Available online at www.sciencedirect.com

ScienceDirect

Knowledge-Based Systems 00 (2022) 1–6

Knowledge-Based Systems

www.elsevier.com/locate/procedia

Comparação experimental de classificadores na base de dados do FIFA 19

Lucas Nildaimon dos Santos Silva[a], Bruno Silva Sette[a], Felipe Alves Cordeiro[a], Vitor Hugo Guilherme[a], Lucas Severi[a]

[a]*Departamento de Computação - Universidade Federal de São Carlos*

Abstract

Keywords:

1. Contextualização

Neste trabalho é avaliado o resultado de experimentos com seis classificadores na base de dados do FIFA 19, a tarefa de classificação neste caso consiste em rotular a posição de campo dos jogadores, são elas defensor, meio campista e atacante. Desta forma, tem-se uma tarefa de classificação multi-classe com três possíveis rótulos.

Para a realização dos experimentos foram selecionados classificadores de propriedades diferentes:

1.1. Regressão Logística

Segundo a definição de Urso [1]:

> Em geral, o classificador de regressão logística pode usar uma combinação linear de mais de um valor de recurso ou variável explicativa como o argumento da função sigmóide. A saída correspondente da função sigmoide é um número entre 0 e 1. O valor médio é considerado o limite para a classe 1 e a classe 0. Em particular, uma entrada produzindo um resultado maior que 0,5 é considerada como sendo classe 1. Inversamente, se a saída for menor que 0,5, então a entrada correspondente é classificada como pertencente à classe 0.

1.2. SVM

De acordo com a definição de Gove [2]:

> Os Support Vector Machines (SVMs) são um conjunto de métodos de aprendizado supervisionados relacionados, que são populares para realizar análises de classificação e regressão usando análise de dados e reconhecimento de padrões. Os métodos variam na estrutura e nos atributos do classificador. O SVM mais conhecido é um classificador linear, que prevê a classe de membro de cada entrada entre duas classificações possíveis. Uma definição mais precisa indicaria

> que uma máquina de vetores de suporte constrói um hiperplano ou conjunto de hiperplanos para classificar todas as entradas em um espaço de alta dimensão ou mesmo infinito. Os valores mais próximos da margem de classificação são conhecidos como vetores de suporte. O objetivo do SVM é maximizar a margem entre o hiperplano e os vetores de suporte.

1.3. KNN

O K-Nearest Neighbors (KNN) é um método padrão de aprendizado de máquina que foi estendido aos esforços de mineração de dados em grande escala.

Neath [3] define o uso de KNN na tarefa de classificação da seguinte maneira:

> A abordagem de classificação do k mais próximo ao vizinho é uma abordagem relativamente simples da classificação que é completamente não-paramétrica. Dado um ponto x que desejamos classificar em um dos grupos K, encontramos os k pontos de dados observados que estão mais próximos de x. A regra de classificação é atribuir x à população que possui os pontos de dados mais observados dos k vizinhos mais próximos. Pontos para os quais não há maioria são classificados para uma das populações majoritárias aleatoriamente, ou não classificados.

1.4. Árvore de decisão

As árvores de decisão são modelos de suporte à decisão que classificam padrões usando uma sequência de regras bem definidas. Eles são gráficos em forma de árvore nos quais cada nó da ramificação representa uma opção entre um número de alternativas e cada nó da folha representa um resultado das escolhas cumulativas [4].

De acordo com a definição de Kounelakis [5]:

> Algoritmos de árvore de decisão formam uma classe de algoritmos populares de reconhecimento de padrões, cujo princípio básico é dividir e conquistar. Cada nó de decisão em uma árvore divide um conjunto de dados em duas partes. A partição só acontece quando precisa. Quando uma partição é necessária, isso significa que existem duas classes de padrões no conjunto de dados. Isso significa que o conjunto de dados não é puro ou a pureza não é satisfeita. Na prática, medidas de impureza, como a entropia, são frequentemente usadas. Os algoritmos de árvore de decisão mais comumente usados são CART [6] e ID3, C4.5, C5 [7].

