

MACHINE LEARNING

(Tugas 5)



Disusun Oleh:

Nama: Sifa Maryam Rahman

NPM: 41155050210031

Kelas: TIF A1

**TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS LANGLANGBUANA
2024**

1. Berikut adalah hasil praktik dari video youtube

<https://youtu.be/4zARMcgc7hA?si=x6RoHQXFF4NY76X8>

1.1. Persiapan sample dataset

```
import pandas as pd

sensus = {
    'tinggi': [158, 170, 183, 191, 155, 163, 180, 158, 178],
    'berat': [64, 86, 84, 80, 49, 59, 67, 54, 67],
    'jk': [
        'pria', 'pria', 'pria', 'pria', 'wanita', 'wanita', 'wanita', 'wanita',
        'wanita'
    ]
}
sensus_df = pd.DataFrame(sensus)
sensus_df
```

Nama : Sifa Maryam Rahman - 41155050210031

```
[2]:
```

	tinggi	berat	jk
0	158	64	pria
1	170	86	pria
2	183	84	pria
3	191	80	pria
4	155	49	wanita
5	163	59	wanita
6	180	67	wanita
7	158	54	wanita
8	178	67	wanita

1.2. Visualisasi dataset

```
[6]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

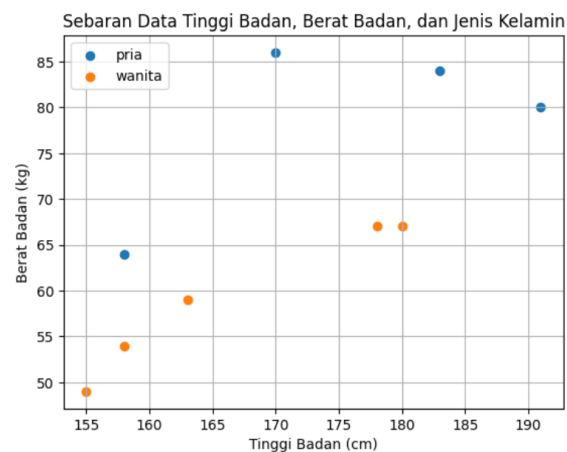
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
for jk, d in sensus_df.groupby('jk'):
    ax.scatter(d['tinggi'], d['berat'], label=jk)

plt.legend(loc='upper left')
plt.title('Sebaran Data Tinggi Badan, Berat Badan, dan Jenis Kelamin')
plt.xlabel('Tinggi Badan (cm)')
plt.ylabel('Berat Badan (kg)')
plt.grid(True)
plt.show()
```

Nama : Sifa Maryam Rahman - 41155050210031

Nama : Sifa Maryam Rahman - 41155050210031



1.3. Pengantar classification dengan K-Nearest Neighbours | KNN

1.4. Preprocessing dataset dengan Label Binarizer

```
[7]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

import numpy as np

X_train = np.array(sensus_df[['tinggi', 'berat']])
y_train = np.array(sensus_df['jk'])

print(f'X_train:\n{X_train}\n')
print(f'y_train: {y_train}')

Nama : Sifa Maryam Rahman - 41155050210031

X_train:
[[158  64]
 [170  86]
 [183  84]
 [191  80]
 [155  49]
 [163  59]
 [180  67]
 [158  54]
 [178  67]]

y_train: ['pria' 'pria' 'pria' 'pria' 'wanita' 'wanita' 'wanita' 'wanita' 'wanita']
```

```
[9]: from sklearn.preprocessing import LabelBinarizer

lb = LabelBinarizer()
y_train = lb.fit_transform(y_train)
print(f'y_train:\n{y_train}')

y_train:
[[0]
 [0]
 [0]
 [0]
 [1]
 [1]
 [1]
 [1]
 [1]]

[11]: y_train = y_train.flatten()
print(f'y_train:\n{y_train}')

y_train:
[0 0 0 0 1 1 1 1 1]

[ ]:
```

1.5. Training KNN Classification Model


```
[16]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

from sklearn.neighbors import KNeighborsClassifier

K = 3
model = KNeighborsClassifier(n_neighbors=K)
model.fit(X_train, y_train)

Nama : Sifa Maryam Rahman - 41155050210031

[16]: KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

[]: 

1.6. Prediksi dengan KNN Classification Model

```
[20]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f"Nama : {nama}\n")

      tinggi_badan = 155
      berat_badan = 70
      X_new = np.array([tinggi_badan, berat_badan]).reshape(1, -1)
      X_new

