um projeto utilizando modelos de aprendizagem abordados ao longo do semestre e tem por base duas fases principais.

A primeira cujo objetivo é consultar, analisar e selecionar um *dataset* de entre os possíveis de diferentes fontes, tais como *Google Dataset Search* e *Kaggle*.

A segunda fase, passa pela conceção e otimização dos modelos tanto para um *dataset* selecionado pela equipa docente, como para o *dataset* cuja seleção foi inteiramente da responsablidade do grupo de trabalho, sendo que para este último se recorreu à plataforma *Kaggle* para a obtenção do dataset. É ainda nesta fase que se faz a respetiva análise crítica dos resultados através da conceção de modelos de visualização, com o intuito de compreender inteiramente o dataset e as suas especificações.

2 Metodologia e aplicação

De modo a que o desenvolvimento de um projeto de *Machine Learning*, utilizando os modelos de aprendizagem, seja realizado com a maior qualidade possível, é importante seguir-se uma metodologia de extração de conhecimento. A metodologia seguida nos dois casos de estudo presentes, tanto na Previsão da Qualidade de Vinhos como na Previsão do Preço do Bilhete de Voos, foi a CRISP-DM, que contém seis etapas.

1 - *Business Understanding*, passou por diferentes períodos. Primeiramente, a parte de análise e seleção de um *dataset* de entre os que estão acessíveis a partir de fontes disponibilizadas. Seguidamente, passou-se para a perceção do *dataset* selecionado, onde se estudou em detalhe aquilo que são um dos vários tipos de voo, bem como as características dos vinhos para que seja possível um bom entendimento do que se está a desenvolver.

2 - A etapa de *Data Understanding*, onde se analisou cada atributo percebendo-se se estes teriam ou não relevância para os temas em estudo.

3 - A etapa de *Data Preparation*, que assenta na limpeza de dados, onde se aplicou técnicas básicas e avançadas e comprovou-se a análise anterior levando à remoção de atributos sem relevância, deixando assim os dados preparados para a fase seguinte.

4 - A etapa de *Modeling*, onde se aplicaram diferentes modelos com diversos parâmetros de forma a perceber qual deles o melhor para ser aplicado aos dados em questão.

5 - Por fim, a etapa de *Evaluate* baseou-se na comparação entre os dados obtidos através da aplicação dos

modelos e os objetivos dos negócios.

Uma vez que o projeto se enquadra numa unidade curricular, a etapa de *Deployment*, não é aplicada nos casos de estudo, já que o projeto foi desenvolvido em contexto académico.

3 Arquitetura do Workflow Knime

Na construção do workflow foram seguidas algumas boas práticas que se descrevem a seguir:

- **Anotações** - cada nodo tem associado a descrição do seu objectivo.
- **Metanodos** - algumas tarefas mais complexas, que exigem múltiplos nodos foram compactados num único nodo (metanodo).
- **Secções** - principais fases do pipeline de machine learning seguidas na análise deste problema.

A adoção destas práticas permitem a escalabilidade do processo, no sentido de facilitar a administração e crescimento do workflow por parte da equipa.

Foram criadas 5 secções:

- Dataset Reader (Roxo)
- Data Preparation (Verde)
- Data Visualization (Magenta)
- Models Learners (Laranja)
- Models Analysis (Azul)

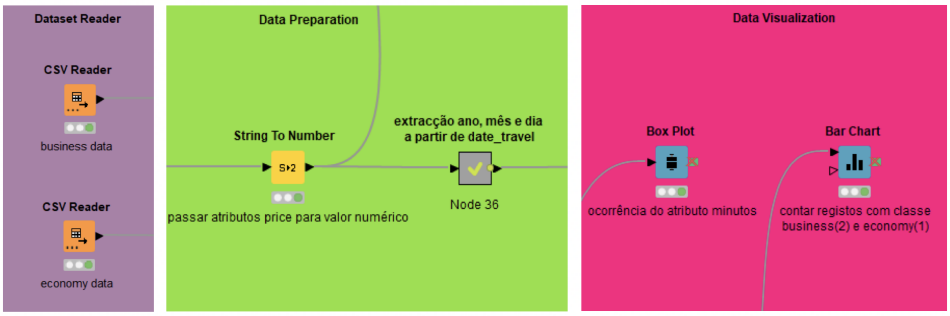

Figura 3.1: Excertos do workflow relativos às secções readers, preparation e visualization

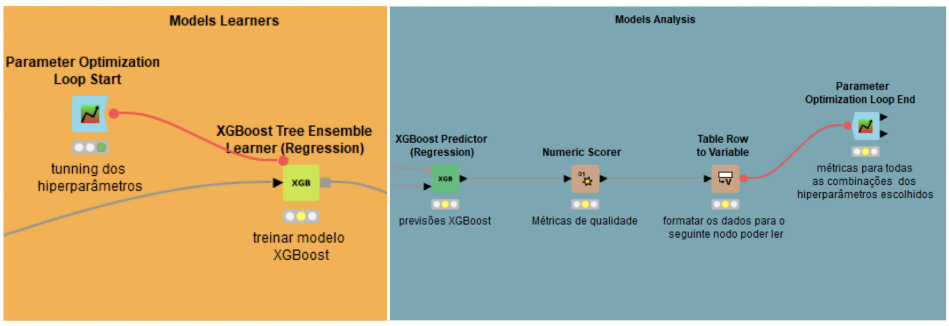

Figura 3.2: Excertos do workflow relativos às secções models learners e models analysis

4 Grupo Wines

4.1 Domínio e objetivos

O dataset analisado no presente capítulo é referente ao 1º tema proposto pela Equipa Docente, uma vez que o grupo é o Grupo 40. Deste modo, corresponde ao tema *"Wine Quality Classification"*, e tem como base a análise e classificação da qualidade dos vinhos (sendo este o *target*), com base nas suas propriedades e características químicas.

Relativamente ao presente estudo, explica-se em seguida os principais objetivos:

- A construção de um modelo capaz de efetuar uma boa previsão do atributo **quality** de um vinho;
- Analisar relações entre os atributos que possam influenciar a qualidade do vinho;

Assim, ao longo do presente estudo, utilizar-se-á a plataforma KNIME, junto com as suas técnicas de preparação, tratamento e visualização como mecanismo para alcançar os objetivos apresentados, bem como gerar conclusões acerca do dataset em estudo.

4.2 Exploração, Visualização e Tratamento dos Dados

4.2.1 Descrição dos atributos do dataset

Inicialmente, de modo a ter uma perceção do problema em questão, começou-se por fazer a leitura dos atributos do dataset, sendo eles:

- **fixed acidity** - variável continua que representa os ácidos constantemente envolvidos no vinho
- **volatile acidity** - variável contínua que representa a quantidade de ácido acético no vinho. Em níveis bastante elevados poderá levar a um sabor desagradável e avinagrado.
- **citric acid** - variável contínua que representa a caracteristica de adicionar sabor e frescura ao vinho
- **residual sugar** - variável continua que representa a quantidade de açúcar presente no vinho após ciclos de fermentação
- **chlorides** - variável contínua que representa a salinidade do vinho
- **free sulfur dioxide** - variável contínua com propriedades anti-microbianas.
- **total sulfur dioxide** - concentração total de SO2. Representado através de uma variável contínua
- **density** - Descreve a densidade do vinho através de uma variável contínua

- **pH** - Descreve o quão acido ou básico é o vinho. É representado numa variável contínua
- **sulphates** - aditivo com propriedade antimicrobial e antioxidante representada através de uma variável contínua
- **alcohol** - variável contínua que representa o volume alcoolico do vinho.
- **quality** - variável binária (0 ou 1) que corresponde à classificação do vinho em *good* ou *bad*. **Corresponde ao target** deste estudo de classificação

4.2.2 Exploração dos dados

Após a leitura e análise da descrição dos atributos, foi necessário ler o dataset (*winequality-red.csv*). Para isto foi usado o nodo **CSV Reader**.

Numa fase inicial, começou-se por efetuar a **Visualização dos Dados**, com o objetivo de analisar e compreender o dataset. Para tal, recorreu-se aos nodos **Statistics**, **Linear Correlation**, **Box Plot** e **Data Explorer**.

Ao analisar a Matriz de Correlação da Figura 4.1, foi possível inferir que características, como o *free sulfur dioxide* e o *total sulfur dioxide*, têm uma correlação bastante positiva, respetivamente 0.6677. Do mesmo modo, a *fixed acidity* com o *citric acid* e a *density* apresentam uma correlação de 0.6717 e 0.6680, respectivamente. Assim, podemos concluir que estes pares dois atributos estarão em grande parte a fornecer o mesmo conhecimento ao modelo, pelo que é de se esperar que o atributo *density* referente a um vinho esteja intrinsecamente relacionada ao seu *fixed acidity*, por exemplo. É também importante destacar que o campo *ph* se relaciona com *fixed acidity* com uma correlação bastante negativa, sendo -0.6830.

