Escola de Engenharia

Universidade do Minho

Trabalho Prático de Grupo Nº1

Realizado por:

Benjamim Oliveira PG42815
Gonçalo Almeida A84610
Nuno Pereira PG42846
Rui Reis A84930

No âmbito do cursos:
MEIE/MEI/MMC

Unidade Curricular:
SBS (Sistemas Baseados em Similaridade)

Conteúdo

1 Introdução & Contextualização

Este trabalho foi desenvolvido no âmbito na unidade curricular "Sistemas Baseados em Similaridade", do perfil "Machine Learning: Fundamentos e Aplicações", do conjunto de cursos MIEI/MEI/MMC, disponibilizados pela Universidade do Minho.

De forma paralela foram explorados dois *datasets*, um fornecido pelos docentes da unidade curricular e o outro escolhido por nós. O *dataset* fornecido visa registos de incidentes rodoviários na cidade de Braga ao longo do ano de 2019, contendo dados como a magnitude do atraso gerado pelos incidentes, as estradas afectadas, o atraso em segundos provocado pelo acidente, entre muitos outros. O nosso objetivo é prever qual a magnitude de incidentes de cada registo com a maior precisão possível, este parâmetro pode assumir 5 valores distintos dentro do conjunto $C_1 = \{\texttt{None}, \texttt{Low}, \texttt{Medium}, \texttt{High}, \texttt{Very\_High}\}$.

O segundo *dataset* utilizado incide sobre partidas do videojogo de *League of Legends*, mais concretamente, sobre os primeiros 10 minutos das mesmas. *League of Legends* é um jogo multi-player online, em que duas equipas de 5 jogadores se defrontam numa partida com uma duração média de 30 a 40 minutos. Com este estudo pretendemos perceber o quão forte é o efeito dos primeiros 10 minutos numa partida em garantir a vitória.

2 Metodologia CRISP-DM

De forma a garantir a qualidade dos modelos produzidos ao longo deste estudo, a metodologia *Cross Industry Standard Process for Data Mining*, ou CRISP-DM, foi empreendida no intuito de garantir inclusão de abordagens normalmente utilizadas pelos profissionais da área com vantagens imediatas na qualidade do projecto desenvolvido.

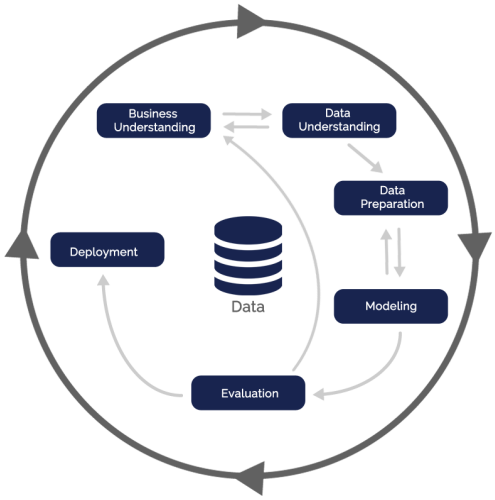

Figura 1: Diagrama ilustrativo da metodologia CRISP-DM.

A figura 1 ilustra os passos considerados por esta metodologia. De imediato, é possível perceber que esta metodologia não possui um início ou fim bem definido, propositadamente este comportamento é definido de forma a conseguir garantir que existe um refinamento do projecto ao longo do tempo, cada iteração contemplando das informações colectadas pela iteração anterior. Esta metodologia pode ser, resumidamente, descrita nos seguintes termos.

1. **Compreensão do Negócio** - É importante perceber qual é o domínio do negócio/tema que estamos a considerar, o que estamos a tentar prever e como utilizar o nosso conhecimento de domínio em nosso prol.

2. **Compreensão dos Dados** - Compreender o conjunto de dados captados, como os utilizar para alcançar o nosso objetivo e explorar estes de forma a descobrir fatores de interesse.

3. **Preparação dos Dados** - Perceber qual a melhor forma para processar os dados de tal forma a que os modelos desenvolvidos possam tirar mais proveito.

4. **Modelação** - Pesar os diferentes modelos utilizados, a diferenças entre si para alcançar o objetivo, bem como ponderar a variação dos hyper-parâmetros.

5. **Apreciação** - Comparar o modelo com os objetivos iniciais propostos e perceber de que forma o modelo permite descrever o desejado.

É importante ter em conta que a fase denotada como *Deployment* da figura 1 foi propositadamente deixada de parte, por não ter sido considerada em grande amplitude ao longo deste projeto.

