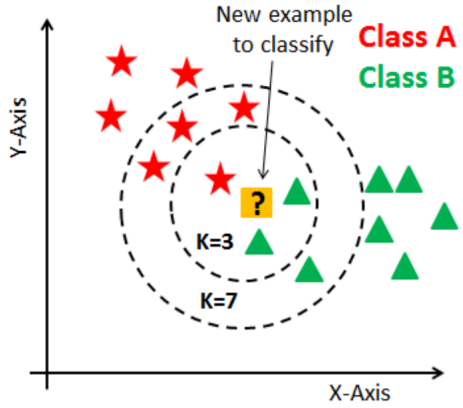
K Nearest Neighbors (KNN)

Cara Kerja



Sumber:
https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8

Pada tahap fitting, model KNN hanya menyimpan data dan kelas target yang berkorespondensi dengan data tersebut. Algoritma utamanya terjadi pada saat tahap prediksi. Berikut tahapan prediksi model KNN:

1. Pada data baru yang diberikan, KNN menghitung jarak data baru tersebut terhadap setiap data yang disimpan pada tahap fitting.
2. Daftar jarak tersebut diurutkan dari yang terkecil. Artinya, elemen-elemen pertama dalam daftar adalah data-data terdekat dari data baru yang diberikan. Jarak dapat dihitung dengan jarak euclidean, manhattan, atau minkowski.
3. Model memilih k data dengan jarak terdekat dengan data baru yang ingin diklasifikasikan.
4. Kelas data baru yang ingin diprediksi ditentukan dengan cara voting kelas mayoritas di antara k data terdekat tersebut.

Evaluasi Model

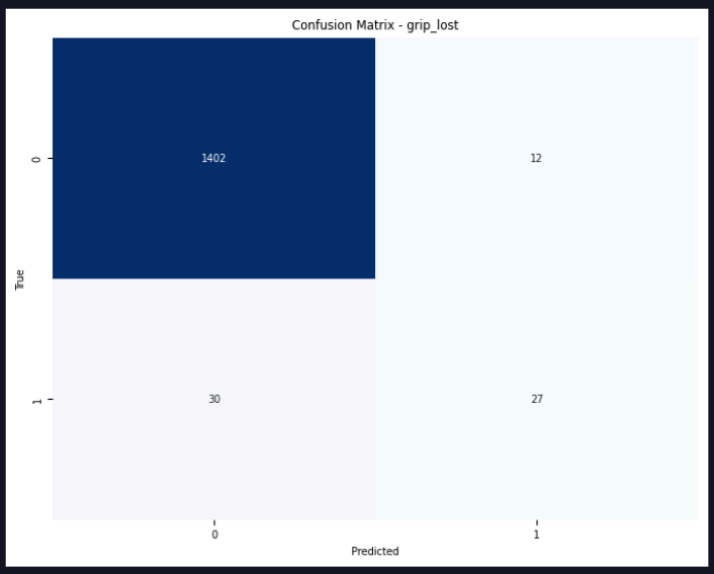
Evaluasi	Hasil
Model from Scratch	

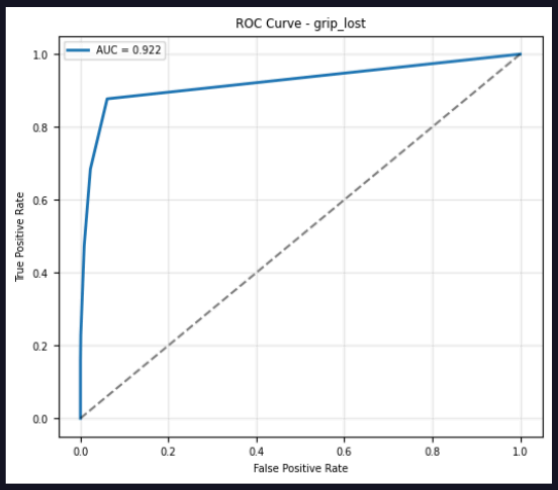
Hold-out validation untuk grip_lost

```
============================================================
Hold-Out Validation for grip_lost:
============================================================
Training time: 0.001996278762817383 seconds
Prediction time: 2.3094820976257324 seconds
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1414
           1       0.69      0.47      0.56        57

    accuracy                           0.97      1471
   macro avg       0.84      0.73      0.77      1471
weighted avg       0.97      0.97      0.97      1471
```