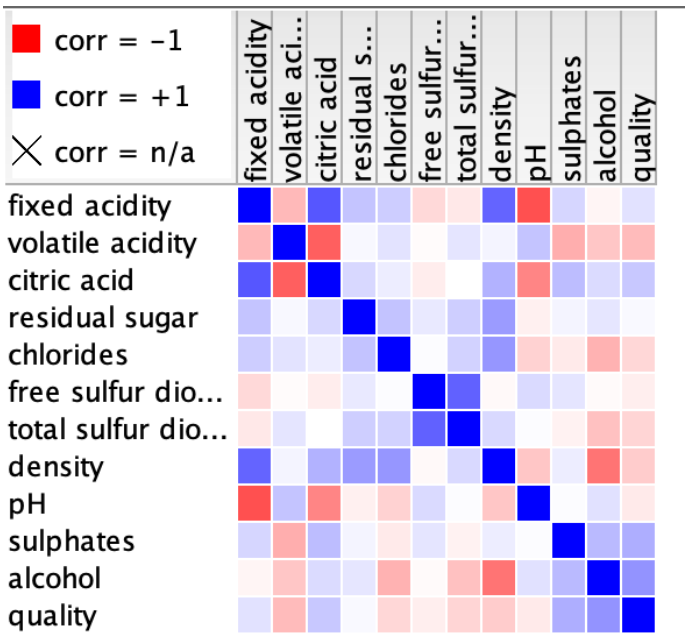

Figura 4.1: Matriz de Correlação

Em sequência, verificou-se através do **Box Plot** da Figura 4.2, que havia uma elevada discrepância em dados da coluna *total sulfur dioxide*, pelo que se decidiu efetuar um tratamento de *outliers*.

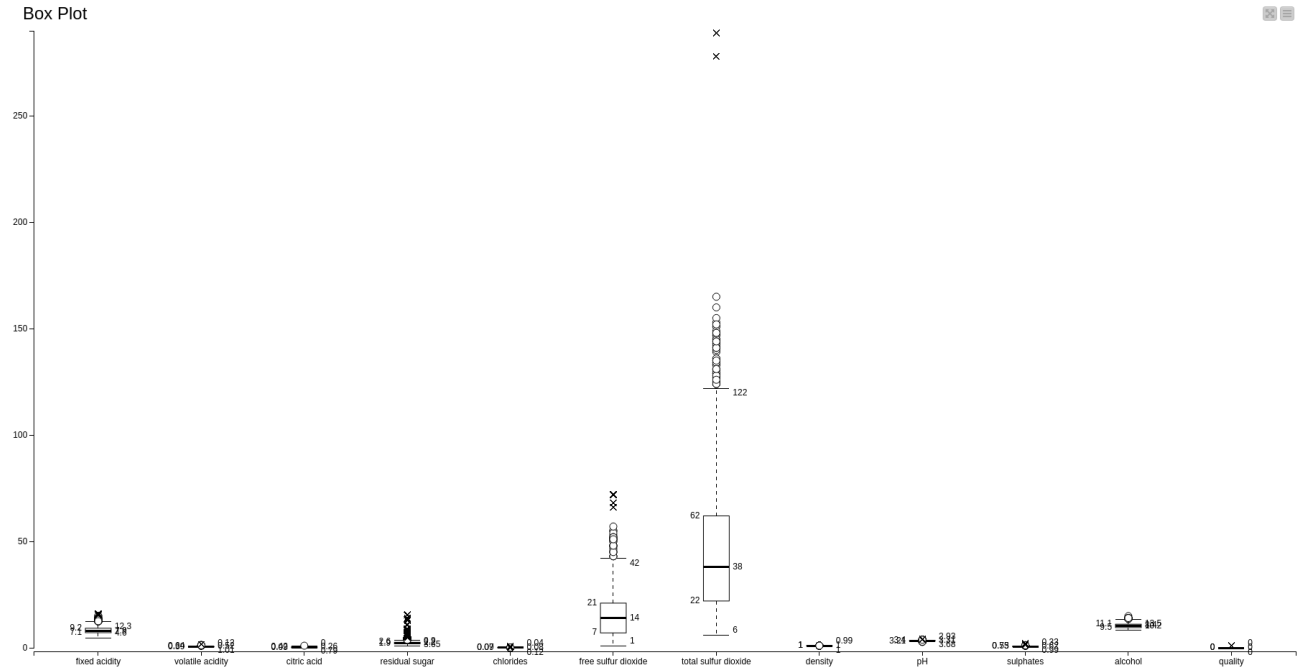

Figura 4.2: Box Plot relativo ao dataset sem qualquer tratamento

Neste tratamento de *outliers*, tendo sido aplicado a todos os atributos exceto o *quality* (uma vez que é o *target*), optou-se por remover a respetiva linha que continha o *outlier*. De referir que se testaram outras opções, como por exemplo *replace outlier values*, porém revelaram menos *accuracy* nos nodos finais, que serão explicados em detalhe mais à frente.

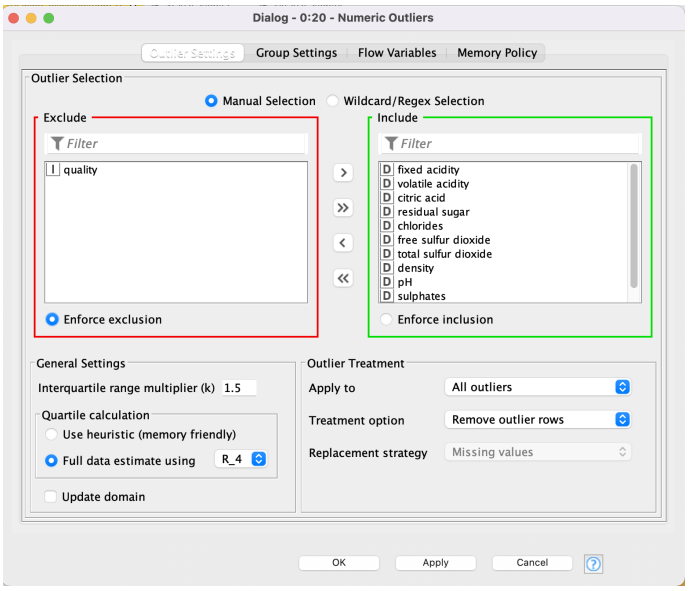

Figura 4.3: Configurações do nodo *Numeric Outliers*

Uma vez que se está a trabalhar com um dataset com o objetivo de classificação, e se pretende recorrer a Learner's, é estritamente necessário a presença de uma coluna com valores nominais. Para tal, recorreu-se ao nodo **Rule Engine** para efetuar a conversão para este formato. Deste modo, definiu-se o valor 0 como *Bad* e o valor 1 como *Good*.

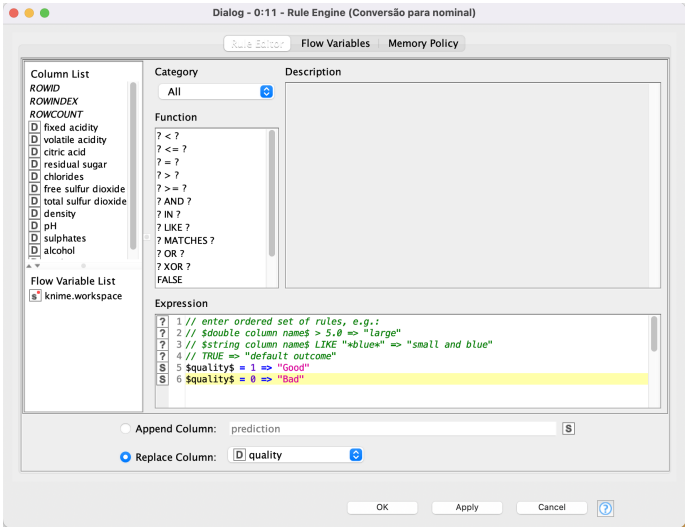

Figura 4.4: Configurações do nodo *Rule Engine*

Posteriormente, como forma de adquirir conhecimento, recorreu-se ao nodo **Partioning**, para efetuar a divisão dos dados em teste (para que o modelo possa colocar à prova o seu conhecimento) e treino (para que o modelo adquira conhecimento). Depois de alguns testes com porporções semelhantes, chegou-se a conclusão que $80/20$ seria a melhor, então optou-se por dividir em 80% para treino e 20% para teste.

É de notar que todo o particionamento foi aplicado recorrendo a uma random seed de 2022, com o objetivo de poder replicar os resultados a qualquer momento.

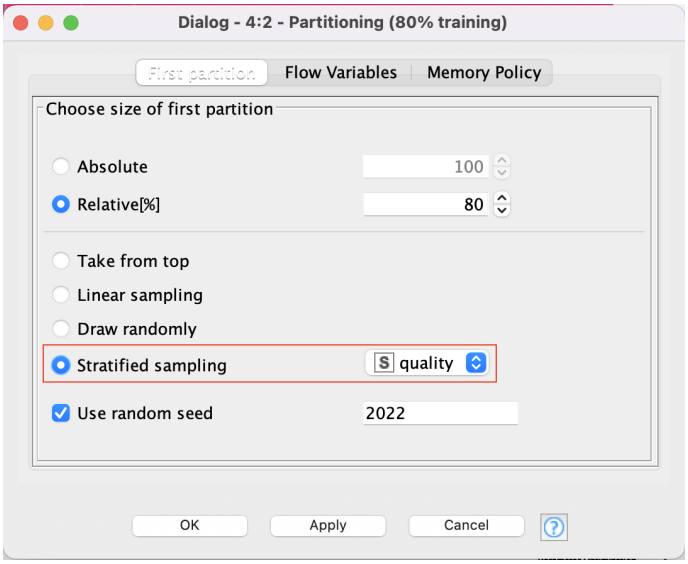

Figura 4.5: Configurações do nodo *Partioning*

Verificou-se também, pela Figura 4.6, que existe uma maior quantidade de entradas do tipo *Bad* do que do tipo *Good*, causando um desbalanceamento. Uma vez que se prentende o partitioning em relação ao target (quality), especificando-se nesse mesmo atributo e escolhendo a mesma quantidade de valores *Good* e *Bad*, recorreu-se à opção **Stratified Sampling**, como se pode verificar na Figura 4.5. Apesar da utilização desta opção, como existe um número bastante mais elevado de entradas do tipo *bad* do que do tipo *good*, fazendo com que a opção *Stratified Sampling* não tenha o efeito pretendido.

