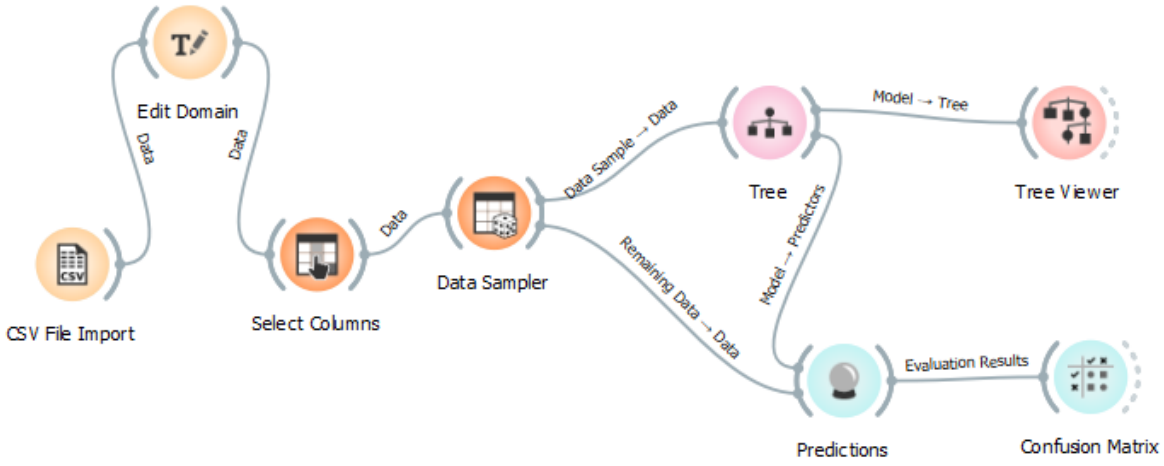
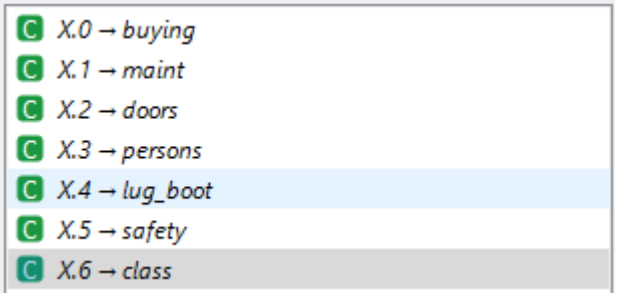
Disciplina: Introdução à Computação
Professor: Adriano Lorena Inacio de Oliveira
Título: Relatório - Projeto Inteligência Artificial
Grupo: Guilherme Siqueira, Walter Crasto, Ivo Neto, Pedro Fischer e Niltton Szpak

Neste trabalho, mostraremos o desenvolvimento da criação de um algoritmo de árvore de decisão utilizando o Orange Data Mining. Inicialmente, escolhemos o conteúdo que seria estudado pela nossa árvore, ou seja, buscamos um dataset que possuísse informações suficientes para o aprendizado da máquina. O dataset escolhido foi um sobre Avaliação de Carros, que possui 1.73k exemplos, de forma que dividimos 80% para o aprendizado do nosso algoritmo e 20% para testes e previsões.

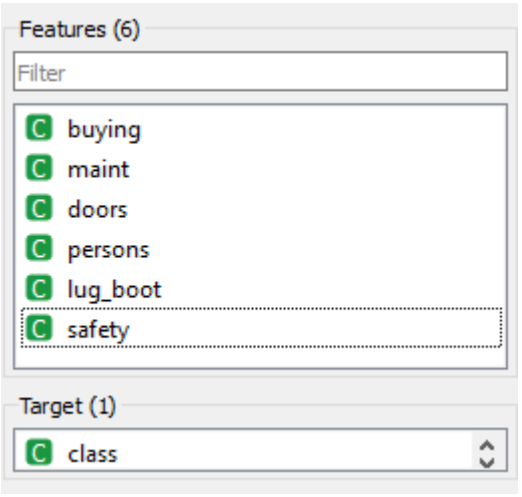


No Orange Data Mining, criamos esse esquema, que funciona da seguinte forma:

1) Primeiro, importamos o nosso dataset escolhido: https://archive.ics.uci.edu/dataset/19/car+evaluation
2) Em seguida, na aba de Edit Domain, realizamos os ajustes de nome nas variáveis do dataset, para a visualização ficar mais fácil depois, visto que, por padrão, as variáveis vem com nome: X.0, X.1, X.2, etc.



