Classificação dos alunos da disciplina Aprendizado de Máquina usando Decision Tree

**Diogo Akio Balboni Miyake, Luiz Gabriel Correia,
Marcelo de Souza Pena, Thais Reis Alves**

21048813, 21023314, 11039314, 11094313

1. Introdução

O problema de classificação abordado neste projeto foi prever a aprovação ou reprovação de um estudante na disciplina Aprendizado de Máquina com base no conceito obtido em outras disciplinas relacionadas, número de faltas, o coeficiente de rendimento, número de disciplinas trancadas, número total de conceitos O's e a quantidade de horas por dia dedicadas ao estudo. A base de dados utilizada e treinamento do modelo foi feita usando bibliotecas *Python*, em particular a biblioteca de *Machine Learning scikitlearn*.

As disciplinas consideradas na base dados são: Algoritmos e Estruturas de Dados I (AED1), Processamento da Informação (PI), Inteligência Artificial (IA), Mineração de Dados (MD) e Álgebra Linear (AL). Em um *dataset* real, um bom modelo de classificação para este problema poderia identificar alunos com maior risco de reprovação e fornecer *insights* para melhorar o seu desempenho. Acreditamos que o melhor desempenho nas outras disciplinas, o número de faltas e o CR sejam determinantes na aprovação de um aluno e o número de trancamentos e reprovações por O sejam pouco relevantes.

2. Base de dados

O dataset tem duas classes: 1 (aprovado) e 0 (reprovado). O tamanho do exemplos gerados por classe serão 700 e 300, respectivamente, totalizando uma amostra de 1000. Para as horas de estudo utilizamos uma gaussiana com centro em 4.0 para os aprovados e com centro em 1.5 para os reprovados. Para o coeficiente de rendimento (CR), utilizamos uma gaussiana com centro em 2.4 para os aprovados e 1.6 para os reprovados. No número de faltas na disciplina foi gerado um número inteiro aleatório entre 0 e 2 para aprovados e entre 0 e 6 para reprovados. Para o número de reprovações por O e o número de trancamentos também foram gerados um números inteiros aleatórios, entre 0 e 1 conceitos O para aprovados, entre 0 e 4 conceitos O para reprovados, entre 0 e 4 disciplinas trancadas para aprovados e entre 2 e 6 disciplinas trancadas para reprovados.

Para a geração das notas, vamos gerar uma nota de 0 a 10 de forma logarítmica, que depende das horas de estudo com um ruído aleatório. O ruído serve para gerar os casos em que o aluno estuda muito e ainda assim reprova e o contrário também.

O *dataset* foi gerado usando as bibliotecas *Python Numpy* e *Pandas*.

3. Algoritmo Decision Trees

O método de aprendizado utilizado foi *Decision Trees*, um algoritmo de aprendizado supervisionado que utiliza um modelo obtido a partir de regras inferidas dos dados para classificar um conjunto de dados. [Scikit-learn 2018]

Como visto em aula, apesar de ter algumas desvantagens, das quais falaremos a seguir, as árvores de decisão são fáceis de entender e fáceis de gerar. Além disso, seu desempenho foi muito superior ao desempenho de outros dois algoritmos: *Naive Bayes* e *Knn*.

4. Treinamento e Desempenho do Modelo

O treinamento foi feito com o método *DecisionTreeClassifier* da biblioteca *scikitlearn* usando o parâmetro *gini* para medir a qualidade. Este parâmetro é calculado pela função um menos a somatória da proporção de acertos e erros ao quadrado. Por exemplo, em um *dataset* D, determinado atributo A pode ser *True* ou *False*, então é calculado o número de exemplos com o atributo A = True que foram classificados acertadamente dividido pelo total de exemplos ao quadrado, o número de exemplos com o atributo A = True que foram classificados errado dividido pelo total de exemplos ao quadrado e o **valor 1** será 1 menos esses dois valores. Da mesma forma o **valor 2** é calculado para o atributo A = False e o *Gini Index* será o número de exemplos com atributo A = True dividido pelo total, vezes o valor 1 somado ao número de exemplos com atributo A = False dividido pelo total, vezes o valor 2. Esse valor varia entre 0.0 (melhor caso) e 0.5 (pior caso).

