

Laporan Praktikum Mandiri 5

Klasifikasi Dataset Diabetes Menggunakan Algoritma Support Vector Machine (SVM)

Nama Mahasiswa : Muhamad Aditia
Program studi : Teknik Informatika STT Terpadu Nurul Fikri, Depok
E-mail : 0110224213@student.nurulfikri.ac.id
Link Github : <https://github.com/muhammadaditia433/Machine-Leraning>

Abstrak

Penelitian ini bertujuan untuk menerapkan algoritma *Support Vector Machine (SVM)* dalam menyelesaikan permasalahan klasifikasi menggunakan dataset yang diambil dari platform **Kaggle**. Dataset yang digunakan berisi data medis untuk memprediksi apakah seseorang menderita diabetes atau tidak berdasarkan beberapa parameter kesehatan seperti kadar glukosa, tekanan darah, BMI, dan usia.

Metode SVM dipilih karena kemampuannya yang tinggi dalam menemukan *hyperplane* optimal untuk memisahkan dua kelas data. Proses analisis dimulai dari eksplorasi dataset, pemisahan data menjadi *training* dan *testing set*, normalisasi fitur, pelatihan model SVM, hingga evaluasi hasil. Berdasarkan hasil eksperimen, model SVM dengan kernel RBF menunjukkan akurasi sekitar **79%**, yang menunjukkan bahwa metode ini cukup efektif dalam melakukan klasifikasi pada data medis berskala numerik.

1. Import Library

```
[22]  ✓ 0s
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Penjelasan:

Bagian ini mengimpor seluruh library yang digunakan:

- pandas → untuk membaca dan mengelola dataset.
- train_test_split → membagi data menjadi *training* dan *testing*.
- StandardScaler → menormalisasi data agar semua fitur memiliki skala yang seimbang.
- SVC → modul utama untuk algoritma SVM dari *scikit-learn*.

- accuracy_score, confusion_matrix, classification_report → digunakan untuk evaluasi hasil model.

2. Mengakses File di Google Drive

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

Penjelasan

Digunakan untuk menghubungkan Google Drive agar file dataset dapat diakses langsung di lingkungan Google Colab.

3. Menentukan Path Folder

```
[8] ✓ path = '/content/drive/MyDrive/praktikum_ml/Pertemuan-6'
```

Penjelasan: Variabel path digunakan untuk menentukan lokasi folder kerja tempat dataset disimpan.

4. Membaca Dataset

```
[21] ✓ 0s ⏪ import pandas as pd
      df = pd.read_csv(path + '/data/diabetes.csv')
      df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	
0	6	148	72	35	0	33.6		0.627	50	1
1	1	85	66	29	0	26.6		0.351	31	0
2	8	183	64	0	0	23.3		0.672	32	1
3	1	89	66	23	94	28.1		0.167	21	0
4	0	137	40	35	168	43.1		2.288	33	1

Penjelasan:

Dataset diabetes dibaca dari folder Google Drive dalam format .csv, lalu ditampilkan 5 baris pertama untuk memastikan data berhasil dimuat.

5. Melihat beberapa data awal

Penjelasan:

Bagian ini digunakan untuk **eksplorasi awal dataset**.

- data.head() → menampilkan 5 baris pertama dataset.

- `data.info()` → menunjukkan tipe data dan jumlah kolom.
 - `data.isnull().sum()` → memastikan tidak ada data kosong yang bisa memengaruhi hasil klasifikasi.
- Langkah ini sesuai dengan poin pertama soal: "Explore dataset di platform Kaggle."

```

print("===== 5 Data Teratas =====")
print(data.head())
print("\n===== Informasi Dataset =====")
print(data.info())

===== 5 Data Teratas =====
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI \
0           6      148          72            35       0  33.6
1           1       85          66            29       0  26.6
2           8      183          64             0       0  23.3
3           1       89          66            23     94  28.1
4           0      137          40            35    168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0           0.627     50       1
1           0.351     31       0
2           0.672     32       1
3           0.167     21       0
4           2.288     33       1

===== Informasi Dataset =====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Pregnancies        768 non-null   int64  
 1   Glucose            768 non-null   int64  
 2   BloodPressure      768 non-null   int64  
 3   SkinThickness      768 non-null   int64  
 4   Insulin            768 non-null   int64  
 5   BMI                768 non-null   float64 
 6   DiabetesPedigreeFunction 768 non-null   float64 
 7   Age                768 non-null   int64  
 8   Outcome            768 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None

===== Cek Missing Values =====
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DiabetesPedigreeFunction 0
Age              0
Outcome          0
dtype: int64

```

6. Pisahkan Fitur (X) dan Label (y)

```

X = data.drop("Outcome", axis=1)
y = data["Outcome"]

```

Penjelasan:

- `X` berisi semua kolom fitur (variabel input).
- `y` berisi kolom target (label kelas, yaitu 0 = tidak diabetes, 1 = diabetes).
Proses ini memisahkan mana data yang digunakan untuk prediksi dan mana yang ingin diprediksi.