3) Como terceiro passo, partimos para a seção de Select Columns, em que escolhemos os atributos que serão utilizados como *features* e o *target* da árvore, ou seja, o que será usado como parte do estudo e a variável que será o objetivo do nosso estudo. Nesse caso, estamos em busca de prever a Classe que o carro se encontra: *unacceptable* (inaceitável), *acceptable* (aceitável), *good* (bom), *very good* (muito bom). E como ferramentas de avaliação usaremos: o preço do carro, o preço de manutenção, o número de portas, a capacidade de pessoas, o tamanho da mala e a segurança do carro.



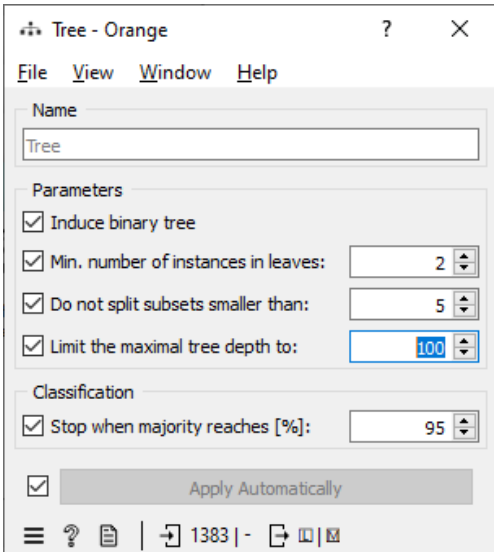
4) Em seguida, usando a aba do Data Sampler, dividimos a nossa amostra em Treinamento e Teste, fazendo uma divisão de 80% para treino e 20% para teste.

5) Agora, após o tratamento dos dados, podemos criar a nossa árvore, que pode ser observada usando o Tree Viewer:

Ou seja, temos uma árvore muito grande, vai tomando decisões em cada nó e chega na conclusão de que classe o carro se encaixa. Nessa árvore, temos

117 nós e 59 folhas, ou seja, existem 117 nós a serem percorridos e 59 “destinos finais” em que se existe uma conclusão.

Vale ressaltar que essa árvore foi criada utilizando os seguintes parâmetros:



Ou seja, no primeiro nós induzimos a criação de uma árvore que se divide de forma binária, gerando 2 nós-filhos de cada nó. Em seguida, temos o mínimo de casos que precisam ser colocados em cada folha, de forma que precisamos de no mínimo 2 casos da amostra de teste se encaixando naquela folha para a criação dela. Depois, temos a proibição da divisão de nós com menos de 5 casos em questão, ou seja, se houver menos de 5 casos, aquele ramo da árvore se encerra ali mesmo. E o último parâmetro é a limitação de níveis da árvore, em que nesse caso, colocamos 100, mas ela não chega atingir isso.

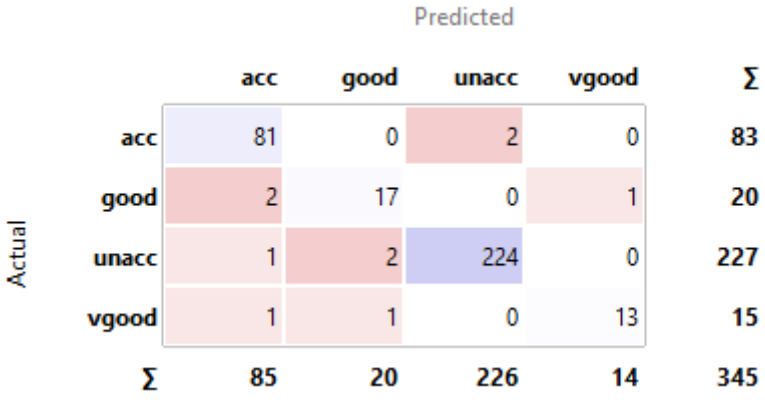
6) Depois da árvore de decisão montada, podemos testar o funcionamento dela na aba Predictions, em que ligamos nossa base de dados e nosso algoritmo e observamos o rendimento dele.

	Tree	error	class	buying	maint	doors	persons	lug_boot	safety
1	un...	0.000	unacc	low	low	2	2	small	high
2	acc	0.000	acc	vhigh	med	4	4	med	med
3	acc	0.000	acc	high	low	3	4	med	high
4	un...	0.000	unacc	low	high	2	2	med	low
5	un...	0.000	unacc	vhigh	med	2	4	small	low
6	un...	0.000	unacc	vhigh	vhigh	2	more	big	high
7	acc	1.000	good	low	med	2	more	big	med
8	un...	0.000	unacc	high	vhigh	2	2	med	high
9	un...	0.000	unacc	low	low	4	2	med	high
10	acc	0.000	acc	low	vhigh	5more	4	small	high
11	un...	0.000	unacc	vhigh	med	2	4	med	med
12	un...	0.000	unacc	vhigh	vhigh	3	4	small	med
13	acc	0.000	acc	high	med	5more	more	small	high
14	un...	0.000	unacc	med	high	4	more	small	med
15	un...	0.000	unacc	high	vhigh	3	4	small	med
16	un...	0.000	unacc	low	high	4	2	big	med
17	go...	0.000	good	low	low	4	more	med	med
18	un...	0.000	unacc	high	med	3	4	med	low
19	go...	0.000	good	low	med	4	more	med	med
20	un...	0.000	unacc	high	high	5more	more	small	low
21	acc	0.000	acc	low	high	4	more	big	med