      Nama : Sifa Maryam Rahman - 41155050210031

[20]: array([[155, 70]])

[21]: y_new = model.predict(X_new)
      y_new

[21]: array([1])

[22]: lb.inverse_transform(y_new)

[22]: array(['wanita'], dtype='<U6')
```

1.7. Visualisasi Nearest Neighbours

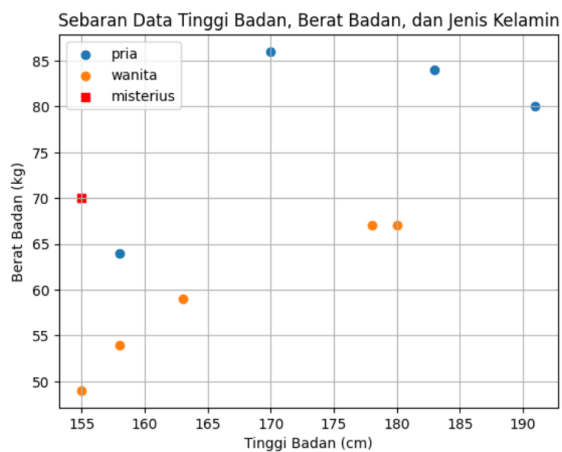
```
[24]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f"Nama : {nama}\n")

      fig, ax = plt.subplots()
      for jk, d in sensus_df.groupby('jk'):
          ax.scatter(d['tinggi'], d['berat'], label=jk)

      plt.scatter(tinggi_badan,
                  berat_badan,
                  marker='s',
                  color='red',
                  label='misterius')

      plt.legend(loc='upper left')
      plt.title('Sebaran Data Tinggi Badan, Berat Badan, dan Jenis Kelamin')
      plt.xlabel('Tinggi Badan (cm)')
      plt.ylabel('Berat Badan (kg)')
      plt.grid(True)
      plt.show()
```

Nama : Sifa Maryam Rahman - 41155050210031



1.8. Kalkulasi jarak dengan Euclidean Distance

```
[25]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f"Nama : {nama}\n")

      misterius = np.array([tinggi_badan, berat_badan])
      misterius

      Nama : Sifa Maryam Rahman - 41155050210031
```

```
[25]: array([155, 70])
```

```
[26]: X_train
```

```
[26]: array([[158, 64],
             [170, 86],
             [183, 84],
             [191, 80],
             [155, 49],
             [163, 59],
             [180, 67],
             [158, 54],
             [178, 67]])
```

```
[31]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f"Nama : {nama}\n")

      from scipy.spatial.distance import euclidean

      data_jarak = [euclidean(misterius, d) for d in X_train]
      data_jarak

      Nama : Sifa Maryam Rahman - 41155050210031
```

```
[31]: [np.float64(6.708203932499369),
      np.float64(21.93171219946131),
      np.float64(31.304951684997057),
      np.float64(37.36308338453881),
      np.float64(21.0),
      np.float64(13.601470508735444),
      np.float64(25.179356624028344),
      np.float64(16.278820596099706),
      np.float64(23.194827009486403)]
```

```
[32]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f"Nama : {nama}\n")

      sensus_df['jarak'] = data_jarak
      sensus_df.sort_values(['jarak'])

      Nama : Sifa Maryam Rahman - 41155050210031
```

```
[32]:
```

	tinggi	berat	jk	jarak
0	158	64	pria	6.708204
5	163	59	wanita	13.601471
7	158	54	wanita	16.278821
4	155	49	wanita	21.000000
1	170	86	pria	21.931712
8	178	67	wanita	23.194827
6	180	67	wanita	25.179357
2	183	84	pria	31.304952
3	191	80	pria	37.363083

```
[ ]:
```

1.9. Evaluasi KNN Classification Model | Persiapan testing set

```
[33]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f>Nama : {nama}\n")

      X_test = np.array([[168, 65], [180, 96], [168, 52], [169, 67]])
      y_test = lb.transform(np.array(['pria', 'pria', 'wanita', 'wanita'])).flatten()

      print(f'X_test:\n{X_test}\n')
      print(f'y_test:\n{y_test}')

      Nama : Sifa Maryam Rahman - 41155050210031

      X_test:
      [[168  65]
       [180  96]
       [168  52]
       [169  67]]

      y_test:
      [0 0 1 1]

