# DA1MCTA018-13SA - Programação Orientada a Objetos - Paulo Henrique Pisani - 2021.2



Descrição | Visualizar envios

## [Bonus] Parte 2 - Algoritmo de classificação

**Data de entrega**: quarta, 4 Ago 2021, 23:59
**Arquivos requeridos**: ListaExemplos.java, KNN.java (Baixar)
**Tipo de trabalho**: Trabalho individual

## Parte 2 - Algoritmo de classificação

Na área de Aprendizado de Máquina (Faceli et al., 2021), um algoritmo que pode ser usado para classificação e regressão é o k-Vizinhos mais próximos (*k-Nearest Neighbours* - k-NN). Neste exercício, deverá ser implemetada uma versão para classificação, conforme descrito a seguir.

### k-NN

Antes de ser utilizado para classificação, o algoritmo precisa de dados de treinamento. Os dados de treinamento serão representados por uma lista de exemplos, como exemplificado a seguir. Cada exemplo possui atributos de entrada e um rótulo de classe correspondente.

| a1 | a2 | a3 | rótulo da classe |
|---|---|---|---|
| 1,0 | 0,9 | 1,1 | **1** |
| 1,1 | 0,8 | 1,0 | **1** |
| 4,0 | 8,0 | 0,9 | **2** |
| 1,2 | 1,1 | 0,9 | **1** |
| 4,0 | 6,0 | 0,1 | **2** |

A partir dos dados de treinamento, o algoritmo realizará predições dos rótulos de classe para novos dados. Para os valores de atributos de entrada [5, 7, 1], um algoritmo de classificação poderia predizer o rótulo de classe 2. Essa predição será o resultado da classificação realizada pelo algoritmo.

No k-NN, a predição do rótulo de classe para novos valores de atributos de entrada (dados de teste) é realizada da seguinte forma:

1. Calcule a distância entre os valores dos atributos de entrada passados (dados de teste) e cada um dos exemplos nos dados de treinamento (utilize distância Euclidiana);
2. Determine quais são os *k* exemplos mais próximos (ou seja, os que tem as *k* menores distâncias). O valor de *k* é definido previamente.
3. O rótulo de classe predito (resultado da classificação) será aquele mais frequente dentre os rótulos dos *k* exemplos mais próximos.

## Demonstração de classificação considerando os dados de treinamento da tabela anterior e os valores de atributos de entrada [5, 7, 1]

1. Distância entre os valores dos atributos de entrada de teste [5, 7, 1] e cada um dos exemplos de treinamento (**o cálculo da distância considera apenas os atributos de entrada, no caso a1, a2 e a3**):

| a1 | a2 | a3 | rótulo da classe |
|---|---|---|---|
| 1,0 | 0,9 | 1,1 | **1** |
| 1,1 | 0,8 | 1,0 | **1** |
| 4,0 | 8,0 | 0,9 | **2** |
| 1,2 | 1,1 | 0,9 | **1** |
| 4,0 | 6,0 | 0,1 | **2** |

| distância |
|---|
| **7,30** |
| **7,32** |
| **1,42** |
| **7,02** |
| **1,68** |

2. Para *k*=3, os *k* mais próximos são destacados a seguir:

| a1 | a2 | a3 | rótulo da classe |
|---|---|---|---|
| 4,0 | 8,0 | 0,9 | **2** |
| 4,0 | 6,0 | 0,1 | **2** |
| 1,2 | 1,1 | 0,9 | **1** |
| 1,0 | 0,9 | 1,1 | **1** |
| 1,1 | 0,8 | 1,0 | **1** |

| distância |
|---|
| **1,42** |
| **1,68** |
| **7,02** |
| **7,30** |
| **7,32** |

3. Entre os *k*=3 exemplos mais próximos, o rótulo de classe mais frequente é o 2 . Portanto, o resultado da predição é 2.

## Classes a serem implementadas

Devem ser implementadas as classes KNN e ExcecaoDadosInvalidos. Além disso, também submeta o código das classes Exemplo e ListaExemplos. Essas duas classes foram descritas em [Bonus] Parte 1 - Lista de exemplos e são necessárias para a implementação da classe KNN.

[acesso package] Classe **ExcecaoDadosInvalidos** (subclasse de Exception - não é RuntimeException): esta exceção é lançada pelo método predizer quando a quantidade de atributos de entrada é inválida (mais detalhes na descrição do método predizer da classe KNN).

- public int getQtdAtributosTreinamento() - retorna a quantidade de atributos de entrada nos dados de treinamento. Esse valor deve ser sido armazenado dentro da instância da exceção no momento em que a exceção é criada.
- public int getQtdAtributosPredizer() - retorna a quantidade de atributos de entrada do parâmetro passado ao método predizer. Esse valor deve ser sido armazenado dentro da instância da exceção no momento em que a exceção é criada.

[acesso public] Classe **KNN**:

- public KNN(int k) - construtor que recebe *k* e armazena na instância da classe.
- public void setDadosTreinamento(ListaExemplos lista) - armazena uma lista de exemplos como conjunto de dados de treinamento.
- public int predizer(double[] atributosEntrada) - recebe valores dos atributos de entrada (dados de teste) e retorna o rótulo de classe predito pelo algoritmo k-NN, conforme procedimento descrito no início do enunciado deste exercício. Esse método predizer deve lançar exceção em algumas situações:
  - Chamar o método predizer sem possuir dados de treinamento: deve ser lançada uma exceção do tipo Exception com o texto "Dados de treinamento - nao inicializado" (observe que não há acento na mensagem). Use o seguinte construtor, já existente na classe Exception: Exception(String message).
  - Chamar o método predizer com um exemplo que possui quantidade de atributos diferente da quantidade de atributos dos exemplos de treinamento: deve ser lançada uma exceção do tipo ExcecaoDadosInvalidos informado a quantidade de atributos de entrada nos exemplos de treinamento e nos dados passados para o método predizer.

**Importante**: O programa principal já existe no sistema de correção automática. Submeta apenas as classes especificadas (todas devem estar no **pacote classificacao**). A classes não podem realizar impressão de dados, java.util ou utilizar import. Os códigos submetidos não podem utilizar métodos prontos para ordenação (por exemplo, sort). Para ordenação, implemente um método e inclua na submissão. Também não use o termo "sort" em nenhuma parte do código.

# Casos de teste

Formato dos casos de teste (que aparecem ao avaliar as classes no sistema de correção automática):

## Entrada

- Sequência de operações no formato:
  - instanciarListaExemplos [qtdMax]
  - lista [índice lista] adicionarExemplo [qtd_atributos_entrada] (valores dos atributos de entrada) [rótulo da classe]
  - instanciarKNN [k]

- knn setDadosTreinamento [índice da lista de exemplos]
- knn predizer [qtd_atributos_entrada] (valores dos atributos de entrada)

## Saída

- Verificação das classes
- Operações executadas e saídas obtidas

**Referências:**
Faceli , K., Lorena, A. C., Gama, J., Almeida, T. A., Carvalho, A. C. P. L. F. (2021). Inteligência Artificial Uma Abordagem de Aprendizado de Máquina, 2a edição, LTC.
Um dos testes é realizado com parte do conjunto de dados Iris: https://archive.ics.uci.edu/ml/datasets/Iris

VPL

◄ [Bonus] Parte 1 - Lista de exemplos | Seguir para... | [Bonus] REC - Parte 1 - Lista de exemplos ►