Aprendizagem automática

Treino, Avaliação, Generalização e sobre-ajustamento

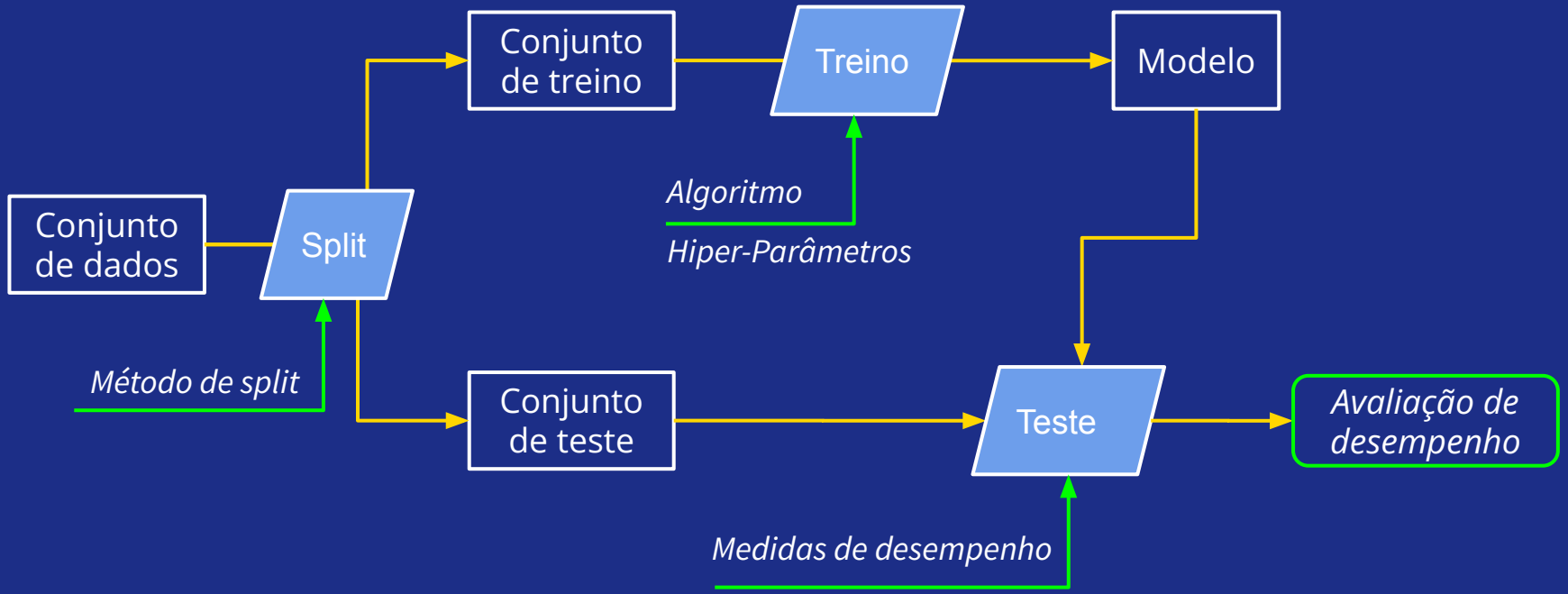
Luís Rato, Universidade de Évora, 2022/23

Sumário

- Processo de construção de um modelo AA
- Uma primeira aplicação - Dataset Iris
  - Conjunto de dados
  - Inspeção dos dados
  - Construção do modelo
  - Avaliação do modelo - exatidão/*accuracy*
- Generalização
- Sobre-ajustamento e sub-ajustamento
- Complexidade

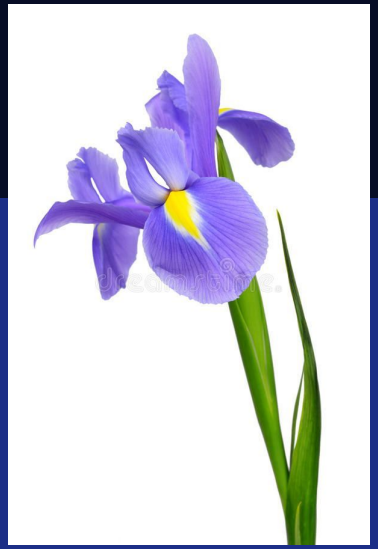
Construção de um modelo

Processo simplificado



Uma aplicação simples
Iris dataset

Espécies Iris



- Problema
  - Distinguir a espécie de flores iris
- Espécies
  - Setosa, Versicolor, Virginica
- Caracterização da flor
  - Comprimento e largura da pétala
  - Comprimento e largura da sépala

https://archive.ics.uci.edu/ml/datasets/Iris

Enquadramento-apagar

- Objetivo
  - Criar um modelo de aprendizagem automática
    - Aprende a espécie a partir das medidas de flores cuja espécie é conhecida
    - É capaz de prever a espécie de uma nova flor
- Tarefa
  - **Aprendizagem supervisionada**
    - A espécie de cada flor é conhecida
  - Problema de **classificação** com 3 classes
    - Cada flor é caracterizada por 4 **atributos**
    - Cada flor é de uma espécie
    - Cada espécie é uma **etiqueta**

