K-Nearest Neighbors Code

A. Review Teori K-Nearest eighbours

Algoritma k-Nearest Neighbor adalah algoritma supervised learning dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori k-tetangga terdekat. Tujuan dari algoritma ini adalah untuk mengklasifikasikan obyek baru berdasarkan atribut dan sample-sample dari training data. Algoritma k-Nearest Neighbor menggunakan Neighborhood Classification sebagai nilai prediksi dari nilai instance yang baru.

Contoh Kasus Sederhana

Misalnya ada sebuah rumah yang berada tepat di tengah perbatasan antara Kota Malang dan Kabupaten Malang, sehingga pemerintah kesulitan untuk menentukan apakah rumah tersebut termasuk kedalam wilayah Kota Malang atau Kabupaten Malang. Hal tersebut dapat ditentukan dengan menggunakan Algoritma k-NN, yaitu dengan melibatkan jarak antara rumah tersebut dengan rumah-rumah yang ada disekitarnya (tetangganya).

> **Pertama**, tentukan jumlah tetangga yg akan diperhitungkan (k), misalnya ditentukan 3 tetangga terdekat (k = 3).
>
> **Kedua**, hitung jarak setiap tetangga terhadap rumah tersebut, lalu urutkan hasilnya berdasarkan jarak, mulai dari yang terkecil ke yang terbesar.
>
> **Ketiga**, ambil 3 (k) tetangga yg paling dekat, lalu lihat masing-masing dari tetangga tersebut apakah termasuk kedalam wilayah Kota atau Kabupaten. Ada 2 kemungkinan:

Bila dari 3 tetangga tersebut terdapat ada 2 rumah yg termasuk kedalam wilayah Kota Malang, maka rumah tersebut termasuk kedalam wilayah Kota Malang.

Sebaliknya, bila dari 3 tetangga tersebut terdapat 2 rumah yg termasuk kedalam wilayah Kabupaten Malang, maka rumah tersebut termasuk kedalam wilayah Kabupaten Malang..

Cara Kerja Algoritma K-Nearest Neighbors (KNN)

K-nearest neaighbors melakukan klasifikasi dengan proyeksi data pembelajaran pada ruang berdimensi banyak. Ruang ini dibagi menjadi bagian-bagian yang merepresentasikan kriteria data

pembelajaran. Setiap data pembelajaran direpresentasikan menjadi titik-titik c pada ruang dimensi banyak.

- **Klasifikasi Terdekat (Nearest Neighbor Classification)**

Data baru yang diklasifikasi selanjutnya diproyeksikan pada ruang dimensi banyak yang telah memuat titik-titik c data pembelajaran. Proses klasifasikasi dilakukan dengan mencari titik c terdekat dari c-baru (nearest neighbor). Teknik pencarian tetangga terdekat yang umum dilakukan dengan menggunakan formula jarak euclidean. Measkipun ada beberapa formula perhitungan, namun formula yang umum dan banyak digunakan dalam algoritma K-NN adalah Euclidean Distance.

- **Euclidean Distance**

Jarak Euclidean adalah formula untuk mencari jarak antara 2 titik dalam ruang dua dimensi.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- **Banyaknya K Tetangga Terdekat**

Untuk menggunakan algoritma k nearest neighbors, perlu ditentukan banyaknya k tetangga terdekat yang digunakan untuk melakukan klasifikasi data baru. Banyaknya k, sebaiknya merupakan angka ganjil, misalnya k = 1, 2, 3, dan seterusnya. Penentuan nilai k dipertimbangkan berdasarkan banyaknya data yang ada dan ukuran dimensi yang dibentuk oleh data. Semakin banyak data yang ada, angka k yang dipilih sebaiknya semakin rendah. Namun, semakin besar ukuran dimensi data, angka k yang dipilih sebaiknya semakin tinggi.

Algoritma K-Nearest Neighbors

- Tentukan k bilangan bulat positif berdasarkan ketersediaan data pembelajaran.
- Pilih tetangga terdekat dari data baru sebanyak k.
- Tentukan klasifikasi paling umum pada langkah (ii), dengan menggunakan frekuensi terbanyak.
- Keluaran klasifikasi dari data sampel baru.

B. Pembuatan code K-Nearest Neighbors

Untuk implementasi code K-Nearest Neighbors akan spesifik dalam pengklasifikasian dan akan menggunakan dataset bunga iris sebagai contoh permasalahannya.

