LAMPIRAN

LAMPIRAN I Perhitungan Matriks

misal $\theta = 2$

- Data ke-1
  - K (x_1, y_1) = $e\left(\frac{-\|x_1-y_1\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

    = 1
  - K (x_2, y_1) = $e\left(\frac{-\|x_2-y_1\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

    = 1
  - K (x_3, y_1) = $e\left(\frac{-\|x_3-y_1\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

    = 1
  - K (x_4, y_1) = $e\left(\frac{-\|x_4-y_1\|^2}{2\theta^2}\right)$

    K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

    = 0.6
  - K (x_5, y_1) = $e\left(\frac{-\|x_5-y_1\|^2}{2\theta^2}\right)$

    K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

    = 0.6
  - K (x_6, y_1) = $e\left(\frac{-\|x_6-y_1\|^2}{2\theta^2}\right)$

$$K(-1, 1) = e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$$

$$= 0.6$$

- $K(x_7, y_1) = e\left(\frac{-\|x_7-y_1\|^2}{2\theta^2}\right)$

$$K(1, 1) = e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$$

$$= 1$$

- $K(x_7, y_1) = e\left(\frac{-\|x_7-y_1\|^2}{2\theta^2}\right)$

$$K(1, 1) = e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$$

$$= 1$$

- Data ke-2
  - $K(x_1, y_2) = e\left(\frac{-\|x_2-y_2\|^2}{2\theta^2}\right)$

$$K(1, 0) = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

  - $K(x_2, y_2) = e\left(\frac{-\|x_2-y_2\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

  - $K(x_3, y_2) = e\left(\frac{-\|x_3-y_2\|^2}{2\theta^2}\right)$

$$K(1, 0) = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- K (x_4, y_2) = $e\left(\frac{-\|x_4-y_2\|^2}{2\theta^2}\right)$

  K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_5, y_2) = $e\left(\frac{-\|x_5-y_2\|^2}{2\theta^2}\right)$

  K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_6, y_2) = $e\left(\frac{-\|x_6-y_2\|^2}{2\theta^2}\right)$

  K (1, 0) = $e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_7, y_2) = $e\left(\frac{-\|x_7-y_2\|^2}{2\theta^2}\right)$

  K (1, 0) = $e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_8, y_2) = $e\left(\frac{-\|x_8-y_2\|^2}{2\theta^2}\right)$

  K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

  = 0.8

- Data ke-3
  - K (x_1, y_3) = $e\left(\frac{-\|x_1-y_3\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

    = 1
  - K (x_2, y_3) = $e\left(\frac{-\|x_2-y_3\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

= 1

- K (x_3, y_3) = $e\left(\frac{-\|x_3-y_3\|^2}{2\theta^2}\right)$

  K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

  = 1

- K (x_4, y_3) = $e\left(\frac{-\|x_4-y_3\|^2}{2\theta^2}\right)$

  K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_5, y_3) = $e\left(\frac{-\|x_5-y_3\|^2}{2\theta^2}\right)$

  K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_6, y_3) = $e\left(\frac{-\|x_6-y_3\|^2}{2\theta^2}\right)$

  K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_7, y_3) = $e\left(\frac{-\|x_7-y_3\|^2}{2\theta^2}\right)$

  K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

  = 1

- K (x_8, y_3) = $e\left(\frac{-\|x_8-y_3\|^2}{2\theta^2}\right)$

  K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

  = 0.6

- Data ke-4
  - K (x_1, y_4) = $e\left(\frac{-\|x_1-y_4\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_2, y_4) = e\left(\frac{-\|x_2-y_4\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_3, y_4) = e\left(\frac{-\|x_3-y_4\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_4, y_4) = e\left(\frac{-\|x_4-y_4\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_5, y_4) = e\left(\frac{-\|x_5-y_4\|^2}{2\theta^2}\right)$

$$K(1, 0) = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_6, y_4) = e\left(\frac{-\|x_6-y_4\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_7, y_4) = e\left(\frac{-\|x_7-y_4\|^2}{2\theta^2}\right)$

$$K(1, 0) = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K(x_8, y_4) = e\left(\frac{-\|x_8-y_4\|^2}{2\theta^2}\right)$

$$K(-1, 0) = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- Data ke-5
  - $K(x_1, y_5) = e\left(\frac{-\|x_1-y_5\|^2}{2\theta^2}\right)$

