Tugas 3 Kecerdasan Buatan

K-Nearest-Neighbors

Disusun Oleh

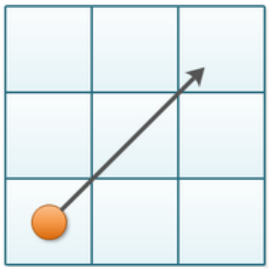
Riandi Kartiko
IF-40-02
1301164300

Deskripsi Masalah

K-Nearest Neighbors (KNN) adalah algoritma yang bertujuan untuk melakukan klasifikasi suatu data dimana hasil dari *instance* yang baru diklasifikasikan diambil dari K-mayoritas tetangga terdekat.

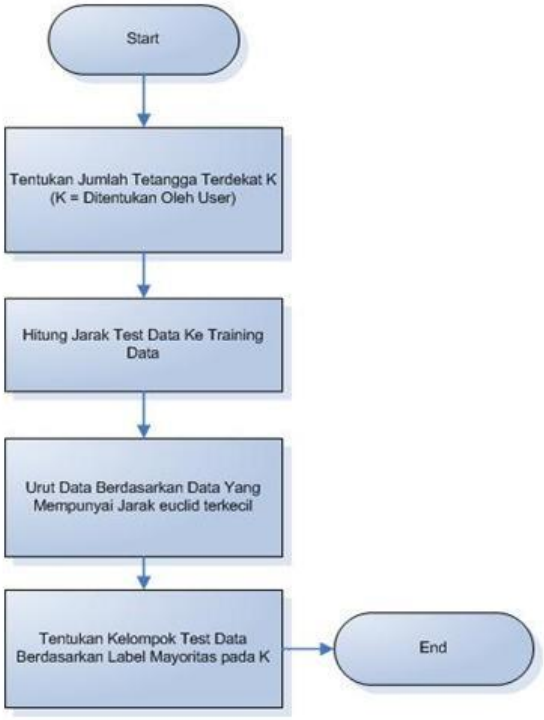
- Cara Kerja Algoritma K-Nearest Neighbors (KNN)
    1. Klasifikasi Terdekat (Nearest Neighbor Classification)
       Data baru yang diklasifikasi selanjutnya diproyeksikan pada ruang dimensi banyak yang telah memuat titik-titik c data pembelajaran. Proses klasifikasi dilakukan dengan mencari titik c terdekat dari c-baru (nearest).

