Tugas 5: Tugas Mandiri 5 – SVM

Amaya Eshia - 0110224102*

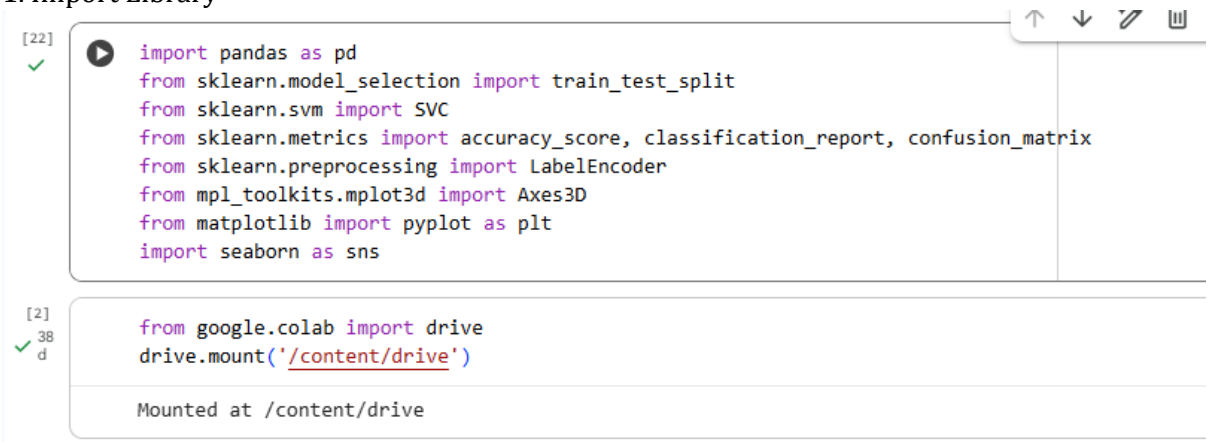
[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: name@institution.edu – **0110224102@student.nurulfikri.ac.id**

Abstract. Penelitian ini bertujuan untuk membangun, mengevaluasi, dan mengoptimalkan model klasifikasi penyakit jantung menggunakan algoritma Support Vector Machine (SVM). Dataset yang digunakan adalah heart.csv, yang berisi data kesehatan pasien termasuk fitur-fitur seperti usia, tekanan darah, kolesterol, dan detak jantung maksimum. Proses diawali dengan **data *preprocessing***, meliputi pengecekan dan penghapusan data duplikat, serta **Standard Scaling** untuk menstandardisasi fitur, yang sangat penting untuk performa optimal SVM. Data kemudian dibagi menjadi set pelatihan dan pengujian. **Model SVM** awal dibangun menggunakan *kernel* Radial Basis Function (RBF). Selanjutnya, dilakukan **Hyperparameter Tuning** menggunakan metode **Grid Search** untuk mencari kombinasi parameter C dan γ terbaik guna mencegah *overfitting* dan meningkatkan generalisasi model. Evaluasi model terbaik dilakukan menggunakan metrik **Akurasi**, **Classification Report**, dan **Confusion Matrix** pada data pengujian. Hasil tuning menunjukkan peningkatan performa dan stabilitas, yang dikonfirmasi melalui **Cross-Validation**, membuktikan bahwa model SVM yang dioptimalkan efektif dalam memprediksi risiko penyakit jantung.

Kata Kunci: Support Vector Machine (SVM), Klasifikasi, Penyakit Jantung, Standard Scaling, Hyperparameter Tuning, Grid Search, Cross-Validation, Machine Learning.

1. Import Library



```
[22]
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import pyplot as plt
import seaborn as sns
```

```
[2]
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Bagian ini mengimpor semua *library* yang diperlukan untuk seluruh proses *machine learning* (analisis data, *preprocessing*, pemodelan **SVM**, dan visualisasi) serta menghubungkan ke Google Drive untuk mengakses dataset.

2. Melihat informasi Umum Dataset

```
[4]
df = pd.read_csv('/content/drive/MyDrive/Praktikum Machine Learning_Amaya Eshia_0110224102_Ai02/Praktikum 6/Data/heart.csv')  # Asumsi path serupa dengan Iris.csv
df
```

	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

1025 rows × 14 columns

```
[8]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Kode ini membaca file dataset heart.csv dan menampilkan 5 baris pertama serta informasi umum struktur data (df.info()).

