Capítulo 4 - Classificação e Regras de Classificação

Introdução

No contexto de Aprendizado de Máquina, a Classificação pressupõe que um Modelo tenha sido gerado ou induzido a partir de Exemplos de Treinamento. Com este Modelo, novos Exemplos de Teste podem ser classificados. A Figura 4.1 ilustra as três fases desse processo: **Treinamento**, **Aprendizado** e **Classificação**.

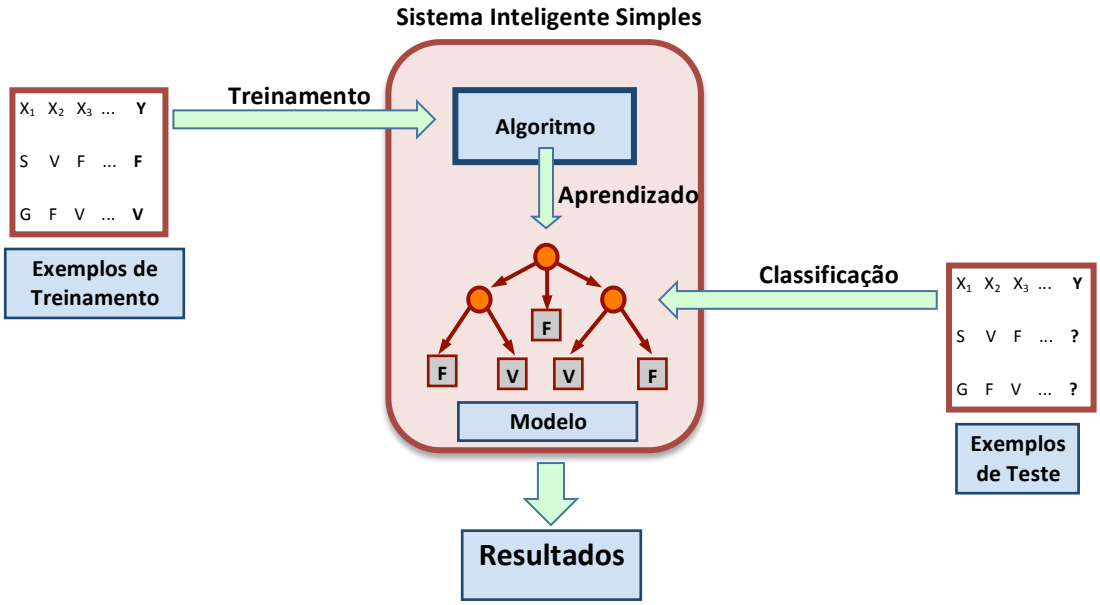

Figura 4.1 – Treinamento, Aprendizado e Classificação em um Sistema Inteligente Simples.

Há várias maneiras de representar o conhecimento embutido num Modelo, sendo as mais comuns Árvores de Decisão e **Regras de Classificação**. As Regras de Classificação assumem a forma genérica:

IF Condição THEN Valor

sendo "Valor" um resultado discreto, como sim/não, baixo/médio/alto verdadeiro/falso etc. Quando os resultados esperados não pertencem a classes

discretas, ou seja, quando “Valor” for uma variável real, a classificação recebe o nome de **Regressão**.

A Classificação através de Regras de Classificação e o processo de geração automática de Regras de Classificação serão os temas desta unidade.

Classificação

O **Aprendizado Supervisionado** se dá através de um **Algoritmo de Aprendizado**, cuja função é criar uma representação do conhecimento extraído de um conjunto de **Exemplos de Treinamento**. Há várias formas possíveis de representação, mas a que nos interessa aqui são as **Regras de Classificação**.

Os Exemplos de Treinamento, por sua vez, são uma forma conveniente de estruturar os dados de uma empresa ou de um certo domínio do saber. Todos os Exemplos de Treinamento têm o mesmo número de atributos, sendo a diferença entre eles representada pelos valores que cada atributo assume.

