Uma análise de técnicas de classificação para predição de valores atribuídos aos jogadores no jogo FIFA

Maurício D.C. Balboni[1], **Helida Santos**[1], **Giancarlo Lucca**[2]

[1]Programa de Pós-graduação em Engenharia da Computação
Centro de Ciências Computacionais
Universidade Federal de Rio Grande (FURG)

{balboni,helida}@furg.br

[2]Programa de Pós-Graduação em Modelagem Computacional
Universidade Federal de Rio Grande (FURG)

giancarlo.lucca@furg.br

Abstract. *This work seeks to compare an application of two different machine learning algorithms. Precisely we take in consideration Public-C and a rule-based algorithm fuzzy, FURIA. For this, it uses two tools that help in the elaboration of predictive models, the KEEL software and an R programming language. The work uses accuracy as a measure of the quality of predictive models, in addition to presenting their confusion matrices. Showing that the model using FURIA presents a performance considered satisfactory.*

Resumo. *Esse trabalho busca realizar um comparativo de uma aplicação de dois diferentes algoritmos de aprendizado de máquina. Precisamente levamos em consideração o Public-C e um algoritmo baseado em regras fuzzy, o FURIA. Para isso, utilizamos duas ferramentas que auxiliam na elaboração de modelos preditivos, o software KEEL e a linguagem de programação R. O trabalho utiliza a acurácia como medida de qualidade dos modelos preditivos além de apresentar suas matrizes de confusão. Mostrando que o modelo utilizando o FURIA apresenta um desempenho considerado satisfatório.*

1. Introdução

O aprendizado de máquina [Zhang 2020] é um subcampo da ciência da computação dedicado a desenvolver técnicas e algoritmos na qual possibilitam o computador a desenvolver modelos que aprendem automaticamente sobre determinada aplicação. Os aprendizados podem ser divididos em supervisionado e não supervisionado [Monard and Baranauskas 2003a]. No presente trabalho será abordado apenas o aprendizado supervisionado, que consiste em gerar modelos de aprendizado a partir de resultados pré-definidos, utilizando valores passados pela variável destino do modelo para a geração dos resultados de saídas. Esses valores são utilizados para realizar a supervisão das previsões do modelo, permitindo o ajuste do mesmo com base nos erros preditivos [Ayodele 2010].

Os problemas de classificação buscam predizer a qual classe uma determinada instância pertence dentro de um conjunto finito de classes. Esses problemas podem ter apenas duas classes, por exemplo: se uma pessoa possui determinada doença ou não

$\{$tem a doença, não tem a doença$\}$. Além disto, podem existir n classes possíveis de predizer, por exemplo: afim de predizer como está a temperatura, podemos dividir em três classes, como $\{$quente, morno, frio$\}$, ou em cinco classes como $\{$muito quente, quente, morno, frio, muito frio$\}$. Existem diversas maneiras de lidar com esse problema, e um deles são as árvores de decisões [Monard and Baranauskas 2003b]. Estas árvores são um mapeamento dos possíveis resultados de uma determinada aplicação, subdividindo os dados em conjuntos cada vez menores e mais específicos conforme a árvore vai se expandindo, até gerar um modelo que tenha a capacidade de predizer as classes com base nas características da aplicação.

Na lógica clássica também conhecida como lógica booleana [Chenci et al. 2011], os conjuntos são denominados puros e um dado elemento desse domínio pode pertencer, representado por 1, ou não pertencer, representado por 0, ao referido conjunto. Essa lógica mostra-se limitada ao definir alguns tipos de sistemas, visto que não existe uma fronteira bem definida para decidir quando um elemento pertence ou não a alguns respectivos conjuntos. Visando contornar essa situação, surge a lógica *fuzzy* [Zadeh 1965], na qual considera infinitos valores entre o intervalo $[0, 1]$. Assim, um valor é atribuído ao elemento do conjunto e chamado de grau de pertinência, que indica o quanto este elemento (ou informação) pertence ao conjunto no determinado intervalo.

Nos dias atuais, uma das aplicações da teoria dos conjuntos *fuzzy* desenvolvida por Zadeh [Zadeh 1965] são os sistemas de classificação baseados em regras *fuzzy* (FRBSs[1]). Estes tipos de sistemas constituem uma extensão dos sistemas baseados na lógica clássica.

Um FRBS apresenta dois componentes principais, sendo eles:

1. Sistema de inferência: coloca em prática o processo de inferência *fuzzy* necessário para obter uma determinada saída do FRBS quando uma entrada é especificada.
2. Base de regras *fuzzy*: representa o conhecimento sobre determinado problema a ser resolvido, na qual é constituída pelo conjunto de regras *fuzzy*.