    $$K(1, 1) = e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$$

    $$= 1$$
  - $K(x_2, y_5) = e\left(\frac{-\|x_2-y_5\|^2}{2\theta^2}\right)$

    $$K(1, 1) = e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$$

    $$= 1$$
  - $K(x_3, y_5) = e\left(\frac{-\|x_3-y_5\|^2}{2\theta^2}\right)$

    $$K(1, 1) = e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$$

    $$= 1$$
  - $K(x_4, y_5) = e\left(\frac{-\|x_5-y_5\|^2}{2\theta^2}\right)$

    $$K(-1, 1) = e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$$

    $$= 0.6$$
  - $K(x_5, y_5) = e\left(\frac{-\|x_5-y_5\|^2}{2\theta^2}\right)$

    $$K(-1, 1) = e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$$

    $$= 0.6$$
  - $K(x_6, y_5) = e\left(\frac{-\|x_6-y_5\|^2}{2\theta^2}\right)$

    $$K(1, 1) = e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$$

    $$= 1$$

- K (x_7, y_5) = $e\left(\frac{-\|x_7-y_5\|^2}{2\theta^2}\right)$

  K (-1, 1) = $e\left(\frac{-\|-1-1\|^2}{2(2)^2}\right)$

  = 0.6
- K (x_8, y_5) = $e\left(\frac{-\|x_8-y_5\|^2}{2\theta^2}\right)$

  K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

  = 1

- Data ke-6
  - K (x_1, y_6) = $e\left(\frac{-\|x_1-y_6\|^2}{2\theta^2}\right)$

    K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

    = 0.8
  - K (x_2, y_6) = $e\left(\frac{-\|x_2-y_6\|^2}{2\theta^2}\right)$

    K (1, 0) = $e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$

    = 0.8
  - K (x_3, y_6) = $e\left(\frac{-\|x_3-y_6\|^2}{2\theta^2}\right)$

    K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

    = 0.8
  - K (x_4, y_6) = $e\left(\frac{-\|x_4-y_6\|^2}{2\theta^2}\right)$

    K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

    = 0.8
  - K (x_5, y_6) = $e\left(\frac{-\|x_5-y_6\|^2}{2\theta^2}\right)$

$$K\ (1, 0) \quad = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K\ (x_6, y_6) = e\left(\frac{-\|x_6-y_6\|^2}{2\theta^2}\right)$

$$K\ (-1, 0) \quad = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K\ (x_7, y_6) = e\left(\frac{-\|x_7-y_6\|^2}{2\theta^2}\right)$

$$K\ (-1, 0) \quad = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K\ (x_8, y_6) = e\left(\frac{-\|x_8-y_6\|^2}{2\theta^2}\right)$

$$K\ (-1, 0) \quad = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- Data ke-7
  - $K\ (x_1, y_7) = e\left(\frac{-\|x_1-y_7\|^2}{2\theta^2}\right)$

$$K\ (1, 0) \quad = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K\ (x_2, y_7) = e\left(\frac{-\|x_2-y_7\|^2}{2\theta^2}\right)$

$$K\ (-1, 0) \quad = e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- $K\ (x_3, y_7) = e\left(\frac{-\|x_3-y_7\|^2}{2\theta^2}\right)$

$$K\ (1, 0) \quad = e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$$

$$= 0.8$$

- K (x_4, y_7) = $e\left(\frac{-\|x_4-y_7\|^2}{2\theta^2}\right)$

  K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_5, y_7) = $e\left(\frac{-\|x_5-y_7\|^2}{2\theta^2}\right)$

  K (1, 0) = $e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_6, y_7) = $e\left(\frac{-\|x_6-y_7\|^2}{2\theta^2}\right)$

  K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_7, y_7) = $e\left(\frac{-\|x_7-y_7\|^2}{2\theta^2}\right)$

  K (-1, 0) = $e\left(\frac{-\|-1-0\|^2}{2(2)^2}\right)$

  = 0.8
- K (x_8, y_7) = $e\left(\frac{-\|x_8-y_7\|^2}{2\theta^2}\right)$

  K (1, 0) = $e\left(\frac{-\|1-0\|^2}{2(2)^2}\right)$

  = 0.8

- Data ke-8
  - K (x_1, y_8) = $e\left(\frac{-\|x_1-y_8\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

    = 1
  - K (x_2, y_8) = $e\left(\frac{-\|x_2-y_8\|^2}{2\theta^2}\right)$

    K (1, 1) = $e\left(\frac{-\|1-1\|^2}{2(2)^2}\right)$

= 1

- K (x_3, y_8) $= e\left(\frac{-\|x_3 - y_8\|^2}{2\theta^2}\right)$

  K (1, 1) $= e\left(\frac{-\|1 - 1\|^2}{2(2)^2}\right)$

  = 1

- K (x_4, y_8) $= e\left(\frac{-\|x_4 - y_8\|^2}{2\theta^2}\right)$

  K (-1, 1) $= e\left(\frac{-\|-1 - 1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_5, y_8) $= e\left(\frac{-\|x_5 - y_8\|^2}{2\theta^2}\right)$

  K (-1, 1) $= e\left(\frac{-\|-1 - 1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_6, y_8) $= e\left(\frac{-\|x_6 - y_8\|^2}{2\theta^2}\right)$

  K (-1, 1) $= e\left(\frac{-\|-1 - 1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_7, y_8) $= e\left(\frac{-\|x_7 - y_8\|^2}{2\theta^2}\right)$

  K (-1, 1) $= e\left(\frac{-\|-1 - 1\|^2}{2(2)^2}\right)$

  = 0.6

- K (x_8, y_8) $= e\left(\frac{-\|x_8 - y_8\|^2}{2\theta^2}\right)$

  K (-1, 1) $= e\left(\frac{-\|-1 - 1\|^2}{2(2)^2}\right)$

  = 0.6

LAMPIRAN II Source Code Secara Umum

INPUT DATA

In [376]:
```
import pandas as pd

data = pd.read_csv('diabetes.csv')
data
```

Out[376]:

	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
...	...	...	...	...	...	...	...	...	...
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