O algoritmo de aprendizado recebe este nome porque ele cria um modelo cuja aplicação específica não foi pensada pelo seu projetista. Ou seja, ele demonstra certa capacidade de adaptação que melhora seu desempenho depois de uma fase de treinamento. Alguns algoritmos de aprendizado demonstram a capacidade de aprendizado incremental e podem ir melhorando seu desempenho com o acúmulo de experiência. Esta capacidade é muito apreciada porque concede certa **autonomia** aos **Sistemas Inteligentes**.

Uma equipe de físicos, engenheiros e cientistas da computação, responsáveis pelo lançamento de sondas espaciais, conseguiu recentemente desenvolver um algoritmo de aprendizado que permite a uma sonda espacial sair de sua trajetória para desviar-se de uma possível colisão com um meteorito ou cometa inesperado, e retornar à rota original. Alguns robôs atualmente conseguem, sem ajuda externa, voltar à posição vertical depois de uma queda acidental causada por irregularidade no terreno e continuar normalmente sua tarefa de rotina.

Algoritmos de Aprendizado

Um algoritmo de aprendizado pode variar desde aqueles que simplesmente escolhem um dos atributos do Conjunto de Treinamento como a resposta possível a um teste, caso do algoritmo ***oneR*** (uma Regra), passando por aqueles cuja resposta a um novo teste é uma combinação linear dos valores dos atributos, até a utilização de complicados modelos não-lineares, como as ***Redes Neurais***, ou o aprendizado estatístico das ***Máquinas de Vetor de Suporte***, ou ***Support Vector Machines***, **SVM**.

A finalidade desta unidade não é estudar detalhadamente algoritmos de aprendizado. Como há um grande número de ferramentas de acesso livre que implementam estes algoritmos, nosso objetivo é passar uma ideia geral do funcionamento desses algoritmos, cujo entendimento é indispensável para o ajuste adequado de seus parâmetros e para a correta interpretação de seus resultados.

Algoritmo *oneR* ou 1R (uma Regra)

É possivelmente o Algoritmo de Aprendizado para classificação mais simples e, no entanto, seu desempenho pode ser surpreendentemente bom, dependendo da Base de Dados (WITTEN & FRANK, 2005). Esse algoritmo aposta na hipótese de que basta consultar apenas um dos atributos para classificar corretamente os Exemplos de Teste. A tarefa então do algoritmo é encontrar durante o treinamento o atributo que apresenta a menor taxa de erros de classificação.

O algoritmo *oneR* tem bom desempenho para Bases de Dados em que um dos atributos é claramente mais importante que o restante, porque seus valores quase sempre darão uma pista de qual deve ser a classificação correta. Para Bases de Dados em que todos os atributos contribuem com igual peso na resposta, a taxa de acerto do algoritmo tende a ser inversamente proporcional ao número de atributos.

Para compreendermos como o *oneR* calcula a taxa de erros de cada atributo, vamos utilizar a clássica Tabela do Tempo, criada por (QUINLAN, 1986), e reproduzida na Tabela 4.1.

Tabela 4.1 - Tabela do Tempo.

Dia	Temperatura	Umidade	Vento	Partida
Ensolarado	Elevada	Alta	Falso	Não
Ensolarado	Elevada	Alta	Verdadeiro	Não
Nublado	Elevada	Alta	Falso	Sim
Chuvoso	Amena	Alta	Falso	Sim
Chuvoso	Baixa	Normal	Falso	Sim
Chuvoso	Baixa	Normal	Verdadeiro	Não
Nublado	Baixa	Normal	Verdadeiro	Sim
Ensolarado	Amena	Alta	Falso	Não
Ensolarado	Baixa	Normal	Falso	Sim
Chuvoso	Amena	Normal	Falso	Sim
Ensolarado	Amena	Normal	Verdadeiro	Sim
Nublado	Amena	Alta	Verdadeiro	Sim
Nublado	Elevada	Normal	Falso	Sim
Chuvoso	Amena	Alta	Verdadeiro	Não

Primeiramente isolamos um dos atributos, digamos “Dia”, e verificamos qual a distribuição das classes “Sim” e “Não” no atributo de saída “Partida”.

Tabela 4.2 - Relação entre “Dia” e “Partida”.