O presente trabalho tem como objetivo realizar um comparativo de uma aplicação de aprendizado de máquina utilizando diferentes algoritmos e utiliza-se de diferentes ferramentas que auxiliam na elaboração de modelos preditivos. Foi escolhido o ambiente de programação R-Studio[2] para compilar a linguagem de programação R, e nele foi treinado um modelo de classificação com as configurações predefinidas pelo ambiente e com validação cruzada igual a dez. A ferramenta KEEL[3] foi utilizada para fins comparativos, com o algoritmo PUBLIC-C [Rastogi and Shim 2000] para modelos de classificação e o algoritmo FURIA [Hühn and Hüllermeier 2009] para realizar um modelo de regras *fuzzy*.

O presente trabalho está dividido da seguinte forma, na Seção 2 é possível encontrar a fundamentação teórica necessária para entendimento do trabalho. A Seção 3 apresenta a metodologia e na Seção 4 encontram-se os resultados das aplicações dos modelos de classificação, além das métricas de qualidade do mesmo. A Seção 5 apresenta a conclusão e trabalhos futuros.

[1]Neste trabalho, usaremos a sigla em inglês para *Fuzzy Rule-based Systems*

[2]www.rstudio.com

[3]Disponível em: www.keel.es

2. Fundamentação Teórica

Nesta seção, os principais conceitos relacionados ao trabalho são apresentados brevemente. Começamos apresetando os tópicos sobre a lógica *fuzzy*. Na sequência, os modelos utilizados são discutidos.

2.1. Teoria dos Conjuntos *Fuzzy*

Na teoria clássica, um elemento pode pertencer ou não a um conjunto. Dado um universo de discurso denotado por U, e um elemento x pertencente a U, define-se o grau de pertinência pela equação 1, essa função é chamada de função característica na teoria clássica de conjuntos.

$$\mu_U(x) = \begin{cases} 1, & \text{se } x \in U \\ 0, & \text{se } x \notin U \end{cases} \tag{1}$$

A fim de se obter uma caracterização mais ampla, uma vez que alguns elementos podem pertencer a um conjunto mais do que os outros, surge a logica *fuzzy* [Zadeh 1965], onde o valor de pertinência, pode assumir valores entre 0 e 1, sendo 0 indicando uma exclusão absoluta no conjunto e 1 assumindo uma inclusão total na pertinência. Essa generalização aumenta a capacidade de expressão da função característica. Formalmente seja U o conjunto universo na qual pode ser continuo ou discreto, e um conjunto *fuzzy* A pertencente ao universo U, define-se a função de pertinencia $\mu_A(x) :\rightarrow [0; 1]$

Para expressar classes é muito comum a utilização de valores qualitativos para expressar valores quantitativos. Uma variável categórica tem como característica assumir valores dentro de um conjunto de termos linguísticos, ou seja, palavras ou definições. Assim, ao invés de assumir instâncias numéricas, as mesmas assumem instâncias linguísticas, como por exemplo, a altura de uma pessoa, podemos representar essa variável quantitativa com 170 centímetros, ou discretizar esse valor para que o mesmo pertença a uma palavra ou definição, como o conjunto alto, baixo, médio.

2.2. O método FURIA

FURIA (*Fuzzy Unordered Rule Induction Algorithm*) [Hühn and Hüllermeier 2009] é uma extensão do algoritmo evolutivo de regras *fuzzy* RIPPER [Cohen 1995], preservando as vantagens oferecidas pelo RIPPER, o FURIA recebe algumas modificações visando um melhor desempenho e aprende regras *fuzzy* no lugar de regras convencionais e conjuntos de regras não ordenados ao em vez de listas de regras. Além disto, o FURIA utiliza-se de funções de pertinência trapezoidais [Amendola et al. 2005] para melhor se adaptar aos problemas.

2.3. O método PUBLIC-C

Uma árvore de decisão pode ser gerada com todos as instâncias do conjunto de treinamento. No entanto para gerar uma árvore com maior precisão e menos sensíveis a novas instâncias de teste pode ser utilizado uma árvore menor que também classifica todos os registros conhecidos [Quinlan and Rivest 1989]. Para isso, a maioria dos algoritmos realiza uma fase de poda após a sua fase de construção. A partir disto, o surge o Public-C, um algoritmo de classificação de árvore de decisão aprimorado, que integra a fase de poda com

a construção inicial da árvore. É previamente determinado se o nó deverá ser podado ou não durante a fase de poda, para isto, o PUBLIC-C calcula um limite inferior da subárvore de custo mínimo enraizada no nó. Essa estimativa é utilizada para a identificação dos nós que certamente serão podados, e com isso evitando o custo computacional em suas divisões [Rastogi and Shim 2000].