[34]: y_pred = model.predict(X_test)
      y_pred

[34]: array([1, 0, 1, 1])
```

1.10. Evaluasi model dengan accuracy score

```
[35]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f>Nama : {nama}\n")

      from sklearn.metrics import accuracy_score

      acc = accuracy_score(y_test, y_pred)

      print(f'Accuracy: {acc}')

      Nama : Sifa Maryam Rahman - 41155050210031

      Accuracy: 0.75
```

1.11. Evaluasi model dengan precision score

```
[36]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f>Nama : {nama}\n")

      from sklearn.metrics import precision_score

      prec = precision_score(y_test, y_pred)

      print(f'Precision: {prec}')

      Nama : Sifa Maryam Rahman - 41155050210031

      Precision: 0.6666666666666666
```

1.12. Evaluasi model dengan recall score

```
[37]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f>Nama : {nama}\n")

      from sklearn.metrics import recall_score

      rec = recall_score(y_test, y_pred)

      print(f'Recall: {rec}')

      Nama : Sifa Maryam Rahman - 41155050210031

      Recall: 1.0
```

1.13. Evaluasi model dengan F1 score

```
[40]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

from sklearn.metrics import f1_score

f1 = f1_score(y_test, y_pred)

print(f'Recall: {f1}')

Nama : Sifa Maryam Rahman - 41155050210031

Recall: 0.8
```

[]:

1.14. Evaluasi model dengan classification report

```
[41]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

from sklearn.metrics import classification_report

cls_report = classification_report(y_test, y_pred)

print(f'Classification Report:\n{cls_report}')

Nama : Sifa Maryam Rahman - 41155050210031

Classification Report:
      precision    recall  f1-score   support

     0       1.00      0.50      0.67         2
     1       0.67      1.00      0.80         2

 accuracy          0.83
 macro avg          0.83
weighted avg          0.83
```

[]:

1.15. Evaluasi model dengan Mathews Correlation Coefficient

```
[42]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

from sklearn.metrics import matthews_corrcoef

mcc = matthews_corrcoef(y_test, y_pred)

print(f'MCC: {mcc}')

