Nama : Ulfah Haanah

NIM: 1227030036

Tugas Modul 11 Support Vector Machine

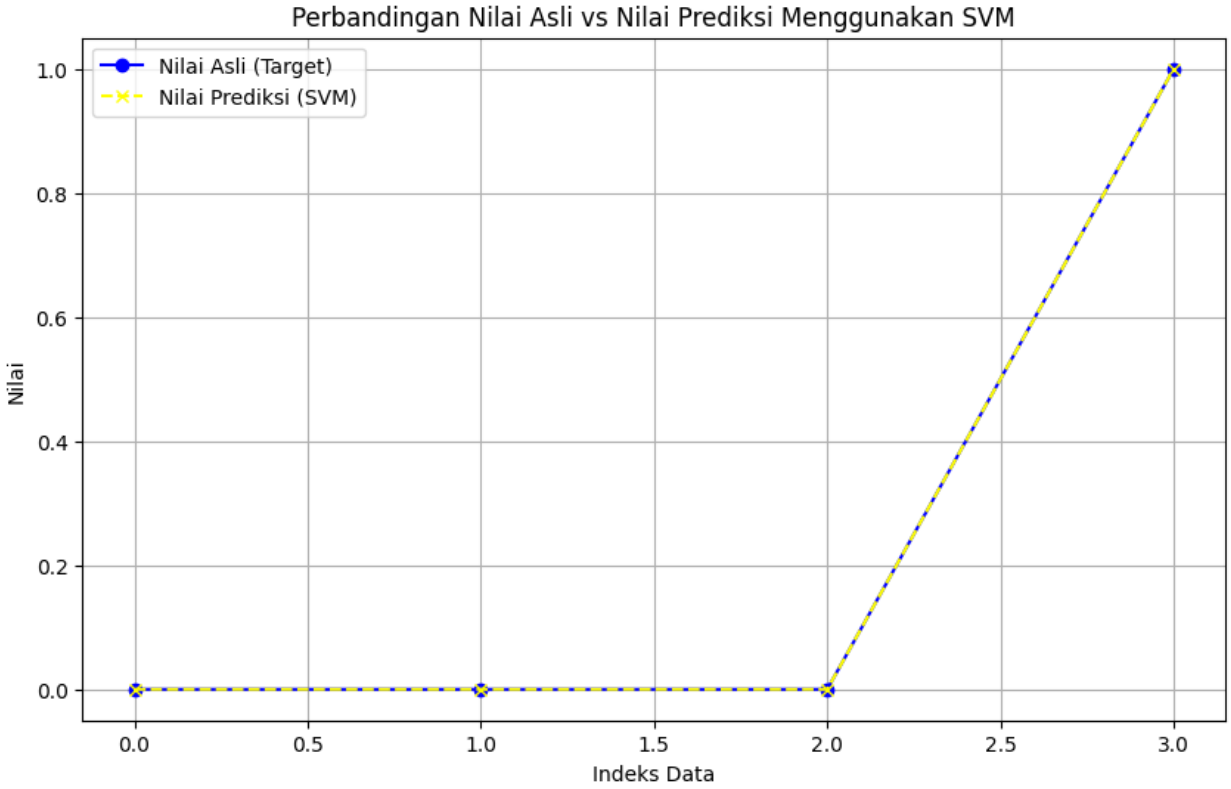
```
from sklearn import svm

#Database : Gerbang Logika AND
#X=Data, y=target
x = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 0, 0, 1]

#training and classify
clf = svm.SVC()
clf.fit(x,y)

#Prediksi
print ("Logika AND Metode Support Vector Machine (SVM) ")
print ("Logika = Prediksi")
print ("0 0 = ",clf.predict([[0, 0]]))
print ("0 1 = ",clf.predict([[0, 1]]))
print ("1 0 = ",clf.predict([[1, 0]]))
print ("1 1 = ",clf.predict([[1, 1]]))
```

```
Logika AND Metode Support Vector Machine (SVM)
Logika = Prediksi
0 0 =  [0]
0 1 =  [0]
1 0 =  [0]
1 1 =  [1]
```

```
# Import library yang diperlukan
import numpy as np
import pandas as pd
from sklearn import svm
from google.colab import drive
import matplotlib.pyplot as plt

# Mount Google Drive
drive.mount("/content/drive")

# Path ke file Database.txt di Google Drive
file_path = "/content/drive/My Drive/svm.txt"  # Ganti dengan path sesuai lokasi file Anda di Google Drive

# Membaca data dari file
Database = pd.read_csv(file_path, sep=",", header=0)  # Sesuaikan dengan struktur file Anda

# Data (X) dan target (y)
X = Database[['a', 'b']]  # Pastikan kolom sesuai dengan nama yang ada di file
y = Database["Target"]

# Membuat dan melatih model SVM
clf = svm.SVC()
clf.fit(X.values, y)

# Melakukan prediksi
y_pred = clf.predict(X.values)
```

```python
# Menampilkan hasil prediksi
print("Hasil prediksi:")
for i, pred in enumerate(y_pred):
    print(f"({X.iloc[i, 0]}), ({X.iloc[i, 1]}), ({pred})")

# Membuat plot perbandingan nilai asli dengan nilai prediksi
plt.figure(figsize=(10, 6))
plt.plot(range(len(y)), y, 'o-', label='Nilai Asli (Target)', color='blue')
plt.plot(range(len(y_pred)), y_pred, 'x--', label='Nilai Prediksi (SVM)', color='yellow')