3 Acidentes em Braga no ano de 2019

Segundo a PORDATA[1], no ano de 2018, em Portugal, foram registados 34235 acidentes rodoviários com vítimas, totalizando 508 vítimas mortais. Como tal, a motivação para o desenvolvimento de um trabalho deste tipo é imediata.

Ao conseguirmos compreender melhor os factores que causam acidentes, trabalhando em específico com os dados da zona de Braga, pretendemos ser capazes de generalizar e compreender quais atributos são os mais informativos no sentido de prever o número de acidentes num determinado dia.

Com um modelo capaz de efectivamente utilizarmos os atributos mais informativos, na prática seriamos capazes de desenvolver campanhas personalizadas para promover a segurança rodoviária, entre outras medidas cujo impacto possa ser directamente mensurável a partir do nosso modelo.

3.1 Compreensão dos Dados

Os dados utilizados neste ponto foram cedidos pelos docentes da unidade curricular e versam incidentes rodoviários na cidade de Braga no ano de 2019.

Estes dados focam informações como por exemplo:

- As estradas afetadas num determinado momento
- O atraso causado pelo incidente
- O dia e hora do incidente

[1] `https://www.pordata.pt/Portugal/Acidentes+de+via%c3%a7%c3%a3o+com+v%c3%adtimas++feridos+e+mortos+++Continente-326`

No conjunto de dados inicial temos 13 colunas.

3.2 Preparação dos Dados

No que toca à preparação de dados:

- Extraímos informações extra da data/hora, com o intuito de tentar encontrar alguma associação entre o nível de incidentes e estas variáveis.
- Guardamos o número total de estradas afetadas, para cada incidente registado.
- Considerando que todos os registos são referentes á cidade de Braga não fazia sentido mantermos o campo `city_name`, assim sendo foi removido.
- Atributos com pouca variância ou alta correlação foram também removidos. Tal é o caso de atributos como `avg_atm_pressure` ou `avg_precipitation`.
- No atributo `magnitude_of_delay` mapeamos as classes com baixa frequência como sendo do tipo "MODERATE"

Uma vez processados os diversos atributos e aplicadas técnicas de `Feature Engeneering` ficamos com a seguinte matriz de correlação.

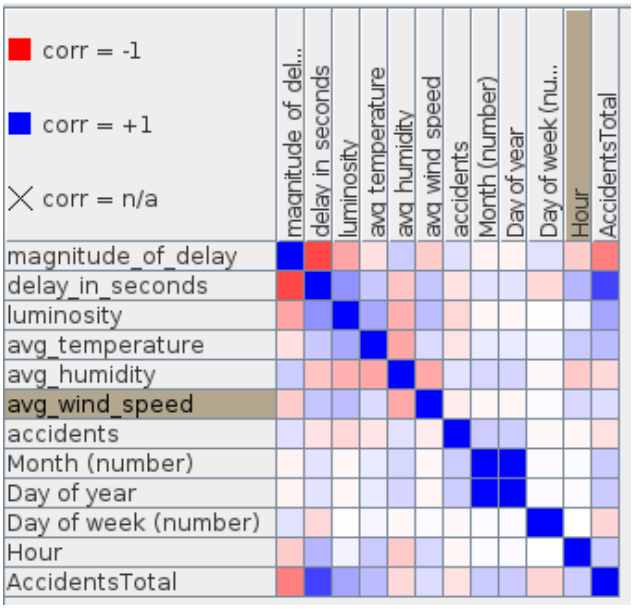

Figura 2: Correlação entre features após processamento.

3.3 Modelação

De forma a garantir a ausência de overfitting decidimos que o ideal seria considerar um modelo baseado em *random forest* que, devido à sua aleatoriadade, permite reduzir fortemente o overfitting.

Com métodos auxiliares de hyper-parameter tuning, fomos capazes de descobrir os hyper-parâmetros ideias para o nosso problema e re-aplicar esses parâmetros directamente na previsão do nosso test set, obtendo uma accuracy perto de 92%, como se verifica em capítulos adiante.

3.4 Apreciação dos Modelos

Utilizando *cross validation* com 10 *folds*, foi possível chegar à conclusão de que a *accuracy* do nosso modelo tende a rondar os 92% que é, de longe, muito melhores que os nossos modelos iniciais, onde a *accuracy* rondava sempre os 88-89%. Como tal, este modelo foi seleccionado para ser submetido.

4 League of Legends - Liga Diamante

O *League of Legends* tem vindo a se apresentar como uma nova tendência que atrai cada vez mais jovens, o seu crescimento exponencial levou à criação de uma modalidade de *esports* que desde então tem somado atração pública. Ao longo dos últimos anos, em 2479 torneios, foram atribuídos perto de 80 milhões de dólares em prémios, sendo que o campeonato mundial de 2018 arrecadou um prémio de cerca de 6 milhões de dólares.

