

HEART DISEASE CLASSIFICATION PROBLEM

**Supervised Learning vs Unsupervised
Learning Model Comparison**

up to 97% Accuracy

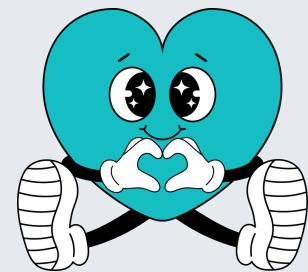
Fikri Adyatma (235150201111015)

Nadhif Rif'at Rasendriya (235150201111074)

Reyno Benedict (235150207111048)

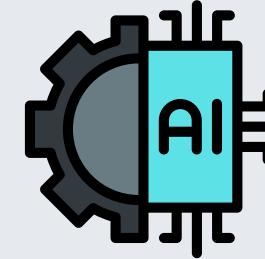
**TIF - A
TIM 8**

Latar Belakang



Penyakit jantung merupakan penyebab utama kematian global, sehingga deteksi dini menjadi sangat krusial untuk mencegah komplikasi serius. Perkembangan teknologi machine learning membuka peluang besar dalam membantu diagnosis penyakit jantung melalui analisis data klinis. Penelitian ini membandingkan dua pendekatan utama machine learning, yaitu supervised learning (menggunakan Random Forest dan K-Nearest Neighbors) dan unsupervised learning (menggunakan K-Means Clustering dan DBSCAN) dalam mengklasifikasi penyakit jantung. Evaluasi dilakukan berdasarkan akurasi, presisi, recall, F1-score, serta metrik clustering seperti silhouette score dan adjusted rand index. Hasil penelitian menunjukkan bahwa pendekatan supervised learning, khususnya Random Forest, memberikan performa terbaik dan paling stabil, sementara unsupervised learning tetap berguna dalam eksplorasi pola data ketika label belum tersedia.

Tujuan



1. Membandingkan performa model supervised learning (Random Forest dan K-Nearest Neighbors) dan unsupervised learning (K-Means Clustering dan DBSCAN) dalam klasifikasi penyakit jantung.
2. Mengevaluasi akurasi, presisi, recall, F1-score, serta kestabilan model supervised learning untuk diagnosis medis.
3. Menganalisis efektivitas unsupervised learning dalam mengidentifikasi pola laten pada data tanpa label.
4. Mengetahui fitur-fitur paling berpengaruh dalam klasifikasi penyakit jantung berdasarkan analisis feature importance.
5. Menguji pengaruh hyperparameter tuning terhadap peningkatan performa model.
6. Memberikan rekomendasi pendekatan machine learning yang paling sesuai untuk pengembangan sistem deteksi dini penyakit jantung.

DATASET



[https://www.kaggle.com/datasets/
johnsmith88/heart-disease-
dataset](https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset)

3 STEPS OF MACHINE LEARNING



Data Exploration (exploratory data analysis or EDA)



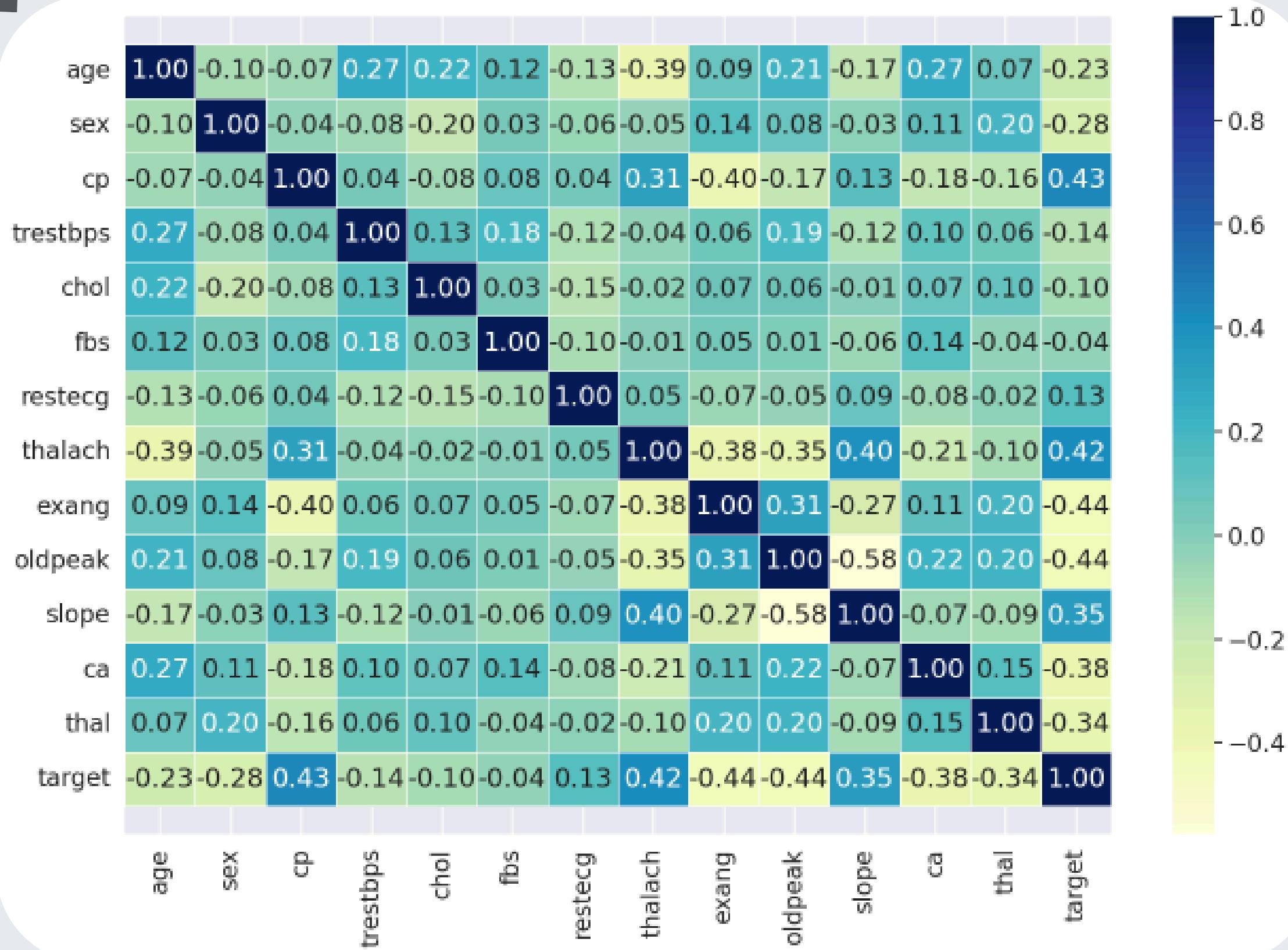
Tujuan dari tahap ini adalah untuk mempelajari lebih dalam mengenai dataset yang sedang digunakan

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|------|-----|-----|-----|----------|------|-----|---------|---------|-------|---------|-------|-----|------|--------|
| 0 | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2 | 3 | 0 |
| 1 | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | 0 |
| 2 | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | 0 |
| 3 | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1 | 3 | 0 |
| 4 | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1020 | 59 | 1 | 1 | 140 | 221 | 0 | 1 | 164 | 1 | 0.0 | 2 | 0 | 2 | 1 |
| 1021 | 60 | 1 | 0 | 125 | 258 | 0 | 0 | 141 | 1 | 2.8 | 1 | 1 | 3 | 0 |
| 1022 | 47 | 1 | 0 | 110 | 275 | 0 | 0 | 118 | 1 | 1.0 | 1 | 1 | 2 | 0 |
| 1023 | 50 | 0 | 0 | 110 | 254 | 0 | 0 | 159 | 0 | 0.0 | 2 | 0 | 2 | 1 |
| 1024 | 54 | 1 | 0 | 120 | 188 | 0 | 1 | 113 | 0 | 1.4 | 1 | 1 | 3 | 0 |

- 1.Pertanyaan / permasalahan apa yang ingin kamu selesaikan?
 - 2.Jenis data apa yang kita miliki dan bagaimana cara menangani berbagai jenis data tersebut?
 - 3.Apa yang hilang dari data dan bagaimana cara menanganinya?
 - 4.Di mana letak outlier dan mengapa hal itu penting untuk diperhatikan?
 - 5.Bagaimana cara menambah, mengubah, atau menghapus fitur agar mendapatkan lebih banyak informasi dari data?

- **age** Usia pasien dalam tahun.
- **sex** Jenis kelamin pasien. (0 = perempuan, 1 = laki-laki)
- **cp** Jenis nyeri dada (chest pain type). 0 = typical angina, 1 = atypical angina, 2 = non-anginal pain, 3 = asymptomatic.
- **trestbps** Tekanan darah saat istirahat (dalam mm Hg).
- **chol** Kadar kolesterol serum dalam darah (mg/dl).
- **fbs** Kadar gula darah puasa lebih dari 120 mg/dl. (1 = ya, 0 = tidak)
- **restecg** Hasil elektrokardiogram saat istirahat. 0 = normal, 1 = ST-T wave abnormality, 2 = left ventricular hypertrophy.
- **thalach** Detak jantung maksimum yang dicapai selama uji latihan.
- **exang** Apakah ada angina (nyeri dada) yang diinduksi oleh olahraga. (1 = ya, 0 = tidak)
- **oldpeak** Penurunan segmen ST akibat aktivitas fisik dibandingkan saat istirahat (indikasi iskemia).
- **slope** Kemiringan segmen ST saat puncak latihan. 0 = upsloping, 1 = flat, 2 = downsloping.
- **ca** Jumlah pembuluh darah besar (0-3) yang terlihat melalui fluoroskopi.
- **thal** Jenis kelainan thalassemia. 1 = normal, 2 = fixed defect, 3 = reversible defect.
- **target** Diagnosis akhir apakah pasien memiliki penyakit jantung atau tidak. 0 = tidak ada penyakit jantung, 1 = memiliki penyakit jantung.