3. Data Pre-Processing

3.1 Cek Missing Value

```
[28]
✓ 0 d
df.isnull().sum()
```

	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	0
thal	0
target	0

dtype: int64

Memastikan tidak ada data yang hilang (*missing value*) pada setiap kolom dengan menghitung total nilai kosong. Hasil 0 menunjukkan tidak ada data kosong.

3.2 Cek dan Menghapus Data Duplikat

```
[29]
✓ 0 d
df.duplicated().sum()

np.int64(723)
```

```
[30]
✓ 0 d
df = df.drop_duplicates()
df.duplicated().sum()

np.int64(0)
```

Memeriksa dan menghapus baris data yang memiliki nilai duplikat di semua kolom.

4. Data Understanding (EDA)

4.1 Analisis Distribusi Target

```
[31]
✓ 0d
df["target"].value_counts()
```

target	count
1	164
0	138

dtype: int64

```
[32]
✓ 0d
df["target"].value_counts(normalize=True) * 100
```

target	proportion
1	54.304636
0	45.695364

dtype: float64

Menghitung jumlah data untuk setiap kategori di kolom target (0 = Tidak Sakit Jantung, 1 = Sakit Jantung) dan persentasenya.

4.2 Pemilihan Fitur X dan Target Y

```
[34]
✓ 0d
# Feature Selection
X = df[['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]  # Pilih 5 fitur contoh
# Atau gunakan semua fitur kecuali target:
# X = df.drop('target', axis=1)

y = df['target']  # Kolom target

X = df.drop('target', axis=1)  # Fitur
y = df['target']  # Target (0 atau 1 untuk penyakit jantung)

# Split data menjadi train dan test (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fitur (X): Variabel input yang akan digunakan model untuk belajar dan memprediksi. Kode ini menunjukkan dua opsi:

1. Memilih hanya 5 fitur spesifik ('age', 'trestbps', 'chol', 'thalach', 'oldpeak') sebagai contoh.
2. Menggunakan semua kolom kecuali kolom target (df.drop('target', axis=1)). Opsi kedua ini yang akhirnya digunakan untuk **X**.

Target (y): Variabel output yang ingin diprediksi, yaitu kolom target (klasifikasi: **0** = Tidak Sakit Jantung, **1** = Sakit Jantung).

4.3 Tampilan Data Fitur Training

[19] ✓ 0 d

```
X.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2

Ini adalah langkah verifikasi untuk melihat struktur dan nilai data fitur yang akan masuk ke proses pelatihan.

4.4 Inisialisasi dan Pelatihan Model SVM

[6] ✓ 3 d

```
# Inisialisasi model SVM dengan kernel linear (bisa diganti ke 'rbf', 'poly', dll.)
model = SVC(kernel='linear', random_state=42)

# Train model
model.fit(X_train, y_train)
```

```
SVC
SVC(kernel='linear', random_state=42)
```

SVC: Singkatan dari **Support Vector Classifier**, model yang digunakan untuk tugas klasifikasi.
Kernel='linear': Menentukan tipe *kernel* yang digunakan oleh SVM. *Kernel* 'linear' mencoba menemukan bidang pemisah (hyperplane) yang lurus di ruang fitur. (Catatan: Meskipun model ini diinisialisasi dengan 'linear', bagian selanjutnya dari tugasmandiri6(2).py kemungkinan akan menggunakan *kernel* **'rbf'** setelah proses *scaling*, yang umum untuk model SVM).
random_state=42: Digunakan untuk inisialisasi internal, memastikan hasil model yang konsisten.
model.fit(X_train, y_train): Ini adalah langkah inti di mana model SVM **belajar** pola dari data training (**X_train**) untuk memetakan fitur ke label target (**y_train**).