Implementasi Code Pada Dataset Bunga Iris.

Dataset bunga iris yang akan digunakan dalam jobsheet ini adalah spesifik dalam klasifikasi Bunga iris.

Pada dataset tersebut terdapat terdiri dari 150 pengamatan bunga iris dari tiga spesies yang berbeda. Ada 4 model pengukuran bunga iris yang dilakukan, meliputi: panjang sepal, lebar sepal, panjang kelopak dan lebar kelopak, dimana semuanya dalam satuan sentimeter yang sama. Atribut yang diprediksi nantinya adalah spesies, yang merupakan salah satu dari setosa, versicolor atau virginica.

Gambar Species Bunga Iris

Data tersebut merupakan dataset standar di mana spesies ini dikenal untuk semua contoh kasus. Dengan demikian, nantinya data akan dapat dibagi ke dalam pelatihan kemudian menguji kumpulan data serta menggunakan hasilnya untuk mengevaluasi implementasi dari algoritma K-Nearest Neighbors. Akurasi klasifikasi yang baik yang nantinya didapatkan dalam contoh kasus ini adalah di atas 90% atau biasanya 96% atau bahkan bisa lebih baik.

Hal pertama yang perlu dilakukan adalah letakan/simpan data iris pada *working directory* anda, dan beri nama filenya dengan iris.data. selanjutnya ikuti langkah-langkah percobaan implementasi K-Nearest Neighbors pada bunga iris sebagai berikut,

Tahapan percobaan :

1. Handle Data: Buka dataset dari CSV dan bagi menjadi dataset test / train
2. Similarity: Hitung jarak antara dua instance data
3. Neighbors: Cari k instances data yang paling mirip
4. Response: Generate a response dari serangkaian data instances.
5. Accuracy: melakukan cek keakurasian

1. Handle Data

Hal pertama yang perlu dilakukan adalah melakukan *load* file data bunga iris. Data tersebut dalam format CSV dimana tanpa ada baris header ataupun kutipan. Pengelolaan data tersebut dilakukan secara langsung melalui fungsi *reader* pada modul cvs sebagaimana berikut.

Handle Data

```
import csv
with open(rb'E:\KomKog\KNN\iris.data') as csvfile: #sesuaikan direktori dengan dimana file iris.data disimpan
    lines = csv.reader(csvfile)
    for row in lines:
        print (', '.join(row))
```

Hasil

```
5.1, 3.5, 1.4, 0.2, Iris-setosa
4.9, 3.0, 1.4, 0.2, Iris-setosa
4.7, 3.2, 1.3, 0.2, Iris-setosa
4.6, 3.1, 1.5, 0.2, Iris-setosa
5.0, 3.6, 1.4, 0.2, Iris-setosa
5.4, 3.9, 1.7, 0.4, Iris-setosa
4.6, 3.4, 1.4, 0.3, Iris-setosa
5.0, 3.4, 1.5, 0.2, Iris-setosa
4.4, 2.9, 1.4, 0.2, Iris-setosa
4.9, 3.1, 1.5, 0.1, Iris-setosa
5.4, 3.7, 1.5, 0.2, Iris-setosa
4.8, 3.4, 1.6, 0.2, Iris-setosa
4.8, 3.0, 1.4, 0.1, Iris-setosa
4.3, 3.0, 1.1, 0.1, Iris-setosa
5.8, 4.0, 1.2, 0.2, Iris-setosa
5.7, 4.4, 1.5, 0.4, Iris-setosa
```

