


Introdução

Foram gerados para o trabalho 10.6MB de dados correspondentes a 5471 jogos de sueca.

Estes dados foram gerados pelo nosso programa com base em jogos em que 4 agentes (SmartBotAgent) faziam uso das 4 estratégias implementadas:

- GreedyAgent - joga sempre a carta de maior valor que pode jogar;
- RandomAgent - joga uma carta aleatória;
- CortaAgent - sempre que pode corta a jogada;
- CarefulAgent - joga a carta de menor valor que pode jogar.

Relativamente ao primeiro trabalho foram feitas algumas alterações em código para garantir que obtemos dados com um número uniforme de estratégias de cada tipo.



Descrição do problema de análise de dados

Row No.	card1	suit1	card2	suit2	card3	suit3	card4	suit4	trunfo	handNumber	strategy
1	12	3	3	3	5	3	2	3	2	1	CarefulAGENT
6	11	3	11	0	4	3	15	3	2	6	GreedyAGENT

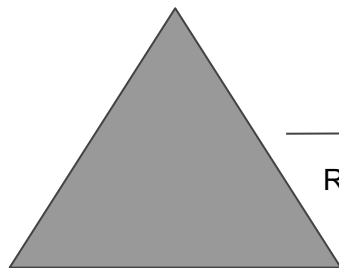
1) Dados gerados a partir do programa

Para melhor processamento dos dados decidimos criar uma abstração que nos reduz as variáveis de jogo sem perda de informação, são elas:

Naipes -
{Espadas, Copas, Paus, Ouros}

Cartas -
{2,3,4,5,6, 11-Dama, 12-Valete, 13-Rei, 14-Sete, 15-Âs}

Mao jogador -
{c1,c2,c3,c4,c5,c6,c7,c8,c9,c10}



Abstraction

Reduz as variáveis que descrevem um jogo em estados/ações



Cartas/Mao Jogador

ValueOfNaipes {2..6, 11...15} - Corresponde à carta mais alta que o jogador tem do naipe x

NumOfNaipes {0..10} - Corresponde ao número de cartas de cada naipe que o jogador na posição 3 tem.

MaoJog3 {NumOfNaipes, ValueOfNaipes}

▽ **Naipes in hand**

Estado de jogo/Ações

Player3CanAssist? {true,false} - Se no instante em que é o jogador 3 a jogar este consegue jogar uma carta do naipe suit1

Player3PodeTrunfar? {true,false} - Se o jogador não consegue assistir à jogada e tem numOfTrunfo>0

Winner1/2 - Vencedor da ronda após os jogadores 1 e 2 jogarem

2) Dados após abstração

valueOfSpa...	valueOfCo...	valueOfPa...	valueOu...	numOfS...	numOfCop...	numOfP...	numOfOu...	WinnerOf1/2	Player3CanAssist?	Player3PodeTrunfar?	Player3Trunfo
12	15	12	15	2	3	2	3	true	true	false	false


Experiências realizadas

Objetivo: Pretendemos classificar a jogada, realizada pelo terceiro jogador numa ronda de sueca, como boa ou má

Variável Objetivo: Good3rdPlay {true,false}

Foi considerada como boa uma jogada que ganha pontos à equipa do jogador 3. No apêndice adicionamos uma abordagem em que é analisada a jogada sem ter em conta o nosso parceiro.

Feature engineering: Começamos por incluir as variáveis mais globais (obtidas diretamente pela abstração) e incrementalmente fomos adicionando as mais particulares construídas a partir destas. No próximo diapositivo temos a correlação entre as variáveis de dados e a coluna alvo.



Estatísticas sobre os dados recolhidos

Set 1 - Simplified

Name	Correlation
WinnerOf1/2	36,43%
Player3Trunfou	3,72%
Player3Assistiu	2,48%

Neste primeiro exemplo obtivemos uma *accuracy* de 78.93% e um erro de 21.07%. Era esperável visto que introduzimos poucos dados para o modelo.

