(3)DC(p,q)=maxi(|pi,qi|)
=
(
q
1
,
…
,
q
n
)

Organização e tabulação dos dados

Published on 2016 M07 28

Andressa Kutschenko Nahas Follow
Consultora Estatística nas empresas PH3A e AKN Consultoria Estatística

E então você terminou a coleta dos dados da sua pesquisa e tem aquele tanto de questionários impressos para digitar e assim poder enviar para o estatístico...

E aí, como tabular os dados?

Hoje vão algumas dicas de como tabular os dados da sua pesquisa para garantir a qualidade dos dados e evitar restruturações na hora nas análises:

1. O primeiro passo é a **escolha do software para digitação dos dados**:

Eu particularmente não tenho preferências quanto a isso, deixo livre para o pesquisador escolher o software que tem mais familiaridade. Excel, SPSS, EPi-Info... na dúvida indico o Excel mesmo, o mais básico e que todo mundo tem no computador. E mesmo que você tenha digitado os dados no SPSS e o estatístico te diga que irá usar o SAS ou R para realizar as análises, você consegue exportar os dados do SPP para formatos de Excel (xls, csv...). E a partir do SAS e R também é possível ler os dados de um arquivo SPSS, então fica tudo certo!

Em seguida, é partir para a digitação, mas não antes sem terminar de ler as dicas!

2. Codificar as variáveis qualitativas (categóricas).

- Você pode digitar por exemplo "feminino" e "masculino" para a variável sexo, mas será que você não irá perder muito tempo digitando a palavra inteira sendo que você poderia digitar apenas "F" e "M" ou "0" e "1"? Além de que, se ora você digitar "feminino" e ora "Feminino", o software entenderá como duas coisas diferentes e você terá muita correção para fazer depois.
- Variáveis qualitativas ordinais valem deixar com codificação numérica (0,1,2...). Na variável escolaridade, por exemplo, digitando "Sem instrução", "Ensino Fundamental", "Ensino Médio", na hora de gerar as análises, a

ordem das respostas virá geralmente na ordem alfabética. Então melhor codificar por 1,2 e 3, respectivamente, por exemplo.

3. Criar um número de identificação de sujeitos no banco de dados e questionário.

Manter o nome de pessoas no banco de dados pode ser desconfortável e sempre será antiético. Numere todos os questionários e use essa numeração para identificar os sujeitos no banco de dados. Essa identificação no banco de dados é essencial pois em caso de revisões de alguma variável, você localizará os questionários com maior rapidez.

4. Definir a estrutura da tabulação dos dados

Digitar os dados da maneira que será necessário para a análise de dados: na dúvida, procure orientação de um estatístico. As variáveis sempre ficam na primeira linha (cabeçalho), uma em cada coluna, sendo a primeira variável uma identificação do sujeito. E dessa maneira, em cada linha irão todas as

informações de daquele sujeito. Se você fez um estudo que precisasse medir algumas informações do sujeito em dois ou mais tempos, digamos dois para exemplificar, você terá uma linha para o este indivíduo no primeiro tempo e uma segunda para este indivíduo no segundo tempo. Lembrando que é essencial ter uma variável indicadora do tempo (1 e 2, Pré e Pós, Antes e Depois, etc).

Exemplos de estrutura de banco de dados:

