Projeto Aprendizado de Máquina

Carlos Renato de Andrade Figueiredo*, Gustavo Vieira Queiroz†,
Nickolas Batista Mendonça Machado‡ e Samara Ribeiro Silva§
Instituto Tecnológico de Aeronáutica - ITA
São José dos Campos, Brasil
Email: *carlos.figueiredo@ga.ita.br, †gustavo.queiroz@ga.ita.br, ‡nickolas.machado@ga.ita.br, §samara.silva@ga.ita.br,

I. Objetivo do Trabalho e Descrição da Implementação

Nessa sessão, tratar-se-á sobre os objetivos do trabalho, as ferramentas utilizadas, detalhes da execução e a revisão teórica sobre aprendizado de máquina utilizando árvores de decisão.

A. Objetivos

O objetivo do trabalho é por em prática o conteúdo abordado durante as aulas da disciplina CTC-17, de inteligência artificial, sobre aprendizado de máquina utilizando árvores de decisão através de programação de algoritmos clássicos.

B. Ferramentas e execução

A linguagem escolhida para o trabalho foi a linguagem Python, com notebooks interativos. A IDE utilizada pelo grupo foi o Visual Studio Code. O código foi compartilhado via ferramenta de versionamento git e está disponível em um repositório público no github[1]. Para executar o projeto é suficiente executar as células dos notebooks Python na ordem.

C. Árvore de Decisão

Árvore de decisão é um algoritmo de aprendizado de máquina supervisionado amplamente utilizado, visto que possui estrutura de fácil compreensão e apresenta bons resultados. Uma árvore de decisão recebe como entrada um conjunto de atributos e retorna um único valor, uma ”decisão”.

A ”decisão”é alcançada através de uma série de testes e divisão do problema para execução recursiva. Os nós da árvores (não folhas) representam os testes, as arestas (ramos) representam os resultados dos testes e as folhas a decisão ou classe. A escolha da ordem dos atributos testados influenciam na qualidade da decisão apresentada. Um atributo ideal divide os dados em conjuntos, um totalmente positivo e o outro totalmente negativo [2].

Para definir qual atributo deve ser testado primeiro, é necessário uma métrica para medir a ”importância”de cada um para o resultado. Essa ”importância”é medida através da função ganho (equação 2) que utiliza o conceito de entropia (equação 1) no seu cálculo.

A entropia de um conjunto é a medida do grau de desordem desse conjunto. Se todos os elementos de uma mesma classe apresentarem um único tipo de valor (todos positivos ou todos negativos), a entropia é nula. Já se uma classe apresentar a mesma quantidade de elementos positivos e negativos a entropia tem valor unitário. Ou seja, a diminuição da entropia significa um ganho de informação, visto que um atributo ideal possui entropia igual a zero.

$$H(V) = -\sum_{k} P(v_k) log_2 P(v_k) \tag{1}$$

O ganho de informação do atributo A, relativo ao conjunto S, é dado pela equação 2, que mede a redução esperada na entropia.

$$Ganho(S, A) = H(S) - \sum (\frac{|S_v|}{|S|}) H(S) \tag{2}$$

Onde $S_v = \{s \in S \mid A(s) = v\}$ e $|S_v|$ representa o número de elementos de S_v.

Alguns tipos de atributos do conjunto de entrada podem dificultar a tomada de decisão. E por isso faz-se necessário um pré-processamento dos dados.

A omissão de dados em um ou mais atributos no conjunto de exemplos deve ser evitada, pois requer uma regra especial para a classificação quando o dado for inexistente e requer, também, uma adaptação da fórmula do ganho de informação. Por vezes, descartar atributos que apresentam omissão de dados é a melhor opção para obter resultados satisfatórios. Já para atributos multivalorados, ou seja quando existem muitos valores possíveis para o atributo, a medida do ganho pode não representar a corretamente a ”importância”desse atributo. Um exemplo de atributo multivalorado é a data, que tem valor único para cada exemplo. Nesses casos, uma boa saída pode ser a categorização desse atributo, como a separação por dias da semana, meses, trimestre, estações do ano etc.