5.4, 3.9, 1.3, 0.4, Iris-setosa
5.1, 3.5, 1.4, 0.3, Iris-setosa
5.7, 3.8, 1.7, 0.3, Iris-setosa
5.1, 3.8, 1.5, 0.3, Iris-setosa
5.4, 3.4, 1.7, 0.2, Iris-setosa
5.1, 3.7, 1.5, 0.4, Iris-setosa
4.6, 3.6, 1.0, 0.2, Iris-setosa
5.1, 3.3, 1.7, 0.5, Iris-setosa
4.8, 3.4, 1.9, 0.2, Iris-setosa
5.0, 3.0, 1.6, 0.2, Iris-setosa
5.0, 3.4, 1.6, 0.4, Iris-setosa
5.2, 3.5, 1.5, 0.2, Iris-setosa
5.2, 3.4, 1.4, 0.2, Iris-setosa
4.7, 3.2, 1.6, 0.2, Iris-setosa
4.8, 3.1, 1.6, 0.2, Iris-setosa
5.4, 3.4, 1.5, 0.4, Iris-setosa
5.2, 4.1, 1.5, 0.1, Iris-setosa
5.5, 4.2, 1.4, 0.2, Iris-setosa
4.9, 3.1, 1.5, 0.1, Iris-setosa
5.0, 3.2, 1.2, 0.2, Iris-setosa
5.5, 3.5, 1.3, 0.2, Iris-setosa
4.9, 3.1, 1.5, 0.1, Iris-setosa
4.4, 3.0, 1.3, 0.2, Iris-setosa
5.1, 3.4, 1.5, 0.2, Iris-setosa
5.0, 3.5, 1.3, 0.3, Iris-setosa
4.5, 2.3, 1.3, 0.3, Iris-setosa
4.4, 3.2, 1.3, 0.2, Iris-setosa
5.0, 3.5, 1.6, 0.6, Iris-setosa
5.1, 3.8, 1.9, 0.4, Iris-setosa
4.8, 3.0, 1.4, 0.3, Iris-setosa
5.1, 3.8, 1.6, 0.2, Iris-setosa
4.6, 3.2, 1.4, 0.2, Iris-setosa
5.3, 3.7, 1.5, 0.2, Iris-setosa
5.0, 3.3, 1.4, 0.2, Iris-setosa
7.0, 3.2, 4.7, 1.4, Iris-versicolor
6.4, 3.2, 4.5, 1.5, Iris-versicolor
6.9, 3.1, 4.9, 1.5, Iris-versicolor
5.5, 2.3, 4.0, 1.3, Iris-versicolor
6.5, 2.8, 4.6, 1.5, Iris-versicolor
5.7, 2.8, 4.5, 1.3, Iris-versicolor
6.3, 3.3, 4.7, 1.6, Iris-versicolor
4.9, 2.4, 3.3, 1.0, Iris-versicolor
6.6, 2.9, 4.6, 1.3, Iris-versicolor
5.2, 2.7, 3.9, 1.4, Iris-versicolor
5.0, 2.0, 3.5, 1.0, Iris-versicolor
5.9, 3.0, 4.2, 1.5, Iris-versicolor

6.0, 2.2, 4.0, 1.0, Iris-versicolor
6.1, 2.9, 4.7, 1.4, Iris-versicolor
5.6, 2.9, 3.6, 1.3, Iris-versicolor
6.7, 3.1, 4.4, 1.4, Iris-versicolor
5.6, 3.0, 4.5, 1.5, Iris-versicolor
5.8, 2.7, 4.1, 1.0, Iris-versicolor
6.2, 2.2, 4.5, 1.5, Iris-versicolor
5.6, 2.5, 3.9, 1.1, Iris-versicolor
5.9, 3.2, 4.8, 1.8, Iris-versicolor
6.1, 2.8, 4.0, 1.3, Iris-versicolor
6.3, 2.5, 4.9, 1.5, Iris-versicolor
6.1, 2.8, 4.7, 1.2, Iris-versicolor
6.4, 2.9, 4.3, 1.3, Iris-versicolor
6.6, 3.0, 4.4, 1.4, Iris-versicolor
6.8, 2.8, 4.8, 1.4, Iris-versicolor
6.7, 3.0, 5.0, 1.7, Iris-versicolor
6.0, 2.9, 4.5, 1.5, Iris-versicolor
5.7, 2.6, 3.5, 1.0, Iris-versicolor
5.5, 2.4, 3.8, 1.1, Iris-versicolor
5.5, 2.4, 3.7, 1.0, Iris-versicolor
5.8, 2.7, 3.9, 1.2, Iris-versicolor
6.0, 2.7, 5.1, 1.6, Iris-versicolor
5.4, 3.0, 4.5, 1.5, Iris-versicolor
6.0, 3.4, 4.5, 1.6, Iris-versicolor
6.7, 3.1, 4.7, 1.5, Iris-versicolor
6.3, 2.3, 4.4, 1.3, Iris-versicolor
5.6, 3.0, 4.1, 1.3, Iris-versicolor
5.5, 2.5, 4.0, 1.3, Iris-versicolor
5.5, 2.6, 4.4, 1.2, Iris-versicolor
6.1, 3.0, 4.6, 1.4, Iris-versicolor
5.8, 2.6, 4.0, 1.2, Iris-versicolor
5.0, 2.3, 3.3, 1.0, Iris-versicolor
5.6, 2.7, 4.2, 1.3, Iris-versicolor
5.7, 3.0, 4.2, 1.2, Iris-versicolor
5.7, 2.9, 4.2, 1.3, Iris-versicolor
6.2, 2.9, 4.3, 1.3, Iris-versicolor
5.1, 2.5, 3.0, 1.1, Iris-versicolor
5.7, 2.8, 4.1, 1.3, Iris-versicolor
6.3, 3.3, 6.0, 2.5, Iris-virginica
5.8, 2.7, 5.1, 1.9, Iris-virginica
7.1, 3.0, 5.9, 2.1, Iris-virginica
6.3, 2.9, 5.6, 1.8, Iris-virginica
6.5, 3.0, 5.8, 2.2, Iris-virginica
7.6, 3.0, 6.6, 2.1, Iris-virginica
4.9, 2.5, 4.5, 1.7, Iris-virginica
7.3, 2.9, 6.3, 1.8, Iris-virginica

