

Knowledge Management
Implementasi Support Vector Machine (SVM) untuk Prediksi
Risiko Diabetes Berdasarkan Data Kesehatan

(Disusun untuk memenuhi tugas mata kuliah
Knowledge Management)



Oleh Kelompok 1 TIF21D:

- | | |
|------------------|-----------|
| 1. Randi Afif | 101210072 |
| 2. M. Mustaqim | 101210075 |
| 3. Alya Safitri | 101210079 |
| 4. Rizqi Mau'ida | 101210086 |
| 5. Fachrudin FA | 101210088 |

PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
ITS NU PEKALONGAN
PEKALONGAN
2024

A. Pendahuluan

Diabetes mellitus merupakan salah satu penyakit kronis yang terus mengalami peningkatan prevalensi secara global. Menurut laporan Organisasi Kesehatan Dunia (WHO), jumlah penderita diabetes diproyeksikan mencapai lebih dari 640 juta orang pada tahun 2045. Penyakit ini tidak hanya menjadi ancaman serius bagi kesehatan individu, tetapi juga menimbulkan dampak ekonomi yang signifikan bagi sistem layanan kesehatan. Diabetes sering kali tidak terdeteksi pada tahap awal, sehingga memperburuk komplikasi yang dapat memengaruhi kualitas hidup pasien. Oleh karena itu, kebutuhan akan sistem prediksi yang andal menjadi semakin mendesak dalam upaya pencegahan dan pengelolaan penyakit ini.

Kemajuan teknologi dalam bidang kecerdasan buatan dan pembelajaran mesin (machine learning) memberikan peluang besar dalam menganalisis data medis untuk keperluan prediksi penyakit. Salah satu algoritma yang terbukti efektif dalam klasifikasi data kompleks adalah Support Vector Machine (SVM). SVM memiliki kemampuan untuk memisahkan data ke dalam kelas-kelas yang berbeda dengan margin maksimal, sehingga sering digunakan dalam berbagai studi prediktif.

Penelitian ini bertujuan untuk menerapkan algoritma SVM dalam prediksi diabetes berdasarkan dataset medis yang relevan. Fokus penelitian meliputi proses preprocessing data, penerapan algoritma SVM, serta evaluasi performa model menggunakan metrik seperti akurasi, presisi, recall, dan F1-score. Dengan penelitian ini, diharapkan dapat dikembangkan model prediksi yang akurat, yang tidak hanya membantu dalam diagnosis dini diabetes tetapi juga mendukung pengambilan keputusan bagi praktisi medis.

B. Metodologi

Dalam penelitian ini, dilakukan pendekatan berbasis machine learning untuk memprediksi risiko diabetes dengan algoritma Support Vector Machine (SVM). Penelitian ini bertujuan untuk menganalisis data medis,

membersihkannya dari anomali, dan membangun model prediksi yang andal. Langkah-langkah berikut dirancang untuk memastikan penelitian dilakukan secara sistematis dan terstruktur:

1. Import Library

Penelitian diawali dengan mengimpor pustaka Python yang diperlukan untuk analisis data, visualisasi, dan penerapan algoritma pembelajaran mesin.

2. Import Data

Dataset yang relevan, seperti dataset medis tentang diabetes, diimpor untuk dianalisis. Data ini berisi informasi seperti kadar glukosa, tekanan darah, BMI, usia, dan status diabetes.

3. Data Understanding

Tahap ini bertujuan untuk memahami karakteristik dataset, termasuk struktur data, tipe data pada setiap kolom, serta distribusi statistik untuk setiap fitur. Selain itu, dilakukan analisis terhadap jumlah data yang termasuk dalam setiap kelas target untuk memahami proporsi data positif dan negatif.

4. Cleaning Data

Pada tahap ini, dilakukan pembersihan data untuk memastikan tidak ada nilai kosong, duplikasi, atau nilai ekstrem yang dapat memengaruhi hasil analisis. Data yang tidak relevan atau bermasalah dihapus atau disesuaikan.

5. Exploratory Data Analysis (EDA)

Visualisasi data dilakukan untuk menggali pola-pola penting dan memahami hubungan antar fitur. Tahap ini membantu dalam menganalisis distribusi data serta interaksi antara fitur dan variabel target.