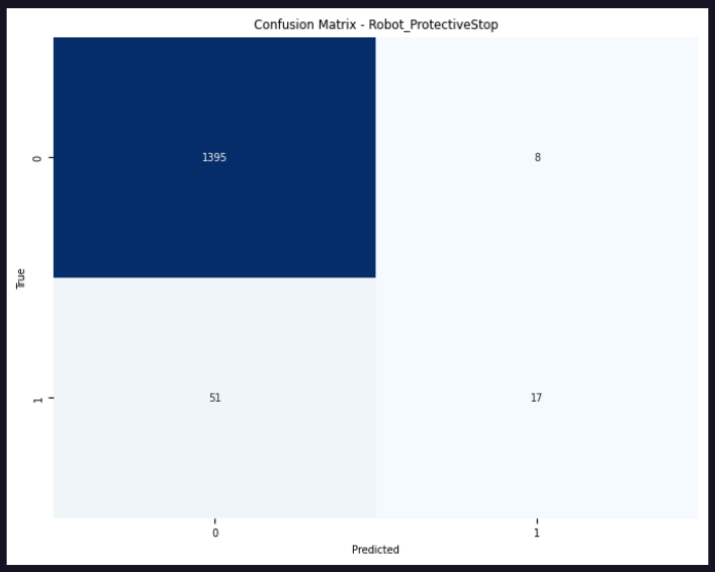

```
AUC-ROC: 0.9224
```

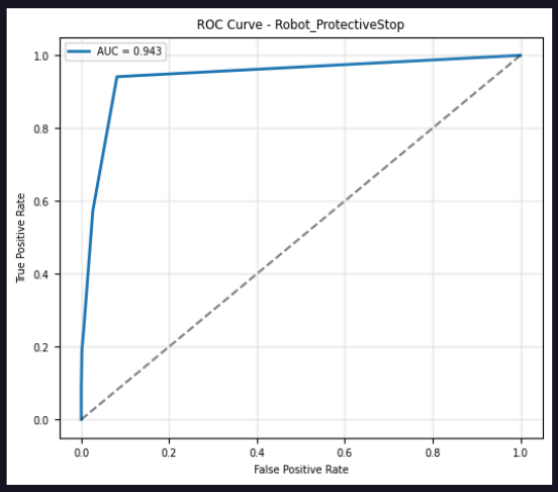
Hold out validation untuk Robot_ProtectiveStop

```
============================================================
Hold-Out Validation for Robot_ProtectiveStop:
============================================================
Training time: 0.0029976367950439453 seconds
Prediction time: 2.3029496669769287 seconds
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      1403
           1       0.68      0.25      0.37        68

    accuracy                           0.96      1471
   macro avg       0.82      0.62      0.67      1471
weighted avg       0.95      0.96      0.95      1471
```






```
AUC-ROC: 0.9429
```