Correlation Matrix

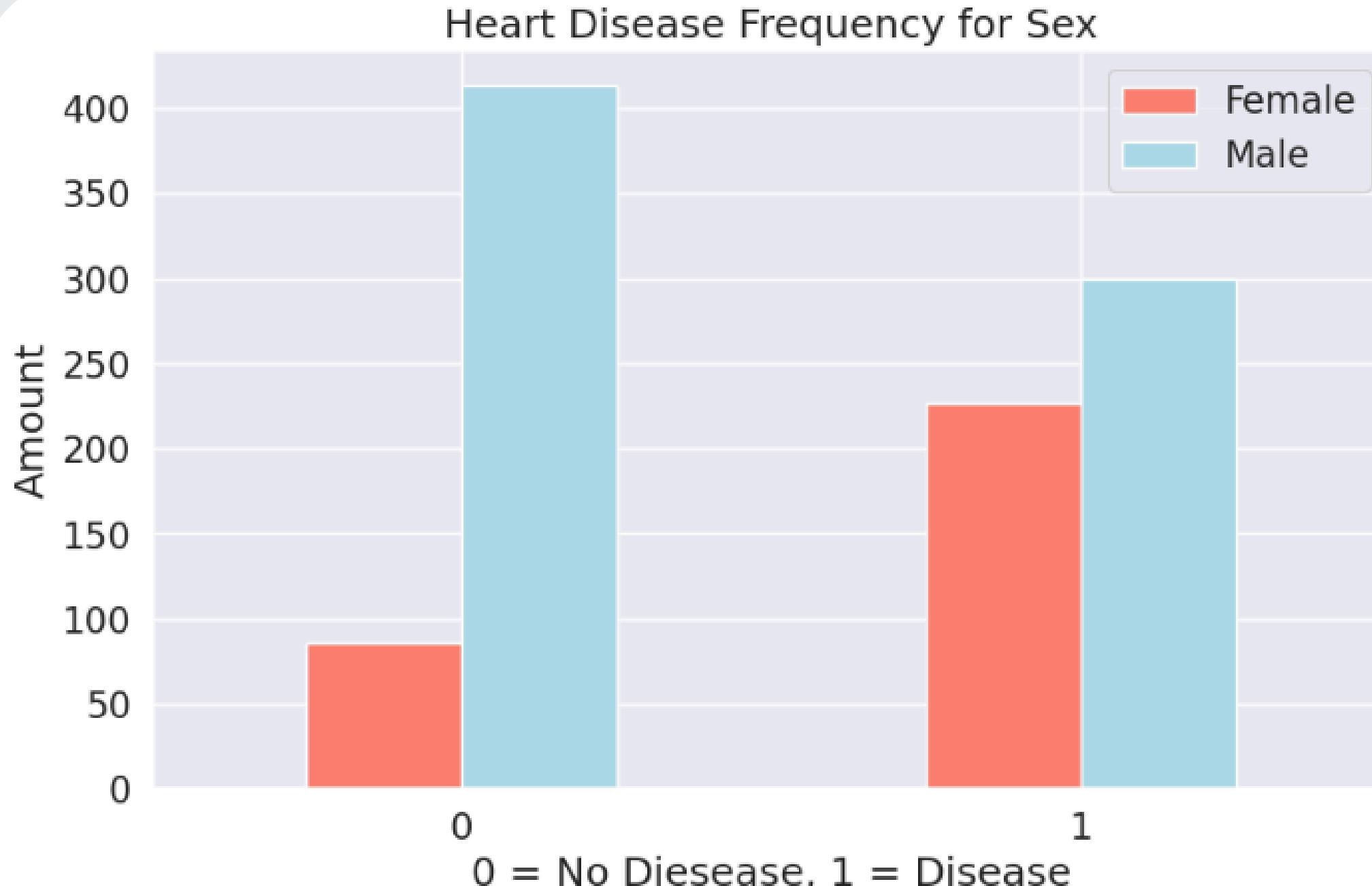


Warna kuning menunjukkan korelasi antara dua variabel rendah, sedangkan warna yang lebih mendekati biru menunjukkan korelasi yang tinggi.

Di sini, terdapat 3 fitur yang memiliki pengaruh paling berpengaruh:

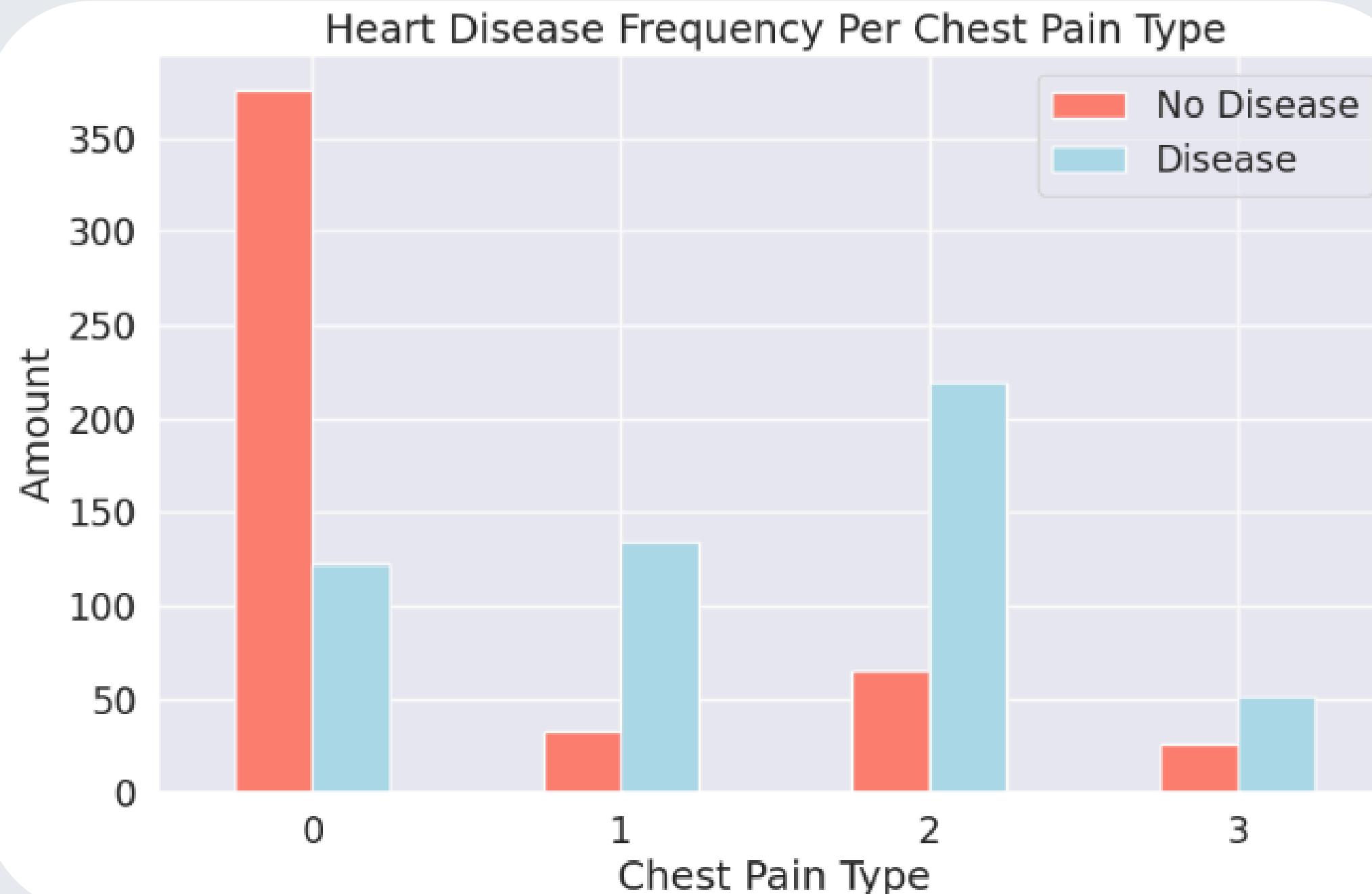
CP, thalach, dan slope.

Analisis Penyakit Jantung Berdasarkan Jenis Kelamin



- Dalam dataset ini, jumlah pria lebih banyak.
- Namun, persentase penderita penyakit jantung lebih tinggi pada wanita.
 - 226 dari 312 wanita (sekitar 72%) menderita penyakit jantung
 - 300 dari 713 pria (sekitar 42%) menderita penyakit jantung
- Meskipun secara jumlah pria lebih banyak, wanita memiliki proporsi penyakit jantung yang jauh lebih besar.

Frekuensi Penyakit Jantung Berdasarkan Jenis Nyeri Dada (Chest Pain Type)



Keterangan Tipe CP:

- CP = 0: Typical angina → Nyeri dada khas akibat kurangnya suplai darah ke jantung
- CP = 1: Atypical angina → Nyeri dada tidak berhubungan langsung dengan jantung
- CP = 2: Non-anginal pain → Nyeri bukan karena jantung, misalnya kejang otot esofagus
- CP = 3: Asymptomatic → Tanpa gejala, tidak menunjukkan tanda-tanda penyakit

Frekuensi Penyakit Jantung Berdasarkan Jenis Nyeri Dada (Chest Pain Type)

- Jenis nyeri dada CP = 1 (atypical angina) dan CP = 2 (non-anginal pain) ternyata lebih sering dikaitkan dengan penyakit jantung dibandingkan CP = 0.
- Pasien dengan CP = 0 (typical angina) justru lebih banyak yang tidak mengidap penyakit jantung, menunjukkan bahwa nyeri dada khas belum tentu menjadi indikator utama.
- Bahkan, pada kategori asymptomatic (CP = 3), penyakit jantung tetap terdeteksi, menegaskan pentingnya pemeriksaan meskipun tanpa gejala.

MODELING

Supervised Learning

Logistic Regression, K-Nearest Neighbors, Random Forest

Kita akan melatih model (mencari pola) menggunakan data latih.
Lalu kita akan mengujinya (menggunakan pola tersebut) dengan data uji.
Kita akan mencoba 3 model machine learning yang berbeda.

Logistic Regression

Logistic Regression

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score

X = df.drop("target", axis=1)
y = df["target"]

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

log_model = LogisticRegression(max_iter=1000)
log_scores = cross_val_score(log_model, X_scaled, y, cv=5, scoring='accuracy')

print("Logistic Regression")
print(f"Mean Accuracy: {log_scores.mean():.4f}")
print(f"Standard Deviation: {log_scores.std():.4f}")
```

Logistic Regression
Mean Accuracy: 0.8459
Standard Deviation: 0.0279

Logistic Regression

Cara Kerja:

- Logistic Regression adalah model linier yang digunakan untuk klasifikasi, bukan regresi (meskipun namanya "regression").
- Bekerja dengan cara menghitung kombinasi linier dari fitur
- Jika probabilitas > 0.5 , maka diprediksi sebagai kelas 1, jika tidak, kelas 0.

Kelebihan:

- Cepat, sederhana, dan cukup efektif untuk dataset linier.
- Bisa memberikan probabilitas klasifikasi (bukan hanya label).

Kekurangan:

- Kurang efektif jika hubungan antara fitur dan label tidak linier.
- Rentan terhadap outlier.

K-Nearest Neighbors (KNN)

▼ K-Nearest Neighbors (KNN)

```
✓ ⏴ from sklearn.neighbors import KNeighborsClassifier  
  
knn_model = KNeighborsClassifier()  
knn_scores = cross_val_score(knn_model, X_scaled, y, cv=5, scoring='accuracy')  
  
print("K-Nearest Neighbors")  
print(f"Mean Accuracy: {knn_scores.mean():.4f}")  
print(f"Standard Deviation: {knn_scores.std():.4f}")
```

→ K-Nearest Neighbors
Mean Accuracy: 0.8332
Standard Deviation: 0.0181

K-Nearest Neighbors (KNN)

Tujuan: Klasifikasi berdasarkan kemiripan data baru dengan data yang sudah ada.

Cara Kerja:

- Untuk mengklasifikasikan satu data baru, KNN:
 - a. Menghitung jarak (misalnya Euclidean) ke semua data dalam training set.
 - b. Memilih k tetangga terdekat (misalnya k=5).
 - c. Menentukan kelas mayoritas dari k tetangga tersebut.
- Tidak ada proses "training" yang nyata – model menyimpan semua data dan melakukan pencarian saat prediksi.

Kelebihan:

- Mudah dipahami dan diimplementasikan.
- Bagus untuk data dengan pola kompleks.

Kekurangan:

- Lambat jika jumlah data besar (karena harus hitung jarak ke semua titik).
- Sangat sensitif terhadap scaling fitur.
- Sulit untuk data dengan dimensi tinggi (curse of dimensionality).

Random Forest

✓ Random Forest