NORMALIZE

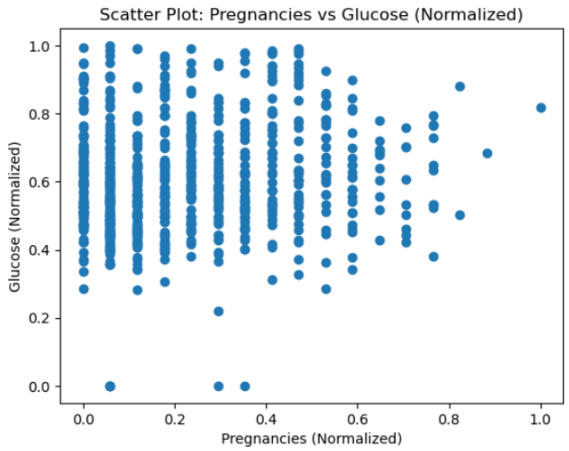
In [377]:
```
columns = data.columns

df = pd.DataFrame(data, columns=columns)

normalized_df = (df - df.min()) / (df.max() - df.min())

normalized_df['Outcome'] = normalized_df['Outcome'].replace(0, -1)

print("Data Asli:\n", df.head())
print("\nData yang sudah dinormalisasi Min-Max secara manual:\n", normalized_df)
```

```
Data Asli:
    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
```

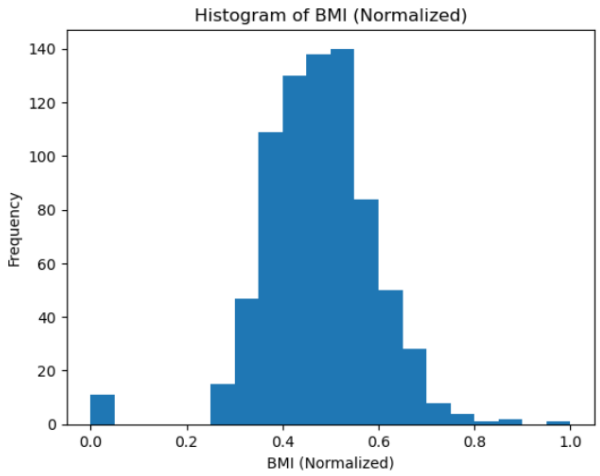
```
Data yang sudah dinormalisasi Min-Max secara manual:
      Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  \
0        0.352941  0.743719       0.590164       0.353535  0.000000  0.500745
1        0.058824  0.427136       0.540984       0.292929  0.000000  0.396423
2        0.470588  0.919598       0.524590       0.000000  0.000000  0.347243
3        0.058824  0.447236       0.540984       0.232323  0.111111  0.418778
4        0.000000  0.688442       0.327869       0.353535  0.198582  0.642325
..            ...       ...            ...            ...       ...       ...
763      0.588235  0.507538       0.622951       0.484848  0.212766  0.490313
764      0.117647  0.613065       0.573770       0.272727  0.000000  0.548435
765      0.294118  0.608040       0.590164       0.232323  0.132388  0.390462
766      0.058824  0.633166       0.491803       0.000000  0.000000  0.448584
767      0.058824  0.467337       0.573770       0.313131  0.000000  0.453055

     DiabetesPedigreeFunction       Age  Outcome
0                    0.234415  0.483333      1.0
1                    0.116567  0.166667     -1.0
2                    0.253629  0.183333      1.0
3                    0.038002  0.000000     -1.0
4                    0.943638  0.200000      1.0
..                        ...       ...      ...
763                  0.039710  0.700000     -1.0
764                  0.111870  0.100000     -1.0
765                  0.071307  0.150000     -1.0
766                  0.115713  0.433333      1.0
767                  0.101196  0.033333     -1.0

[768 rows x 9 columns]
```

VISUALIZATION DATA

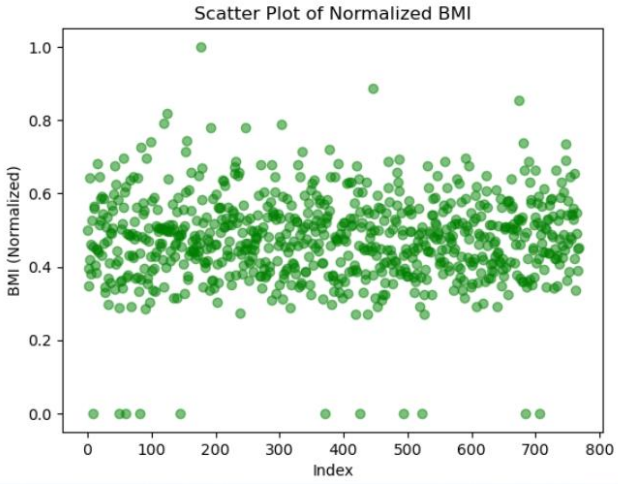
In [378]:
```
import matplotlib.pyplot as plt

# Scatter plot dari dua fitur yang dipilih (misalnya 'Pregnancies' dan 'Glucose')
plt.scatter(normalized_df['Pregnancies'], normalized_df['Glucose'])
plt.xlabel('Pregnancies (Normalized)')
plt.ylabel('Glucose (Normalized)')
plt.title('Scatter Plot: Pregnancies vs Glucose (Normalized)')
plt.show()
```



