BAB	: LEARNING VECTOR QUANTIZATION
NAMA	: DIMAS TRI MUSTAKIM
NIM	: 205150200111049
TANGGAL	: 31/10/2022
ASISTEN	: ANDIKA IRZA PRADANA

A. Praktikum

1. Buka Google Colaboratory melalui tautan ini.
2. Tulis kode berikut ke dalam setiap *cell* pada *notebook* tersebut.
    a. Fungsi self-organizing maps

```
import numpy as np

def lvq_fit(train, target, lrate, b, max_epoch):
  label, train_idx = np.unique(target, return_index=True)
  weight = train[train_idx].astype(np.float64)
  train = np.array([e for i, e in enumerate(zip(train, target)) if
i not in train_idx], dtype=object)
  train, target = train[:, 0], train[:, 1]
  epoch = 0

  while epoch < max_epoch:
    for i, x in enumerate(train):
      distance = [sum((w - x) ** 2) for w in weight]
      min = np.argmin(distance)
      sign = 1 if target[i] == label[min] else -1
      weight[min] += sign * lrate * (x - weight[min])
    lrate *= b
    epoch += 1
  return weight, label
```

    b. Fungsi testing LVQ

```
def lvq_predict(X, model):
  center, label = model
  Y = []
  for x in X:
    d = [sum((c - x) ** 2) for c in center]
    Y.append(label[np.argmin(d)])
  return Y
```

    c. Fungsi hitung akurasi

```
def calc_accuracy(a, b):
  s = [1 if a[i] == b[i] else 0 for i in range(len(a))]
```

```
  return sum(s) / len(a)
```

d. Percobaan data acak dengan visualisasi

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs, make_classification

X, y = make_classification(n_samples=31, n_features=2,
n_redundant=0, n_informative=2, n_classes=3,
n_clusters_per_class=1)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
model = lvq_fit(X_train, y_train, lrate=.5, b=.8, max_epoch=50)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

for x, label in zip(X_train, y_train):
  plt.plot(x[0], x[1], colors[label] + '.')
for center, label in zip(model[0], model[1]):
  plt.plot(center[0], center[1], colors[label] + 'o')
for x, label in zip(X_test, output):
  plt.plot(x[0], x[1], colors[label] + 'x')
```

B. Screenshot

a. Fungsi training LVQ

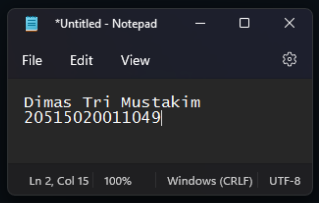
b. Fungsi testing LVQ

b) Fungsi *Testing* LVQ

Tulis kode ke dalam *cell* di bawah ini:

```
def lvq_predict(X, model):
  center, label = model
  Y = []
  for x in X:
    d = [sum((c - x) ** 2) for c in center]
    Y.append(label[np.argmin(d)])
  return Y
```

2] ✓ 0.4s

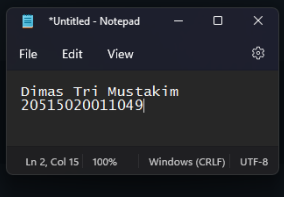


c. Fungsi hitung akurasi

c) Fungsi Hitung Akurasi

```
def calc_accuracy(a, b):
  s = [1 if a[i] == b[i] else 0 for i in range(len(a))]
  return sum(s) / len(a)
```

[3] ✓ 0.3s



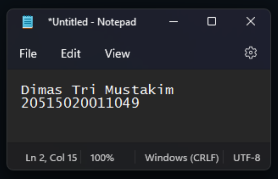
d. Percobaan data acak dengan visualisasi

d) Percobaan LVQ

Tulis kode ke dalam *cell* di bawah ini:

```
from random import uniform
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs, make_classification

X, y = make_classification(n_samples=31, n_features=2,
n_redundant=0, n_informative=2, n_classes=3, n_clusters_per_class=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = lvq_fit(X_train, y_train, lrate=.5, b=.8, max_epoch=50)
output = lvq_predict(X_test, model)
accuracy = calc_accuracy(output, y_test)
colors = 'rgbcmyk'
print('Accuracy:', accuracy)

for x, label in zip(X_train, y_train):
  plt.plot(x[0], x[1], colors[label] + '.')
for center, label in zip(model[0], model[1]):
  plt.plot(center[0], center[1], colors[label] + 'o')
for x, label in zip(X_test, output):
  plt.plot(x[0], x[1], colors[label] + 'x')
```

[6] ✓ 0.1s



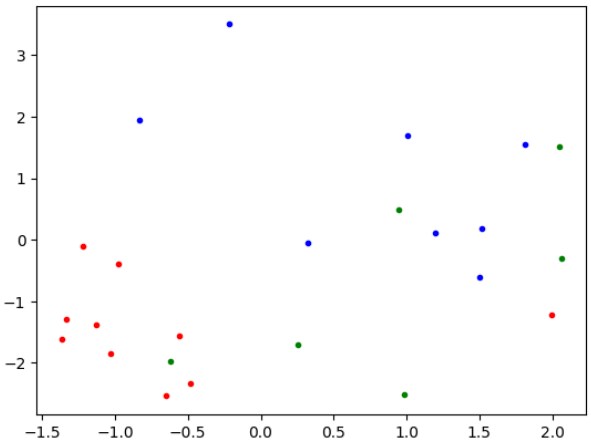