# Menambahkan label dan judul
plt.xlabel('Indeks Data')
plt.ylabel('Nilai')
plt.title("Perbandingan Nilai Asli vs Nilai Prediksi Menggunakan SVM")
plt.legend()
plt.grid()
plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Hasil prediksi:
(0), (0), (0)
(0), (1), (0)
(1), (0), (0)
(1), (1), (1)
```



Ketika batas a dan b nya adalah a = i+1 dan b = i+2!

```
def Trapezoid(a,b,f):
  n = 100
  def trapezoid(f,a,b,n=1000):
    h = (b-a)/n
    sum = 0.0
    for i in range (1,n):
      x = a+i*h
      sum = sum + f(x)
    integral = (h/2)*(f(a)+2*sum+f(b))
    return integral
  integral = trapezoid(f,a,b,n)
  print(a,",",b,",",round(integral,2))

# Melakukan looping untuk membuat database dari beberapa soal integral
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 2*x)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 2*x + 2)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 2*x + 4)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 4*x + 6)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 6*x + 8)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 8*x + 10)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 10*x + 12)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 12*x + 14)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 14*x + 12)
for i in range(0,5):
  Trapezoid(i+1,i+2,lambda x: 20*x + 40)
```

```
1 , 2 , 3.0
2 , 3 , 5.0
3 , 4 , 7.0
4 , 5 , 9.0
5 , 6 , 11.0
1 , 2 , 5.0
2 , 3 , 7.0
3 , 4 , 9.0
4 , 5 , 11.0
5 , 6 , 13.0
1 , 2 , 7.0
2 , 3 , 9.0
3 , 4 , 11.0
4 , 5 , 13.0
5 , 6 , 15.0
1 , 2 , 12.0
2 , 3 , 16.0
3 , 4 , 20.0
4 , 5 , 24.0
5 , 6 , 28.0
1 , 2 , 17.0
2 , 3 , 23.0
3 , 4 , 29.0
4 , 5 , 35.0
5 , 6 , 41.0
1 , 2 , 22.0
2 , 3 , 30.0
3 , 4 , 38.0
4 , 5 , 46.0
5 , 6 , 54.0
1 , 2 , 27.0
2 , 3 , 37.0
3 , 4 , 47.0
4 , 5 , 57.0
5 , 6 , 67.0
```

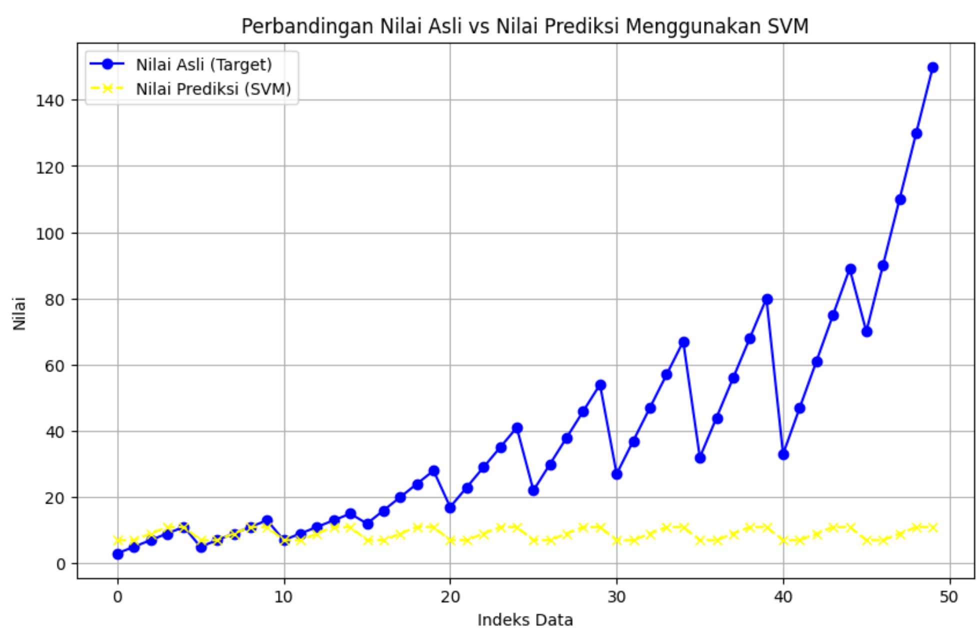
```
1 , 2 , 32.0
2 , 3 , 44.0
