CODE TRAINERS

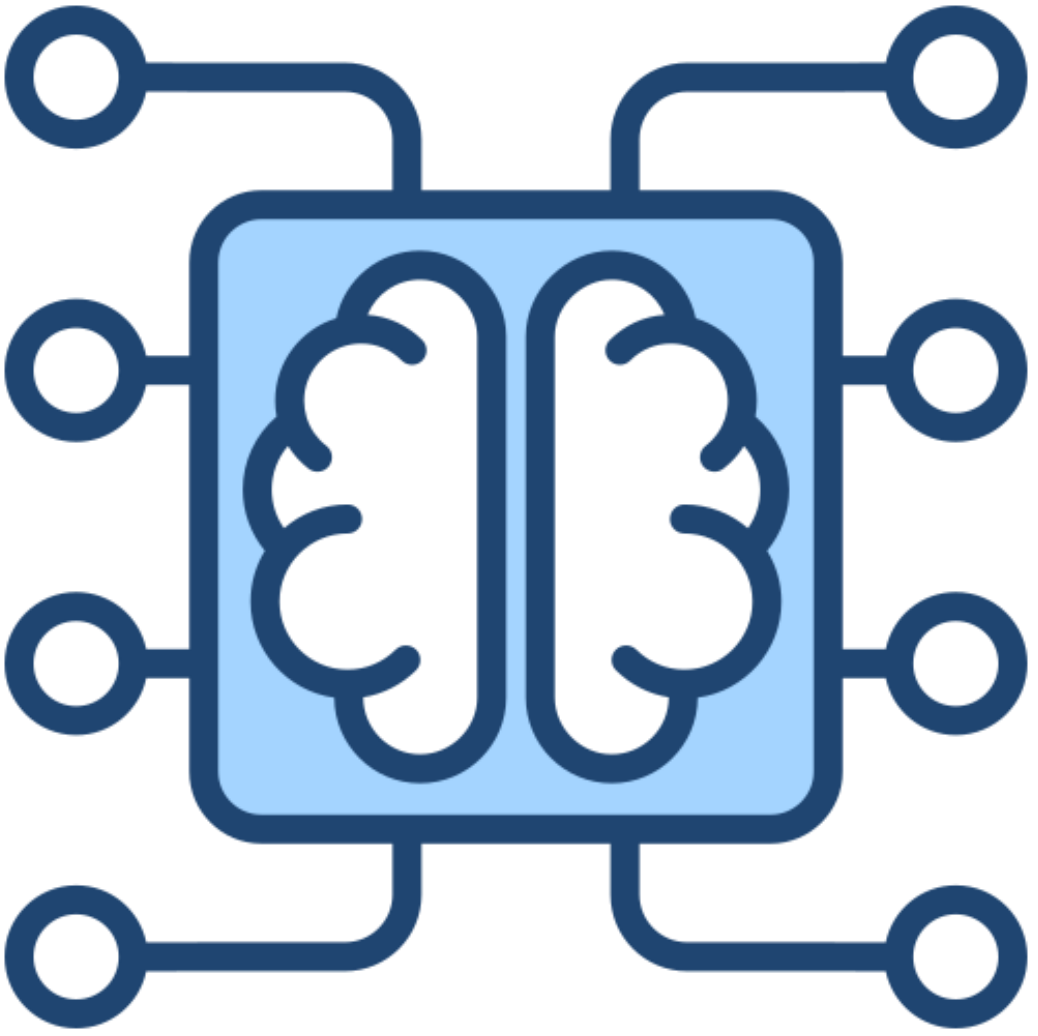
THE JOURNEY OF MACHINE LEARNING

Andrei Rech

Introdução ao Machine Learning

Principais Modelos e Exemplos de Código

O aprendizado de máquina é uma área da inteligência artificial que permite que computadores aprendam e façam previsões com base em dados. Este eBook apresentará os principais modelos de aprendizado de máquina com exemplos práticos de código, explicados de maneira simples e direta para que você consiga se transformar num verdadeiro profissional da área!



REGRESSÃO LINEAR

01

Regressão Linear

A regressão linear é um modelo utilizado para prever valores contínuos. A ideia é encontrar a linha que melhor se ajusta aos dados. A regressão linear assume uma relação linear entre a variável independente (*input*) e a variável dependente (*output*). É amplamente utilizada em áreas como economia e engenharia, onde a previsão de valores futuros é essencial. A equação da linha é dada por $y = ax + b$, onde a é a inclinação e b é o intercepto.