1.5. Floresta Aleatória

Segundo Breiman [8], as florestas aleatórias podem ser definidas da seuinte maneira:

> Florestas aleatórias são uma combinação de preditores de árvores, de tal forma que cada árvore depende dos valores de um vetor aleatório amostrado de forma independente e com a mesma distribuição para todas as árvores na floresta. O erro de generalização para as florestas converge quase certamente para um limite, à medida que o número de árvores na floresta se torna grande. O erro de generalização de uma floresta de classificadores de árvores depende da força das árvores individuais na floresta e da correlação entre elas. Usar uma seleção aleatória de recursos para dividir cada nó gera taxas de erro que se comparam favoravelmente ao Adaboost [9], mas são mais robustas em relação ao ruído. As estimativas internas monitoram o erro, a força e a correlação e são usadas para mostrar a resposta ao aumento do número de recursos usados na divisão. Estimativas internas também são usadas para medir a importância variável. Essas idéias também são aplicáveis à regressão.

1.6. Gradient Boosting

Gradient Boosting é uma técnica de aprendizado de máquina para problemas de regressão e classificação, que produz um modelo de previsão na forma de um conjunto de modelos de previsão fracos, geralmente árvores de decisão.

Segundo a definição de Friedman [10]:

> O aprimoramento de gradiente constrói modelos de regressão aditiva ajustando sequencialmente uma função parametrizada simples (aprendizado básico) aos atuais "pseudo" -residuais por mínimos quadrados em cada iteração. Os pseudo-resíduos são o gradiente da perda funcional sendo minimizado, com relação aos valores do modelo em cada ponto de dados de treinamento avaliado na etapa atual. Mostra-se que tanto a precisão da aproximação quanto a velocidade de execução do aumento de gradiente podem ser substancialmente melhoradas pela incorporação de randomização no procedimento. Especificamente, em cada iteração, uma subamostra dos dados de treinamento é desenhada aleatoriamente (sem substituição) do conjunto completo de dados de treinamento. Essa subamostra selecionada aleatoriamente é então usada no lugar da amostra completa para se ajustar ao aluno básico e calcular a atualização do modelo para a iteração atual. Esta abordagem aleatória também aumenta a robustez contra o excesso de capacidade do aprendente de base.

1.7. Auto-sklearn

Com base no pacote de aprendizado de máquina em Python, scikit-learn [11], o Auto-sklearn [12] é um sistema AutoML para criação de pipelines de classificação e regressão. É composto por 15 classificadores, 14 recurso de métodos de pré-processamento e 4 métodos de preparação de dados da biblioteca scikit-learn [13].

O Auto-sklearn usa o meta-aprendizado para iniciar a otimização bayesiana de forma mais eficiente, pois ele leva em conta automaticamente o desempenho passado em conjuntos de dados semelhantes. Este meta-aprendizado é alimentado por uma extensa avaliação de diferentes pipelines em 140 conjuntos de dados distintos. A base de meta-aprendizado não é atualizada quando novos pipelines e conjuntos de dados são avaliados.

Nesta ferramenta, o sistema define uma estrutura fixa para os pipelines. Inicialmente, há uma etapa de preparação de dados que inclui um conjunto de métodos como imputação e codificação categórica. Posteriormente, um algoritmo opcional pré-processamento e um modelo obrigatório são selecionados e otimizados via SMAC. Além disso, o Auto-sklearn aplica o aprendizado conjunto para tornar o resultado mais robusto contra overfitting. O sistema armazena os modelos avaliados no espaço de busca e, em seguida, aplica a seleção de conjunto [14], um método de pós-processamento de modelo que inicia de um conjunto vazio e através de um procedimento ganancioso, adiciona iterativamente o modelo que maximiza o desempenho da validação de conjunto , portanto, construindo um conjunto dos modelos treinados anteriores.

1.8. Otimização de hyperparametros

Para a realização da otimização de hiperparâmetros dos classificadores descritos anteriormente, com exceção da ferramenta de AutoML Auto-sklearn, foi utilizado o algoritmo Random Search. Como o nome sugere, os hiperparâmetros dos classificadores são selecionados aleatoriamente até que um critério de parada seja satisfeito.