Nama : Sifa Maryam Rahman - 41155050210031

MCC: 0.5773502691896258
```

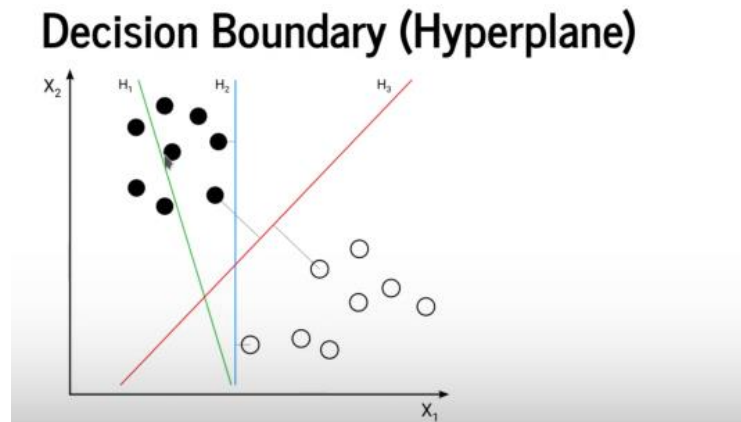
[]:

2. Berikut adalah hasil praktik dari video youtube

https://youtu.be/z69XYXpvVrE?si=KR_hDSlwjGIMcT0w

2.1 Pengenalan Decision Boundary & Hyperplane

Decision Boundary (Hyperplane)



Gambar ini menunjukkan ilustrasi sederhana dari *decision boundary* (batas keputusan) atau *hyperplane* yang digunakan dalam klasifikasi pembelajaran mesin, khususnya dalam ruang dua dimensi dengan sumbu yang dilabeli X_1 dan X_2 . Dua set data, yang direpresentasikan dengan lingkaran hitam dan putih, mewakili dua kelas yang berbeda.

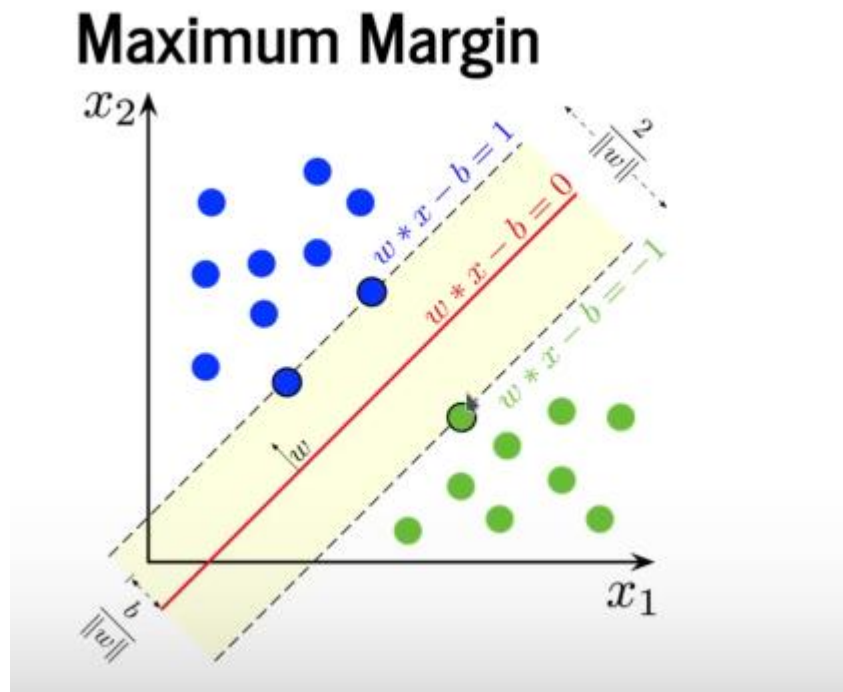
Berikut penjelasan dari masing-masing hyperplane:

1. **H_1 (Garis Hijau)** - Hyperplane ini adalah salah satu kemungkinan batas antara dua kelas. Namun, posisinya sangat dekat dengan titik-titik hitam, yang berarti tidak memberikan margin yang cukup besar di antara kedua kelas. Dalam klasifikasi, batas keputusan seperti ini mungkin tidak ideal karena bisa menyebabkan kesalahan klasifikasi jika data sedikit bervariasi.
2. **H_2 (Garis Biru)** - Hyperplane ini mewakili batas yang lebih baik yang memaksimalkan margin, sehingga menjadi *optimal hyperplane* dalam konteks *support vector machine* (SVM). SVM bertujuan untuk menemukan hyperplane yang memaksimalkan margin ini untuk meningkatkan akurasi dan ketahanan dalam klasifikasi.
3. **H_3 (Garis Merah)** - Hyperplane ini tidak memisahkan dua kelas dengan baik karena memotong kedua lingkaran hitam dan putih. Jika digunakan sebagai batas keputusan, ini akan menyebabkan kesalahan klasifikasi.

Secara keseluruhan, dari ketiga pilihan ini, H_2 adalah *optimal hyperplane* karena memaksimalkan margin antara dua kelas, yang merupakan prinsip utama dalam klasifikasi SVM.

2.2 Pengenalan Support Vector & Maximum Margin

Vector & Maximum Margin



Gambar ini menunjukkan konsep *Maximum Margin* dalam *Support Vector Machine* (SVM) pada ruang dua dimensi dengan sumbu x_1 dan x_2 . SVM adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi, dan tujuannya adalah untuk menemukan *hyperplane* (batas keputusan) yang memisahkan dua kelas dengan margin maksimum.

Berikut penjelasan elemen-elemen penting dalam gambar:

1. *yperplane* (garis merah, $w \cdot x - b = 0$) - Garis ini adalah *decision boundary* atau batas keputusan yang memisahkan dua kelas (titik biru dan titik hijau). Dalam SVM, *hyperplane* ini ditentukan oleh persamaan $w \cdot x - b = 0$
2. Support Vectors (titik biru dan hijau di garis putus-putus) - Titik-titik ini adalah data terdekat dari masing-masing kelas ke *hyperplane*. Mereka disebut *support vectors* dan memiliki pengaruh langsung dalam menentukan posisi *hyperplane*.

Dalam gambar, beberapa titik biru dan satu titik hijau berfungsi sebagai *support vectors*.