Como tal, facilmente conseguimos perceber que o interesse neste tipo de modalidades já não é o simples entretenimento, e que existem quantidade avultadas de dinheiro a ser movidas, o que implica que qualquer análise mais aprofundada aos comportamentos do jogo podem vir a ser lucrativos.

Então, pretendemos tentar modelar os efeitos que os primeiros minutos de jogo podem ter sobre o resultado final de forma a perceber os fatores decisivos numa partida.

4.1 Compreensão dos Dados

Para estes dados utilizamos recursos disponíveis na internet[2] e descobrimos um dataset que fornece, de forma sucinta, dados sobre os 10 primeiros minutos das duas equipas (azul e vermelha) utilizando um universo amostral de 10000 membros de uma das ligas mais elevadas do jogo, estes dados incidem exaustivamente sobre todas as metas no jogo como, por exemplo,

- Ouro total recolhido, para cada equipa.
- Número de execuções, para cada equipa.
- Número de wards colocadas, para cada equipa.

No total, existem 19 atributos distintos para cada uma das equipa, o que perfaz um total de 38 atributos. Utilizando conhecimento de domínio facilmente conseguimos perceber que ter o mesmo atributo para cada uma das equipas não é indicativo, pelo que seria ideal conseguir agregar ambas as equipas em atributos idênticos.

É também importante denotar que, por conhecimento de domínio, existem atributos que estão necessariamente correlacionados. Por exemplo, o número de execuções da equipa azul e o número de mortes da equipa vermelha estão fortemente correlacionados negativamente, como seria de esperar.

[2] `https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min`

4.2 Preparação dos Dados

A nível de preparação de dados pouco foi feito, no entanto é importante denotar os seguintes aspectos chaves, que foram cruciais na nossa investigação.

- A atributo `gameID` é completamente irrelevante pois é uma chave única associada a cada jogo, sem nenhum tipo de relevància física, por isso esse atributo foi removido.

- Os atributos `blueWins` e `redWins` são ambos binários e disjuntos, o que significa que a soma será sempre 1, pois numa dada partida apenas uma equipa pode ganhar. Como tal, consideramos apenas o atributo `blueWins`, tornando o problema num de prever a vitória ou derrota desta mesma equipa.

- De igual forma, os atributos `blueFirstBlood` e `redFirstBlood` seguem a mesma condição, pelo que só vale a pena ponderar um destes atributos.

- Nos restantes atributos, considerar tanto os atributos da equipa azul e vermelha não faz sentido. Na verdade, estamos interessados em perceber qual a diferença entre os resultados obtidos pela equipa azul num dado atributo e aqueles obtidos pela equipa vermelha. Assim sendo, os restantes atributos, 36, foram reduzidos a 18 por via de *feature engineering*, sendo que no final os atributos em si passam a representar a diferença entre as 2 equipas.

Com isto em mente, podemos agora analisar as correlações existentes de forma mais compreensível.

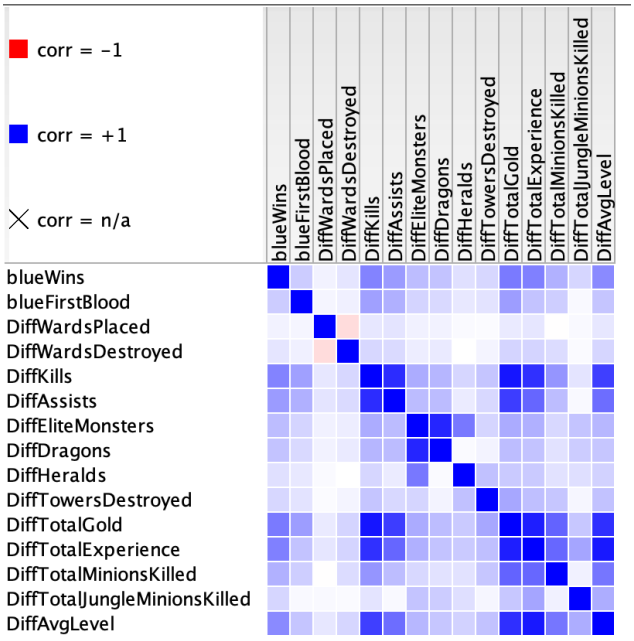

Figura 3: Correlação entre features após processamento.

