

Avaliação de Modelos

Departamento de Engenharia Informática (DEI/ISEP)

Fátima Rodrigues

mfc@isep.ipp.pt

Avaliação de Modelos

- O desempenho de um modelo para além do algoritmo usado na sua construção depende de vários outros fatores:
 - Distribuição das classes
 - Tamanho dos conjuntos Treino/Teste
 - Custo más classificações
- **Métodos de amostragem para avaliação de modelos**
 - Como obter estimativas fiáveis ?
- **Métricas para avaliação de modelos**
 - Como avaliar o desempenho de um modelo ?
- **Como comparar modelos**
 - Como comparar o desempenho relativo entre diferentes modelos ?

Métodos de Amostragem para

Avaliação de Modelos

Objetivo Principal da Estimativa Desempenho

Obter uma **estimativa fiável** do erro de previsão esperado de um modelo sobre uma distribuição de dados desconhecida

- Idealmente devemos **repetir os testes várias vezes**
- Desta forma, podemos recolher uma série de estimativas e fornecer como **estimativa final** a média das estimativas individuais, sempre acompanhada do respetivo desvio padrão

A regra de ouro de Estimativa de desempenho:

Os dados utilizados para a avaliação de qualquer modelo não podem ser usados durante o desenvolvimento do modelo

Método *Holdout*



O conjunto inicial de dados é dividido aleatoriamente em dois conjuntos disjuntos:

- **Treino** → $2/3$ conj^{to} inicial
 - ↳ dados para desenho do modelo
 - ↳ quanto maior o conj^{to} treino melhor o modelo
- **Teste** → $1/3$ conj^{to} inicial
 - ↳ dados para testar o modelo
 - ↳ quanto maior o conj^{to} teste mais fiável a estimativa do erro

Python: Holdout Estratificado

```
from sklearn.model_selection import train_test_split

X = dfScale
y = df.diagnosis

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, random_state=1, stratify=y)

print("Train B/M distrib.\n",y_train.value_counts(normalize=True))
print("Test B/M distrib.\n",y_test.value_counts(normalize=True))

Train B/M distribution
0 0.628141
1 0.371859
Name: diagnosis, dtype: float64
Test B/M distribution
0 0.625731
1 0.374269 Name: diagnosis, dtype: float64
```

Limitações Método Holdout

- Só é possível aplicar quando existe um **volume considerável de dados** e todas as classes estiverem bem representadas (largo nº de instâncias por classe)
- O modelo é altamente dependente da composição dos conj^{tos} de treino e teste
- Os **conj^{tos} de treino e teste não são independentes**, uma vez que são subconjuntos do mesmo conjunto inicial

Método Holdout Repetido

O método *holdout* pode ser repetido várias vezes em ordem a melhorar a estimativa do desempenho do modelo → **Método Holdout Repetido**

O erro do modelo é a média da previsão do erro obtido em cada iteração:

$$\bar{E} = \frac{1}{k} \sum_{i=1}^k E_i$$

O respetivo desvio-padrão desta estimativa

$$\sigma = \sqrt{\frac{1}{k} \sum_{i=1}^k (E_i - \bar{E})^2}$$

Se a amostra de dados disponível é muito grande as repetições podem ser muito exigentes em termos computacionais

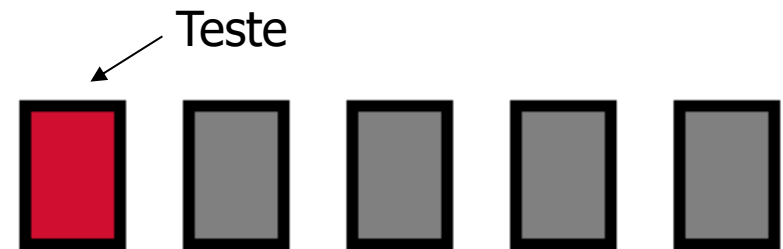
Validação Cruzada (*Cross-validation*)


O conjunto inicial de dados é dividido em K -subconj^{tos} de igual tamanho



Em cada iteração são usados:

- $(k-1)$ subconj^{tos} para treino
- 1 subconj^{to} para teste



K Repetições { 



O erro do modelo é a média da previsão do erro obtido nas K iterações

Validação Cruzada

A avaliação é realizada k vezes (**K=5** ou **K=10**)

Intervalos de Confiança para Cross Validation

Erro total = média dos erros das k-iterações

➔ **estimativa mais robusta**

O **intervalo de confiança** é o desvio padrão dos K-fold

Vantagens

- Utiliza o máximo de dados possível para treino
- Os conjuntos de teste são mutuamente exclusivos – **cobrem todo o conjunto de dados**
- Estratificação **reduz a variância das estimativas**