3. Margin Maksimum (daerah berwarna kuning) - Margin maksimum adalah jarak antara garis putus-putus (di atas dan di bawah hyperplane) yang melewati *support vectors* dari kedua kelas. Dalam SVM, kita berusaha memaksimalkan jarak ini agar model lebih akurat dan memiliki ketahanan terhadap perubahan data. Margin ini berukuran $\frac{2}{||w||}$
4. Garis Batas (garis putus-putus) - Garis-garis ini memiliki persamaan $w \cdot x - b = 1$ dan $w \cdot x - b = -1$, dan menandai batas margin maksimum. Setiap titik yang berada di luar atau di tepi margin maksimum dianggap sebagai anggota kelas masing-masing.

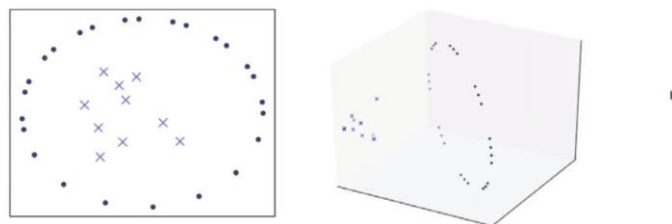
Inti dari konsep ini adalah memilih hyperplane yang memberikan margin terbesar antara dua kelas, yang membuat model lebih andal dan lebih tahan terhadap kesalahan klasifikasi pada data baru.

2.3 Pengenalan kondisi Linearly Inseparable dan Kernel Tricks

Linearly Inseparable dan Kernel Tricks

Linearly Inseparable & Kernel Tricks

Referensi: <https://www.quora.com/What-is-the-kernel-trick>



Gambar ini menjelaskan konsep data yang *linearly inseparable* (tidak dapat dipisahkan secara linear) dan bagaimana *kernel trick* dapat membantu dalam kasus tersebut. Konsep ini sering digunakan dalam *Support Vector Machine* (SVM) untuk menangani data yang tidak dapat dipisahkan dengan garis lurus atau *hyperplane* dalam ruang dua dimensi.

Berikut penjelasan mengenai gambar:

1. Gambar Kiri (Data Tidak Dapat Dipisahkan secara Linear):

Di gambar ini, terdapat dua kelompok data, yang ditandai dengan titik hitam dan tanda silang biru, yang tidak dapat dipisahkan dengan garis lurus dalam ruang dua dimensi. Jika kita mencoba menggambar garis lurus di antara kedua kelompok ini, kita tidak akan bisa memisahkan keduanya secara sempurna. Masalah seperti ini disebut sebagai *linearly inseparable*.

2. Gambar Kanan (Penggunaan *Kernel Trick* untuk Meningkatkan Dimensi):

- Untuk mengatasi masalah *linearly inseparable*, digunakan *kernel trick*. Dengan *kernel trick*, kita mentransformasikan data ke dimensi yang lebih tinggi, sehingga data yang sebelumnya tidak dapat dipisahkan menjadi mungkin untuk dipisahkan.
- Di gambar ini, data diubah dari dua dimensi menjadi tiga dimensi, yang memungkinkan pemisahan antara dua kelompok data menggunakan sebuah *hyperplane* (atau bidang dalam tiga dimensi). Dalam ruang tiga dimensi, kita bisa menarik bidang pemisah di antara dua kelompok data ini, yang sebelumnya tidak mungkin di ruang dua dimensi.

Inti dari *Kernel Trick*

Kernel trick memungkinkan model untuk bekerja di ruang berdimensi tinggi tanpa harus benar-benar menghitung koordinat data dalam dimensi tersebut. Ini dilakukan dengan menggunakan fungsi kernel untuk menghitung jarak antara titik data secara tidak langsung. Beberapa fungsi kernel yang populer adalah *linear kernel*, *polynomial kernel*, dan *radial basis function (RBF) kernel*.

Dalam konteks SVM, *kernel trick* sangat berguna karena memungkinkan kita memisahkan data yang tidak dapat dipisahkan secara linear dalam dimensi yang lebih rendah.

2.4 Pengenalan MNIST Handwritten Digits Dataset

