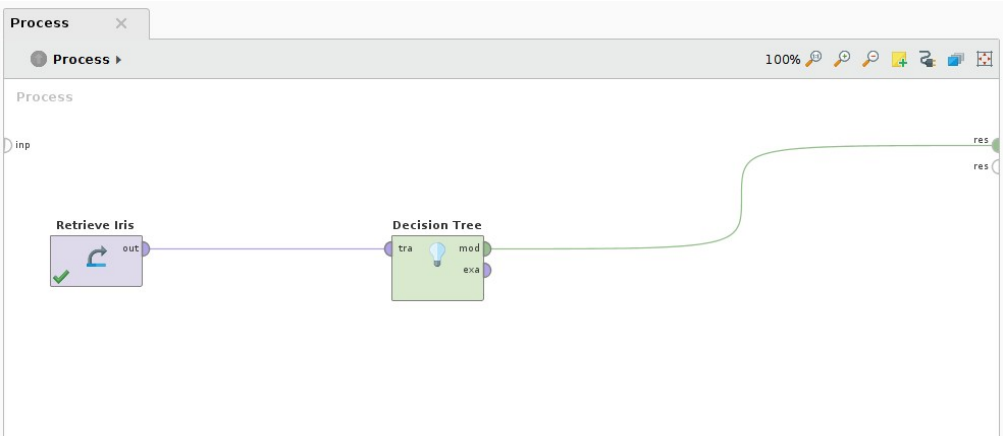
Tutorial básico de classificação em RapidMiner

Neste tutorial, aprenderemos a utilizar as funcionalidades básicas para classificação em Rapidminer. Aplicaremos um algoritmo de classificação qualquer em uma base de dados, avaliaremos o modelo gerado através de uma amostragem *holdout*, *Random subsampling*, validação cruzada e *leave one out*.

1. Aplicando um modelo

Nesta seção, aplicaremos uma árvore de decisão na base de dados Iris. Poderíamos aplicar qualquer algoritmo de classificação em qualquer base suportada pelo algoritmo.

1.1 Chame a base Iris e aplique o operador *Decision Tree* sobre ela, como indicado na figura abaixo. Ligando a saída da árvore de decisão na saída do processo, podemos observar o modelo induzido.



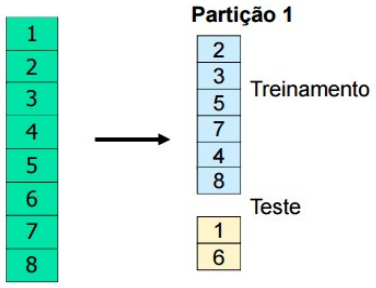
Pronto! Agora temos um modelo para classificar novas flores iris!

Questionamento: Como podemos saber se nosso modelo é capaz de classificar corretamente as flores? Como podemos saber se o viés deste algoritmo de classificação é ideal para nosso conjunto de dados? Como podemos saber se o modelo induzido não sofre overfitting?