4.5 Evaluasi Model SVM Awal (Kernel Linear, Tanpa Scalling)

[7]
0d

```
# Prediksi pada data test
y_pred = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print(f'Akuracy: {accuracy * 100:.2f}%')

print('Classification Report:')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Akuracy: 80.49%
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.71      0.78       102
           1       0.76      0.90      0.82       103

    accuracy                           0.80       205
   macro avg       0.82      0.80      0.80       205
weighted avg       0.82      0.80      0.80       205

Confusion Matrix:
[[72 30]
 [10 93]]
```

Kode ini evaluasi awal model SVM (Kernel Linear) yang dilatih dengan data mentah (tanpa Scaling). Model digunakan untuk memprediksi label pada testing, menghitung persentase prediksi yang benar (contoh : output 80.49%). Menampilkan matrix rinci perkelas (Precision, Recall, F1-Score). Serta menghitung matriks yang menunjukkan jumlah prediksi benar dan salah.

4.6 Scaling Data dan Pelatihan Ulang SVM (Kernel RBF)

[24]

```
# Gunakan scaling (penting untuk SVM!)
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = SVC(kernel='rbf', C=1.0, random_state=42)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
```

Langkah *scaling* ini penting karena SVM sensitif terhadap skala fitur, dan *kernel* 'rbf' umumnya bekerja optimal pada data yang telah distandardisasi.6. Encoding Data Kategorikal (mapping Label ke Kode Numerik)

5. Visualisasi

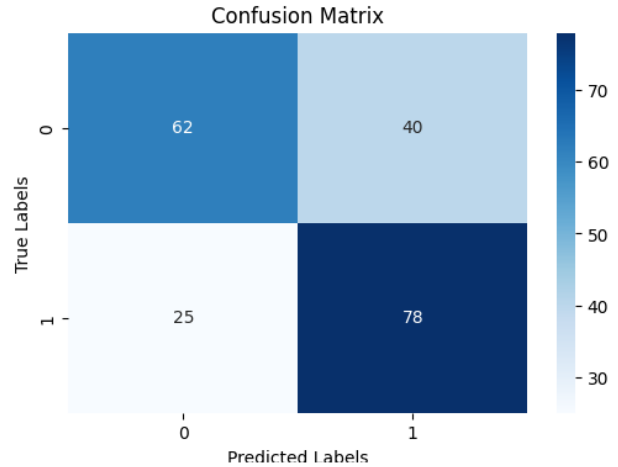
5.1 Confusion Matrix

```
[23]
# Hitung confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Tampilkan dalam bentuk teks
print("Confusion Matrix:")
print(cm)

# Visualisasi dengan seaborn heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

```
Confusion Matrix:
[[62 40]
 [25 78]]
```



Kode ini pertama-tama menghitung Confusion Matrix (cm) yang membandingkan hasil prediksi model SVM (y_pred) dengan nilai target sebenarnya (y_test). Matriks ini kemudian ditampilkan dalam bentuk teks, yang dalam contoh menunjukkan [[62 40], [25 78]]. Langkah selanjutnya adalah visualisasi matriks tersebut menggunakan fungsi sns.heatmap() dari library Seaborn. Visualisasi heatmap dengan skema warna biru (cmap='Blues') dan menampilkan nilai angka di setiap sel (annot=True, fmt='d') mempermudah interpretasi performa model secara visual, di mana sumbu Y mewakili True Labels dan sumbu X mewakili Predicted Labels.

5.2 Scatter Plot

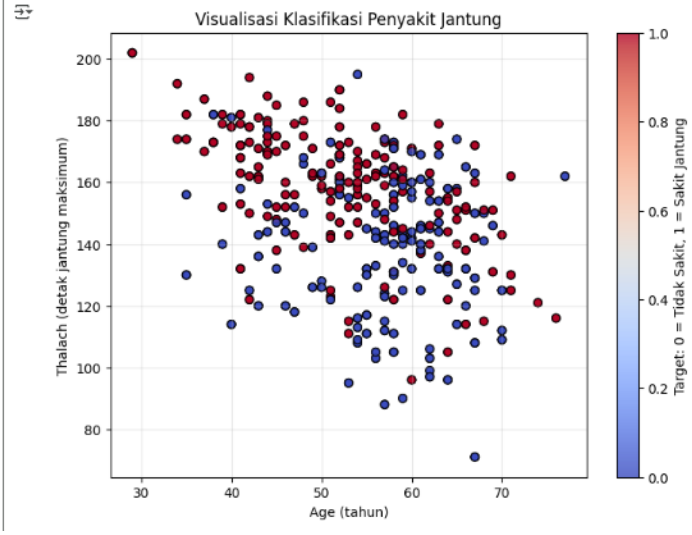
```
[25]
plt.figure(figsize=(8, 6))