3. Metodologia

Para a realização desse trabalho, é importante ter conhecimento da base de dados utilizado para que se possa entender as decisões de pré-processamento utilizadas visando a obtenção de melhores resultados. Com isto, essa seção detalha todo o processo realizado.

3.1. Descrição da Base de Dados

A base de dados Fifa, foi retirado da plataforma "sofifa"[4], essa base de dados se refere as informações referentes aos jogadores de futebol do jogo "FIFA 2018", o mesmo compreende 88 atributos referentes a cada uma das suas 18207 instâncias de teste, esses atributos constituem as características dos jogadores, como seus atributos físicos, os valores de desempenho do atleta e também características relacionadas ao valor de mercado que o mesmo tem. O objetivo proposto é que a partir da base de dados em questão, se possa estimar o atributo "Value", na qual se refere ao valor de mercado de cada jogador presente no jogo.

3.2. Pré-processamento

Inicialmente é preciso destacar que a base de dados é muito grande. Precisamente, ele tem dimensão inicial de 18208 instâncias x 84 atributos, além de diversos dados fora de padrões reconhecidos pelas linguagens de programação e com muito valores faltantes em alguns instâncias de teste. Sendo assim, indispensável a realização de redução de atributos.

Para isto, foi excluído todas as instâncias de teste em que tinham valores faltantes, além de exclusão de atributos em que não eram relevantes computacionalmente para o modelo, como por exemplo os atributos "ID, PHOTO, Nationality, Flag, logo"e outros. Além disto, foi padronizados valores da base de dados, como no lugar de K em valores numéricos, foi substituído pelo valor 000, e no lugar de M foi substituído por 000000, além da retirada de caracteres não benéficos para o modelo, como por exemplo o caractere $. Com isto foi reduzido um total de 40 atributos, resultando numa base de dados com dimensões de 17605 instâncias x 44 atributos. Contudo ainda não é o ideal para realização dos treinamentos. Para tratar o atributo alvo, primeiramente foi verificado que ele possui valores numéricos, como a proposta do trabalho é realizar um aprendizado de classificação, foi realizado a discretização dos valores. As classes geradas foram definidas como "Entre 5700 e 8500, Entre 8500 e 11500, Maior que 11500 e Menor que 5700".

Ainda assim, o *dataset* continua com uma dimensionalidade muito grande, então foi realizada uma análise mais detalhada, e foi verificado como se comportavam os valores baixos, foi visualizado que todos as instâncias que tinha o valor menor que 3000 eram muito parecidas, praticamente iguais, então foi realizado a exclusão dessas instâncias, restando apenas 1583 instâncias. Ao final, a base de dados foi reduzida em 91% em

[4]Base de dados disponível em: https://www.sofifa.com.br

relação a minha base inicial. Resultando em uma dimensionalidade de 1582 instâncias x 44 atributos.

3.3. Configuração da proposta

Os dois algoritmos selecionados neste trabalho foram executados com validação cruzada igual a dez e com as configurações pré-definidas pelo KEEL, sendo elas:

3.3.1. FURIA

Número de otimizações igual a dois e número de divisões igual a três

3.3.2. PUBLIC-C

O parâmetro *NodeBetweenPrune* igual a vinte e cinco.

4. Resultados

Foi realizado o treinamento do modelo em R utilizando uma validação cruzada [Schaffer 1993] igual a dez e dividido em 80% dos dados para treinamento e 20% dos dados para teste. Na Tabela 1 se encontra a matriz de confusão [Li et al. 2004] dos dados para teste do modelo, e podemos visualizar que a grande maioria das predições foram errôneas nas classes laterais dela, apenas na classe ”Maior que 11500”que se encontra um erro consistente para todas as classes. O modelo obteve uma acurácia de 0.851. Na Figura 1 se observa as medidas de qualidade entre as classes, nela podemos visualizar que a classe que obteve a pior precisão nas predições foi a classe ”Entre 8500 e 11500”, enquanto a que apresentou a melhor precisão foi a classe ”Menor que 5700”.