Atributos de entrada com valores contínuos e inteiros também necessitam de um pré processamento. Assim como nos atributos multivalorados, a divisão em categorias pode ser uma solução, por exemplo, a separação em faixas etária de um atributo de idade.

Para analisar os resultados da árvore de decisão pode-se utilizar alguns indicadores como: a taxa de acerto (acurácia), o erro quadrático médio e a estatística *kappa*.

A taxa de acerto pode ser calculado utilizando a equação 3.

$$Acurácia = \frac{Acertos}{Total\ de\ elementos} \tag{3}$$

A taxa de acerto indica a percentagem em que o resultado fornecido pela árvore de decisão foram corretos.

[1]Repositório disponível em: https://github.com/montanha22/ctc17

O erro quadrático médio pode ser calculado utilizando a equação 4.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_{observado} - Y_{previstos})^2 \tag{4}$$

Em geral, menores valores de MSE significam menores valores de variância. Um alto valor de variância pode indicar um superajuste (*overfitting*) do algoritmo ao conjunto de treinamento, o que não é desejado, pois um algoritmo com *overfitting* perde a generalidade e por sua vez obtém bons resultados apenas para o conjunto de treinamento.

O coeficiente *kappa* pode ser calculado utilizando a equação 5.

$$k = \frac{p_o - p_e}{1 - p_e} \tag{5}$$

Onde p_o representa a acurácia da concordância observada e p_e é a estimativa de concordância casual (preditor aleatório).

O coeficiente *kappa* varia entre 0 e 1, sendo o valor 1 indica perfeita acurácia e 0 significa que a classificação não é melhor que uma classificação aleatória.

II. Resultados Obtidos

Nesta seção tratar-se-á sobre a implementação das soluções para os problemas propostos. Todas as soluções foram implementadas utilizando a linguagem de programação Python.

A. Classificador baseado em árvore de decisão

Foi realizado um pré processamento dos dados de entrada categorizando o atributo ”data”em dias da semana.

A implementação da árvore de decisão seguiu os seguintes passos:

1) verifica-se se o atributo analisado é um atributo ideal e, caso seja, o nó é definido como folha e é retornado com o *label* igual ao atributo;
2) Se restar apenas um atributo na lista de atributos, o nó é definido como folha e é retornado com o *label* igual à classe mais frequente desse atributo;
3) calcula-se o melhor atributo a partir do ganho de informação;
4) remove-se esse atributo da lista de atributos; e
5) executa-se os itens anteriores recursivamente;

O conjunto de treinamento utilizado foi composto por 80% do conjunto de dados, conforme solicitado em [1]. Esse conjunto foi definido utilizando a biblioteca *scikit-learn* do Python. Essa biblioteca, também, foi utilizada para o cálculo da matriz de confusão, MSE e coeficiente de *kappa*.

A matriz de confusão obtida pelo classificador obtido pode ser observada na figura 1 e os valores das métricas de análise estão disponíveis na tabela I.

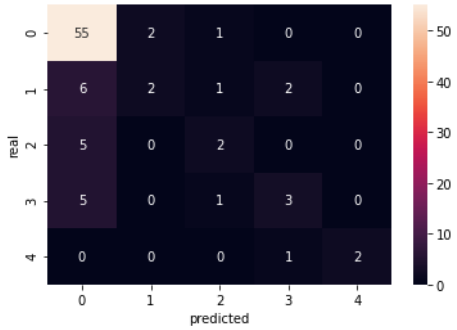

Figura 1. Classificação baseada em árvore de decisão

B. Classificador a priori

O classificador *a priori* foi implementado utilizando como critério de classificação a moda, ou seja, retorna classe mais comum na predição.

A matriz de confusão obtida pelo classificador obtido pode ser observada na figura 2 e os valores das métricas de análise estão disponíveis na tabela I.