Euclidean Distance



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

    2. Menentukan banyak K tetangga terdekat.



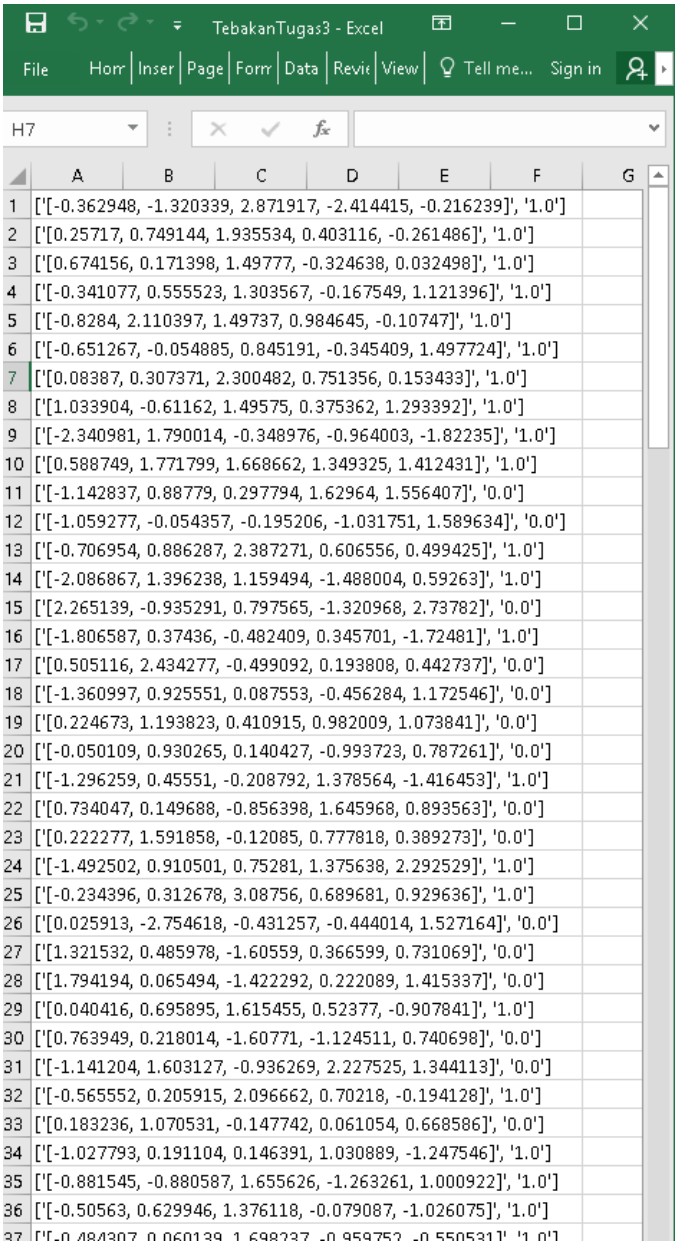
Algoritma

1. Program membaca file DataTrain_Tugas3_AI.csv dan DataTest_Tugas3_AI.csv
2. Mengubah data hanya sesuai column yang dinginkan dan memasukkan kedalam file temporary.
3. User input nilai K
4. Data dimasukkan ke fungsi utama program dengan parameter input data training, data test, dan nilai K
5. Membuat variable-variabel yang dibutuhkan, seperti array untuk menambpung jarak dan output, dan klasifikasi yang dibutuhkan
6. Membuat perulangan sebanyak data test dan membuat perulangan didalam perulangan diatas sebanyak data training.
7. Setiap data test dihitung jarak dengan data di data training
8. Mencari neighbor berdasarkan jarak dan nilai K
9. Mencari nilai response
10. Menulis output kedalam file TebakanTugas3.csv
11. Hapus file temporary.

Screenshot program

```python
def load_data_set(filename): #load file csv
    try:
        with open(filename, newline='') as iris:
            return list(reader(iris, delimiter=','))
    except FileNotFoundError as e:
        raise e

def convert_data(data,filename): #Hapus index,header pada datatugas
    data_file = open(filename, 'wt', newline ='')
    with data_file:
        writer = csv.writer(data_file, delimiter=',')
        writer.writerows(data)

def convert_to_float(data_set, mode): #masukkan csv ke array
    new_set = []
    if mode == 'training':
        for data in data_set:
            new_set.append([float(x) for x in data[:len(data)-1]] + [data[len(data)-1]])
    elif mode == 'test':
        for data in data_set:
            new_set.append([float(x) for x in data])
    else:
        print('Invalid mode, program will exit.')
        exit()
    return new_set

def get_classes(training_set): #Mendapatkan nilai Y pada datatrain
    return list(set([c[-1] for c in training_set]))

def find_neighbors(distances, k): #mendapatkan neighbor
    return distances[0:k]

def find_response(neighbors, classes):
    votes = [0] * len(classes)
    for instance in neighbors:
        for ctr, c in enumerate(classes):
            if instance[-2] == c:
                votes[ctr] += 1
    return max(enumerate(votes), key=itemgetter(1))

def knn(training_set, test_set, k):
    distances = [] #array untuk menampung nilai jarak
    temp = [] #array untuk menampung output
    dist = 0 #variabel jarak
    limit = len(training_set[0]) - 1 #mencari nilai Y
    # generate response classes from training data
    classes = get_classes(training_set)
    for test_instance in test_set:
        for row in training_set:
            for x, y in zip(row[:limit], test_instance):
                dist += (x-y) * (x-y)     #loop untuk mencari nilai jarak
            distances.append(row + [sqrt(dist)])
            dist = 0
        distances.sort(key=itemgetter(len(distances[0])-1))
        # find k nearest neighbors
        neighbors = find_neighbors(distances, k)
        # get the class with maximum votes
        index, value = find_response(neighbors, classes)
        temp.append([str(test_instance),classes[index]]) #menampung output ke array y
        distances.clear()

    with open('TebakanTugas3.csv', 'w+') as f: #write array ke file
        for item in temp:
            f.write("%s\n" % item)
    print(temp)

#mengambil data dari file csv dgn lib dari numpy
data_train = np.genfromtxt('DataTrain_Tugas3_AI.csv', delimiter=',', skip_header=1)
data_test  = np.genfromtxt('DataTest_Tugas3_AI.csv', delimiter=',', skip_header=1)

#memilih column yang ingin diambil
indexRemovedTrain = data_train[:,1:7] #column ke 1 hingga 7 dengan semua row
indexRemovedTest  = data_test[:,1:6] #column ke 1 hingga 6 dengan semua row

#mengubah data ke file csv sementara
convert_data(indexRemovedTrain,'convertedtrain.csv') #mengubah data
convert_data(indexRemovedTest,'convertedtest.csv')

k = int(input('Enter the value of k : ')) #user menentukan nilai K

# load the training and test data
#ubah file csv sementara ke dalam bentuk array
training_set = convert_to_float(load_data_set('convertedtrain.csv'), 'training')
test_set = convert_to_float(load_data_set('convertedtest.csv'), 'test')
if k > len(training_set):
    print('Jumlah K yang ditentukan melebihi data training')
else:
    knn(training_set, test_set, k) #fungsi utama program
#hapus file temporary
os.remove("convertedtrain.csv")
os.remove("convertedtest.csv")
```