Dia	Partida
Ensolarado	Sim
Ensolarado	Sim
Ensolarado	Não
Ensolarado	Não
Ensolarado	Não
Nublado	Sim
Nublado	Sim
Nublado	Sim
Nublado	Sim
Chuvoso	Sim
Chuvoso	Sim
Chuvoso	Sim
Chuvoso	Não
Chuvoso	Não

Vamos considerar como “sucesso” a classe (“Sim” ou “Não”) que aparecer com maior frequência, i.e., a maioria, para cada uma das opções possíveis (“Ensolarado”, “Nublado”, “Chuvoso”) do atributo “Dia”, e como “erro” a menos frequente. Então uma inspeção na Tabela 4.2 mostra a seguinte distribuição:

Tabela 4.3 – Taxa de Erros do Atributo "Dia".

Valor do Atributo	Exemplos com Partida=Não	Exemplos com Partida=Sim	Maioria	Erros
Ensolarado	3	2	Não	2/5
Nublado	0	4	Sim	0/4
Chuvoso	2	3	Sim	2/5
Total de Erros				**4/14**

Com base na Tabela 4.3 já podemos gerar algumas regras de classificação iniciais:

Dia(Ensolarado) → Partida=Não ou ***IF*** Dia=Ensolarado ***THEN*** Partida=Não (R 4.1)

Dia(Nublado) → Partida=Sim ou ***IF*** Dia=Nublado ***THEN*** Partida=Sim (R 4.2)

Dia(Chuvoso) → Partida=Sim ou ***IF*** Dia=Chuvoso ***THEN*** Partida=Sim (R 4.3)

Os resultados mostram que em dia ensolarado não há partidas, possivelmente por se tratar de um esporte em quadra coberta e talvez porque os participantes prefiram neste tipo de dia outra atividade a céu aberto. Convém ressaltar que este conjunto de regras baseadas unicamente no atributo "Dia" apresenta uma taxa de erros de 4 em 14, ou 4/14. Vamos reproduzir a tabela de (WITTEN & FRANK, 2005) que aplica o mesmo procedimento para os outros atributos e ver se algum deles apresenta uma taxa de erros menor.

Tabela 4.4 – Taxa de Erros dos Atributos.

Atributo	Regras	Erros	Total de Erros
Dia	Ensolarado → Não	2/5	4/14
	Nublado → Sim	0/4	
	Chuvoso → Sim	2/5	
Temperatura	Elevada → Não	2/4	5/14
	Amena → Sim	2/6	
	Baixa → Sim	1/4	
Umidade	Alta → Não	3/7	4/14
	Normal → Sim	1/7	
Vento	Falso → Sim	2/8	5/14
	Verdadeiro → Não	3/6	

A Tabela 4.4 revela que os atributos "Dia" e "Umidade" apresentam as menores taxas de erros. Adotando qualquer critério arbitrário de desempate, vamos ficar com o conjunto de erros gerados pelo atributo "Umidade" e gerar as seguintes Regras de Classificação:

Umidade(Alta) → Partida=Não ou ***IF*** Umidade**=**Alta ***THEN*** Partida**=**Não (R 4.4)

Umidade(Normal) → Partida=Sim ou ***IF*** Umidade**=**Normal ***THEN*** Partida**=**Sim (R 4.5)

Portanto, quando o algoritmo **oneR** tiver que classificar um novo exemplo, somente o atributo "Umidade" será considerado, e o resultado será baseado nas Regras de Classificação "R 4.4" e "R 4.5". Isso significa que se os Exemplos de Treinamento forem usados como Exemplos de Teste, e supondo que entre os 14 Exemplos de Treinamento não haja contradição entre si, o algoritmo **oneR** deve acertar 10 vezes e errar 4.

Mas, e se o Conjunto de Teste for diferente do Conjunto de Treinamento? Não será a estimativa de 10 acertos e 4 erros demasiadamente otimista ou ela se confirmará com os novos dados? E se o número de Exemplos de Treinamento tivesse sido 140, em vez de 14, que implicações isso teria nas estimativas de acertos?