6.7, 2.5, 5.8, 1.8, Iris-virginica
7.2, 3.6, 6.1, 2.5, Iris-virginica
6.5, 3.2, 5.1, 2.0, Iris-virginica
6.4, 2.7, 5.3, 1.9, Iris-virginica
6.8, 3.0, 5.5, 2.1, Iris-virginica
5.7, 2.5, 5.0, 2.0, Iris-virginica
5.8, 2.8, 5.1, 2.4, Iris-virginica
6.4, 3.2, 5.3, 2.3, Iris-virginica
6.5, 3.0, 5.5, 1.8, Iris-virginica
7.7, 3.8, 6.7, 2.2, Iris-virginica
7.7, 2.6, 6.9, 2.3, Iris-virginica
6.0, 2.2, 5.0, 1.5, Iris-virginica
6.9, 3.2, 5.7, 2.3, Iris-virginica
5.6, 2.8, 4.9, 2.0, Iris-virginica
7.7, 2.8, 6.7, 2.0, Iris-virginica
6.3, 2.7, 4.9, 1.8, Iris-virginica
6.7, 3.3, 5.7, 2.1, Iris-virginica
7.2, 3.2, 6.0, 1.8, Iris-virginica
6.2, 2.8, 4.8, 1.8, Iris-virginica
6.1, 3.0, 4.9, 1.8, Iris-virginica
6.4, 2.8, 5.6, 2.1, Iris-virginica
7.2, 3.0, 5.8, 1.6, Iris-virginica
7.4, 2.8, 6.1, 1.9, Iris-virginica
7.9, 3.8, 6.4, 2.0, Iris-virginica
6.4, 2.8, 5.6, 2.2, Iris-virginica
6.3, 2.8, 5.1, 1.5, Iris-virginica
6.1, 2.6, 5.6, 1.4, Iris-virginica
7.7, 3.0, 6.1, 2.3, Iris-virginica
6.3, 3.4, 5.6, 2.4, Iris-virginica
6.4, 3.1, 5.5, 1.8, Iris-virginica
6.0, 3.0, 4.8, 1.8, Iris-virginica
6.9, 3.1, 5.4, 2.1, Iris-virginica
6.7, 3.1, 5.6, 2.4, Iris-virginica
6.9, 3.1, 5.1, 2.3, Iris-virginica
5.8, 2.7, 5.1, 1.9, Iris-virginica
6.8, 3.2, 5.9, 2.3, Iris-virginica
6.7, 3.3, 5.7, 2.5, Iris-virginica
6.7, 3.0, 5.2, 2.3, Iris-virginica
6.3, 2.5, 5.0, 1.9, Iris-virginica
6.5, 3.0, 5.2, 2.0, Iris-virginica
6.2, 3.4, 5.4, 2.3, Iris-virginica
5.9, 3.0, 5.1, 1.8, Iris-virginica

Selanjutnya data perlu dibagi menjadi dataset *training* yang nantinya digunakan men-training K-Nearest Neighbors untuk membuat prediksi, serta dataset *test* yang digunakan untuk mengevaluasi akurasi dari model yang ada.