In [379]:
```
# Histogram dari fitur 'BMI'
plt.hist(normalized_df['BMI'], bins=20)
plt.xlabel('BMI (Normalized)')
plt.ylabel('Frequency')
plt.title('Histogram of BMI (Normalized)')
plt.show()
```



In [380]:
```
import matplotlib.pyplot as plt

# Scatter plot dari nilai 'BMI' yang sudah dinormalisasi
plt.scatter(range(len(normalized_df['BMI'])), normalized_df['BMI'], color='green', alpha=0.5)
plt.xlabel('Index')
plt.ylabel('BMI (Normalized)')
plt.title('Scatter Plot of Normalized BMI')
plt.show()
```



TRAINING DATA

In [381]:
```
import numpy as np

# Fungsi kernel RBF
def rbf_kernel(x1, x2, gamma=0.1):
    return np.exp(-gamma * np.linalg.norm(x1 - x2)**2)
```

In [382]:
```
# Fungsi untuk mendapatkan matriks kernel
def compute_kernel_matrix(X, gamma=0.1):
    n_samples = X.shape[0]
    kernel_matrix = np.zeros((n_samples, n_samples))

    for i in range(n_samples):
        for j in range(n_samples):
            kernel_matrix[i, j] = rbf_kernel(X[i], X[j], gamma)

    return kernel_matrix
```

In [459]:
```
def train_svm(X, y, C=1.0993765999285414, gamma=0.8960879267840364, tol=1e-3, max_iter=6):
    n_samples, n_features = X.shape
    alpha = np.zeros(n_samples)
    b = 0
    kernel_matrix = compute_kernel_matrix(X, gamma)

    for _ in range(max_iter):
        for i in range(n_samples):
            # Hitung margin
            margin = np.sum(alpha * y * kernel_matrix[i]) + b
            # Hitung error
            error = margin - y[i]

            # Perbarui alpha jika diperlukan
            if (y[i] * error < -tol and alpha[i] < C) or (y[i] * error > tol and alpha[i] > 0):
                alpha[i] += y[i] * error

        # Perbarui nilai b
        b = np.mean(y - np.dot(alpha * y, kernel_matrix))

        # Cetak informasi
        print("Iterasi:", _)
        print("Jumlah vektor dukungan:", np.sum(alpha > 0))

        # Tambahkan kondisi untuk mencetak sampel yang dianggap sebagai vektor dukungan
        if np.sum(alpha > 0) > 0:
            print("Contoh vektor dukungan:", X[alpha > 0][0])

    # Hanya menyimpan vektor dukungan dan labelnya
    support_vectors = X[alpha > 0]
    support_labels = y[alpha > 0]

    print("Jumlah vektor dukungan setelah pelatihan:", len(support_labels))

    return alpha, b, support_vectors, support_labels
```

LAMPIRAN III Source Code Random State 42 [80 20]

Random State 42 [80 20]

Random State 42 [80 20]

In [461]:
```
# Menyiapkan data latih (80% train)
train_size = int(0.8 * len(normalized_df))
train_data = normalized_df[:train_size]
test_data = normalized_df[train_size:]

# fitur dan label
X_train = train_data.drop('Outcome', axis=1).values
y_train = train_data['Outcome'].values
X_test = test_data.drop('Outcome', axis=1).values
y_test = test_data['Outcome'].values
```

In [462]:
```
train_data
```

Out[462]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
609	0.058824	0.557789	0.508197	0.131313	0.215130	0.357675	0.025619	0.033333	-1.0
610	0.176471	0.532663	0.442623	0.212121	0.186761	0.460507	0.091375	0.050000	-1.0
611	0.176471	0.874372	0.475410	0.222222	0.229314	0.490313	0.219898	0.250000	1.0
612	0.411765	0.844221	0.721311	0.424242	0.379433	0.569300	0.302733	0.316667	1.0
613	0.352941	0.527638	0.655738	0.282828	0.000000	0.484352	0.341588	0.083333	-1.0

614 rows × 9 columns

In [463]:
```
X_train
```

Out[463]:
```
array([[0.35294118, 0.74371859, 0.59016393, ..., 0.50074516, 0.23441503,
        0.48333333],
       [0.05882353, 0.42713568, 0.54098361, ..., 0.39642325, 0.11656704,
        0.16666667],
       [0.47058824, 0.91959799, 0.52459016, ..., 0.34724292, 0.25362938,
        0.18333333],
       ...,
       [0.17647059, 0.87437186, 0.47540984, ..., 0.49031297, 0.21989752,
        0.25      ],
       [0.41176471, 0.84422111, 0.72131148, ..., 0.56929955, 0.30273271,
        0.31666667],
       [0.35294118, 0.52763819, 0.6557377 , ..., 0.48435171, 0.34158839,
        0.08333333]])
```

