1. Considere o dataset ***labor***. Usando os algoritmos *J48*e *simpleCart* (Weka) apresente resultados (as diferentes árvores) que ilustram o uso das várias opções de pruning estudadas e.g. sub-tree raising, replacement, laplace, minimal complexity pruning, etc.

Algoritmo J48:

```
J48 pruned tree
------------------

wage-increase-first-year <= 2.5: bad (15.27/2.27)
wage-increase-first-year > 2.5
|   statutory-holidays <= 10: bad (10.77/4.77)
|   statutory-holidays > 10: good (30.96/1.0)
```

*Figura 1 - J48 com pruning: sub-tree replacement **true**, sub-tree raising **true**. Laplace **false***

```
J48 pruned tree
------------------

wage-increase-first-year <= 2.5: bad (15.27/2.27)
wage-increase-first-year > 2.5: good (41.73/7.0)
```

*Figura 2 - J48 com pruning: sub-tree replacement **true**, sub-tree raising **false**. Laplace **false***

```
J48 unpruned tree
------------------

wage-increase-first-year <= 2.5
|   education-allowance = yes
|   |   wage-increase-first-year <= 2.1
|   |   |   pension = none: bad (2.43/0.43)
|   |   |   pension = ret_allw: bad (0.0)
|   |   |   pension = empl_contr: good (3.16/1.5)
|   |   wage-increase-first-year > 2.1: bad (2.04/0.04)
|   education-allowance = no
|   |   contribution-to-health-plan = none: bad (3.39)
|   |   contribution-to-health-plan = half: good (0.18/0.05)
|   |   contribution-to-health-plan = full: bad (4.06)
wage-increase-first-year > 2.5
|   longterm-disability-assistance = yes
|   |   statutory-holidays <= 10
|   |   |   wage-increase-first-year <= 3: bad (2.0)
|   |   |   wage-increase-first-year > 3: good (3.99)
|   |   statutory-holidays > 10: good (25.67)
|   longterm-disability-assistance = no
|   |   contribution-to-health-plan = none: bad (4.07/1.07)
|   |   contribution-to-health-plan = half: bad (3.37/1.37)
|   |   contribution-to-health-plan = full: good (2.62)
```

*Figura 3 - J48 sem pruning: sub-tree replacement **false**, sub-tree raising **false**. Laplace **false***

Algoritmo SimpleCart:

```
CART Decision Tree

wage-increase-first-year < 2.65: bad(13.0/2.26)
wage-increase-first-year >= 2.65: good(34.73/7.0)
```

Figura 4 - SimpleCart com pruning.

```
CART Decision Tree

wage-increase-first-year < 2.65
|  working-hours < 36.0: good(1.14/1.0)
|  working-hours >= 36.0
|  |  cost-of-living-adjustment=(tcf): bad(1.56/0.91)
|  |  cost-of-living-adjustment!=(tcf)
|  |  |  contribution-to-health-plan=(half): good(0.21/0.07)
|  |  |  contribution-to-health-plan!=(half): bad(10.36/0.0)
wage-increase-first-year >= 2.65
|  statutory-holidays < 10.5
|  |  vacation=(generous): good(3.23/0.0)
|  |  vacation!=(generous)
|  |  |  cost-of-living-adjustment=(tcf): good(1.36/0.2)
|  |  |  cost-of-living-adjustment!=(tcf)
|  |  |  |  education-allowance=(yes): good(0.18/0.03)
|  |  |  |  education-allowance!=(yes): bad(5.75/0.0)
|  statutory-holidays >= 10.5
|  |  contribution-to-health-plan=(half)|(full): good(29.28/0.0)
|  |  contribution-to-health-plan!=(half)|(full): bad(1.0/0.67)
```

Figura 5 - Figura 6 - SimpleCart sem pruning.

Relativamente ás árvores criadas para os casos analisados, é possível constatar que sempre que a função de pruning se encontra desativada, as árvores resultantes apresentam uma maior complexidade a nível de estrutura e profundidade. Isto acontece porque os métodos de sub-tree raising ou sub-tree replacement não são realizados, e, portanto, o modelo cresce conforme os dados de treino que lhe são fornecidos.

O uso do método de pruning torna-se assim vital para reduzir o risco de overffiting perante os dados de treino, levando à criação de uma árvore mais simples de compreender e iterar.

Dentro dos métodos de pruning, considerando que o método de sub-tree raising é mais exigente a nível de tempo de execução e complexidade, pode até afirmar-se que utilizar apenas o método de sub-tree replacement já é suficiente para reduzir significativamente a complexidade da árvore.

