1. Execute um 10-XVal para o dataset *vote* com o algoritmo *J48* e com o *AdaBoostM1* sobre *J48*. Apresente os resultados obtidos em termos de erro. Sugira uma explicação para a comparação dos erros obtidos e também para os valores de AUC.

O algoritmo AdaBoostM1 implementa o conceito de *Boosting*, que realiza a combinação de múltiplos modelos para obter várias previsões de classificação para um dado exemplo de teste. A classificação final é depois feita através de um processo de votação ou médias destas previsões dos N sub modelos utilizados.

Muito semelhante ao método de *Bagging,* a técnica de Boosting realiza a geração de um novo modelo tenho em consideração as características e erros de previsões do modelos anterior. Ou seja, enquanto que em *Bagging* os N modelos são gerados em paralelo, na técnica de B*oosting* os modelos são gerados sequencialmente, aprimorando e corrigindo os erros da classificação anterior.

Desta forma, será de esperar que este processo de correção e aprendizagem do algoritmo ao longo das iterações, vá levar a um modelo final com uma área abaixo da curva ROC muito próxima de 1 (possível overffiting do modelo aos dados de treino).
Por causa destas características, justifica-se assim o aumento do parâmetro *ROC Area* quando se conjuga com o *J48* o algoritmo *AdaBoostM1*.

		Parametros avaliação modelo			
Algoritmo	Opção Teste	Error rate	RMSE	F-Measure	ROC Area
J48	Cross-Validation	3,67%	0,1748	0,963	0,9715
adaBoost.M1 + J48	Cross-Validation	4,14%	0,1892	0,959	0,9903

Figura 1 - Medidas de desempenho dos algoritmos J48 e adaBoost sobre J48

Visualizando o gráfico da curva ROC através de *Visualize Threshold curve -> democrat* :

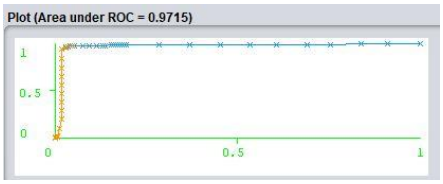

Figura 2- Curva ROC algoritmo J48

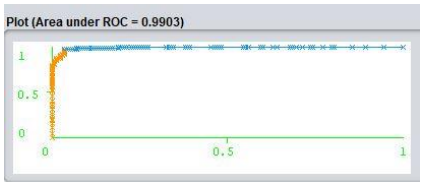

Figura 3 - Curva ROC algoritmo AdaBoost aplicado sobre J48

2. Que tipo de benefícios esperaria da aplicação de *Bagging* sobre *NaiveBayes* num dataset específico, sabendo que o resultado do modelo individual *NaiveBayes* nesse dataset é erro = 0.085. Justifique.

A técnica de Bagging traz vantagens quando aplicada em algoritmos instáveis, onde pequenas variações no conjunto de dados levam a alterações significativas no modelo final gerado. Estes modelos caracterizam-se por possuírem uma variância elevada, como ocorre com alguns processos de geração de árvores de decisão.

Ao aplicar Bagging numa situação destas, as pequenas variações vão ser teoricamente tidas em conta nos N modelos que são gerados em paralelo pelo algoritmo. Quando o modelo combinado final for testado sobre um novo caso, cada um dos submodelos gerados vai votar nesta classificação, permitindo assim reduzir os erros de classificação e aumentar a área abaixo da curva ROC.

Contudo, o algoritmo NaiveBayes é um algoritmo estável e com variância baixa, pelo facto de olhar para cada classe considerando que as mesmas são independentes entre si.

O algoritmo não apresenta assim alterações significativas quando ocorrem ligeiras alterações no dataset e, portanto, será de espera que nesta situação especifica, técnicas de Bagging não tragam vantagens quando aplicadas sobre NaiveBayes .

3. Para o dataset "diabetes" apresente um estudo sobre custo de erros. Usando por exemplo J48 e NaiveBayes, apresente resultados para diferentes matrizes de custo. Faça as avaliações orientadas aos custos e articule conclusões sobre os resultados obtidos usando modelos sensíveis ao custo.

Para resolver esta questão foi usado o classificador *CostSensitiveClassifier*, utilizando o parâmetro *classifier* para selecionar o algoritmo de classificação e o parâmetro *costMatrix* para definir a matriz de custo de acordo com o contexto do problema.