In [464]:
```
y_train
```

Out[464]:
```
array([ 1., -1.,  1., -1.,  1., -1.,  1., -1.,  1.,  1., -1.,  1., -1.,
        1.,  1.,  1.,  1.,  1., -1.,  1., -1., -1.,  1.,  1.,  1.,  1.,
        1., -1., -1., -1., -1.,  1., -1., -1., -1., -1., -1.,  1.,  1.,
        1., -1., -1., -1.,  1., -1.,  1., -1., -1.,  1., -1., -1., -1.,
       -1.,  1., -1., -1.,  1., -1., -1., -1., -1.,  1., -1., -1.,  1.,
       -1.,  1., -1., -1., -1., -1.,  1., -1.,  1., -1., -1., -1., -1.,
        1., -1., -1., -1., -1., -1.,  1., -1., -1., -1.,  1., -1., -1.,
       -1., -1.,  1., -1., -1., -1., -1., -1.,  1.,  1., -1., -1., -1.,
       -1., -1., -1., -1., -1.,  1.,  1.,  1., -1., -1.,  1.,  1.,  1.,
       -1., -1., -1.,  1., -1., -1., -1.,  1.,  1., -1., -1.,  1.,  1.,
        1.,  1.,  1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,
        1., -1., -1., -1., -1., -1., -1., -1., -1.,  1., -1.,  1.,  1.,
       -1., -1., -1.,  1., -1., -1., -1., -1.,  1.,  1., -1., -1., -1.,
       -1.,  1.,  1., -1., -1., -1., -1.,  1., -1.,  1., -1., -1., -1.,
       -1., -1., -1., -1.,  1.,  1.,  1.,  1.,  1., -1., -1., -1., -1.,
        1., -1., -1., -1., -1., -1.,  1., -1., -1.,  1.,  1., -1., -1.,
       -1.,  1.,  1.,  1.,  1., -1., -1., -1.,  1.,  1., -1.,  1., -1.,
       -1., -1., -1., -1., -1., -1., -1.,  1.,  1., -1., -1., -1.,  1.,
       -1.,  1., -1., -1.,  1., -1.,  1., -1., -1.,  1.,  1., -1., -1.,
       -1., -1., -1.,  1., -1., -1., -1.,  1., -1., -1.,  1.,  1., -1.,
```

TEST DATA

In [468]:
```
# Menyiapkan data uji (20% test)
test_data = normalized_df[train_size:]

# fitur dan label
X_test = test_data.drop('Outcome', axis=1).values
y_test = test_data['Outcome'].values
```

In [469]:
```
test_data
```

Out[469]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
614	0.647059	0.693467	0.606557	0.262626	0.170213	0.538003	0.204526	0.483333	1.0
615	0.176471	0.532663	0.590164	0.000000	0.000000	0.384501	0.055081	0.100000	-1.0
616	0.352941	0.587940	0.786885	0.000000	0.000000	0.427720	0.033732	0.150000	-1.0
617	0.117647	0.341709	0.508197	0.131313	0.017730	0.299553	0.076430	0.033333	-1.0
618	0.529412	0.562814	0.672131	0.242424	0.000000	0.420268	0.514091	0.483333	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

154 rows × 9 columns

In [478]:
```
X = X_train
y = y_train
```

In [477]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train, y_train)

# Prediksi
predictions = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7398373983739838
```

LAMPIRAN IV Source Code Random State 42 [70 30]

[70 30]

In [8]:
```
# Menyiapkan data latih (70% train)
train_size2 = int(0.7 * len(normalized_df))
train_data2 = normalized_df[:train_size2]
test_data2 = normalized_df[train_size2:]

# fitur dan label
X_train2 = train_data2.drop('Outcome', axis=1).values
y_train2 = train_data2['Outcome'].values
X_test2 = test_data2.drop('Outcome', axis=1).values
y_test2 = test_data2['Outcome'].values
```

In [9]:
```
train_data2
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
532	0.058824	0.432161	0.540984	0.525253	0.076832	0.615499	0.358241	0.133333	-1.0
533	0.352941	0.457286	0.000000	0.000000	0.000000	0.444113	0.180615	0.166667	-1.0
534	0.058824	0.386935	0.459016	0.303030	0.066194	0.496274	0.500854	0.050000	-1.0
535	0.235294	0.663317	0.000000	0.000000	0.000000	0.490313	0.095645	0.033333	1.0
536	0.000000	0.527638	0.737705	0.000000	0.000000	0.441133	0.050811	0.416667	-1.0

537 rows × 9 columns

In [13]:
```
# Menyiapkan data uji (30% test)
test_data2 = normalized_df[train_size2:]

# fitur dan label
X_test2 = test_data2.drop('Outcome', axis=1).values
y_test2 = test_data2['Outcome'].values
```

In [14]:
```
test_data2
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
537	0.000000	0.286432	0.491803	0.000000	0.000000	0.323398	0.280529	0.766667	-1.0
538	0.000000	0.638191	0.655738	0.373737	0.248227	0.540984	0.309991	0.033333	-1.0
539	0.176471	0.648241	0.754098	0.494949	0.183215	0.542474	0.380017	0.183333	1.0
540	0.470588	0.502513	0.606557	0.404040	0.254137	0.587183	0.248933	0.366667	1.0
541	0.176471	0.643216	0.590164	0.252525	0.224586	0.482861	0.201110	0.100000	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

231 rows × 9 columns

In [18]:
```
X2 = X_train2
y2 = y_train2
```

In [19]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.3, random_state=42)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train2, y_train2)

# Prediksi
predictions = model.predict(X_test2)

# Evaluasi model
accuracy = accuracy_score(y_test2, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8271604938271605
```