```
Accuracy: 1.0

C:\Users\tridi\AppData\Local\Temp\ipykernel_10968\1782378773.py:6: VisibleDeprecationWarning: Creat
sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes
this, you must specify 'dtype=object' when creating the ndarray.
  train = np.array([e for i, e in enumerate(zip(train, target)) if i not in train_idx])
```

C. Analisis

1. Jalankan kode d beberapa kali hingga didapat akurasi kurang dari 1. Amati dan analisis di mana terjadi error.

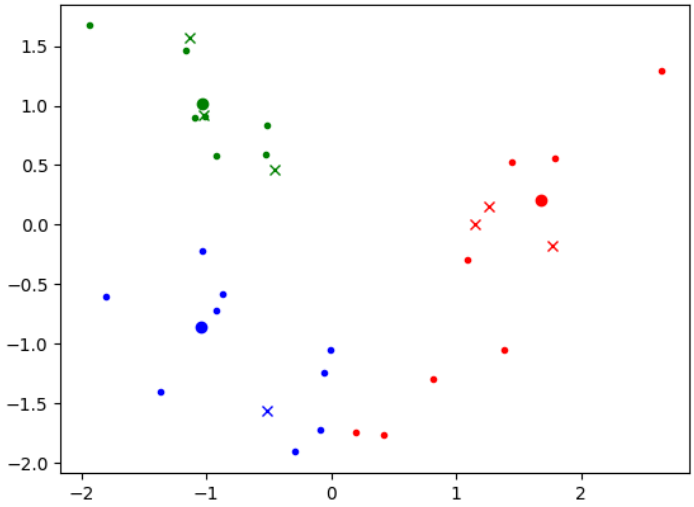
   Jawab:

   Pada kode program tersebut, akurasi yang didapatkan saat pelatihan terjadi fluktuasi dan tidak stabil di angka 1. Hal tersebut karena dataset yang dihasilkan melalui fungsi make_classification dari scikit-learn untuk pelatihan tidak sama dan terdapat data yang terlihat kurang membentuk pola saat di-*ploting*. Contohnya adalah data seperti gambar dibawah yang terdapat data dengan kelas biru, hijau, dan merah yang bercampur.



   LVQ melakukan klasifikasi menggunakan bobot yang memanfaatkan jarak antar bobot dengan data untuk melakukan klasifikasi, hampir sama seperti SOM. Ketika terdapat data yang terlihat tercampur seperti diatas, maka

algoritma ini tidak bisa mengklasifikasikannya dengan baik. Hal ini juga dapat dilihat ketika datanya lebih terlihat terurut, LVQ dapat memiliki akurasi dengan nilai 1 yang contohnya adalah seperti berikut.



D. Kesimpulan

Adaline LVQ atau Learning Vector Quantization merupakan jaringan saraf tiruan yang dirancang Kohonen setelah SOM. Arsitektur LVQ memiliki kemiripan dengan algoritma SOM. Bedanya adalah bahwa LVQ merupakan algoritma supervised learning untuk kebutuhan klasifikasi, sedangkan SOM merupakan algoritma unsupervised learning untuk kebutuhan klastering dan mengurangi dimensi data. Proses pelatihan keduanya juga berbeda karena pada LVQ terdapat data target. Data target digunakan untuk menentukan centroid/weight awal yang akan dilatih, kemudian saat proses update, bobot akan dijauhkan atau didekatkan tergantung apakah kelas bobot dengan data sudah sesuai.

LVQ juga berbeda dengan JST lain yang telah dipelajari sebelumnya seperti Perceptron dan Adaline. LVQ dapat mengklasifikasikan data multi-class dan tidak hanya binary seperti algoritma klasifikasi sebelumnya. Perbedaan selanjutnya adalah algoritma ini menggunakan bobot sebagai centroid untuk menentukan kelas menggunakan fungsi jarak (distance function) dan hanya melakukan update pada bobot terdekat saat proses pelatihan. LVQ juga tidak menggunakan fungsi aktivasi. Tidak seperti algoritma klasifikasi sebelumnya yang mengupdate keseluruhan bobot dan menggunakan fungsi aktivasi untuk menentukan kelas.