Hal selanjutnya yang dilakukan adalah mengubah nilai bunga yang ter-*load* sebagai string menjadi angka. Berikutnya membagi dataset secara acak menjadi dataset *train* datan dataset *test*. Gunakan rasio standar 67/33 untuk membagi dataset *train / test*.

Gunakan fungsi *loadDataset* berikut untuk me-*load* CSV dengan nama file yang telah disediakan tadi dan membaginya secara acak ke dalam file dataset train dan dataset test menggunakan rasio split yang telah ditentukan sebagaimana berik.

LoadDataset

```
import csv
import random

def loadDataset(filename, split, trainingSet=[] , testSet=[]):
    with open(rb'E:\KomKog\KNN\iris.data') as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(len(dataset)-1):
            for y in range(4):
                dataset[x][y] = float(dataset[x][y])
            if random.random() < split:
                trainingSet.append(dataset[x])
            else:
                testSet.append(dataset[x])
```

Selanjutnya file dataset csv akan tersimpan pada direktori, kemudian fungsi berikut digunakan untuk melakukan test untuk menguji keluaran dari dataset bunga iris tersebut,

LoadDataset Test

```
trainingSet=[]
    testSet=[]
    split = 0.67
    loadDataset('iris.data', split, trainingSet, testSet)
    print('Train set: ' + repr(len(trainingSet)))
    print('Test set: ' + repr(len(testSet)))
```

2. **Similarity**

Seanjutnya lakukan perhitungan Eclidean

Hal ini diperlukan untuk melakukan prediksi dengan terlebih dahulu dilakukan perhitungan kesamaan antara dua *instance* data yang diberikan. Hal tersebut diperlukan agar dapat ditemukan contoh instans K yang paling mirip dalam dataset training untuk anggota tertentu dari dataset pengujian dan selanjutnya nanti akan digunakan untuk melakukan prediksi.

Data keempat pengukuran bunga bersifat numerik dan memiliki satuan yang sama, maka dapat langsung menggunakan ukuran jarak Euclidean, yang merukapakan akar kuadrat dari jumlah perbedaan kuadrat antara dua array angka.

Selain itu, juga dilakukan kontrol bidang mana yang akan dimasukkan dalam perhitungan jarak, yang secara khusus, hanya menyertakan 4 atribut pertama saja, dengan melakukan pendekatan yaitu membatasi jarak euclidean, dan mengabaikan dimensi akhir. Fungsi euclideanDistance dijalankan sebagai berikut:

```
def euclideanDistance(instance1, instance2, length):
    distance = 0
    for x in range(length):
        distance += pow((instance1[x] - instance2[x]), 2)
    return math.sqrt(distance)
```

3. **Neighbors**

Langkah selanjutnya adalah melakukan proses penghitungan jarak untuk semua instance dan memilih subset dengan nilai jarak terkecil.

Masukan code dibawah ini yang merupakan fungsi getNeighbors yang berfungsi mengembalikan mengembalikan sebagian besar K terhadap *similar neighbors* dari dataset pelatihan untuk contoh pengujianuji yang diberikan.

getNeighbors

```
def getNeighbors(trainingSet, testInstance, k):
    distances = []
    length = len(testInstance)-1
    for x in range(len(trainingSet)):
        dist = euclideanDistance(testInstance, trainingSet[x], length)
        distances.append((trainingSet[x], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

4. **Response**

Setelah mendapatkan *similar neighbors* yang paling mirip untuk contoh pengujian, tugas selanjutnya adalah menyusun respons yang didapatkan dari hasil prediksi berdasarkan *similar neighbors* tersebut. Hal ini dapat dilakukan dengan mengizinkan setiap *similar neighbors* untuk memilih atribut kelas mereka, dan mengambil pilihan respon terbanyak sebagai prediksi.

Masukan code bawah yang merupakan fungsi untuk mendapatkan respons pilihan respon terbanyak dari sejumlah similar neighbors.

getResponse

```
def getResponse(neighbors):
    classVotes = {}
    for x in range(len(neighbors)):
        response = neighbors[x][-1]
        if response in classVotes:
```