2. Avaliação experimental dos classificadores

O conjunto de dados representa diversos jogadores e seus respectivos atributos relacionados. Atributos que em sua grande maioria representam valores numéricos contínuos, como visto na Tabela 1 .

Os dados foram divididos em treinamento/dev e teste. Os dados de treinamento/dev representam 80% do conjunto de dados e foi selecionado utilizando o trantestsplit, método de mode selection da scikit-learn. O

Table 1. Descrição do dataset FIFA 2019

Atributos	Float types	Int Types	Nº de classes
42	37	5	3

treinamento/dev foi dividido usando cross validation com 10 folds, também da model selection. A semente de seleção (randomstate) foi compartilhada entre todos os algoritmos para uma melhor comparação.

Os primeiros testes foram com os parâmetros padrões fornecidos pela biblioteca scikit-learn, com exceção do algoritmo KNN que foi utilizado com K=3 e dos algoritmos que utilizam árvores, utilizando profundidade máxima de 3 níveis, apenas para testes iniciais. A Tabela 2 mostra os resultados obtidos utilizando os 6 classificadores. Os valores de acurária e de tempo representam a média dos 10 folds no *cross validation*.

Table 2. Resultados iniciais FIFA 2019

Algoritmo	Acc	Acc std	Tempo	Acc teste	Acc std teste
Logistic Regression	0.878	0.001	0.680	0.875	0.010
SVM - Linear	0.8857	0.0013	4.379	0.881	0.008
KNN - k=3	0.911	0.0009	0.047	0.838	0.008
Decision Tree	0.762	0.002	0.089	0.759	0.006
Random Forest	0.789	0.005	0.105	0.785	0.009
Gradient Boosting	0.908	0.0007	6.915	0.883	0.006

O algoritmo KNN teve uma acurácia alta no conjunto de treinamento, porém, não teve o melhor resultado no conjunto de testes. O desvio padrão baixo dos classificadores no conjunto de teste mostra uma estabilidade entre os *folds* no *cross validation*. O algoritmo Gradient Boosting teve os melhores resultados tanto no treinamento quanto no teste com uma diferença bem pequena de acurácia. O algoritmo SVM com kernel linear teve o maior equilíbrio entre a acurácia do treinamento e do teste.

A árvore de decisão e o Random Forest tiveram resultados parecidos e bem abaixo dos demais algoritmos. Em parte, devido a limitação de 3 níveis de altura da árvore imposta nos experimentos iniciais.

O algoritmo Gradient Boosting, apesar de ter o melhor desempenho, está no limite para um *overfitting*. Um teste utilizando o parâmetro *learning rate* mostra bem esse limite (Figura 1) :

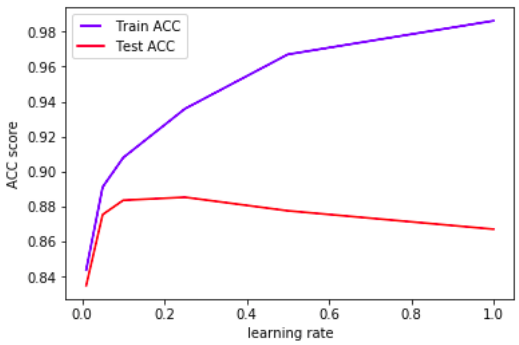

Fig. 1. Variação de precisão com parâmetro *learning rate*

Na segunda etapa de testes os algoritmos foram otimizados com a técnica Random Search, cada um com seu respectivo espaço de busca definido por seus hiperparametros. Os resultados estão descritos na Tabela 3.

Table 3. Resultados finais FIFA 2019

Algoritmo	Acc	Acc teste
Logistic Regression	0.878	0.884
SVM - Linear	0.8853	0.8960
KNN - k=25	0.911	0.838
Decision Tree	0.937	0.832
Random Forest	0.874	0.876
Gradient Boosting	0.823	0.829

A otimização dos algoritmos favoreceu principalmente os algoritmos baseado em árvores. O parâmetro da altura máxima da árvore foi crucial para este ganho de acurácia, como visto na Figura 2.