```
[58] from sklearn.ensemble import RandomForestClassifier  
      rf_model = RandomForestClassifier()  
      rf_scores = cross_val_score(rf_model, X, y, cv=5, scoring='accuracy') # Tanpa scaling  
  
      print("Random Forest")  
      print(f"Mean Accuracy: {rf_scores.mean():.4f}")  
      print(f"Standard Deviation: {rf_scores.std():.4f}")
```

→ Random Forest
Mean Accuracy: 0.9971
Standard Deviation: 0.0059

Random Forest

Tujuan: Model ensambel untuk klasifikasi yang lebih kuat dan stabil.

Cara Kerja:

- Random Forest terdiri dari banyak pohon keputusan (Decision Trees).
- Untuk membuat satu pohon:
 - a. Ambil subset acak dari data (dengan penggantian, disebut bootstrap).
 - b. Di tiap node, gunakan subset acak dari fitur untuk split terbaik.
- Untuk prediksi:
 - Semua pohon memberikan voting klasifikasi → hasil akhir adalah kelas mayoritas.

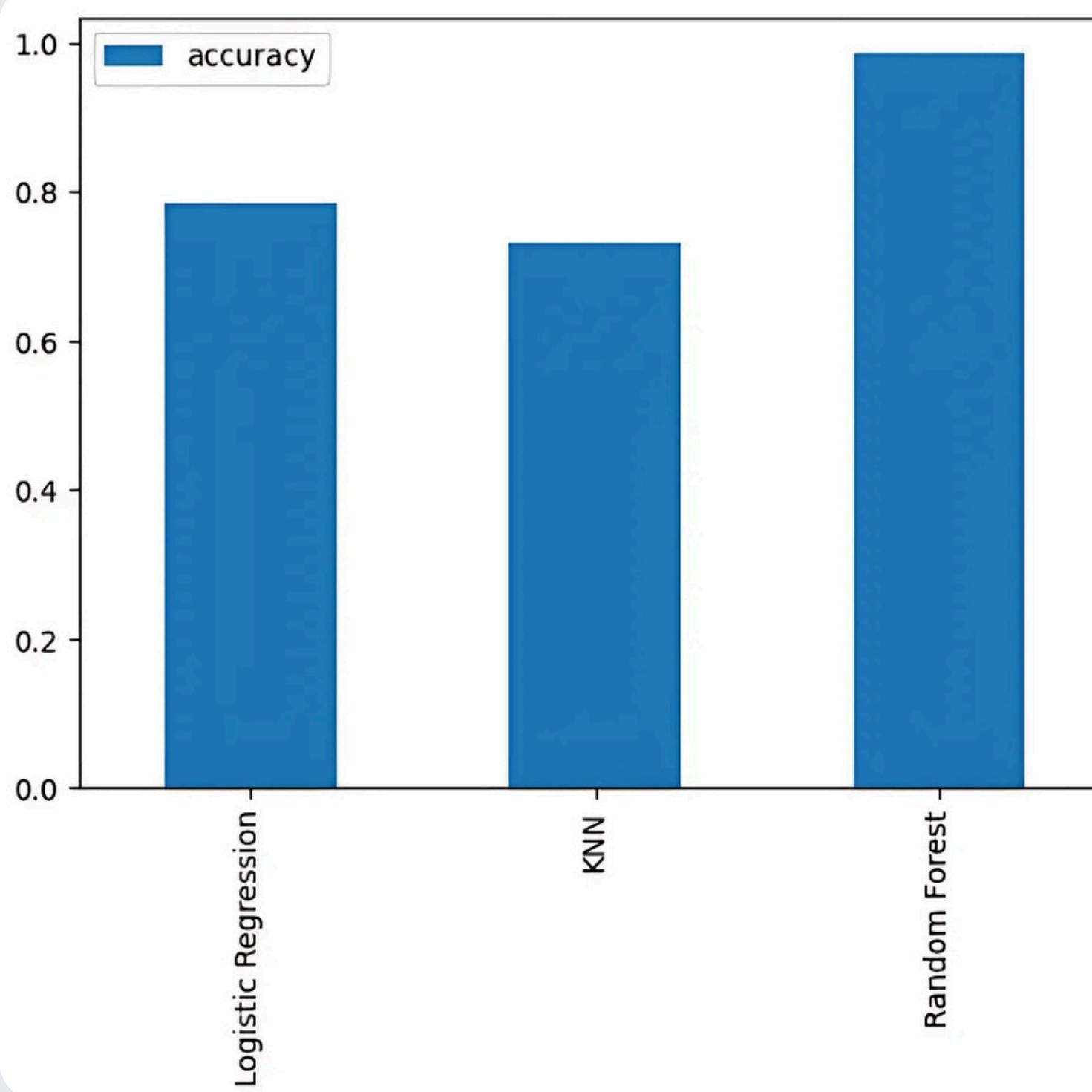
Kelebihan:

- Akurat, bahkan tanpa banyak tuning.
- Tidak mudah overfitting seperti pohon tunggal.
- Bisa menangani data non-linier dan campuran (numerik + kategorikal).

Kekurangan:

- Lebih lambat dibanding model sederhana.
- Kurang interpretatif dibanding logistic regression.

MODEL COMPARISON



Berdasarkan hasil perbandingan model, Random Forest memiliki akurasi tertinggi yaitu **1.0 (100%)**. Namun, karena ini masih tahap pemodelan, kita tetap memerlukan evaluasi tambahan. Kita harus bertanya: apakah model dengan akurasi 1.0 ini benar-benar sebaik itu? Oleh karena itu, kita perlu melakukan validasi menggunakan **Hyperparameter Tuning**, karena ada kemungkinan model mengalami **overfitting**.

Model Scores

**Logistic Regression: 0.7804878048780488,
KNN: 0.7317073170731707,
Random Forest: 0.9853658536585366**

Ini berarti:

- Logistic Regression mencapai akurasi sekitar 78.05%
- K-Nearest Neighbors (KNN) mencapai akurasi sekitar 73.17%
- Random Forest mencapai akurasi tertinggi yaitu sekitar 98.36%

Ringkasan Singkat

| Algoritma | Sifat | Kelebihan | Kekurangan |
|---------------------|----------------|--|---|
| Logistic Regression | Linier | Cepat, sederhana, interpretatif | Tidak cocok untuk data non-linier |
| K-Nearest Neighbors | Nom-parametrik | Mudah dipahami, fleksibel | Lambat di prediksi, sensitif terhadap scaling |
| Random Forest | Ensambel | Akurat, stabil, tangguh terhadap outlier | Kurang interpretatif, relatif lambat |

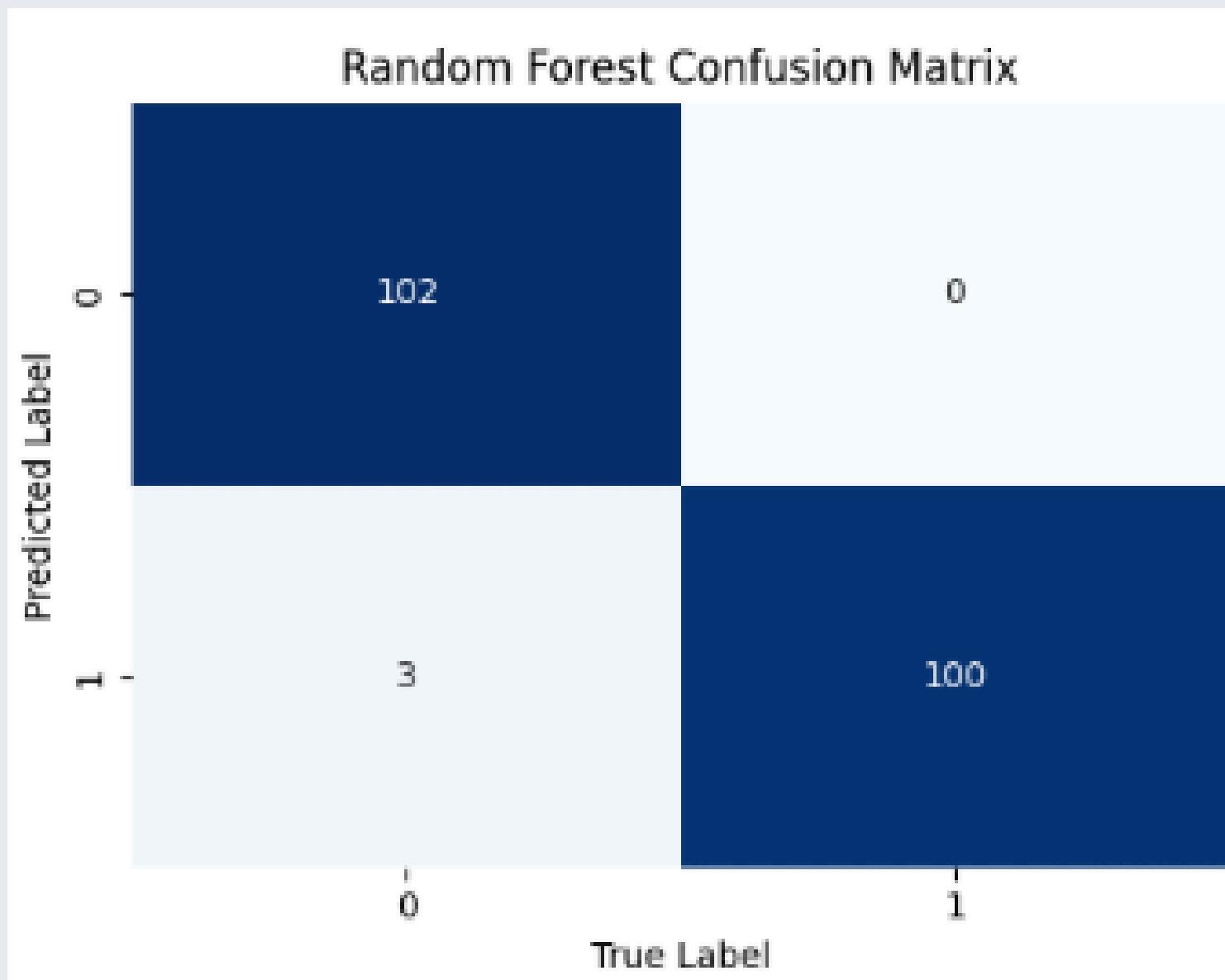
Confusion Matrix

Dalam permasalahan klasifikasi, terdapat empat jenis prediksi:

- **True Positive (TP)** – Model dengan benar memprediksi kelas positif saat kelas aktual juga positif.
- **True Negative (TN)** – Model dengan benar memprediksi kelas negatif saat kelas aktual juga negatif.
- **False Positive (FP)** – Model secara salah memprediksi kelas positif saat kelas aktualnya negatif (dikenal juga sebagai error Tipe I).
- **False Negative (FN)** – Model secara salah memprediksi kelas negatif saat kelas aktualnya positif (dikenal juga sebagai error Tipe II).

- 1 -> $P = T, A = T \rightarrow TP$
- 2 -> $P = T, A = F \rightarrow TN$
- 3 -> $P = F, A = F \rightarrow FP$
- 4 -> $P = F, A = T \rightarrow FN$

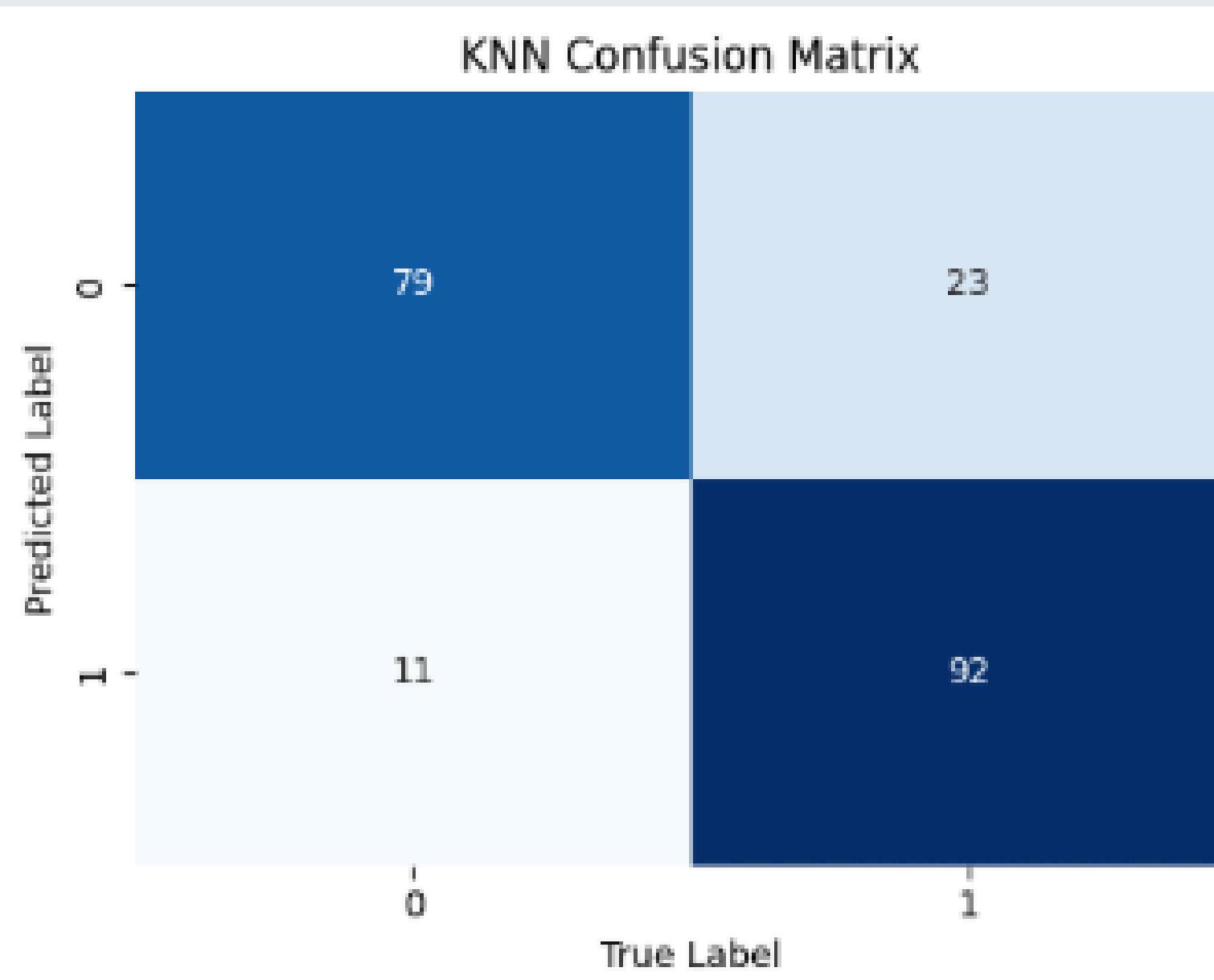
Confusion Matrix - Random Forest



- True Negative (TN): $102 \rightarrow$ Prediksi = 0, dan memang benar (label asli = 0)
- False Positive (FP): $0 \rightarrow$ Prediksi = 1, tapi seharusnya 0
- False Negative (FN): $3 \rightarrow$ Prediksi = 0, tapi seharusnya 1
- True Positive (TP): $100 \rightarrow$ Prediksi = 1, dan memang benar (label asli = 1)

Model ini sangat baik: tidak ada FP sama sekali, hanya 3 FN.

Confusion Matrix - KNN

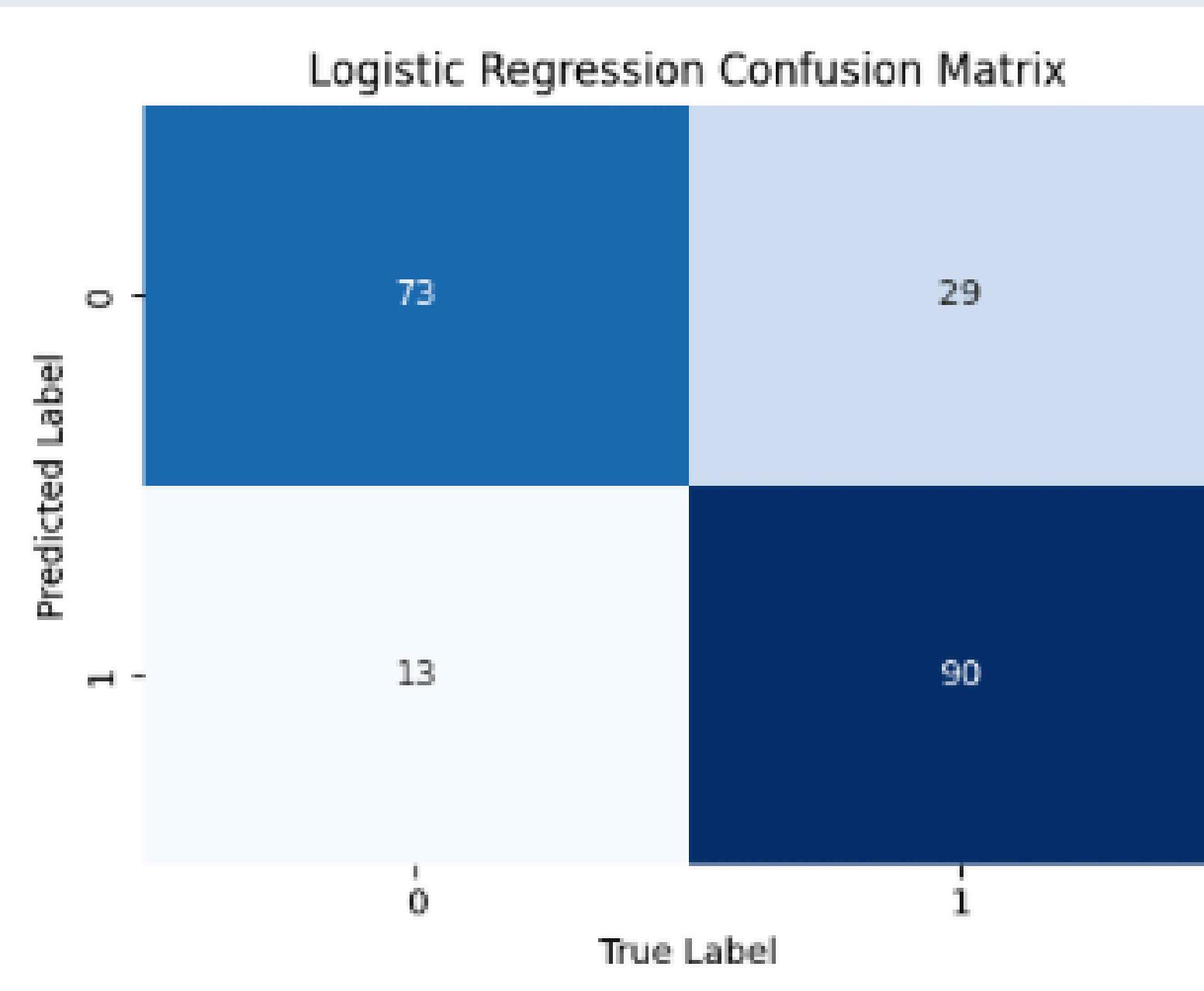


- True Negative (TN): 79 → Benar prediksi negatif
- False Positive (FP): 23 → Salah prediksi positif
- False Negative (FN): 11 → Salah prediksi negatif
- True Positive (TP): 92 → Benar prediksi positif

Performa cukup baik, namun ada:

- 23 data negatif yang salah dikira positif (FP),
- dan 11 data positif yang salah dikira negatif (FN).

Confusion Matrix - Logistic Regression



- True Negative (TN) = 73 → Model memprediksi 0, dan benar (label asli = 0)
- False Positive (FP) = 29 → Model memprediksi 1, tapi seharusnya 0