Há outros algoritmos de classificação bem mais refinados que o **oneR** e que, na maioria dos casos, produzem resultados com taxa de sucesso mais elevada. Um dos algoritmos de classificação mais famosos é o ***PRISM*** (CENDROWSKA, 1987), que utiliza o princípio de "cobertura", i.e., ele vai criando regras que se aplicam ao maior número possível de exemplos do Conjunto de Treinamento, até que toda a tabela esteja "coberta" pelas regras produzidas. Seu desenvolvimento foi inspirado nos "pontos fracos" do algoritmo de indução de Árvores de Decisão ID3 (QUINLAN, 1986), como a dificuldade de entender as árvores muito grandes e complexas geradas pelo algoritmo ID3.

Não vamos aqui nos deter em particularidades do **PRISM** porque os resultados gerados pelo oneR são suficientemente representativos para os nossos propósitos de abordar os métodos de avaliação dos resultados produzidos pelo modelo induzido. E ao avaliarmos os resultados, estamos de certa forma avaliando a capacidade de predição de um modelo para determinada Base de Dados.

Abordar um modelo pelo seu desempenho é interessante porque há evidências empíricas de que nenhum algoritmo tem desempenho superior aos demais para qualquer Base de Dados. A estrutura interna do conjunto de dados desempenha um papel decisivo no desempenho do algoritmo e na qualidade dos resultados.

O algoritmo oneR, com toda sua simplicidade, pode ser imbatível para uma Base de Dados que tenha um atributo que se destaque sobre os demais, cujos valores são redundantes ou irrelevantes. E pode ser uma catástrofe para uma Base de Dados constituída por centenas ou milhares de atributos, todos igualmente importantes. Da mesma forma, qualquer algoritmo pode ter uma taxa de sucesso baixa se o Conjunto de Treinamento não constituir uma amostra representativa do universo de teste.

Avaliação dos Resultados

Após o treinamento para gerar um Modelo através do Algoritmo de Aprendizado, é de grande importância fazer uma avaliação do desempenho do modelo. Em outras palavras, interessa saber quão preditivo é o Modelo Aprendido. Há várias metodologias consagradas para este fim. Vamos iniciar nossa abordagem ao tema relembrando a **Acurácia** de um Classificador Binário.

Considere que nosso Classificador Binário esteja sendo usado para fazer um diagnóstico médico. Se as respostas possíveis para este diagnóstico forem “Positivo” e “Negativo”, quatro possibilidades de predição podem ocorrer:

1. Se o paciente for portador de uma doença e o Classificador acertar no diagnóstico, dizemos que este caso é um **Verdadeiro Positivo** ou **VP**;
2. Se o paciente não for portador da doença e o Classificador acertar no diagnóstico, dizemos que este caso é um **Verdadeiro Negativo** ou **VN**;
3. Se o paciente for portador da doença, mas o Classificador errar no diagnóstico indicando que ele está são, dizemos que este caso é um **Falso Negativo** ou **FN**;
4. Se o paciente não for portador da doença, mas o Classificador errar no diagnóstico indicando que ele está doente, dizemos que este caso é um **Falso Positivo** ou **FP**.

Essas quatro combinações de resultados estão representadas na **Matriz de Confusão** mostrada na Tabela 4.5.

Tabela 4.5 – Matriz de Confusão.

	Positivo Previsto	Negativo Previsto
Positivo Real	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Negativo Real	Falso Positivo (FP)	Verdadeiro Negativo (VN)

A Precisão ou Acurácia do Classificador se expressa pelo número de classificações corretas (VP+VN) divididas pelo número total de classificações (VP+VN+FP+FN),

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \times 100\%$$

Ocorre que em situações reais o custo de um **Falso Positivo** pode não ser igual ao de um **Falso Negativo**, e a Acurácia não consegue captar adequadamente essa situação de interesse.

Suponha que um classificador usado para Detecção de Anomalia tenha que atribuir a cada um dos 100 testes um rótulo de "Situação=Normal" ou "Situação=Anormal". Suponha ainda que a relação entre donos honestos de cartão de crédito e golpistas seja de 96 para 4 e o classificador tenha colocado 99 portadores de cartão na classe "Normal" e apenas um dos golpistas na classe "Anormal". Neste caso a Acurácia do Classificador será de,

$$Acurácia_{Classif} = \frac{96 + 1}{96 + 1 + 3 + 0} \times 100\% = 97\%$$

A interpretação baseada apenas na Acurácia indicaria um excelente desempenho, mas na realidade este é um péssimo classificador para uma operadora de cartões de crédito, porque seu interesse em detectar os golpistas é bem maior do que os donos honestos de cartão! Há um sem número de situações semelhantes a esta. Pense nos danos diferenciados, do ponto de vista de saúde pública, entre fornecer um falso diagnóstico positivo para um paciente são e um falso diagnóstico negativo para um paciente com uma doença contagiosa.