Show perfomance scores Target class: (Average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
Tree	0.976	0.971	0.971	0.971	0.971	0.943

Em que nas informações da árvore, podemos observar que obtemos uma precisão de 0.971, ou seja, 97,1% de precisão na análise dos dados de teste. Para observar de maneira mais clara, podemos observar uma matriz de performance para avaliar o desempenho do nosso algoritmo:

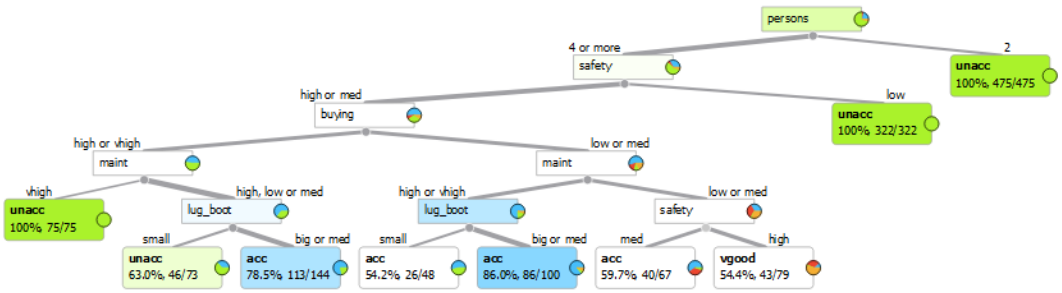


Em que temos:

- Dos 83 aceitáveis, acertamos 81
- Dos 20 bons, acertamos 17
- Dos 227 inaceitáveis, acertamos 224
- Dos 15 muito bons, acertamos 13

Em seguida, iremos alterar alguns valores dos parâmetros para observar o comportamento da árvore e como essas mudanças alteram o desempenho da mesma.

1) Vamos alterar o tamanho máximo da árvore. Na nossa configuração normal, usamos o tamanho máximo como 100, dando liberdade para a árvore crescer bastante. Acabou que nosso algoritmo ficou com 11 níveis e agora vamos alterar para 5 para ver o que ocorre com o desempenho da mesma.



Nesse caso, observamos que existem diversas folhas que não possuem conclusões certeiras sobre o veredito da classe que aquele automóvel se encontra, o que vai afetar na precisão de acerto da árvore.

	Tree	error	class	buying	maint	doors	persons	lug_boot	safety
1	un...	0.000	unacc	low	low	2	2	small	high
2	acc	0.215	acc	vhigh	med	4	4	med	med
3	acc	0.215	acc	high	low	3	4	med	high
4	un...	0.000	unacc	low	high	2	2	med	low
5	un...	0.000	unacc	vhigh	med	2	4	small	low
6	un...	0.000	unacc	vhigh	vhigh	2	more	big	high
7	acc	0.642	good	low	med	2	more	big	med
8	un...	0.000	unacc	high	vhigh	2	2	med	high
9	un...	0.000	unacc	low	low	4	2	med	high
10	acc	0.458	acc	low	vhigh	5more	4	small	high
11	acc	0.785	unacc	vhigh	med	2	4	med	med
12	un...	0.000	unacc	vhigh	vhigh	3	4	small	med
13	un...	0.630	acc	high	med	5more	more	small	high
14	acc	0.542	unacc	med	high	4	more	small	med
15	un...	0.000	unacc	high	vhigh	3	4	small	med
16	un...	0.000	unacc	low	high	4	2	big	med
17	acc	0.642	good	low	low	4	more	med	med
18	un...	0.000	unacc	high	med	3	4	med	low
19	acc	0.642	good	low	med	4	more	med	med
20	un...	0.000	unacc	high	high	5more	more	small	low
21	acc	0.140	acc	low	high	4	more	big	med

Show perfomance scores Target class: (Average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
Tree	0.944	0.817	0.802	0.803	0.817	0.660

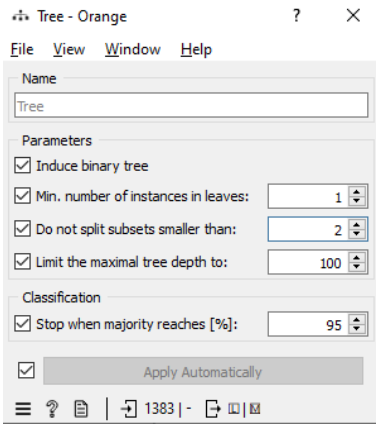
Podemos observar que nossa árvore ficou com 80,2% de precisão, reduzindo bastante em relação ao comportamento dela com 11 níveis.