Conjuntos de dados

Caracterização do conjunto de dados

- Instâncias: 150
- Atributos: 4 (numéricos)
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width
- Classes: 3
  - Setosa
  - Versicolor
  - Virginica

```
4.4,3.0,1.3,0.2,Iris-setosa
5.1,3.4,1.5,0.2,Iris-setosa
5.0,3.5,1.3,0.3,Iris-setosa
4.5,2.3,1.3,0.3,Iris-setosa
4.4,3.2,1.3,0.2,Iris-setosa
5.0,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
6.7,3.1,4.4,1.4,Iris-versicolor
5.6,3.0,4.5,1.5,Iris-versicolor
5.8,2.7,4.1,1.0,Iris-versicolor
6.2,2.2,4.5,1.5,Iris-versicolor
5.6,2.5,3.9,1.1,Iris-versicolor
```

Conjuntos de treino e teste

- É necessário avaliar o modelo antes de o utilizar
  - Verificar se de facto funciona e se podemos confiar nas suas predições
- A avaliação **não** pode ser feita sobre os dados usados na construção do modelo
  - O modelo pode apenas memorizar os dados
  - Não nos dá indicação se o modelo generaliza bem
  - É necessário avaliar o modelo sobre novos dados (não vistos antes)
- Solução
  - Dividir os dados em 2 sub-conjuntos
    - Conjunto de treino: utilizado para construir o modelo
    - Conjunto de teste: utilizado para avaliar o modelo

Divisão treino/teste

- Proporção
  - Conj. treino: 70% - 75% - valores típicos
  - Conj. teste: 30% - 25%
- Os devem ser “baralhados” antes da divisão para prevenir conjuntos com distribuições distintas
  - Normalmente realizado através da geração de nº aleatórios
- Para poder replicar as experiências
  - Deve garantir-se a mesma divisão através da definição da “semente” dos nº aleatórios