3 , 4 , 56.0
4 , 5 , 68.0
5 , 6 , 80.0
1 , 2 , 33.0
2 , 3 , 47.0
3 , 4 , 61.0
4 , 5 , 75.0
5 , 6 , 89.0
1 , 2 , 70.0
2 , 3 , 90.0
3 , 4 , 110.0
4 , 5 , 130.0
5 , 6 , 150.0
```

```python
# Import library yang diperlukan
import numpy as  np
import pandas as pd
from sklearn import svm
from google.colab import drive
import matplotlib.pyplot as plt

# Mount Google Drive
drive.mount('/content/drive')

# Path ke fie Database.txt di Google Drive
file_path = '/content/drive/My Drive/trapezoid_tugas ulfah.txt' #Ganti dengan path sesuai lokasi file Anda di Google Drive

# Membaca data dari file
Database = pd.read_csv(file_path, sep=",", header=0)  # Sesuaikan dengan struktur file Anda

# Data (X) dan target (y)
X = Database[['a', 'b']]  # Pastikan kolom sesuai dengan nama yang ada di file
y = Database["Target"]

# Membuat dan melatih model SVM
clf = svm.SVC()
clf.fit(X.values, y)

# Melakukan prediksi
y_pred = clf.predict(X.values)

# Menampilkan hasil prediksi
print("Hasil prediksi:")
for i, pred in enumerate(y_pred):
    print(f"({X.iloc[i, 0]}), ({X.iloc[i, 1]}), ({pred})")
```

```python
# Membuat plot perbandingan nilai asli dengan nilai prediksi
plt.figure(figsize=(10, 6))
plt.plot(range(len(y)), y, 'o-', label='Nilai Asli (Target)', color='blue')
plt.plot(range(len(y_pred)), y_pred, 'x--', label='Nilai Prediksi (SVM)', color='yellow')

# Menambahkan label dan judul
plt.xlabel('Indeks Data')
plt.ylabel('Nilai')
plt.title("Perbandingan Nilai Asli vs Nilai Prediksi Menggunakan SVM")
plt.legend()
plt.grid()
plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
Hasil prediksi:
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
```

```
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
(1), (2), (7.0)
(2), (3), (7.0)
(3), (4), (9.0)
(4), (5), (11.0)
(5), (6), (11.0)
```



Ketika batas a dan b nya adalah a = i+2 dan b = i+4!

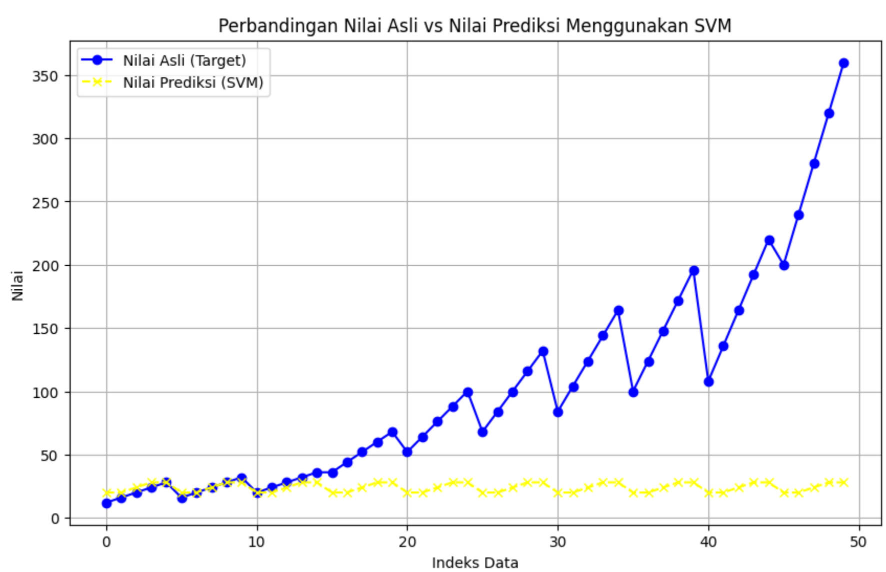
```
# Melakukan looping untuk membuat database dari beberapa soal integral
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 2*x)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 2*x + 2)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 2*x + 4)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 4*x + 6)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 6*x + 8)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 8*x + 10)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 10*x + 12)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 12*x + 14)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 14*x + 12)
for i in range(0,5):
  Trapezoid(i+2,i+4,lambda x: 20*x + 40)
```

```
2 , 4 , 12.0
3 , 5 , 16.0
4 , 6 , 20.0
5 , 7 , 24.0
6 , 8 , 28.0
2 , 4 , 16.0
3 , 5 , 20.0
4 , 6 , 24.0
5 , 7 , 28.0
6 , 8 , 32.0
2 , 4 , 20.0
3 , 5 , 24.0
4 , 6 , 28.0
5 , 7 , 32.0
6 , 8 , 36.0
2 , 4 , 36.0
3 , 5 , 44.0
4 , 6 , 52.0
5 , 7 , 60.0
6 , 8 , 68.0
2 , 4 , 52.0
3 , 5 , 64.0
4 , 6 , 76.0
5 , 7 , 88.0
6 , 8 , 100.0
2 , 4 , 68.0
3 , 5 , 84.0
4 , 6 , 100.0
5 , 7 , 116.0
6 , 8 , 132.0
2 , 4 , 84.0
3 , 5 , 104.0
4 , 6 , 124.0
5 , 7 , 144.0
6 , 8 , 164.0
2 , 4 , 100.0
3 , 5 , 124.0
4 , 6 , 148.0
5 , 7 , 172.0
6 , 8 , 196.0
2 , 4 , 108.0
3 , 5 , 136.0
4 , 6 , 164.0
5 , 7 , 192.0
6 , 8 , 220.0
2 , 4 , 200.0
3 , 5 , 240.0
4 , 6 , 280.0
5 , 7 , 320.0
6 , 8 , 360.0
```

Dan hasil prediksinya adalah tercantum dibawah ini:

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Hasil prediksi:
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
(2), (4), (20.0)
(3), (5), (20.0)
(4), (6), (24.0)
(5), (7), (28.0)
(6), (8), (28.0)
```

To exit full screen, press



- Penjelasan Algoritma

Algoritma program ini menggabungkan dua konsep utama: perhitungan integral menggunakan metode trapezoid dan prediksi nilai integral menggunakan model Support Vector Machine (SVM). Proses ini dirancang untuk menghitung nilai integral secara numerik terlebih dahulu, lalu melatih model pembelajaran mesin agar mampu memprediksi nilai integral berdasarkan pola data yang telah dipelajari. Pendekatan ini mengilustrasikan kombinasi antara metode numerik klasik dan teknologi pembelajaran mesin modern untuk mencapai efisiensi dan fleksibilitas dalam perhitungan integral.

