CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
ALGORITMOS E LÓGICA DE PROGRAMAÇÃO
PROF. ISMAEL SANTANA SILVA (ismaelsantana@cefetmg.br)

Prática 1 – Análise dos Parâmetros do classificador kNN

Observações:

Avaliar o impacto de diferentes configurações de parâmetros do algoritmo KNN sobre o desempenho da classificação em conjuntos de dados reais, utilizando arquivos .csv. Ou seja, avaliar o desempenho do algoritmo KNN em diferentes configurações de parâmetros, como:

- Número de vizinhos (k)
- Tipo de distância (métrica)
- Tipo de ponderação (pesos)

Ferramentas utilizadas:

- Google Colab (Python Notebook)
- Python
- Bibliotecas: pandas, scikit-learn, matplotlib, numpy

1. Preparação dos dados

Você irá utilizar os seguintes conjuntos de dados (arquivos .csv):

- iris.csv (classe: species)
- WineQT.csv (classe: quality)
- evasao.csv (classe: Evadiu)

Eles devem conter:

- Apenas atributos numéricos
- Uma coluna de classe (y)

2. Crie um python notebook no Google Colab:
   2.1. https://colab.research.google.com/

3. Exemplo com o arquivo **`iris.csv`**. Crie uma célula de código e execute cada trecho de código a seguir. Acrescente uma célula de comentário e comente o resultado.
   3.1. Exemplo carregando o arquivo `iris.csv`

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Carregar os dados
df = pd.read_csv('iris.csv')
```

```
# Verificar as primeiras linhas
print(df.head())

# Separar atributos e classe
X = df.drop(columns=['species'])  # substitua 'species' pelo nome correto da coluna da classe
y = df['species']

# Normalização
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Divisão treino/teste
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

3.2. Avaliação dos parâmetros

3.2.1. Número *k* de vizinhos (n_neighbors)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

k_values = range(1, 31)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracies.append(acc)

plt.plot(k_values, accuracies, marker='o')
plt.xlabel('Número de Vizinhos (k)')
plt.ylabel('Acurácia')
plt.title('Acurácia em função de k')
plt.grid()
plt.xticks(ticks=range(1, 31, 1))
plt.show()
```

3.2.2. Distâncias (*metric*)

Para executar esse trecho de código, escolha o melhor k (`neighbors`) a partir do experimento anterior.

```
metrics = ['euclidean', 'manhattan']
for metric in metrics:
    knn = KNeighborsClassifier(n_neighbors=5, metric=metric)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"Distância: {metric} | Acurácia: {acc:.4f}")
```

3.2.3. Pesos (`weights`)

```
weights_list = ['uniform', 'distance']
for w in weights_list:
    knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean', weights=w)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"Peso: {w} | Acurácia: {acc:.4f}")
```

3.2.4.Sem normalização (comparação opcional)

Remova a etapa de ***StandardScaler*** e refaça os testes para observar o impacto da escala dos atributos.

4. Replicação. Após finalizar com o **`iris.csv`**, repita todos os testes com os arquivos:
   4.1. WineQT.csv (classe: quality)
   4.2. evasao.csv (classe: Evadiu)

5. Entrega Final:
   5.1. O python notebook (.ipynb) deve ser entregue executado com os gráficos e saída das células
   5.2. Resumo no final com:
      5.2.1. Discussão sobre qual configuração teve melhor desempenho e por quê para cada conjunto de dados
      5.2.2.Observações sobre a influência da normalização