Set 2 - No imperfect information

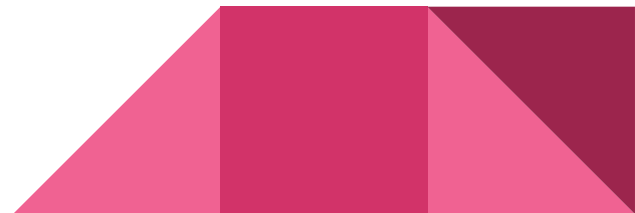
Name	Correlation
WinnerOf1/2	36,43%
Player3Trunfou	3,72%
Player3Assistiu	2,48%
card1	10,18%
card2	3,96%
card3	4,47%

No segundo exemplo já introduzimos dados mais relevantes para o modelo como por exemplo as cartas jogadas pelos jogadores 1, 2 e 3. Assim, já conseguimos obter uma *accuracy* de 82.30% e um erro menor de 17.70%.

No terceiro, já introduzimos uma informação imperfeita (card4) como tal obtivemos uma *accuracy* maior de 88.65% e um erro de 11.35%.

Set 3 - Dealing with imperfect information

Name	Correlation
WinnerOf1/2	36,43%
Player3Trunfou	3,72%
Player3Assistiu	2,48%
card1	10,18%
card2	3,96%
card3	4,47%
card4	4,22%



Análise dos dados com RapidMiner

Modelo utilizado: árvore de classificação (decision tree)

Este modelo pareceu-nos o mais indicado para o problema de classificação já que nos permite analisar de forma organizada e intuitiva o resultado do nosso modelo.

Este algoritmo usa uma árvore de decisão (como um modelo preditivo) para ir de observações sobre um item (representado nos ramos) até conclusões sobre o valor alvo do item (representado nas folhas).

Aplicamos o nosso modelo a três conjuntos de dados diferentes e como seria de esperar obtivemos resultados/conclusões diferentes, por isso abordamos em detalhe cada um fazendo uma suma dos prós e contras para cada um deles.

Executamos ainda diversas iterações com diferentes depths para analisar a performance em função da depth.

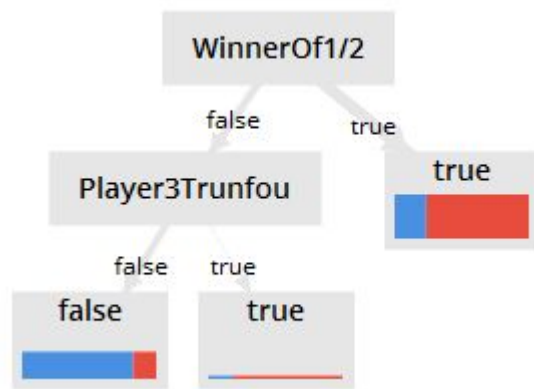


Árvore de classificação - Simplificado

WinnerOf1/2 {true,false} - Se no momento da terceira jogada quem está a ganhar a ronda é o primeiro jogador

Player3Trunfou? - Se o jogador 3 decidiu trunfar (não tem carta do naipe jogada pelo jogador 1)

Player3Assitiu? - Se o jogador 3 assistiu à jogada (se jogou do mesmo naipe que o jogador 1)



Árvore de classificação - Simplificado

Como podemos verificar na árvore gerada pelo modelo o parâmetro *Player3Assistiu?* não foi contabilizado para a árvore uma vez que o seu valor de correlação é inferior ao de *Player3Trunfou?* .

Para este conjunto de variáveis obtivemos uma taxa de acerto de ~70%.

O valor alto surpreendeu-nos, mas pensamos que pode ser explicado pelo facto de o *WinnerOf1/2* ser um fator muito determinante para o resultado objetivo.

accuracy: 78.93%

	true false	true true	class precision
pred. false	3311	649	83.61%
pred. true	1656	5326	76.28%
class recall	66.66%	89.14%	

Árvore de classificação - Sem informação imperfeita

Nesta abordagem decidimos adicionar as cartas jogadas pelos jogadores 1,2,3 com intuito de perceber como a carta jogada pelo jogador três afeta *Good3rdPlay*.