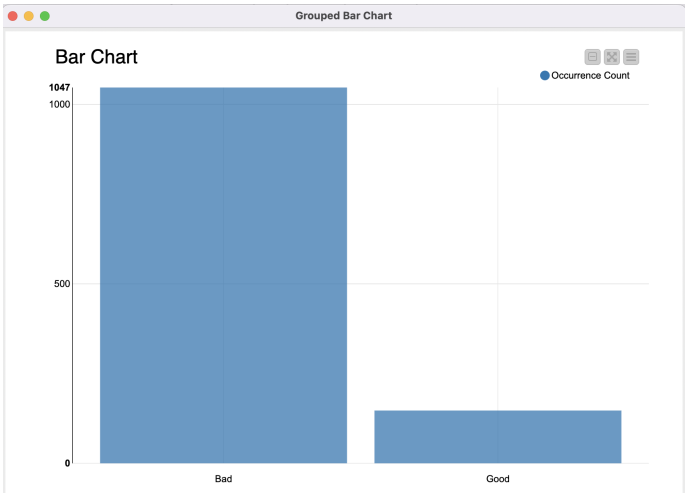

Figura 4.6: Distribuição dos dados obtidos através do *Bar Chart*

Para contornar este problema, tentou-se usar o nodo **Equal Size Sampling**, no entanto, apesar da previsão das entradas *Good* ter melhorado, a previsão das entradas *Bad* piorou significativamente, como se pode nas Figuras abaixo, pelo que a accuracy do modelo piorou bastante. É de referir que este nodo foi colocado apenas para os dados de treino, com a finalidade de não interferir nos dados de teste do modelo.

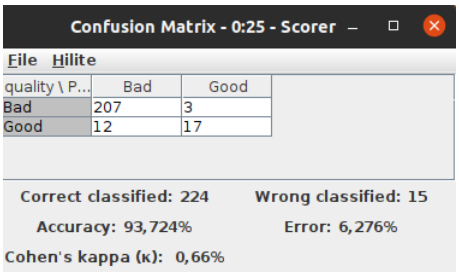

Figura 4.7: XGBoost sem Equal Size Sampling

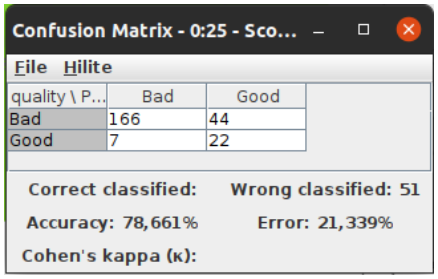

Figura 4.8: XGBoost com Equal Size Sampling

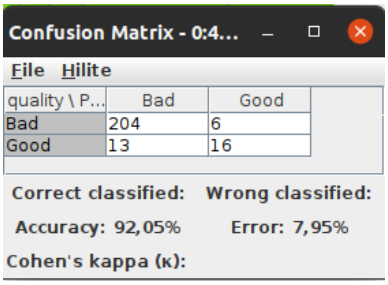

Figura 4.9: Gradient Boosted sem Equal Size Sampling

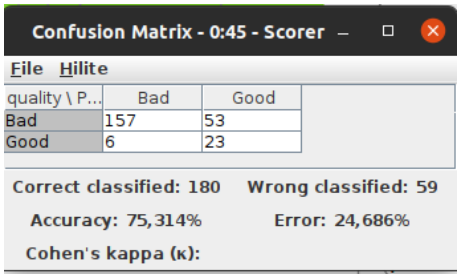

Figura 4.10: Gradient Boosted com Equal Size Sampling

É importante referir que através de uma análise das Matrizes de Confusão se pode verificar que, apesar da diminuição da *accuracy* com recurso ao Equal Size Sampling, se verificou uma melhoria em termos de previsão de resultados good.

Deste modo, tem-se um modelo que é bom para prever casos bad, mas era mau para previsão de good, tendo uma *accuracy* global maior. Com recurso a *Equal Size Sampling*, tem-se um melhor modelo para prever os good, mesmo a *accuracy* global sendo significativamente mais baixa. O grupo optou por uma *accuracy* maior, uma vez que se interpreta que o objetivo desta previsão não é prever apenas os casos *good*.

4.3 Modelos

Antes de construir o modelo de *machine learning*, foi necessário efetuar uma pesquisa inicial acerca de quais as técnicas mais adequadas para o presente problema.

Deste modo, os modelos foram treinados usando os seguintes nodos:

- **Decision Tree Learner**
- **Random Forest Learner**
- **XGBoost Tree Ensemble Learner**
- **Gradient Boosted Trees Learner**

enquanto as previsões foram criadas com os nodos:

- **Decision Tree Predictor**
- **Random Forest Predictor**
- **XGBoost Predictor**
- **Gradient Boosted Trees Predictor**

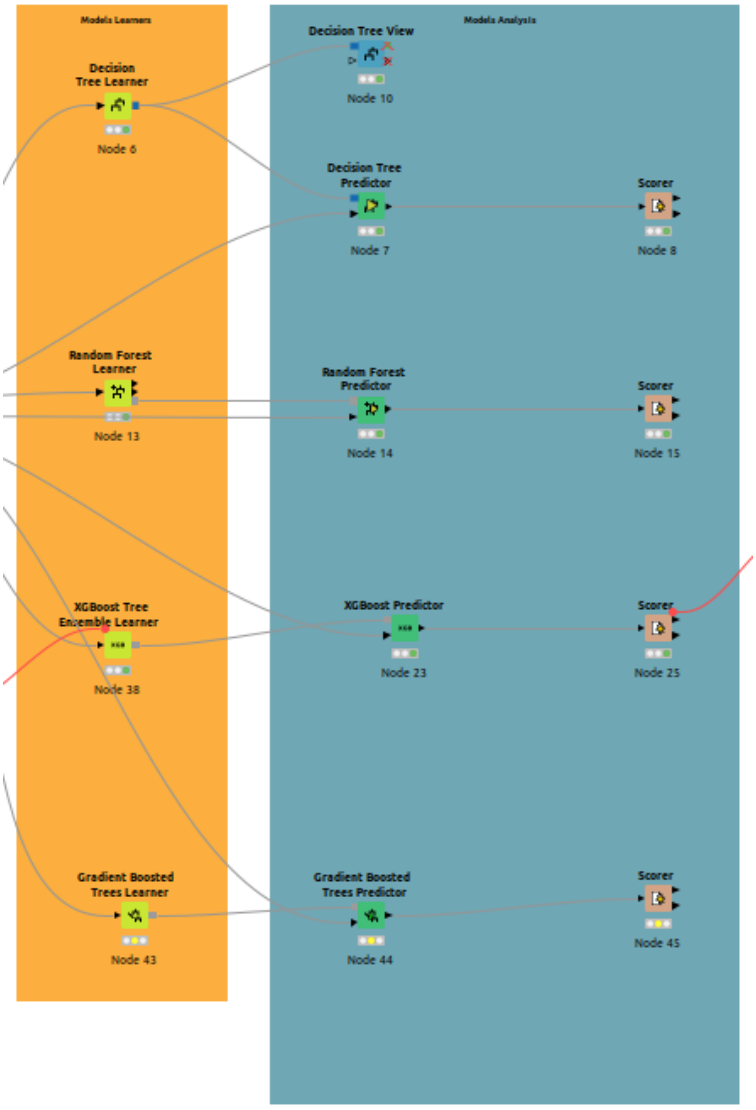

Figura 4.11: Nodos utilizados relativamente a Learning, Predictioning e análise de modelos

4.3.1 Tunning dos Hyperparâmetros

De modo a maximizar e ajustar os parâmetros do algoritmo XGBoost, recorreu-se aos nodos **Parameter Optimization Loop Start** e **Parameter Optimization Loop End**, tendo sido ajustados alguns parâmetros que conferiram melhorias significativas no modelo em questão. Apesar de este tema ser desenvolvido com mais detalhe na Secção 5.4.2, pelo que se explica em seguida os parâmetros que foram alterados para aumentar a performance. Após se ter análisado os vários parametros, apenas se recorreu a dois deles, sendo que eram os que conferiam um maior *Objective Value*:

- ETA
- Maximum Depth

4.3.2 Decision Tree

O *Decision Tree* é um tipo de *supervised machine learning* usado para categorizar (no presente caso) ou prever o valor de uma variável, aprendendo regras de decisão simples inferidas a partir de dados anteriores (dados de treinamento).

Uma *Decision Tree* é um grafo hierarquizado (árvore) em que:

- Cada **ramo** representa a **seleção entre um conjunto de alternativas**
- Cada **folha** representa uma **decisão**

Dada uma uma árvore de decisão treinada, o **processo de decisão** desenvolve-se do seguinte modo:

1. Começar do nodo correspondente ao atributo "raiz";
2. Identificar o valor do atributo;
3. Seguir pelo ramo correspondente ao valor identificado;
4. Alcançar o nodo relativo ao ramo percorrido;
5. Voltar a 2. até que o nodo seja uma folha;
6. O nodo alcançado indica a decisão para o problema.

4.3.3 Gradient Boosted Trees Predictor

O algoritmo *Gradient Boosted Trees*, também conhecido como GBT, deriva do algoritmo *XGBoost*, que também foi utilizado no outro dataset. Desta forma, aprende Gradient Boosted Trees com o objetivo de classificação, usando árvores de regressão muito rasas e uma forma especial de reforço para construir um conjunto de árvores.

Ao recorrer ao *Gradient Boosted Trees*, este método constrói uma árvore de cada vez, onde cada árvore ajuda a corrigir os erros que possam ter sido cometidos pela árvore que foi treinada anteriormente.