Para detectar estes casos de conjuntos não-balanceados de Falso Positivo e Falso Negativo, podemos definir a **Taxa de Verdadeiro Negativo**, também conhecida por **Especificidade**, como sendo o número de Verdadeiro Negativo (VN) dividido

pelo número total de negativos, que é a soma de Verdadeiro Negativo (VN) mais Falso Positivo (FP), ou seja,

$$Taxa\_VN = \frac{VN}{VN + FP} \times 100\%$$

Se o indicador Taxa de Verdadeiro Negativo for utilizado para o caso citado dos cartões de crédito, teremos uma Taxa de Verdadeiro Negativo de,

$$Taxa\_VN = \frac{1}{1 + 3} \times 100\% = 25\%$$

Para outras situações, pode ser mais conveniente utilizar a **Taxa de Verdadeiro Positivo** , também conhecida por **Sensibilidade**, como sendo o número de Verdadeiro Positivo (VP) dividido pelo número total de positivos, que é a soma de Verdadeiro Positivo (VP) mais Falso Negativo (FN), ou seja,

$$Taxa\_VP = \frac{VP}{VP + FN} \times 100\%$$

Com estes indicadores em mente, suponha que se queira avaliar qual será o desempenho do Modelo gerado pelo Algoritmo de Aprendizado para determinada Base de Dados. Se utilizarmos o mesmo Conjunto de Treinamento como Conjunto de Teste, muito possivelmente a estimativa de desempenho resultará excessivamente otimista para testes reais, com novos Conjuntos de Teste.

Outra alternativa é reservar parte do Conjunto de Treinamento para ser usada como Conjunto de Teste. Mas qual o tamanho ideal da partição do conjunto de Exemplos de Treinamento? E como escolher os elementos deste subconjunto do Conjunto de Treinamento que serão usados para teste? E se o Conjunto de Teste for muito pequeno? Felizmente a Estatística tem estudos bastante robustos sobre questões dessa natureza que podem nos auxiliar.

Avaliação de Desempenho do Classificador

As duas técnicas citadas, conhecidas como **Técnica de Ressubstituição** e **Técnica de Reamostragem**, apresentam pontos fortes e fracos, e justificam a criação de

vários **métodos de avaliação de desempenho** de classificadores. Vamos analisar apenas alguns dos principais métodos de avaliação de desempenho.

Método da Ressubstituição ou *"Use training set"*

Neste método o Conjunto de Treinamento é também utilizado como Conjunto de Teste, conforme mostra a Figura 4.2. Se o Conjunto de Treinamento for uma amostra representativa do universo do problema, suas estimativas de desempenho para um Conjunto de Teste composto por Exemplos não vistos anteriormente podem ser muito boas. Caso contrário, o modelo poderá apresentar muitos erros de generalização durante os testes, seja por problemas de excesso de complexidade do Modelo, que costuma causar *overfitting*, ou por Poda inadequada.

Por outro lado, o fato de o conjunto completo de treinamento ser usado para gerar o Modelo constitui uma vantagem sobre os métodos de reamostragem, principalmente se o número de Exemplos de Treinamento for pequeno.

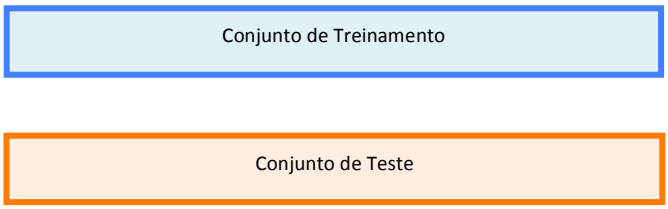

Figura 4.2 – Na Ressubistituição, o Conjunto de Treinamento também é o Conjunto de Teste.

