| PLANIFICAÇÃO |
|---|
| **Curso**: UFCD **10810** |
| **UFCD/Módulo/Temática:** UFCD 10810 - **Fundamentos do desenvolvimento de modelos analíticos em Python** |
| **Ação: 10810_1L** |
| **Formador/a**: Sandra Liliana Meira de Oliveira |
| **Data**: |
| **Nome do Formando/a**: |

# Sessão 3 : Introdução à Aprendizagem Automática e Componentes

## Essenciais

### Objetivos

- Compreender os fundamentos da aprendizagem automática
- Identificar os elementos principais de um sistema de aprendizagem

### Conteúdo Teórico Detalhado

O formador deve iniciar com uma introdução à Inteligência Artificial (IA), explicando que esta representa o desenvolvimento de sistemas que simulam a inteligência humana, capazes de executar tarefas como reconhecimento de fala, visão computacional, jogos, entre outros.

### Divisão da IA:

- IA Simbólica: baseada em regras e lógica.
- IA Estatística: baseada em dados e modelos matemáticos (aqui insere-se o Machine Learning).

### O que é Machine Learning (ML)?

É um ramo da IA que permite que os computadores aprendam com dados, sem serem explicitamente programados para cada tarefa específica.

**Tipos de ML:**

- Supervisionado: usa dados com rótulos (ex: classificação, regressão).
- Não-Supervisionado: usa dados sem rótulos (ex: agrupamento).
- Por Reforço: aprendizagem por tentativa e erro (ex: jogos, robótica).

**Componentes Essenciais de um Sistema de ML:**

- **Dados:** entradas (features) e saídas (labels, quando supervisionado).
- **Modelo/Algoritmo:** método usado para aprender os padrões (ex: Regressão Logística, KNN).
- **Função Objetivo:** métrica a ser minimizada (ex: erro quadrático médio).
- **Ciclo de treino e teste:** divisão dos dados para avaliar desempenho real.

## Exemplo prático detalhado - Regressão Logística:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Dados fictícios
X = pd.DataFrame({"idade": [22, 25, 47, 52, 46], "rendimento": [1500, 1800, 2500, 2700, 2200]})
y = [0, 0, 1, 1, 1]  # 0 = não compra, 1 = compra

# Divisão treino/teste
treino_X, teste_X, treino_y, teste_y = train_test_split(X, y, test_size=0.2)

# Modelo e treino
modelo = LogisticRegression()
modelo.fit(treino_X, treino_y)
```

**Explicações:**

- **DataFrame(...):** organiza os dados em formato tabular.
- **train_test_split(...):** separa 80% treino e 20% teste aleatoriamente.
- **LogisticRegression():** define o modelo de regressão logística.
- **fit(...):** aplica o treino com os dados.

# Sessão 4: Desenvolvimento de Classificadores e Avaliação

**Objetivos**

- Explicar o processo de desenvolvimento de classificadores
- Compreender e aplicar formas de avaliação de desempenho

**Conteúdo Teórico Detalhado**

**Ciclo de Desenvolvimento de um Modelo:**

1. **Pré-processamento e exploração dos dados:** tratamento de valores ausentes, visualização de distribuições.
2. **Seleção do modelo:** escolha do algoritmo adequado ao tipo de problema.
3. **Treino e validação:** treino do modelo e validação cruzada para estimar performance.
4. **Teste final:** avaliar o modelo em dados nunca vistos.

**Conceitos chave:**

- **Overfitting:** quando o modelo aprende demais os dados de treino e generaliza mal.
- **Underfitting:** quando o modelo é demasiado simples para aprender os padrões relevantes.

**Métricas de Avaliação:**

- **Acurácia (accuracy):** proporção de acertos totais.
- **Precisão (precision):** acertos entre os que foram classificados como positivos.
- **Recall (sensibilidade):** proporção de positivos identificados corretamente.
- **F1-score:** média harmónica entre precisão e recall.
- **Matriz de confusão:** tabela que mostra verdadeiros e falsos positivos e negativos.

**Exemplo - Validação Cruzada com Árvore de Decisão:**

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score

iris = load_iris()
modelo = DecisionTreeClassifier()
scores = cross_val_score(modelo, iris.data, iris.target, cv=5)
print("Accuracy média:", scores.mean())
```

**Explicações:**

- load_iris(): carrega o dataset Iris com três classes.
- DecisionTreeClassifier(): cria o modelo baseado em árvores.
- cross_val_score(...): executa validação cruzada k-fold.
- mean(): calcula a média das acurácias.

## Sessão 5: Aprendizagem Não-Supervisionada e Algoritmos de Clustering

**Objetivos**

- Introduzir a aprendizagem não-supervisionada
- Aplicar análise de clusters com diferentes abordagens

**Conteúdo Teórico Detalhado**

**Conceito:**
A aprendizagem não supervisionada tem como objetivo descobrir padrões ou estrutura escondida nos dados sem usar rótulos.

**Algoritmos mais comuns:**

- **KMeans:** particiona os dados em k grupos com base na distância média aos centróides.
- **Hierárquico:** constrói uma árvore de agrupamentos baseando-se na similaridade.

**Aplicações práticas:**

- Segmentação de clientes
- Agrupamento de textos
- Agrupamento de imagens

**Exemplo - KMeans:**

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

X = [[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]]
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
print("Labels:", kmeans.labels_)
```

**Explicações:**

- **n_clusters=2**: define dois grupos desejados.
- **fit(X)**: aplica o algoritmo e encontra os grupos.
- **labels_**: mostra o grupo ao qual cada ponto pertence.

**Exemplo - Hierárquico (Dendrograma):**

```
from scipy.cluster.hierarchy import dendrogram, linkage

Z = linkage(X, method='ward')
dendrogram(Z)
plt.show()
```

**Explicações:**

- **linkage(...):** calcula a hierarquia com método 'ward'.
- **dendrogram(...):** desenha o dendrograma visual dos agrupamentos.

**Sessão 6: Aprendizagem Supervisionada e Projeto Final**

**Objetivos**

- Aplicar algoritmos supervisionados
- Desenvolver projeto final com aplicação prática

**Conteúdo Teórico Detalhado**

**K-Nearest Neighbors (KNN):**

- Classifica um novo ponto com base na maioria dos seus k vizinhos mais próximos.
- Vantagens: simples, interpretável.

- Limitações: sensível à escala dos dados, custo computacional com muitos dados.

**Árvore de Decisão:**

- Divide os dados de forma recursiva com base em critérios como ganho de informação.
- Vantagens: visual, interpretável.
- Limitações: pode overfitar, sensível a pequenas mudanças nos dados.

**Exemplo - KNN:**

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(treino_X, treino_y)
pred = knn.predict(teste_X)
print(pred)
```

**Explicações:**

- **KNeighborsClassifier(...):** define o número de vizinhos.
- **fit(...):** treino do modelo.
- predict(...): previsão dos dados de teste.

**Exemplo - Árvore de Decisão:**

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

arvore = DecisionTreeClassifier()
arvore.fit(treino_X, treino_y)
plot_tree(arvore, filled=True)
plt.show()
```

**Explicações:**

- **DecisionTreeClassifier():** cria o classificador.
- **plot_tree(...):** desenha a árvore treinada.

**Projeto Final:**

- Dataset sugerido: Iris, Titanic, Vendas ou Banco.
- Etapas:
    1. Exploração e visualização inicial
    2. Tratamento de dados (valores ausentes, codificação)
    3. Aplicação de dois modelos supervisionados
    4. Avaliação com métricas
    5. Discussão dos resultados