Francesco Michael Kusuma

13522038

kNN (k-Nearest-Neighbors)

Cara Kerja:

1. Tentukan jumlah tetangga k
2. Hitung jarak antara data test dengan training dataset menggunakan metrik jarak, misalnya, *Euclidean distance, Manhattan distance, atau Minkowski distance.*
3. Urutkan *training dataset* berdasarkan jarak.
4. Pilih k data terdekat.
5. Klasifikasikan data baru dengan mayoritas kelas dari k tetangga terdekat.

Perbandingan:

- *Scratch*
  - *Euclidean distance*

[19]:
```
from knn import KNNScratch

knn_euclidean_scratch = KNNScratch(neighbors=3, metric='euclidean')
knn_euclidean_scratch.fit(X_train, y_train)
y_pred_knn_euclidean_scratch = knn_euclidean_scratch.predict(X_test)

validate_model(knn_euclidean_scratch, method_name="kNN with Euclidean distance from Scratch")
```

```
Hold-Out Validation (kNN with Euclidean distance from Scratch):
F1 Score: 0.88
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        72
           1       1.00      0.79      0.88        42

    accuracy                           0.92       114
   macro avg       0.94      0.89      0.91       114
weighted avg       0.93      0.92      0.92       114


K-Fold Cross-Validation (kNN with Euclidean distance from Scratch):
F1 Scores for each fold: [0.9230769230769231, 0.9393939393939394, 0.9295774647887324, 0.8235294117647
058, 0.8135593220338984]
Mean F1 Score: 0.8858274122116399
Standard Deviation of F1 Score: 0.05527148218059224
```

- *Manhattan distance*

[21]:
```
knn_manhattan_scratch = KNNScratch(neighbors = 3, metric = 'manhattan')
knn_manhattan_scratch.fit(X_train, y_train)
y_pred_knn_manhattan_scratch = knn_manhattan_scratch.predict(X_test)

validate_model(knn_manhattan_scratch, method_name="kNN with Manhattan distance from Scratch")
```

```
Hold-Out Validation (kNN with Manhattan distance from Scratch):
F1 Score: 0.8648648648648649
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        72
           1       1.00      0.76      0.86        42

    accuracy                           0.91       114
   macro avg       0.94      0.88      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (kNN with Manhattan distance from Scratch):
F1 Scores for each fold: [0.9375, 0.9538461538461539, 0.9142857142857143, 0.8615384615384616, 0.8]
Mean F1 Score: 0.8934340659340659
Standard Deviation of F1 Score: 0.056166666995138954
```

- *Minkowski distance*

[23]:
```
knn_minkowski_scratch = KNNScratch(neighbors = 5, metric = 'minkowski')
knn_minkowski_scratch.fit(X_train, y_train)
y_pred_knn_minkowski_scratch = knn_minkowski_scratch.predict(X_test)

validate_model(knn_minkowski_scratch, method_name="kNN with Minkowski distance from Scratch")
```

```
Hold-Out Validation (kNN with Minkowski distance from Scratch):
F1 Score: 0.868421052631579
              precision    recall  f1-score   support

           0       0.89      0.99      0.93        72
           1       0.97      0.79      0.87        42

    accuracy                           0.91       114
   macro avg       0.93      0.89      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (kNN with Minkowski distance from Scratch):
F1 Scores for each fold: [0.9230769230769231, 0.9090909090909091, 0.9459459459459459, 0.8695652173913
043, 0.7636363636363637]
Mean F1 Score: 0.8822630718282891
Standard Deviation of F1 Score: 0.0643032552295382
```

- *Library*
  - *Euclidean distance*

[20]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_euclidean = KNeighborsClassifier(n_neighbors = 3, metric = 'euclidean')
knn_euclidean.fit(X_train, y_train)
y_pred_knn_euclidean = knn_euclidean.predict(X_test)

validate_model(knn_euclidean, method_name="kNN with Euclidean distance from Library")
```

```
Hold-Out Validation (kNN with Euclidean distance from Library):
F1 Score: 0.88
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        72
           1       1.00      0.79      0.88        42

    accuracy                           0.92       114
   macro avg       0.94      0.89      0.91       114
weighted avg       0.93      0.92      0.92       114


K-Fold Cross-Validation (kNN with Euclidean distance from Library):
F1 Scores for each fold: [0.9230769230769231, 0.9393939393939394, 0.9295774647887324, 0.8235294117647
058, 0.8135593220338984]
Mean F1 Score: 0.8858274122116399
Standard Deviation of F1 Score: 0.05527148218059224
```

  - *Manhattan distance*

[22]:
```
knn_manhattan = KNeighborsClassifier(n_neighbors = 3, metric = 'manhattan')
knn_manhattan.fit(X_train, y_train)
y_pred_knn_manhattan = knn_manhattan.predict(X_test)

validate_model(knn_manhattan, method_name="kNN with Manhattan distance from Library")
```

```
Hold-Out Validation (kNN with Manhattan distance from Library):
F1 Score: 0.8648648648648649
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        72
           1       1.00      0.76      0.86        42

    accuracy                           0.91       114
   macro avg       0.94      0.88      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (kNN with Manhattan distance from Library):
F1 Scores for each fold: [0.9375, 0.9538461538461539, 0.9142857142857143, 0.8615384615384616, 0.8]
Mean F1 Score: 0.8934340659340659
Standard Deviation of F1 Score: 0.056166666995138954
```

- *Minkowski distance*

Library

[24]:
```
knn_minkowski = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski')
knn_minkowski.fit(X_train, y_train)
y_pred_knn_minkowski = knn_minkowski.predict(X_test)

validate_model(knn_minkowski, method_name="kNN with Minkowski distance from Library")
```

```
Hold-Out Validation (kNN with Minkowski distance from Library):
F1 Score: 0.868421052631579
              precision    recall  f1-score   support

           0       0.89      0.99      0.93        72
           1       0.97      0.79      0.87        42

    accuracy                           0.91       114
   macro avg       0.93      0.89      0.90       114
weighted avg       0.92      0.91      0.91       114


K-Fold Cross-Validation (kNN with Minkowski distance from Library):
F1 Scores for each fold: [0.9230769230769231, 0.9253731343283582, 0.9166666666666666, 0.8529411764705
882, 0.7636363636363637]
Mean F1 Score: 0.8763388528357801
Standard Deviation of F1 Score: 0.06239301703039033
```

Dari hasil *F1 Score*, baik menggunakan metrik *Euclidean, Manhattan, atau Minkowski,* penggunaan kNN dengan *scratch* menghasilkan skor yang sama dengan penggunaan kNN secara *library*.

Improvement:

Improvement yang dapat dilakukan pada algoritma kNN secara *scratch* dapat dilakukan dengan memperbaiki cara untuk mencari jarak terdekat pada titik test.