De fato, na simulação Weka da Tabela do Tempo (Tabela 4.1) com o algoritmo oneR usando o método *"Use training set"*, o número de instâncias ou exemplos classificados corretamente foi 10 (71%), e 4 (29%) classificados incorretamente, conforme esperado. A Matriz de Confusão para os 14 Exemplos está mostrada na Tabela 4.6.

Tabela 4.6 – Matriz de Confusão para a Tabela do Tempo com o oneR e o Método *"Use training set"*.

	Não Previsto	Sim Previsto
Não Real	3	2
Sim Real	2	7

Quando repetimos os mesmos procedimentos, porém usando o algoritmo *PRISM*, os resultados foram os seguintes: simulação Weka da Tabela do Tempo (Tabela 4.1) com o algoritmo *PRISM* usando o método *"Use training set"*, o número de instâncias ou exemplos classificados corretamente foi 14 (100%), e 0 (0%) classificados incorretamente. A Matriz de Confusão para os 14 Exemplos está mostrada na Tabela 4.7.

Tabela 4.7 – Matriz de Confusão para a Tabela do Tempo com o *PRISM* e o Método *"Use training set"*.

	Não Previsto	Sim Previsto
Não Real	5	0
Sim Real	0	9

Método da Divisão da Amostra ou *Holdout* ou *Percentage Split*

Consiste na divisão dos Exemplos de Treinamento em dois subconjuntos disjuntos, um para Treinamento, outro para Teste, conforme mostra a Figura 4.3. O valor das porcentagens de cada um dos conjuntos é geralmente expresso numa única porcentagem maior que 50%, estando subentendido que o conjunto menor é o complemento para 100%. O valor de divisão mais comum é 66% para Treinamento e 34% para Teste, embora não haja evidências empíricas que justifiquem essa escolha de 2/3 e 1/3.

Sua vantagem é a simplicidade, mas dependendo da composição obtida, as classes dos Exemplos podem não estar igualmente representadas nos dois conjuntos. Outra limitação desse método está no fato de que menos Exemplos são usados no Treinamento, podendo ter um impacto negativo no desempenho do Modelo induzido.

Figura 4.3 – No *Holdout*, os Exemplos de Treinamento são Divididos em Dois Subconjuntos.

Para Conjuntos de Teste excessivamente pequenos, dividir o já escasso número de Exemplos de Teste pode ter um efeito desastroso ou na geração do Modelo ou na sua avaliação de desempenho. De fato, na simulação Weka da Tabela do Tempo com o algoritmo oneR usando o método da *"Percentage Split"*, com o Conjunto de Treinamento correspondendo a 66% dos Exemplos, o número de instâncias ou exemplos classificados corretamente foi 2 (40%), e 3 (60%)

classificados incorretamente!. A Matriz de Confusão para os 5 Exemplos considerados está mostrada na Tabela 4.8.

Tabela 4.8 – Matriz de Confusão para a Tabela do Tempo com o *oneR* e o Método da *"Percentage Split"*.

	Não Previsto	Sim Previsto
Não Real	0	2
Sim Real	1	2

Quando repetimos os mesmos procedimentos, porém usando o algoritmo *PRISM*, os resultados foram os seguintes: simulação Weka da Tabela do Tempo com o algoritmo *PRISM* usando o método *"Percentage split"*, com o Conjunto de Treinamento correspondendo a 66% dos Exemplos, o número de instâncias ou exemplos classificados corretamente foi 4 (80%), e 1 (20%) classificado incorretamente. A Matriz de Confusão para os 5 Exemplos está mostrada na Tabela 4.9.

Tabela 4.9 – Matriz de Confusão para a Tabela do Tempo com o *PRISM* e o Método *"Use training set"*.

	Não Previsto	Sim Previsto
Não Real	1	1
Sim Real	0	3

Método da Validação Cruzada ou *Cross-validation*

Neste método, os Exemplos de Treinamento são aleatoriamente divididos em *k* partições mutuamente exclusivas ou "*folds*", sendo *k* normalmente igual a 10. A Figura 4.4 mostra um exemplo para *k = 4*, ou seja, 3/4 dos Exemplos de Treinamento são usados para Treinamento e 1/4 dos Exemplos de Treinamento são reservados para a fase de Teste.