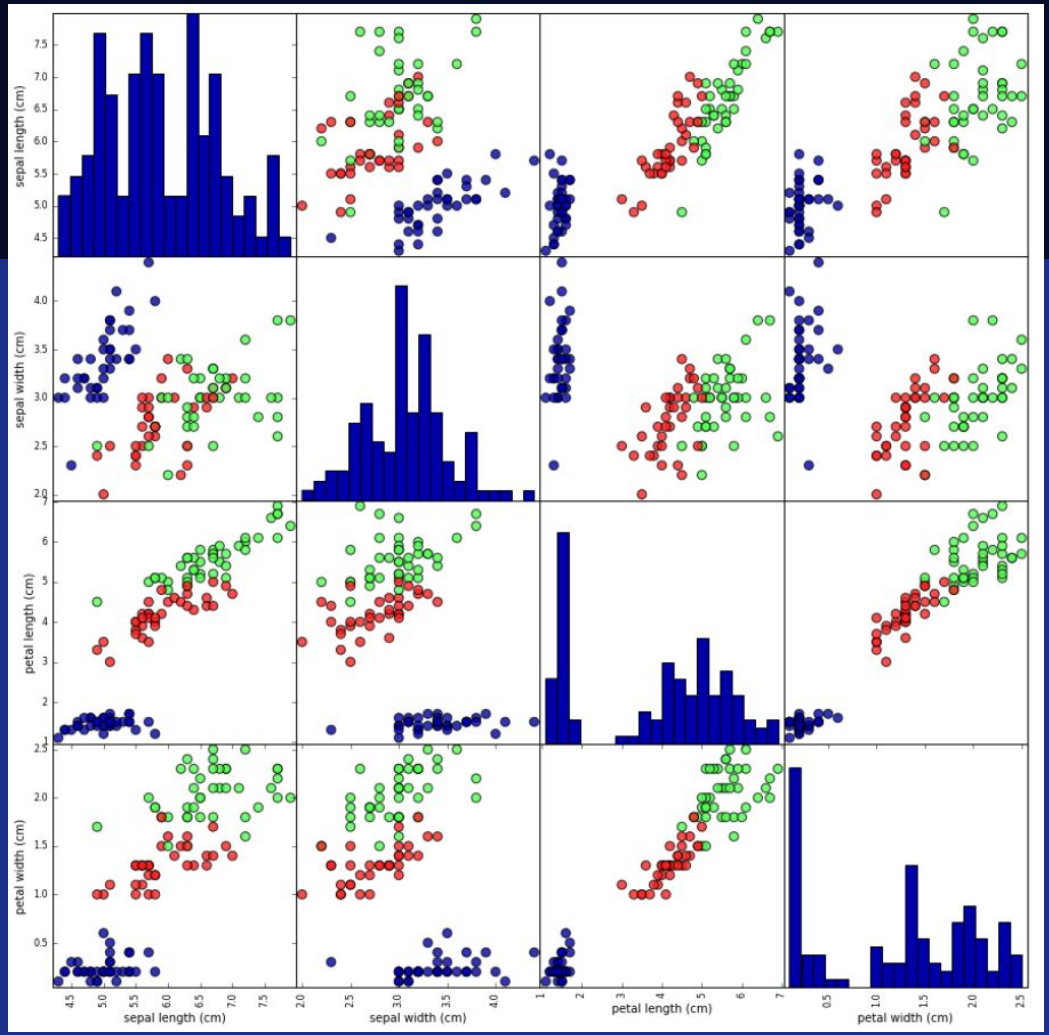
Inspeção dos dados

Inspeção dos dados

- Antes de criar um modelo é importante inspecionar os dados
  - Verificar se a tarefa é facilmente resolvida sem AA
  - Ver se os dados contêm a informação desejada
  - Encontrar anormalidades
    - inconsistências ou medições inesperadas
- Uma das melhores formas é **visualizar** os dados
  - Utilizar um **scatter plot**
    - Coloca um atributo em cada eixo
    - Marca um ponto no espaço para cada instância
  - Apenas é possível utilizar 2 (ou 3) atributos de cada vez

Pair plots

- Pair plot
  - scatter plot para todos os pares de atributos
- Não mostra a interação entre todos os atributos ao mesmo tempo
  - Alguns aspectos interessantes dos dados podem não ser revelados



Construção do modelo

Algoritmo KNN

- Escolher um algoritmo de classificação (entre a grande variedade existente)
  - KNN - K-vizinhos-mais-próximos
- Algoritmo
  - Construção do modelo
    - Guardar o conjunto de dados
  - Fazer uma previsão
    - Encontrar o ponto mais próximo do novo ponto
    - Etiquetar o novo ponto com a etiqueta do ponto de treino encontrado
  - K
    - Em vez de utilizar o vizinho mais próximo, considera K vizinhos
      - Faz a previsão usando, por exemplo, a maioria

Avaliação do modelo

Avaliação

- Utilizar o conjunto de teste
  - Para cada flor do conjunto, fazer a previsão e compará-la com a sua etiqueta (espécie)
- Analisar a matriz de confusão
  - As linhas correspondem à verdadeira classe
  - As colunas correspondem à classe prevista
  - Cada entrada tem a contagem de exemplos para essa combinação
- Calcular uma (ou mais) medidas de desempenho
  - **Exatidão**: fração de flores com previsão correta da espécie

Matriz de confusão

- Etiquetas - scikit-learn transforma em números
  - setosa: 0 versicolor: 1 virginica: 2
- Etiquetas conjunto de teste (38 exemplos)
  - Verdadeira 2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 1
  - Prevista 2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 2
- Matriz de confusão

real \ prevista	0	1	2
0	13	0	0
1	0	15	**1**
2	0	0	9

Exatidão (accuracy) = (13+15+9) / 38

Exatidão = **previsões corretas / total dos testes**

Scikit-Learn

Código - Aula prática

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix

iris_dataset = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris_dataset['data'],
iris_dataset['target'],  random_state=0 )

knn = KNeighborsClassifier( n_neighbors=1 )
knn.fit(X_train, y_train)

pred_knn = knn.predict(X_test)