- False Negative (FN) = 13 → Model memprediksi 0, tapi seharusnya 1
- True Positive (TP) = 90 → Model memprediksi 1, dan benar (label asli = 1)

Evaluation Metrics

Logistic Regression Evaluation

- Accuracy: 0.7951
- Precision: 0.7563
- Recall: 0.8738
- F1 Score: 0.8108
- Matthews Correlation Coefficient (MCC): 0.5973

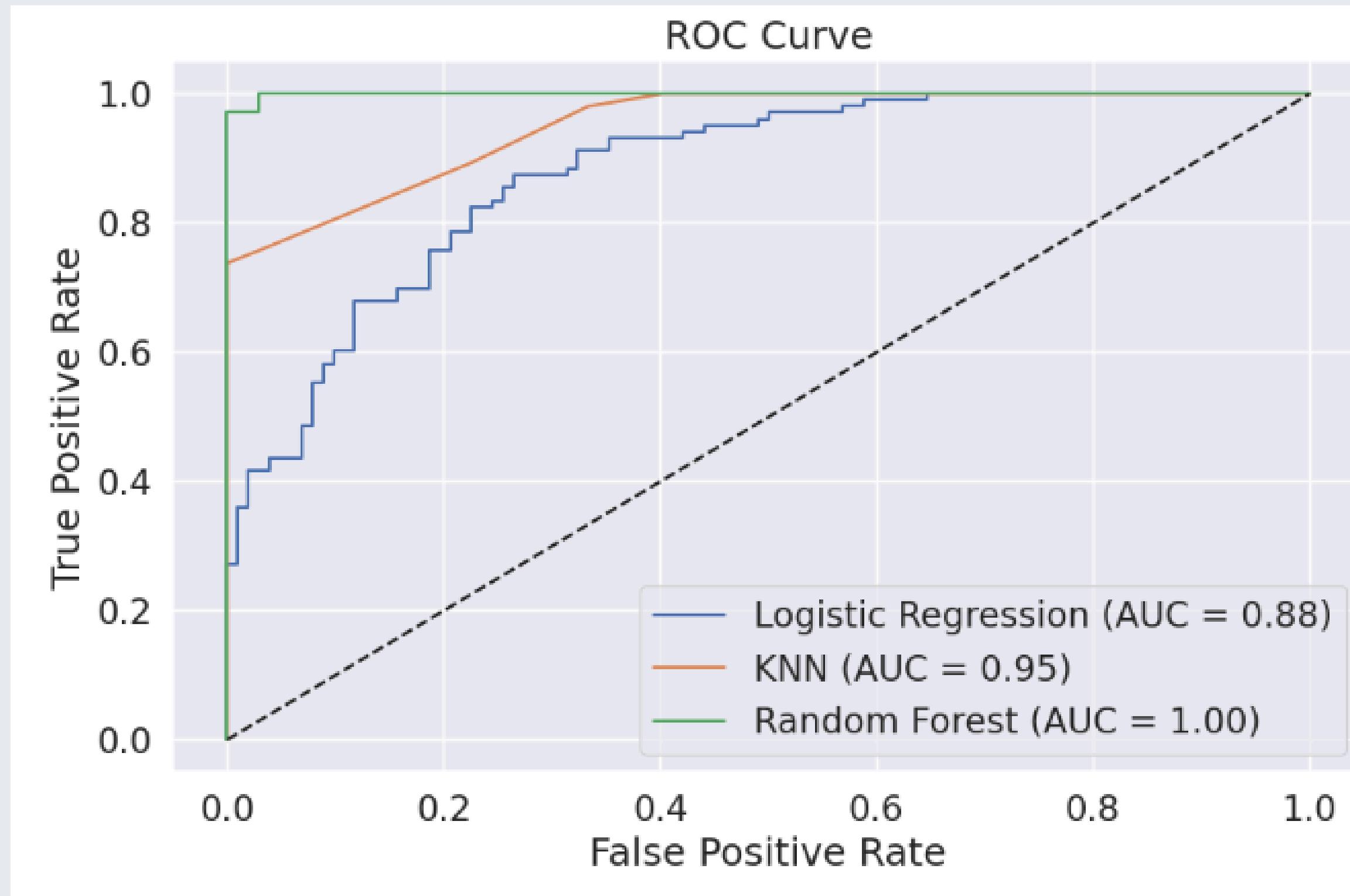
Random Forest Evaluation Metrics

- Accuracy: 0.9854
- Precision: 1.0000
- Recall: 0.9709
- F1 Score: 0.9852
- Matthews Correlation Coefficient (MCC): 0.9712

KNN Evaluation Metrics

- Accuracy: 0.8341
- Precision: 0.8000
- Recall: 0.8932
- F1 Score: 0.8440
- Matthews Correlation Coefficient (MCC): 0.6727

ROC Curve



ROC Curve

Logistic Regression (AUC = 0.88):

- Kurva Logistic Regression menunjukkan kinerja yang cukup baik dengan AUC 0.88, yang berarti model ini memiliki kemampuan diskriminatif yang baik.

KNN (AUC = 0.95):

- Kurva KNN menunjukkan kinerja yang sangat baik dengan AUC 0.95. Ini menandakan bahwa model KNN memiliki kemampuan yang sangat kuat dalam membedakan kelas positif dan negatif. Kurva ini berada cukup dekat dengan sudut kiri atas.

Random Forest (AUC = 1.00):

- Kurva Random Forest memiliki AUC sempurna sebesar 1.00. Ini menunjukkan bahwa model Random Forest mampu membedakan dengan sempurna antara kelas positif dan negatif.
- Namun penting untuk dicatat bahwa AUC 1.00 pada data nyata seringkali merupakan tanda overfitting, di mana model telah terlalu "menghafal" data pelatihan dan mungkin tidak akan berkinerja sebaik ini pada data baru yang belum pernah dilihat.
- Perlu dilakukan validasi silang (cross-validation) atau pengujian pada data uji yang independen untuk memastikan kinerja model yang sesungguhnya.

Kesimpulan Sementara

1. Random Forest mengungguli dua model lainnya secara signifikan di semua metrik.
 - F1 Score dan MCC sangat tinggi, menandakan performa stabil bahkan untuk data yang imbalance.
 - Precision = 1.0 artinya tidak ada false positive sama sekali.
2. KNN performanya lebih baik dari Logistic Regression, namun kalah jauh dibanding Random Forest.
 - KNN cocok ketika data cukup bersih dan tidak terlalu besar, tapi tetap butuh scaling.
3. Logistic Regression paling sederhana namun paling lemah performanya.
 - Masih cukup baik dalam recall, artinya model ini jarang meloloskan pasien yang sakit (FN).

To avoid overfitting, we will apply

Hyperparameter Tuning with RandomizedSearchCV

Catatan: Berdasarkan perbandingan model pada bagian awal, kita menemukan bahwa model **Random Forest** memiliki **akurasi tertinggi** (1.0). Oleh karena itu, kita akan melanjutkan dengan penyetelan hyperparameter menggunakan RandomizedSearchCV untuk model ini.

Hyperparameter Tuning

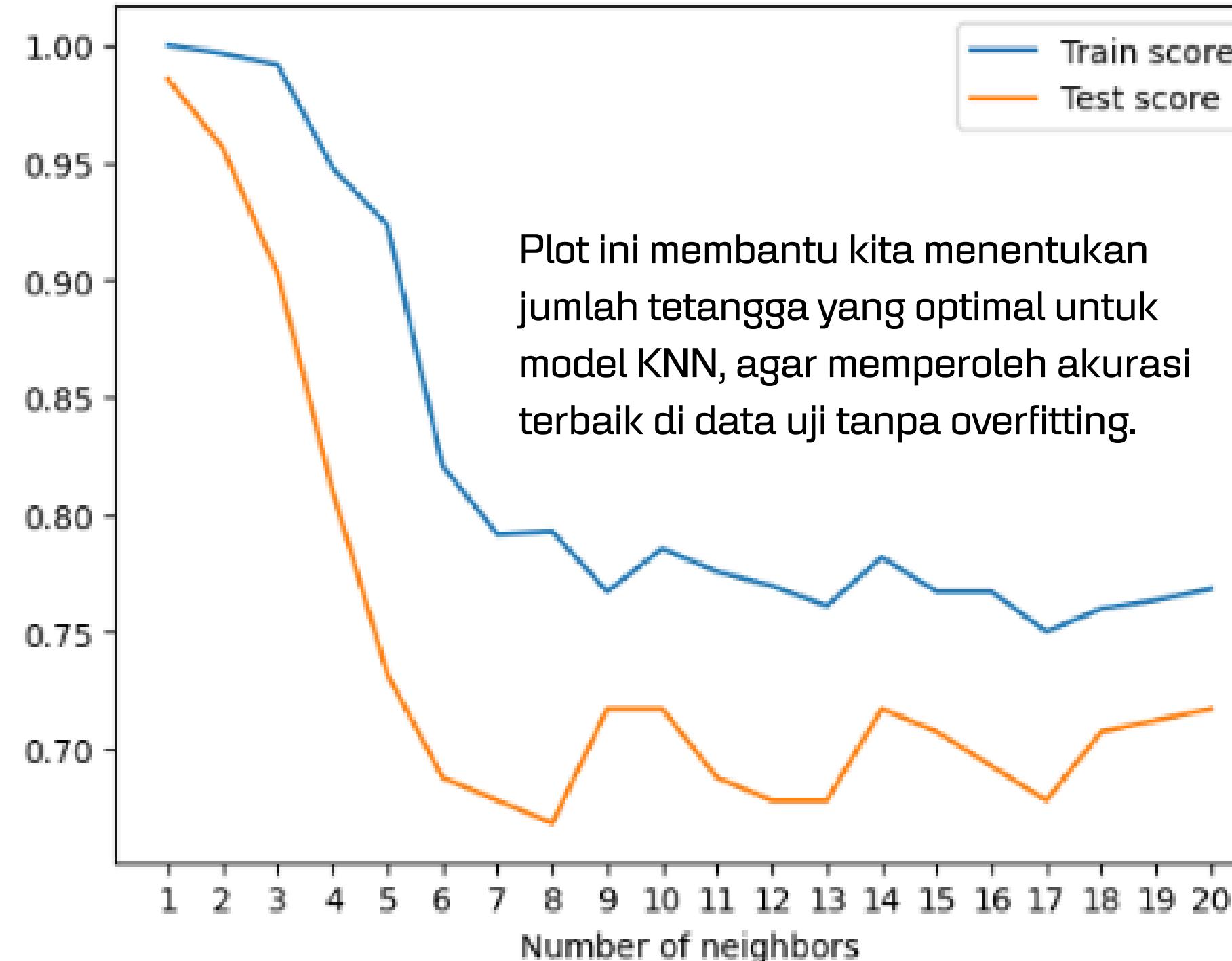
Hyperparameter tuning adalah proses untuk mencari kombinasi parameter terbaik yang dapat meningkatkan performa model machine learning. Berbeda dengan parameters yang dipelajari oleh model saat training (seperti bobot pada regresi), hyperparameters adalah nilai yang ditentukan sebelum training, misalnya jumlah pohon dalam Random Forest atau nilai C dalam Logistic Regression.

RandomizedSearchCV adalah metode pencarian hyperparameter secara acak dari kombinasi yang tersedia, kemudian mengevaluasinya menggunakan cross-validation. Ini lebih efisien dibanding GridSearchCV karena tidak mencoba semua kombinasi secara eksak, melainkan sampling acak terbatas.



Additional - KNN

Maximum KNNN score on the test data: 98.54%



Plot ini membantu kita menentukan jumlah tetangga yang optimal untuk model KNN, agar memperoleh akurasi terbaik di data uji tanpa overfitting.

Saat melakukan plotting dalam penyetelan hyperparameter, ditemukan bahwa skor KNN mencapai nilai maksimum sebesar **98.54%**, dan grafik menunjukkan adanya kemungkinan mencapai akurasi **1.0 (100%)**.

Selain itu, saya juga menemukan bahwa jika jumlah tetangga (neighbors) kurang dari 5, model mencapai akurasi yang sangat tinggi.

Pada model KNN, jika kita mengatur **n = 1**, artinya model hanya mempertimbangkan **satu titik data terdekat**, sehingga membuat model sangat **sensitif terhadap variasi data**.

Logistic Regression