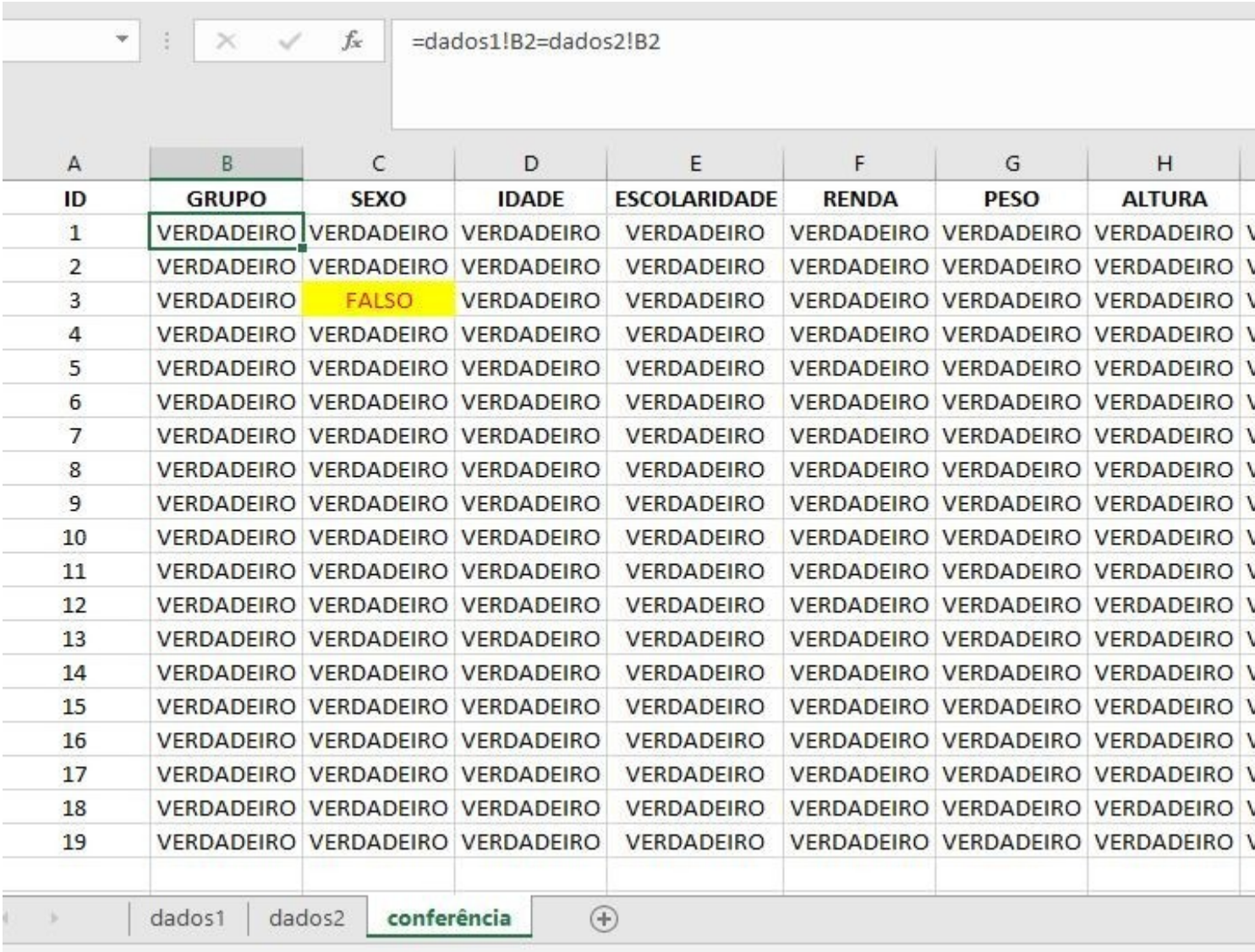
ID	Grupo	Sexo	Idade	Diabetes	Depressao	IMC
1	1	F	50	N	S	35
2	1	M	47	N	N	20
3	2	F	39	S	N	36
4	2	M	54	S	S	39

ID	Grupo	Tempo	Sexo	IMC	Colest_HDL	Colest_LDL
1	1	1	F	50	65	190
1	1	2	F	50	60	170
2	1	1	M	47	26	118
2	1	2	M	47	23	110
3	2	1	M	54	50	100
3	2	2	M	54	49	90

5. Dupla digitação dos dados

Recomenda-se que pelo menos duas pessoas façam a digitação do banco de dados e utilize alguma função para fazer a checagem da digitação. Isto minimiza consideravelmente os erros de digitação. No Excel, por exemplo, uma pessoa pode digitar numa primeira planilha e uma segunda pessoa em outra

planilha, seguindo a mesma estruturação e ordem de digitação. Você pode numa terceira planilha do Excel usar uma fórmula simples para verificar se as células da primeira planilha estão iguais a célula da segunda planilha. Em caso positivo, a célula com a formula mostrará “VERDADEIRO”. Se FALSO, você deverá verificar no questionário correspondente qual é a informação verdadeira e fazer a correção na planilha incorreta.



Durante o processo de criação de um modelo de machine learning nós precisamos medir a qualidade dele de acordo com o objetivo da tarefa. Existem funções matemáticas que nos ajudam a avaliar a capacidade de erro e acerto dos nossos modelos, e agora você conhecerá algumas das mais utilizadas. No artigo, usarei a palavra métrica para me referir a essas funções.