confusion = confusion_matrix(y_test, pred_knn)
print("Confusion matrix:\n{}".format(confusion))
print("Test set accuracy: {:.2f}".format(knn.score(X_test, y_test)))
```

Classificação e Regressão

Classificação e Regressão

- Classificação
  - Prever uma **etiqueta** (entre uma lista pré-definida de possibilidades)
  - Tipos
    - Binária (2 classes)
    - Multi-classe (mais de 2 classes)
  - Exemplos
    - Ireis, mensagens spam, língua de um texto
- Regressão
  - Prever um valor **numérico contínuo**
  - Exemplos
    - Rendimento anual de uma pessoa (dada a educação, idade, onde vive, ...)
    - produção de uma cultura (dada produção anterior, tempo, nº empregados, …)

Generalização e complexidade

- Generalização
  - O modelo **generaliza** a partir do conj treino para o conj teste quando faz previsões exatas sobre dados novos
- Complexidade
  - Se o modelo for **muito complexo**, pode ser tão exato quanto quisermos sobre o conj de treino. Há memorização dos dados
  - Neste caso, normalmente o desempenho sobre o conj teste será fraco

Exemplo

- Tarefa
  - Prever se um cliente vai comprar um barco
- Modelo 1
  - age>45 *and* (nr_children<3 *or* marital_status != divorced)
  - exatidão=100%
  - Parece ser demasiado complexo!
- Modelo 2
  - age>50
  - Se explicar o comportamento de todos os cliente, seria preferível ao modelo 1

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no

- Pretende-se encontrar o modelo mais simples que explique os dados!

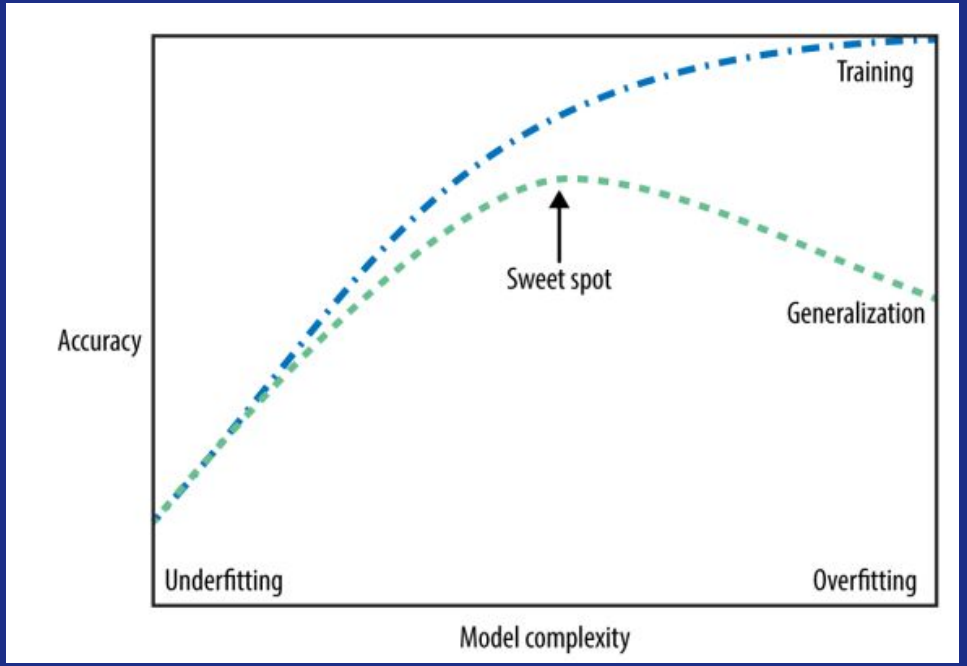
Sobre-ajustamento - *overfitting* e sub-ajustamento - *underfitting*

- Sobre-ajustamento
  - *Overfitting*
- O que é?
  - Construção de um modelo **demasiado complexo** para a quantidade de informação disponível
- Quando acontece?
  - Quando o modelo é ajustado às particularidades do conj treino
  - Obtém-se um modelo que funciona bem para o conj treino, mas não consegue generalizar para novos dados

- Sub-ajustamento
  - *Underfitting*
- O que é?
  - Construção de um modelo **demasiado simples**
- Quando acontece?
  - Quando não consegue capturar todos os aspetos e variabilidade dos dados
  - Obtém-se um modelo que nem no conj de treino tem um bom desempenho

Complexidade do modelo

- Quanto **mais complexo**, melhor será capaz de prever sobre o conjunto de treino
- Se for demasiado complexo, começa a focar-se demasiado em cada exemplo do conjunto de treino e **não generaliza** bem
- Se for insuficientemente complexo não consegue modelar a complexidade



Complexidade e tamanho do conj de dados

- Complexidade e sobre-ajustamento
  - Quanto **maior a variedade** de exemplos no conjunto de dados, mais complexo pode ser o modelo sem existência de sobre-ajustamento
- Variedade e tamanho do conjunto de dados
  - Normalmente, a existência de mais dados corresponde a uma maior variedade
  - Mas… duplicar ou colecionar dados muito semelhantes não ajuda…
- Exemplo
  - Se a regra do Modelo 1 (slide 8, *age>45 and (nr_children<3 or marital_status != divorced)*) fosse aplicável a 10000 exemplos e não apenas a 12, mais facilmente diríamos que esta era uma **boa regra**

Questões para reflexão

- … e se houver **duplicados** no data set ?

- … e se não houver duplicados mas existirem **quase-duplicados**?