Perante a opção que permite a utilização do método de Laplace, pode constatar-se que o método não altera a estrutura da árvore, mas leva a melhores resultados, reduzindo o parâmetro associado ao erro *RMSE*.

1. Apresente a justificação para os resultados obtidos com as avaliações por 10-xval e por training data.

		Parametros avaliação modelo		
Algoritmo	Opção Teste	Error rate	RMSE	F-Measure
J48	Cross-Validation	26,31%	0,4669	0,74
	Training set	12,28%	0,304	0,88
Simple Cart	Cross-Validation	21,05%	0,4292	0,792
	Training set	15,79%	0,3667	0,836

Independente do algoritmo escolhido, é possível analisar que com o uso de dados de treino para teste, o rácio de erro diminui de forma acentuada.

Apesar dos parâmetros apresentados melhorarem quando se usa a opção "*Training set*", estes dados de avaliação podem ser falaciosos para tirar conclusões, isto porque estamos a usar como dados de teste registos que já foram utilizados para treinar o modelo, sendo assim já conhecidos pelo mesmo. Desta forma, a árvore gerada vai modelar muito bem os dados de treino, mas será pouco apropriada para analisar novos casos em utilização futura.

2. Considere o dataset **soybean**. Apresente um estudo comparativo usando validação cruzada e os algoritmos NaiveBayes, SimpleCART e J48. Apresente conclusões sobre os melhores desempenhos para cada classe (das 19 existentes).

Class	ROC Area			Melhor Algoritmo por Classe
	J48	NaiveBayes	SimpleCart	
diaporthe-stem-canker	0,974	1	1	Naive Bayes e SimpleCart
charcoal-rot	1	1	1	Todos
rhizoctonia-root-rot	0,948	1	0,974	Naive Bayes
phytophthora-rot	0,99	1	0,99	Naive Bayes
brown-stem-rot	1	1	1	Todos
powdery-mildew	1	1	1	Todos
downy-mildew	1	1	1	Todos
brown-spot	0,912	0,989	0,984	Naive Bayes e SimpleCart
bacterial-blight	0,975	1	0,997	Naive Bayes
bacterial-pustule	0,974	1	0,946	Naive Bayes
purple-seed-stain	1	1	1	Todos
anthracnose	0,914	1	0,991	Naive Bayes
phyllosticta-leaf-spot	0,753	0,994	0,97	Naive Bayes
alternarialeaf-spot	0,843	0,991	0,961	Naive Bayes
frog-eye-leaf-spot	0,761	0,98	0,955	Naive Bayes
diaporthe-pod-&-stem-blight	0,968	1	1	Naive Bayes e SimpleCart
cyst-nematode	1	1	0,997	J48 e Naive Bayes
2-4-d-injury	0,872	1	0,996	Naive Bayes
herbicide-injury	0,61	1	0,985	Naive Bayes

Figura 7 - Valores da área ROC por classe, usando três diferentes classifeirs

Como podemos analisar pelos valores obtidos na tabela apresentada, conforme a importância que seja necessária de atribuir a uma determinada classe, pode ser relevante o uso de um diferente algoritmo, que favoreça a precisão para os casos dessa classe.

Ou seja, em determinados casos um algoritmo pode ser vantajoso mas, para o estudo de outras classes ou de um outro contexto, essa algoritmo pode já não ser o mais indicado.

De uma forma geral, para o dataset SoyBean, o classifer *Naive Beyes* foi em média o que obteve melhores resultados para qualquer classe. Isto deve-se ao facto do algoritmo olhar para cada classe de forma independente, ao contrário do que acontece com os dois algoritmos de árvores também em análise.

3. Escolha uma classe e 2 classificadores onde o valor de AUC contraria o valor do erro nessa classe. Apresente todos os resultados.

Algoritmo	ROC Area	Previsões Erradas
J48	0.983	1/20
Naive Bayes	1	2/20

Figura 8 - Comparação dos valores ROC Area e rácio de erros para a classe bacterial-pustule

Considerando como classifiers os algoritmos Naive Bayes e J48 (com uso de LaPlace), foi possível analisar que para a classe j = bacterial-pustule, o valor da área da curva contradiz o valor do erro nessa classe.

Dado que em Naive Bayes com 2 previsões erradas a área da curva ROC = 1, seria de esperar que no J48, dado que tem menos previsões erradas, a área da curva fosse superior. Contudo, o seu valor é inferior a 1.

Isto indica que a medida da área da curva ROC é mais robusta para avaliar um modelo, dado que o número de previsões erradas pode variar conforme o método de teste ou os novos dados que sejam avaliados no futuro com o modelo