TUGAS PERTEMUAN 5

NAMA : MAYA NURROHMAH

NPM : 41155050210019

KELAS : INF_A1

1. K-Nearest Neighbours (KNN). Lakukan praktik dari https://youtu.be/4zARMcgc7hA?si=x6RoHQXFF4NY76X8 , buat screenshot dengan nama kalian pada coding, kumpulkan dalam bentuk pdf, dari kegiatan ini:
    1) Persiapan sample dataset

```
[1]: 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
```

[1]:

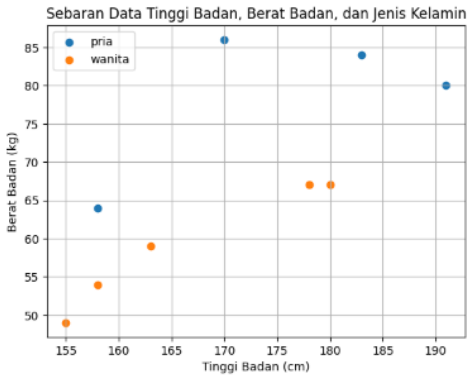
	tinggi	berat	jk
0	158	64	pria
1	170	86	pria
2	183	84	pria
3	191	80	pria
4	155	49	wanita
5	163	59	wanita
6	180	67	wanita
7	158	54	wanita
8	178	67	wanita

```
[2]: nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

    2) Visualisasi dataset

```
[3]: 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()
```




```
[4]: nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

3) Pengantar classification dengan K-Nearest Neighbours | KNN
   Kita akan melakukan KNN untuk melakukan klasifikasi jenis kelamin berdasrkan data tinggi dan berat badan. Sesuai dengan Namanya model machine learning satu ini akan melakukan prediksi dalam kasus ini adalah prediksi gender atau prediksi jenis kelamin berdasarkan kemiripan karakteristik dengan dataset yang kita miliki.
4) Preprocessing dataset dengan Label Binarizer

[6]:
```
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}')
```

```
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']
```

[7]:
```
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: {y_train}')
```

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

[12]:
```
nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

5) Training KNN Classification Model

[14]:
```
from sklearn.neighbors import KNeighborsClassifier

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

[14]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

6) Prediksi dengan KNN Classification Model

[16]:
```
tinggi_badan = 155
berat_badan = 70
X_new = np.array([tinggi_badan, berat_badan]).reshape(1, -1)
X_new
```

[16]:
```
array([[155,  70]])
```

[17]:
```
y_new = model.predict(X_new)
y_new
```

[17]:
```
array([1])
```

[18]:
```
lb.inverse_transform(y_new)
```

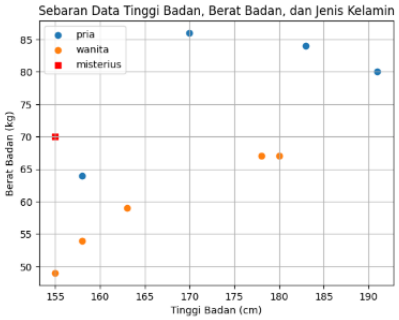
[18]:
```
array(['wanita'], dtype='<U6')
```

[19]:
```
nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

7) Visualisasi Nearest Neighbours

```
[20]: 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.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()
```




```
[21]: nama = "Maya Nurrohmah"
```

8) Kalkulasi jarak dengan Euclidean Distance

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

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

```
[24]: X_train
```

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

```
[25]: from scipy.spatial.distance import euclidean

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

```
[25]: [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)]
```

```
[26]: sensus_df['jarak'] = data_jarak
sensus_df.sort_values(['jarak'])
```

[26]:

	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

```
[27]: nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

9) Evaluasi KNN Classification Model | Persiapan testing set

[28]:
```
X_test = np.array([[168, 65], [180, 96], [160, 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}')
```

```
X_test:
[[168  65]
 [180  96]
 [160  52]
 [169  67]]

y_test:
[0 0 1 1]
```

[29]:
```
y_pred = model.predict(X_test)
y_pred
```

[29]:
```
array([1, 0, 1, 1])
```

[30]:
```
nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

10) Evaluasi model dengan accuracy score

[33]:
```
from sklearn.metrics import accuracy_score

acc = accuracy_score(y_test, y_pred)

print(f'Accuracy: {acc}')

nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Accuracy: 0.75
Nama : Maya Nurrohmah
```