Name	Type	Size	Value
data_test	float64	(200, 7)	[[1.000000e+00 -3.629480e-01 -1.320339e+00 ... -2.414415e+00 -2.162 ...
data_train	float64	(800, 7)	[[1.000000e+00 -1.608052e+00 -3.779920e-01 ... 1.313808e+00 1.218 ...
indexRemovedTest	float64	(200, 5)	[[-0.362948 -1.320339 2.871917 -2.414415 -0.216239] [0.25717 0.74 ...
indexRemovedTrain	float64	(800, 6)	[[-1.608052 -0.377992 1.204209 1.313808 1.218265 1.] [0.39 ...
k	int	1	50
test_set	list	200	[[-0.362948, -1.320339, 2.871917, -2.414415, -0.216239], [0.25717, 0.7 ...
training_set	list	800	[[-1.608052, -0.377992, 1.204209, 1.313808, 1.218265, ...], [0.393766, ...



	A	B	C	D	E	F	G
1	['[-0.362948, -1.320339, 2.871917, -2.414415, -0.216239]', '1.0']						
2	['[0.25717, 0.749144, 1.935534, 0.403116, -0.261486]', '1.0']						
3	['[0.674156, 0.171398, 1.49777, -0.324638, 0.032498]', '1.0']						
4	['[-0.341077, 0.555523, 1.303567, -0.167549, 1.121396]', '1.0']						
5	['[-0.8284, 2.110397, 1.49737, 0.984645, -0.10747]', '1.0']						
6	['[-0.651267, -0.054885, 0.845191, -0.345409, 1.497724]', '1.0']						
7	['[0.08387, 0.307371, 2.300482, 0.751356, 0.153433]', '1.0']						
8	['[1.033904, -0.61162, 1.49575, 0.375362, 1.293392]', '1.0']						
9	['[-2.340981, 1.790014, -0.348976, -0.964003, -1.82235]', '1.0']						
10	['[0.588749, 1.771799, 1.668662, 1.349325, 1.412431]', '1.0']						
11	['[-1.142837, 0.88779, 0.297794, 1.62964, 1.556407]', '0.0']						
12	['[-1.059277, -0.054357, -0.195206, -1.031751, 1.589634]', '0.0']						
13	['[-0.706954, 0.886287, 2.387271, 0.606556, 0.499425]', '1.0']						
14	['[-2.086867, 1.396238, 1.159494, -1.488004, 0.59263]', '1.0']						
15	['[2.265139, -0.935291, 0.797565, -1.320968, 2.73782]', '0.0']						
16	['[-1.806587, 0.37436, -0.482409, 0.345701, -1.72481]', '1.0']						
17	['[0.505116, 2.434277, -0.499092, 0.193808, 0.442737]', '0.0']						
18	['[-1.360997, 0.925551, 0.087553, -0.456284, 1.172546]', '0.0']						
19	['[0.224673, 1.193823, 0.410915, 0.982009, 1.073841]', '0.0']						
20	['[-0.050109, 0.930265, 0.140427, -0.993723, 0.787261]', '0.0']						
21	['[-1.296259, 0.45551, -0.208792, 1.378564, -1.416453]', '1.0']						
22	['[0.734047, 0.149688, -0.856398, 1.645968, 0.893563]', '0.0']						
23	['[0.222277, 1.591858, -0.12085, 0.777818, 0.389273]', '0.0']						
24	['[-1.492502, 0.910501, 0.75281, 1.375638, 2.292529]', '1.0']						
25	['[-0.234396, 0.312678, 3.08756, 0.689681, 0.929636]', '1.0']						
26	['[0.025913, -2.754618, -0.431257, -0.444014, 1.527164]', '0.0']						
27	['[1.321532, 0.485978, -1.60559, 0.366599, 0.731069]', '0.0']						
28	['[1.794194, 0.065494, -1.422292, 0.222089, 1.415337]', '0.0']						
29	['[0.040416, 0.695895, 1.615455, 0.52377, -0.907841]', '1.0']						
30	['[0.763949, 0.218014, -1.60771, -1.124511, 0.740698]', '0.0']						
31	['[-1.141204, 1.603127, -0.936269, 2.227525, 1.344113]', '0.0']						
32	['[-0.565552, 0.205915, 2.096662, 0.70218, -0.194128]', '1.0']						
33	['[0.183236, 1.070531, -0.147742, 0.061054, 0.668586]', '0.0']						
34	['[-1.027793, 0.191104, 0.146391, 1.030889, -1.247546]', '1.0']						
35	['[-0.881545, -0.880587, 1.655626, -1.263261, 1.000922]', '1.0']						
36	['[-0.50563, 0.629946, 1.376118, -0.079087, -1.026075]', '1.0']						
37	['[-0.484307, 0.060139, 1.698237, -0.959752, -0.550531]', '1.0']						