A cada iteração um desses *folds* será usado como Conjunto de Teste, enquanto que os outros serão usados para Treinamento. O nome de validação cruzada se justifica então pelo fato de que cada *fold* será usado *(k-1)* vezes para Treinamento e uma vez para Teste. O erro total será a soma dos erros de todas as *k* execuções, enquanto que o erro médio é o erro total dividido pelo número *k* de partições.

Figura 4.4 – Na Validação Cruzada, o Conjunto de Treinamento é Dividido em *k* Partições.

De fato, na simulação Weka da Tabela do Tempo com o algoritmo oneR usando o método da "Cross-validation", com *k* = 10 *folds*, o número de instâncias ou exemplos classificados corretamente foi 5 (36%), e 9 (66%) classificados incorretamente!. A Matriz de Confusão para os 14 Exemplos considerados está mostrada na Tabela 4.10.

Tabela 4.10 – Matriz de Confusão para a Tabela do Tempo com o *oneR* e o Método da *"Cross-validation"*.

	Não Previsto	Sim Previsto
Não Real	3	2
Sim Real	7	2

Na simulação Weka da Tabela do Tempo com o algoritmo *PRISM* usando o método da *"Cross-validation"*, com *k* = 10 *folds*, o número de instâncias ou Exemplos classificados corretamente foi 12 (86%), e 0 (0%) classificados incorretamente! (com 3 Exemplos não classificados). A Matriz de Confusão para os Exemplos considerados está mostrada na Tabela 4.11.

Tabela 4.11 - Matriz de Confusão para a Tabela do Tempo com o *PRISM* e o Método da *"Cross-validation"*.

	Não Previsto	Sim Previsto
Não Real	5	0
Sim Real	0	7

Método Deixe-Um-De-Fora ou *Leave-One-Out*

É um caso especial do Método da Validação Cruzada em que o número de partições *k* é igual ao número de Exemplos *N*, isto é, *k* = *N*, e cada partição é composta por apenas um Exemplo, como mostra a Figura 4.5. A vantagem é que mais dados são usados para o Treinamento, mas a desvantagem é seu custo computacional para os casos em que *N* for muito grande.

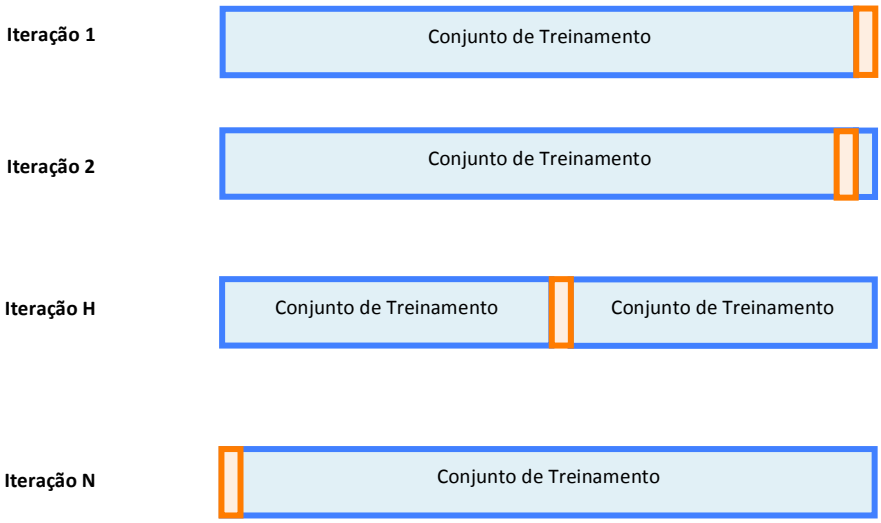

Figura 4.5 – No *Leave-One-Out*, o Conjunto de Treinamento é Dividido em *N* Partições.