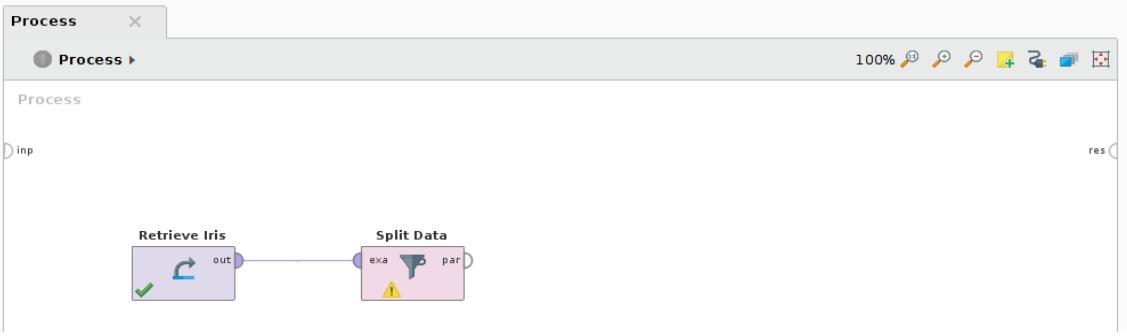
---

2. *Holdout*

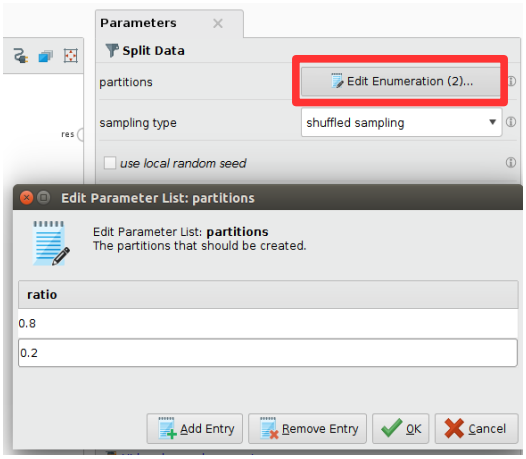
No método *holdout*, dividimos o conjunto de dados em conjunto de treinamento e conjunto de teste. A figura abaixo ilustra este método.



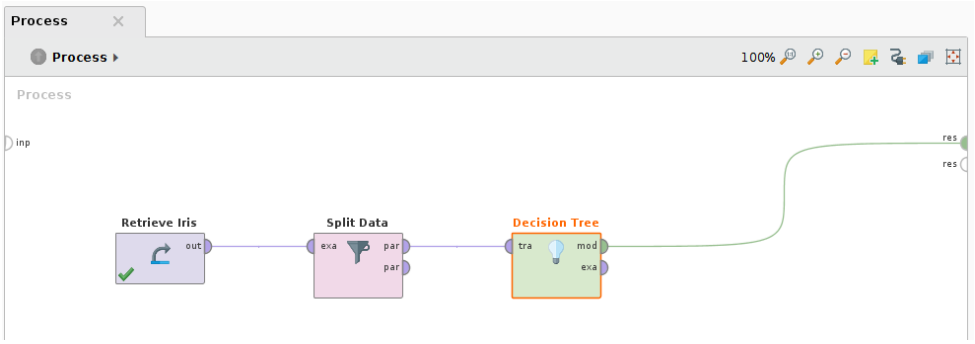
2.1 Chame a base Iris e aplique o operador *Split Data* sobre ela.



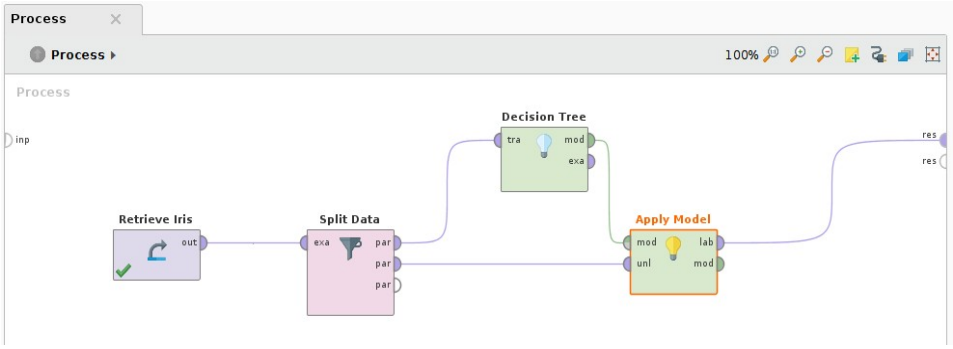
2.2 Nos parâmetros do operador *Split Data*, clique sobre o botão Edit e*numaration*. Uma caixa de diálogo abrir-se-á. Insira as entradas indicadas na figura abaixo. Assim, dividiremos nosso conjunto de dados em um conjunto com 80% e outro com 20%.
OBS: Você pode deixar os conjuntos estratificados modificando o parâmetro *sampling type* – esta opção é muito importante para dados desbalanceados.