Langkah-Langkah Utama dalam Algoritma

1. Perhitungan Integral Menggunakan Metode Trapezoid

Proses perhitungan dimulai dengan menggunakan metode trapezoid, sebuah pendekatan numerik untuk menghitung integral dari sebuah fungsi kontinu. Metode ini bekerja dengan membagi interval integrasi, yaitu antara batas bawah (a) dan batas atas (b), menjadi beberapa segmen kecil yang berukuran sama. Setiap segmen dianggap sebagai trapesium, dan luas masing-masing trapesium dihitung menggunakan rumus:

$$\text{Luas} = 1/2 \cdot (f(a) + f(b)) \cdot (b - a)$$

Di mana:

- $f(a)$ adalah nilai fungsi pada titik awal segmen,
- $f(b)$ adalah nilai fungsi pada titik akhir segmen,
- $(b - a)$ adalah panjang segmen.

Proses ini diulang untuk beberapa fungsi, seperti $2x+2$, $4x+6$, dan seterusnya, dengan batas integrasi yang bervariasi, misalnya dari $a=2, b=4$ hingga $a=6, b=8$. Hasil perhitungan integral ini digunakan untuk membentuk dataset yang mencakup pasangan input berupa batas bawah dan atas, serta output berupa nilai integral yang dihitung. Metode ini cukup efisien untuk menghitung nilai integral dari berbagai fungsi matematis dengan tingkat akurasi yang memadai, terutama jika segmen yang digunakan cukup kecil.

Hasil dari perhitungan integral untuk beberapa fungsi linear dan interval yang berbeda dihitung menggunakan program. Misalnya, untuk fungsi f(x)=2x dengan batas integrasi 2 hingga 4, nilai integral yang dihitung adalah 12.0. Pendekatan yang sama diterapkan untuk fungsi-fungsi lainnya, seperti 2x+2, 4x+6, dan sebagainya.

2. Pelatihan Model SVM untuk Prediksi Integral

Setelah nilai integral dihitung, algoritma melanjutkan dengan memanfaatkan Support Vector Machine (SVM) untuk memprediksi nilai integral berdasarkan batas integrasi. Proses ini melibatkan langkah-langkah berikut:

1. **Persiapan Dataset**: Dataset dibentuk dari hasil perhitungan integral yang telah dilakukan sebelumnya. Dataset terdiri dari:
    - **Input**: Pasangan batas bawah (a) dan batas atas (b).
    - **Output**: Nilai integral yang dihitung menggunakan metode trapezoid.
2. **Pelatihan Model SVM**: Model SVM diinisialisasi menggunakan library scikit-learn. Model ini dilatih dengan dataset input (a, b) dan output (nilai integral). Selama proses pelatihan, model SVM mencoba mempelajari pola hubungan antara input dan output untuk memprediksi nilai integral.
3. **Prediksi Nilai Integral**: Setelah pelatihan selesai, model SVM digunakan untuk memprediksi nilai integral berdasarkan pasangan batas integrasi baru yang belum pernah dilihat sebelumnya. Hasil prediksi ini dibandingkan dengan nilai asli yang dihitung menggunakan metode trapezoid untuk mengevaluasi akurasi model. Perbandingan dilakukan dengan menghitung tingkat kesalahan antara nilai prediksi dan nilai asli.
4. **Evaluasi Model**: Hasil prediksi dibandingkan dengan nilai asli integral yang dihitung menggunakan metode trapezoid. Selisih antara prediksi dan nilai asli menunjukkan seberapa akurat model dalam menangkap pola data.. Dalam implementasi ini, hasil prediksi SVM menunjukkan bahwa model tidak mampu menangkap variasi nilai asli integral dengan baik. Sebagai contoh, untuk beberapa kombinasi batas integrasi, hasil asli dari metode trapezoid adalah 12.0, 16.0, atau 20.0, tetapi model SVM cenderung memprediksi

nilai tetap, seperti 20.0 atau 28.0 Hal ini menunjukkan bahwa model mengalami *underfitting,* yaitu kondisi di mana model tidak mampu mempelajari pola data dengan baik.

5. **Visualisasi Hasil**: Untuk analisis lebih lanjut, algoritma membuat grafik perbandingan antara nilai asli integral (hasil metode trapezoid) dan nilai prediksi model SVM. Grafik ini membantu mengevaluasi performa model dalam mengikuti pola data.

Algoritma ini menunjukkan kombinasi menarik antara metode numerik dan pembelajaran mesin. Metode trapezoid memberikan hasil integral yang akurat secara numerik, sementara model SVM berpotensi untuk memprediksi nilai integral secara otomatis dengan kecepatan yang lebih tinggi.

- **Penjelasan analisis grafik**

Selain itu, hasil dari kode program tersebut divisualisasikan dalam bentuk grafik, di mana nilai asli dan prediksi ditampilkan untuk mempermudah analisis. Grafik yang ditampilkan memperlihatkan perbandingan antara nilai asli (target) dan nilai prediksi yang dihasilkan oleh model SVM berdasarkan data yang telah dilatih. Garis prediksi tersebut tidak mengikuti pola data asli (target) dalam grafik menunjukkan bahwa model SVM yang digunakan tidak mampu menangkap pola data secara optimal. Grafik ini memperlihatkan perbedaan yang mencolok antara nilai asli (ditampilkan dengan garis biru) dan nilai prediksi (ditampilkan dengan garis kuning), terutama pada indeks data yang lebih tinggi, di mana nilai asli meningkat secara signifikan tetapi nilai prediksi tetap stabil dalam rentang yang sempit. Ketidaksesuaian ini mengindikasikan bahwa model SVM gagal merepresentasikan hubungan antara input dan output dengan baik

Kemungkinan besar, pola pada data target cukup kompleks atau menunjukkan fluktuasi yang tajam, sementara model SVM yang digunakan terlalu sederhana untuk menangkap kerumitan ini. Kernel default pada SVM biasanya bersifat linear, yang tidak cocok jika data memiliki pola non-linear atau hubungan yang lebih rumit. Akibatnya, prediksi yang dihasilkan cenderung rata atau stabil, sehingga tidak mencerminkan perubahan nilai asli.

Selain itu, model SVM ini kemungkinan mengalami *underfitting*, yaitu kondisi di mana model tidak mampu mempelajari pola dengan baik karena terlalu sederhana atau kurang fleksibel. Hal ini dapat terjadi jika parameter-parameter model, seperti kernel, nilai regularisasi (C), atau gamma, tidak diatur dengan baik untuk menangani data dengan pola yang kompleks. Misalnya,

parameter C yang terlalu kecil dapat membuat model terlalu kaku, sehingga tidak mampu mempelajari pola outlier atau fluktuasi besar pada data target.

Faktor lain yang mungkin memengaruhi adalah kualitas data itu sendiri. Jika data yang digunakan untuk melatih model tidak mencukupi atau tidak cukup mewakili variasi pola target, maka model akan kesulitan memahami hubungan antara input dan output. Selain itu, jika data input belum dinormalisasi atau di-standardisasi, perbedaan skala pada nilai input dapat memengaruhi performa model, mengingat SVM sangat sensitif terhadap skala data.

Untuk mengatasi permasalahan ini, beberapa langkah perbaikan dapat dilakukan. Pertama, menggunakan kernel non-linear seperti RBF atau Polynomial yang lebih fleksibel untuk menangkap pola data yang kompleks. Kedua, melakukan *hyperparameter tuning* untuk menemukan kombinasi parameter terbaik, seperti nilai C dan gamma, yang sesuai dengan dataset. Ketiga, memastikan bahwa data input telah dinormalisasi atau di-standardisasi sehingga memiliki skala yang seragam. Keempat, menambah jumlah data yang lebih representatif atau meningkatkan kualitas data melalui proses *data augmentation*.

Kesimpulannya, grafik ini menunjukkan bahwa model SVM yang digunakan masih kurang mampu menangani pola data yang kompleks. Dengan melakukan penyesuaian pada model, baik melalui pemilihan kernel yang lebih sesuai, tuning parameter, maupun pengolahan data yang lebih baik, model diharapkan dapat menghasilkan prediksi yang lebih akurat dan mendekati nilai target.