Tão importante quanto saber escolher um bom modelo, é saber escolher a métrica correta para decidir qual é o melhor entre eles.

Existem métricas mais simples, outras mais complexas, algumas que funcionam melhor para datasets com determinadas características, ou outras personalizadas de acordo com o objetivo final do modelo.

Ao escolher uma métrica deve-se levar em consideração fatores como a proporção de dados de cada classe no dataset e o objetivo da previsão (probabilidade, binário, ranking, etc). Por isso é importante conhecer bem a métrica que será utilizada, já que isso pode fazer a diferença na prática.

Nenhuma destas funções é melhor do que as outras em todos os casos. É sempre importante levar em consideração a aplicação prática do modelo. O objetivo deste artigo não é ir a fundo em cada uma delas, mas apresentá-las para que você possa pesquisar mais sobre as que achar interessante.

Classificação

Estas métricas são utilizadas em tarefas de classificação, e a maioria delas pode ser adaptada tanto para classificação binária quanto de múltiplas classes. Nas tarefas de classificação buscamos prever qual é a categoria a que uma amostra pertence como, por exemplo, determinar se uma mensagem é spam.

Precisão Geral (Accuracy)

$$Precisão\ Geral = \frac{P}{P + N}$$

Esta é a métrica mais simples. É basicamente o número de acertos (positivos) divido pelo número total de exemplos. Ela deve ser usada em datasets com a mesma

proporção de exemplos para cada classe, e quando as penalidades de acerto e erro para cada classe forem as mesmas.

Em problemas com classes desproporcionais, ela causa uma falsa impressão de bom desempenho. Por exemplo, num dataset em que 80% dos exemplos pertençam a uma classe, só de classificar todos os exemplos naquela classe já se atinge uma precisão de 80%, mesmo que todos os exemplos da outra classe estejam classificados incorretamente.

F1 Score

$$F1 = \frac{2 * precisão * recall}{precisão + recall}$$

O F1 Score é uma média harmônica entre precisão (que, apesar de ter o mesmo nome, não é a mesma citada acima) e recall. Veja abaixo as definições destes dois termos.

Ela é muito boa quando você possui um dataset com classes desproporcionais, e o seu modelo não emite probabilidades. Isso não significa que não possa ser usada com modelos que emitem probabilidades, tudo depende do objetivo de sua tarefa de machine learning.

Em geral, quanto maior o F1 score, melhor.

Precisão (Precision)

$$Precisão = \frac{PV}{PV + FP}$$

Número de exemplos classificados como pertencentes a uma classe, que realmente são daquela classe (positivos verdadeiros), dividido pela soma entre este número, e o número de exemplos classificados nesta classe, mas que pertencem a outras (falsos positivos).

Recall

$$Recall = \frac{PV}{P}$$

Número de exemplos classificados como pertencentes a uma classe, que realmente são daquela classe, dividido pela quantidade total de exemplos que pertencem a esta classe, mesmo que sejam classificados em outra. No caso binário, positivos verdadeiros divididos por total de positivos.

AUC – Area Under the ROC Curve

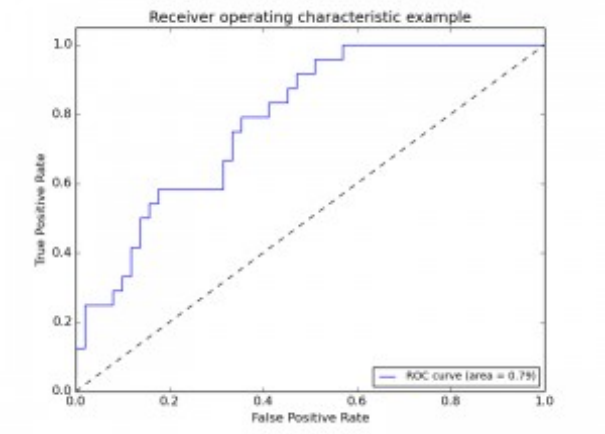


Imagem retirada da documentação do Scikit-Learn

Esta é uma métrica interessante para tarefas com classes desproporcionais. Nela, mede-se a área sob uma curva formada pelo gráfico entre a taxa de exemplos positivos, que realmente são positivos, e a taxa de falsos positivos.

Uma das vantagens em relação ao F1 Score, é que ela mede o desempenho do modelo em vários pontos de corte, não necessariamente atribuindo exemplos com probabilidade maior que 50% para a classe positiva, e menor, para a classe negativa.