Pela figura 3 conseguimos detetar algumas correlações menos óbvios do que obtidas anteriormente. Por exemplo, podemos observar que a diferença em kills está fortemente correlacionada positivamente à diferença em ouro entre as duas equipas.

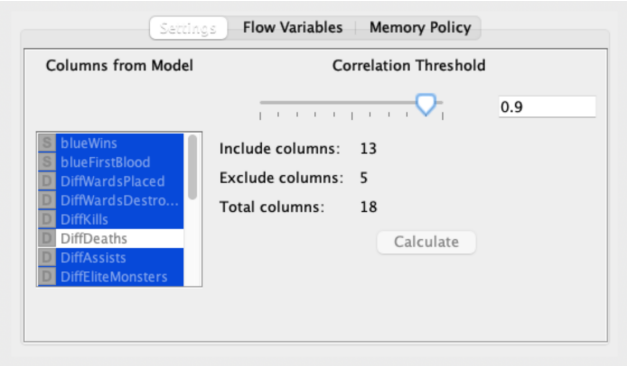

Figura 4: Seleção automática de features correlacionadas.

Como é desnecessário considerar atributos fortemente correlacionados entre si, utilizamos um correlation filter com as configurações da figura 4.

4.3 Modelação

De forma a garantir a ausência de overfitting decidimos que o ideal seria considerar um modelo baseado em *random forest* que, devido à sua aleatoriadade, permite reduzir fortemente o overfitting.

Com métodos auxiliares de hyper-parameter tuning, fomos capazes de descobrir os hyper-parâmetros ideias para o nosso problema e re-aplicar esses parâmetros directamente na previsão do nosso test set, obtendo uma accuracy perto de 73%, como se verifica em capítulos adiante.

4.4 Apreciação dos Modelos

Consideramos que apesar da *accuracy* relativamente baixa, o nosso modelo é capaz de explicar bem suficientemente a variação em vitórias, tendo em conta o pressuposto que estamos só a trabalhar com poucos minutos de jogo.

5 Workflows Desenvolvidos

5.1 Acidente em Braga no ano de 2019

5.1.1 Principais nodos

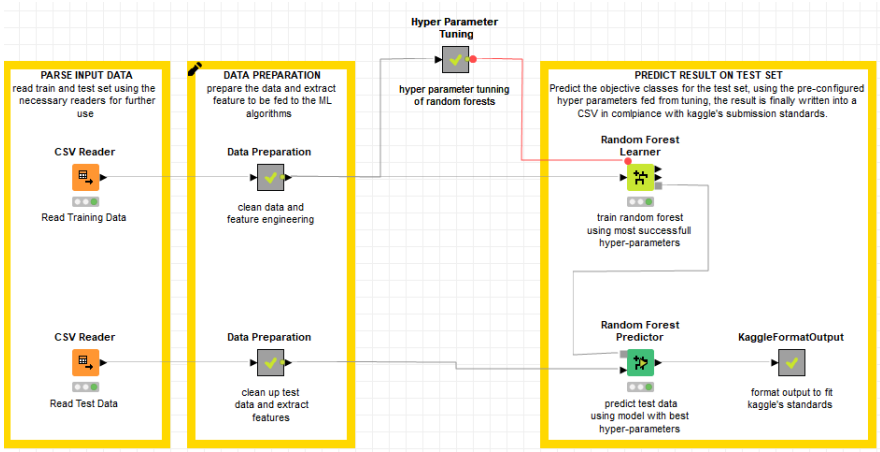

Figura 5: Visão geral do workflow gerado para trabalhar o *dataset*.

1. **Parse Input Data**
   Neste ponto foram carregados para o Knime dois ficheiros, ambos com os mesmos campos de informação, relativos ao número de incidentes rodoviários em Braga. Um dos ficheiros possui dados para usar no treino do modelo, enquanto o outro possui dados para testarmos o modelo.

2. **Data Preparation**
   Relativamente à preparação dos dados apresentamos na figura abaixo o conteúdo do metanodo desenvolvido para esse efeito.

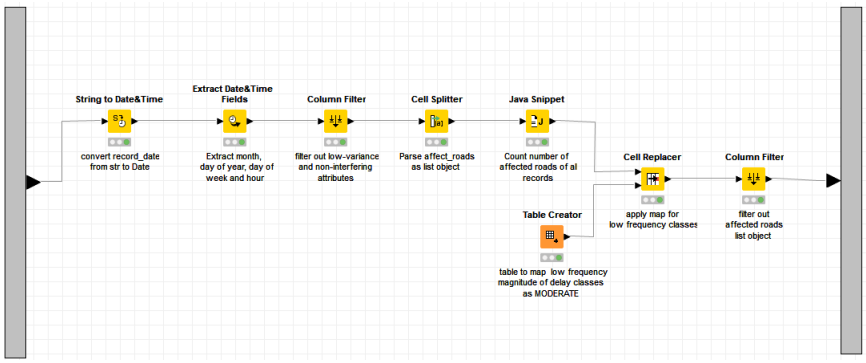

Figura 6: Visão geral do workflow gerado para trabalhar o dataset.