Predicted

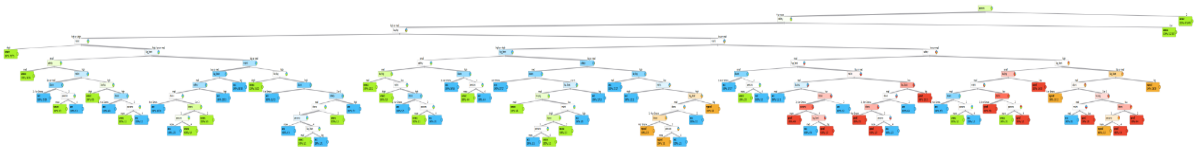
Actual	acc	good	unacc	vgood	Σ
acc	74	0	8	1	83
good	15	0	0	5	20
unacc	26	0	199	2	227
vgood	6	0	0	9	15
Σ	121	0	207	17	345

Na nossa matriz de análise, podemos perceber falhas claras, como, por exemplo, a falta da classe “good” na árvore, de forma que nenhuma folha possui essa conclusão, e das 20 possibilidades na amostra de teste, nenhuma delas foi adivinhada. Além disso, a falta de precisão na hora de adivinhar os veículos da classe “very good”, em que de 15 possibilidades, apenas 9 foram previstas corretamente, e tivemos 14 erros distribuídos pela tabela em relação a essa classe, seja o veículo sendo “very good” e a classe prevista foi outra, ou o veículo sendo de outra classe mas a conclusão do algoritmo foi “very good”.

2) A próxima alteração nos parâmetros que iremos fazer é a busca de uma precisão extremamente grande, de forma que as folhas poderão representar casos isolados e os nós podem se dividir mesmo que tenham poucos casos a serem resolvidos. Além disso, iremos colocar o limite de níveis da árvore em 100, dando liberdade para ela crescer bastante.



Utilizando esses parâmetros vamos obter a árvore:



Que é claramente uma árvore muito grande (temos 145 nós e 73 folhas) que busca o veredito utilizando diversos detalhes. O que se espera de uma árvore assim? Que a precisão seja superior a todas as outras possíveis árvores, visto que estaríamos analisando minuciosamente cada atributo que influencia no veredito final. Mas, por incrível que pareça, a precisão diminui em relação aos parâmetros utilizados na árvore do início do projeto, tendo apenas 96,5% de precisão (menos que o 97,1% obtido anteriormente)

	Tree	error	class	buying	maint	doors	persons	lug_boot	safety
1	un...	0.000	unacc	low	low	2	2	small	high
2	acc	0.000	acc	vhigh	med	4	4	med	med
3	acc	0.000	acc	high	low	3	4	med	high
4	un...	0.000	unacc	low	high	2	2	med	low
5	un...	0.000	unacc	vhigh	med	2	4	small	low
6	un...	0.000	unacc	vhigh	vhigh	2	more	big	high
7	acc	1.000	good	low	med	2	more	big	med
8	un...	0.000	unacc	high	vhigh	2	2	med	high
9	un...	0.000	unacc	low	low	4	2	med	high
10	acc	0.000	acc	low	vhigh	5more	4	small	high
11	un...	0.000	unacc	vhigh	med	2	4	med	med
12	un...	0.000	unacc	vhigh	vhigh	3	4	small	med
13	acc	0.000	acc	high	med	5more	more	small	high
14	un...	0.000	unacc	med	high	4	more	small	med
15	un...	0.000	unacc	high	vhigh	3	4	small	med
16	un...	0.000	unacc	low	high	4	2	big	med
17	go...	0.000	good	low	low	4	more	med	med
18	un...	0.000	unacc	high	med	3	4	med	low
19	go...	0.000	good	low	med	4	more	med	med
20	un...	0.000	unacc	high	high	5more	more	small	low
21	acc	0.000	acc	low	high	4	more	big	med

Show perfomance scores Target class: (Average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
Tree	0.961	0.965	0.965	0.965	0.965	0.931

Esse fenômeno é conhecido como *overfitting* que é o ajuste demasiado dos dados de treinamento de forma que dados que possuem algum tipo de “ruído” acabam perdendo precisão em estruturas de árvores mais complexas.

		Predicted				
		acc	good	unacc	vgood	Σ
Actual	acc	79	0	4	0	83
	good	4	16	0	0	20
	unacc	0	2	225	0	227
	vgood	1	1	0	13	15
	Σ	84	19	229	13	345