Em sistemas que se interessam apenas pela classe, e não pela probabilidade, ela pode ser utilizada para definir o melhor ponto de corte para atribuir uma ou outra classe a um exemplo. Este ponto de corte normalmente é o ponto que se localiza mais à esquerda, e para o alto, no gráfico, mas depende bastante do custo do erro na previsão de uma determinada classe.

Log Loss

$$Log\ Loss = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

A fórmula do exemplo é para o caso binário, neste caso: **p** é a probabilidade do exemplo pertencer à classe 1, e **y** é o valor real da variável dependente.

Esta função pune previsões incorretas muito confiantes. Por exemplo, prever uma classe com uma probabilidade de 95%, e na realidade a correta ser outra. Ela pode ser utilizada para problemas binários ou com múltiplas classes, mas eu particularmente não gosto de usar em datasets com classes desproporcionais. O valor dela sempre terá a tendência de melhorar se o modelo estiver favorecendo a maior classe presente.

Tomando os cuidados acima, nas situações em que a probabilidade de um exemplo pertencer a uma classe for mais importante do que classificá-lo diretamente, esta função é preferível a usar simplesmente a precisão geral.

Regressão

Neste parte estão as funções mais comuns utilizadas para avaliar o desempenho de modelos de regressão. Na regressão buscamos prever um valor numérico, como, por exemplo, as vendas de uma empresa para o próximo mês. Nos exemplos abaixo:

$$y_i = valor\ real$$
$$\hat{y}_i = valor\ previsto$$

Mean Squared Error – MSE

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

Talvez seja a mais utilizada, esta função calcula a média dos erros do modelo ao quadrado. Ou seja, diferenças menores têm menos importância, enquanto diferenças maiores recebem mais peso.

Existe uma variação, que facilita a interpretação: o Root Mean Squared Error. Ele é simplesmente a raiz quadrada do primeiro. Neste caso, o erro volta a ter as unidades de medida originais da variável dependente.

Mean Absolute Error – MAE

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

Bastante parecido com MSE, em vez de elevar a diferença entre a previsão do modelo, e o valor real, ao quadrado, ele toma o valor absoluto. Neste caso, em vez de atribuir um peso de acordo com a magnitude da diferença, ele atribui o mesmo peso a todas as diferenças, de maneira linear.

Se imaginarmos um exemplo simples, onde temos apenas a variável que estamos tentando prever, podemos ver um fato interessante que difere o MSE do MAE, e que devemos levar em conta ao decidir entre os dois: o valor que minimizaria o primeiro erro seria a média, já no segundo caso, a mediana.

Mean Absolute Percentage Error – MAPE

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

Este erro calcula a média percentual do desvio absoluto entre as previsões e a realidade. É utilizado para avaliar sistemas de previsões de vendas e outros sistemas nos quais a diferença percentual seja mais interpretável, ou mais importante, do que os valores absolutos.

Métricas específicas para tarefas

Em alguns casos, o ideal é usar uma métrica que tenha um significado específico para a tarefa em questão. Por exemplo, na segmentação de anúncios, a taxa de cliques; num sistema para comprar e vender ações, verificar o retorno médio. As métricas acima são importantes e podem ser utilizadas de maneira geral, mas se houver uma alternativa melhor, mais adequada ao contexto, ela deve ser utilizada.

Introdução

Após um pequeno hiato no blog, hoje vamos falar um pouco sobre um dos classificadores clássicos mais conhecidos, o K vizinhos mais próximos (do inglês:K nearest neighboors – KNN). O KNN foi proposto por Fukunaga e Narendra em 1975 [1]. É um dos classificadores mais simples de ser implementado, de fácil compreensão e ainda hoje pode obter bons resultados dependendo de sua aplicação. Antes de iniciar, caso você não tenha afinidade com o problema de classificação, sugiro que leia nosso post sobre classificação de dados. Agora, sem mais delongas, vamos ao que interessa.

Funcionamento do KNN

A ideia principal do KNN é determinar o rótulo de classificação de uma amostra baseado nas amostras vizinhas advindas de um conjunto de treinamento. Nada melhor do que um exemplo para explicar o fucionamento do algoritmo como o da Figura 1, na qual temos um problema de classificação com dois rótulos de classe e com $k=7$. No exemplo, são aferidas as distâncias de uma nova amostra, representada por uma estrela, às demais amostras de treinamento, representadas pelas bolinhas azuis e amarelas. A variável k representa a quatidade de vizinhos mais próximos que serão utilizados para averiguar de qual classe a nova amostra pertence. Com isso, das sete amostras de treinamento mais

próximas da nova amostra, 4 são do rótulo A e 3 do rótulo B. Portanto, como existem mais vizinhos do rótulo A, a nova amostra receberá o mesmo rótulo deles, ou seja, A.