```
→ Fitting 5 folds for each of 20 candidates, totalling 100 fits
    ▶ RandomizedSearchCV
        ▶ best_estimator_: LogisticRegression
            LogisticRegression(C=1.623776739188721, solver='liblinear')
                ▶ LogisticRegression
                    LogisticRegression(C=1.623776739188721, solver='liblinear')

[38] rs_log_reg.best_params_
→ {'solver': 'liblinear', 'C': 1.623776739188721}

[39] rs_log_reg.score(X_test, y_test)
→ 0.7853658536585366
```

Random Forest

```
→ Fitting 5 folds for each of 20 candidates, totalling 100 fits
    ▶ RandomizedSearchCV
        ▶ best_estimator_: RandomForestClassifier
            RandomForestClassifier(min_samples_split=14, n_estimators=510)
                ▶ RandomForestClassifier
                    RandomForestClassifier(min_samples_split=14, n_estimators=510)

[41] # Find the best hyperparameters
    rs_rf.best_params_
→ {'n_estimators': 510,
   'min_samples_split': 14,
   'min_samples_leaf': 1,
   'max_depth': None}

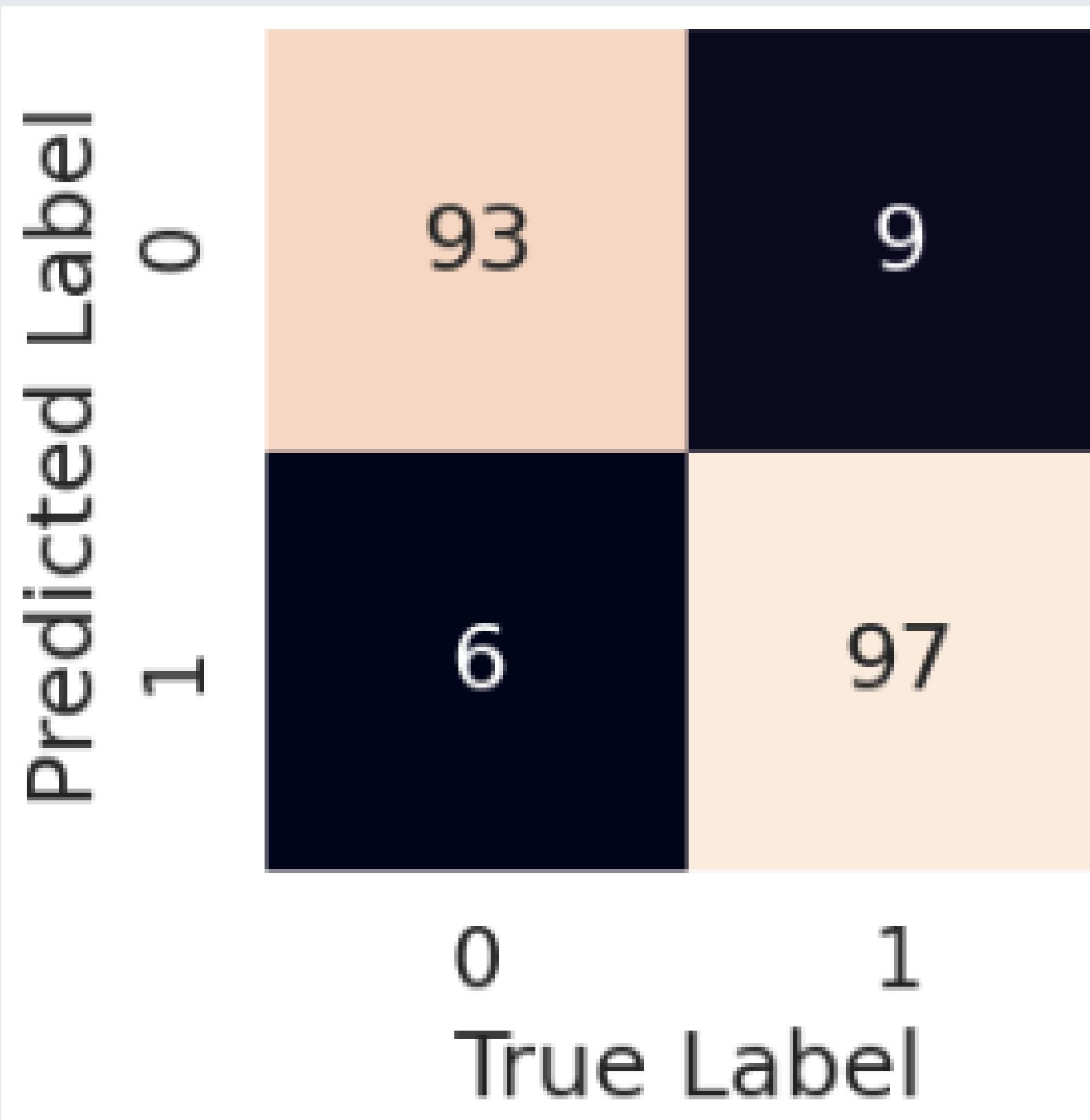
[42] rs_rf.score(X_test, y_test) # score untuk test
→ 0.926829268292683
```

Logistic Regression

- Tujuan: Menemukan kombinasi nilai C (regularisasi) dan solver terbaik.
- Hasil:
 - solver: 'liblinear' → solver yang cocok untuk dataset kecil dan binary classification.
 - C: 1.623... → mengatur seberapa kuat regularisasi diterapkan (semakin besar, semakin lemah regularisasi).
- Akurasi akhir: ~78.54% pada data uji.

Random Forest (Kanan)

- Tujuan: Mencari kombinasi terbaik dari beberapa parameter seperti:
 - n_estimators: jumlah pohon (dalam contoh: 510).
 - min_samples_split: jumlah minimal sampel untuk membagi node (dalam contoh: 14).
 - min_samples_leaf: jumlah minimal sampel di daun pohon (dalam contoh: 1).
- Hasil:
 - Model yang ditemukan lebih kompleks namun juga lebih stabil dan akurat.
- Akurasi akhir: ~96.83% pada data uji.



Confusion Matrix

Setelah Hyperparameter Tuning

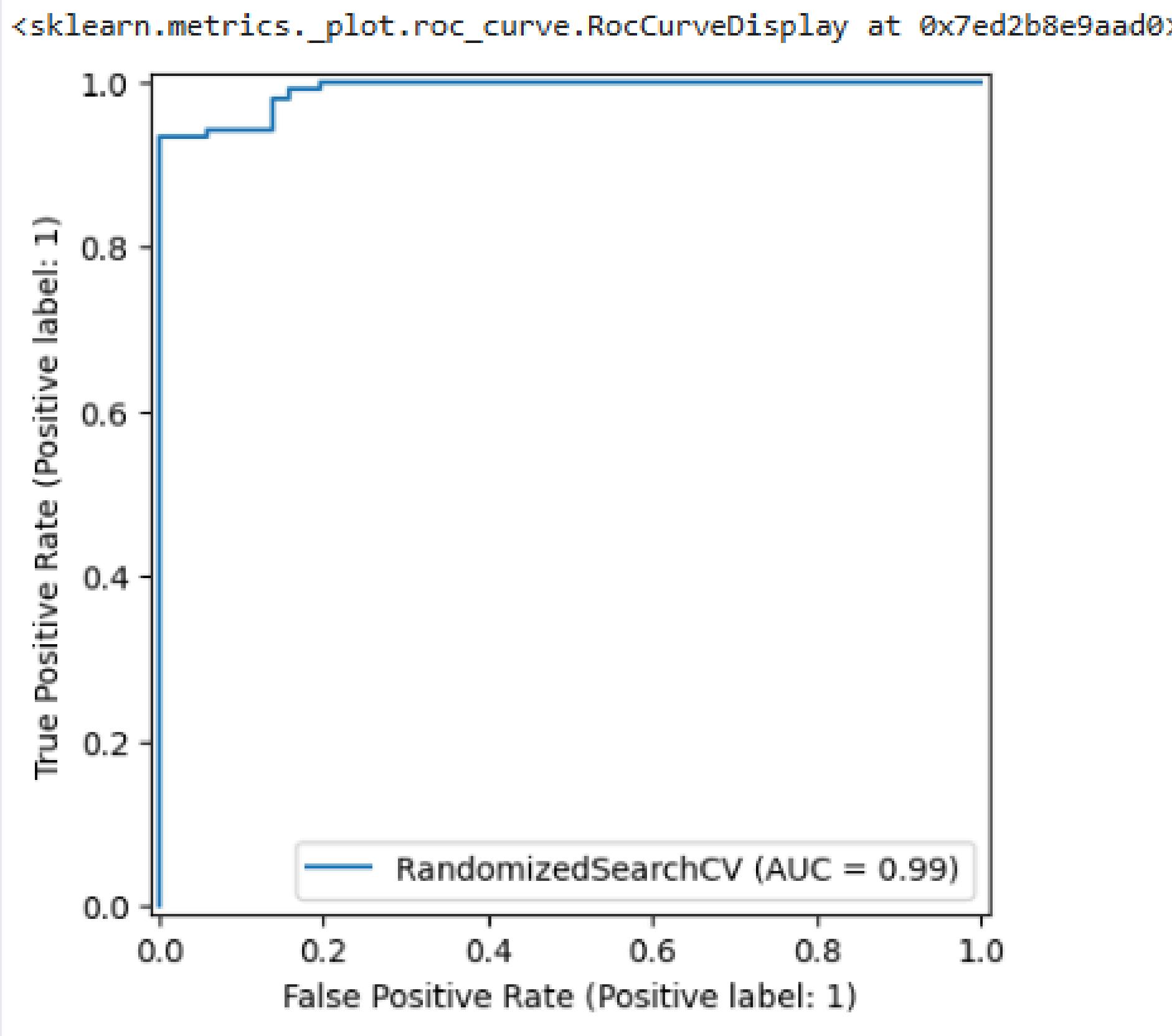
Dalam confusion matrix visualization,
terdapat **15** data ($6 + 9$) yang
diklasifikasikan secara salah.

ROC Curve Display

Di sini, kita dapat melihat bahwa

AUC = 0.99

Jika nilai AUC di atas 0.90, itu menunjukkan bahwa modelnya bagus. Luas area di bawah kurva (error area) sangat kecil.



Evaluasi Model

| Model | Accuracy | Precision | Recall | Mean F1 Score |
|---------------------|----------|-----------|--------|---------------|
| Logistic Regression | 0.8478 | 0.8159 | 0.8116 | 0.808 |
| KNN | 0.7938 | 0.7456 | 0.7187 | 0.7317 |
| Random Forest | 0.9661 | 0.9602 | 0.9592 | 0.9661 |

- Random Forest unggul di semua metrik: akurasi, presisi, recall, dan F1 score.
- Logistic Regression cukup baik dan stabil, tapi sedikit kalah performa.
- KNN memiliki akurasi dan performa yang lebih rendah dibanding dua model lainnya.

Standar Deviasi Metrik (Stability Check)

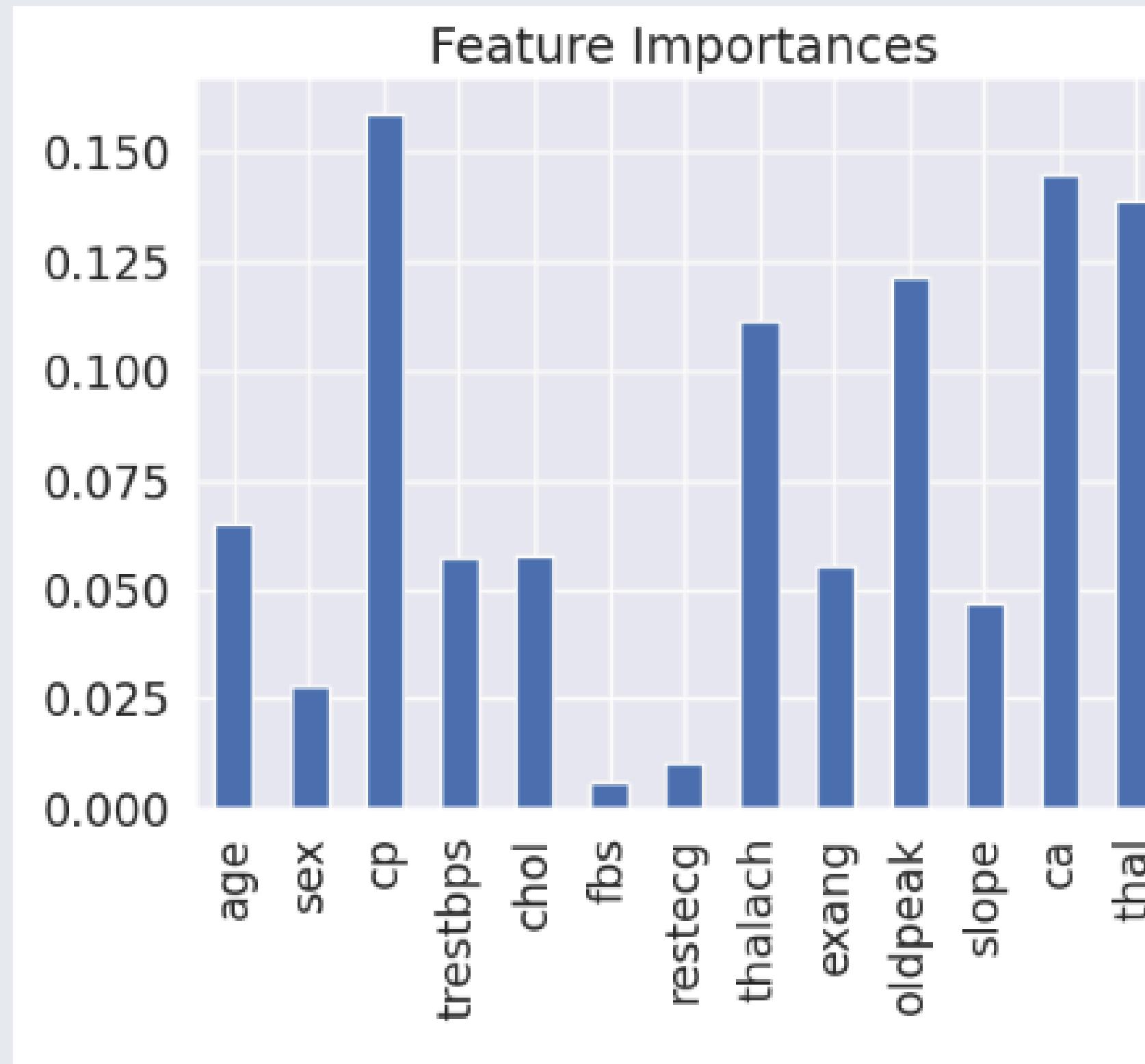
| Model | Accuracy Std | Precision Std | Recall Std | F1 Std |
|---------------------|--------------|---------------|------------|--------|
| Logistic Regression | 0.0143 | 0.0254 | 0.0261 | 0.0116 |
| KNN | 0.027 | 0.0252 | 0.0351 | 0.028 |
| Random Forest | 0.0078 | 0.0008 | 0.0152 | 0.0078 |

- Random Forest tidak hanya akurat, tapi juga paling stabil (standar deviasi paling kecil).
- KNN paling tidak stabil, artinya performanya sangat bergantung pada data.

Feature Importance



Feature Importance



Fitur Paling Berpengaruh (kontribusi terbesar):

- **cp (Chest Pain Type): Ini adalah fitur paling berpengaruh terhadap prediksi.**
- ca (Jumlah pembuluh darah utama yang terlihat)
- thal (Kondisi thalassemia)
- exang (Angina akibat olahraga)
- thalach (Denyut jantung maksimum)
- slope (Kemiringan segmen ST)

Fitur Kurang Penting:

- **fbs (Fasting blood sugar): Hampir tidak berpengaruh**
- restecg (Hasil elektrokardiogram saat istirahat)
- sex (Jenis kelamin)

MODELING

Unupervised Learning

K-Means Clustering & DBSCAN

K-Means Clustering

✓ KMeans Clustering