Sem custos		Matriz C1 [0, 1] [1, 0]		Matriz C2 [0, 20] [50, 0]		Mariz C3 [0, 50] [20, 0]	
NaiveBeyes	J48	NaiveBeyes	J48	NaiveBeyes	J48	NaiveBeyes	J48
[422, 78]	[407, 93]	[422, 78]	[407, 93]	[377, 123]	[354, 146]	[460, 40]	[469, 31]
[104, 164]	[108, 160]	[104, 184	[108, 160]	[75, 193]	[65, 203]	[147, 121]	[183, 85]

Figura 4 - Relação entre a atribuição de pesos e a descida de falsos positivos/falsos negativos

Como é visível nos resultados da tabela anterior, quando atribuímos um peso maior aos casos falsos negativos (posição [1,0] da matriz de confusão **C2**), na iteração do algoritmo este custo é visto como uma penalização, levando-o a procurar aprender e reduzir os erros nestas classificações. No caso do NaiveBayes passou de 104 classificações erradas para 75 e no J48, de 108 para 65.

Se tomarmos a decisão oposta, onde é dado um peso superior aos falsos positivos (posição [0,1] da matriz de confusão **C3**), é possível ver que na iteração do algoritmo estes casos incorretamente classificados vão ser também reduzidos significativamente.

De uma forma geral, as matrizes de custos são relevantes de aplicar em *datasets* onde existe uma representatividade reduzida de algumas classes, como neste exemplo, onde existem mais casos de pessoas sem diabetes do que pessoas com a doença. Contudo, é preciso de garantir que mesmo com poucos casos de treino, as pessoas com diabetes são corretamente identificadas e, para isso, a atribuição de pesos às classificações incorretamente realizadas permite ao algoritmo aprender a corrigir estas situações ao longo da sua iteração.

Sendo este problema em torna da classificação se um utente apresenta ou não a doença diabetes, situações de falsos negativos são mais prejudiciais e intoleráveis que situações de falsos negativos, e portanto, uma matriz semelhante à **C2** seria a mais apropriada, porque penaliza exatamente as situações de falsos negativos.

4. **Apresente as principais diferenças entre algoritmos de clustering baseadas em partições e algoritmos hierárquicos. Use exemplos para ilustrar a sua resposta.**

No processo de clustering por partição, as instancias de um *dataset* são, nos casos mais simples, representadas num espaço de N dimensões, onde N representa o número de atributos relevantes a utilizar na análise do dataset. Nestas situações, procura-se descrever um conjunto de regras ou padrões que permitam associar os dados a uma determinada classe, definindo assim um conjunto de híper planos que dividam as instancias do *dataset* nas classes que as representam (figura 5 (a)).

Em algumas abordagens, existe ainda a flexibilidade de permitir que uma instancia pertença a mais do que uma classe, gerando uma separação em partições que pode ser vista como um diagrama de *Venn* (figura 5 (b)).

Numa outra abordagem, existem algoritmos de clustering que produzem uma estrutura hierárquica representada na forma de um dendrograma (Figura 5 (c)). Neste processo, na raiz do esquema é feita uma divisão mais genéricas dos dados em apenas em alguns clusters e, à medida que se desce na profundidade da árvore, cada cluster dá origem a mais sub-clusters, de forma recursiva.

Desta forma, existe uma maior similaridade entre os elementos dos clusters próximos das folhas da árvore. No exemplo da figura 5 (c), as classes **g** e **a** são mais semelhantes que por exemplo as classes **g** e **f**, devido às distancia que as separa no dendrograma. Quanto mais se desce na árvore, menor será também a representatividade das instancias desse cluster no dataset em estudo.

Este tipo de representação é relativamente útil para a deteção e identificação de *outliers*: quanto se visualizar um cluster que não leve a mais nenhuma descendência, podemos estar na presença de uma instancia *outlier*.

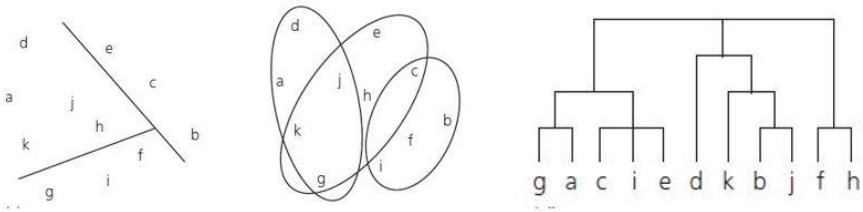

Figura 5 - Esquema de clustering por partição (a) e (b) Esquema por hierarquização (c)

Processo de clustering por partição devem ser aplicados em situações onde se pretende baixar o erro de classificação, procurando uma *accuracy* de classificação elevada.

Os métodos de hierarquização, são utilizados quando estamos num significado mais abrangente das classes, onde a cada cluster é possível atribuir uma taxonomia ou classificação, tendo assim uma elevada divisão e significado atribuído aos clusters. Além disso, o algoritmo precisa apenas de uma noção de distancia entre os dados para os conseguir agrupar por características.