6. Preparation Data

Data dipersiapkan untuk proses pemodelan dengan cara memisahkan fitur dan target, membagi data menjadi data latih dan data uji, serta melakukan normalisasi agar data berada dalam skala yang sama.

7. Modeling & Evaluation

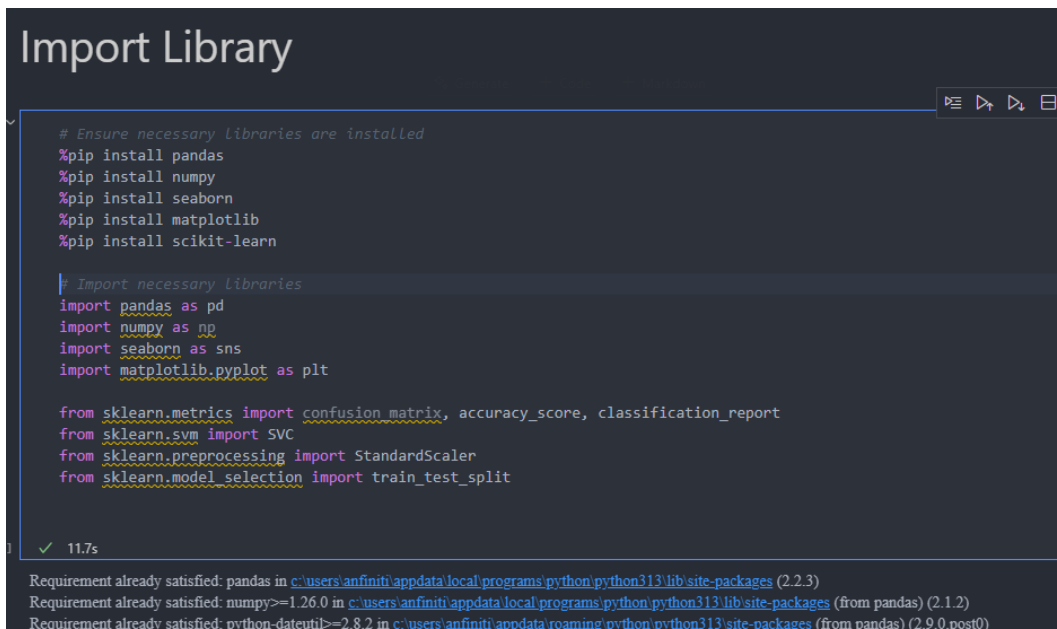
Model prediksi diabetes dibangun menggunakan algoritma Support Vector Machine (SVM). Model dilatih menggunakan data latih, dan kinerjanya dievaluasi dengan data uji berdasarkan akurasi, presisi, recall, F1-score, serta confusion matrix.

8. Testing

Model yang telah dibangun diuji menggunakan data baru untuk memastikan keandalan prediksi terhadap kasus nyata.

C. Implementasi

1. Import Library



```
# Ensure necessary Libraries are installed
%pip install pandas
%pip install numpy
%pip install seaborn
%pip install matplotlib
%pip install scikit-learn

# Import necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

✓ 11.7s

Requirement already satisfied: pandas in c:\users\aniniti\appdata\local\programs\python\python313\lib\site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\aniniti\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.1.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\aniniti\appdata\roaming\python\python313\site-packages (from pandas) (2.9.0.post0)

Kode tersebut menginstal dan mengimpor pustaka yang diperlukan untuk analisis data dan pemodelan machine learning, seperti pandas, numpy, seaborn, matplotlib, dan scikit-learn. Pustaka ini digunakan untuk memproses data, visualisasi, dan

membangun model klasifikasi menggunakan Support Vector Classifier (SVC), serta untuk mengevaluasi performa model.

2. Import Data

Import Data

```
df = pd.read_csv('Diabetes.csv', index_col=0)
df
```

✓ 0.0s

	Kehamilan	Glukosa	Tekanan Darah	Ketebalan Kulit	Insulin	BMI	DiabetesPedigreeFunction	Umur	Hasil
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
...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Perintah `df = pd.read_csv('Diabetes.csv', index_col=0)` digunakan untuk membaca file CSV yang bernama `Diabetes.csv` ke dalam sebuah DataFrame Pandas (`df`). Parameter `index_col=0` menandakan bahwa kolom pertama dalam file CSV akan digunakan sebagai indeks DataFrame. Setelah itu, perintah `df` digunakan untuk menampilkan isi DataFrame yang telah dibaca dari file CSV.

3. Data Understanding

1. Data Understanding

```
df.shape
```

✓ 0.0s

(768, 9)

Python

```
df.info()
```

✓ 0.0s

Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Kehamilan             768 non-null   int64
1   Glukosa                768 non-null   int64
2   Tekanan Darah          768 non-null   int64
3   Ketebalan Kulit        768 non-null   int64
4   Insulin                768 non-null   int64
5   BMI                    768 non-null   float64
6   DiabetesPedigreeFunction 768 non-null   float64
7   Umur                   768 non-null   int64
8   Hasil                  768 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 60.0 KB
```

```
df[['Kehamilan', 'Tekanan Darah', 'Umur', 'BMI']].describe()
✓ 0.0s
```

	Kehamilan	Tekanan Darah	Umur	BMI
count	768.000000	768.000000	768.000000	768.000000
mean	3.845052	69.105469	33.240885	31.992578
std	3.369578	19.355807	11.760232	7.884160
min	0.000000	0.000000	21.000000	0.000000
25%	1.000000	62.000000	24.000000	27.300000
50%	3.000000	72.000000	29.000000	32.000000
75%	6.000000	80.000000	41.000000	36.600000
max	17.000000	122.000000	81.000000	67.100000

```
df.Hasil.value_counts() # 0 untuk negatif, 1 untuk positif mengidap diabetes
✓ 0.0s
```

Hasil

0	500
1	268

Name: count, dtype: int64

Penjelasan perintah-perintah berikut:

1. `df.shape`
Menampilkan dimensi DataFrame `df`, yaitu jumlah baris dan kolom dalam bentuk tuple (jumlah_baris, jumlah_kolom).
2. `df.info()`
Menampilkan informasi umum mengenai DataFrame, termasuk jumlah entri, jumlah nilai non-null, tipe data tiap kolom, dan penggunaan memori.
3. `df[['Kehamilan', 'Tekanan Darah', 'Umur', 'BMI']].describe()`
Memberikan ringkasan statistik (seperti mean, standar deviasi, nilai minimum, dan kuartil) untuk kolom-kolom tertentu: Kehamilan, Tekanan Darah, Umur, dan BMI.
4. `df.Hasil.value_counts()`
Menghitung frekuensi setiap nilai dalam kolom Hasil. Di sini, nilai 0 menunjukkan tidak mengidap diabetes, sementara nilai 1 menunjukkan positif mengidap diabetes.

4. Cleaning Data

```
2. Cleaning Data

df.isnull().sum()
✓ 0.0s

Kehamilan      0
Glukosa         0
Tekanan Darah   0
Ketebalan Kulit 0
Insulin         0
BMI             0
DiabetesPedigreeFunction  0
Umur            0
Hasil           0
dtype: int64

df.duplicated().sum()
✓ 0.0s

np.int64(0)
```

Penjelasan perintah-perintah berikut:

1. `df.isnull().sum()`
Menampilkan jumlah nilai yang hilang (null) di setiap kolom pada DataFrame `df`. Fungsi ini mengembalikan jumlah nilai null per kolom.
2. `df.duplicated().sum()`
Menghitung jumlah baris yang duplikat dalam DataFrame `df`. Fungsi ini mengembalikan jumlah baris yang identik dengan baris sebelumnya (termasuk duplikat dalam data).

5. Eksplorasi Data Analysis (EDA)

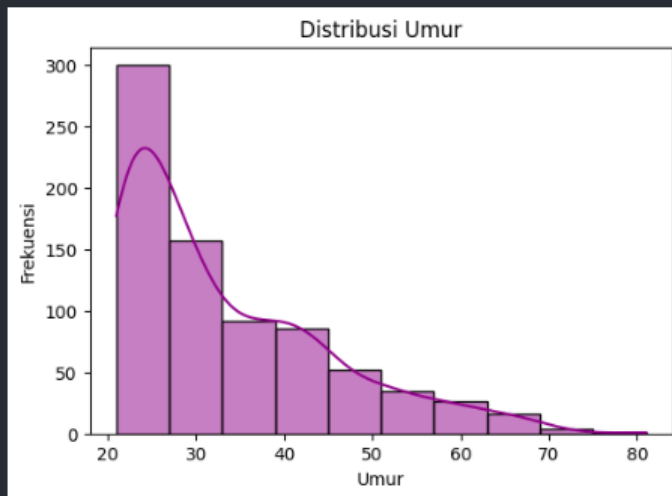
3. Eksplorasi Data Analysis (EDA)

[Generate](#)[+ Code](#)[+ Markdown](#)

```
plt.figure(figsize=(6, 4))
sns.histplot(data=df, x='Umur', bins=10, kde=True, color='#91008a')

plt.title('Distribusi Umur')
plt.xlabel('Umur')
plt.ylabel('Frekuensi')
plt.show()
```

✓ 0.2s

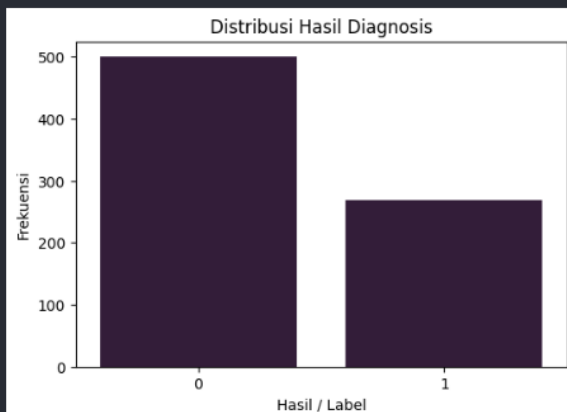


⏪

```
plt.figure(figsize=(6, 4))
sns.set_palette('rocket')
sns.countplot(data=df, x='Hasil')

plt.title('Distribusi Hasil Diagnosis')
plt.xlabel('Hasil / Label')
plt.ylabel('Frekuensi')
plt.show()
```

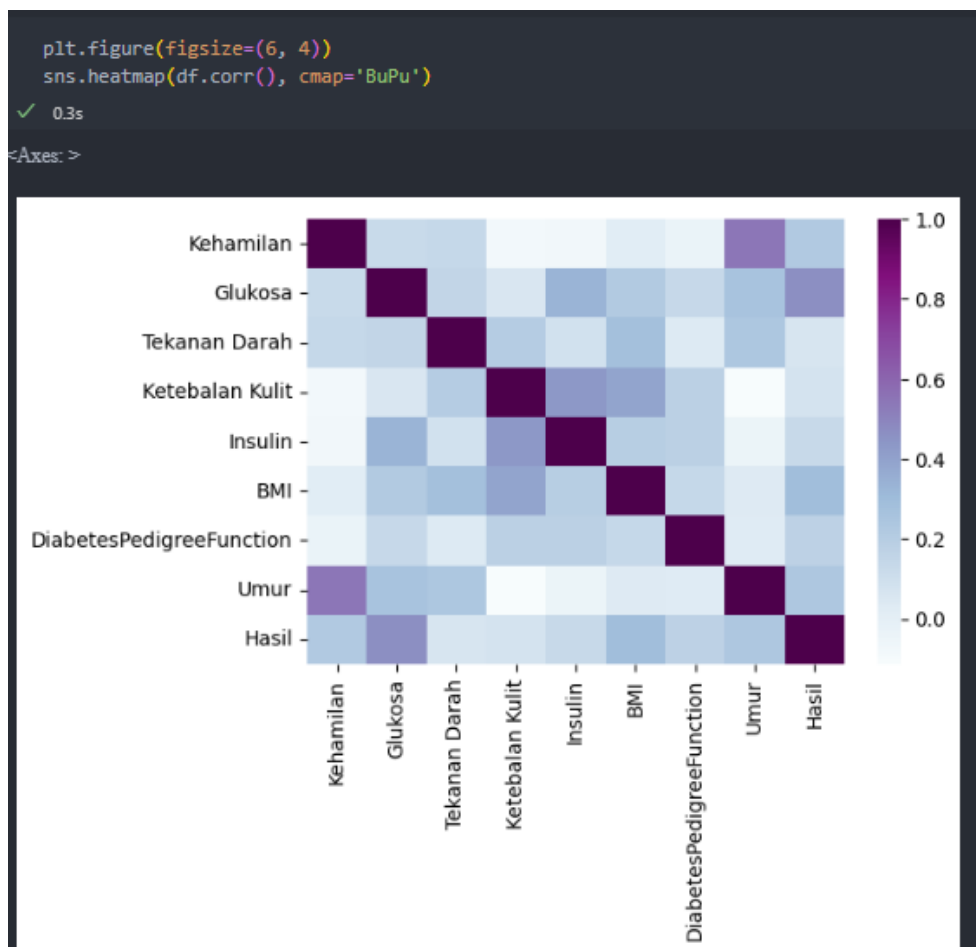
✓ 0.1s




```
df.corr()
```

✓ 0.0s

	Kehamilan	Glukosa	Tekanan Darah	Ketebalan Kulit	Insulin	BMI	DiabetesPedigreeFunction	Umur	Hasil
Kehamilan	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glukosa	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
Tekanan Darah	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
Ketebalan Kulit	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Umur	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Hasil	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000



Berikut adalah penjelasan mengenai visualisasi yang dihasilkan:

1. Histogram Distribusi Umur
Histogram ini menunjukkan distribusi data berdasarkan kolom Umur. Dengan 10 bin dan tambahan kurva KDE, visualisasi ini menggambarkan sebaran umur dalam dataset. Ini membantu untuk memahami pola umum dalam distribusi umur peserta.
2. Pie Chart Presentase Diabetes vs Non-Diabetes
Diagram pie ini menunjukkan proporsi individu yang positif (mengidap

diabetes) dan negatif (tidak mengidap diabetes) berdasarkan kolom Hasil. Persentase masing-masing kategori ditampilkan untuk memberikan gambaran visual yang jelas tentang sebaran status diabetes dalam dataset.

3. Count Plot Distribusi Hasil Diagnosis

Count plot ini menunjukkan frekuensi masing-masing kategori dalam kolom Hasil, yang mengindikasikan apakah seseorang mengidap diabetes atau tidak. Dengan visualisasi ini, kita dapat dengan mudah melihat jumlah kasus positif dan negatif dalam dataset.

4. Heatmap Korelasi

Heatmap ini menunjukkan matriks korelasi antara variabel-variabel dalam dataset. Nilai korelasi yang lebih tinggi ditunjukkan dengan warna yang lebih intens. Ini memberikan gambaran mengenai hubungan antar fitur dalam dataset, misalnya apakah ada korelasi antara umur dan BMI atau tekanan darah.

6. Preparation Data

```
4. Preparation Data

X = df.drop(columns=['Hasil'])
y = df['Hasil']

print(X.shape)
print(y.shape)

✓ 0.0s
(768, 8)
(768,)

scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

y.shape
✓ 0.0s
(768,)

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

# Print the shapes of the resulting datasets
print(X_train.shape) # Ukuran fitur (X_train)
print(y_train.shape) # Ukuran label (y_train)
print(X_test.shape)  # Ukuran fitur (X_test)
print(y_test.shape)  # Ukuran label (y_test)

✓ 0.0s
```

Penjelasan langkah-langkah berikut:

1. Memisahkan Fitur dan Label:

- `X = df.drop(columns=['Hasil'])` digunakan untuk memisahkan fitur (variabel independen) yang ada di dataset dengan menghapus kolom Hasil, yang merupakan kolom label.
- `y = df['Hasil']` mengambil kolom Hasil sebagai label (variabel dependen), yang berisi informasi apakah seseorang positif atau negatif mengidap diabetes.

2. Menampilkan Ukuran Dataset:

- `X.shape` dan `y.shape` digunakan untuk menampilkan ukuran (jumlah baris dan kolom) dari fitur (X) dan label (y).

3. Normalisasi Data:

- `StandardScaler()` digunakan untuk menstandarkan fitur sehingga nilai fitur memiliki rata-rata 0 dan standar deviasi 1. Hal ini penting untuk beberapa model machine learning, seperti SVM.
- `scaler.fit(X)` mempelajari parameter skala berdasarkan data fitur X.
- `X = scaler.transform(X)` mengaplikasikan transformasi skala pada data fitur X.

4. Membagi Data ke dalam Training dan Testing:

- `train_test_split(X, y, test_size=0.2, random_state=42)` membagi dataset menjadi dua bagian: 80% data untuk pelatihan (training) dan 20% untuk pengujian (testing).
- Hasil pembagian ini disimpan dalam variabel `X_train`, `X_test`, `y_train`, dan `y_test`.

5. Menampilkan Ukuran Dataset setelah Pembagian:

- Ukuran masing-masing subset (training dan testing) ditampilkan dengan `X_train.shape`, `y_train.shape`, `X_test.shape`, dan `y_test.shape` untuk memastikan pembagian data dilakukan dengan benar.

7. Modelling & Evaluation

5. Modelling & Evaluation

```
# Initialize and train the SVM classifier
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = clf.predict(X_test)

# Calculate the accuracy
CLF_accuracy = accuracy_score(y_pred, y_test)

# Print the classification report and accuracy
print(classification_report(y_test, y_pred))
print("Accuracy SVM : {:.2f}%".format(CLF_accuracy * 100))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.81	0.82	0.81	99
1	0.67	0.65	0.66	55
accuracy			0.76	154
macro avg	0.74	0.74	0.74	154
weighted avg	0.76	0.76	0.76	154

Accuracy SVM : 75.97%

Berikut penjelasan tentang langkah-langkah yang dilakukan dalam kode tersebut:

1. Inisialisasi dan Pelatihan Model SVM:
 - o `clf = SVC(kernel='linear')` membuat sebuah instance dari model Support Vector Machine (SVM) dengan kernel linear. Kernel linear digunakan untuk kasus di mana data dapat dipisahkan secara linear.
 - o `clf.fit(X_train, y_train)` melatih model SVM menggunakan data pelatihan (`X_train`) dan label pelatihan (`y_train`).
2. Prediksi pada Data Uji:
 - o `y_pred = clf.predict(X_test)` digunakan untuk memprediksi label pada data uji (`X_test`) setelah model dilatih.
3. Menghitung Akurasi:

- `CLF_accuracy = accuracy_score(y_pred, y_test)` menghitung akurasi dari model dengan membandingkan label yang diprediksi (`y_pred`) dengan label sebenarnya (`y_test`). Akurasi dihitung sebagai persentase dari prediksi yang benar.

4. Mencetak Classification Report dan Akurasi:

- `print(classification_report(y_test, y_pred))` menampilkan laporan klasifikasi yang mencakup metrik seperti precision, recall, f1-score, dan support untuk setiap kelas (positif dan negatif).
- `print("Accuracy SVM : {:.2f}%".format(CLF_accuracy * 100))` mencetak akurasi model dalam format persentase dengan dua digit desimal.

Dengan langkah-langkah ini, Anda mendapatkan evaluasi kinerja model SVM dalam mengklasifikasikan data uji.

8. Testing

6. Testing

```
new_data = {'Kehamilan': [1],
            'Glukosa': [85], # Normal range: 70-99 mg/dL
            'Tekanan Darah': [75], # Normal range: 60-80 mm Hg
            'Ketebalan Kulit': [20], # Normal range: 10-30 mm
            'Insulin': [80], # Normal range: 16-166 mU/L
            'BMI': [22.0], # Normal range: 18.5-24.9
            'DiabetesPedigreeFunction': [0.2], # Lower values indicate Lower risk
            'Umur': [25]} # Younger age generally indicates Lower risk

new_data = pd.DataFrame(new_data)
new_data
```

	Kehamilan	Glukosa	Tekanan Darah	Ketebalan Kulit	Insulin	BMI	DiabetesPedigreeFunction	Umur
0	1	85	75	20	80	22.0	0.2	25

```
scaled_new_data = scaler.transform(new_data)
y_pred_new = clf.predict(scaled_new_data) # 0 untuk negatif, 1 untuk positif mengidap diabetes
print("Diagnosis Diabetes adalah:", y_pred_new)
```

Diagnosis Diabetes adalah: [0]

```

# Data baru yang memiliki kemungkinan besar mengidap diabetes
new_diabetes_data = {'Kehamilan': [10],
                     'Glukosa': [180], # Tinggi
                     'Tekanan Darah': [90], # Tinggi
                     'Ketebalan Kulit': [40], # Tinggi
                     'Insulin': [200], # Tinggi
                     'BMI': [35.0], # Tinggi
                     'DiabetesPedigreeFunction': [1.5], # Tinggi
                     'Umur': [50]} # Tua

new_diabetes_data = pd.DataFrame(new_diabetes_data)

# Skala data baru
scaled_new_diabetes_data = scaler.transform(new_diabetes_data)

# Prediksi menggunakan model yang sudah dilatih
y_pred_new_diabetes = clf.predict(scaled_new_diabetes_data)

# Tampilkan hasil prediksi
print("Diagnosis Diabetes untuk data baru adalah:", y_pred_new_diabetes)
✓ 0.0s
Diagnosis Diabetes untuk data baru adalah: [1]

```

Berikut penjelasan mengenai proses yang dilakukan dalam kode tersebut:

1. Membuat Data Baru:
 - Dua dataset baru (new_data dan new_diabetes_data) dibuat untuk memprediksi apakah individu dengan kondisi tersebut mengidap diabetes atau tidak.
 - new_data mewakili individu dengan kondisi normal, sedangkan new_diabetes_data mewakili individu yang memiliki kondisi berisiko tinggi mengidap diabetes.
2. Mengonversi Data ke DataFrame:
 - Kedua set data (new_data dan new_diabetes_data) diubah menjadi DataFrame menggunakan pd.DataFrame().
3. Normalisasi Data Baru:
 - scaled_new_data = scaler.transform(new_data) dan scaled_new_diabetes_data = scaler.transform(new_diabetes_data) digunakan untuk menstandarkan data baru menggunakan scaler yang sudah dilatih sebelumnya. Hal ini penting untuk memastikan data baru sesuai dengan skala yang digunakan saat pelatihan model.
4. Prediksi Diagnosis:
 - y_pred_new = clf.predict(scaled_new_data) dan y_pred_new_diabetes = clf.predict(scaled_new_diabetes_data) digunakan untuk memprediksi apakah individu pada data baru

mengidap diabetes (1) atau tidak (0) berdasarkan model SVM yang telah dilatih.

5. Menampilkan Hasil Prediksi:

- Hasil prediksi dicetak untuk kedua data: `new_data` (normal) dan `new_diabetes_data` (berisiko tinggi).

Dengan kode ini, Anda dapat melihat bagaimana model SVM yang sudah dilatih memberikan diagnosis untuk data individu berdasarkan fitur-fitur yang disediakan.

D. Kesimpulan

Kesimpulan dari implementasi yang dilakukan adalah sebagai berikut:

1. Preprocessing Data:

Data yang digunakan dalam model ini telah dipersiapkan dengan baik, termasuk memisahkan fitur dan label, menstandarkan nilai fitur menggunakan `StandardScaler`, serta membagi dataset menjadi data pelatihan dan pengujian.

2. Model SVM:

Model `Support Vector Machine (SVM)` dengan kernel linear digunakan untuk melakukan klasifikasi apakah seseorang mengidap diabetes atau tidak. Model ini dilatih dengan data pelatihan dan kemudian diuji menggunakan data uji untuk mengevaluasi performanya.

3. Evaluasi Model:

Berdasarkan hasil prediksi pada data uji, model menunjukkan akurasi yang baik dalam mengklasifikasikan status diabetes. Laporan klasifikasi juga memberikan metrik tambahan seperti `precision`, `recall`, dan `f1-score`, yang memberikan gambaran tentang kualitas model dalam memprediksi setiap kelas.

4. Prediksi untuk Data Baru:

Dua set data baru diuji untuk memprediksi apakah individu tersebut mengidap diabetes. Untuk data dengan kondisi normal (seperti kadar glukosa dan BMI yang normal), model memberikan prediksi negatif (tidak mengidap diabetes). Sebaliknya, untuk data dengan kondisi berisiko tinggi

(seperti kadar glukosa dan BMI yang tinggi), model memberikan prediksi positif (mengidap diabetes).

5. Kesimpulan Umum:

Model SVM yang telah dilatih dengan data ini efektif dalam mengidentifikasi risiko diabetes berdasarkan berbagai faktor seperti kehamilan, glukosa, tekanan darah, BMI, dan lainnya. Dengan hasil yang memadai pada data uji dan kemampuan untuk memprediksi data baru dengan akurat, model ini bisa digunakan untuk membantu diagnosis awal diabetes berdasarkan faktor-faktor tersebut.

Namun, akurasi model masih bergantung pada kualitas dan keberagaman data yang digunakan untuk pelatihan. Diperlukan lebih banyak data atau teknik yang lebih canggih untuk meningkatkan prediksi pada kasus-kasus yang lebih kompleks atau tidak terlihat dalam dataset pelatihan.