2.3 Utilizaremos o conjunto de 80% para treinarmos nossa árvore de decisão. Se ligarmos a saída da árvore na saída do processo, podemos ver o modelo gerado com o conjunto de treinamento.

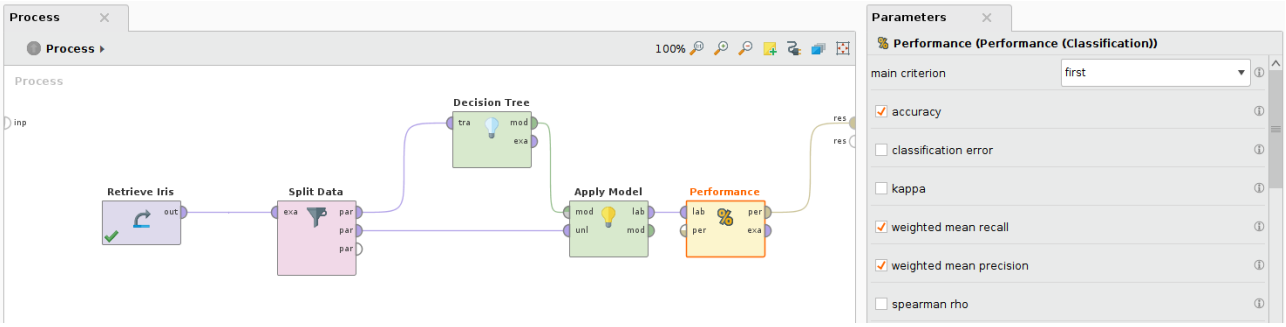


2.4 Utilizaremos o conjunto de 20% como conjunto de teste para nosso modelo. Para isso, aplique o modelo sobre o conjunto utilizando o operador *Apply Model*. Se ligarmos a saída indicado do operador na saída do processo, podemos ver qual a predição dada a cada um dos exemplos no conjunto de teste.



Com esta tabela conseguimos contar quantos exemplos nosso classificador errou. Não parece muito prático, não é?

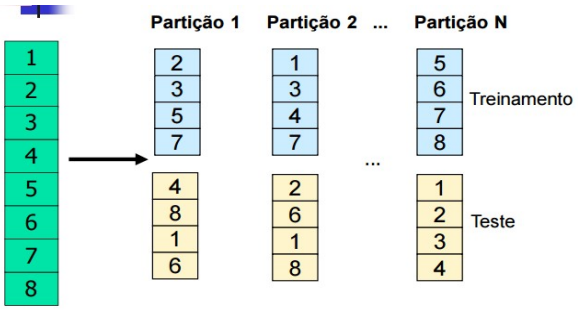
2.5 Podemos utilizar o operador *Performance (Classification)* para medir o desempenho de nosso classificador em diferentes medidas, como acurácia, precisão e revocação. Se ligarmos a saída do operador *Performance* na saída do processo, podemos ver as medidas gerais, medidas por classe e a matriz de confusão.