De forma geral extraímos, da data inicial, os seguintes valores: mês, dia do ano, dia da semana e hora. De seguida filtramos dados que, através de uma análise de covariância, descobrimos não estarem relacionados ao nível de acidentes. Filtramos também variáveis que possuiam baixa variação, uma vez que também não nos auxiliavam no processo de previsão. Uma vez terminado este ponto extraimos

o número de estradas afetas por cada registo. Para tal convertemos a coluna "affect_roads"de string para lista e, usando a linguagem de programação Java, efetuamos a contagem dos elementos.

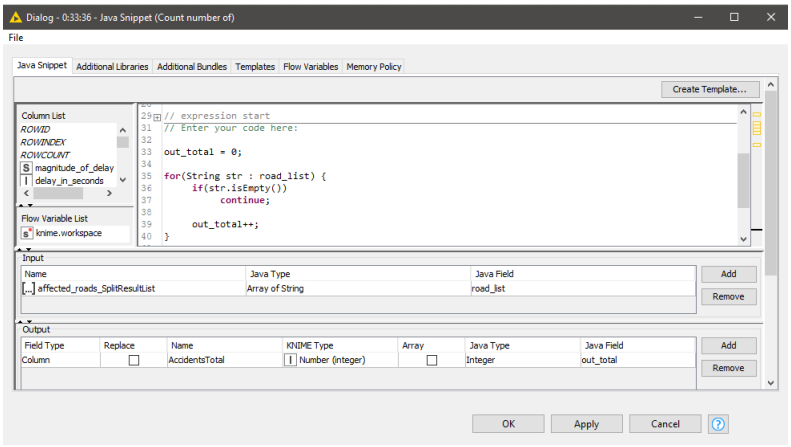

Figura 7: Visão geral do workflow gerado para trabalhar o dataset.

De seguida convertemos o valor de alguns campos relativos a "magnitude_of_delay". Optámos por converter as classes deste campo que tinham uma baixa frequência para a classe "MODERATE".

3. **Hyper Parameter Tuning**
   Para melhorarmos a precisão do modelo gerado optamos por fazer um "tuning"dos parâmetros utilizados. Para o fazer utilizamos loops, testando diferentes combinações entre os critérios de divisão de Random Forests e múltiplos valores para o número de modelos e profundidade da árvore. Uma vez terminados estes loops passamos os valores obtidos como "ideais"para variável de flow e treinamos um modelo com estas definições, já no próximo ponto. Pode ser vista uma representação desta parte do processo mais á frente, mais concretamente na secção "Descrição do modelo gerado".

4. **Previsões**
   Uma vez optimizados os parâmetros do modelo procedemos ao treino do mesmo, para o efeito utilizando um *Random Forest Learner*. Com o modelo gerado prevemos o nível de acidentes para cada um dos registos presentes no *dataset* de teste. Os resultados desta previsão são então exportados, de acordo com as regras estabelecidas para a competição na plataforma *Kaggle*.

5.1.2 Descrição do modelo gerado

1. **Tuning**

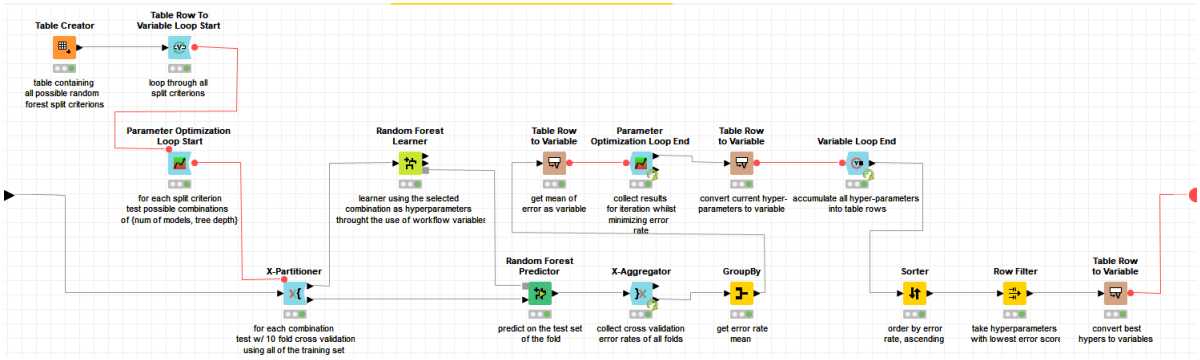

Figura 8: Metanodo utilizado para optimização do hiper-parâmetros.

