Escolha de Ciências e Tecnologia
Departamento de Informática
Licenciatura em Engenharia Informática
Unidade Curricular Aprendizagem Automática
Ano letivo 2020/2021

Relatório do Trabalho 1

Docentes:
Professorar Teresa Gonçalves
Professor Luís Rato

Discente
José Santos nº 43017

Évora, dezembro de 2020

Introdução

No âmbito da unidade curricular de Aprendizagem Automática lecionada pela Professora Teresa Gonçalves e pelo Professor Luís Rato, foi solicitado, como primeiro trabalho, que os alunos implementassem uma árvore de decisão com diversos critérios de pureza e opção de pruning.

Para este fim, foram criadas duas classes, Node e DTree, constituídas por métodos e atributos auxiliares à resolução deste problema, e construiu-se, ainda, a classe MyDecisionTreeREPrune formada pelos métodos, fit, score, poda, split, calcImp, actYconsoanteDadosNo, poda_aux, poda_um_no whichClass.

O calcImp recebe um nó, calcula a sua impureza, consoante o critério escolhido, e devolve-a.

O actYconsoanteDadosNo recebe um nó, a Data e o arrY do pai, que contem todos os exemplos e a que classes pertencem respetivamente. Este método vai criar o arrY do nó, consoante os seus dados.

O poda_aux é um método que, dada uma raiz e uma lista de nós, coloca nessa lista, todos os nós que podem fazer poda, e devolve-a

O poda_um_no que dado um uma raiz e um nó, percorre a árvore, até encontrar o nó, faz a poda, e devolve a árvore.

O whichClass recebe um exemplo e, percorrendo a árvore, devolve a que classe pertence.

Programa

Sem entrar em muito detalhe, no início os dados são divididos em treino e teste, é chamado o método fit(), com a indicação do critério, e se vai ser realizado prune.

Dentro do fit, se o prune for True, os dados de treino são divididos em treino e poda. É criada a árvore usando o split(), este faz o split da árvore até que esta esteja completa. Com a árvore já criada, se for necessário realizar a poda, chama-se o método poda() que devolve a árvore podada com mais exatidão, tendo em conta os dados de poda.

Assim, o método fit devolve a árvore que criou, sendo esta utilizada para o score, onde se utiliza os dados de teste para percorrer a árvore, e, compara-se se a classe a que estes pertencem é igual à classe calculada pela árvore.

Por fim, tem-se o valor da exatidão, tendo em conta a percentagem de exemplos corretamente classificados.

Resultados

Após correr o programa, com todos os conjuntos de dados fornecidos, com random_state = 0, tanto para o split de treino e teste, como o de treino e prune e usando todas as combinações possíveis, de pruning e critério, obteve-se os seguintes resultados.

	Sem pruning			Com pruning		
	entropia	**gini**	**erro**	**entropia**	**gini**	**erro**
weather.nominal	100%	100%	100%	75%	75%	75%
contact-lenses	66.67%	66.67%	83.33%	83.33%	83.33%	100%
vote	93.10%	93.10%	94.83%	94.83%	94.83%	94.83%
soybean	86.52%	80.14%	83.69%	80.14%	73.05%	73.05%

Conclusão

Ao longo do trabalho, tentei ao máximo colocar comentários no meu código que fossem o mais específico possível para conseguir explicar o meu pensamento, o que fiz, e a necessidade que encontrei em usar certas estruturas para guardar valores.

Face aos valores de exatidão obtidos com o conjunto weather.nominal, achei que estavam dentro das expetativas, visto conter poucos exemplos. Pelo contrário estranho os valores do contact-lenses, uma vez que esperava obter uma exatidão superior sem utilizar pruning, creio que possa ser um caso de overfitting. Relativamente ao vote tem tudo dentro do esperado. Por fim no soybean, pensei que os valores com prune fossem superiores.

Em suma, surgiram algumas dificuldades, maioritariamente por não ter utilizado arrays numpy, no sentido em que o código possa estar um pouco confuso devido a existirem tantos for's, e consequentemente algum impacto na complexidade temporal. Apesar disto revelou-se um trabalho fundamental ao meu futuro, tendo gostado imenso de o realizar.