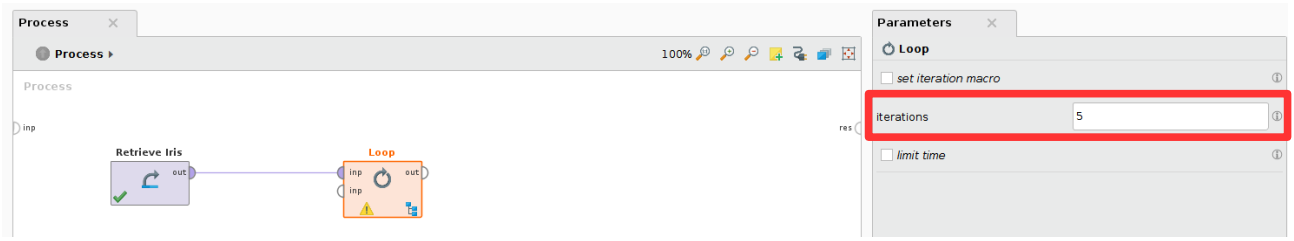
---

3. *Random Subsampling*

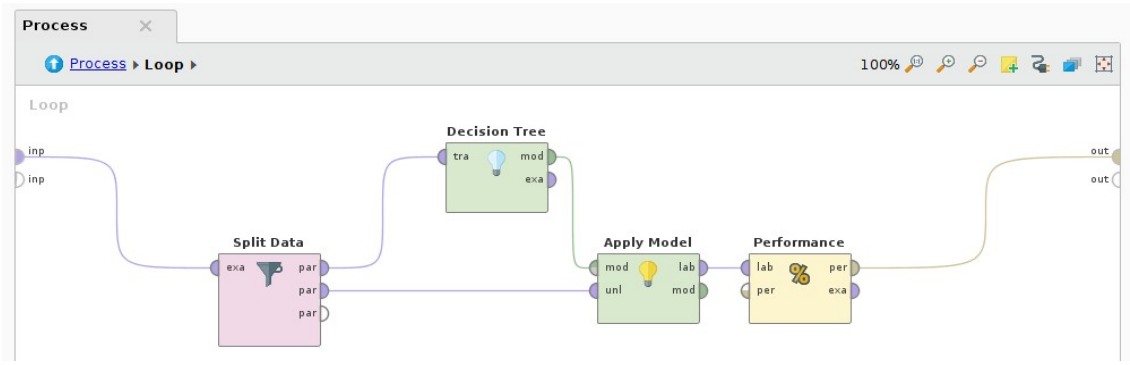
É possível confiarmos no desempenho obtido através de apenas um conjunto de treinamento e um conjunto de teste?
Random subsampling divide o conjunto de dados diversas vezes em treinamento e teste. Equivalente a se fazer *holdout* várias vezes.



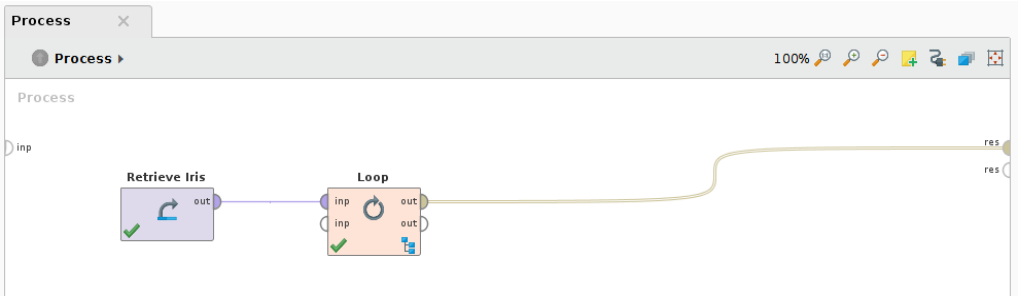
3.1 Vamos aplicar um *Random subsampling* com 5 partições. Aplique o operador *Loop* sobre o conjunto Iris. Defina o número de iterações como 5.