Tabela 1. Matriz de Confusão do modelo em R

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	133	6	0	0
Entre 5700 e 8500	1	100	9	0
Entre 8500 e 11500	1	10	22	3
Maior que 11500	8	8	13	82

```
Statistics by Class:

                     Class: Entre 5700 e 8500 Class: Entre 8500 e 11500
Precision                              0.9091                    0.61111
Recall                                 0.8065                    0.50000
F1                                     0.8547                    0.55000
Prevalence                             0.3131                    0.11111
Detection Rate                         0.2525                    0.05556
Detection Prevalence                   0.2778                    0.09091
Balanced Accuracy                      0.8848                    0.73011
                     Class: Maior que 11500 Class: Menor que 5700
Precision                            0.7387                0.9568
Recall                               0.9647                0.9301
F1                                   0.8367                0.9433
Prevalence                           0.2146                0.3611
Detection Rate                       0.2071                0.3359
Detection Prevalence                 0.2803                0.3510
Balanced Accuracy                    0.9357                0.9532
```

Figura 1. Medidas de Qualidade das Classes

Na tabela 2 encontra-se a matriz de confusão das predições realizadas utilizando o software ”KEEL”com o algoritmo Public-C, nela foi utilizada uma validação cruzada igual a dez e o modelo foi dividido em treino e teste pela definição padrão do KEEL. A partir disto, temos a acurácia do modelo de teste definida em 0.857.

Tabela 2. Matriz de Confusão do modelo Public-C

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	575	35	0	24
Entre 5700 e 8500	40	356	19	20
Entre 8500 e 11500	1	18	146	32
Maior que 11500	0	6	31	279

Por fim, com base na tabela 3 encontra-se a matriz de confusão das predições realizadas pelo KEEL com o algoritmo de classificação *fuzzy* FURIA. Com base na tabela, podemos inferir que a acurácia do modelo de teste foi 0.865.

Tabela 3. Matriz de Confusão do modelo FURIA

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	596	15	1	22
Entre 5700 e 8500	35	357	28	15
Entre 8500 e 11500	3	37	124	33
Maior que 11500	6	3	14	293

5. Conclusão

Neste trabalho pode-se conhecer o funcionamento de duas ferramentas que auxiliam na execução de modelos preditivos de aprendizado de máquina, além de comparar qual retornou o melhor resultado. Além disto, foi aplicado técnicas de pré-processamento para tratar os dados, modelando as informações para que seja maximizado o desempenho computacional e a precisão na predição dos modelos treinados.

Com base nos resultados apresentados, podemos visualizar o ganho de precisão nos modelos ao adotar um modelo de classificação baseado em regras *fuzzy*, onde o mesmo teve um ganho de 0.9% na sua precisão em relação ao modelo utilizando o KEEL com o algoritmo Public-C. Além disto, o modelo obteve um aumento de 1.6% na acurácia em relação ao modelo de classificação obtido na linguagem de programação R.

Agradecimentos

Os autores gostariam de agradecer às agências de fomento CAPES (PNPD 464880/2019-00) e FAPERGS (ARD 19/2551-0001279-9) pelo auxílio financeiro.

Referências

Amendola, M., Souza, A. d., and BARROS, L. C. (2005). Manual do uso da teoria dos conjuntos fuzzy no matlab 6.5. *FEAGRI & IMECC/UNICAMP*, pages 1–44.

Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, 3:19–48.

Chenci, G. P., Rignel, D. G., and Lucas, C. A. (2011). Uma introdução á lógica fuzzy. *Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica*, 1(1).

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier.

Hühn, J. and Hüllermeier, E. (2009). Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319.

Li, Y.-X., Tan, C. L., Ding, X., and Liu, C. (2004). Contextual post-processing based on the confusion matrix in offline handwritten chinese script recognition. *Pattern Recognition*, 37(9):1901–1912.

Monard, M. C. and Baranauskas, J. A. (2003a). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32.

Monard, M. C. and Baranauskas, J. A. (2003b). Indução de regras e árvores de decisão. *Sistemas Inteligentes-Fundamentos e Aplicações*, 1:115–139.

Quinlan, J. R. and Rivest, R. L. (1989). Inferring decision trees using the minimum description lenght principle. *Information and computation*, 80(3):227–248.

Rastogi, R. and Shim, K. (2000). Public: A decision tree classifier that integrates building and pruning. *Data Mining and Knowledge Discovery*, 4(4):315–344.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, 13(1):135–143.

Zadeh, L. A. (1965). Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific.

Zhang, X.-D. (2020). Machine learning. In *A Matrix Algebra Approach to Artificial Intelligence*, pages 223–440. Springer.