```
[ ] from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import numpy as np

X_unsup = df.drop("target", axis=1)

X_scaled_unsup = StandardScaler().fit_transform(X_unsup)

kmeans = KMeans(n_clusters=2, random_state=42, n_init='auto')
kmeans_labels = kmeans.fit_predict(X_scaled_unsup)

kmeans_silhouette = silhouette_score(X_scaled_unsup, kmeans_labels)
print("K-Means Silhouette Score:", round(kmeans_silhouette, 4))
```

→ K-Means Silhouette Score: 0.1687

K-Means Clustering

- K-Means bekerja dengan mengelompokkan data ke dalam K cluster berdasarkan jarak terdekat ke centroid (titik pusat).
- Algoritma:
 1. Pilih K titik centroid awal secara acak.
 2. Kelompokkan data ke centroid terdekat (menggunakan jarak Euclidean).
 3. Hitung ulang posisi centroid sebagai rata-rata dari semua titik dalam cluster.
 4. Ulangi langkah 2-3 sampai posisi centroid tidak berubah banyak.
- Cocok untuk data yang bentuknya bulat/berkelompok jelas, tidak terlalu noisy.

DBSCAN

✓ DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

```
[ ] from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=1.2, min_samples=5)
dbscan_labels = dbscan.fit_predict(X_scaled_unsup)

n_clusters_dbscan = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
print("DBSCAN Estimated Clusters:", n_clusters_dbscan)