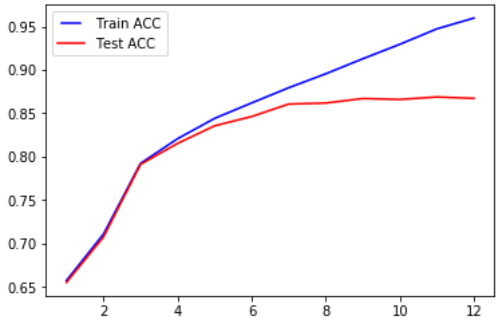

Fig. 2. Variação de acurácia com parâmetro *max depth* no algoritmo Random Forest

O algoritmo KNN também foi fortemente favorecido com a otimização. Encontrou-se um k=25 que obtêm acurácia de 91% no conjunto de treinamento e 83% no conjunto de teste.

O Svm com kernel linear e a regreção logistica tiveram desempenhos similares, ambos na casa de 88% de acurácia no treinamento e 89% no teste. Não muito diferente dos resultados já obtidos sem otimização.

A ferramenta de automl foi testada nas mesmas configurações dos algoritmos de classificação. Um *cross validation* no conjunto de treinamento/dev e a validação no conjunto de testes. Os resultados estão ilustrados na Tabela 4.

Table 4. Resultados automl FIFA 2019

AutoML	Acc	Acc teste
Autosklearn	1	0.892

O automl AutoSklearn conseguiu incríveis 100% de acurácia no conjunto de treinamento, o que pode indicar um overfitting. O mesmo conseguiu 89% nos dados de teste.

3. Conclusões

É notável que cada algoritmo possui diferentes propriedades em relação a uns aos outros, deste modo, não existe um único e melhor algoritmo para todos os casos de uma tarefa de classificação. Isso é justificado

através do teorema "Não há almoço grátis", em que a grosso modo, afirma que, em média, sobre todos os problemas possíveis (com uma probabilidade a priori uniforme no espaço de problemas), o desempenho de todos os classificadores é o mesmo.

Neste trabalho testamos 6 algoritmos de classificação e uma ferramenta de AutoML no dataset do FIFA 19. Um problema com 3 classes e diversos atributos contínuos. Os algoritmos mostraram-se eficientes, com boas acurácias no conjunto de teste definido. O baixo desvio padrão obtido no *cross validation* destes algoritmos denota uma boa estabilidade dos modelos gerados, viabilizando-os para um possível ambiente de produção. O algoritmo SVM com kernel linear e otimizado foi o que gerou o modelo com maior acurácia no conjunto de teste.

References

[1] A. Urso, A. Fiannaca, M. La Rosa, V. Ravì, R. Rizzo, Data mining: Prediction methods, Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics (2018) 413.
[2] R. Gove, J. Faytong, Machine learning and event-based software testing: Classifiers for identifying infeasible gui event sequences, in: Advances in Computers, Vol. 86, Elsevier, 2012, pp. 109–135.
[3] R. C. Neath, M. S. Johnson, Discrimination and classification.
[4] J. C. Tong, S. Ranganathan, Computer-aided vaccine design, Elsevier, 2013.
[5] M. Kounelakis, M. Zervakis, X. Kotsiakis, The impact of microarray technology in brain cancer, in: Outcome Prediction in Cancer, Elsevier, 2007, pp. 339–388.
[6] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and regression trees. wadsworth int, Group 37 (15) (1984) 237–251.
[7] J. R. QUINLAN, Program for machine learning, C4.5.
URL `https://ci.nii.ac.jp/naid/10015645285/en/`
[8] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.
[9] Y. Freund, R. Schapire, Machine learning: Proceedings of the thirteenth international conference, San Francisco, CA: Morgan Kauffman (1996) 148–156.
[10] J. Friedman, Stochastic gradient boosting, Computational Statistics Data Analysis 38 (2002) 367–378. doi:10.1016/S0167-9473(01)00065-2.
[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.
[12] M. Feurer, F. Hutter, Towards further automation in automl, in: ICML AutoML workshop, 2018.
[13] F. Hutter, L. Kotthoff, J. Vanschoren, Automatic machine learning: methods, systems, challenges, Springer, 2019.
[14] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 18.