# Scatter plot: x = age, y = thalach, warna berdasarkan target
plt.scatter(df['age'], df['thalach'],
            c=df['target'],          # Warna berdasarkan target (0 atau 1)
            cmap='coolwarm',         # Biru = 0, Merah = 1
            alpha=0.8,
            edgecolors='k')

# Label dan judul
plt.xlabel('Age (tahun)')
plt.ylabel('Thalach (detak jantung maksimum)')
plt.title('Visualisasi Klasifikasi Penyakit Jantung')

# Tambahkan colorbar
cbar = plt.colorbar()
cbar.set_label('Target: 0 = Tidak Sakit, 1 = Sakit Jantung')

plt.grid(True, alpha=0.3)
plt.show()
```



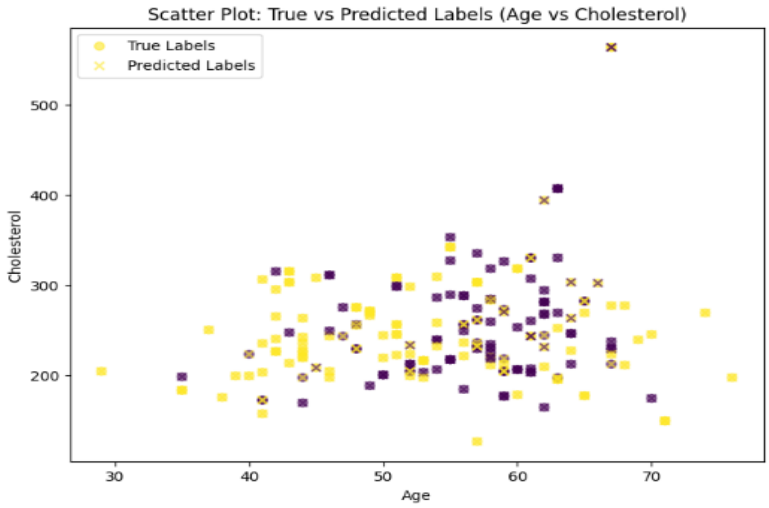
Scatter Plot yang memetakan hubungan antara **Usia (*Age*)** di sumbu-X dengan **Detak Jantung Maksimum (*Thalach*)** di sumbu-Y, menggunakan data dari df. Titik-titik data diwarnai berdasarkan kolom **target** (c=df['target']), di mana skema warna coolwarm digunakan untuk membedakan antara pasien yang **Tidak Sakit Jantung (0 - Biru)** dan yang **Sakit Jantung (1 - Merah)**. Visualisasi ini, berjudul "Visualisasi Klasifikasi Penyakit Jantung", dilengkapi dengan *colorbar* untuk menginterpretasikan warna target dan *grid* untuk memudahkan pembacaan titik-titik data.

5.3 Scatter Plot : True vs Predicted Labels