Embora a taxa de acerto tenha-se mantido muito parecida ao conjunto anterior a árvore de classificação torna-se bastante mais complexa o que nos seria útil posteriormente para o desenvolvimento de um agente que tomasse decisões com base na árvore.

Um erro claro nesta abordagem é que ao não considerar *card4* como variável a taxa de erro na classificação em que *WinnerOf1/2* é false é de 100%. Isto é observável na árvore gerada, sendo que uma das ramificações de *WinnerOf1/2* não foi desenvolvida.

A árvore gerada foi anexada por ser demasiado grande. (*Anexo A*)

accuracy: 82.30%

	true false	true true	class precision
pred. false	3073	43	98.62%
pred. true	1894	5932	75.80%
class recall	61.87%	99.28%	


Árvore de classificação - Informação imperfeita

Na nossa última abordagem usamos também como input o valor de card4. Com isto pretendia-mos obter uma estimativa do quão boa é a jogada do jogador 3, em função das possíveis jogadas do jogador quatro.

Para este conjunto de variáveis obtemos a melhor taxa de acerto (88.65%) até ao momento, o que é lógico uma vez que até agora não tínhamos em conta a tomada de decisão do jogador quatro.

A árvore gerada é muito grande e demasiado complexa para humanamente tirar conclusões estratégicas.

Algo imediatamente visível na árvore é que o nosso problema da abordagem anterior (em que WinnerOf1/2=2 dava resultados mal classificados) foi resolvido e essa parte da árvore foi desenvolvida como seria de esperar.



Árvore de classificação - Informação imperfeita

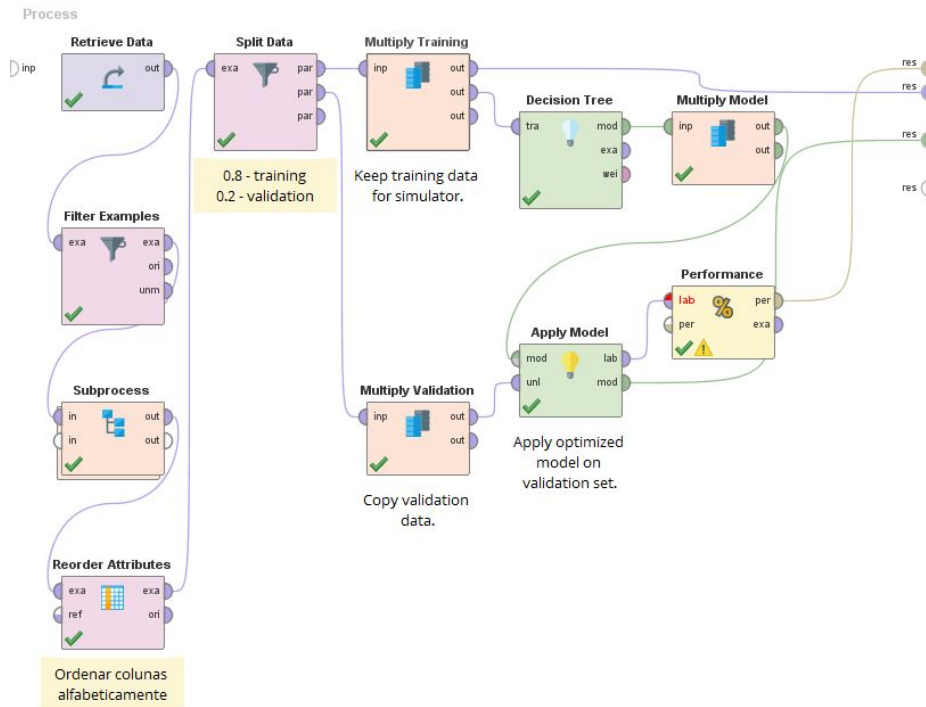
Algo que pensamos para reduzir o tamanho da árvore e consequentemente facilitar a interpretação estratégica dos resultados seria simplificar mais ainda as variáveis card1..4 para relações do género: jogou carta mais alta do mesmo naipe, jogou carta mais baixa de trunfo,...