Para além das razões referidas anteriormente, o algoritmo é também muito eficaz em casos de detecção de anomalias e informação bastante desbalanceada, sendo frequentemente utilizado em situações de testes de ADN,

transações de cartões de crédito ou cibersegurança.

4.4 Conclusão e análise dos resultados obtidos

Relativamente aos resultados, a equipa efetuou alguns testes, de forma a melhorar a *accuracy* dos modelos referidos na secção anterior, sempre priveligiando o senso que cada alteração possa fazer e a alteração da *accuracy*. Em seguida, serão descritos alguns dos testes e alterações que se efectuou para atingir a melhor solução possível:

- **Análise de linhas duplicadas** - Em análise do dataset proposto, o grupo procurou verificar e tratar a presença de registos duplicados através do nodo **Duplicate Row Filter**, procurando verificar a influência que poderia causar na *accuracy* dos modelos, mesmo sabendo que poderia corresponder a registos coincidentemente repetidos e viciar a aprendizagem do modelo. Ao analisar os resultados obtidos, verificou-se que este nodo causou um impacto negativo, pelo que se decidiu aceitar que seriam atributos coincidentemente repetidos e que, em caso de remoção prejudicariam a aprendizagem do modelo.

- **Análise com tratamento VS. sem tratamento** - De modo a estabelecer uma comparação e verificar as melhorias associadas ao tratamento efetuado relativamente ao dataset, apresenta-se a seguinte tabela:

	Sem Tratamento	Com Tratamento
Decision tree	87.8	88.7
Random Forest	90.6	92.5
XGBoost	86.8	94.1
Boosted	90.9	92.0

- **Oversampling/Undersampling** - Após a análise inicial do dataset, decidiu-se verificar a quantidade vinhos com good e bad. Para tal, recorreu-se ao seguinte Bar Chart:

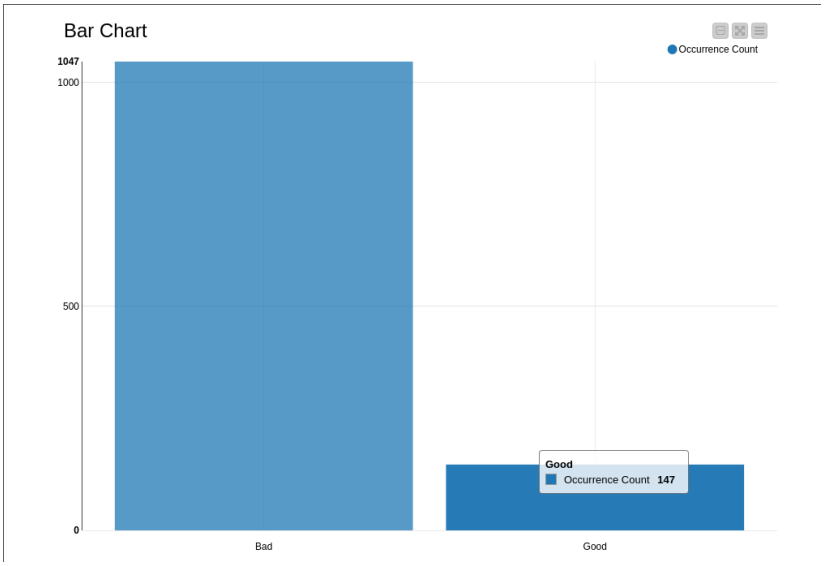

Figura 4.12: Box Plot para análise da quantidade de valores good e bad

Com isto, pensou-se em efetuar um ajuste relativamente ao número de ocorrências good e bad, tendo sido analizadas as técnicas de *oversampling* e *undersampling*. Inicialmente, esta decisão causou alguma dúvida, visto que devido à grande discrepância que se apresentava, se poderia ter um resultado fictício,

uma vez que os dados que seriam usados neste caso de *overfitting* seriam inventados apenas para preencher a diferença que se verifica. No caso de um *underfitting*, o facto de eliminar ocorrências poderá levar a um resultado adulterado, sendo que irá remover alguns dos resultados, que poderia conduzir a uma melhor aprendizagem por parte do modelo. Para estes casos, usou-se os nodos SMOTE e *Equalsize Sampling*, respetivamente.

Após testes, tanto através de SMOTE como de *Equalsize Sampling*, verificou-se um decréscimo na *accuracy*, pelo que se decidiu não recorrer a este tipo de análise, apesar de se ter tido em conta esta possibilidade.

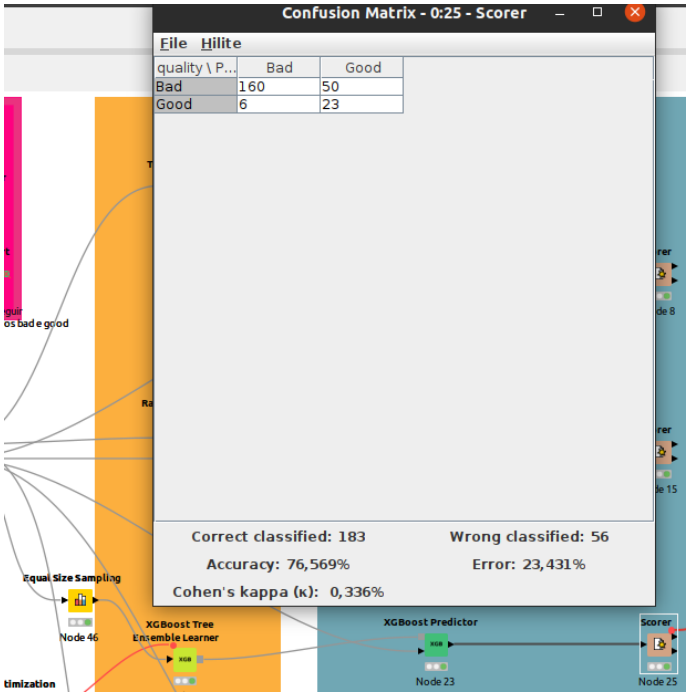

Figura 4.13: Resultado do nodo Equal Size Sampling aplicado aos dados de treino

- **Diferentes valores das proporções no Partitioning** - De modo a verificar qual seria a melhor proporção para a partição, testou-se com valores entre 70/30 e 85/15 e chegou-se à conclusão que a melhor proporção seria 80% para treino e 20% para teste. Todos os outros valores testados faziam com que a *accuracy* diminuísse.

- É de notar que, como os resultados eram melhores no caso do *XGBoost* do que nos outros modelos, optou-se por fazer o tratamento de dados considerando a maximização deste algoritmo. Isto é de importante destaque, uma vez que as técnicas de tratamento dos dados aplicadas dependem do modelo utilizado.

- *Feature Selection* Manualmente - Foi feito uma *Feature Selection* "manualmente", tirando um atributo um a um com o auxílio do nodo *Column Filter*. No entanto não se verificou nenhuma melhoria nos resultados, então não se optou por retirar nenhum atributo.

5 Grupo Flights

5.1 Domínio e objectivos

O dataset analisado neste capítulo contém informação sobre opções de reserva de voos do website "Ease My Trip". "Ease My Trip" é uma empresa indiana de viagens online. O atributo definido como target foi o preço (do bilhete de avião).

Os objectivos principais da análise deste estudo são:

- construir um modelo capaz de fazer uma boa previsão do preço do bilhete de avião com base num conjunto de atributos.
- extrair conhecimento relevante dos dados que permitam à empresa "Ease My Trip" optimizar o seu sistema e tomar boas decisões estratégicas, e assim aumentar o sucesso da empresa.
- extrair informações valiosas que serão de enorme valor para os passageiros.

De modo a atingir estes objectivos utilizar-se-á técnicas de preparação, tratamento, exploração e visualização de dados. Para prever o preço do bilhete de avião serão utilizados modelos de machine learning de regressão, pois o *target* (preço) é uma variável contínua.

Todo o processo de análise do *dataset* foi realizado na plataforma KNIME.

5.2 Exploração e Tratamento dos dados

5.2.1 Descrição dos atributos do dataset

Antes de fazer qualquer análise ao *dataset* leu-se a descrição dos atributos que foi fornecida, juntamente com os dados de treino. Entender a definição de cada um dos atributos é fundamental para a perceção do problema em questão, os valores que estão associados a cada atributo, e perceber as estratégias a aplicar na preparação do *dataset*. A descrição de cada atributo do dataset pode ser encontrada no website Kaggle.

5.2.2 Construção do dataset

Primeiramente foi necessário construir o dataset no KNIME. Os dados do problema estavam inicialmente separados em dois conjuntos de dados: um dataset com os dados relativos aos bilhetes de class económica e outro com os bilhetes de class business. Deste modo cada conjunto de dados (estavam escritos num ficheiro com formato CSV) foi lido usando o nodo **CSV Reader**.

Como os dados estavam organizados separadamente em *business* e *economy*, cada um dos datasets não tinha o atributo *class*. Assim foi usado o nodo **Rule Engine** para adicionar a cada um dos *datasets* uma nova coluna

chamada *class* e foram preenchidos todos os campos com o valor *business* ou *economy* dependendo do caso.

Após a adição da coluna *class* juntou-se os registos dos dois conjuntos de dados em apenas um *dataset* utilizando para isso o nodo **Concatenate**.

Como os nomes dos atributos não eram muito explicativos, para facilitar o reconhecimento dos dados no tratamento e preparação futura do *dataset*, renomeou-se as colunas com o nodo **Column Rename**.

O processo descrito anteriormente está apresentado na figura 5.1.

