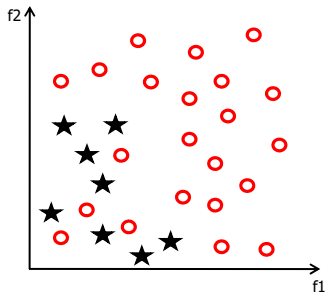
Aprendizado com Classes Desbalanceadas



Stanley R. M. Oliveira

Agenda

- **Classes desbalanceadas**: problema e desafios.
- O algoritmo **k-vizinhos** mais próximos.
- Técnicas para **medir desempenho** de classificadores:
  - Espaço ROC;
  - Geração de curvas ROC;
  - Comparação de curvas ROC.
- Tratamento para **classes desbalanceadas**.
- Sugestões de Leitura.

Introdução

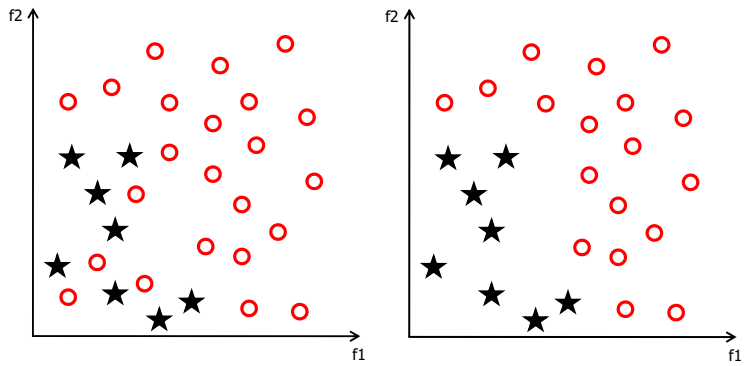
- **Classes desbalanceadas** ocorrem quando existe uma grande desproporção entre o número de exemplos de cada classe.
- Essa situação frequentemente faz com os exemplos da **classe minoritária** sejam **classificados incorretamente**.

Introdução ...

- ASTRAL SCOP Release 1.63
  - 492 estruturas de proteínas;
- Número de classes: seis

Classes de Enzimas	Número de Membros por Classe
Oxidoredutase	77
Transferase	127
Hidrolase	158
Liase	60
Isomerase	51
Ligase	19

Introdução ...



Conjunto de dados desbalanceado

Conjunto de dados com classes bem definidas

Introdução ...

- O problema de **classes desbalanceadas nem sempre é um problema**.
  - Existem conjuntos de dados desbalanceados com boa classificação;
  - Mas a presença de classes desbalanceadas pode tornar o problema mais complicado.
- **Desafio**: ainda existem **dificuldades** em medir o desempenho de classificadores na presença de **classes desbalanceadas**.

Agenda

- **Classes desbalanceadas**: problema e desafios.
- O algoritmo **k-vizinhos** mais próximos.
- Técnicas para **medir desempenho** de classificadores:
  - Espaço ROC;
  - Geração de curvas ROC;
  - Comparação de curvas ROC.
- Tratamento para **classes desbalanceadas**.
- Sugestões de Leitura.

k-Vizinhos Mais Próximos

- **Aprendizado baseado em instâncias** simplesmente armazena os exemplos de treinamento.
- Quando um **novo exemplo** (**caso**) precisa ser classificado, esse exemplo é **comparado** com os **exemplos armazenados**.
- A **classificação** é decidida a partir da **similaridade** entre o exemplo a ser classificado e os exemplos armazenados.

k-Vizinhos Mais Próximos ...

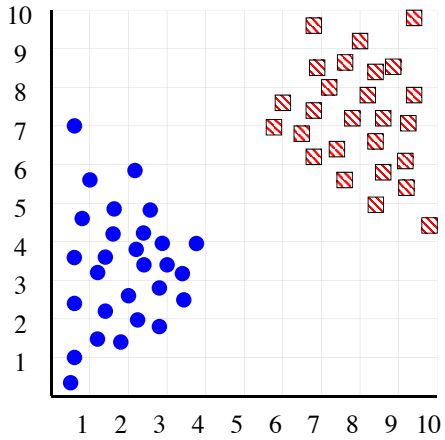
- Aprendizado baseados em instâncias é chamado também de aprendizado "***lazy***".
- Isso se deve ao fato do **processamento** ser **atrasado** até o momento de classificação de um novo exemplo.

k-Vizinhos Mais Próximos ...

- Dois **métodos** bastante conhecidos de aprendizado baseado em instâncias são:
  - k-vizinhos mais próximos;
  - Regressão com pesos locais.
- A ideia do **método k-vizinhos** mais próximos é bastante simples...

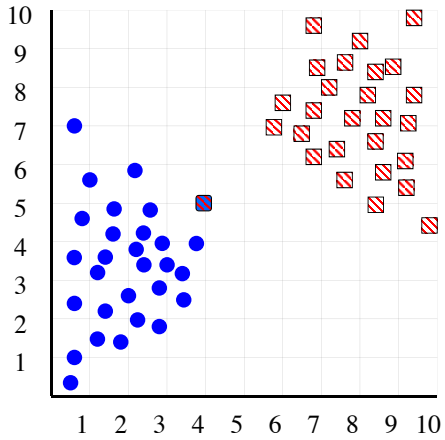
k-Vizinhos Mais Próximos ...

- Inicialmente, os exemplos de treinamento são armazenados.



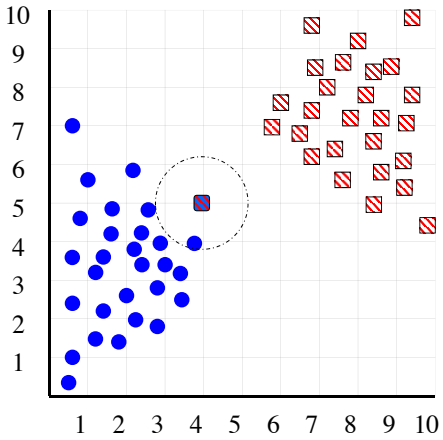
k-Vizinhos Mais Próximos ...

- Inicialmente, os exemplos de treinamento são armazenados.
- Quando um novo exemplo precisa ser classificado...



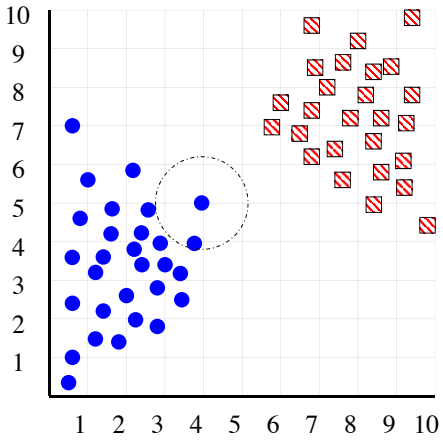
k-Vizinhos Mais Próximos ...