```
[1]: nama = "Sifa Maryam Rahman - 41155050210031"
      print(f>Nama : {nama}\n")

      from sklearn.datasets import fetch_openml

      X, y = fetch_openml('mnist_784', data_home='./dataset/mnist', return_X_y=True)
      X.shape
```

Nama : Sifa Maryam Rahman - 41155050210031

[1]: (70000, 784)

```
[5]: import matplotlib.pyplot as plt
      import matplotlib.cm as cm

      pos = 1
      for data in X.to_numpy()[:8]:
          plt.subplot(1, 8, pos)
          plt.imshow(data.reshape((28, 28)),
                      cmap=cm.Greys_r)
          plt.axis('off')
          pos += 1

      plt.show()
```



[10]: y[:8]

```
[10]: 0 5
      1 0
      2 4
      3 1
      4 9
      5 2
      6 1
      7 3
      Name: class, dtype: category
      Categories (10, object): ['0', '1', '2', '3', ..., '6', '7', '8', '9']
```

```
[9]: X_train = X[:10000]
      y_train = y[:10000]
      X_test = X[69000:]
      y_test = y[69000:]
```

🏠 ⬆ ⬇ ⬅ 🔍 🗑

[]:

2.5 Klasifikasi dengan Support Vector Classifier | SVC

```
[12]: from sklearn.svm import SVC
```

```
model = SVC(random_state=0)  
model.fit(X_train, y_train)
```

```
[12]: SVC  
SVC(random_state=0)
```

```
[13]: nama = "Sifa Maryam Rahman - 41155050210031"  
print(f>Nama : {nama}\n")
```

```
from sklearn.metrics import classification_report  
  
y_pred = model.predict(X_test)  
print(classification_report(y_test, y_pred))
```

Nama : Sifa Maryam Rahman - 41155050210031

	precision	recall	f1-score	support
0	0.93	0.98	0.95	102
1	0.97	0.99	0.98	119
2	0.85	0.82	0.84	99
3	0.97	0.87	0.92	102
4	0.88	0.95	0.91	92
5	0.91	0.86	0.88	85
6	0.93	0.95	0.94	102
7	0.92	0.94	0.93	115
8	0.89	0.94	0.91	94
9	0.92	0.84	0.88	90
accuracy			0.92	1000
macro avg	0.92	0.91	0.91	1000
weighted avg	0.92	0.92	0.92	1000

```
[ ]:
```

2.6 Hyperparameter Tuning dengan Grid Search

```
[16]: nama = "Sifa Maryam Rahman - 41155050210031"  
print(f>Nama : {nama}\n")
```

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {  
    'kernel': ['rbf', 'poly', 'sigmoid'],  
    'C': [0.5, 1, 10, 100],  
    'gamma': ['scale', 1, 0.1, 0.01, 0.001]  
}
```

```
grid_search = GridSearchCV(estimator=SVC(random_state=0),  
                           param_grid=parameters,  
                           n_jobs=6,  
                           verbose=1,  
                           scoring='accuracy')
```

```
grid_search.fit(X_train, y_train)
```

Nama : Sifa Maryam Rahman - 41155050210031

Fitting 5 folds for each of 60 candidates, totalling 300 fits

```
[16]: GridSearchCV  
GridSearchCV(estimator=SVC(random_state=0), n_jobs=6,  
             param_grid={'C': [0.5, 1, 10, 100],  
                         'gamma': ['scale', 1, 0.1, 0.01, 0.001],  
                         'kernel': ['rbf', 'poly', 'sigmoid']}),  
             scoring='accuracy', verbose=1)  
  best_estimator_: SVC  
                  SVC(C=10, random_state=0)
```

```
[15]: print(f'Best Score: {grid_search.best_score_}')

best_params = grid_search.best_estimator_.get_params()
print(f'Best Parameters:')
for param in parameters:
    print(f'\t{param}: {best_params[param]}')

Best Score: 0.907
Best Parameters:
    kernel: rbf
    C: 10
    gamma: scale
```

```
[ ]:
```

2.7 Evaluasi Model

```
[17]: nama = "Sifa Maryam Rahman - 41155050210031"
print(f>Nama : {nama}\n")

y_pred = grid_search.predict(X_test)

print(classification_report(y_test, y_pred))

Nama : Sifa Maryam Rahman - 41155050210031
```

	precision	recall	f1-score	support
0	0.93	0.98	0.96	102
1	0.98	0.99	0.98	119
2	0.87	0.85	0.86	99
3	0.99	0.89	0.94	102
4	0.91	0.95	0.93	92
5	0.92	0.89	0.90	85
6	0.93	0.94	0.94	102
7	0.93	0.93	0.93	115
8	0.89	0.95	0.92	94
9	0.92	0.88	0.90	90
accuracy			0.93	1000
macro avg	0.93	0.92	0.92	1000
weighted avg	0.93	0.93	0.93	1000

```
[ ]:
```

