Análise de métodos de aprendizado supervisionado em problemas de classificação

Daniel de Almeida Duque[1]

Jardim Camburi, Vitória-ES. Brasil

Abstract

O artigo visa comparar cinco métodos de aprendizado de máquina supervisionado - ZeroR (ZR), Naive Bayes Gaussiano (NBG), KMeans Centroides (KMC), K Vizinhos Mais Próximos (KNN) e Árvore de Decisão (AD) - para a classificação da base de dados *wine* utilizando a acurácia como *score* para as comparações. E, com isso, verificar a diferença visual dos classificadores, utilizando uma tabela de acurácia e *boxplots* com os scores de cada método e a diferença estatística comparando cada método par a par utilizando o teste t pareado e o *wilcoxon*.

Keywords: `aprendizado de máquina, aprendizado supervisionado, classificação, wine, acurácia.`

1. Introdução

A classificação é um dos problemas, na área de inteligência artificial, que utiliza métodos de aprendizado supervisionado para prever as classes de determinados dados já rotulados. É evidente a importância dos problemas de classificação e, por isso, o artigo realiza uma comparação experimental, dividida em duas etapas, visando analisar o desempenho (utilizando a acurácia como métrica) dos métodos de aprendizado supervisionado: ZeroR (ZR), Naive Bayes Gaussiano (NBG), KMeans Centroides (KMC), K Vizinhos Mais Próximos (KNN) e Árvore

[1] Aluno de Engenharia de Computação da Universidade Federal do Espírito Santo (UFES)

de Decisão (AD) na predição de classes da base de dados *wine* disponibilizada pela biblioteca *scikit-learn* do *Python*.

O experimento foi dividido em duas etapas, utilizando os classificadores ZR e NBG, sem hiperparâmetros e com hiperparâmetros (KMC, KNN e AD). A primeira etapa foi realizada com 3 rodadas de treino e teste com validação estratificada de 10 *folds* e a segunda etapa foi realizada com 3 rodadas de ciclos aninhados de validação e teste utilizando 10 *folds* para teste externo e 4 *folds* para a validação interna.

De acordo com o experimento, os resultados foram analisados utilizando: uma tabela com a informação da acurária dos métodos para os testes, gráficos *boxplots* da acurácia de cada método para comparar os métodos de forma visual e uma tabela pareada com testes de hipótese (t-test pareado e *wilcoxon*) comparando cada método par a par.

2. Base de Dados

2.1. Descrição do Domínio

O experimento foi conduzido utilizando a base de dados *wine* disponibilizada pela biblioteca *scikit-learn*, do *Python*, que possui diversas ferramentas úteis e acessíveis para aprendizado de máquina. Essa base é muito conhecida e os dados desse *dataset* permitem distinguir 3 classes diferentes. Existem 178 amostras ao todo e cada amostra possui 13 características que permitem definir a classe.

2.2. Definição das Classes e das Características

Nessa base de dados, os atributos que definem uma classe são: *alcohol, malic acid, ash, alcalinity of ash, magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines e proline*. Com isso, são definidas 3 classes: *class_0, class_1, class_2*.

2.3. Número de Instâncias

O *wine dataset* possui 178 instâncias com 59 em *class_0*, 71 em *class_1* e 48 em *class_2*, ou seja, é uma base de dados razoavelmente balanceada. Com base

nisso, foi realizada uma análise visual das características que definem as classes no *wine dataset*:

Figura 1 - Informação das características

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols
count	178.00	178.00	178.00	178.00	178.00	178.00
mean	13.00	2.34	2.37	19.49	99.74	2.30
std	0.81	1.12	0.27	3.34	14.28	0.63
min	11.03	0.74	1.36	10.60	70.00	0.98
25%	12.36	1.60	2.21	17.20	88.00	1.74
50%	13.05	1.87	2.36	19.50	98.00	2.35
75%	13.68	3.08	2.56	21.50	107.00	2.80
max	14.83	5.80	3.23	30.00	162.00	3.88

	flavanoids	nonflav_phenols	proanthocyanins	color_intensity
count	178.00	178.00	178.00	178.00
mean	2.03	0.36	1.59	5.06
std	1.00	0.12	0.57	2.32
min	0.34	0.13	0.41	1.28
25%	1.21	0.27	1.25	3.22
50%	2.13	0.34	1.56	4.69
75%	2.88	0.44	1.95	6.20
max	5.08	0.66	3.58	13.00

	hue	od280/od315	proline	target
count	178.00	178.00	178.00	178.00
mean	0.96	2.61	746.89	0.94
std	0.23	0.71	314.91	0.78
min	0.48	1.27	278.00	0.00
25%	0.78	1.94	500.50	0.00
50%	0.96	2.78	673.50	1.00
75%	1.12	3.17	985.00	2.00
max	1.71	4.00	1680.00	2.00

Verificando a Figura 1, pode-se ver dados quantitativos de cada característica do banco de dados *wine*. Esse tipo de análise antes do desenvolvimento é importante, pois é possível verificar se as características tem grandezas muito diferentes entre si, como, por exemplo, *proline* com valores 1000 vezes maior comparado ao *hue* e, por isso, é necessário realizar uma padronização dos dados antes de executar o classificador.