- Inicialmente, os exemplos de treinamento são armazenados.
- Quando uma **novo** exemplo precisa ser classificado, é verificada a sua **similaridade**.



k-Vizinhos Mais Próximos ...

- Por fim, o **novo exemplo** é classificado segundo a sua **proximidade** com os exemplos de treinamento.



K-Vizinhos mais Próximos: Parâmetro *k*

- O parâmetro ***k*** do algoritmo **k-vizinhos** mais próximos é o **número de vizinhos** a serem considerados na classificação.
- O parâmetro ***k*** é geralmente um inteiro pequeno e ímpar (1,3,5,7,9) para evitar empates.
- Portanto, o **3-vizinhos** mais próximos utiliza na classificação os **3 exemplos** mais **próximos** do **novo exemplo**.

K-Vizinhos mais Próximos: Parâmetro *k*

- Quando ***k* = 1** (**1-vizinho mais próximo**) apenas o exemplo mais próximo ao exemplo a ser classificado é considerado.
- O uso de ***k* = 1** pode levar a classificações incorretas caso existam exemplos com ruído no conjunto de treinamento.

K-Vizinhos mais Próximos: Distância

- **K-vizinhos** mais próximos assume que todos os exemplos correspondem a pontos no espaço *n*-dimensional $\Re^n$.
- Os **vizinhos mais próximos** de um exemplo são definidos por uma medida de distância, tipicamente a **distância euclidiana**.

K-Vizinhos mais Próximos: Distância

Table 2: Data set in attribute-value form.

	A_1	A_2	$\cdots$	A_M	Y
E_1	x_{11}	x_{12}	$\cdots$	x_{1M}	y_1
E_2	x_{21}	x_{22}	$\cdots$	x_{2M}	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	$\cdots$	x_{NM}	y_N

K-Vizinhos mais Próximos: Distância

- A **distância euclidiana** entre dois exemplos E_i e E_j é definida por

$$d(E_i, E_j) = \sqrt{\sum_{r=1}^{M} (x_{ir} - x_{jr})^2}$$

- Os **k-vizinhos** mais próximos podem ser representados tanto por um **atributo classe** (*Y*) **contínuo** quanto **discreto**.

K-Vizinhos mais Próximos: Classificação

- Para o caso no qual ***Y* é discreto** (**problema de classificação**), então deve-se encontrar os *k* exemplos mais próximos do exemplo a ser classificado.
- Dentre os ***k* exemplos**, verifica-se a **classe mais frequente**. Essa classe é atribuída ao novo exemplo.

K-Vizinhos mais Próximos: Classificação

- **Algoritmo de treinamento**:
  - Armazenar todos os exemplos de treinamento em um conjunto *CT*.
- **Algoritmo de classificação**:
  - Dado um novo exemplo E_{novo};
  - Sejam $E'_1,...E'_k$ os *k* exemplos em *CT* mais próximos a E_{novo}.
  - Retornar a classe que ocorre com maior frequência nos exemplo $E'_1,...E'_k$.

K-Vizinhos mais Próximos: Regressão

- O **k-vizinhos** mais próximos pode ser **adaptado** para aproximar um **atributo** ***Y*** com **valores contínuos**.
- Para isso, basta retornar a **média** dos valores do **variável** ***Y*** para os **k-vizinhos mais próximos**.
- Por exemplo, pense em uma aplicação que precisa fornecer o **salário de um funcionário** dada a sua titulação, experiência, cargo, etc.

KNN, RNA e SVM são os únicos que fazem classificação e regressão

K-Vizinhos mais Próximos: Exemplo 1

- Calcule a distância entre o exemplo **Novo** e os demais:

Nr.Ex.	At.1	At.2	At.3	Classe
1	5	1	90	+
2	2	3	105	-
3	1	4	120	+
Novo	**3**	**3**	**100**	**?**

K-Vizinhos mais Próximos: Normalização

- Segundo a distância euclidiana, a **distância** entre o **exemplo 1** e **Novo** é:

$$d(1,novo) = \sqrt{(5-3)^2 + (1-3)^2 + (90-100)^2}$$

- Para resultado final, **10.39**, todos os atributos contribuíram igualmente?

$$= \sqrt{2^2 + (-2)^2 + (-10)^2} = \sqrt{4+4+100} \cong 10.39$$

não contribuem, os atributos de maior range, contribuem mais. por isso, normalizamos os dados no início.

K-Vizinhos mais Próximos: Normalização

- Alguns **atributos** assumem uma **faixa de valores mais ampla** do que outros.
- Para **evitar** que esses atributos tenham uma **influência** maior, deve ser realizada uma **normalização**.
- A **normalização** faz com que todos os atributos fiquem na **mesma faixa** de valores.

K-Vizinhos mais Próximos: Normalização

- Existem diversas formas de **normalização**, uma das mais utilizadas é a **normalização linear** para o intervalo [0,1]:

$$v_n = \frac{v_i - \min}{\max - \min}$$

- Onde:
  - v_n é o valor normalizado;
  - v_i é o valor não normalizado;
  - ***min*** e ***max*** são os valores mínimo e máximo do atributo.

K-Vizinhos mais Próximos: Exemplo 2

- **Normalize** os atributos da tabela do exemplo 1 segundo a normalização linear:

Nr.Ex.	At.1	At.2	At.3	Classe
1	5	1	90	+
2	2	3	105	-
3	1	4	120	+
Novo	**3**	**3**	**100**	**?**

K-Vizinhos mais Próximos: Exemplo 3

- Calcule a **distância** entre o exemplo **Novo** e o **exemplo 1**, dada a tabela normalizada:

Nr.Ex.	At.1	At.2	At.3	Classe
1	1	0	0	+
2	0.25	0.66	0.5	-
3	0	1	1	+
Novo	**0.5**	**0.66**	**0.33**	**?**

K-Vizinhos: Atributos Discretos

- Um segundo problema com a distância euclidiana é o **cálculo com atributos discretos**.