Python: Validação Cruzada

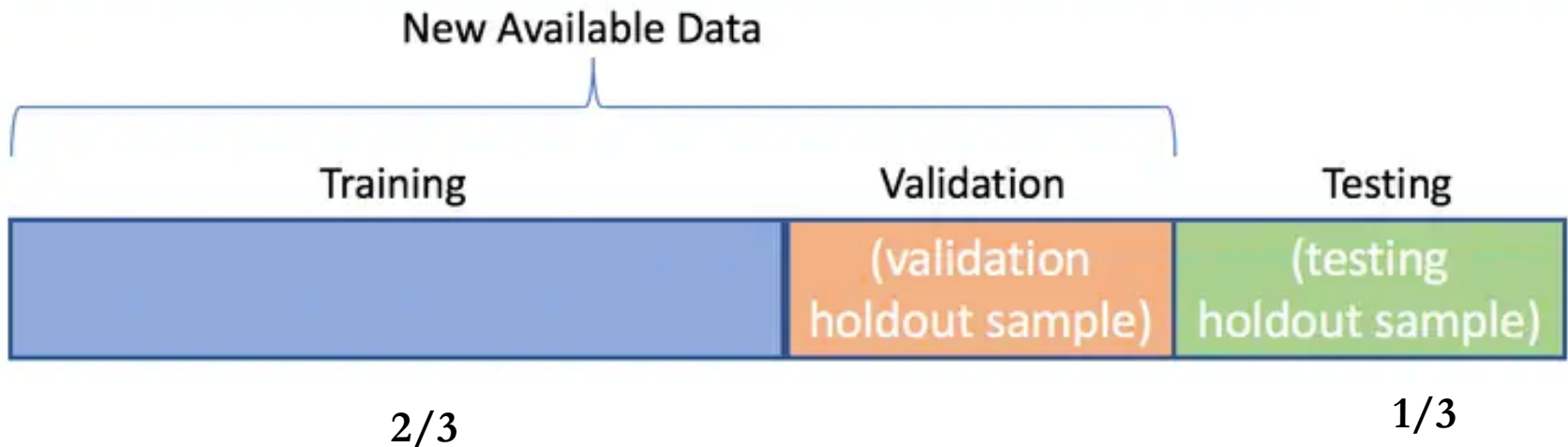
```
from sklearn.model_selection import Kfold
from sklearn.model_selection import cross_val_score

models = []
models.append(('LR', LogisticRegression()))
models.append(('KNN', KNeighborsClassifier()))

results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, shuffle=True, random_state=123)
    cv_results = cross_val_score(model, X_train, y_train,
                                  cv=kfold, scoring='accuracy')

    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(),
                           cv_results.std())
    print(msg)
```