```
            classVotes[response] += 1
        else:
            classVotes[response] = 1
    sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1),
reverse=True)
    return sortedVotes[0][0]
```

5. **Accuracy**

Dalam implementaasi algoritma K-Nearest Neighbors, hal tepenting adalah bagaimana mengevaluasi keakuratan hasil prediksi. Maka cara mudah untuk mengevaluasi keakuratan model dari hasil pelatihan adalah dengan melakukan perhitungan rasio total prediksi yang benar dari semua prediksi yang dibuat, yang disebut sebagai akurasi klasifikasi.

Masukan code di bawah ini ***getAccuracy*** yang beerfungsi untuk menjumlahkan total prediksi yang benar dan mensmpilksn nilsi akurasi sebagai nilai persentase dari klasifikasi yang benar.

getAccuracy

```
def getAccuracy(testSet, predictions):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct += 1
    return (correct/float(len(testSet))) * 100.0
```

6. **Hasil keseluruhan**

Langkah terakhir adalah melaengkapi keseluruhan elemen code/algoritma K-Nearest Neighbors menjadi satu bagian fungsi utama sebagaiamana code berikut:

K-Nearest Neighbors Python

```
import csv
import random
import math
import operator

def loadDataset(filename, split, trainingSet=[] , testSet=[]):
```

```
    with open(rb'E:\KomKog\KNN\iris.data') as csvfile:  #sesuaikan
direktori dengan dimana file iris.data disimpan
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(len(dataset)-1):
            for y in range(4):
                dataset[x][y] = float(dataset[x][y])
            if random.random() < split:
                trainingSet.append(dataset[x])
            else:
                testSet.append(dataset[x])

def euclideanDistance(instance1, instance2, length):
    distance = 0
    for x in range(length):
        distance += pow((instance1[x] - instance2[x]), 2)
    return math.sqrt(distance)

def getNeighbors(trainingSet, testInstance, k):
    distances = []
    length = len(testInstance)-1
    for x in range(len(trainingSet)):
        dist = euclideanDistance(testInstance, trainingSet[x], length)
        distances.append((trainingSet[x], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors

def getResponse(neighbors):
    classVotes = {}
    for x in range(len(neighbors)):
        response = neighbors[x][-1]
        if response in classVotes:
            classVotes[response] += 1
        else:
            classVotes[response] = 1
    sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1),
reverse=True)
    return sortedVotes[0][0]

def getAccuracy(testSet, predictions):
    correct = 0
    for x in range(len(testSet)):
```

```
        if testSet[x][-1] == predictions[x]:
            correct += 1
    return (correct/float(len(testSet))) * 100.0

def main():
    # menyiapkan data
    trainingSet=[]
    testSet=[]
    split = 0.67
    loadDataset('iris.data', split, trainingSet, testSet)
    print('Train set: ' + repr(len(trainingSet)))
    print('Test set: ' + repr(len(testSet)))
    # generate prediksi
    predictions=[]
    k = 3
    for x in range(len(testSet)):
        neighbors = getNeighbors(trainingSet, testSet[x], k)
        result = getResponse(neighbors)
        predictions.append(result)
        print('> prediksi=' + repr(result) + ', aktual=' +
repr(testSet[x][-1]))
    accuracy = getAccuracy(testSet, predictions)
    print('Akurasi: ' + repr(accuracy) + '%')

main()
```

Selanjutnya jalangkan fungsi utama dari algortima K-Nearest neighbors tersebut dengan cara ***right click -> run python file in terminal***

Maka akan didapatkan hasil dari perbandingan setiap prediksi dengan nilai kelas aktual pada dataset tes. Di akhir proses, Anda akan melihat keakuratan model. Dalam hal ini, bernilai 96,2%.

Hasil
Train set: 98 **Test set: 52** **> prediksi='Iris-setosa', aktual='Iris-setosa'** **> prediksi='Iris-setosa', aktual='Iris-setosa'** **> prediksi='Iris-setosa', aktual='Iris-setosa'**

```
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-setosa', aktual='Iris-setosa'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
> prediksi='Iris-versicolor', aktual='Iris-versicolor'
```

> prediksi='Iris-versicolor', aktual='Iris-versicolor'

> prediksi='Iris-versicolor', aktual='Iris-versicolor'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-versicolor', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-versicolor', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

> prediksi='Iris-virginica', aktual='Iris-virginica'

Akurasi: 96.15384615384616%