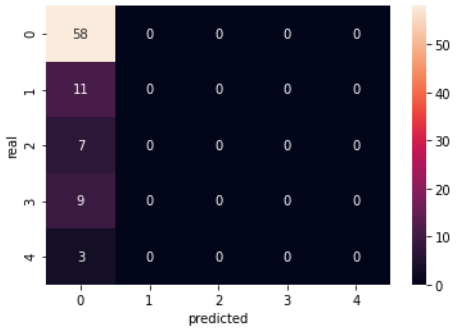

Figura 2. Classificação a priori

C. Análise Comparativa

Tabela I
Análise comparativa

	Árvore de Decisão	*a priori*
Taxa de acerto	0.7273	0.6591
Erro quadrático médio	1.0	1.9091
Estatística kappa	0.3943	0.0

Na tabela I pode-se observar as métricas de desempenho dos classificadores implementados. A árvore de decisão implementada apresenta um melhor resultados em todos os indicadores. A taxa de acerto da árvore de decisão é cerca de 8% maior que a acurácia do classificador *a priori*. Além disso, possui um MSE menor e o coeficiente *kappa* não nulo.

Pelos dados apresentados pode-se concluir que a classificação da árvore de decisão implementada para o conjunto de dados analisado é razoável ($0.2 < k < 0.4$) e apresenta resultado superior ao resultado do classificador *a priori*.

D. Possíveis melhorias para o classificador

Segundo [3], o algoritmo ID3 tem a desvantagem que, por utilizar-se do ganho de informação para determinação do atributo ideal, ele é inclinado a escolher o atributo com maior quantidade de valores diferentes. Desse modo [4] propôs a alteração da função de ganho de informação, para a equação 6.

$$Ganho(S, A) = \frac{2pn}{p+n} - \sum_{i=1}^{V} \frac{2p_i n_i}{p_i + n_i} \tag{6}$$

Onde p e n são, respectivamente a quantidade de exemplos positivos e negativos no conjunto S e p_i e n_i são a quantidade de exemplos positivos e negativos para cada um dos V valores distintos do atributo A.

[4] concluiu que a mudança na função de ganho resultou em um algoritmo com menor tempo de execução e que, apesar de para datasets menores obter acerto menor que o algoritmo ID3 convencional, para dataset maiores apresentou taxa de acerto maior.

Ademais, é possível melhorar o resultado do classificador utilizando *Random Forest* em vez de Árvore de decisão, que, segundo [5], se baseia na utilização de diversas árvores de decisão, que são treinadas com subespaços aleatórios do conjunto de atributos. Desse modo, haverá redução da limitação de complexidade intrínseca ao algoritmo, que, por exemplo, tem preferência por árvores pequenas. Desse modo, a taxa de acerto obtida será, no geral, melhor do que para o algoritmo de Árvore de decisão.

III. Conclusões

Este trabalho serviu para que os alunos pudessem exercitar e fixar os conhecimentos sobre aprendizado de máquina utilizando árvore de decisão para classificação de dados de uma base diversa e que necessita de pré-processamento. Além disso, foi realizada a classificação *a priori* desses dados e uma análise comparativa entre esses dois métodos de classificação. Vale ressaltar que os problemas propostos apresentaram nível de dificuldade compatíveis com os objetivos do trabalho.

O resultado do classificador baseado em árvore de decisão apresentou um melhor desempenho do que o classificador *a priori*, com uma acurácia cerca de 8% maior. Apesar da árvore de decisão apresentar melhor desempenho, existem outras maneiras que podem aperfeiçoar ainda mais o algoritmo, como a mudança da função ganho ou transformá-lo num classificador Random Forest.

Referências

[1] Castro, P. Roteiro de laboratório de CTC-17: Projeto Aprendizado de Máquina. 2021.

[2] Norvig, P. Russel, and S. Artificial Intelligence. A modern approach. Upper Saddle River, NJ, USA:: Prentice Hall, 2002.

[3] JIN, Chen; DE-LIN, Luo; FEN-XIANG, Mu. An improved ID3 decision tree algorithm. In: 2009 4th International Conference on Computer Science & Education. IEEE, 2009. p. 127-130.

[4] YI-BIN, Li; YING-YING, Wang; XUE-WEN, Rong. Improvement of ID3 algorithm based on simplified information entropy and coordination degree. In: 2017 Chinese Automation Congress (CAC). IEEE, 2017. p. 1526-1530.

[5] HO, Tin Kam. Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. IEEE, 1995. p. 278-282.