11) Evaluasi model dengan precision score

[35]:
```
from sklearn.metrics import precision_score

prec = precision_score(y_test, y_pred)

print(f'Precission: {prec}')

nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Precission: 0.6666666666666666
Nama : Maya Nurrohmah
```

12) Evaluasi model dengan recall score

```
from sklearn.metrics import recall_score

rec = recall_score(y_test, y_pred)

print(f'Recall: {rec}')

nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Recall: 1.0
Nama : Maya Nurrohmah
```

13) Evaluasi model dengan F1 score

```
[38]:
from sklearn.metrics import f1_score

f1 = f1_score(y_test, y_pred)

print(f'F1: {f1}')

nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
F1: 0.8
Nama : Maya Nurrohmah
```

14) Evaluasi model dengan classification report

```
[39]:
from sklearn.metrics import classification_report

cls_report = classification_report(y_test, y_pred)

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

nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.67      1.00      0.80         2

    accuracy                           0.75         4
   macro avg       0.83      0.75      0.73         4
weighted avg       0.83      0.75      0.73         4

Nama : Maya Nurrohmah
```

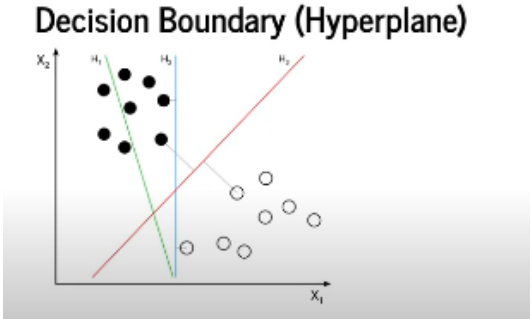
15) Evaluasi model dengan Mathews Correlation Coefficient

```
[40]:
from sklearn.metrics import matthews_corrcoef

mcc = matthews_corrcoef(y_test, y_pred)

print(f'MCC: {mcc}')

nama = "Maya Nurrohmah"

print(f"Nama : {nama}")
```

```
MCC: 0.5773502691896258
Nama : Maya Nurrohmah
```

2. Support Vector Machine (SVM). Lakukan praktik dari https://youtu.be/z69XYXpvVrE?si=KR_hDSlwjGIMcT0w , buat screenshot dengan nama kalian pada coding, kumpulkan dalam bentuk pdf, dari kegiatan ini:
   1) Pengenalan Decision Boundary & Hyperplane

Contoh kasus :



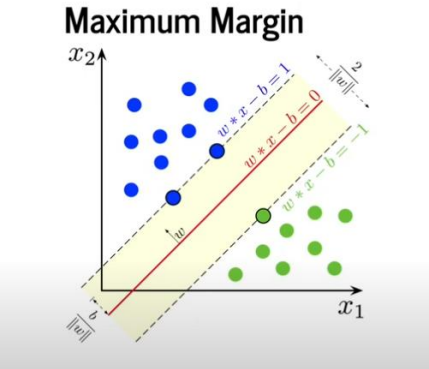
Pada Gambar diatas kita dihadapkan pada sebuah kasus klasifikasi dimana terdapat dua buah class, sebut saja class hitam dan putih. Pada kasus ini juga terdapat dua buah fitur yaitu x1 dan x2, pada kasus disini kita diminta untuk menarik suatu garis lurus atau garis linear yang dapat memisahkan kedua class ini. Dalam classification task pemisah atau pembatas antar class ini juga sering dikenal dengan istilan *Decision Boundary*, Semisal saja kita disini dihadapkan pada tiga pilihan garis linier yaitu H1,H2 dan H3. Dari ketiga garis linear ini manakah yang paling baik untuk digunakan sebagai Decision Boundary atau pemisah antar dua class yang kita miliki?. Jawabannya adalah H3 alasanya adalah karena H3 memiliki margin yang lebih besar bila dibandingkan dengan H2

Beberapa Terminologi Dasar

Terminologi yang pertama adalah terminology hyperplane, Terminologi Hyperplane ini merupakan terminology yang umum digunakan dalam svm untuk merepresentasikan Decision Boundary.