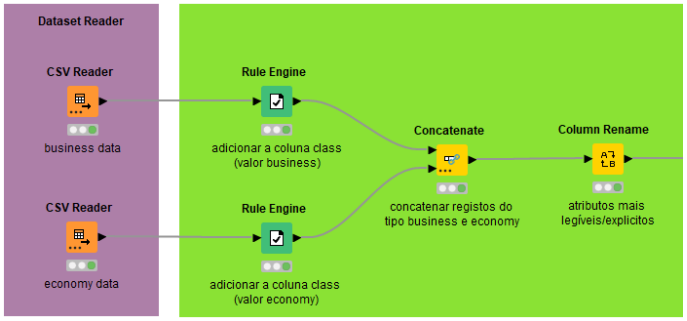

Figura 5.1: Construção do dataset a partir dos dados business e economy

Após a preparação do dataset foi obtido como output o resultado apresentado na figura 5.2. Da visualização geral dos dados do *dataset* foram feitas algumas observações:

- Com que tipo é que o KNIME reconhecia os valores dos vários atributos.
- Qual o formato de representação de cada atributo. Por exemplo, o atributo *duration_flight* está representado em horas e minutos embora este possa poder ser representado em apenas minutos.
- Quais os diferentes valores para cada atributo.

Row ID	data_travel	airline	character_code	numerical_code	departure_time	source_city	duration_flight	stops	arrival_time	destination_city	price	class
Row0	11-02-2022	Air India	AI	868	18:00	Delhi	02h 00m	non-stop	20:00	Mumbai	25,612	business
Row1	11-02-2022	Air India	AI	624	19:00	Delhi	02h 15m	non-stop	21:15	Mumbai	25,612	business
Row2	11-02-2022	Air India	AI	531	20:00	Delhi	24h 45m	1-stop	20:45	Mumbai	42,220	business
Row3	11-02-2022	Air India	AI	839	21:25	Delhi	26h 30m	1-stop	23:55	Mumbai	44,450	business
Row4	11-02-2022	Air India	AI	544	17:15	Delhi	06h 40m	1-stop	23:55	Mumbai	46,690	business
Row5	11-02-2022	Vistara	UK	985	19:50	Delhi	02h 10m	non-stop	22:00	Mumbai	50,264	business
Row6	11-02-2022	Air India	AI	479	21:15	Delhi	17h 45m	1-stop	15:00	Mumbai	50,669	business
Row7	11-02-2022	Air India	AI	473	18:40	Delhi	22h 45m	1-stop	17:25	Mumbai	51,059	business
Row8	11-02-2022	Vistara	UK	871	20:35	Delhi	17h 55m	1-stop	14:30	Mumbai	51,731	business
Row9	11-02-2022	Vistara	UK	977	19:00	Delhi	02h 15m	non-stop	21:15	Mumbai	53,288	business
Row10	11-02-2022	Air India	AI	504	21:35	Delhi	11h 00m	1-stop	08:35	Mumbai	56,081	business

Figura 5.2: Visualização dos primeiros 10 registos do dataset

Da figura 5.2 é possivel ver os vários atributos que foram explorados:

```
data_travel          departure_time       arrival_time
airline              source_city          destination_city
character_code       duration_flight      price
numerical_code       stops                class
```

5.2.3 Redução do tamanho do *dataset*

Explorando a tabela de output do nodo **Concatenate** da figura 5.2 verificou-se que o *dataset* tinha 300261 registos. Este *dataset* é de dimensão relativamente grande, e nestes casos isso pode implicar um grande impacto no tempo de treino do modelo de *machine learning*. De modo a aumentar a performance do processo de treino efectuamos a redução do tamanho do *dataset*. Primeiro aplicou-se o nodo **Row Sampling** para selecionar

apenas 90% dos registos. Escolheu-se este valor pois era a maior percentagem de registos que conseguimos remover sem que houvesse uma redução no valor da métrica *R-Squared*. Escolheu-se a opção *Stratified Sampling* pelo atributo price para que continuar a ter a mesma proporção dos diferentes valores do atributo e usamos também uma *random seed* de valor 2022 de modo a manter a reprodutibilidade das experiências.

O segundo nodo aplicado para reduzir o tamanho do dataset foi o **Duplicate Row Filter** para remover informação repetida. Neste caso este passo não se teve grande impacto já que apenas 2 linhas foram removidas.

O processo descrito está representado na figura 5.3.

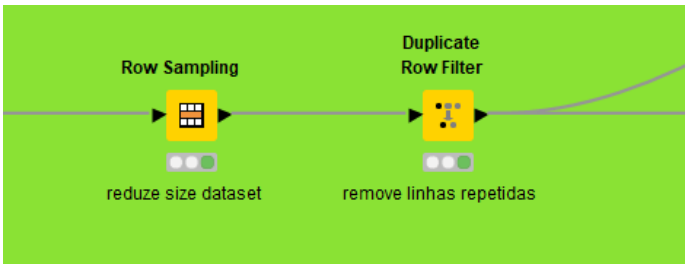

Figura 5.3: Redução do tamanho do dataset

5.2.4 Conversão de tipos

O tipo do atributo stops era string tal como foi visualizado na figura 5.2. Contudo, um atributo que represente o número de paragens do avião, pode também ser representado como valor númerico ordinal (0, 1, 2, ...). O tipo do atributo stops foi convertido de string para númerico ordinal pois deste modo o modelo consegue reconhecer a relação de ordem entre os diferentes valores caso em que não acontecia com o tipo string. Para realizar a conversão o primeiro passo passou por identificar os diferentes valores possíveis para o atributo stops, ou seja, os valores únicos. Uma das formas de conseguir obter os valores únicos dos atributos nominais foi a partir do nodo **Data Explorer**. O resultado obtido com este nodo apresenta-se de seguida:

```
Atributo : stops
Valores únicos (40) :

1-stop,          1-stop Via IXU,     1-stop Via Kolhapur,    1-stop Via GAU,
non-stop,        [...],              1-stop Via IXR,         1-stop Via HYD.
2+-stop,         1-stop Via IDR,     1-stop Via GAY,
```

Dada a relação entre os diversos valores possíveis para o atributo stops, foram atribuidos valores inteiros que respeitassem a ordem entre eles. A ordem estabelecida foi a seguinte:

```
non-stop  →  [1-stop, 1-stop Via IXU, ...]  →  2+-stop
    0     →                 1               →     2
```

A conversão foi realizada utilizando o nodo **Rule Engine**.

Outro dos atributos em que foi necessário alterar o tipo foi o price. Este atributo é o *target* de um problema de regressão e portanto deverá ser convertido para tipo númerico. O nodo **String to Number** foi utilizado para este efeito.

Outra das conversões de tipo necessário foi a do atributo *duration_flight* de string para valor número. A razão pela qual realizamos esta conversão foi semelhante ao caso do atributo *stops*. Neste caso foi necessário

fazer uma primeira conversão de String para *Duration* e depois de *Duration* para *Number*. Foi utilizado o **String Manipulation**, juntamente com conhecimento de expressões regulares, para converter os valores para um formato reconhecivel pelo nodo **String to Duration** e este fez a conversão explícita no nome do nodo. Neste passo 4 registos não puderam ser convertidos. Como estes registos não apresentavam características únicas do dataset decidimos remover essas linhas. Num útlimo passo utilizamos o nodo **Duration to Number** para obter um valor inteiro representativo dos minutos. Este processo está explicito na figura 5.4.

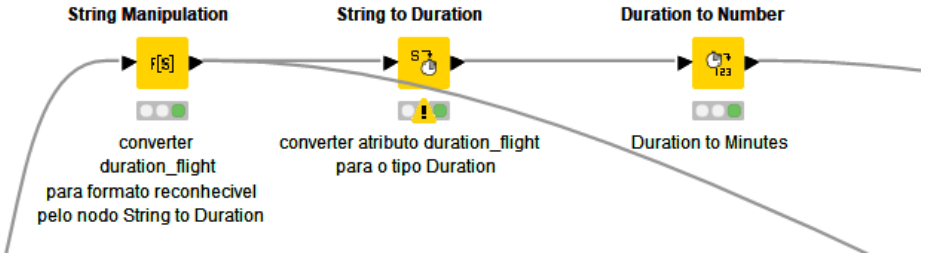

Figura 5.4: Conversão do atributo duration_flight de tipo string para numérico

5.2.5 Extracção de atributos a partir da data

Um dos atributos pertencentes ao dataset era o *data_travel* que corresponde à data da viagem. Os valores do *data_travel* não apresentam muita informação para o modelo da forma como estão representados (como uma *string*). Deste modo para obter as várias informações possíveis a partir da data começou-se por utilizar o nodo **String Manipulation** para efectuar a conversão do formato inicial da data para outro formato capaz de ser reconhecido pelo nodo **String to DateTime** que faz a conversão de tipos. Por fim, o nodo **Extract DateTime Fields** tem como uso obter os seguintes atributos:

- Ano
- Mês
- Dia
- Semana do Ano
- Dia do mês
- Dia da Semana

Estas informações foram adicionadas ao dataset como atributos.

O processo anterior está apresentado na figura 5.5.

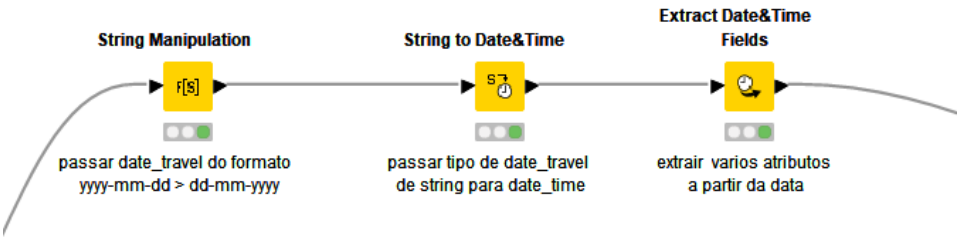

Figura 5.5: Processo de extracção de atributos a partir da data

5.3 Visualização de dados

5.3.1 Distribuição do preço dos bilhetes

Com o objetivo de melhorar os resultados produzidos pelo modelo e também conhecer melhor o dataset selecionado, foi necessário criar uma secção no modelo para a vizualização de dados. De seguida, analisa-se detalhadamente os gráficos e tabelas obtidas, bem como o processo para a sua obtenção.