```
# Train a simple logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Plot similar to the example, using 'age' and 'chol' as features
plt.figure(figsize=(8, 6))
plt.scatter(X_test['age'], X_test['chol'], c=y_test, cmap='viridis', marker='o', label='True Labels', alpha=0.6)
plt.scatter(X_test['age'], X_test['chol'], c=y_pred, cmap='viridis', marker='x', label='Predicted Labels', alpha=0.6)
plt.title('Scatter Plot: True vs Predicted Labels (Age vs Cholesterol)')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.legend()
plt.show()
```



Menyiapkan data (fitur X dan target y) dan membaginya menjadi set *training* dan *testing*, kemudian melatih model klasifikasi **Logistic Regression** sederhana pada data *training* tersebut. Setelah prediksi dibuat pada data *testing*, kode ini menghasilkan **Scatter Plot** berjudul "True vs Predicted Labels (Age vs Cholesterol)", yang memvisualisasikan data *testing* berdasarkan fitur **Age** (sumbu X) dan **Cholesterol** (sumbu Y). Plot ini menggunakan dua jenis penanda: label sebenarnya (*True Labels*) yang ditandai dengan lingkaran ('o') dan label yang diprediksi (*Predicted Labels*) yang ditandai dengan silang ('x'), memungkinkan perbandingan visual antara hasil prediksi model dengan kondisi aktual.

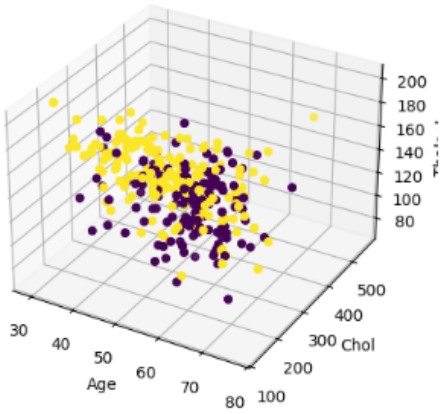
5.4 Visualisasi 3D

[12]

```
# Visualisasi 3D sederhana: Pilih 3 fitur utama untuk plot (misalnya age, chol, thalach)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot data berdasarkan target
ax.scatter(df['age'], df['chol'], df['thalach'], c=df['target'], cmap='viridis')

ax.set_xlabel('Age')
ax.set_ylabel('Chol')
ax.set_zlabel('Thalach')
plt.show()
```



Visualisasi 3D sederhana untuk mengeksplorasi distribusi data berdasarkan tiga fitur utama: **Age** (sumbu X), **Cholesterol** (sumbu Y), dan **Thalach** (Detak Jantung Maksimum, sumbu Z). Data divisualisasikan menggunakan *scatter plot* 3D, di mana setiap titik diwarnai berdasarkan label target (c=df['target']), yang dalam contoh ini menggunakan skema warna viridis untuk membedakan dua kelompok (pasien sakit jantung dan tidak sakit jantung). Tujuan dari plot ini adalah untuk melihat apakah kedua kelas target dapat dipisahkan secara spasial di ruang 3-dimensi yang dibentuk oleh ketiga fitur tersebut.

Referensi:

Munir, S., Seminar, K. B., Sudradjat, Sukoco, H., & Buono, A. (2022). The Use of Random Forest Regression for Estimating Leaf Nitrogen Content of Oil Palm Based on Sentinel 1-A Imagery. *Information*, *14*(1), 10. https://doi.org/10.3390/info14010010

Seminar, K. B., Imantho, H., Sudradjat, Yahya, S., Munir, S., Kaliana, I., Mei Haryadi, F., Noor Baroroh, A., Supriyanto, Handoyo, G. C., Kurnia Wijayanto, A., Ijang Wahyudin, C., Liyantono, Budiman, R., Bakir Pasaman, A., Rusiawan, D., & Sulastri. (2024). PreciPalm: An Intelligent System for Calculating Macronutrient Status and Fertilizer Recommendations for Oil Palm on Mineral Soils Based on a Precision Agriculture Approach. *Scientific World Journal*, *2024*(1). https://doi.org/10.1155/2024/1788726