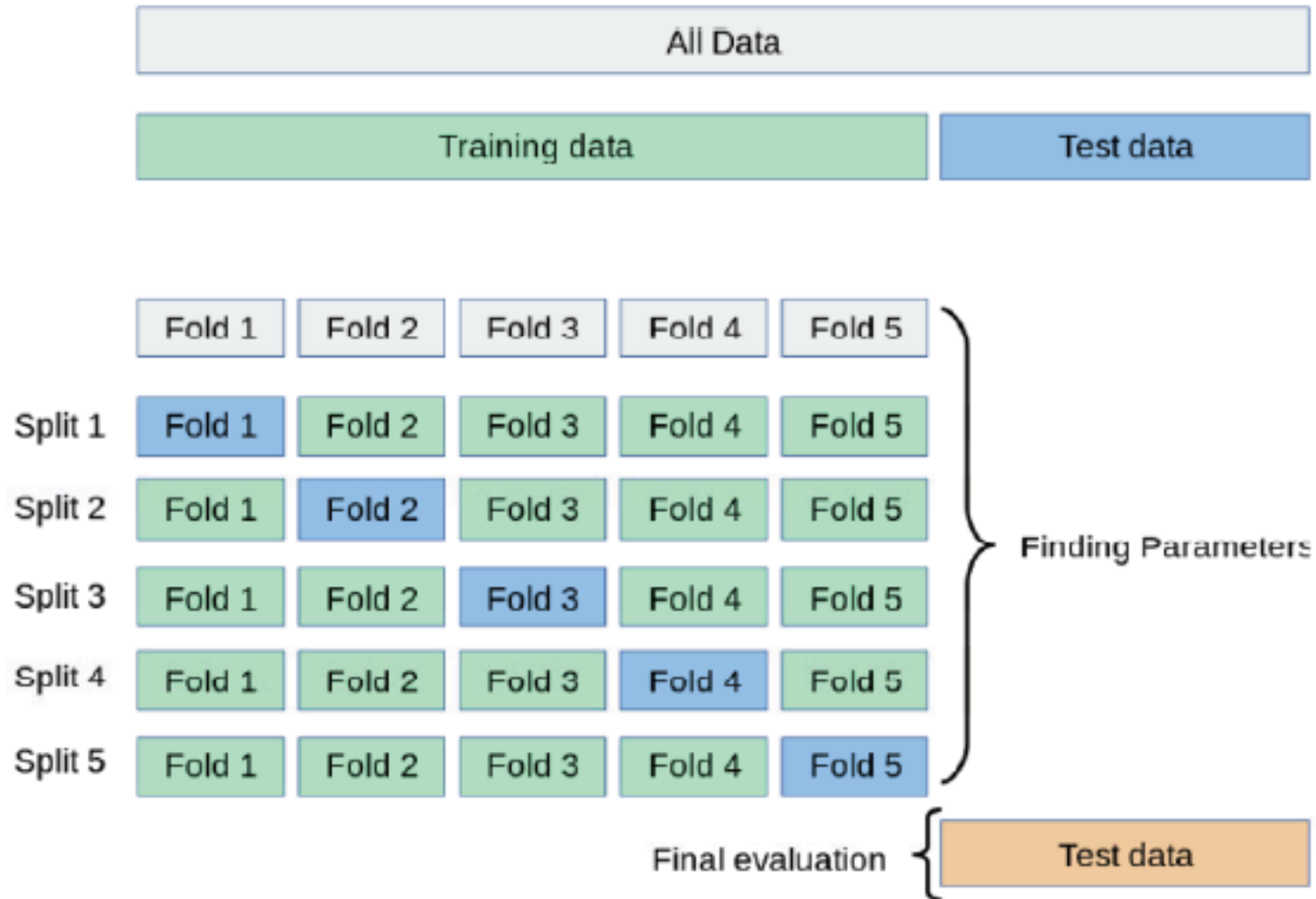
Ajustar os parâmetros dos Algoritmos



Quando é necessário otimizar os parâmetros dos algoritmos considera-se mais um conjunto: **conjunto de validação**

- **Treino** → 2/3 conj^{to} inicial
 - ↳ dados usados para desenho do modelo
- **Validação** → 1/3 conj^{to} Treino
 - ↳ dados usados refinar o modelo
- **Teste** → 1/3 conj^{to} inicial
 - ↳ dados para testar o modelo

Ajustar os parâmetros dos Algoritmos



Leave-One-Out

- Ideia semelhante à Validação Cruzada, mas neste caso em cada iteração um único caso é deixado de fora do conjunto de treino
- Isto significa que **Leave-one-out** é essencialmente equivalente à VC com **n-fold vezes**, onde n é o tamanho do conjunto de dados disponível
- Para uma amostra de tamanho n , o modelo é induzido a partir de $n-1$ exemplos e testado com o exemplo excluído
- O processo é repetido n vezes, o erro é a soma dos erros em cada teste dividido por n
 - ↳ amostragem não estratificada: teste realizado apenas num exemplo
 - ↳ procedimento determinístico: sem amostragem aleatória
 - ↳ **computacionalmente dispendioso**: adequado para amostras muito pequenas
 - ↳ estimativa do desempenho do modelo com **elevada variância**, uma vez que o teste ocorre num só exemplo

Método *Bootstrap* (Amostragem com Reposição)

Conj^{to} de Treino: são selecionadas n instâncias do conj^{to} inicial de dados para formar o conj^{to} de treino, mas estas n instâncias não são retiradas da amostra inicial

Para grandes amostras, com n registros, a probabilidade de uma instância ser selecionada pelo método bootstrap é de

$$1 - \left(1 - \frac{1}{n}\right)^n \approx 1 - e^{-1} = 0.632$$

O que significa que o **conjunto de treino** embora apresente o tamanho n do conjunto original de dados conterá aproximadamente **63.2% das instâncias do conjunto inicial** – pois algumas instâncias são repetidas

Método *Bootstrap* (Amostragem com Reposição)

Conj^{to} de Teste: contém as instâncias não selecionadas para formar o conj^{to} de treino - **out-of-bag samples**

Para grandes amostras, com n registos, a probabilidade de uma instância não ser selecionada pelo método bootstrap é de

$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

Para grandes conjuntos de dados, o **conjunto de teste** conterá aproximadamente **36.8%** das instâncias do conjunto inicial

O procedimento de amostragem é repetido k vezes, sendo a estimativa do erro final

$$e = \frac{1}{k} \sum_{i=1}^k (0.632e_{teste} + 0.368e_{treino+teste})$$

↪ **estimativa pessimista**

Orientações sobre o que usar e quando

- Para **grandes conjuntos de dados**, o método mais adequado é o **Holdout repetido**
- Para conjuntos de dados de **tamanho médio k-fold** é a escolha mais frequente, em particular com $k=10$
- É também muito comum repetir o processo k-fold várias vezes para aumentar a significância estatística e diminuir a variância
- Subamostragem aleatória também é uma possibilidade, embora não tão frequentemente usado
- **Bootstrap** e **leave-one-out** são as escolhas habituais para pequenos conjuntos de dados, com bootstrap sendo a escolha mais frequente com um grande número de repetições (por exemplo 200)

Métricas para Avaliação de Modelos

Avaliação Modelos de Regressão

Avaliação de Modelos Regressão

Existem duas categorias de erros de previsão:

- **erros dependentes de escala:** estão na mesma escala que os dados
- **erros de percentagem:** estão em forma de percentagem e podem ser comparados com modelos aplicados a diferentes conj^{tos} de dados

Medidas de Erros de Escala

Erro médio absoluto

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Raiz quadrada do MSE

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Medidas de percentagem de erros

Medidas de percentagem de erros ($p_t = 100e_t/y_t$) mais comuns:

- Erro percentual absoluto médio

$$MAPE = \left(\frac{1}{n} \sum_{i=1}^n \left| \frac{e_i}{y_i} \right| \right) \times 100$$

- Erro percentual absoluto da média simétrica

$$sMAPE = \frac{2}{n} \times \sum_{i=1}^n \frac{|e_i|}{|y_i| + |\hat{y}_i|}$$

Modelos de Classificação

Matriz de Acertos/Erros ou Matriz Confusão

Classificação Binária

Classe Objectivo	Classe Prevista		
		^Classe +	^Classe -
	Classe +	TP (++)	FN (+-)
	Classe -	FP (- +)	TN (- -)

