**Relatório de Aula Prática**

**Fundamentos de Machine Learning**

*Hudson de Campos Cruz*

Github: <https://github.com/hcamposcruz/-fund_machinelearning.git>

**Script de árvore de decisão**

Este script tem como objetivo utilizar árvores de decisão para classificar solicitações de empréstimo com base nas informações fornecidas. O script realiza a leitura de um arquivo CSV contendo dados dos clientes, processa esses dados, treina um modelo de árvore de decisão e avalia seu desempenho.

Script:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# 1. Importação dos Dados a partir de um Arquivo CSV (separado por ;)

df = pd.read\_csv('dados\_clientes.csv', delimiter=';')

# 2. Carregamento e Preparação dos Dados

# Convertendo variáveis categóricas em numéricas

df['historico\_credito'] = df['historico\_credito'].map({'bom': 1, 'ruim': 0})

df['emprego'] = df['emprego'].map({'empregado': 1, 'desempregado': 0})

df['propriedade'] = df['propriedade'].map({'sim': 1, 'nao': 0})

df['classe'] = df['classe'].map({'conceder': 1, 'negar': 0})

# Separando recursos (X) e rótulos (y)

X = df.drop('classe', axis=1)

y = df['classe']

# 3. Divisão dos Dados em Conjunto de Treinamento e Teste

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# 4. Treinamento da Árvore de Decisão

clf = DecisionTreeClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

# 5. Avaliação da Árvore de Decisão

y\_pred = clf.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Acurácia: {accuracy \* 100:.2f}%')

# 6. Classificação e Predição

def classificar\_emprestimo(idade, renda, historico\_credito, emprego, propriedade):

    dados\_cliente = pd.DataFrame({

        'idade': [idade],

        'renda': [renda],

        'historico\_credito': [1 if historico\_credito == 'bom' else 0],

        'emprego': [1 if emprego == 'empregado' else 0],

        'propriedade': [1 if propriedade == 'sim' else 0]

    })

    resultado = clf.predict(dados\_cliente)

    return 'conceder' if resultado[0] == 1 else 'negar'

# Exemplo de uso da função de classificação

resultado = classificar\_emprestimo(30, 5000, 'bom', 'empregado', 'sim')

print(f'Resultado da classificação: {resultado}')

Para execução deste script foi necessário a instalação das bibliotecas “**pandas**” e “**scikit-learn**”:

* pip install pandas
* pip install scikit-learn

Código disponível no GitHub: <https://github.com/hcamposcruz/-fund_machinelearning.git>

**Importação das bibliotecas**

* **import pandas as pd**: Importa a biblioteca pandas e a atribui ao alias **pd**. Pandas é utilizada para manipulação e análise de dados.
* **from sklearn.model\_selection import train\_test\_split**: Importa a função **train\_test\_split** da biblioteca **scikit-learn**, usada para dividir os dados em conjuntos de treinamento e teste.
* **from sklearn.tree import DecisionTreeClassifier**: Importa a classe **DecisionTreeClassifier** da biblioteca **scikit-learn**, usada para criar e treinar o modelo de árvore de decisão.
* **from sklearn.metrics import accuracy\_score**: Importa a função **accuracy\_score** da biblioteca **scikit-learn**, usada para calcular a acurácia do modelo.

**Carregamento e Preparação dos dados**

* **pd.read\_csv('dados\_clientes.csv', delimiter=';')**: Carrega os dados a partir de um arquivo CSV delimitado por “ ; ”.
* **map**: Converte as variáveis categóricas (**historico\_credito, emprego, propriedade, classe**) para valores numéricos.
* **X = df.drop('classe', axis=1)**: Cria a matriz de características (X) excluindo a coluna classe.
* **y = df['classe']**: Cria o vetor de rótulos (y) contendo a coluna classe.

**Divisão dos dados**

* **train\_test\_split**: Divide os dados em conjunto de treinamento e teste.
* **test\_size=0.3**: Define que 30% dos dados serão usados para teste.
* **random\_state=42**: Garante que a divisão dos dados seja reprodutível.

**Treinamento**

* **DecisionTreeClassifier**: Cria um modelo de árvore de decisão.
* **random\_state=42**: Garante reprodutibilidade.
* **fit(X\_train, y\_train)**: Treina o modelo usando o conjunto de treinamento.

**Avaliação da Árvore**

* **predict(X\_test)**: Gera previsões para o conjunto de teste.
* **accuracy\_score(y\_test, y\_pred)**: Calcula a acurácia comparando as previsões com os rótulos reais.
* **print(f'Acurácia: {accuracy \* 100:.2f}%')**: Exibe a acurácia em porcentagem.

**Função classificar\_emprestimo**

* **pd.DataFrame**: Cria um DataFrame para os dados do cliente.
* **clf.predict**: Usa o modelo treinado para prever se o empréstimo será concedido ou não.
* **Retorno**: Retorna 'conceder' ou 'negar' com base na previsão.

**Uso da função classificar\_emprestimo**

* **classificar\_emprestimo**(30, 5000, 'bom', 'empregado', 'sim'): Exemplo de chamada da função com dados específicos de um cliente.
* **print(f'Resultado da classificação: {resultado}')**: Exibe o resultado da classificação.

Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa