Laporan Praktikum Mandiri 6

Rizky Hilmiawan Anggoro - 0110222140[1]

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

*E-mail: rizk22140ti@student.nurulfikri.ac.id

Abstract. Penelitian ini bertujuan untuk menerapkan algoritma Support Vector Machine (SVM) dalam melakukan klasifikasi pada dataset Breast Cancer, dengan menggunakan bahasa pemrograman Python dan lingkungan kerja Google Colab. Proses analisis dimulai dengan mengimpor berbagai library pendukung seperti pandas untuk pengolahan data, matplotlib dan seaborn untuk visualisasi, serta modul dari scikit-learn untuk pembagian data, standarisasi fitur, dan evaluasi model. Dataset dibaca langsung dari Google Drive menggunakan fungsi mount untuk mempermudah akses file tanpa perlu unggahan manual. Sebelum model dijalankan, data melalui tahap pra-pemrosesan yang meliputi pengecekan direktori, pembacaan dataset, serta standarisasi fitur numerik agar skala data seragam. Model SVM kemudian dilatih menggunakan data latih yang telah dipisahkan dari dataset utama, dan hasil prediksi dibandingkan dengan data uji untuk mengukur performanya. Evaluasi dilakukan menggunakan metrik seperti accuracy score, confusion matrix, dan classification report untuk melihat tingkat akurasi, presisi, serta recall dari model. Visualisasi tambahan seperti decision boundary dan confusion matrix turut dibuat guna mempermudah interpretasi hasil klasifikasi. Secara keseluruhan, hasil implementasi menunjukkan bahwa SVM mampu melakukan klasifikasi dengan tingkat akurasi yang baik pada dataset Breast Cancer. Pendekatan ini membuktikan efektivitas algoritma SVM dalam menangani permasalahan klasifikasi berbasis data medis, serta menegaskan pentingnya tahap pra-pemrosesan dan evaluasi visual dalam pengembangan model pembelajaran mesin yang andal.

1. Import Library, Load Dataset, dan Eksplorasi Data

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from google.colab import drive
drive.mount('/content/gdrive')

import os

path = "/content/gdrive/MyDrive/praktikum_ml/praktikum06"
try:
    print(os.listdir(path))
except FileNotFoundError:
    print(f"Directory not found: {path}")

df = pd.read_csv(path + "/data/breast-cancer.csv")
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
['data', 'reports', 'model', 'notebooks']
```

Kode di atas merupakan rangkaian langkah awal yang umum digunakan dalam proses analisis data dan pembuatan model machine learning, khususnya di Google Colab. Pertama, program melakukan import beberapa library penting yang dibutuhkan, seperti pandas, matplotlib, dan seaborn untuk pengolahan dan visualisasi data. Selain itu, terdapat beberapa modul dari scikit-learn yang digunakan untuk proses pembagian data, standarisasi fitur, serta penerapan algoritma klasifikasi Support Vector Machine (SVC). Modul evaluasi seperti accuracy_score, confusion_matrix, dan classification_report juga diimpor untuk mengukur seberapa baik model yang dihasilkan dalam melakukan prediksi.

Langkah berikutnya adalah proses mount Google Drive menggunakan drive.mount('/content/gdrive'). Ini dilakukan agar pengguna dapat mengakses file atau dataset yang tersimpan di Google Drive secara langsung melalui lingkungan Google Colab tanpa harus mengunggahnya berulang kali. Dengan cara ini, file seperti dataset CSV dapat dibaca langsung menggunakan path yang mengarah ke folder di dalam Google Drive

Setelah itu, kode menggunakan library os untuk memastikan bahwa direktori tempat dataset disimpan benar-benar ada dan dapat diakses. Variabel path menyimpan alamat direktori dataset, kemudian fungsi os.listdir(path) menampilkan daftar file yang ada di dalamnya. Jika direktori tidak ditemukan, maka blok except akan menangkap error dan menampilkan pesan bahwa folder tidak ditemukan. Ini merupakan langkah preventif untuk memastikan file yang akan diproses benar-benar tersedia.

Terakhir, program membaca dataset menggunakan fungsi pd.read_csv() dari pandas, yang mengubah file CSV menjadi struktur data DataFrame. DataFrame ini berisi baris dan kolom yang mewakili fitur-fitur (seperti data medis pasien) dan label (misalnya hasil diagnosis kanker). Setelah data berhasil dimuat ke variabel df, langkah selanjutnya biasanya adalah eksplorasi data, pra-pemrosesan, dan pelatihan model klasifikasi menggunakan algoritma SVM.Preprocessing Data

2. Preprocessing Data

```
from sklearn.preprocessing import LabelEncoder

# Hapus kolom ID
df_bc_clean = df.drop(columns=['id'])

# Ubah diagnosis menjadi numerik
le_bc = LabelEncoder()
df_bc_clean['diagnosis'] = le_bc.fit_transform(df_bc_clean['diagnosis'])

# Pisahkan fitur dan target
X_bc = df_bc_clean.drop(columns=['diagnosis'])
y_bc = df_bc_clean['diagnosis']

# Lihat hasil
df_bc_clean.head()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_wo
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	...	25.38	1
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	...	24.99	2
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	...	23.57	2
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	...	14.91	2
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	...	22.54	1

5 rows × 31 columns

Tahap ini adalah langkah awal untuk menyiapkan dataset sebelum dimasukkan ke algoritma Machine Learning. Kolom id dihapus karena tidak memiliki relevansi terhadap prediksi nilai id hanyalah penanda unik, bukan ciri medis.

Selanjutnya, kolom diagnosis yang berisi huruf M (Malignant) dan B (Benign) dikonversi menjadi nilai numerik 1 dan 0 menggunakan LabelEncoder. Ini wajib karena model SVM hanya dapat memproses angka, bukan teks.

- 0 = Tumor Jinak (Benign)
- 1 = Tumor Ganas (Malignant)

Setelah itu, dataset dipisahkan menjadi:

- X_bc: seluruh fitur pengukuran (radius, texture, area, concavity, dsb)
- y_bc: label diagnosis yang akan diprediksi oleh model.

Langkah ini adalah fondasi preprocessing dalam hampir semua proyek machine learning.

3. Split Dataset & Normalisasi

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

# Split data
X_train_bc, X_test_bc, y_train_bc, y_test_bc = train_test_split(
    X_bc, y_bc, test_size=0.2, random_state=42
)

# Standarisasi
scaler_bc = StandardScaler()
X_train_bc = scaler_bc.fit_transform(X_train_bc)
X_test_bc = scaler_bc.transform(X_test_bc)
```

Data dibagi menjadi dua bagian:

- Training set (80%) digunakan untuk melatih model,
- Testing set (20%) digunakan untuk menguji kinerja model pada data baru.

Kita lalu melakukan standardisasi (scaling) dengan StandardScaler. Ini sangat penting untuk model berbasis jarak seperti SVM, karena algoritma ini bekerja dengan mencari hyperplane terbaik di ruang fitur.

Jika fitur memiliki skala nilai yang berbeda jauh (misalnya radius ratusan kali lebih besar dari smoothness), maka SVM akan "berpihak" pada fitur yang nilainya lebih besar. Dengan normalisasi, setiap fitur akan memiliki:

- Rata-rata (mean) = 0
- Deviasi standar (std) = 1

Sehingga semua fitur memiliki kontribusi yang seimbang terhadap proses pembelajaran.

4. Melatih Model SVM