2. **Características do treino** Após terminado o loop de optimização temos então os valores óptimos para a nossa *Decision Tree*. No caso em concreto os parametros "ideais"foram os seguintes:

   - Número de modelos: 700
   - Profundidade da árvore: 25
   - Split Criterions: Information Gain

Name	Value
numModels	700
treeDepth	25
Objective value	7.919999999999998
splitCriterion	InformationGain
RowID	Row0
knime.workspace	C:\Users\Utilizador\knime-workspace

Figura 9: Settings utilizados para treino do modelo.

5.2 League of Legends - previsão baseada nos 10 primeiros minutos de jogo

5.2.1 Principais nodos

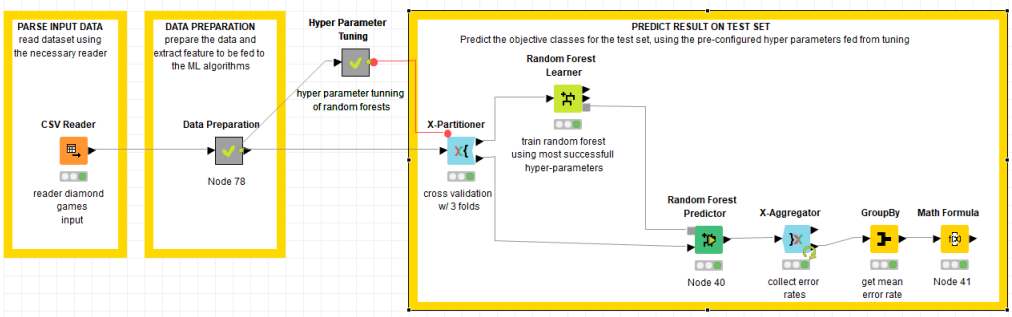

Figura 10: Visão geral do workflow gerado para trabalhar o dataset.

1. **Parse Input Data**
   Como ponto inicial do nosso processo tivemos que ler os dados a serem utilizados. Os dados originais encontravam-se num ficheiro .csv, assim sendo utilizamos o nodo CSV Reader.

2. **Data Preparation**
   Após carregados os dados, passamos ao tratamento/preparação dos mesmos. Para tal desenvolvemos o workflow presente na figura abaixo.

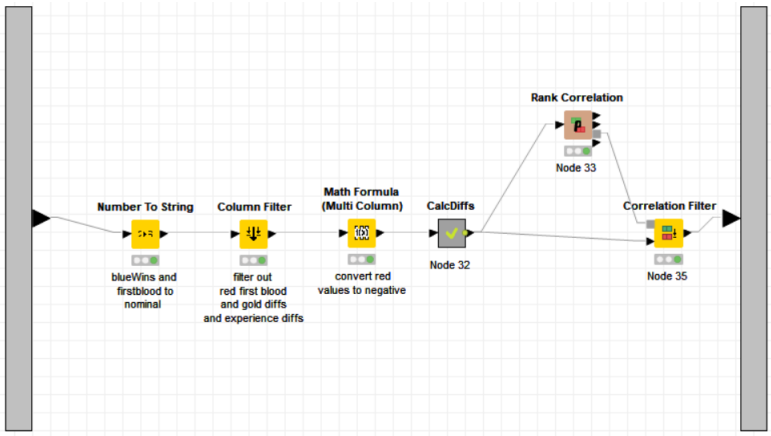

Figura 11: Visão geral do workflow gerado para trabalhar o dataset.

Os dados presentes em cada registo, como já referido antes, eram bastante redundantes, uma vez que o *dataset* possuía todos os dados para ambas as equipas. Desta forma optámos por transformar estes dados num diferencial. Como exemplo, no *dataset*, antes do tratamento, tínhamos dados como o número de mortes que a equipa azul causou e o número de mortes que a equipa vermelha causou. Depois deste processo deixamos de ter estas duas colunas e passamos a ter uma variável que indica a diferença entre o número de mortes causadas por cada equipa. Por fim aplicamos um *correlation filter*, com um *treshold* de quase 1, de forma a eliminar automaticamente as colunas que podiam praticamente ser obtidas a partir das restantes, como por exemplo a diferença entre o número de jogadores mortos pelas equipas e a diferença de vezes que cada equipa morreu.