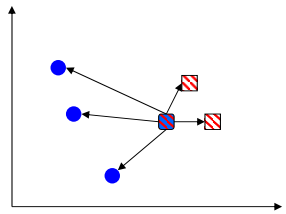
Nr.Ex.	At.1	At.2	At.3	Classe
1	azul	novo	ford	+
2	verde	novo	gm	-
3	verde	antigo	volks	+
Novo	**azul**	**antigo**	**gm**	**?**

K-Vizinhos: Atributos Discretos

- Uma forma simples de solucionar esse problema é utilizar a **medida *overlap***. Nessa medida, a distância é zero se os valores são iguais ou 1 se são diferentes.
- Uma forma mais sofisticada é a ***Value Difference Metric (VDM)*** (**Stanfil, 1986**).

K-Vizinhos mais Próximos: Pesos

- Uma **variação** popular do algoritmo **k-vizinhos** mais próximos é utilizar um **peso** para cada vizinho proporcional à sua **distância**.



K-Vizinhos mais Próximos: Pesos

- Uma sugestão é **ajustar o peso** do voto de cada vizinho pela equação:

$$w_i = \frac{1}{d(E_{novo}, E_i)^2}$$

- Dessa forma, **quanto mais distante** estiver o vizinho mais próximo (E_i) do exemplo a ser classificado E_{novo}, **menor será o seu peso**.

K-Vizinhos mais Próximos: Pesos

- Quando se utiliza os **pesos** para **decidir a classe**, pode-se deixar de usar apenas os *k* vizinhos mais próximos, e passar a utilizar **todo o conjunto de treinamento**.
- Isso porque exemplos **muito distantes** terão **pouca influência** na classificação do novo exemplo.

K-Vizinhos: Observações

- **K-vizinhos** mais próximos é um método bastante **simples**, mas que provê **bons resultados** na prática;
- Ele é **robusto** a **ruído** e **bastante efetivo** quando o conjunto de treinamento não é muito pequeno;
- A **melhor explicação** que o método pode prover é mostrar ao usuário os vizinhos mais próximos do novo exemplo quando o **método é local**.
- O **grau de explicação** desse método pode ser considerada **inferior** ao dos **métodos simbólicos**.

K-Vizinhos: Observações ...

- O **k-vizinhos** mais próximos considera **todos os atributos** ao classificar um novo exemplo. Isso pode ser um **sério problema** quando existem muitos **atributos irrelevantes**.
- Para solucionar o problema de **atributos irrelevantes** pode-se utilizar **pesos** para os atributos na medida de distância;
- Outro problema é o **desempenho** (**em tempo de execução**) para classificar novos casos quando o conjunto de treinamento é muito grande.

Agenda

- **Classes desbalanceadas**: problema e desafios.
- O algoritmo **k-vizinhos** mais próximos.
- Técnicas para **medir desempenho** de classificadores:
  - Espaço ROC;
  - Geração de curvas ROC;
  - Comparação de curvas ROC.
- Tratamento para **classes desbalanceadas**.
- Sugestões de Leitura.

Taxa de acerto

- A **avaliação de classificadores** geralmente é feita a partir da **taxa de erro/acerto** ou **acurácia**.
- **Acurácia**: mede a porcentagem de exemplos classificados corretamente.
- Um classificador com 99,9% de acerto (0,1% de erro) é um **bom** ou **mal classificador**?

  pode acertar a classe majoritária, mas errar a minoritária, que muitas vezes é a de interesse

Taxa de acerto ...

- A taxa de acerto **não** leva em consideração a **prevalência** da classe majoritária.
- Se a classe majoritária tem uma prevalência de **99,9%** é trivial construir um classificador que **acerte 99,9%** das vezes (**simplesmente prediga a classe majoritária**).

Taxa de acerto pode confundir

- **Exemplo**:
  - C1 = pacientes com câncer (**4 pacientes**);
  - C2 = pacientes saudáveis (**500 pacientes**);
  - Acurácia do modelo = 90%;
  - Classificou corretamente 454 pacientes que não tem câncer;
  - **Não** acertou nenhum dos que tem câncer,
- Pode ser considerado um "**bom classificador**"?

Olhando mais de perto

- **Matriz de confusão** tabula os **erros/acertos** para cada uma das classes.
- Para um problema de duas classes:

Classe	Predita Positiva	Predita Negativa
Real Positiva	Verdadeiro Positivo (**TP**)	Falso Negativo (**FN**)
Real Negativa	Falso Positivo (**FP**)	Verdadeiro Negativo (**TN**)

Olhando mais de perto ...

- A taxa de acerto é dada por

$$\frac{TP + TN}{TotalExemplos}$$

- Se uma das classes (**digamos a negativa**) tem uma alta prevalência, ela pode "**mascarar**" a taxa de acerto.

Alguns exemplos

- Ambos têm a mesma taxa de acerto de 99,9%, mas...

Classe	PPos	PNeg
Pos	0	10
Neg	0	9.99

Classe	PPos	PPeg
Pos	0.01	0
Neg	10	9.89

- No primeiro caso, **nenhum** exemplo da classe positiva é classificado corretamente!

Classe menos frequente

- A classe **menos frequente** (**normalmente chamada de positiva**) é geralmente a de maior interesse:
  - Doença rara;
  - Ocorrência de geada;
  - Transação ~~falsa~~ em cartão de crédito, etc.
    fraudulenta
- Um **classificador** que erra muito a classe positiva é de **pouca utilidade**.

Taxa de acerto por classe

- Uma outra alternativa é calcular a **taxa de acerto por classe:**

$$TxAcerto_{pos} = \frac{TP}{TP + FP}$$

- De todos os exemplos que são **classificados como positivo**, quantos estão corretos?
- Uma **desvantagem** é que temos agora uma medida de desempenho para cada classe (**é mais difícil comparar**).

Proporção de exemplos variável

- Proporção de exemplos entre as classes pode mudar:
  - Epidemia;
  - Mudança climática.
- **Taxa de acerto** é dependente da **proporção** de exemplos entre as classes.
- Ela envolve duas linhas diferentes da matriz de confusão.

Proporção de exemplos variável ...

- Caso a **proporção de exemplos** entre as classes **mude**, a **taxa de acerto** também vai **mudar**.
- Entretanto, a "**habilidade**" do classificador de identificar as classes **não** foi necessariamente alterada.