LAMPIRAN V Source Code Random State 42 [90 10]

[90 10]

In [22]:
```
# Menyiapkan data latih (90% train)
train_size3 = int(0.9 * len(normalized_df))
train_data3 = normalized_df[:train_size3]
test_data3 = normalized_df[train_size3:]

# fitur dan label
X_train3 = train_data3.drop('Outcome', axis=1).values
y_train3 = train_data3['Outcome'].values
X_test3 = test_data3.drop('Outcome', axis=1).values
y_test3 = test_data3['Outcome'].values
```

In [23]:
```
train_data3
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
686	0.176471	0.653266	0.524590	0.000000	0.000000	0.344262	0.100769	0.016667	-1.0
687	0.058824	0.537688	0.409836	0.191919	0.000000	0.421759	0.043980	0.133333	-1.0
688	0.058824	0.703518	0.606557	0.262626	0.212766	0.359165	0.320239	0.033333	-1.0
689	0.058824	0.723618	0.672131	0.464646	0.212766	0.687034	0.109735	0.416667	1.0
690	0.470588	0.537688	0.655738	0.000000	0.000000	0.366617	0.332195	0.216667	-1.0

691 rows × 9 columns

In [30]:
```
# Menyiapkan data uji (10% test)
test_data3 = normalized_df[train_size3:]

# fitur dan label
X_test3 = test_data3.drop('Outcome', axis=1).values
y_test3 = test_data3['Outcome'].values
```

In [31]:
```
test_data3
```

Out[31]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
691	0.764706	0.793970	0.934426	0.000000	0.000000	0.630402	0.076430	0.383333	1.0
692	0.117647	0.608040	0.573770	0.323232	0.112293	0.582712	0.345004	0.033333	-1.0
693	0.411765	0.648241	0.557377	0.494949	0.147754	0.573770	0.154142	0.366667	1.0
694	0.117647	0.452261	0.491803	0.000000	0.000000	0.350224	0.048249	0.066667	-1.0
695	0.411765	0.713568	0.737705	0.242424	0.567376	0.453055	0.021349	0.366667	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

77 rows × 9 columns

In [32]:
```
X3 = X_train3
y3 = y_train3
```

In [34]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size=0.1, random_state=42)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train3, y_train3)

# Prediksi
predictions = model.predict(X_test3)

# Evaluasi model
accuracy = accuracy_score(y_test3, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8714285714285714
```

LAMPIRAN VI Source Code Random State 50 [80 20]

Random State 50 [80 20]

In [96]:
```
# Menyiapkan data latih (80% train)
train_size4 = int(0.8 * len(normalized_df))
train_data4 = normalized_df[:train_size4]
test_data4 = normalized_df[train_size4:]

# fitur dan label
X_train4 = train_data4.drop('Outcome', axis=1).values
y_train4 = train_data4['Outcome'].values
X_test4 = test_data4.drop('Outcome', axis=1).values
y_test4 = test_data4['Outcome'].values
```

In [97]:
```
train_data4
```

Out[97]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
609	0.058824	0.557789	0.508197	0.131313	0.215130	0.357675	0.025619	0.033333	-1.0
610	0.176471	0.532663	0.442623	0.212121	0.186761	0.460507	0.091375	0.050000	-1.0
611	0.176471	0.874372	0.475410	0.222222	0.229314	0.490313	0.219898	0.250000	1.0
612	0.411765	0.844221	0.721311	0.424242	0.379433	0.569300	0.302733	0.316667	1.0
613	0.352941	0.527638	0.655738	0.282828	0.000000	0.484352	0.341588	0.083333	-1.0

614 rows × 9 columns

In [100]:
```
# Menyiapkan data uji (20% test)
test_data4 = normalized_df[train_size4:]

# fitur dan label
X_test4 = test_data4.drop('Outcome', axis=1).values
y_test4 = test_data4['Outcome'].values
```

In [101]:
```
test_data4
```

Out[101]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
614	0.647059	0.693467	0.606557	0.262626	0.170213	0.538003	0.204526	0.483333	1.0
615	0.176471	0.532663	0.590164	0.000000	0.000000	0.384501	0.055081	0.100000	-1.0
616	0.352941	0.587940	0.786885	0.000000	0.000000	0.427720	0.033732	0.150000	-1.0
617	0.117647	0.341709	0.508197	0.131313	0.017730	0.299553	0.076430	0.033333	-1.0
618	0.529412	0.562814	0.672131	0.242424	0.000000	0.420268	0.514091	0.483333	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

154 rows × 9 columns

In [102]:
```
X4 = X_train4
y4 = y_train4
```

In [103]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train4, X_test4, y_train4, y_test4 = train_test_split(X4, y4, test_size=0.2, random_state=50)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train4, y_train4)

# Prediksi
predictions = model.predict(X_test4)

# Evaluasi model
accuracy = accuracy_score(y_test4, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7804878048780488
```

LAMPIRAN VII Source Code Random State 50 [70 30]

[70 30]

In [106]:
```
# Menyiapkan data latih (70% train)
train_size5 = int(0.7 * len(normalized_df))
train_data5 = normalized_df[:train_size5]
test_data5 = normalized_df[train_size5:]