3. **Hyper Parameter Tuning**
   Para fazermos o tuning utilizamos o metanodo que previamente desenvolvemos para treinar o modelo para a competição de *Kaggle*. Desta forma o funcionamento do mesmo não será explicitado neste ponto, uma vez que seria repetitivo.

4. **Previsões**
   A partir do momento que temos os parâmetros óptimos para o nosso modelo podemos finalmente treinar e testar o mesmo. Para tal utilizamos o nodo "X Partitioner", utilizando 3 folds. Uma vez terminado este processo calculamos a precisão do modelo, considerando que o método de agrupamento dos resultados nos retorna a média do erro entre as 3 folds.

5.2.2 Descrição do modelo gerado

1. **Tuning**
   Pelo motivo referido acima, não iremos entrar em detalhe neste ponto. Existe uma descrição detalhada do mesmo nos pontos 4.1.1 e 4.1.2

2. **Características do treino**
   Após terminado o loop de optimização temos então os valores óptimos para a nossa *Decision Tree*. No

caso em concreto os parâmetros "ideais"foram os seguintes:

- Número de modelos: 500
- Profundidade da árvore: 10
- Split Criterions: Information Gain

Name	Value
numModels	500
treeDepth	10
Objective value	26.946047170766267
splitCriterion	InformationGain
RowID	Row0
knime.workspace	C:\Users\Utilizador\knime-workspace

Figura 12: Settings utilizados para treino do modelo.

6 Sugestões & Recomendações

6.1 Acidentes em Braga no ano de 2019

Ao longo do desenvolvimento do nosso projeto percebemos, analisando os resultados finais, que existe uma diferença muito pouco definida entre o conjunto de classes {`Low`, `Medium`, `High`} pelo que o nosso modelo, segundo as curvas ROC, é péssimo a distinguir entre estas classes. Apesar de ser muito bom a distinguir entre as classes {`None`, `Very High`}, uma performance na melhoria da accuracy está diretamente relacionada com a boa separação classes, o que não é de todo garantido com o modelo atual.

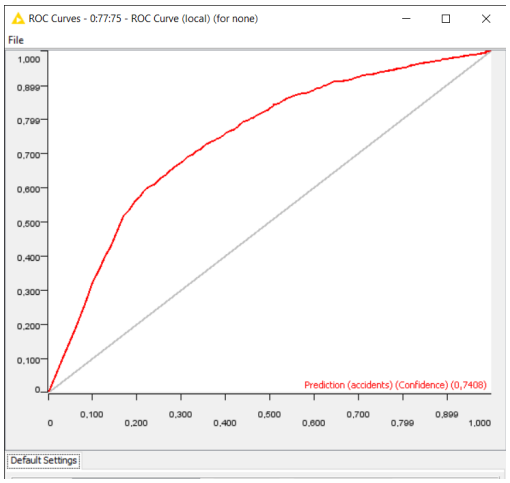

Figura 13: Curva de ROC para a classe `None`.

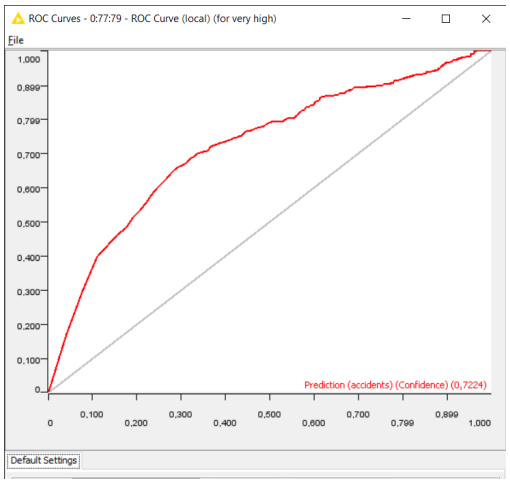

Figura 14: Curva de ROC para a classe `Very High`.

Como podemos ver pelas figuras 13 e 14 rapidamente percebemos que o nosso modelo consegue fazer uma boa distinção entre estas duas. No entanto, se olharmos para as figuras 15, 16 e 17, rapidamente percebemos que o nosso modelo é péssimo nestas classes.