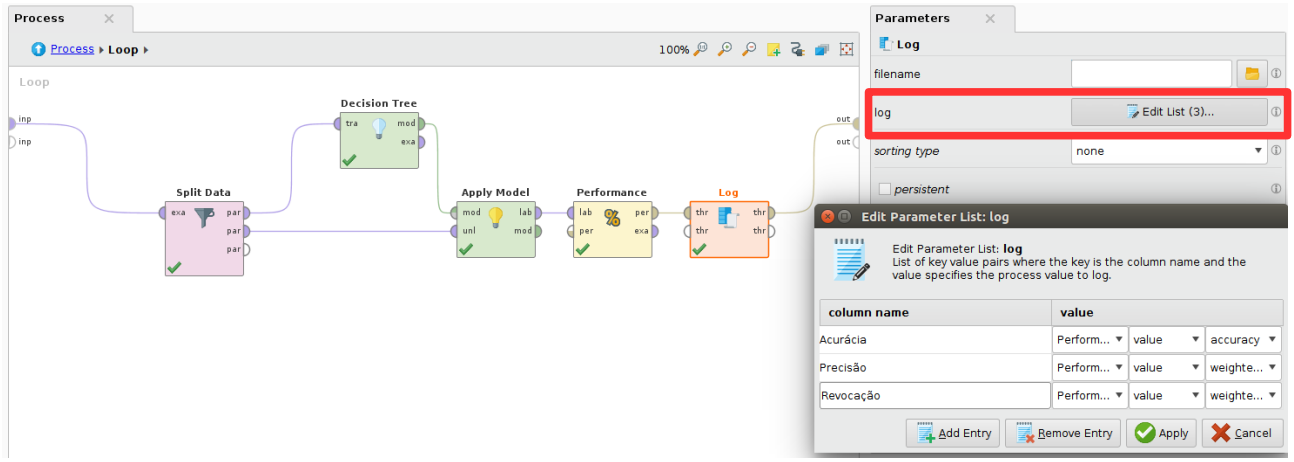
3.2 O operador *Loop* é um subprocesso. Entre no subprocesso do *Loop* e faça o mesmo *holdout* da seção 2.



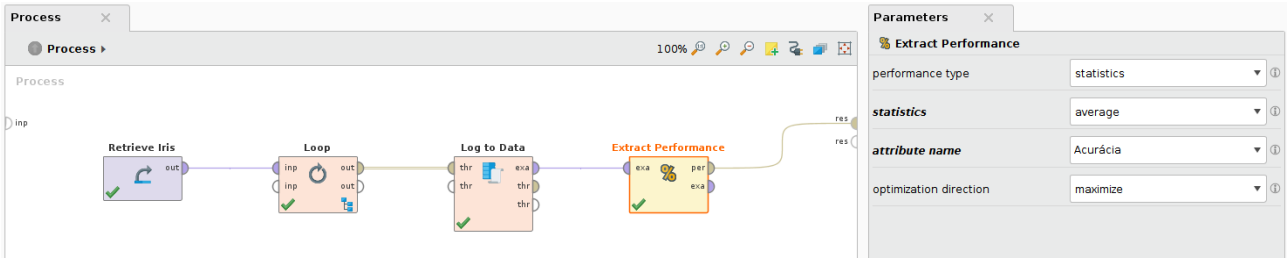
3.3 No processo principal, se ligarmos a saída do operador *Loop* podemos ver o desempenho dos 5 modelos induzidos.



3.4 Se não estivermos contentes com essa visualização de saída, podemos gerar uma tabela através do operador *Log*. Dessa forma, teremos uma tabela com 5 linhas (representando cada uma das partições) e cada uma das colunas sendo uma medida de desempenho. Com esta tabela, podemos gerar gráficos e analisar o desempenho deste algoritmo de classificação em diferentes partições.



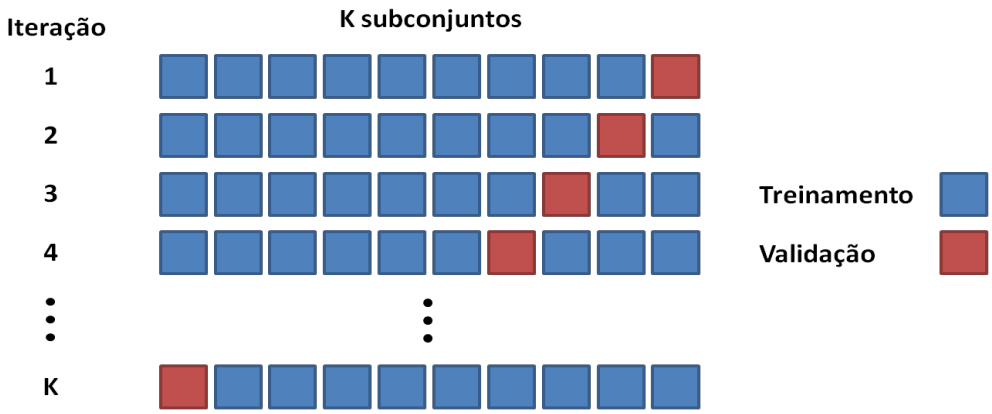
3.5 Caso desejássemos ter como desempenho apenas a média da medida acurácia, podemos transformar nossa tabela de log em um conjunto de dados e extrair a média da acurácia. Para isso, basta utilizarmos os operadores *Log to Data* e *Extract Performance*.