Dalam Kasus diatas karena kita dihadapkan pada dua fitur maka kita emeiliki floating dua dimensi, oleh karenanya decision boundry yang kita peroleh berupa suatu garis dan dalam kasus ini berupa garis lurus atau linear, tetapi ketika kita hanya memiliki satu fitur saja maka decision boundary akan *berupa titik atau nilai threshold*, lalu ketika terdapat 3 picture maka decision boundary nya berupa *sebuah bidang datar atau biasa dikenal dengan istilah flying*, ketika terdapat empat atau lebih picture maka pemisah antar classnya bidang multidimensi atau biasa dikenal dengan istilah *hyperplane.*

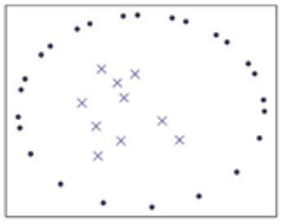
2) Pengenalan Support Vector & Maximum Margin

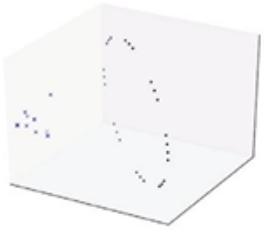


Margin ditentukan berdasarkan jarak terdekat antara Decision Boundary dengan anggota dari class yang ingin dipisahkan.

Pada gambar diatas terdapat dua buah picture yaitu x1 dan x2, lalu pada gambar diatas kita memiliki sebuah decision boundary berupa garis linear berwarna merah yang digunakan untuk memisahkan kedua class yang kita miliki, garis linear merah ini merupakan decision boundary yang memisahkan class biru dengan class hijau dan area yang diarsir dengan warna kuning itulah yang dinamakn dengan margin. Margin diperoleh berdasarkan jarak terdekat antara decision boundary dengan anggota dari class yang ingin dipisahkan, dan setiap anggota class yang berperan untuk menentukan margin dikenal sebagai *super vector.*

3) Pengenalan kondisi Linearly Inseparable dan Kernel Tricks





Pada kasus gambar diatas terdapat dua buah class yaitu class titik dan class x, dsini juga terdapat dua buah picture sehingga ketika dilakukan floating kita akan mendaoatkan floating dua dimensi semacam gambar diatas yang sebelah kiri. Nah pada kasus kali ini tidak memungkinkan bagi kita untuk menarik garis linear sebagai decision boundary kondisi semacam ini dikenal dengan istilah *Linearly Inseperable.* untuk mengatasi kondisi semacam ini svm akan memproyeksi data yang ada ke dimensi yang lebih tinggi, artinya bila data yang sebelumnya berada dalam dua dimensi maka svm akan memproyeksikannya ke 3 dimensi seperti Nampak pada floating di sisi sebelah kanan.

Upaya untuk mempoyeksikan sekumpulan data kedimensi yang lebih tinggi tentunya akan berimbas pada kenaikan beban komputasi, nah untuk menjawab kebutuhan semacam ini svm menawarkan Teknik yang sangat efisien yang dikenal dengan istilah *Kernel Tricks*.

4) Pengenalan MNIST Handwritten Digits Dataset

[1]:
```
from sklearn.datasets import fetch_openml

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

[1]:
```
(70000, 784)
```

[24]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

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

plt.show()
nama = "Maya Nurrohmah"
print(f"Nama : {nama}")
```

```
Nama : Maya Nurrohmah
```

[25]:
```
y[:8]
```

[25]:
```
y[:8]
```

[25]:
```
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']
```

[26]:
```
# X_train = X[:60000]
# y_train = y[:60000]
# X_test = X[60000:]
# y_test = y[60000:]

X_train = X[:1000]
y_train = y[:1000]
X_test = X[69000:]
y_test = y[69000:]
```

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)
```

```
[32]: from sklearn.metrics import classification_report

      y_pred = model.predict(X_test)
      print(classification_report(y_test, y_pred))

      nama = "Maya Nurrohmah"
      print(f"Nama : {nama}")
```

```
              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

Nama : Maya Nurrohmah
```

6) Hyperparameter Tuning dengan Grid Search

```
[30]: 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)
```

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

```
[30]: 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)
          SVC
          SVC(C=10, random_state=0)
```

```
[33]: 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]}')

      nama = "Maya Nurrohmah"
      print(f"Nama : {nama}")
```

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

7) Evaluasi Model

[34]:
```
y_pred = grid_search.predict(X_test)

print(classification_report(y_test, y_pred))
nama = "Maya Nurrohmah"
print(f"Nama : {nama}")
```

```
              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

Nama : Maya Nurrohmah
```