3. O Método KMC

O *KMeans* Centroides (KMC) é um método de classificação que usa o *KMeans*, da biblioteca *scikit-learn*, como método de agrupamento. O KMC cria K (hiperparâmetro) grupos com os dados da base de treino para cada classe. O passo a passo da implementação desse método está descrito a seguir:

1. Forma $K * ncl$ grupos com a base de treino
2. Calcula o centroide de cada grupo
3. Associa o centroide à classe do grupo a partir do qual foi gerado
4. Associa a classe, do centroide mais próximo, ao elemento que se quer classificar

No desenvolvimento do código foi criada a classe *KMCClassifier* que utiliza um *BaseEstimator* como parâmetro e o *KMeans* é passado como parâmetro para criar os K grupos. As funções, em *Python*, para realizar os passos (os números entre colchetes são os passos, descritos anteriormente, em que a função está relacionada):

Função ***fit()*** do KMC:

1. *x_train, y_train = check_X_y(x_train, y_train)*
2. *kmc_init = kmc_init_k_ncl_groups(x_train, y_train, KMeans)* - [1]
3. *class_centroids = calculate_centroid_each_group(kmc_init)* - [2,3]
4. *self.__self_pred* = class_centroids

Função ***predict()*** do KMC:

1. *y_pred = class_of_closer_centroid_each_element(x_test, self.__self_pred)* [4]
2. *return (y_pred)*

4. Descrição dos Experimentos Realizados e seus Resultados

Os experimentos realizados foram divididos em duas etapas, descritas a seguir: A primeira etapa utiliza os classificadores ZR e NBG, ambos sem hiperparâmetros, com 3 rodadas de validação cruzada estratificada de 10 *folds* para o treino e teste dos dados. A segunda etapa do experimento utiliza os classificadores KMC, KNN e AD, todos com hiperparâmetros predefinidos, com 3 rodadas de ciclos aninhados de validação e teste, contendo 4 *folds* no ciclo interno de validação e 10 *folds* no ciclo externo de teste. Nessa etapa utilizou-se a busca em grade (*grid search*) para testes com diferentes valores de hiperparâmetros, figura 2. Em ambos os testes, o conjunto de treino foi normalizado utilizando o *z-score* antes da classificação.

Figura 2 - Tabela de hiperparâmetros da busca em grade

Grid Search	
Classificador	**Hiperparâmetro**
KMC	k = [1, 3, 5, 7]
KNN	n_neighbors = [1, 3, 5, 7]
AD	max_depth = [None, 3, 5, 10]

Os resultados das classificações foram analisados utilizando os *scores* dos métodos nos 10 *folds* a métrica utilizada no *score* foi a acurácia. Uma tabela de acurácia foi definida calculando a média, desvio padrão, limite inferior e limite superior dos *scores* dos métodos de classificação, com intervalo de confiança de 95% (figura 3). Uma tabela pareada dos testes de hipótese com o teste t-pareado na matriz triangular superior e o teste não paramétrico de *wilcoxon* na matriz triangular inferior (figura 4). Os gráficos *boxplots* dos *scores* de cada classificador (figura 5).