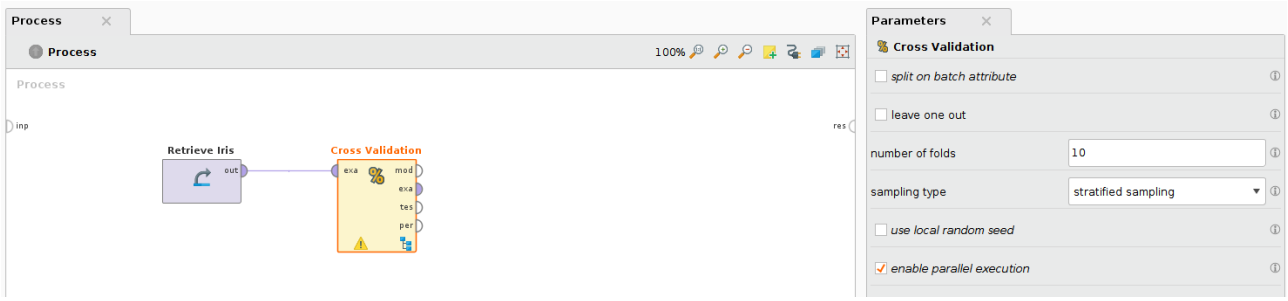
---

4. Validação Cruzada

Um problema do *Random Subsampling* é que alguns exemplos podem nunca estar no conjunto de teste. A validação cruzada mitiga esse problema. Nela, o conjunto de dados é dividido em K partes, sendo que parte k é utilizada como conjunto de teste(ou validação) enquanto as outras K-1 partes são utilizadas como conjunto de treinamento. Assim, a intersecção de todos os conjuntos de teste é sempre vazia e a união é sempre o conjunto de dados completo.

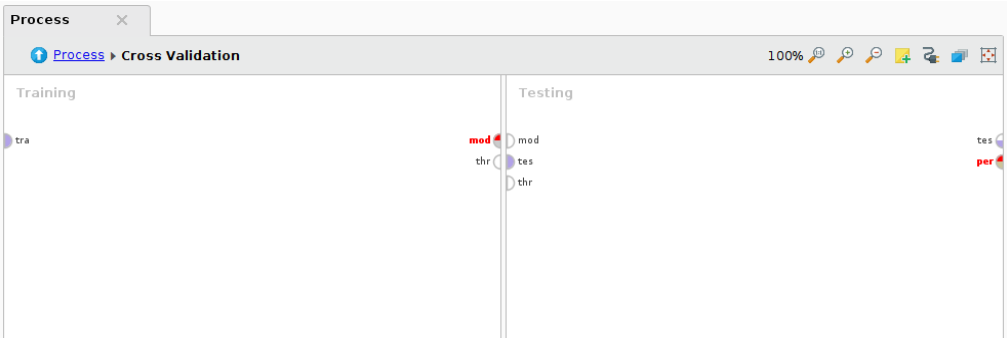


4.1 Para aplicarmos uma validação cruzada com 10 *folds* no conjunto Iris, basta aplicar operador *Cross Validation* sobre o conjunto.