5.3.2 Preço dos bilhetes

De forma a analizar a ordem de grandeza do preço dos bilhetes, foi realizado o *binning* dos dados através do nodo **Bar Chart** de acordo com o preço de cada linha, o que resultou em 10 bins que através do nodo **Bar Chart** originaram o gráfico de barras evidenciado de seguida. É de salientar que a unidade monetária deste dataset é a Rupia indiana.

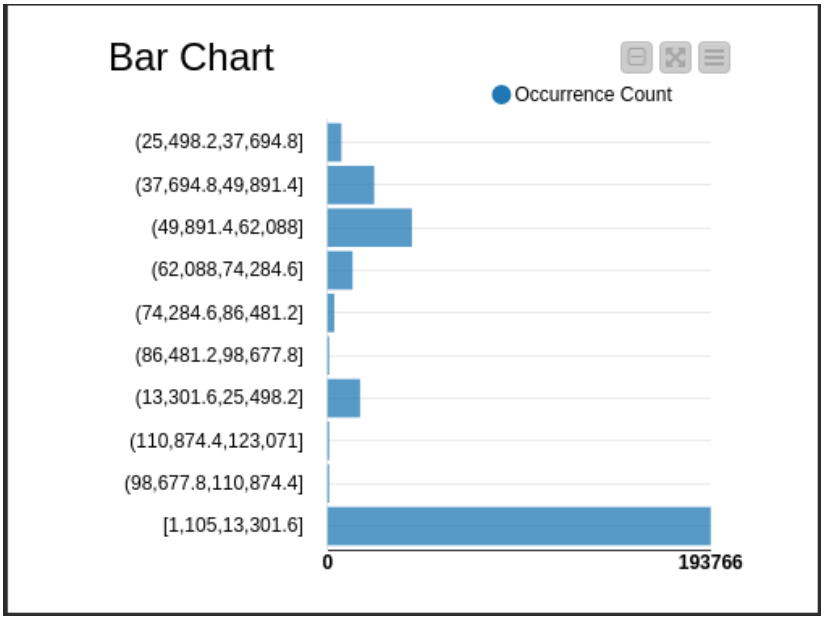

Figura 5.6: Distribuição do preço dos bilhetes

5.3.3 Minutos

De modo a tratar mais eficazmente o modelo foi necessário verificar a duração das viagens referenciadas no *dataset*, com o intuito de perceber a ordem de grandeza a ser tratada. Com este fim, foi utilizado o nodo **Box Plot** com o respetivo output de seguida. Apesar de existirem outliers representados no gráfico resultante neste caso não se deverá aplicar nenhum tratamento, dado que como no *dataset* existem voos com diferentes números de escalas seria possível, com por exemplo a remoção de *outliers*, remover também as viagens com um número de escalas elevadas o que tornaria o modelo despreparado para prever o preço deste tipo de voo. Para além disso os restantes atributos foram analisados e em nenhum faria sentido aplicar a remoção de *outliers*.

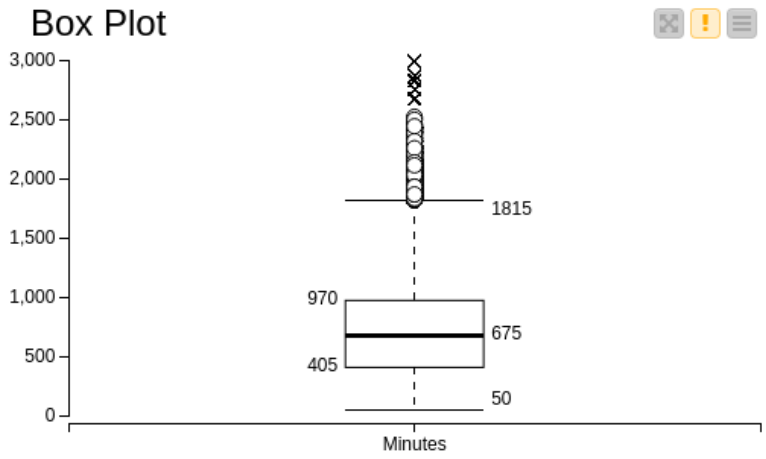

Figura 5.7: Distribuição da feature minutos

5.3.4 Bilhetes vendidos por airline

Para analisar o número de bilhetes vendidos por cada *airline* foi necessário recorrer a um **Interactive Histogram** e a um **Color Manager** de forma a representar também a distribuição da classe do bilhete por companhia aérea. Ao analisar os dados obtidos, é possível verificar que apenas 2 companhias possuem a classe *business*, mas também que existem algumas companhias que realizam um número reduzido de vendas o que poderá ser uma informação útil para a administração da rede de vendas criadora do dataset.

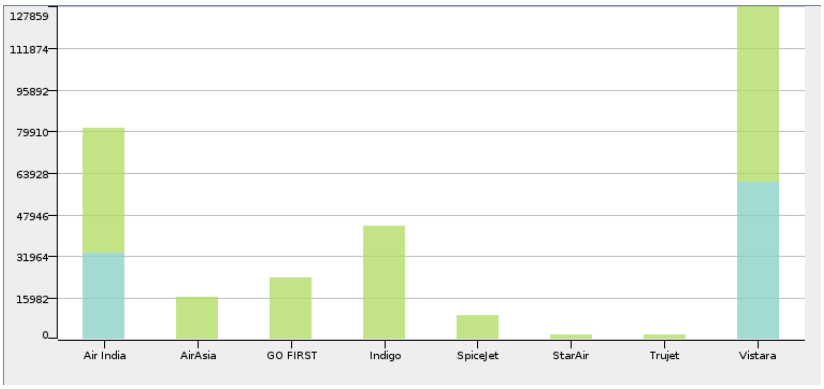

Figura 5.8: Número de Tickets por airline. Verde -> Economy | Azul -> Business

5.3.5 Preço por airline

Uma das melhores features para diferenciar a o preço dos bilhetes é a airline, deste modo foi analisada a distribuição de preço em cada uma das companhias aéreas presentes no dataset, com o auxílio do nodo **Row Sampling** para selecionar de forma aleatória 10 mil linhas, foi possível utilizar um **Scatter Plot** e assim analisar a influência das várias companhias em estudo através do gráfico apresentado de seguida. Com este gráfico, é possível perceber que as empresas "Air India"e "Vistara"mais variadade no preço dos bilhetes, variedade esta que pode ser explicada por apenas estas empresas possuirem bilhetes com da classe business.

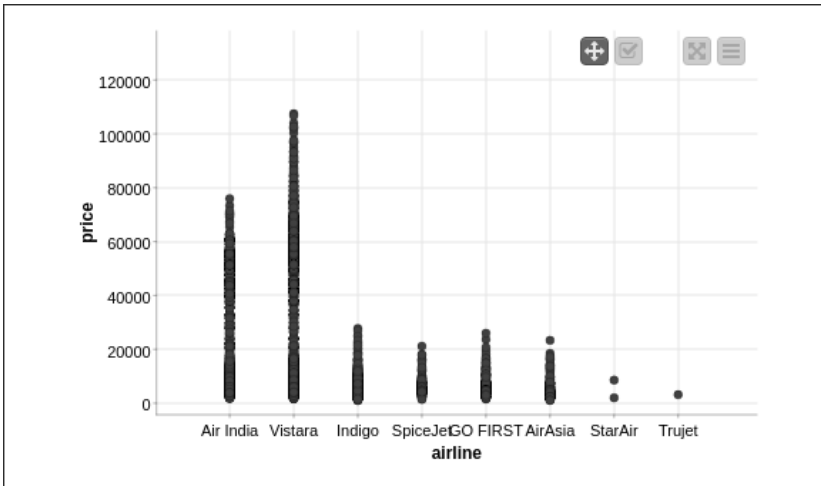

Figura 5.9: Distribuição do preço por airline

Ainda na sequência do estudo da influência das diferentes airlines no preço do bilhete, e para melhor perceber a sua relevância neste contexto, mostrou-se necessária analisar também a média dos bilhetes vendidos por companhia aérea, os resultados obtidos através do nodo **Bar Chart**, com o auxílio de um **GroupBy** apresentam-se de seguida.