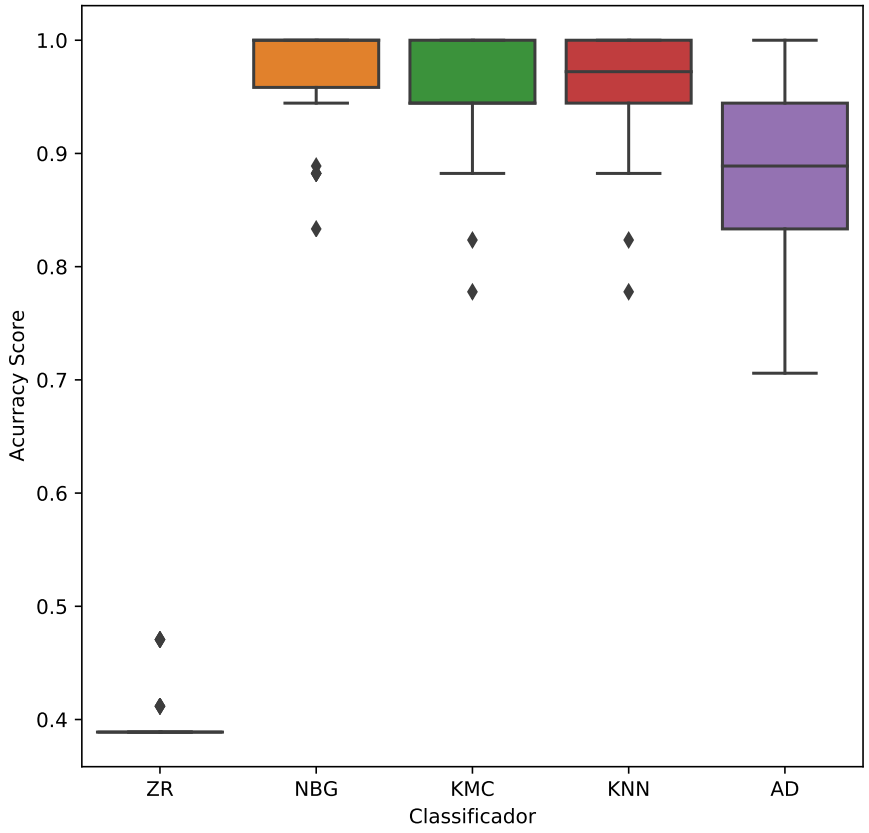
Figura 3 - Tabela de acurácia

	media	desvio_padrao	limite_inferior	limite_superior
ZR	0.40	0.02	0.39	0.41
NBG	0.97	0.05	0.96	0.99
KMC	0.95	0.05	0.93	0.97
KNN	0.96	0.05	0.94	0.98
AD	0.89	0.08	0.86	0.92

Figura 4 - Tabela pareada dos testes de hipótese

	ZR	NBG	KMC	KNN	AD
ZR	ZR	**8.057e-07**	**1.267e-06**	**1.249e-06**	**1.565e-06**
NBG	**1.143e-29**	NBG	**0.029**	0.115	**1.546e-05**
KMC	**9.019e-29**	**0.010**	KMC	0.248	**0.000**
KNN	**1.130e-28**	0.053	0.255	KNN	**3.385e-05**
AD	**1.996e-24**	**1.766e-08**	**1.307e-05**	**2.145e-07**	AD

Figura 5 - *Boxplots* das acurácias



5. Conclusões

5.1. Análise geral dos resultados

A partir das tabelas e dos *boxplots* gerados é possível perceber algumas informações sobre os resultados das classificações. A tabela (figura 3) permite uma visão geral da acurácia de cada classificador que, conforme era esperado, o classificador ZeroR (ZR) acertou apenas 40% das predições, porque ele utiliza a classe majoritária para classificar e, como a base de dados *wine* tem uma classe com 40% dos dados nela, então esse classificador teve um desempenho ruim. Enquanto que, em média, os outros 4 classificadores tiveram resultados muito próximos, com quase nenhuma diferença entre si.

Utilizando os *boxplots* (figura 5) é possível tirar a mesma conclusão, porém, percebe-se que o classificador Naive Bayes Gaussiano (NBG) obteve resultados mais consistentes, ou seja, com menor variação comparado aos outros classificadores. Além disso, percebe-se que a Árvore de Decisão (AD) não obteve resultados tão consistentes quanto os outros.

Por fim, a tabela pareada (figura 4) permite observar que houve diferença estatística nítida entre o classificador ZeroR e os outros 4 classificadores. A Árvore de Decisão também teve diferença estatística comparado aos outros classificadores, porém não tanto comparado ao ZeroR, enquanto que isso não ocorreu entre os outros classificadores, os outros tiveram *p-values* altos, ou seja, a diferença estatística insignificante e, com isso, mudar entre um classificador ou outro não mudaria o resultado da classificação.

5.2. Contribuições do Trabalho

Os classificadores ZR, NBG, KMC, KNN e AD foram comparados entre si utilizando o banco de dados *wine* e, com isso, foi possível perceber as diferenças de acurácia para o trabalho de aprendizado supervisionado para a classificação dessa base de dados. Outros trabalhos podem usar como referência adicionando outros classificadores ou até mesmo outras bases de dados para realizar comparações mais complexas.

5.3. Melhorias e trabalhos futuros

Devido a utilização de apenas um *dataset*, percebe-se que, em trabalhos futuros, é importante utilizar várias bases de dados diferentes pode torná-lo mais completo. Além disso, pode-se adicionar outros classificadores como *Random Forest* e *Suport Vector Machine* para comparar mais métodos de aprendizado. Por fim, pode-se usar mais valores durante o *Grid Search* para cobrir uma área maior de hiperparâmetros e, com isso, ampliar o experimento.

Referências Bibliográficas

- https://numpy.org/doc/stable/
- https://pandas.pydata.org/docs/
- https://seaborn.pydata.org/
- https://matplotlib.org/stable/index.html
- https://scikit-learn.org/stable/
- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html
- https://scikit-learn.org/stable/datasets/toy_dataset.html
- https://docs.scipy.org/doc/scipy/
- Slides das aulas de inteligência artificial em 2022