Exemplo – Filtro de Spam

- Imagine que um filtro tenha a seguinte matriz de confusão.
- Taxa de acerto = 85,1%

	Predito Spam	Predito Não Spam
Spam	TP 10	FN 5
Não Spam	FP 2	TN 30

- Se em um "**ataque**" de spammers, os spans triplicam.
- Taxa de acerto = 77,9%

	Predito Spam	Predito Não Spam
Spam	30	15
Não Spam	2	30

I) 10/12 = 83.33%
II) 30/32 = 93.75%
Apesar de o primeiro ser melhor global, o segundo é melhor para predizer spam

Taxa de verdadeiros/falsos positivos

- São "**independentes**" da proporção de exemplos entre as classes

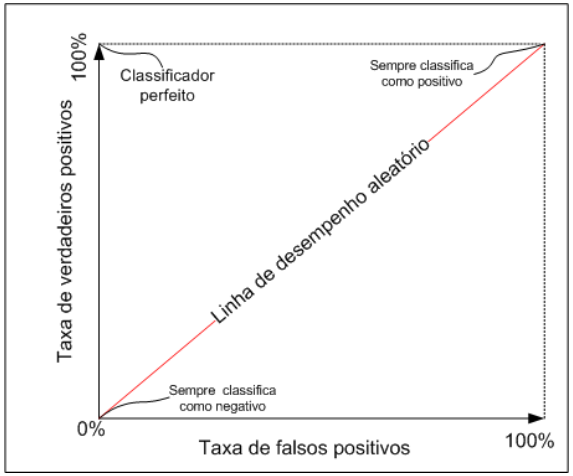
$$Tpr = \frac{TP}{TP + FN}$$

- De todos os exemplos positivos, quantos eu acertei?

Espaço ROC

- **ROC** (**Receiver Operating Characteristics**): termo usado em detecção de sinais para caracterizar a relação de perda e ganho entre a taxa de acerto e a taxa de falso alarme em um canal de ruído.
- **ROC** é uma ferramenta muito útil na **análise** e **comparação** de classificadores.
- Formado pela taxa de **falsos positivos** no **eixo *x*** e a de **verdadeiros positivos** no **eixo y**.
- Cada taxa de **classificador** corresponde a **um ponto** no espaço **ROC**.

Espaço ROC ...

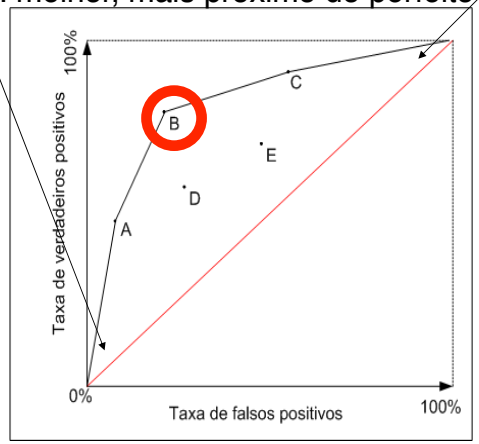


- **Importante**: um classificador que aparece **abaixo** da **diagonal principal** é pior que o desempenho aleatório.

Espaço ROC ...

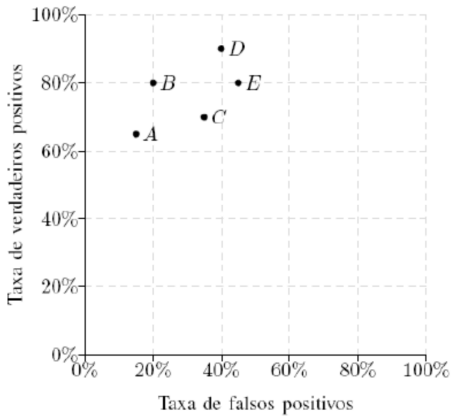
B: melhor, mais próximo do perfeito

Conservador: classificador que aceita poucos "**False Positives**", mas consequentemente penaliza bastante o desempenho dos "**True Positives**"



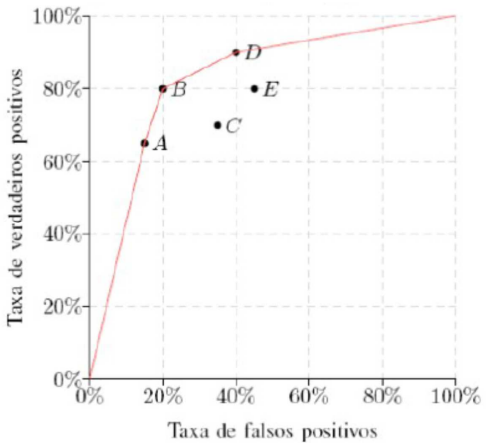
Liberal: classificador que não se importa muito em aceitar bastante "**False Positives**". Por outro lado, seu desempenho nos "**True Positives**" é muito bom.

Espaço ROC ...



- A Figura ao lado mostra um gráfico ROC com 5 pontos representando 5 modelos de classificação diferentes (**A**, **B**, **C**, **D** e **E**).
- O classificador **A** é o mais conservativo e **D** é o mais liberal.

Espaço ROC ...



• Os modelos que se encontram no envelope externo convexo (**convex hull**), que mais se aproximam do ponto (0, 100), são considerados os melhores (**A**, **B** e **D**).

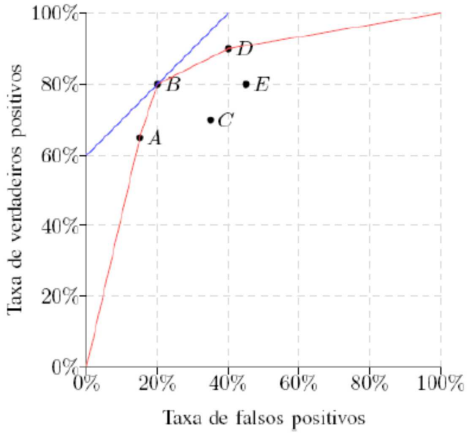
• Os outros modelos (**C** e **E**) são descartados.

Convex Hull

- **Convex hull** é o menor polígono que, dado um conjunto disperso de pontos S, consegue abranger no seu interior todos os pontos do conjunto S com o menor numero de arestas.
- Os classificadores "**interiores**" ao fecho convexo (**convex hull**) **não** podem ter uma **taxa de acerto maior** do que os que estão no **fecho convexo**.
- **Cada ponto** no **fecho convexo** pode ter a **maior taxa de acerto**, dependendo da proporção de exemplos entre as classes.

no exemplo dado, é fácil de visualizar. quando temos mais pontos, se torna mais difícil.