Valor 1 (D, A = True) = 1 – (acertos / total true)^2 – (erros / total true)^2 = valor 1

Valor 2 (D, A = False) = 1 – (acertos / total false)^2 – (erros / total false)^2 = valor 2

Gini index = (total true / total) x valor 1 + (total false / total) x valor 2

O método *DecisionTreeClassifier* permite ainda controlar a profundidade máxima e a quantidade mínima de exemplos nos nós folha. Para otimizar esses parâmetros, avaliamos o classificador utilizando diferentes valores: profundidade máxima variando entre 1 e 6 e quantidade mínima de exemplos nos nós folha variando entre 1 e 30. O melhor desempenho obtido utilizou profundidade máxima 4 e no mínimo 6 exemplos nos nós folha. Limitar esses dois parâmetros ajudam a evitar os dois principais problemas das árvores de decisão: *overfitting* e árvores grandes demais.

Para fazer a avaliação utilizamos validação cruzada com o método *cross_val_score* e cv = 10, ou seja, dividindo a base de dados em dez partes, treinando com nove e testando com uma, fazendo isso dez vezes, sempre alterando a parte usada no teste. Cada um desses dez testes é avaliado e consideramos a média. Com os parâmetros utilizados conseguimos um desempenho de **0.885**.

5. Visualização da Árvore de decisão

Utilizando o método *export_graphviz* e o software *Graphviz* foi possível gerar uma imagem da árvore de decisão (que pode ser melhor vista *aqui*).

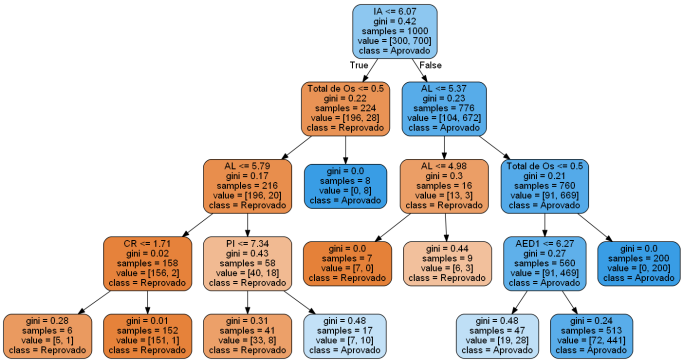

Figura 1. Decision Tree gerada

Na árvore é possível ver o valor do atributo escolhido para cada decisão, o valor do *Gini Index* (também representado por cores mais fortes nos melhores casos e mais fracas nos piores), o número de exemplos, o número de reprovados e aprovados e a classificação a cada nó se fosse feita a poda.

Alguns casos interessantes ocorrem, como alunos que tiraram 6,07 ou menos na disciplina Inteligência Artificial e que possuem uma ou mais reprovações por conceito O foram aprovados. É o caso de 8 exemplos.

6. Considerações Finais

A escolha desta base de dados se baseou no desejo de escolher um problema potencialmente real, que possa ter algum impacto no nosso meio universitário. Diante da impossibilidade de coletar dados reais para este projeto, decidimos gerar dados artificiais com base no mesmo problema e deixar em aberto a possibilidade de conseguir dados reais sobre os alunos e as disciplinas da computação na UFABC para um uso futuro com os modelos desenvolvidos.

Referências

Scikit-learn (2018). Naive bayes - scikit-learn v0.20.3 documentation. `https://scikit-learn.org/stable/modules/naive_bayes.html`.

SciPy (2018). Numpy and scipy documentation. `https://docs.scipy.org/doc/`.