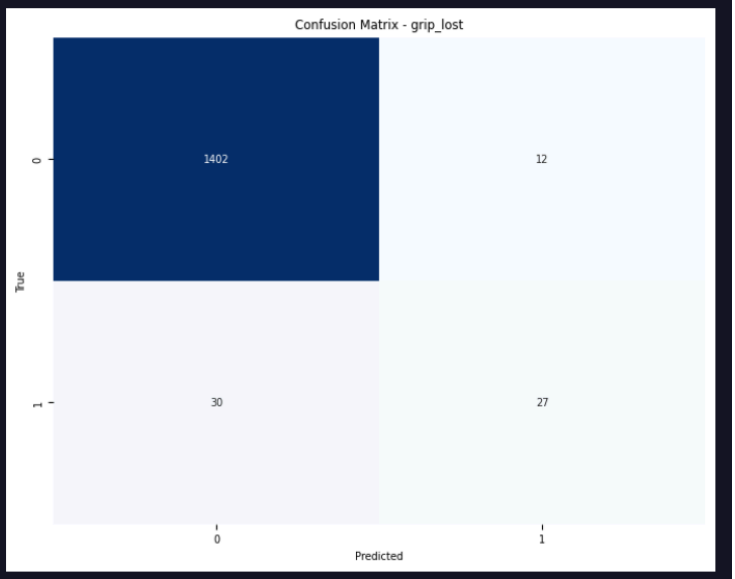
K-fold cross validation untuk grip_lost	``` == Cross Validation for grip_lost: == Model Performance (5-Fold Cross Validation): fit_time: [0.00200105 0.00199914 0.00099707 0.00153756 0.00399995] score_time: [2.47326875 2.64387369 2.23221064 3.51442313 5.59420013] test_precision: [0.69230769 0.8 0.68571429 0.75 0.74074074] Average test_precision: 0.73 test_recall: [0.47368421 0.5 0.54545455 0.42857143 0.45454545] Average test_recall: 0.48 test_f1: [0.5625 0.61538462 0.60759494 0.54545455 0.56338028] Average test_f1: 0.58 ```
K-fold cross validation untuk Robot_ProtectiveStop	``` == Cross Validation for Robot_ProtectiveStop: == Model Performance (5-Fold Cross Validation): fit_time: [0.00099874 0.00099373 0.0010004 0.00100279 0.00115085] score_time: [2.77496719 2.61382675 2.59141111 2.66289234 2.21306896] test_precision: [0.68 0.51851852 0.48 0.55555556 0.5] Average test_precision: 0.55 test_recall: [0.25 0.26415094 0.2 0.21276596 0.24] Average test_recall: 0.23 test_f1: [0.3655914 0.35 0.28235294 0.30769231 0.32432432] Average test_f1: 0.33 ```
Model Scikit-Learn	

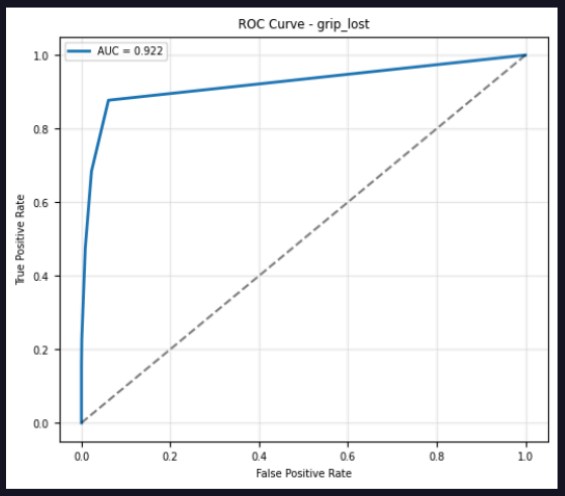
Hold-out validation untuk grip_lost

```
============================================================
Hold-Out Validation for grip_lost:
============================================================
Training time: 0.0029926300048828125 seconds
Prediction time: 0.8508744239807129 seconds
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1414
           1       0.69      0.47      0.56        57

    accuracy                           0.97      1471
   macro avg       0.84      0.73      0.77      1471
weighted avg       0.97      0.97      0.97      1471
```