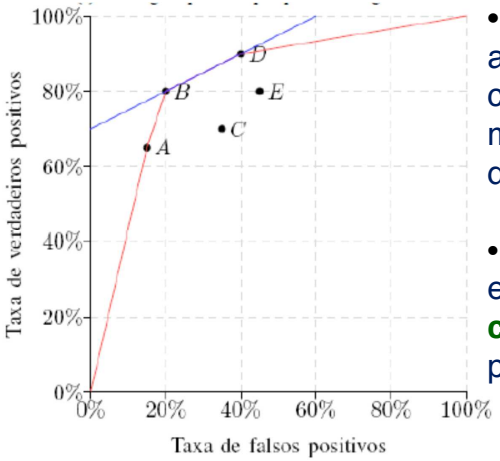
ROC: linha de isodesempenho



• Nessas condições, o modelo de classificação **B** irá apresentar a **menor taxa de erro** (ou o menor custo de classificação).

• Custos iguais para exemplos **positivos** e **negativos**.

ROC: linha de isodesempenho ...



• Nessas condições, ambos os modelos de classificação **B** e **D** têm a mesma taxa de erro/custo de classificação global.

• No entanto, as taxas de erro separadas **por classes** são diferentes para **B** e **D**.

Gerando uma Curva ROC ...

- D = conjunto de amostras classificadas.
- Amostragem de D = (Tr, Te)
  - Tr = Treinamento ; Te = Testes ; Tr U Te = D.
- Uma amostragem (Tr, Te) induz um **modelo M** do classificador.
- Classificação de uma amostra X
  - P_i = probabilidade de X ser classificada na classe c_i = porcentagem de modelos que classifica X na classe c_i

Gerando uma Curva ROC

- O classificador precisa produzir, para cada **tupla X** (ou instância), a probabilidade que a **tupla X** seja classificada na classe **Positiva**.
- Classificadores como **redes neurais** e **redes bayesianas** produzem tais probabilidades.
- Para outros tipos de classificadores, é preciso calcular esta probabilidade.

Gerando uma Curva ROC ...

- Escolhe-se aleatoriamente m amostras da massa de dados: x1, ..., xm
- Calcula-se p_i = probabilidade de x_i ser classificada na classe ***positiva***.
- Ordena-se as amostras xi por ordem crescente das probabilidades
  - x1, x2, ..., xm
- Existem modelos M_1, M_2, ..., M_m, M_{m+1} tais que:
  - M1: Classificam todos os xi como positivos
  - M2: Classificam um como negativo e os outros como positivos
  - ... Mi: Classificam (i-1) como negativos e os outros como positivos
- Logo, é razoável supor que:

Modelo	Negativos	Positivos
M_1	nenhum	$\{x_1, ..., x_m\}$
M_2	$\{x_1\}$	$\{x_2, ..., x_m\}$
M_2	$\{x_1, x_2\}$	$\{x_3, ..., x_m\}$
...	...	...
M_m	$\{x_1, ..., x_{m-1}\}$	$\{x_m\}$
M_{m+1}	$\{x_1, ..., x_m\}$	nenhum

Gerando uma Curva ROC ...

- Para o M_1: TP = número de amostras positivas e FP = número de amostras negativas. $TN = 0$ e $FN = 0$. Logo TPR = 1 e FPR = 1.
- Para cada M_i $(i > 1)$: verifica-se a classe real da amostra x_{i-1}:
  - se for positiva:
    - TP de M_i = TP de $M_{i-1} - 1$
    - FP de M_i = FP de M_{i-1}.
    - TN de M_i = TN de M_{i-1}
    - FN de M_i = FN de $M_{i-1} + 1$.
  - se for negativa:
    - TP de M_i = TP de M_{i-1}
    - FP de M_i = FP de $M_{i-1} - 1$.
    - TN de M_i = TN de $M_{i-1} + 1$
    - FN de M_i = FN de M_{i-1}.

Exemplo: Curva ROC

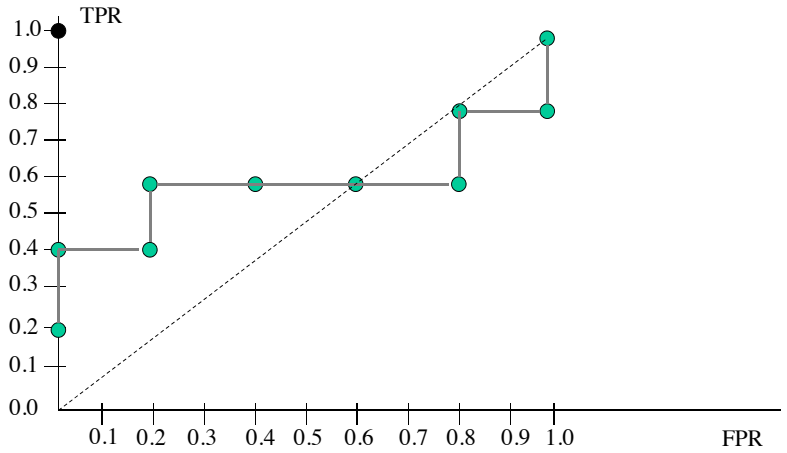
Classe	+	-	+	-	-	-	+	-	+	+	
	0.25	**0.43**	**0.53**	**0.76**	**0.85**	**0.85**	**0.85**	**0.87**	**0.93**	**0.95**	**1.00**

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

começa em 5, termina em 0. começa em 0, termina em 5.
como rola isso aí?

Exemplo: Curva ROC ...

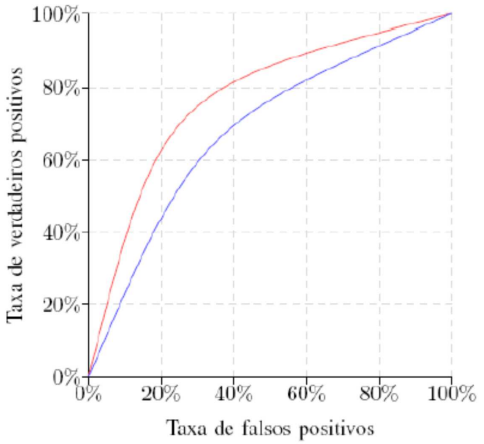
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0



Exercício sobre Curva ROC

- Dado o conjunto de amostras na planilha **ROC_Exercício.xls**, em ordem crescente de probabilidades, preencher os valores da planilha e construir o **gráfico ROC** correspondente.
- Em seguida, calcule a **área abaixo** da **curva ROC**.