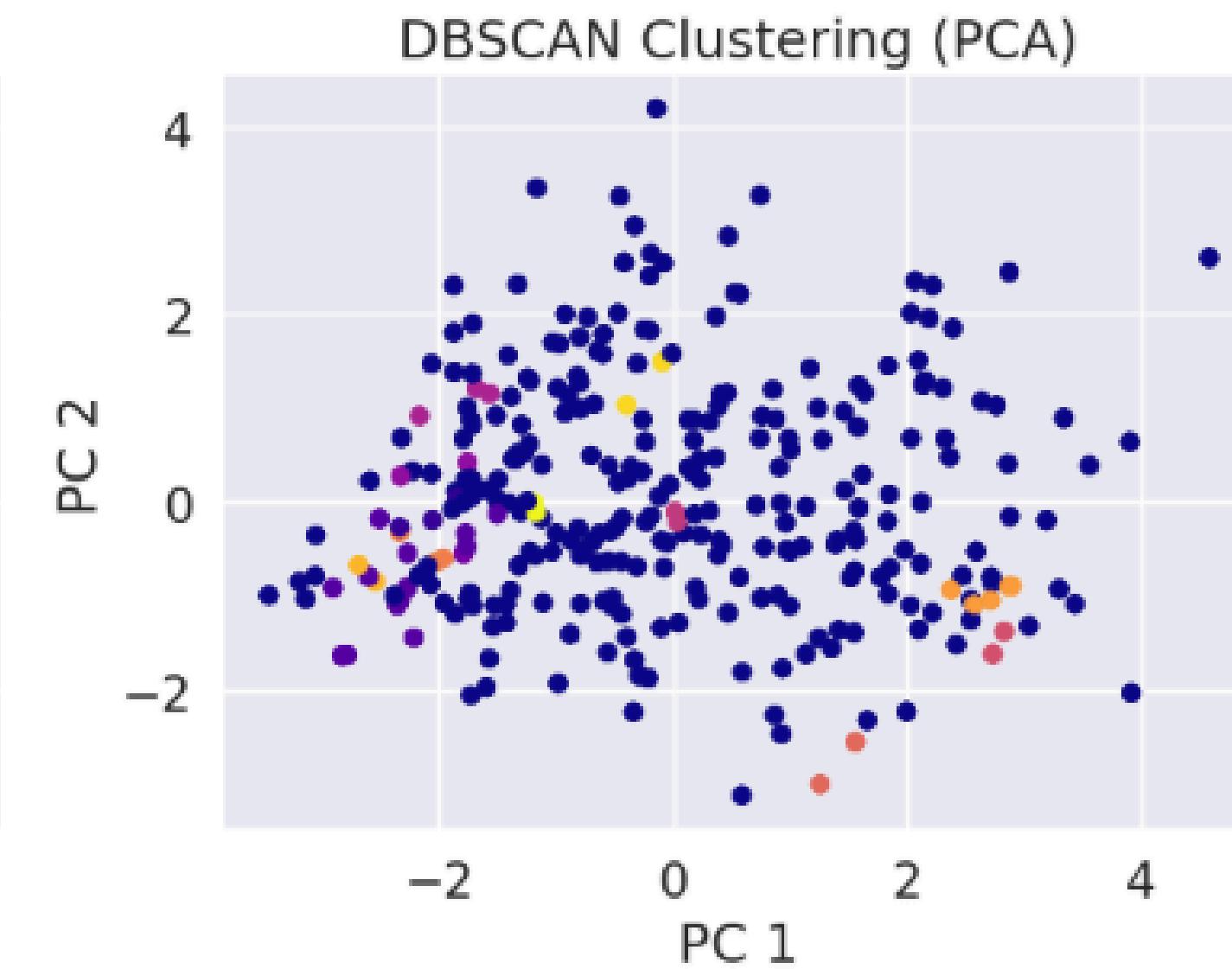
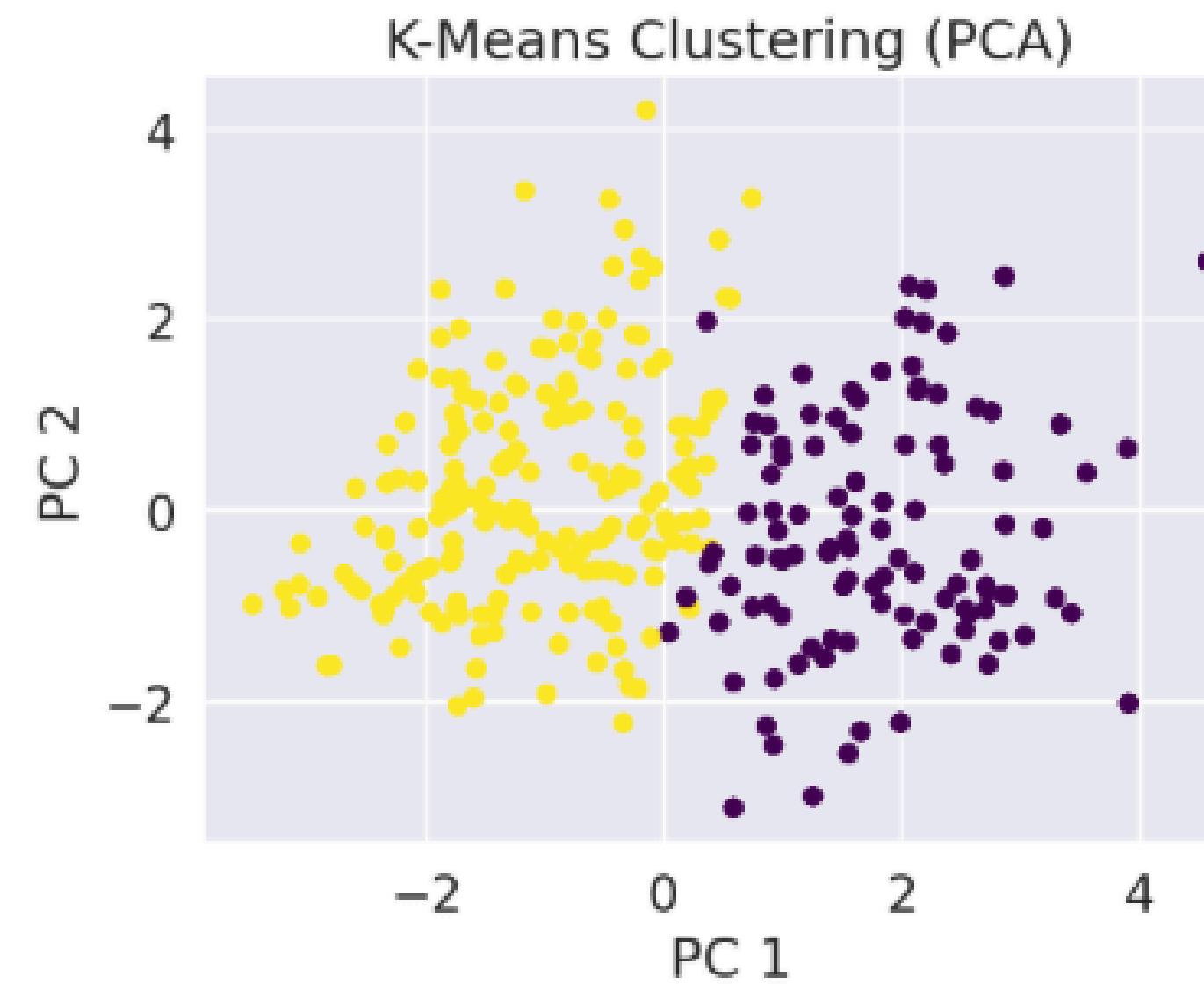
if n_clusters_dbscan > 1:
    dbscan_silhouette = silhouette_score(X_scaled_unsup, dbscan_labels)
    print("DBSCAN Silhouette Score:", round(dbscan_silhouette, 4))
else:
    print("DBSCAN tidak membentuk cluster yang cukup untuk menghitung Silhouette Score.")
```

→ DBSCAN Estimated Clusters: 13
DBSCAN Silhouette Score: -0.211

DBSCAN

- DBSCAN adalah algoritma berbasis kerapatan, bagus untuk data yang memiliki bentuk cluster tidak beraturan.
- Algoritma:
 - 1.Untuk setiap titik, cari tetangga dalam radius eps .
 - 2.Jika jumlah tetangga $\geq \text{min_samples}$, titik tersebut adalah core point.
 - 3.Cluster dibentuk dari core point dan semua titik yang dapat dijangkau secara kerapatan.
 - 4.Titik yang tidak dapat dijangkau dari core point manapun dianggap sebagai noise (-1).
- Cocok untuk data dengan cluster tak beraturan, banyak outlier/noise.

Visualisasi dengan PCA



Visualisasi dengan PCA

Hasil Clustering:

K-Means Clustering (PCA)

- Data berhasil terbagi menjadi 2 klaster utama (terlihat sebagai dua warna: kuning dan ungu).
- Klaster cukup terpisah dengan jelas secara visual, menandakan bahwa K-Means mampu mengelompokkan data dengan cukup baik berdasarkan informasi dua komponen utama PCA.
- Cocok digunakan jika data memiliki pola yang relatif simetris atau bentuk bundar (spherical clusters).

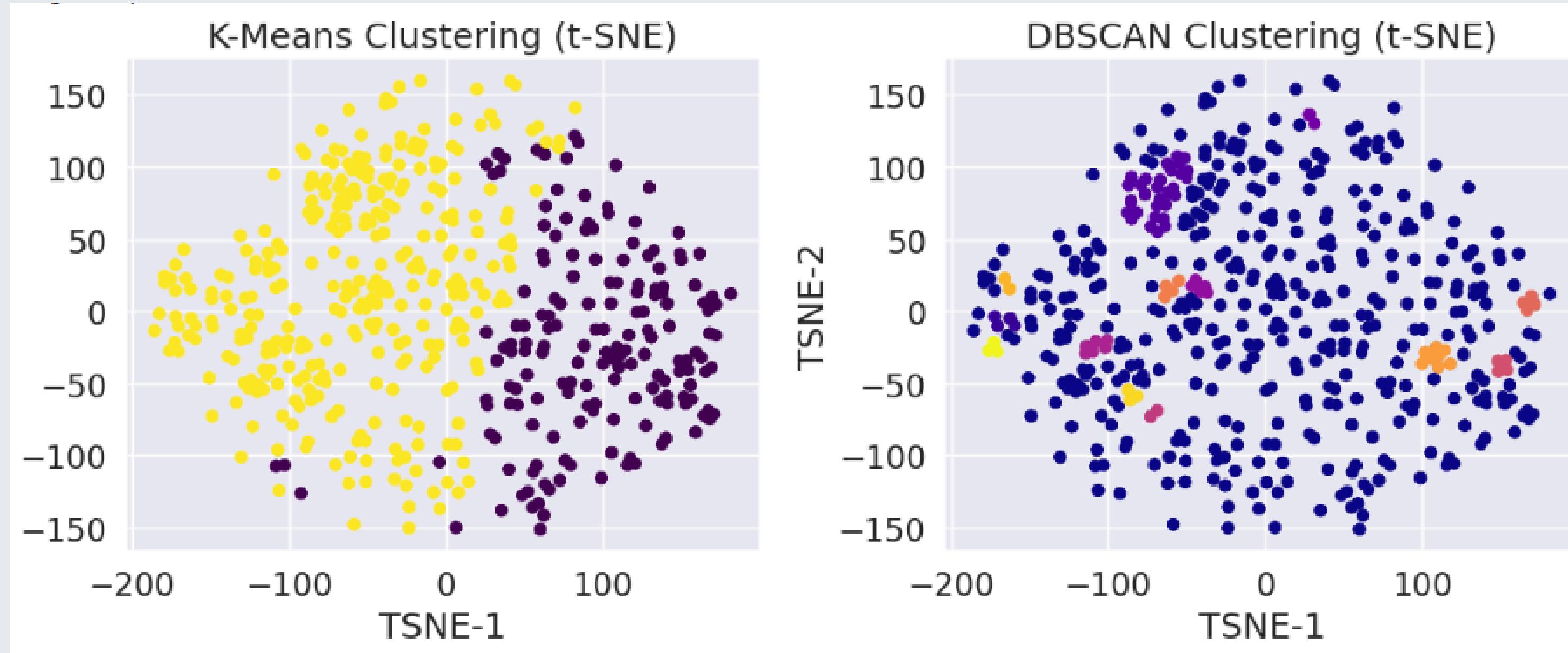
DBSCAN Clustering (PCA)

- Hasil clustering dari DBSCAN menunjukkan klaster yang lebih tersebar dan tidak beraturan.
- Ada beberapa titik dengan warna berbeda yang kemungkinan merupakan noise (outlier) atau titik yang tidak termasuk dalam klaster manapun.
- DBSCAN lebih cocok untuk data dengan bentuk klaster yang tidak beraturan dan dapat mengidentifikasi outlier, namun sensitif terhadap parameter eps dan min_samples.

Kesimpulan Visualisasi PCA

- K-Means memberikan klaster yang lebih bersih dan terdefinisi secara visual dibanding DBSCAN.
- DBSCAN memberikan fleksibilitas dalam mendeteksi bentuk klaster yang tidak biasa dan noise, tapi hasilnya lebih kompleks untuk diinterpretasi.
- PCA membantu menyederhanakan data dan memberikan gambaran umum tentang bagaimana algoritma clustering bekerja terhadap data.

Visualisasi dengan t-SNE



Visualisasi dengan t-SNE

1. K-Means Clustering (t-SNE)

- Terbentuk dua kelompok dominan yang relatif simetris.
- Klaster cukup jelas terlihat dan terpisah secara spasial, walau ada sedikit tumpang tindih di batas antar cluster.
- Warna kuning dan ungu menunjukkan label klaster hasil K-Means.
- t-SNE menunjukkan bahwa K-Means berhasil mengelompokkan sebagian besar data dengan cukup baik, tetapi masih ada titik-titik yang mungkin tidak konsisten.

2. DBSCAN Clustering (t-SNE)

- DBSCAN menghasilkan klaster yang lebih kompleks, dengan banyak warna yang tersebar.
- Beberapa titik diwarnai berbeda (merah/oranye) yang menunjukkan noise atau outlier, hal yang memang menjadi kekuatan DBSCAN.
- Secara umum, hasil clustering kurang terstruktur dibanding K-Means.
- t-SNE menunjukkan bahwa meskipun DBSCAN bisa menangani bentuk klaster yang tidak teratur, dalam data ini ia tidak membentuk klaster yang konsisten secara visual.

Kesimpulan Visualisasi t-SNE

- K-Means: Klaster terlihat lebih terdefinisi, lebih cocok untuk struktur data ini.
- DBSCAN: Menghasilkan lebih banyak noise, kurang rapi, dan klaster tersebar.
- t-SNE: Memberikan visualisasi spasial yang lebih informatif dibanding PCA.

Evaluasi dengan metrik clustering (Silhouette & Rand Index)

| Metrik | K-Means | DBSCAN |
|---------------------------|------------------------|---------------------------|
| Silhouette Score | 0.1687 | -0.211 |
| Adjusted Rand Index (ARI) | 0.376 | 0.0068 |
| Jumlah Cluster | 2 (sesuai n_clusters) | 13 (otomatis dari DBSCAN) |

Interpretasi Hasil

K-Means

- Silhouette Score = 0.1687
 - Angka ini menunjukkan struktur cluster yang lemah. Artinya data tidak terkelompok sangat jelas, tapi masih lebih baik daripada DBSCAN.
- Adjusted Rand Index = 0.376
 - Ini adalah korelasi antara hasil clustering dengan label asli. Nilai ini cukup lumayan untuk clustering tanpa supervisi, menunjukkan bahwa sebagian besar label berhasil dipetakan ke cluster yang sesuai.

DBSCAN