# fitur dan label
X_train5 = train_data5.drop('Outcome', axis=1).values
y_train5 = train_data5['Outcome'].values
X_test5 = test_data5.drop('Outcome', axis=1).values
y_test5 = test_data5['Outcome'].values
```

In [107]:
```
train_data5
```

Out[107]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
532	0.058824	0.432161	0.540984	0.525253	0.076832	0.615499	0.358241	0.133333	-1.0
533	0.352941	0.457286	0.000000	0.000000	0.000000	0.444113	0.180615	0.166667	-1.0
534	0.058824	0.386935	0.459016	0.303030	0.066194	0.496274	0.500854	0.050000	-1.0
535	0.235294	0.663317	0.000000	0.000000	0.000000	0.490313	0.095645	0.033333	1.0
536	0.000000	0.527638	0.737705	0.000000	0.000000	0.441133	0.050811	0.416667	-1.0

537 rows × 9 columns

In [110]:
```
# Menyiapkan data uji (30% test)
test_data5 = normalized_df[train_size5:]

# fitur dan label
X_test5 = test_data5.drop('Outcome', axis=1).values
y_test5 = test_data5['Outcome'].values
```

In [111]:
```
test_data5
```

Out[111]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
537	0.000000	0.286432	0.491803	0.000000	0.000000	0.323398	0.280529	0.766667	-1.0
538	0.000000	0.638191	0.655738	0.373737	0.248227	0.540984	0.309991	0.033333	-1.0
539	0.176471	0.648241	0.754098	0.494949	0.183215	0.542474	0.380017	0.183333	1.0
540	0.470588	0.502513	0.606557	0.404040	0.254137	0.587183	0.248933	0.366667	1.0
541	0.176471	0.643216	0.590164	0.252525	0.224586	0.482861	0.201110	0.100000	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

231 rows × 9 columns

In [112]:
```
X5 = X_train5
y5 = y_train5
```

In [113]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train5, X_test5, y_train5, y_test5 = train_test_split(X5, y5, test_size=0.3, random_state=50)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train5, y_train5)

# Prediksi
predictions = model.predict(X_test5)

# Evaluasi model
accuracy = accuracy_score(y_test5, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7654320987654321
```

LAMPIRAN VIII Source Code Random State 50 [90 10]

[90 10]

In [116]:
```
# Menyiapkan data latih (90% train)
train_size6 = int(0.9 * len(normalized_df))
train_data6 = normalized_df[:train_size6]
test_data6 = normalized_df[train_size6:]

# fitur dan label
X_train6 = train_data6.drop('Outcome', axis=1).values
y_train6 = train_data6['Outcome'].values
X_test6 = test_data6.drop('Outcome', axis=1).values
y_test6 = test_data6['Outcome'].values
```

In [117]:
```
train_data6
```

Out[117]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
686	0.176471	0.653266	0.524590	0.000000	0.000000	0.344262	0.100769	0.016667	-1.0
687	0.058824	0.537688	0.409836	0.191919	0.000000	0.421759	0.043980	0.133333	-1.0
688	0.058824	0.703518	0.606557	0.262626	0.212766	0.359165	0.320239	0.033333	-1.0
689	0.058824	0.723618	0.672131	0.464646	0.212766	0.687034	0.109735	0.416667	1.0
690	0.470588	0.537688	0.655738	0.000000	0.000000	0.366617	0.332195	0.216667	-1.0

691 rows × 9 columns

In [120]:
```
# Menyiapkan data uji (10% test)
test_data6 = normalized_df[train_size6:]

# fitur dan label
X_test6 = test_data6.drop('Outcome', axis=1).values
y_test6 = test_data6['Outcome'].values
```

In [121]:
```
test_data6
```

Out[121]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
691	0.764706	0.793970	0.934426	0.000000	0.000000	0.630402	0.076430	0.383333	1.0
692	0.117647	0.608040	0.573770	0.323232	0.112293	0.582712	0.345004	0.033333	-1.0
693	0.411765	0.648241	0.557377	0.494949	0.147754	0.573770	0.154142	0.366667	1.0
694	0.117647	0.452261	0.491803	0.000000	0.000000	0.350224	0.048249	0.066667	-1.0
695	0.411765	0.713568	0.737705	0.242424	0.567376	0.453055	0.021349	0.366667	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

77 rows × 9 columns

In [122]:
```
X6 = X_train6
y6 = y_train6
```

In [123]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train6, X_test6, y_train6, y_test6 = train_test_split(X6, y6, test_size=0.1, random_state=50)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train6, y_train6)

# Prediksi
predictions = model.predict(X_test6)

# Evaluasi model
accuracy = accuracy_score(y_test6, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7857142857142857
```

LAMPIRAN IX Source Code Random State 60 [80 20]

Random State 60 [80 20]

In [137]:
```
# Menyiapkan data latih (80% train)
train_size7 = int(0.8 * len(normalized_df))
train_data7 = normalized_df[:train_size7]
test_data7 = normalized_df[train_size7:]