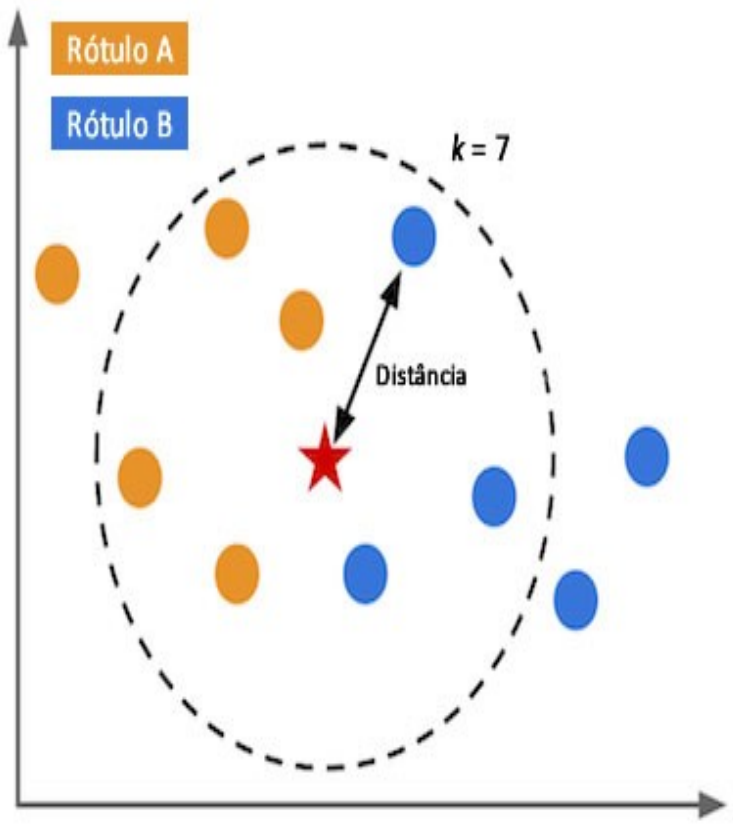


Figura 1: exemplo de classificação do KNN com dois rótulos de classe e k = 7

Dois pontos chaves que devem ser determinados para aplicação do KNN são: a métrica de distância e o valor de k. Portanto, vamos discutir cada uma delas.

Cálculo da distância

Calcular a distância é fundamental para o KNN. Existem diversas métricas de distância, e a escolha de qual usar varia de acordo com o problema. A mais utilizada é a distância Euclidiana, descrita pela equação 1.

$$D_E(p,q) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad (1)$$

Outros exemplos de distância, é a de Minkowsky:

$$D_M(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^r\right)^{\frac{1}{r}} \quad (2)$$

E também, a distância de Chebyshev:

$$D_C(p,q) = \max_i(|p_i, q_i|) \quad (3)$$

Em todos os casos, $p = (p_1, \cdots, p_n)$ e $q = (q_1, \cdots, q_n)$ são dois pontos n-dimensionais e na equação 2, r é uma constante que deve ser escolhida. No exemplo da Figura 1, essas distâncias seriam calculadas entre as bolinhas (azuis e laranjas) e a estrela (a nova entrada). Como o exemplo é 2D, cada uma cada ponto teria seu valor em x e em y. Para problemas com dimensões maiores a abordagem é a exatamente a mesma, porém, a visualização das amostras no espaço é mais complicada.

A escolha de K

Em relação a escolha do valor k, não existe um valor único para a constante, a mesma varia de acordo com a base de dados. É recomendável sempre utilizar valores ímpares/primos, mas o valor ótimo varia de acordo com a base de dados. Dependendo do seu problema, você pode utilizar um algoritmo de otimização (PSO, GA, DE etc) para encontrar o melhor valor para o seu problema. Todavia, você pode deixar o desempenho geral do modelo bem lento na etapa de seleção de k. Outra maneira e simplesmente testar um conjunto de valores e encontrar o valor de k empiricamente.