Classe + : Classe de Interesse

TP– Positivos Verdadeiros

nº de casos positivos correctamente classificados

FP– Falsos Positivos

nº de casos negativos incorrectamente classificados como positivos

TN – Negativos Verdadeiros

nº de casos negativos correctamente classificados

FN – Falsos Negativos

nº de casos positivos incorrectamente classificados como negativos

Métricas para Avaliação Desempenho

CLASSE OBJECTIVO	CLASSE PREVISTA		
		^Classe +	^Classe -
	Classe +	TP	FN
	Classe -	FP	TN

$$\text{Taxa de Acerto (Accuracy)} = \frac{TP + TN}{TP + TN + FP + FN}$$

A métrica *accuracy* representa a taxa de acerto de todo o classificador, isto é, a razão entre a soma dos acertos nas classes a prever e o número total de previsões

$$\text{Taxa de Erro} = 1 - \text{Accuracy}$$

$$\text{Taxa de Erro} = 1 - \frac{TP + TN}{TP + FP + TN + FN} = \frac{FP + FN}{TP + FP + TN + FN}$$

Estatística Kappa

A estatística Kappa avalia se o valor de accuracy foi gerado puramente por acaso

$$Kappa = \frac{accuracy - random\ accuracy}{1 - random\ accuracy}$$

$$random\ accuracy = \frac{(TN + FP)(TN + FN) + (FN + TP)(FP + TP)}{(TP + TN + FP + FN)^2}$$

Kappa [-1,..., 1]

Uma interpretação comum da estatística Kappa:

- < 0.20 acordo Pobre
- 0.20 – 0.40 acordo Fraco
- 0.40 – 0.60 acordo Moderado
- 0.60 – 0.80 Bom acordo
- 0.80 – 1.0 Muito boa concordância

Limitação da medida Tx. Acerto (accuracy)

Accuracy Paradox

Modelo M_1	Classe Prevista		
Classe Objectivo		+	-
	+	0	10
	-	0	990

$$\text{Accuracy}(M1) = 99\%$$

Modelo M_2	Classe Prevista		
Classe Objectivo		+	-
	+	6	4
	-	292	698

$$\text{Accuracy}(M2) = 70.43\%$$

A medida *Accuracy* dá a mesma importância a todas as classes pelo que **não é adequada**:

- para avaliar conjuntos de dados desbalanceados
- Para avaliar classes com diferentes custos associados

Matriz de Custo

Quando **diferentes erros** envolvem **diferentes custos**

Matriz Custo	Classe Prevista		
	$C(i j)$	^Classe +	^Classe -
	Classe +	$C(+ +)$	$C(+ -)$
	Classe -	$C(- +)$	$C(- -)$

$C(i|j)$: Custo de classificar classe i como classe j

Dependendo do problema que estamos a modelar o custo associado com as Falsas Previsões pode ser diferente

Calculo do Custo da Classificação

Matriz Custo	Classe Prevista		
	C(i j)	+	-
	+	0	100
	-	1	0

Modelo M ₁	Classe Prevista		
Classe Objectivo		+	-
	+	150	40
	-	60	250

Modelo M ₂	Classe Prevista		
Classe Objectivo		+	-
	+	250	45
	-	5	200

$$C_t(M) = TP \times C(+,+) + FP \times C(-,+) + FN \times C(+,-) + TN \times C(-,-)$$

Accuracy (M₁) = 80%

C_t(M₁) = **4060**

Accuracy (M₂) = 90%

C_t(M₂) = **4505**

Métricas Sensíveis ao Custo

Classe Objectivo	Classe Prevista		
		Classe +	Classe -
	Classe +	TP	FN
	Classe -	FP	TN

Taxa de acerto na classe Positiva (Sensitivity or recall)

fração de exemplos positivos corretamente previstos

$$TPR = \frac{TP}{TP + FN}$$

Taxa de acerto na classe Negativa (Specificity)

fração de exemplos negativos corretamente previstos

$$TNR = \frac{TN}{TN + FP}$$

Taxa de Acerto Balanceada

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2}$$

	Classe Prevista		
		+	-
Classe Objetivo	+	5	10
	-	50	10000

$$\text{Accuracy} = \frac{5 + 10000}{5 + 50 + 10 + 10000} = 99.4\%$$

$$\text{Sensitivity} = \frac{5}{5 + 10} = 33.3\%$$

$$\text{Specificity} = \frac{10000}{50 + 10000} = 99.5\%$$

$$\text{Balanced Accuracy} = \frac{33.3\% + 99.5\%}{2} = 66.4\%$$

	Classe Prevista		
		+	-
Classe Objetivo	+	0	15
	-	0	10050

$$\text{Accuracy} = \frac{10050}{0 + 0 + 15 + 10050} = 99.9\%$$

$$\text{Sensitivity} = \frac{0}{0 + 15} = 0\%$$

$$\text{Specificity} = \frac{10050}{0 + 10050} = 100\%$$

$$\text{Balanced Accuracy} = \frac{0\% + 100\%}{2} = 50\%$$

Precision/Recall

Precisão indica a percentagem de previsões positivas corretas

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{True Positives}}{\text{Predicted Positives}}$$