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Carregar dados
data = pd.read_csv('house_prices.csv')
X = data[['size']]
y = data['price']

# Dividir dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Treinar o modelo
model = LinearRegression()
model.fit(X_train, y_train)

# Fazer previsões
y_pred = model.predict(X_test)

# Avaliar o modelo
mse = mean_squared_error(y_test, y_pred)
print(f"Erro Quadrático Médio: {mse}")
```

REGRESSÃO LOGÍSTICA

02

Regressão Logística

A regressão logística é usada para tarefas de classificação, como prever se um e-mail é spam ou não. Apesar do nome, a regressão logística é usada para classificação binária. Ela estima a probabilidade de uma determinada classe ou evento, aplicando a função sigmoide para mapear qualquer valor real para o intervalo (0, 1). É amplamente utilizada em setores como saúde para diagnóstico de doenças e marketing para segmentação de clientes.

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

# Carregar dados
data = pd.read_csv('emails.csv')
X = data.drop('spam', axis=1)
y = data['spam']

# Dividir dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Treinar o modelo
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Fazer previsões
y_pred = model.predict(X_test)

# Avaliar o modelo
accuracy = accuracy_score(y_test, y_pred)
print(f"Acurácia: {accuracy}")
```

ÁRVORES DE DECISÃO

03

Árvores de Decisão

As árvores de decisão são modelos que separam os dados com base em perguntas binárias, criando uma estrutura similar a uma árvore. Cada nó interno da árvore representa uma característica ou atributo do dado, cada ramo representa uma decisão ou regra, e cada folha representa um resultado ou classe. Árvores de decisão são populares por sua interpretabilidade e facilidade de visualização, sendo amplamente utilizadas em finanças para análise de risco e em bioinformática para classificação de espécies.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

# Carregar dados
data = pd.read_csv('loan_approval.csv')
X = data.drop('approved', axis=1)
y = data['approved']

# Dividir dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Treinar o modelo
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Fazer previsões
y_pred = model.predict(X_test)

# Avaliar o modelo
report = classification_report(y_test, y_pred)
print(report)
```

REDES NEURAIS

04

Redes Neurais

Redes neurais são modelos inspirados no cérebro humano, usados para tarefas complexas como reconhecimento de imagens e processamento de linguagem natural. Uma rede neural consiste em camadas de neurônios artificiais, onde cada neurônio aplica uma transformação linear seguida por uma função de ativação não linear. Redes neurais podem ser treinadas usando técnicas de *backpropagation* e são a base dos modelos de *deep learning* que têm revolucionado áreas como visão computacional e tradução automática.

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

# Carregar dados
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0
y_train, y_test = to_categorical(y_train), to_categorical(y_test)

# Construir o modelo
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compilar e treinar o modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.2)

# Avaliar o modelo
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Acurácia no teste: {test_acc}")
```

K-NEAREST NEIGHBORS (KNN)

05

k-NN

KNN é um modelo simples e intuitivo que classifica os dados com base na proximidade dos vizinhos mais próximos. O KNN não assume nenhuma distribuição subjacente dos dados, sendo um método não paramétrico. Ele é especialmente útil em problemas onde a decisão depende fortemente dos casos mais próximos, como na recomendação de produtos e na classificação de imagens. No entanto, pode ser computacionalmente caro para grandes *datasets*, pois requer a análise de todos os pontos de treino para cada previsão.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Carregar dados
data = pd.read_csv('iris.csv')
X = data.drop('species', axis=1)
y = data['species']

# Dividir dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Treinar o modelo
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

# Fazer previsões
y_pred = model.predict(X_test)

# Avaliar o modelo
accuracy = accuracy_score(y_test, y_pred)
print(f"Acurácia: {accuracy}")
```

AGRADECIMENTOS

Se leu até aqui – Obrigado!

Esse ebook foi gerado inteiramente por IA, sendo diagramado por um humano.

Esse conteúdo foi gerado com o objetivo de aprender a utilizar as inteligências artificiais existentes e integrar as mesmas em um projeto, idealizado pelo curso de Aprendizado de Máquina da DIO.

Não houve verificações nem validações sobre o conteúdo aqui presente, logo, pode haver erros no mesmo.

Andrei Rech