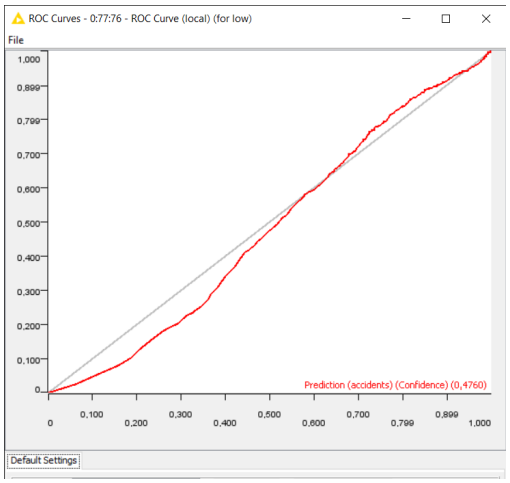

Figura 15: Curva de ROC para a classe `Low`.

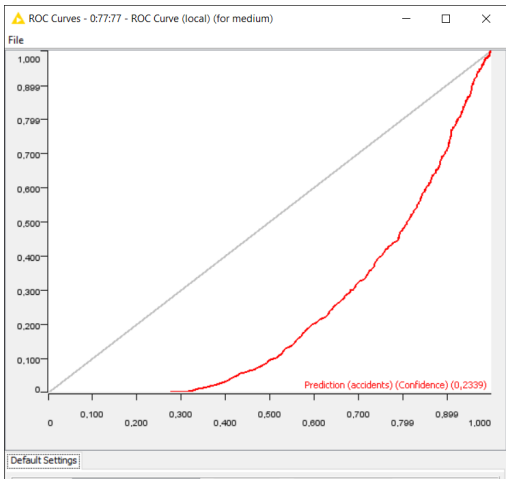

Figura 16: Curva de ROC para a classe `Medium`.

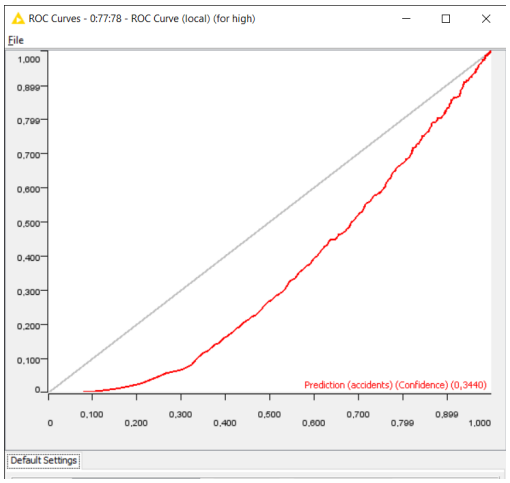

Figura 17: Curva de ROC para a classe `High`.

6.2 League of Legends

Apesar dos resultados bastante promissores apenas com os primeiros 10 minutos de jogo, é notável que, num jogo com dinâmicas tão complexas, utilizar apenas os primeiros 10 minutos é pouco representativo no resultado final.

Existem muitos objetivos, como batalhas em equipa e obtenção de monstros épicos que fornecem pontos de viragem cruciais em jogos, que tendem a concentrarem-se em períodos após os primeiros 10 minutos.

Como tal, e em formato de recomendação, ponderamos que com um aumento do número de minutos considerados poderemos vir a obter *accuracies* mais elavadas.

7 Conclusão

Uma vez terminada a exposição do trabalho efectuado podemos então tirar algumas conclusões sobre o mesmo.

Como ponto inicial começamos por ressalvar a utilidade da metodologia CRISP-DM ao longo deste projeto. Serviu não só para garantir a qualidade dos modelos desenvolvidos como também para estruturar e organizar o desenvolvimento do projeto. Consideramos que esta metodologia foi, em parte, responsável pelos bons resultados obtidos, quer na exploração do *dataset* relativo a acidentes em Braga em 2019, quer na exploração do *dataset* relativo ao videojogo League of Legends.

Incidindo então concretamente sobre os modelos gerados, de uma forma geral estamos satisfeitos com os resultados obtidos. Conseguimos uma precisão alta com o modelo gerado para prever a magnitude de incidentes rodoviários na cidade de Braga, obtendo uma precisão final de cerca de 92% de acertos, reflexo também da exploração e preparação dos dados por nós feita.

Relativamente ao modelo desenhado para prever o vencedor de partidas de League of Legends, embora à primeira vista a precisão possa parecer baixa, rondando os 73%, consideramos que foi também este um resultado bastante bom, confirmando que realmente existe um impacto grande dos 10 primeiros minutos de jogo no resultado final da partida.

Assim sendo, considerando os processos desenvolvidos, apresentados neste relatório, e os resultados dos mesmo, consideramos que conseguimos obter resultados satisfatórios em ambos os modelos desenvolvidos. Conseguimos ainda identificar pontos que poderiam ser trabalhados/modificados de forma a conseguir resultados ainda melhores.