Recall indica a percentagem de exemplos positivos corretamente previstos

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{True Positives}}{\text{Actual Positives}}$$

Precision/Recall

M1	Classe Prevista		
Classe Objectivo		^criminal	^not criminal
	criminal	12	8
	not criminal	3	34

precision = $12/15 = 0,80$
recall = $12/20 = 0,60$

M2	Classe Prevista		
Classe Objectivo		^criminal	^not criminal
	criminal	16	4
	not criminal	9	31

precision = $16/25 = 0,64$
recall = $16/20 = 0,80$

É importante testar uma variedade de modelos, a fim de encontrar a combinação de *Precision* e *Recall* que atende às necessidades do projeto

Construir modelos que maximizam simultaneamente a *Precision* e *Recall* é o desafio dos algoritmos de classificação

Medida F1

Média harmónica ponderada da *Precision* e *Recall*

$$\text{Medida } F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Minimiza todos os custos exceto TN

Classe Objectivo	Classe Prevista		
		Classe +	Classe -
	Classe +	1	99
	Classe -	0	1000

Precision = **100%** FP=0

Recall = **1%** FN=99

F1 = **2%** FN >> TP

Nota: As medidas *Precision* e *Recall* não são adequadas para avaliar modelos inferidos a partir de dados com classes desbalanceadas

Métrica Alternativa: Medida F_α

A medida F combina as duas métricas *precision* e *recall* em uma única métrica. Fórmula genérica $\alpha \in \mathbb{R}, \alpha > 0$

$$F_\alpha = \frac{(1 + \alpha)(precision \times recall)}{(\alpha \times precision + recall)}$$

A medida F é parametrizada e pode ser ajustada para especificar a importância relativa da *precision* versus *recall*:

- medida F_1 é a **média harmónica ponderada** da *precision* e *recall*

$$F_1 = \frac{2 \times precision \times recall}{(precision + recall)}$$

- medida F_2 pondera *recall* o dobro da *precision*

$$F_2 = \frac{3 \times precision \times recall}{(2 \times precision + recall)}$$

- medida $F_{0.5}$ pondera *precision* o dobro da *recall*

$$F_{0.5} = \frac{1.5 \times precision \times recall}{(0.5 \times precision + recall)}$$

Classificação Multi-classe

		Classe Prevista			
Classe Objectivo		C_1	C_2	...	C_n
	C_1	n_{11}	n_{12}	...	n_{1n}
	C_2	n_{21}	n_{22}	...	n_{2n}
	...				
	C_n	n_{n1}	n_{n2}	...	n_{nn}

Numa matriz de acertos/erros multi-classe não há classe positiva versus classe negativa

Em problemas de **classificação multi-classe** existem **n matrizes binárias**, uma para cada classe

O desempenho global do classificador é feito através da combinação das medidas de avaliação das classes individuais através do cálculo das medidas **micro e macro-média**

Micro-Média vs. Macro-Média

Havendo mais de duas classes apenas a accuracy pode ser generalizada

$$accuracy = \frac{\sum_c TP_c}{N}$$

TP - True Positive para a classe c

C – nº de diferentes classes do conj^{to} teste

N – nº total instâncias do conj^{to} teste

Havendo mais de duas classes a combinação de múltiplas medidas de desempenho (precision e recall) num só valor é feita através da:

- **Macro-média**: Calcula as medidas de desempenho para cada classe, e obtém a média das classes – **dá o mesmo peso a todas as classes**
- **Micro-média**: Faz o cálculos dos valores necessários às medidas para todas as classes, e só depois calcula as medidas de desempenho

Matriz de acertos/erros Multi-classe

		Previsão		
		jovem	adulto	idoso
Real	jovem	8	2	0
	adulto	4	9	3
	idoso	0	3	12

$$Taxa\ acerto = \frac{8 + 9 + 12}{41}$$

Matriz de acertos/erros Multi-classe

		Previsão		
		jovem	adulto	idoso
Real	jovem	8	2	0
	adulto	4	9	3
	idoso	0	3	12

		Previsão	
		jovem	não jovem
Real	jovem	8	2
	não jovem	4	27

		Previsão	
		adulto	não adulto
Real	adulto	9	7
	não adulto	5	20

		Previsão	
		idoso	não idoso
Real	idoso	12	3
	não idoso	3	23

Métricas de Avaliação por Classe

		Previsão				
		jovem	adulto	idoso	<i>Recall</i>	<i>F1-score</i>
Real	jovem	8	2	0	0.8	0.727
	adulto	4	9	3	0.563	0.6
	idoso	0	3	12	0.8	0.8
<i>Precision</i>		0.667	0.643	0.8		

Métricas de Avaliação Gerais do Modelo

		Previsão			Recall	F1-score
		jovem	adulto	idoso		
Real	jovem	8	2	0	0.8	0.727
	adulto	4	9	3	0.563	0.6
	idoso	0	3	12	0.8	0.8
Precision		0.667	0.643	0.8		