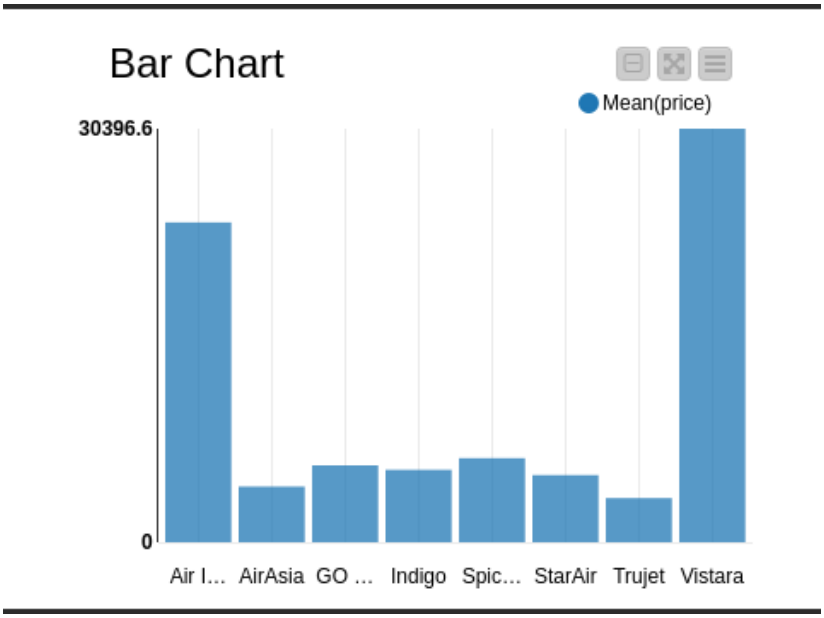

Figura 5.10: Média do preço do bilhete por airline

5.3.6 Preço do bilhete ao fim de semana e à semana

Intuitivamente, o preço dos bilhetes ao fim de semana seria maior e por isso foi aplicado tratamento de dados nesse sentido. Com o intuito de conseguir separar o dataset em entradas ao fim de semana e à semana utilizou-se o nodo **Rule Engine** para acrescentar uma coluna com esta informação. Desta forma, foi possível criar o gráfico apresentado de seguida, que apresenta a média do preço dos bilhetes em ambas as situações recorrendo ao nodo **Bar Chart**. No entanto, não se verificou um desvio relevante nos casos em estudo para a coluna previamente criada ser aplicada no modelo.

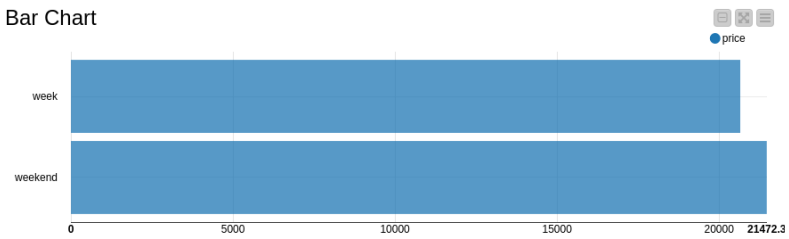

Figura 5.11: Média preço do bilhete à semana e ao fim de semana

5.3.7 Linear Correlation

Para verificar a importância das diversas features para a interpolação do target, foi utilizado o nodo **Linear Correlation**, desta forma foi possível verificar ao longo do desenvolvimento do modelo as features que necessitavam de tratamento para obterem uma maior correlação, ou até mesmo de modo a retirar features irrelevantes no caso em estudo, como por exemplo a feature *character code* que por ter uma correlação de 1 com a feature *airline* se demonstrou desnecessária, visto que não acrescenta nenhuma informação. Os valores de correlação finais encontram-se apresentados em seguida.

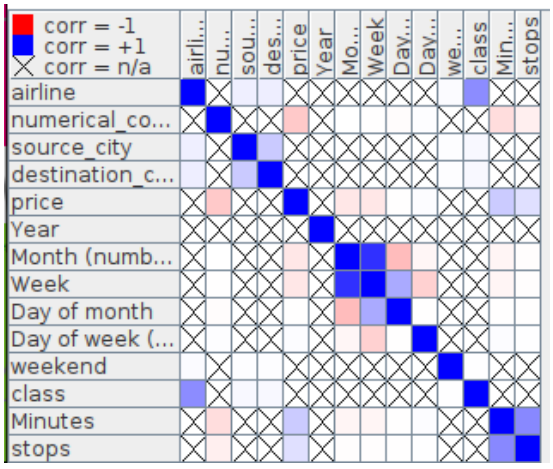

Figura 5.12: Correlação linear das features do modelo

5.4 Modelos

5.4.1 Técnicas de aprendizagem

Para construir o modelo de machine learning para fazer as previsões foi necessário começar por identificar quais as técnicas mais adequadas para o problema em questão. Neste sentido fez-se uma pesquisa sobre quais as técnicas de regressão mais populares para problemas de previsão do preço do voo e várias fontes referenciaram o XGBoost como o algoritmo mais comum.

Em problemas de regressão o algoritmo Random Forest também costuma apresentar bons resultados numa grande variedade de problemas, portanto também se usou inicialmente esta técnica.

Como os resultados das métricas eram melhores no caso do XGBoost do que no Random Forest decidiu-se fazer o tratamento dos dados considerando este algoritmo. Esta parte é importante pois às técnicas de tratamento

dos dados aplicadas dependem da técnica utilizada. Um exemplo disto foi o facto de não aplicar a técnica de normalização de valores pois esta técnica de aprendizagem não necessita desse tratamento.

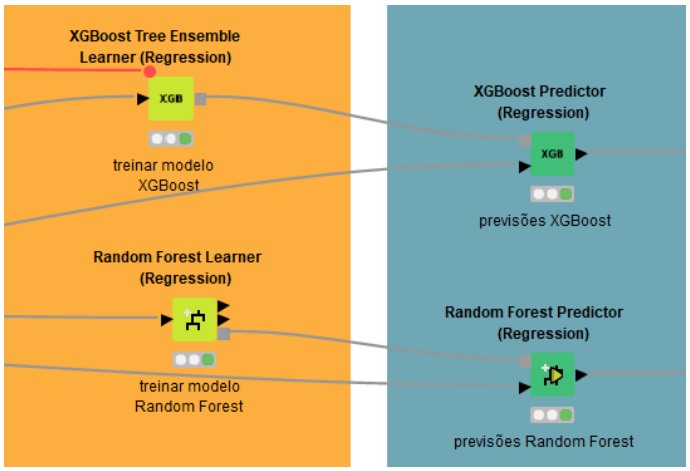

Figura 5.13: Treino dos modelos e previsões dos casos de teste

Os modelos foram treinados usando os nodos:

- **XGBoost Tree Ensemble Learner(Regression)**
- **Random Forest Learner(Regression)**

enquanto as previsões foram criadas com os nodos:

- **XGBoost Predictor(Regression)**
- **Random Forest Predictor(Regression)**

A figura 5.13 apresenta o processo descrito. Para o treino do modelo também foi usada uma random seed de valor 2022 (de notar que foi utilizada sempre a mesma ao longo de ambos os datasets), de modo a manter a reprodutibilidade das experiências.

5.4.2 Tunning dos Hiperparâmetros

O algoritmo XGBoost tem associado múltiplos parâmetros de treino. De modo a obter os melhores resultados devemos escolher os valores dos parâmetros mais indicados. Como o número de parâmetros desta técnica é elevado começou-se por escolher aqueles que tinham mais influência. Realizamos uma pesquisa dos parâmetros mais importantes e após algumas experiências de treino do modelo escolheu-se os seguintes, que foram os que tiveram maior impacto positivo nas métricas de qualidade:

- Eta (Learning Rate)
- Maximum depth

Para optimizar estes parâmetros usou-se um nodo que faz uso da técnica de grid-search. O grid search recorre a diferentes combinações de todos os hyperparâmetros específicados e os seus valores e calcula a performance para cada um dos casos. O nodo em questão é o **Parameter Optimization Loop Start**. A grelha de parâmetros pode ser observada na figura 5.14 e também estes valores foram aconselhados pela pesquisa referida anteriormente.

No nodo **Parameter Optimization Loop End** foi especificada a métrica que temos objectivo de maximizar: *R-Squared*. Os resultados obtidos apresentam-se a seguir:

Parameters

Parameter	Start value	Stop value	Step size	Integer?
eta	0,2	0,8	0.2	☐
max_depth	7	11	1.0	☑

Figura 5.14: Grelha de parâmetros usada no treino do modelo XGBoost

	Eta	Max depth	Objective Value		Eta	Max depth	Objective Value
1	0.2	7	0.9828	11	0.6	9	0.9895
2	0.4	7	0.9861	12	0.8	9	0.9891
3	0.6	7	0.9867	13	0.2	10	0.9890
4	0.8	7	0.9869	14	0.4	10	0.9904
5	0.2	8	0.9858	15	0.6	10	0.9896
6	0.4	8	0.9879	16	0.8	10	0.9891
7	0.6	8	0.9885	17	0.2	11	0.9899
8	0.8	8	0.9883	18	0.4	11	0.9907
9	0.2	9	0.9876	19	0.6	11	0.9898
10	0.4	9	0.9891	20	0.8	11	0.9886

Os melhores valores para os parâmetros foram Eta=0.4 e Max depth=11 o que representa um valor de *R-Squared*= 0.9907. Comparando com outras combinações podemos observar aumentos na casa dos 0.01, o que indica que esta técnica tem grande um grande impacto no modelo. Apesar de 0.01 não parecer muito à primeira vista, pequenos aumentos tem muito maior significado quando o valor de *R-Squared* está muito próximo de 1.

5.4.3 Partitioning

Em relação ao particonamento dos dados, na comunidade científica é muito comum utilizar uma porporção de 70/30 para o particionamento do dataset, no entanto este valor deve-se adequar ao modelo em questão, como tal, foram efetuados alguns testes com particionamentos semelhantes a este, sendo que as melhores métricas foram atingidas com a porporção 80/20 e portanto esta foi a utilizada no modelo.

No particionamento foi aplicada também a metodologia stratified sampling tendo em conta a coluna class, dado que se procura igualar o número de ocorrências de dados do tipo business e economy, de modo a que o modelo não treine de forma desbalanceada. Apesar da utilização desta metodologia, como o dataset usado possui um número muito maior de entradas do tipo economy do que do tipo business foi verificada uma distribuição assimétrica deste parâmetro como se pode verificar na figura 5.15.