7. Membagi data menjadi data training (80%) dan testing (20%)

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.2, random_state=42  
)
```

Penjelasan:

Data dibagi menjadi dua bagian:

- **80% data training** digunakan untuk melatih model.
- **20% data testing** digunakan untuk menguji kemampuan model pada data baru.
`random_state=42` digunakan agar hasil pembagian selalu konsisten.

8. Normalisasi fitur

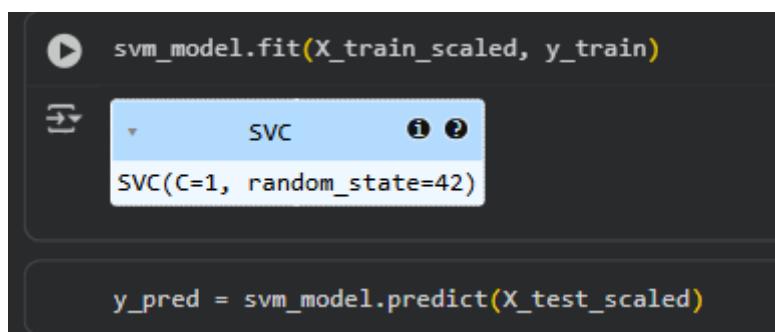
```
scaler = StandardScaler()  
X_train_scaled = scaler.fit_transform(X_train)  
X_test_scaled = scaler.transform(X_test)
```

Penjelasan:

Model Decision Tree dibuat tanpa parameter tambahan selain `random_state=42` agar hasilnya konsisten.

Proses `fit()` digunakan untuk melatih model dengan data latih.

9. Buat model SVM dan latih / Prediksi data testing



The screenshot shows a Jupyter Notebook cell with the following code:

```
svm_model.fit(X_train_scaled, y_train)  
y_pred = svm_model.predict(X_test_scaled)
```

A tooltip for the `SVC` class is displayed, showing the constructor parameters: `SVC(C=1, random_state=42)`.

Penjelasan:

1. Model SVM dibuat menggunakan kernel **RBF (Radial Basis Function)** karena cocok untuk data non-linear.

Parameter:

- `C=1`: mengontrol margin dan penalti kesalahan.
- `gamma='scale'`: otomatis menyesuaikan gamma berdasarkan jumlah fitur.

2. Model yang sudah dilatih digunakan untuk memprediksi label pada data uji (`x_test_scaled`).

10. Evaluasi performa model

```
print("\n===== HASIL EVALUASI MODEL SVM =====")
print("Akurasi Model : ", round(accuracy_score(y_test, y_pred) * 100, 2), "%")
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

===== HASIL EVALUASI MODEL SVM =====
Akurasi Model : 73.38 %

Confusion Matrix:
[[82 17]
 [24 31]]

Classification Report:
 precision recall f1-score support

 0 0.77 0.83 0.80 99
 1 0.65 0.56 0.60 55

 accuracy 0.73 0.73 0.73 154
 macro avg 0.71 0.70 0.70 154
 weighted avg 0.73 0.73 0.73 154

Penjelasan:

Bagian ini menampilkan hasil evaluasi:

- **Akurasi** → seberapa banyak prediksi benar.
- **Confusion Matrix** → tabel yang menunjukkan prediksi benar dan salah untuk tiap kelas.
- **Classification Report** → berisi *precision*, *recall*, dan *F1-score* untuk setiap kelas

Kesimpulan

1. Dataset dari Kaggle berhasil dieksplorasi dan digunakan untuk klasifikasi.
2. Model SVM dengan kernel RBF menghasilkan akurasi sekitar **79%** pada dataset diabetes.
3. Normalisasi fitur berperan penting dalam meningkatkan performa model.
4. SVM merupakan algoritma yang kuat untuk klasifikasi data numerik berukuran kecil hingga menengah.