Micro F1

$$\text{Total TP} = 8 + 9 + 12 = 29$$

$$\text{Total FP} = 4 + 5 + 3 = 12$$

$$\text{Total FN} = 2 + 7 + 3 = 12$$

$$\text{Precision} = 29/(29+12) = 0.70$$

$$\text{Recall} = 29/(29+12) = 0.70$$

$$\text{F1} = 0.70$$

Macro F1

$$\text{F1 (Jovem)} = 0.73$$

$$\text{F1 (adulto)} = 0.6$$

$$\text{F1 (idoso)} = 0.8$$

$$\text{F1} = (0.73+0.6+0.8)/3=0.71$$

Métodos para Comparar Modelos

Comparação de Modelos

- Dados dois modelos:
 - Modelo M1: *Accuracy* = 85%, testado em 30 instâncias
 - Modelo M2: *Accuracy* = 75%, testado em 5000 instâncias
- Podemos afirmar que M1 é melhor do que M2?
 - Qual a confiança que podemos depositar na exatidão (*accuracy*) de M1 e M2?
 - ♦ Esta questão diz respeito à estimativa do intervalo de confiança da exatidão do modelo
 - Pode a diferença de exatidão (*accuracy*) entre os dois modelos ser explicada como resultado de flutuações aleatórias do conjunto de teste?
 - ♦ Esta questão diz respeito à significância estatística do desvio observado

Curva ROC

Curva ROC (*Receiver Operating Characteristic*) é uma representação gráfica da Taxa dos Verdadeiros Positivos (TPR) (eixo das ordenadas) versus a Taxa dos Falsos Positivos (FPR) (eixo das abcissas)

$$TPR \text{ (true positive rate)} = \frac{TP}{TP + FN} \quad FPR \text{ (false positive rate)} = \frac{FP}{FP + TN}$$

Uma curva ROC sumaria o conj^{to} possível de matrizes de confusão 2×2
O desempenho de cada classificador é representado como um ponto na curva ROC

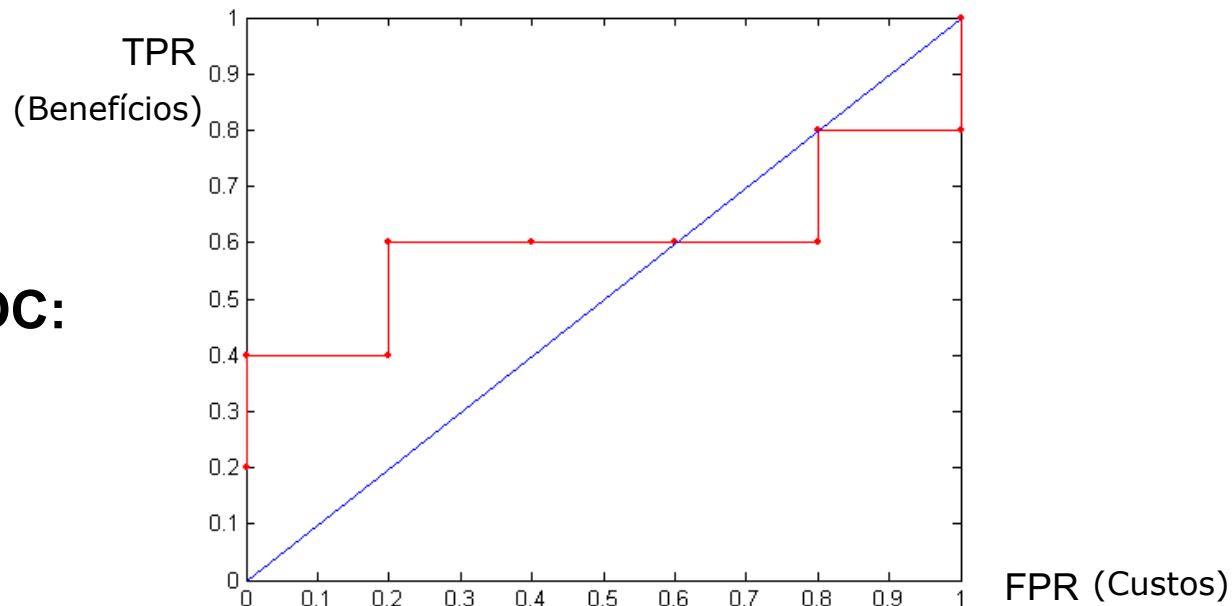
Mudanças:

- no desempenho do algoritmo
- na distribuição da amostra
- na matriz de custo

Muda a localização dos pontos na curva ROC

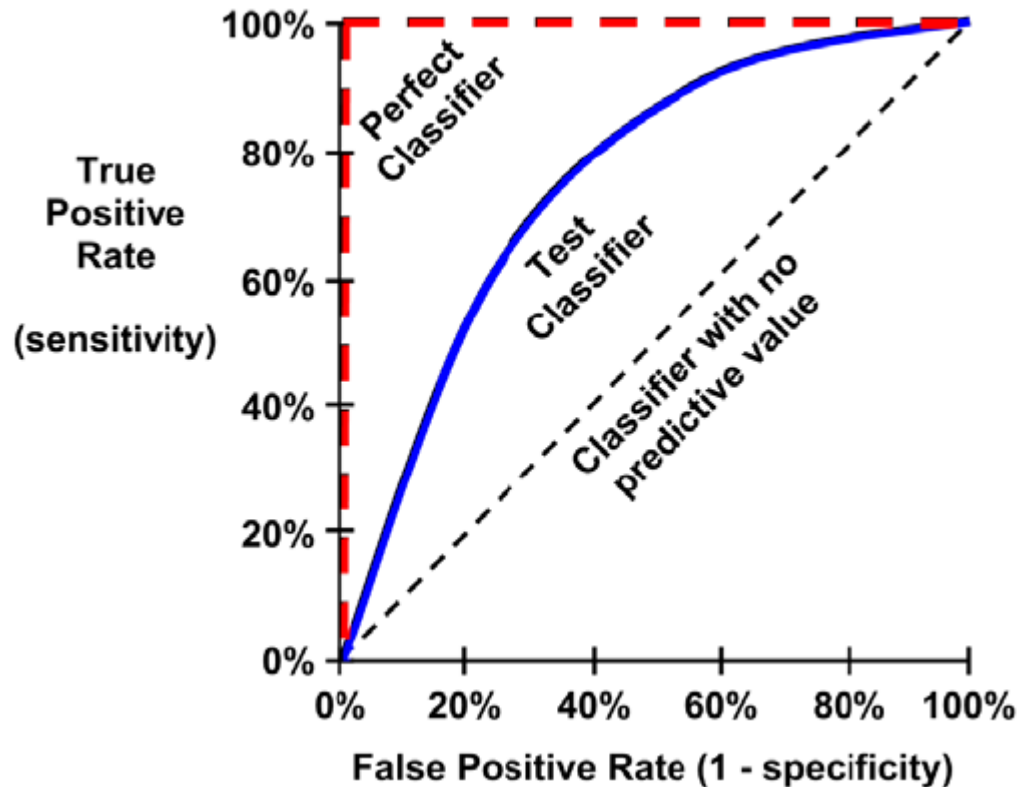
Como Construir uma Curva ROC

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0



Curva ROC:

Curva ROC

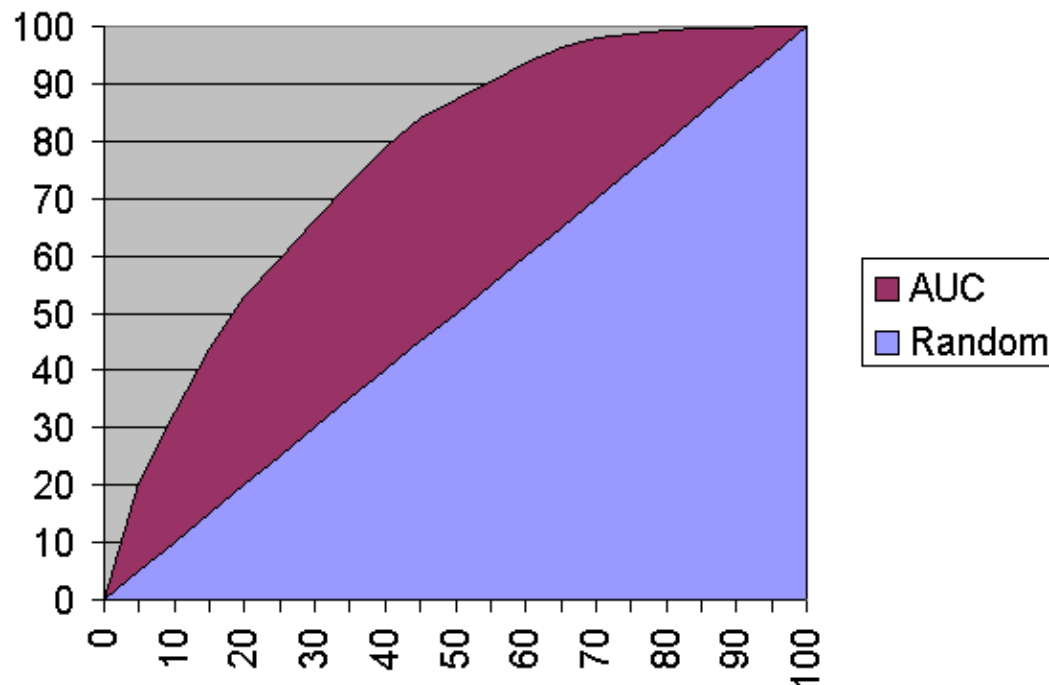


- Linha Diagonal: Aprendizagem aleatória
- Abaixo da linha diagonal: previsão é oposta à classe correta