# fitur dan label
X_train7 = train_data7.drop('Outcome', axis=1).values
y_train7 = train_data7['Outcome'].values
X_test7 = test_data7.drop('Outcome', axis=1).values
y_test7 = test_data7['Outcome'].values
```

In [138]:
```
train_data7
```

Out[138]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
609	0.058824	0.557789	0.508197	0.131313	0.215130	0.357675	0.025619	0.033333	-1.0
610	0.176471	0.532663	0.442623	0.212121	0.186761	0.460507	0.091375	0.050000	-1.0
611	0.176471	0.874372	0.475410	0.222222	0.229314	0.490313	0.219898	0.250000	1.0
612	0.411765	0.844221	0.721311	0.424242	0.379433	0.569300	0.302733	0.316667	1.0
613	0.352941	0.527638	0.655738	0.282828	0.000000	0.484352	0.341588	0.083333	-1.0

614 rows × 9 columns

In [142]:
```
# Menyiapkan data uji (20% test)
test_data7 = normalized_df[train_size7:]

# fitur dan label
X_test7 = test_data7.drop('Outcome', axis=1).values
y_test7 = test_data7['Outcome'].values
```

In [143]:
```
test_data7
```

Out[143]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
614	0.647059	0.693467	0.606557	0.262626	0.170213	0.538003	0.204526	0.483333	1.0
615	0.176471	0.532663	0.590164	0.000000	0.000000	0.384501	0.055081	0.100000	-1.0
616	0.352941	0.587940	0.786885	0.000000	0.000000	0.427720	0.033732	0.150000	-1.0
617	0.117647	0.341709	0.508197	0.131313	0.017730	0.299553	0.076430	0.033333	-1.0
618	0.529412	0.562814	0.672131	0.242424	0.000000	0.420268	0.514091	0.483333	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

154 rows × 9 columns

In [144]:
```
X7 = X_train7
y7 = y_train7
```

In [145]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train7, X_test7, y_train7, y_test7 = train_test_split(X7, y7, test_size=0.2, random_state=60)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train7, y_train7)

# Prediksi
predictions = model.predict(X_test7)

# Evaluasi model
accuracy = accuracy_score(y_test7, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7886178861788617
```

LAMPIRAN X Source Code Random State 60 [70 30]

[70 30]

In [161]:
```
# Menyiapkan data latih (70% train)
train_size8 = int(0.7 * len(normalized_df))
train_data8 = normalized_df[:train_size8]
test_data8 = normalized_df[train_size8:]

# fitur dan label
X_train8 = train_data8.drop('Outcome', axis=1).values
y_train8 = train_data8['Outcome'].values
X_test8 = test_data8.drop('Outcome', axis=1).values
y_test8 = test_data8['Outcome'].values
```

In [162]:
```
train_data8
```

Out[162]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1.0
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	0.116567	0.166667	-1.0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	0.253629	0.183333	1.0
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	0.038002	0.000000	-1.0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1.0
...	...	...	...	...	...	...	...	...	...
532	0.058824	0.432161	0.540984	0.525253	0.076832	0.615499	0.358241	0.133333	-1.0
533	0.352941	0.457286	0.000000	0.000000	0.000000	0.444113	0.180615	0.166667	-1.0
534	0.058824	0.386935	0.459016	0.303030	0.066194	0.496274	0.500854	0.050000	-1.0
535	0.235294	0.663317	0.000000	0.000000	0.000000	0.490313	0.095645	0.033333	1.0
536	0.000000	0.527638	0.737705	0.000000	0.000000	0.441133	0.050811	0.416667	-1.0

537 rows × 9 columns

In [165]:
```
# Menyiapkan data uji (30% test)
test_data8 = normalized_df[train_size8:]

# fitur dan label
X_test8 = test_data8.drop('Outcome', axis=1).values
y_test8 = test_data8['Outcome'].values
```

In [166]:
```
test_data8
```

Out[166]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
537	0.000000	0.286432	0.491803	0.000000	0.000000	0.323398	0.280529	0.766667	-1.0
538	0.000000	0.638191	0.655738	0.373737	0.248227	0.540984	0.309991	0.033333	-1.0
539	0.176471	0.648241	0.754098	0.494949	0.183215	0.542474	0.380017	0.183333	1.0
540	0.470588	0.502513	0.606557	0.404040	0.254137	0.587183	0.248933	0.366667	1.0
541	0.176471	0.643216	0.590164	0.252525	0.224586	0.482861	0.201110	0.100000	1.0
...	...	...	...	...	...	...	...	...	...
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	-1.0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.111870	0.100000	-1.0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.071307	0.150000	-1.0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.115713	0.433333	1.0
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.033333	-1.0

231 rows × 9 columns

LAMPIRAN XI Source Code Random State 60 [90 10]

In [181]:
```
X9 = X_train9
y9 = y_train9
```

In [182]:
```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Contoh: X adalah fitur, y adalah target
X_train9, X_test9, y_train9, y_test9 = train_test_split(X9, y9, test_size=0.1, random_state=60)

# Contoh: kernel='rbf' untuk kernel RBF (default)
model = svm.SVC(kernel='rbf')

# Latih model
model.fit(X_train9, y_train9)

# Prediksi
predictions = model.predict(X_test6)

# Evaluasi model
accuracy = accuracy_score(y_test9, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5428571428571428
```