Isto porque apesar de termos obtido uma taxa de acerto razoável as decisões (folhas) da nossa árvore possuem valores numéricos para card1..4 e ao converter em decisões binárias podíamos diminuir drasticamente o resultado da árvore. Infelizmente por termos perdido já muito tempo na preparação dos dados isto não foi possível.

accuracy: 88.65%

	true false	true true	class precision
pred. false	4062	337	92.34%
pred. true	905	5638	86.17%
class recall	81.78%	94.36%	

Processos RapidMiner



Primeiro, usamos o RapidMiner para fazer o tratamento de dados do nosso jogo.

Depois do tratamento exportamos o resultado para o ficheiro que vai ser usado neste processo e importamos no processo da imagem ao lado.

De seguida retiramos os valores que estavam em falta, escolhemos o *target value*, removemos as colunas desnecessárias e ordenamos as mesmas.

Posteriormente, dividimos os dados em 80% dados de treino e 20% de validação.

Escolhemos o modelo *Decision Tree*, aplicamos ao modelo de treino e verificamos os resultados obtidos.

Resultados de outras experiências interessantes

Nesta experiência o nosso objectivo foi parecido à anterior, isto é, pretendemos classificar a jogada do terceiro jogador como boa ou má porém, em vez de se tratar de um jogo normal de sueca, nesta experiência fizemos o terceiro jogador jogar sozinho.

Variável Objetivo: Good3rdPlayWithoutPartner {true,false}

Foi considerada como boa uma jogada a que ganha pontos ao jogador 3.

Como podemos verificar na imagem ao lado, e como era de prever, a correlação que o *WinnerOf1/2* tinha com a variável alvo desceu bastante (de 36.43% para 0.45%).

Tal se verifica devido ao facto da estratégia que o jogador 3 usa para ganhar ser completamente diferente agora.

Visto que joga sozinho, já não é tão necessário saber quem ganhou entre o jogador 1 e 2.

Name	Correlation
AssistiuParaGanhar	32,93%
Good3rdPlay	22,99%
card3	19,09%
biggestAssistInHand	14,08%
Player3GanhouDepoisDeTrunfo	12,92%
roundWinningPlayer	9,03%
Player3TemTrunfo?	5,65%
Player3CanAssist?	4,97%
Player3PodeTrunfar?	4,58%
card1	2,16%
Player3Assistiu	2,04%
valueOfSpades	1%
valueOfPaus	0,92%
valueOfOuros	0,89%
valueOfCopas	0,83%
card4	0,68%
card2	0,65%
Player3CanAssistOuros?	0,60%
Player3CanAssistCopas?	0,59%
Player3CanAssistPaus?	0,58%
Player3CanAssistSpades?	0,53%
strategy	0,53%
WinnerOf1/2	0,45%
handNumber	0,36%
Player3Trunfo	0,31%
playerNum4	0,21%
playerNum3	0,16%
numOfSpadesInHand	0,13%
numOfPausInHand	0,12%
numOfCopasInHand	0,11%
numOfOurosInHand	0,08%
playerNum2	0,03%
playerNum1	0,01%
suit1	0%
trunfo	0%
suit4	0%
suit2	0%
suit3	0%

Conclusões

Neste trabalho tivemos bastante trabalho na preparação dos dados uma vez que, como tínhamos de apresentar dados relevantes para determinar se uma jogada era boa ou não, gastamos muito tempo na preparação dos mesmos.

Na primeira parte do trabalho tivemos de guardar a informação necessária do jogo num dataSet porém, para poder usar essa informação e torná-la útil tivemos de criar uma abstração.

Conseguimos concluir que, por vezes, não conta só o processo que se tem mas também a qualidade dos dados que introduzimos no processo.