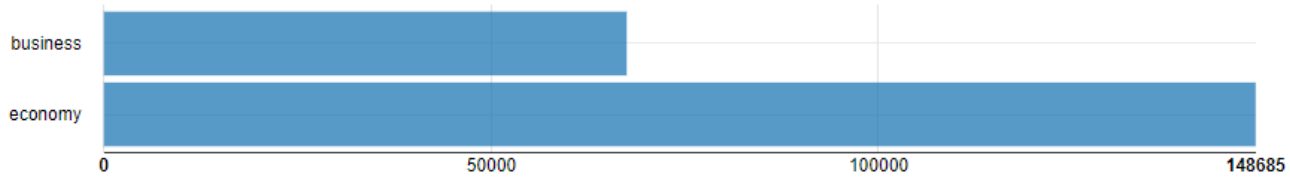

Figura 5.15: Número de ocorrências dos valores business e economy nos registos do dataset

Para contornar este desbalanceamento, foi então introduzido o nodo "Equal Size Sampling"antes de efetuar o particionamento do dados, no entanto, apesar de a previsão do preço de bilhetes business ter melhorado residualmente, a previsão do preço de bilhetes economy piorou como se pode verificar nas figuras 5.4.3, 5.4.3, 5.4.3, 5.4.3 , pelo que, globalmente o R^2 do modelo piorou.

Com estes novos dados, decidiu-se portanto treinar o modelo sem o nodo "Equal Size Sampling", já que este nodo piorou os resultados do modelo.

Row ID	D Predic...
R^2	0.911
mean abs...	2,224.254
mean squ...	14,640,3...
root mean...	3,826.273
mean sign...	-15.002
mean abs...	0.04
adjusted R...	0.911

Métricas para dados business com o nodo Equal Size Sampling

Row ID	D Predic...
R^2	0.909
mean abs...	2,265.686
mean squ...	15,036,1...
root mean...	3,877.652
mean sign...	-10.123
mean abs...	0.041
adjusted R...	0.909

Métricas para dados business sem o nodo Equal Size Sampling

Row ID	D Predic...
R^2	0.832
mean abs...	871.947
mean squ...	2,394,18...
root mean...	1,547.316
mean sign...	-7.516
mean abs...	0.132
adjusted R...	0.832

Métricas para dados economy com o nodo Equal Size Sampling

Row ID	D Predic...
R^2	0.85
mean abs...	830.022
mean squ...	2,123,73...
root mean...	1,457.306
mean sign...	1.832
mean abs...	0.127
adjusted R...	0.85

Métricas para dados economy sem o nodo Equal Size Sampling

É de notar que todo o particionamento foi aplicado recorrendo a uma random seed de 2022 com o objetivo de poder replicar os resultados a qualquer momento.

5.4.4 Cross Validation

Na construção de um modelo de machine learning um dos conceitos que é necessário ter em atenção é o overfitting. É importante que não ocorra overfitting para que o modelo possa generalizar bem dos dados de treino para dados que ainda não tenham sido vistos. Para evitar overfitting foi utilizada a técnica cross validation.

O nodo **X-Partitioner** foi usado para fazer as várias partições do dataset e escolheu-se 10 validações. O dataset tem muitos registos por isso decidiu-se não usar um valor grande de validações e para além disso este valor é recomendado como suficiente para vários tipos de problemas. Escolheu-se Stratified Sampling pela classe price para manter a mesma porporção de valores de price nas diferentes validações e usou-se a random seed de valor 2022. O nodo **X-Aggregator** foi usado para agregar os resultados das previsões para as várias validações.

O cross validation foi usado com um modelo XGBoost e com os parâmetros ideais definidos na secção tunning dos hyperparâmetros. A métrica *R-Squared* manteve-se com o mesmo valor comparando com o caso sem cross validation. Ainda assim, usamos esta técnica para dar mais robustez ao modelo na previsão de outros dados de teste.

5.4.5 Métricas de qualidade

Tal como já foi referido em algumas secções anteriores, a métrica definida como alvo foi a *R-Squared*. Todo o tratamento foi realizado com vista a maximização deste valor. Os resultados finais de várias métricas de qualidade podem ser vistos na figura 5.16.

R^2:	0,991
Mean absolute error:	999,511
Mean squared error:	4 399 199,32
Root mean squared error:	2 097,427
Mean signed difference:	8,271
Mean absolute percentage error:	0,079
Adjusted R^2:	0,991

Figura 5.16: Métricas de qualidade para o modelo XGBoost

Das métricas também é possível observar qual o erro em média na previsão (MAE): 1000. É necessário primeiro perceber o atributo price antes de entender este resultado. A moeda do atributo price é Indian Rupee e não Euros. Convertendo para Euros é possível ver que o valor médio do atributo price é 21000 Rupee corresponde a cerca de 250 Euros e o MAE é 1000 Rupee que corresponde a cerca de 12 Euros. Analisando este caso entendemos que o valor obtido para esta métrica é bastante satisfatório. AS diferentes métricas foram obtidas a partir do nodo **Numeric Scorer**.

5.5 Resultados principais, análise crítica e trabalho futuro

Da exploração do *dataset* foi possível obter uma série de informações úteis não só para a empresa "Ease My Trip"como para os utilizadores da aplicação web:

- Da figura 5.6 é possível verificar que a grande maioria dos utilizadores compra bilhetes baratos (um valor entre 1105 e 13301.6 rupias indianas). A empresa pode com base nesta informação aplicar estratégias de marketing adequadas para este caso.
- Algumas das airlines contribuem muito pouco para a venda de bilhetes no site, StarAir e Trujet, enquanto outras como a Vistara e Air India representam mais de 60% dos bilhetes vendidos (ver figura 5.8). A empresa pode, com base nisto, rever se compensa manter ligação com as airlines que pouco contribuem com a venda de bilhetes.
- Os utilizadores podem usar as previsões do preço do bilhete para perceber o melhor momento para a compra do bilhete. O fornecimento desta informação aos utilizadores tem de ser considerada com cuidado pela empresa. Hipoteticamente, se a empresa disponibilizar as informações do preço dos bilhetes relativas aos anteriores, isto pode provocar uma mudança drástica de costumes no presente ano, o que poderá ter impacto nos custos e disponibilidade de recursos da airline. Para além disso disponibilizar previsões de preços que se verifiquem erradas no futuro, podem fazer com que os utilizadores se sintam enganados.

Como trabalho futuro existem uma série de aspectos de deveriam ser considerados:

- O modelo pode ficar vulnerável a eventos particulares de cada ano. Acontecimentos recentes como a guerra na Ucrânia e o covid são casos particulares de eventos que podem alterar drasticamente os preços dos bilhetes. Deve-se ter em conta estes factores quando se pretende aplicar modelos de machine learning em casos como o preço de bilhetes.
- Da exploração de dados verificou-se que apenas haviam registos relativos ao mês de Fevereiro e Março. Para fazer uma boa previsão para diferentes alturas do ano seria essencial pelo menos ter registos representativos

de um ano completo. Deveríamos então contactar a empresa "Ease My Trip"e recolher mais dados.
- Uma das explorações que pretendemos fazer foi a identificação de época alta ou época baixa. Contudo não o fizemos pela razão do ponto anterior, apenas tinhamos registos relativos ao mês de Fevereiro e Março.
- Neste trabalho as métricas definidas como objectivo de minimização ou maximização foram esocolhidas pelo grupo. Num caso real os objectivos podem já estar bem definidos pelo contexto do problema ou pela empresa e portanto as métricas a analisar seriam diferentes.
- Qualquer modelo de machine learning apenas tem sucesso quando é aplicado na vida real. Neste trabalho estamos a usar dados de treino e teste estáticos. Um dos passos futuros deveria ser o *deployment* do modelo.
- Da análise do negócio percebeu-se que o intervalo de dias entre a data de compra do bilhete e a data do voo tem influência no preço. Deveríamos pedir à empresa que também fossem fornecidos estes dados com o objectivo de melhorar o modelo.
- Poder-se-ía tentar obter a data e hora de registo do bilhete no site, de modo a obter conhecimento útil como a afluência de utilizadores no site. Isto pode ter benefícios para a empresa no sentido de balancear melhor os recursos computacionais necessários para manter a aplicação web em funcionamento, consoante exista muita ou pouca afluência de utilizadores.

O modelo que revelou melhores resultados foi o XGBoost e o valor da métrica definida como objetivo de maximização, o *R-Squared*, teve um bom resultado 0.991. A métrica *MAE* também teve um resultado bastante satisfatório de 1000 Rupee, que corresponde a cerca de 12 Euros, sendo que o valor médio do preço do bilhete era de 21000 Rupee que corresponde a cerca de 250 Euros.

6 Conclusão

Durante o presente trabalho, entendeu-se que um tratamento adequado é uma parte fulcral para obter bons modelos de previsão. Não só é necessário ter em atenção esta fase, como todas as fases são importantes e estão relacionadas. Por exemplo, o entendimento da lógica de negócio (contexto, atributos, termos técnicos) é uma competência necessária para fazer um bom tratamento dos dados. Para além disto, compreendeu-se melhor sobre que tipos de tratamento se deve fazer utilizando certos algoritmos de machine learning.

Os aspectos mais desafiantes neste trabalho foram a identificação das técnicas de tratamentos dos dados que têm mais efeito no modelo, bem como a percepção relativamente a quando se deve evitar determinadas técnicas de tratamento.

Por outro lado, o presente trabalho permitiu ganhar experiência na área de machine learning, com o objectivo de aplicar os conhecimentos adquiridos durante as aulas T e TP em problemas semelhantes de regressão ou classificação. O facto da escolha do dataset de Flights ter sido de regressão permitiu ao grupo diversificar os seus conhecimentos, pelo que conferiu um impacto positivo na aprendizagem e prática dos conteúdos lecionados.