- Silhouette Score = -0.211
 - Nilai negatif menunjukkan bahwa banyak titik lebih dekat ke cluster lain dibanding ke cluster-nya sendiri. Ini menandakan struktur cluster yang buruk atau banyak outlier/noise.
- Adjusted Rand Index = 0.0068
 - Sangat rendah. Ini berarti hasil clustering DBSCAN hampir tidak berkorelasi sama sekali dengan label asli. Hal ini bisa disebabkan oleh:
 - Parameter `eps` dan `min_samples` yang tidak optimal.
 - DBSCAN sangat sensitif terhadap parameter dan skala data.

Kesimpulan Perbandingan Sementara Pada Unsupervised Learning

K-Means:

- Parameter utama yang berpengaruh: n_clusters
- Performa pada data yang tidak berbentuk bulat & bergelombang: Berhasil baik
- Performa pada data dengan noise: Berhasil baik (ARI & silhouette)
- Jumlah cluster: Ditentukan secara manual

DBSCAN:

- Parameter utama yang berpengaruh: eps, min_samples
- Performa pada data yang tidak berbentuk bulat & bergelombang: Tidak beraturan, ada noise
- Performa pada data dengan noise: Kurang baik, perlu tuning eps dan min_samples
- Jumlah cluster: Dihitung secara otomatis (hasil: 13)

Tambahan:

- DBSCAN sangat sensitif terhadap parameter dan skala data.
- Pada dataset 'heart' ini, K-Means lebih efektif dibandingkan DBSCAN.
- Jika ingin tetap menggunakan DBSCAN, perlu dilakukan:
 - Tuning parameter eps dan min_samples menggunakan Grid Search atau evaluasi visual.
 - Coba juga reduksi dimensi (PCA atau t-SNE) untuk membantu clustering DBSCAN.

KESIMPULAN

▼ Perbandingan Supervised vs Unsupervised Learning

| Aspek | Supervised Learning | Unsupervised Learning |
|--------------------|--|---|
| Label Data | Membutuhkan label (target sudah diketahui) | Tidak membutuhkan label |
| Tujuan Utama | Prediksi dan klasifikasi | Eksplorasi struktur dan pola data |
| Akurasi Model | Umumnya lebih tinggi bila label tersedia | Umumnya lebih rendah karena tidak ada label |
| Evaluasi | Jelas: accuracy, precision, recall, F1, MCC | Lebih sulit: clustering score, silhoutte |
| Kapan Lebih Unggul | Saat data berlabel tersedia dan target penting | Saat data belum dilabeli atau eksplorasi awal |

Kapan Supervised Learning Lebih Cocok?

- Saat kita mempunyai label/class seperti "punya penyakit jantung" atau tidak.
- Ketika tujuan utama adalah klasifikasi/prediksi.
- Cocok untuk aplikasi nyata seperti diagnosa penyakit, karena hasilnya bisa diinterpretasikan langsung (misalnya, pasien A kemungkinan besar sakit).
- Dapat mengevaluasi model secara kuantitatif (akurasi, recall, dll).

★ Contoh



Dataset heart disease, karena kita punya kolom target sebagai label (1 = sakit, 0 = tidak).

Kapan Unsupervised Learning Lebih Cocok?

- Saat label belum tersedia, misalnya data mentah dari survei atau sensor.
- Untuk eksplorasi pola tersembunyi, seperti segmentasi pasien atau deteksi anomali.
- Ketika kita ingin mengetahui kelompok pasien dengan pola serupa meskipun belum tahu diagnosis pastinya.



Contoh



Awal pengumpulan data pasien baru yang belum ada diagnosis medis.

Refleksi Terhadap Dataset Heart Disease

- Dataset ini memiliki label target (0 = tidak sakit, 1 = sakit) → **Supervised Learning jelas lebih cocok.**
- Kita bisa melakukan klasifikasi langsung dan mengevaluasi performanya.
- Random Forest menghasilkan **skor sangat tinggi** (Accuracy = 0.9854, Precision = 1.0) → menandakan model ini **mampu mendeteksi penyakit dengan sangat baik tanpa false positive.**
- Recall tinggi di Logistic Regression dan KNN juga berguna dalam konteks kesehatan: lebih baik memberikan "alarm palsu" (false positive) daripada gagal mendeteksi pasien sakit (false negative).

- Untuk dataset heart disease, **supervised learning adalah pendekatan terbaik** karena:
 1. Kita punya label diagnosis.
 2. Model bisa dilatih untuk prediksi akurat dan diuji dengan metrik standar.
 3. Model seperti Random Forest terbukti sangat efektif.
- **Unsupervised learning bisa menjadi pelengkap**, misalnya untuk clustering pasien dengan gejala mirip, sebelum diagnosis ditegakkan, atau dalam studi eksploratif.

Project Link



Github

https://github.com/nadhif_royal/HeartDisease-Team8



Dataset

<https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset>

Thankyou

Team 8

Fikri Adyatma (23515020111015)

Nadhif Rif'at Rasendriya (23515020111074)

Reyno Benedict (23515020711048)