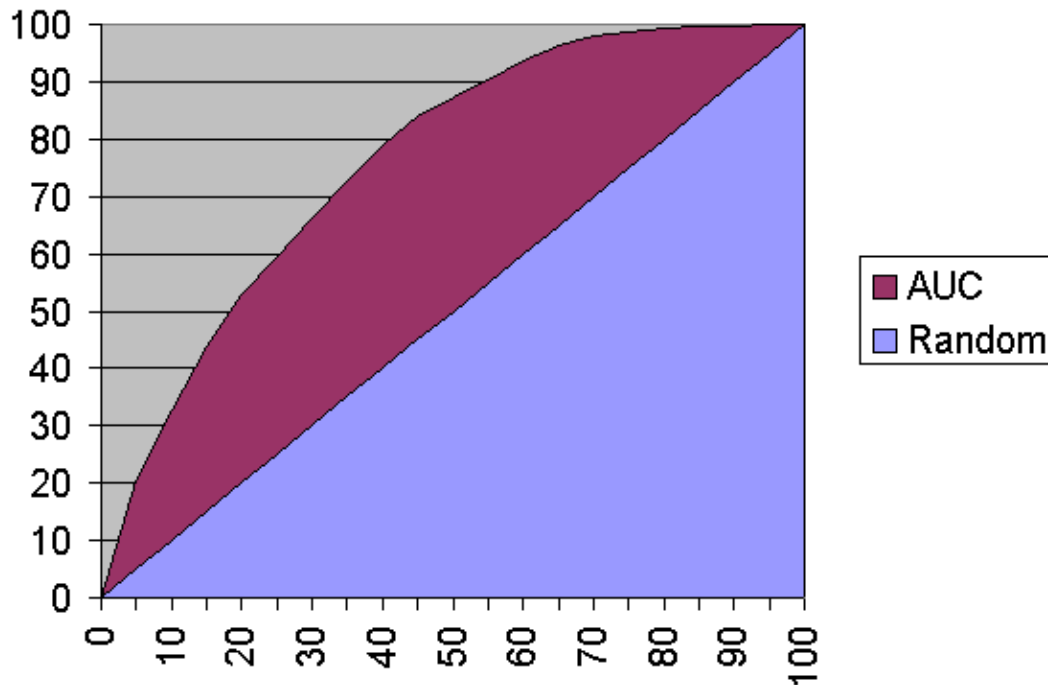
Curva ROC para Comparação de Modelos

Uma curva ROC é uma demonstração bidimensional do desempenho de um classificador. **Para comparar classificadores é preciso reduzir a curva ROC a um valor escalar** - calculo da área abaixo da curva ROC (AUC – *Area Under Curve*)

Como a AUC é uma porção da área do quadrado unitário (espaço ROC) seus valores estão compreendidos entre 0 e 1: $0 < \text{AUC} < 1$

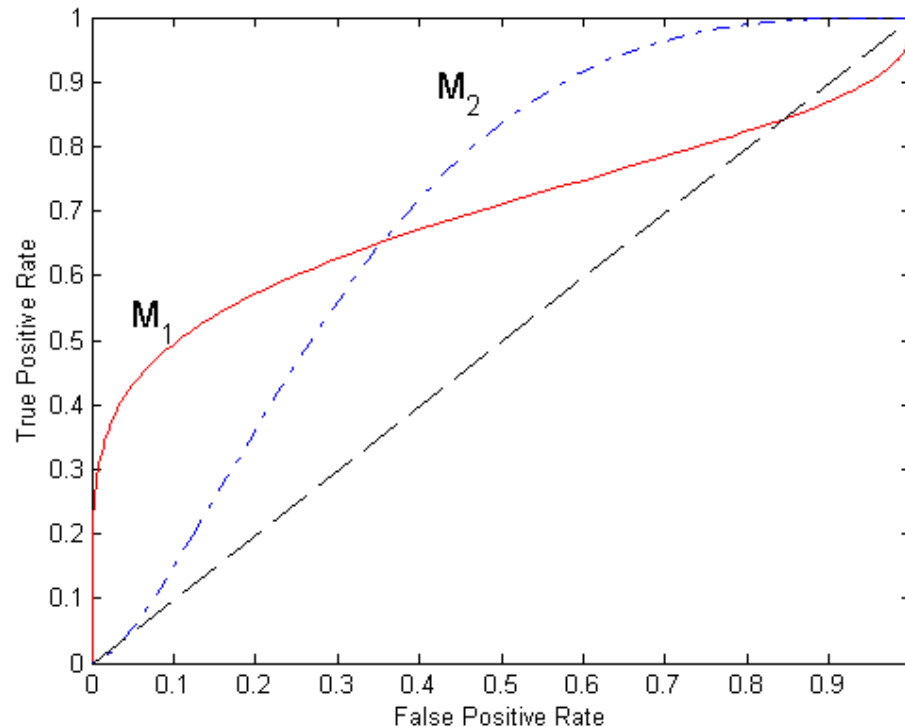


Curva ROC para Comparação de Modelos



Para comparar modelos em que as classes estão desbalanceadas a AUC é mais robusta do que a Accuracy

Curva ROC para Comparação de Modelos



- Se duas curvas ROC não se cruzam, um método domina o outro
- Se duas curvas ROC se cruzam, um método é melhor para alguns rácios de custos, e o outro é melhor para outros rácios de custos

Resumo

- **Métodos de Amostragem**

- Grandes conjuntos de dados
 - ♦ *Holdout*
- Pequenos conjuntos de dados
 - ♦ Método *holdout* Repetido
 - ♦ Validação Cruzada
 - ♦ *Bootstrap*

- **Avaliação Desempenho dos Modelos**

- Classes Balanceadas, sem considerar custos classificação
 - ♦ *Accuracy (exactidão)*, Estatística *Kappa*
- Classes Desbalanceadas, diferentes custos classificação
 - ♦ *Precision, Recall, Medida F1*
 - ♦ *Gráfico Lift*

- **Comparação Desempenho Modelos**

- Curvas ROC – cálculo AUC (*Area Under Curve*)