Pseudocódigo

Para melhor compreensão do algoritmo, apresento também o pseudocódigo do mesmo.

```
1  inicialização:
2      Preparar conjunto de dados de entrada e saída
3      Informar o valor de k;
4  para cada nova amostra faça
5      Calcular distância para todas as amostras
6      Determinar o conjunto das k's distâncias mais próximas
7      O rótulo com mais representantes no conjunto dos k's
8      vizinhos será o escolhido
9  fim para
10 retornar: conjunto de rótulos de classificação
```

Código do KNN em Python

Resumidamente, a grande vantagem do KNN é sua abordagem simples de ser compreendida e implementada. Todavia, calcular distância é tarefa custosa e caso o problema possua grande número de amostras o algoritmo pode consumir muito tempo computacional. Além disso, o método é sensível à escolha do k. Por fim, deixo linkado uma implementação do KNN em Python. No repositório existe bases de dados comuns da literatura, como Iris e Australian Credit. Todavia, você pode utilizar o código para qualquer que seja a base. Bom proveito!

Implementação do KNN em Python

Pesquisa: Técnicas de pré-processamento

Enunciado da atividade:

Faça uma pesquisa e construa um quadro comparativo sobre as seguintes técnicas de pré-processamento para redução de dimensionalidade:

- Força bruta.
- Determinação da relevância.
- Seleção baseada em correlação.

O quadro deve contemplar as seguintes características:

- Ideia geral.
- Pontos positivos.
- Pontos negativos.

Resposta:

Força bruta

Ideia geral: o objetivo desta técnica é reduzir o número de atributos para um subgrupo ótimo. A ideia de algoritmos de força bruta é testar todas as combinações de atributos possíveis.

Ponto positivo: encontra sempre o melhor subgrupo de atributos.

Ponto negativo: computacionalmente proibitivo.

Determinação de relevância

Ideia geral: a seleção de atributos é embutida no algoritmo indutor e utiliza ganho de informação ao determinar quais são as características mais relevantes. Exemplos de algoritmos que utilizam esta técnica incluem árvores de decisão, como C4.5, ID3 e *Classification and Regression Trees* (CART).

Ponto positivo: como esta abordagem encontra-se embutida em certos algoritmos, a etapa de pré-processamento é simplificada.

Pontos negativos: a seleção tende a favorecer atributos que apresentam muita flutuação e a se tornar impraticável quando o problema possui alta dimensionalidade (número muito grande de atributos)

Seleção baseada em correlação

Ideia geral: enquanto abordagens tradicionais avaliam atributos de acordo com a relevância individual, este método analisa a relevância em subconjuntos. Um subconjunto é considerado bom se os atributos possuem alta correlação com o atributo *meta* e baixa correlação entre os elementos internos.

Ponto positivo: analisa subconjuntos de atributos, em vez de avaliar atributos individualmente, o que tende a ser mais eficaz do que os algoritmos supracitados.

Pontos negativos: em bases reais, é pouco provável que existam atributos altamente correlacionados com a classe; caso existissem, estratégias de *machine learning* não seriam tão úteis quanto a própria estatística. Em síntese, o método tende a não se adaptar bem a cenários de contexto real.

1. Como a profundidade de um nó é medida?

Ela é medida a partir do primeiro nó Raiz que é o nó 0, a partir dai a cada geração de nós é contado +1

2. O que é um percurso em árvore (ou caminhamentos em árvore)? Cite três exemplos.

É uma forma de percorrer todos os nós de uma arvore:

- ✓ Percurso Pré ordem
- ✓ Percurso Ordenado
- ✓ Percurso Pós Ordem

3. Qual é a diferença entre uma árvore binária e uma árvore AVL?

Arvore binária é a arvore que tem no minimo 0 ou até 2 filhos, a arvore AVL é uma arvore binária otimizada para busca, balanceada, enquanto a binária normal leva um tempo proporcional de busca aos níveis da arvore, a arvore AVL reduz essa quantidade de níveis tornando a busca mais rápida

4. Qual árvore permite uma busca mais eficiente: binária ou AVL?

AVL → O(log n)

5. Qual critério é utilizado para verificar se uma árvore AVL está desbalanceada?

Subtrai-se o numero de níveis na sub-arvore da esquerda da sub-arvore da direita, se esse númeor é maior que abs(1) está desbalanceada. Para resolvermos precisamos usar o método de rotações.