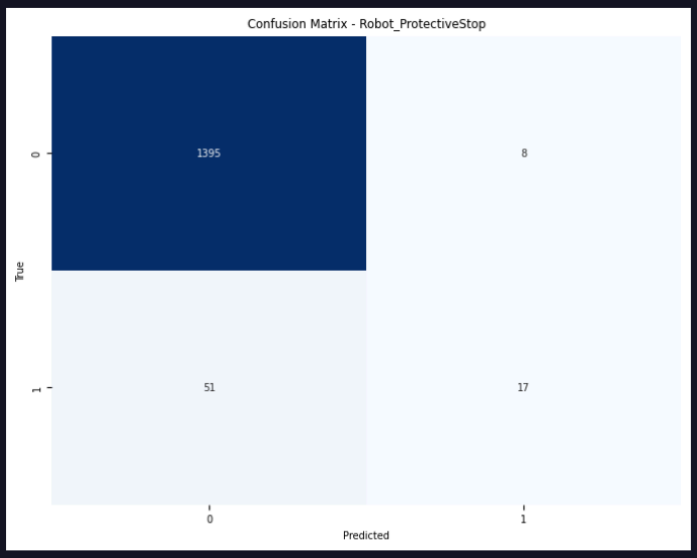

```
AUC-ROC: 0.9224
```

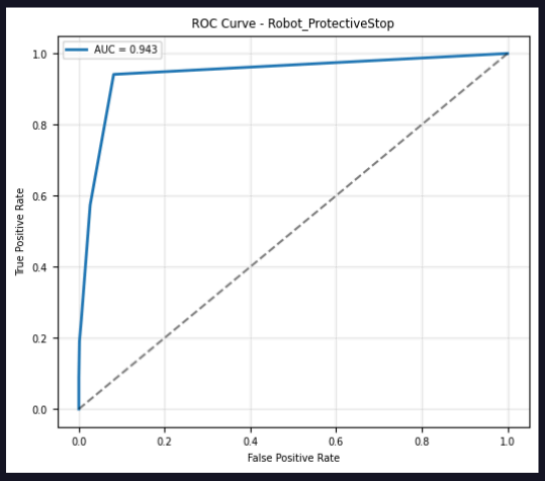
Hold out validation untuk Robot_ProtectiveStop

```
============================================================
Hold-Out Validation for Robot_ProtectiveStop:
============================================================
Training time: 0.0030088424682617188 seconds
Prediction time: 0.15711045265197754 seconds
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      1403
           1       0.68      0.25      0.37        68

    accuracy                           0.96      1471
   macro avg       0.82      0.62      0.67      1471
weighted avg       0.95      0.96      0.95      1471
```






```
AUC-ROC: 0.9429
```

K-fold cross validation untuk grip_lost	``` == Cross Validation for grip_lost: == Model Performance (5-Fold Cross Validation): fit_time: [0.00099397 0.00099945 0.00105739 0.00200009 0.00100088] score_time: [0.0813117 0.07756948 0.0831511 0.11633062 0.06867933] test_precision: [0.69230769 0.8 0.68571429 0.75 0.74074074] Average test_precision: 0.73 test_recall: [0.47368421 0.5 0.54545455 0.42857143 0.45454545] Average test_recall: 0.48 test_f1: [0.5625 0.61538462 0.60759494 0.54545455 0.56338028] Average test_f1: 0.58 ```
K-fold cross validation untuk Robot_ProtectiveStop	``` == Cross Validation for Robot_ProtectiveStop: == Model Performance (5-Fold Cross Validation): fit_time: [0.00199938 0.00099993 0.00200081 0.00099945 0.00200009] score_time: [0.07245827 0.09199238 0.05663466 0.06304765 0.08302188] test_precision: [0.68 0.51851852 0.48 0.55555556 0.5] Average test_precision: 0.55 test_recall: [0.25 0.26415094 0.2 0.21276596 0.24] Average test_recall: 0.23 test_f1: [0.3655914 0.35 0.28235294 0.30769231 0.32432432] Average test_f1: 0.33 ```

Implementasi KNN dari awal ini memiliki kemampuan prediksi yang sama persis dengan implementasi scikit-learn. Besar kemungkinan hal ini terjadi karena algoritmanya yang cukup *straight forward* dan tidak terlalu kompleks sehingga mudah ditiru. Perbedaan yang cukup signifikan terdapat pada waktu eksekusinya, yakni rata-rata waktu prediksi model yang dibuat dari awal adalah sekitar 2 detik dan model dari scikit-learn hanya membutuhkan waktu prediksi di bawah 0.1 detik.

Improvement

Secara implementasi, hal yang dapat dikembangkan lebih lanjut adalah penggunaan struktur data yang lebih efisien untuk mencari k data tetangga terdekat. Struktur data yang dapat digunakan adalah k-d tree atau ball tree. Dengan struktur data berbasis pohon ini, waktu prediksi dapat berkurang karena menghadirkan waktu pencarian yang logaritmik. Namun, struktur data ini dapat meningkatkan waktu fitting karena harus membangun struktur pohon tersebut.

Pengembangan lain adalah menerapkan oversampling dengan SMOTE karena fitur target bersifat unbalanced. Berdasarkan eksperimen, hal ini dapat meningkatkan recall secara drastis namun menurunkan presisi, menyebabkan banyaknya *false positive*.