1. Jelaskan cara kerja dari algoritma tersebut!
Jawab:
K-Nearest Neighbors (KNN) adalah algoritma machine learning yang digunakan untuk masalah klasifikasi dan regresi. Algoritma ini termasuk dalam kategori lazy learning karena tidak memiliki proses pelatihan yang eksplisit. Sebagai gantinya, KNN mengklasifikasikan data baru berdasarkan seberapa mirip data tersebut dengan data yang ada dalam dataset. Cara kerja KNN adalah sebagai berikut:

- Inisialisasi: Tentukan nilai k yang merupakan jumlah tetangga terdekat yang akan digunakan untuk prediksi.
- Pengukuran Jarak: Untuk data baru yang ingin diklasifikasikan atau diprediksi, hitung jarak antara data tersebut dan semua data lain dalam dataset. Jarak ini dapat dihitung menggunakan beberapa metrik, termasuk:
    - Euclidean Distance: Jarak lurus antara dua titik dalam ruang Euclidean.
    - Manhattan Distance: Jarak antara dua titik di grid berbentuk kotak (jumlah dari perbedaan absolut antara titik-titik pada masing-masing dimensi).
    - Minkowski Distance: Generalisasi dari Euclidean dan Manhattan distance, dengan parameter untuk menyesuaikan antara keduanya.
- Pilih Tetangga Terdekat: Urutkan semua jarak yang dihitung dan pilih k data poin terdekat.
- Voting (Klasifikasi): Untuk tugas klasifikasi, lakukan voting mayoritas di antara k tetangga terdekat. Data baru akan diberi label sesuai dengan kategori yang paling umum di antara tetangga-tetangga tersebut.
- Prediksi (Regresi): Untuk tugas regresi, ambil rata-rata nilai dari k tetangga terdekat untuk memprediksi nilai data baru.

4. Bandingkan hasil evaluasi model pada nomor 2 dan 3, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!
Jawab:

Menggunakan Implementasi From Scratch

```
import importlib
import knn
importlib.reload(knn)

from knn import KNN

knn = KNN(k=3, distance_metric="euclidean")

# Using holdout validation
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
ho_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {ho_accuracy}")

# Using k-fold cross-validation
cv_accuracy = cross_val_score(knn, X, y, cv=kfold, scoring="accuracy").mean()
print(f"Cross-validation Accuracy: {cv_accuracy}")
```

✓ 0.7s

```
Accuracy: 0.7049180327868853
Cross-validation Accuracy: 0.7946448087431695
```

Menggunakan Implementasi dari Library

```
knn_lib = KNeighborsClassifier(n_neighbors=3, metric="euclidean")

# Using holdout validation
knn_lib.fit(X_train, y_train)
y_pred_lib = knn_lib.predict(X_test)
ho_accuracy_lib = accuracy_score(y_test, y_pred_lib)
print(f'Accuracy: {ho_accuracy_lib}')

# Using k-fold cross-validation
cv_accuracy_lib = cross_val_score(knn_lib, X, y, cv=kfold, scoring="accuracy").mean()
print(f'Cross-validation Accuracy: {cv_accuracy_lib}')
```

✓ 0.1s

```
Accuracy: 0.7049180327868853
Cross-validation Accuracy: 0.7946448087431695
```

Kedua model memberikan hasil akurasi yang sama, baik pada holdout validation maupun pada cross-validation.

5. Jelaskan improvement apa saja yang bisa Anda lakukan untuk mencapai hasil yang lebih baik dibandingkan dengan hasil yang Anda punya saat ini! Improvement yang dimaksud tidak terbatas pada bagaimana algoritma diimplementasikan, namun juga mencakup tahap sebelum modeling and validation.

Jawab:

- Optimisasi jumlah tetangga (k) menggunakan metode seperti grid search atau random search bisa membantu meningkatkan akurasi.
- Melakukan eksplorasi untuk membuat fitur baru yang lebih informatif atau menggabungkan fitur yang ada untuk membantu model belajar lebih baik.