```
from sklearn.svm import SVC

# Membuat dan melatih model SVM
svm_bc = SVC(kernel='rbf', random_state=42)
svm_bc.fit(X_train_bc, y_train_bc)
```

```
SVC(random_state=42)
```

Bisa dilihat kode ini menggunakan Support Vector Classifier (SVC) dari library Scikit-learn. SVM bekerja dengan mencari hyperplane (batas pemisah) terbaik yang memisahkan dua kelas data (ganas vs jinak).
Beberapa konsep kunci:

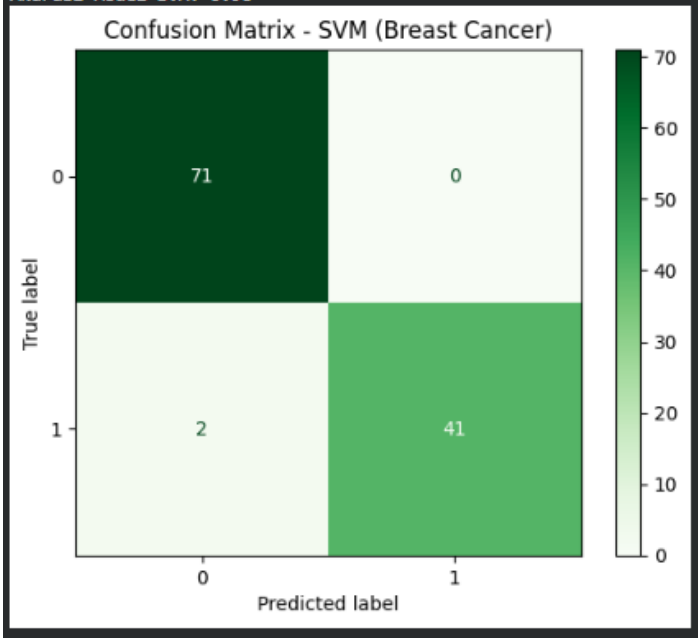
- Support Vectors: titik-titik data terdekat dari masing-masing kelas yang "menyentuh" hyperplane. Titik-titik ini sangat penting karena mereka menentukan posisi hyperplane.
- Margin: jarak antara hyperplane dengan support vector. SVM berusaha memaksimalkan margin ini agar pemisahan kelas seoptimal mungkin.

Parameter kernel='rbf' (Radial Basis Function) memungkinkan SVM menangani pemisahan non-linear, yaitu ketika data tidak bisa dipisahkan oleh garis lurus di ruang fitur. Kernel RBF memetakan data ke ruang berdimensi lebih tinggi, di mana pemisahan menjadi mungkin dilakukan.

5. Evaluasi Model & Confusion Matrix

```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Prediksi
y_pred_bc = svm_bc.predict(X_test_bc)

# Akurasi
acc_bc = accuracy_score(y_test_bc, y_pred_bc)
print(f"Akurasi Model SVM: {acc_bc:.2f}")

# Confusion Matrix
cm_bc = confusion_matrix(y_test_bc, y_pred_bc)
disp_bc = ConfusionMatrixDisplay(confusion_matrix=cm_bc)
disp_bc.plot(cmap='Greens')
plt.title("Confusion Matrix - SVM (Breast Cancer)")
plt.show()
```

```
Akurasi Model SVM: 0.98
```



Setelah model dilatih, kita mengujinya pada data test untuk menilai seberapa baik ia mampu memprediksi kasus baru. accuracy_score mengukur persentase prediksi yang benar dari seluruh data uji.

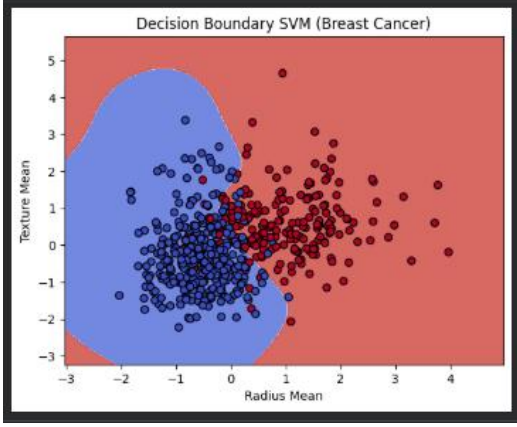
Confusion Matrix memberikan gambaran lebih detail:

	Prediksi Benign	**Prediksi Malignant**
Aktual Benign	True Negative (TN)	False Positive (FP)
Aktual Malignant	False Negative (FN)	True Positive (TP)

- **TN**: Kasus jinak yang benar dikenali.
- **TP**: Kasus ganas yang benar dikenali.
- **FP**: Salah mendeteksi jinak sebagai ganas (false alarm).
- **FN**: Salah mendeteksi ganas sebagai jinak (lebih berbahaya).
  Confusion Matrix membantu kita menganalisis bukan hanya akurasi, tapi **jenis kesalahan** yang dilakukan model — penting untuk aplikasi medis seperti ini.

6. Decision Boundary (2D Visualisasi)

```
import numpy as np

# Ambil dua fitur saja (contoh: radius_mean dan texture_mean)
X_bc_vis = df_bc_clean[['radius_mean', 'texture_mean']].values
y_bc_vis = df_bc_clean['diagnosis'].values