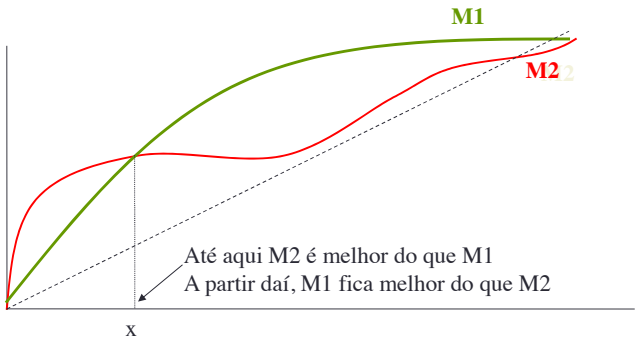
ROC: comparação de duas curvas



- Ao se comparar **duas ou mais** curvas ROC, caso não haja nenhuma intersecção, a curva que mais se aproximar do ponto (0, 100) é a de melhor desempenho.
- Idealmente, a curva deve ser **convexa** e sempre crescente.

ROC: comparação de duas curvas

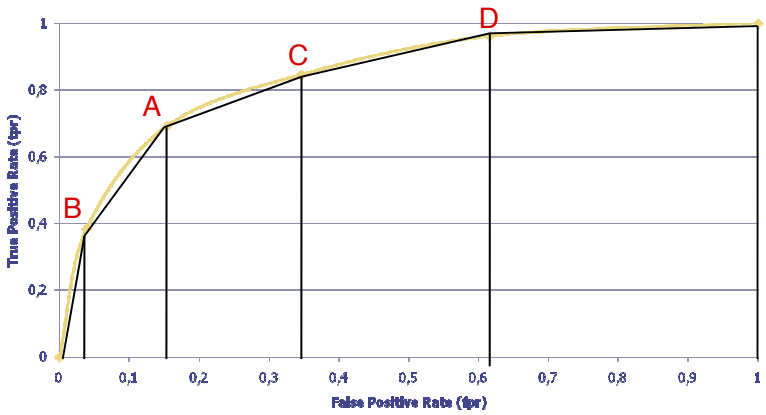
- **Curvas ROC** são utilizadas para se medir a performance relativa de diferentes classificadores.



AUC – Area Under the Curve

- Uma maneira de expressar o **desempenho** do classificador com um **único número** é calcular a **área** abaixo da **curva ROC**.
- **Probabilidade** que um **exemplo positivo** vai estar **ranqueado** acima de um exemplo negativo.
- É uma medida de "**quão bem**" o classificador é capaz de separar as duas classes.
- Pode ser calculado pela **regra do trapézio**.
- Quanto **maior** a área, **melhor** é o desempenho médio do classificador.

AUC – regra do trapézio



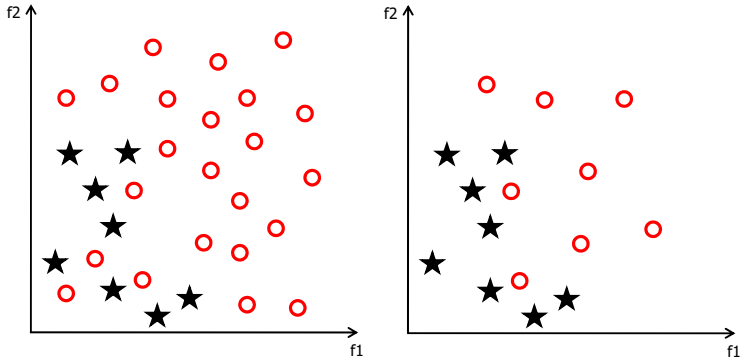
Agenda

- **Classes desbalanceadas**: problema e desafios.
- O algoritmo **k-vizinhos** mais próximos.
- Técnicas para **medir desempenho** de classificadores:
  - Espaço ROC;
  - Geração de curvas ROC;
  - Comparação de curvas ROC.
- Tratamento para **classes desbalanceadas**.
- Sugestões de Leitura.

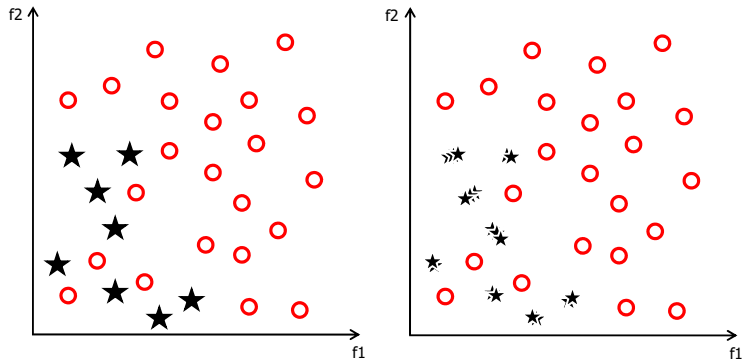
Random Under e Over-sampling

- A forma mais direta de tratar o problema de classes desbalanceadas é **replicar** ou **eliminar** exemplos:
  - **Random Under-sampling**: elimina aleatoriamente exemplos da classe majoritária;
  - **Random Over-sampling**: replica aleatoriamente exemplos da classe minoritária. overfitting

Random Under-sampling



Random Over-sampling



Random Under e Over-sampling

- Ambos os **métodos** são **simples**, **rápidos** e podem gerar qualquer distribuição entre as classes.
- **Entretanto**:
  - **Random Under-sampling** pode **eliminar** exemplos importantes para o aprendizado;
  - **Random Over-sampling** pode causar ***overfitting***.

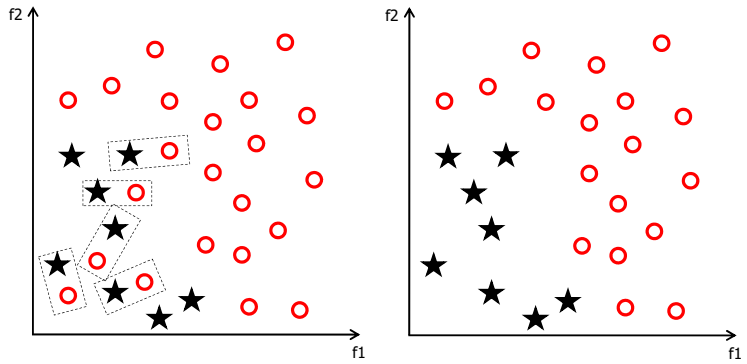
Random Under e Over-sampling

- Com o objetivo de **sanar as limitações** de ambos os métodos, foram propostos diversos métodos que utilizam heurísticas para **inserir** ou **remover** exemplos.
- Diversos desses métodos utilizam o algoritmo **k-Vizinhos mais próximos** como base.