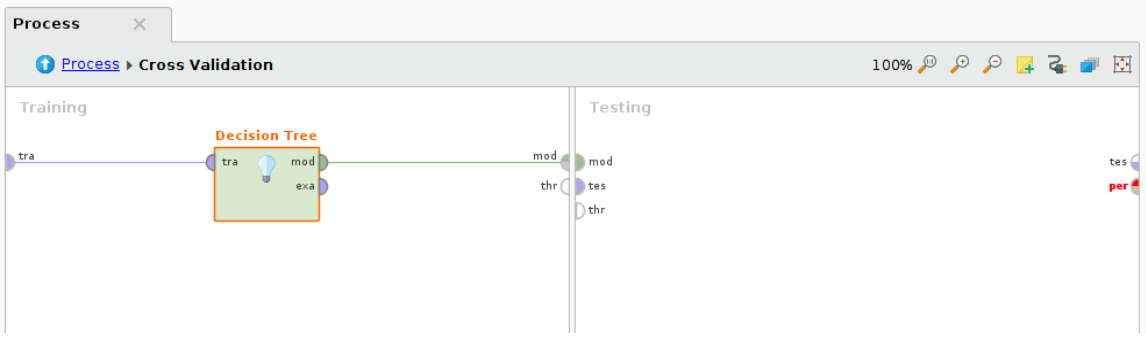


4.2 O operador *Cross Validation* é um subprocesso. Entrando nele podemos observar que ele é dividido em treinamento e teste. Na parte de treinamento, definimos como vamos gerar nosso

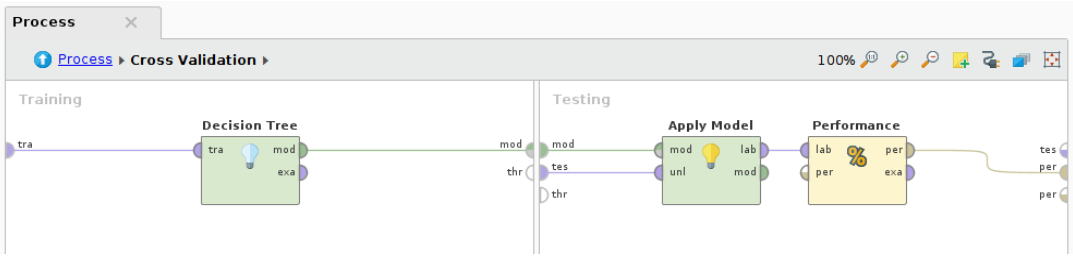
modelo de classificação para cada uma das 10 combinações de conjunto de treinamento. Na parte de teste, definimos como avaliaremos o modelo gerado.



4.3 Vamos gerar uma árvore de decisão para cada partição da validação cruzada.



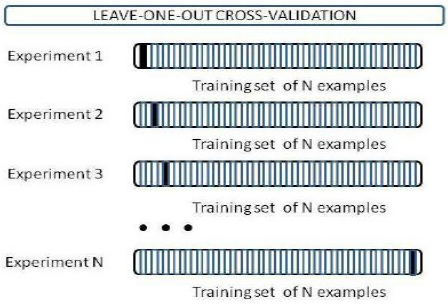
4.4 Agora vamos definir a avaliação de cada um dos modelos gerados. Faremos isso da mesma forma que fizemos anteriormente.



---

5. *Leave one out*

Leave one out é uma variação da validação cruzada. Considere um conjunto de dados com N exemplos. No *leave one out*, os dados são divididos em N partições, sendo que em cada partição apenas um exemplo é utilizado como teste e os outros N-1 são utilizados como treinamento.



5.1 Para aplicar um *leave one out*, basta alterar um parâmetro no operador *Cross Validation*.

QUESTIONAMENTOS

Utilizando apenas as técnicas de amostragem acima, conseguimos realmente estimar o viés adequado para nosso conjunto de dados? O que mais é necessário para isso?