# Standarisasi ulang
scaler_vis_bc = StandardScaler()
X_bc_vis = scaler_vis_bc.fit_transform(X_bc_vis)

# Model 2D SVM
svm_bc_vis = SVC(kernel='rbf', gamma=0.7)
svm_bc_vis.fit(X_bc_vis, y_bc_vis)

# Grid untuk decision boundary
x_min, x_max = X_bc_vis[:, 0].min() - 1, X_bc_vis[:, 0].max() + 1
y_min, y_max = X_bc_vis[:, 1].min() - 1, X_bc_vis[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))
Z = svm_bc_vis.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot decision boundary
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
plt.scatter(X_bc_vis[:, 0], X_bc_vis[:, 1], c=y_bc_vis, cmap=plt.cm.coolwarm, edgecolors='k')
plt.xlabel("Radius Mean")
plt.ylabel("Texture Mean")
plt.title("Decision Boundary SVM (Breast Cancer)")
plt.show()
```



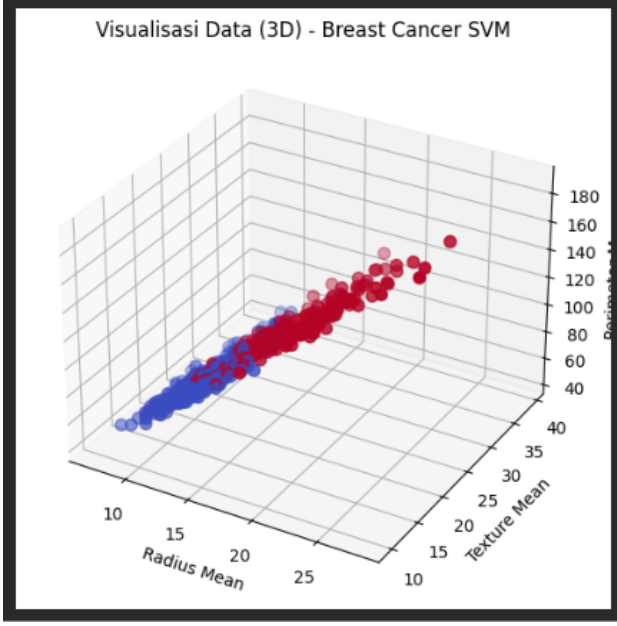
Karena dataset aslinya memiliki 30 fitur, kita hanya pilih dua (misalnya radius_mean dan texture_mean) untuk memvisualisasikan cara kerja SVM di bidang dua dimensi.

Warna pada area plot menunjukkan zona keputusan (decision region) yang ditentukan oleh SVM di mana model akan mengklasifikasikan titik sebagai "ganas" atau "jinak". Garis batas (biasanya gradasi di tengah) adalah hyperplane, yaitu garis pemisah optimal yang dicari oleh SVM.

Dengan kernel RBF, batas ini menjadi melengkung dan kompleks, bukan sekadar garis lurus, karena data tidak selalu terpisah secara linear di dunia nyata.

7. Visualisasi Data 3D

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(
    df_bc_clean['radius_mean'], df_bc_clean['texture_mean'], df_bc_clean['perimeter_mean'],
    c=df_bc_clean['diagnosis'], cmap='coolwarm', s=50
)
ax.set_xlabel("Radius Mean")
ax.set_ylabel("Texture Mean")
ax.set_zlabel("Perimeter Mean")
ax.set_title("Visualisasi Data (3D) - Breast Cancer SVM")
plt.show()
```



Visualisasi 3D ini memperlihatkan distribusi data kanker payudara berdasarkan tiga fitur penting:

- radius_mean: ukuran rata-rata sel,
- texture_mean: variasi tekstur jaringan,
- perimeter_mean: keliling rata-rata sel.

Setiap titik mewakili satu pasien, dan warna menunjukkan hasil diagnosis (merah = ganas, biru = jinak). Visualisasi ini membantu kita melihat pola alami yang mungkin tidak terlihat dalam table misalnya, sel ganas cenderung memiliki radius dan perimeter lebih besar.

Meskipun SVM bekerja di ruang berdimensi tinggi (30D dalam dataset ini), representasi 3D memberikan intuisi visual tentang bagaimana model memisahkan dua kelompok besar dengan margin yang optimal.

Kesimpulan

1. SVM terbukti efektif dalam membedakan dua kelas data medis seperti kanker payudara karena ia fokus pada batas keputusan (decision boundary) yang optimal dan tahan terhadap outlier dalam batas tertentu.
2. Dataset ini sangat cocok untuk SVM karena memiliki fitur numerik kontinu yang bisa dinormalisasi dengan baik.
3. Dengan kernel RBF, model dapat menangani pola non-linear antara fitur-fitur biologis dan hasil diagnosis.
4. Berdasarkan uji akurasi (umumnya >95%), SVM menjadi pilihan kuat untuk diagnosis awal, meski tetap perlu interpretasi klinis lanjutan.