Tomek Links

- Dados dois exemplos E_i e E_j, o par (E_i, E_j) forma um **Tomek Link** se:
  - E_i e E_j são de classes diferentes, e;
  - Não existe um exemplo E_k tal que:
    - $d(E_k, E_i) < d(E_j, E_i)$;
    - $d(E_k, E_j) < d(E_j, E_i)$

Tomek Links



Tomek Links

- Se (E_i, E_j) forma um Tomek Link então:
  - E_i ou E_j é um exemplo com ruído;
  - E_i e E_j são exemplos próximos à borda.
- **Tomek** links podem ser utilizado como:
  - método de **limpeza** de dados: ambos E_i e E_j são removidos;
  - método de **under-sampling**: somente o exemplo da classe majoritária é removido.

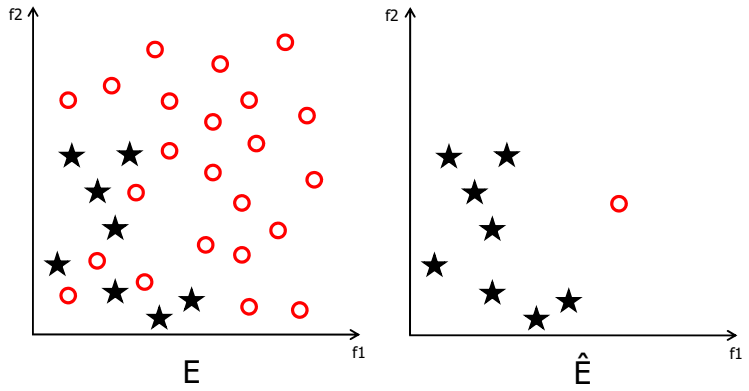
Condensed Nearest Neighbor Rule (CNN)

- Um conjunto de exemplo Ê ⊆ E é consistente com E se, utilizando 1-NN, Ê classifica corretamente os exemplos de E.
- Um algoritmo para encontrar um subconjunto consistente como método de **under-sampling** é descrito em [**Kubat & Matwin, 1997**]

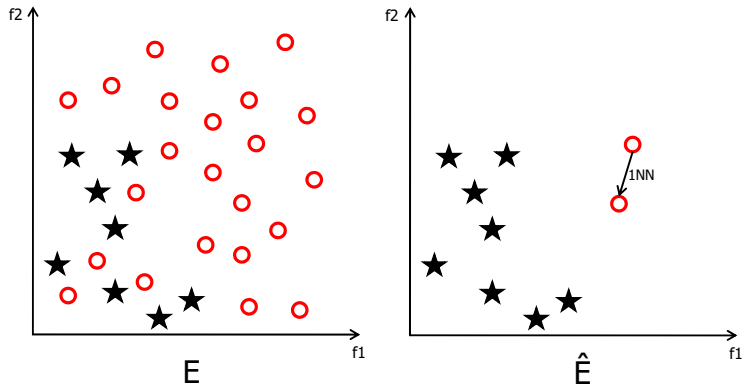
Consistent Subsets

- **Algoritmo**:
  - Inicia-se com 1 exemplo da classe majoritária e todos os exemplos da classe minoritária em Ê;
  - Utiliza-se os exemplos de Ê para classificar os exemplos em E;
  - Cada exemplo classificado incorretamente é inserido em Ê.

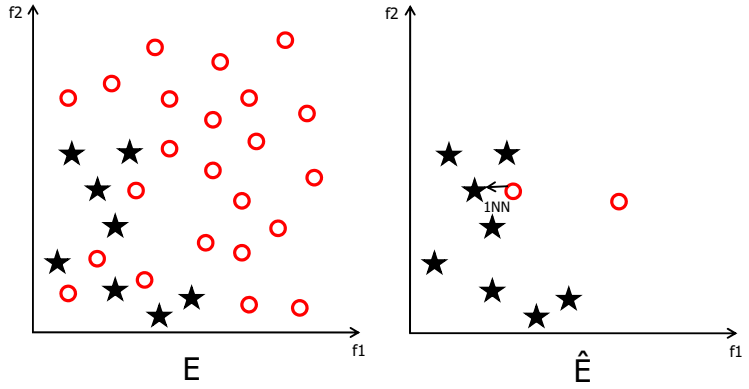
Consistent Subset



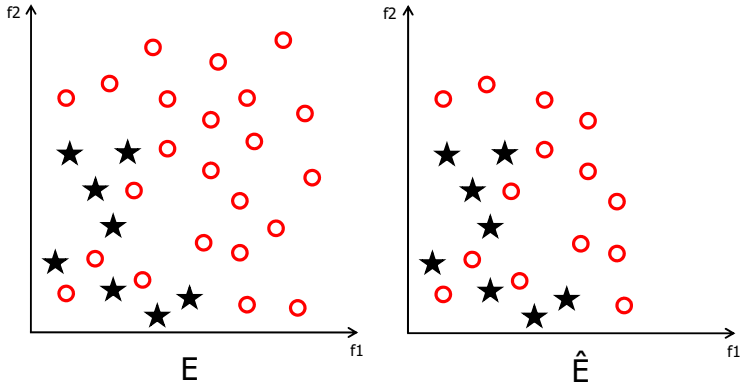
Consistent Subset



Consistent Subset



Consistent Subset

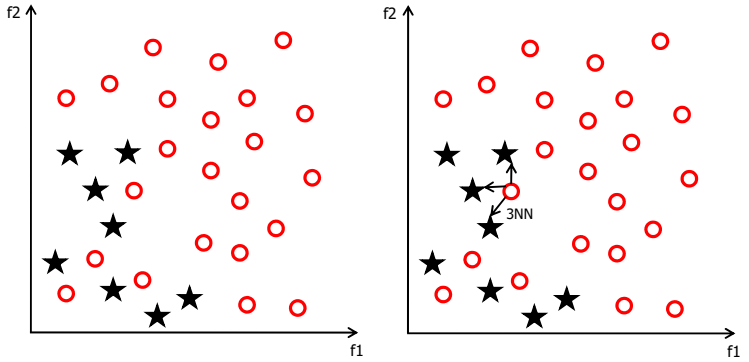


Edited Nearest Neighbor Rule (ENN)