Na simulação Weka da Tabela do Tempo com o algoritmo oneR usando o método da *"Cross-validation"*, com *k* = 14 *folds*, número idêntico ao de Exemplos ou instâncias, portanto *k* = *N*, o resultado da classificação foi idêntico ao da *"Cross-validation"*, sendo o número de Exemplos classificadas corretamente 5 (36%), e 9 (66%) classificados incorretamente!. A Matriz de Confusão para os 14 Exemplos considerados está mostrada na Tabela 4.12.

Tabela 4.12 – Matriz de Confusão para a Tabela do Tempo com o *oneR* e o Método da *"Leave-one-out"*.

	Não Previsto	Sim Previsto
Não Real	3	2
Sim Real	7	2

Repetindo o mesmo Conjunto de Teste, porém com o Algoritmo PRISM, usando o método da *"Cross-validation"*, com *k* = 14 *folds*, o resultado da classificação foi idêntico ao da *"Cross-validation"*, sendo o número de Exemplos classificadas corretamente 11 (79%), 0 (0%) classificados incorretamente e 3 Exemplos não classificados. A Matriz de Confusão para os 11 Exemplos considerados está mostrada na Tabela 4.13.

Tabela 4.13 – Matriz de Confusão para a Tabela do Tempo com o *oneR* e o Método da *"Leave-one-out"*.

	Não Previsto	Sim Previsto
Não Real	5	0
Sim Real	0	6

Nesta série de simulações, o algoritmo PRISM produziu um classificador com melhor desempenho que o classificador do algoritmo oneR para o mesmo Conjunto de Treinamento.

Considerações Finais

Nesta unidade, vimos um algoritmo simples, conhecido como oneR, usado para gerar Regras de Classificação. Outros algoritmos mais refinados usam princípios mais complexos, como o de cobertura, para produzir Regras de Classificação (caso do PRISM).

Foram apresentados alguns indicadores que auxiliam o usuário a decidir se os resultados obtidos são satisfatórios ou não. Também foram apresentados alguns métodos para estimar o desempenho futuro do classificador em situação de testes reais.

Duas simulações comparativas entre o desempenho do algoritmo oneR e PRISM para a Tabela do Tempo foram apresentadas para ajudar a fixar os conceitos aprendidos.

Lista Exercícios

1. Explique **com suas próprias palavras** as vantagens e desvantagens das **Árvores de Decisão** e **Regras de Classificação**.

2. Explique **com suas próprias palavras** os Métodos *"**Cross-validation**"* e *"**Holdout**"*.

3. Carregue o arquivo **"iris.arff"** (anexo) no Weka e faça a Classificação usando o algoritmo **"oneR"**, com o método ***"Cross-validation"*** (10 *Folds*).

(a) Faça uma análise da **Matriz de Confusão** gerada, reproduzindo os valores obtidos na simulação.

(b) Ainda com base nos resultados da simulação no Weka, explique **com suas próprias palavras** por que a classe **"Iris Setosa"** tem a mais alta taxa de **Verdadeiro Positivo** *("TP Rate")*, enquanto que a **"Iris Versicolor"**, a mais baixa (consulte a Figura 3.1 do Capítulo 3).

Referência Bibliográfica

CENDROWSKA, J. **PRISM: An Algorithm for Inducing Modular Rules**. International Journal of Man-Machine Studies. Vol. 27, pp. 349-370, 1987. In http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1987-Cendrowska-IJMMS.pdf. Acessado em 06.03.2013.

QUINLAN, J. R. **Induction of Decision Trees**. Machine Learning, Vol. 1, No. 1, pp. 81-106. Boston: Kluwer Academic Publishers, 1986.

REZENDE, S. O. (Organizadora). **Sistemas Inteligentes: Fundamentos e Aplicações.** Barueri: Editora Manole Ltda, 2005.

ROCHA, M.; CORTEZ, P. & NEVES, J. M. **Análise Inteligente de Dados: Algoritmos e Implementação em Java.** Lisboa: Editora de Informática, 2008.

TAN, P.N.; STEINBACH, M. & KUMAR, V. **Introdução ao Data Mining Mineração de Dados.** Rio de Janeiro: Editora Ciência Moderna Ltda., 2009.

WITTEN, I. H. & FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques.** Second Edition. Amsterdam: Morgan Kaufmann Publishers, 2005.