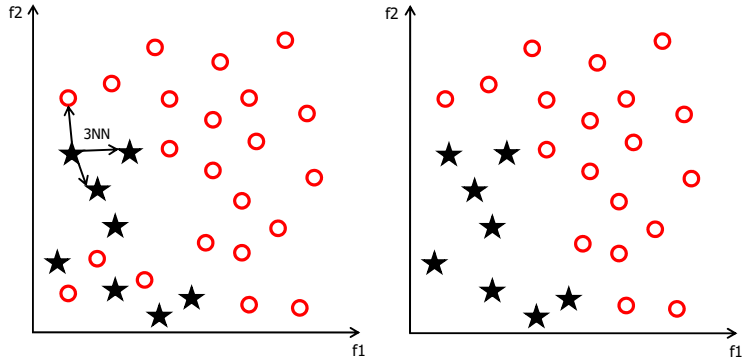
- É um **método de limpeza** de dados.
- Consiste em:
  - Para cada exemplo do conjunto de dados, identifica-se os 3 vizinhos mais próximos a um exemplo E_i;
  - Se 2 ou mais dos 3 vizinhos são de classes diferentes da classe de E_i, então E_i é removido.

tendência: tirar classe majoritária

Edited Nearest Neighbor Rule (ENN)



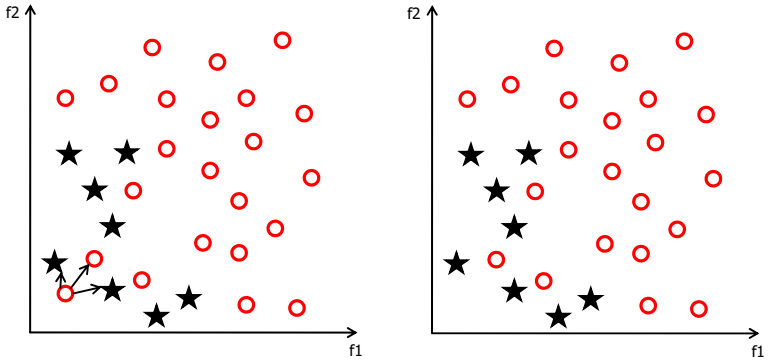
Edited Nearest Neighbor Rule (ENN)



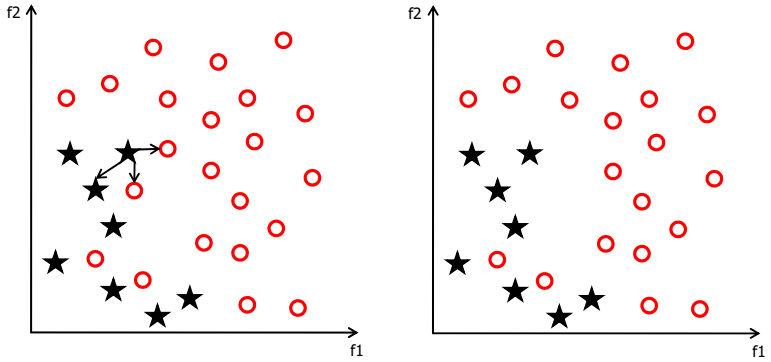
Neighborhood Cleaning Rule

- **NCL** modifica **ENN** para aumentar a limpeza dos dados e se tornar um método de **under-sampling**:
  - Se o exemplo E_i é da classe majoritária, e é contradito pelo 3 vizinhos mais próximos, então E_i é removido;
  - Se E_i é da classe minoritária, e é contradito pelos 3 vizinhos mais próximos, então os exemplos da classe majoritária que contradizem E_i são removidos.

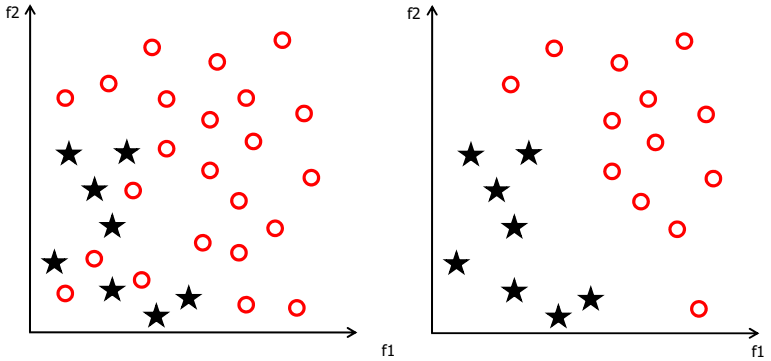
Neighborhood Cleaning Rule



Neighborhood Cleaning Rule



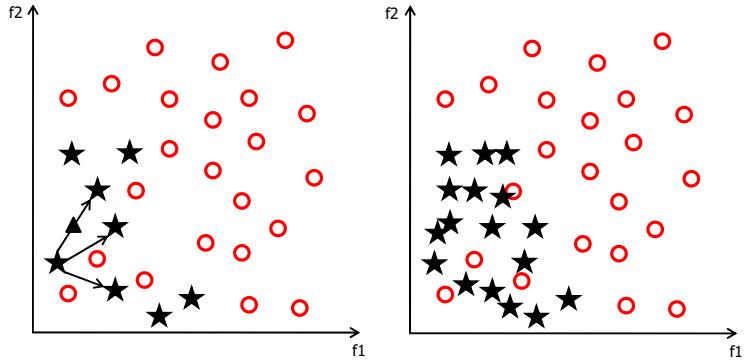
Neighborhood Cleaning Rule



SMOTE

- **SMOTE** é um método de **over-sampling**:
  - Para **evitar** que **exemplos** sejam **replicados**, é feita uma **interpolação** entre dois exemplos próximos da classe minoritária;
  - Utilizando **k-Vizinhos** mais próximos, alguns dos vizinhos são selecionados e um novo exemplo é gerado por meio de **interpolação**.

SMOTE



Referências

- Batista, G.; Prati, R.; Monard M. *A **Study of the Behavior of Several Methods for Balancing Machine Learning Training Data***. SIGKDD Explorations, v. 6, n. 1, p. 20-29, 2004.
- Bentley, J. ***Multimensional binary search trees used for associative searching***. Communications of the ACM, 18(9), 509-517, 1975.
- Ciaccia, P.; Patella, M.; Zezula, P. ***M-tree: an efficient access method for similarity search in metric spaces***. Proceedings of VLDB, 426-435, 1997.
- Mitchell, T. *Machine Learning*. McGraw-Hill, 1997.
- Traina, Jr., C.; Traina, A.; Seeger, B.; Faloutsos, C. ***Slim-trees: high performance metric trees minimizing overlap between nodes***. Proceedings of ETBT, 51-65, 2000.
- Shepard, D. ***A two-dimensional interpolation function for irregularly spaced data***. Proceedings of the 23rd National Conference of the ACM, 517-523, 1968.
- Stanfil, C.; Waltz, D. ***Towards Memory-based reasoning***. Communications of the ACM, 29, 1213-1228, 1986.