**Pokok Bahasan V**

**K-Nearest Neighbor (KNN).**

**Kode Pokok Bahasan**: TIK.RPL03.004.00.01

**Deskripsi Pokok Bahasan**:

Membahas bagaimana penerapan K-Nearest Neighbor untuk melakukan Klasifikasi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Elemen Kompetensi | Indikator Kinerja | Jml Jam | Hal |
| 1. | Menampilkan hasil Klasifikasi dari kasus yang diberikan. | Mampu melakukan analisis terhadap klasifikasi data dari diagram yangmuncul. | 1 |  |
| 2. | Melakukan perhitungan manual menggunakan excel. | Mampu melakukan perhitungan peluang secara manual menggunakan excel pada pada data yang ditentukan | 1 |  |

**TUGAS PENDAHULUAN**

Hal yang harus dilakukan dan acuan yang harus dibaca sebelum praktikum :

1. Menginstal R pada PC masing-masing praktikan.

2. Menginstal R Studio pada PC masing-masing praktikan.

3. Menginstal aplikasi pengolah data (Excel).

**DAFTAR PERTANYAAN**

1. Apa itu algoritma K-Nearest Neighbor?

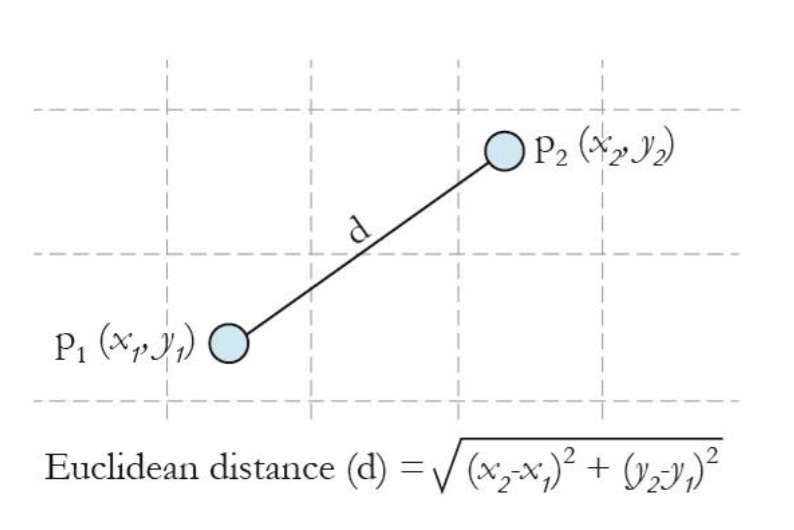
2. Apa kegunaan K-Nearest Neighbor?

3. Sebutkan tahapan dari proses algoritma K-Nearest Neighbor!

**TEORI SINGKAT**

Algoritma K-Nearest Neighbor (K-NN) adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasikan sebelumya. Termasuk dalam supervised learning, dimana hasil query instance yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam K-NN

KNN, Dapat digunakan untuk tujuan klasifikasi, Tidak menyusun model atau mengekstrak aturan logika tertentu sebagai hasil dari analisis, Identikasi k buah individu tetangga terdekat dilakukan dengan terlebih dahulu menghitung jarak dari individu yang akan diduga dengan setiap individu yang ada pada gugus data training. Jika ini sudah dilakukan maka tinggal mencari k buah amatan yang jaraknya paling kecil. Penghitungan jarak dari dua amatan A dan B dapat menggunakan formula Euclid distance.



**LAB SETUP**

Hal yang harus disiapkan dan dilakukan oleh praktikan untuk menjalankan praktikum modul ini.

1. Menginstall library yang dibutuhkan untuk mengerjakan modul.

2. Menjalankan R Studio.

3. Aplkasi pengolah data (Excel)

**ELEMEN KOMPETENSI I**

**Deskripsi:**

Menampilkan hasil klasifikasi dari kasus yang diberikan.

**Kompetensi Dasar**:

Menampilkan hasil Klasifikasi dari kasus yang diberikan.

**Latihan 1.1.1**

**Penjelasan Singkat :**

Pada latihan ini anda akan diminta melakukan analisis terhadap klasifikasi data dari diagram yang muncul.

**Langkah-Langkah Praktikum:**

1. Siapkan data

|  |  |  |
| --- | --- | --- |
| X | y | kelas |
| 0.15 | 0 | 1 |
| 0.3 | 0.1 | 1 |
| 0.11 | 0.12 | 1 |
| 0.04 | 0.13 | 1 |
| 0.18 | 0.17 | 1 |
| 0.1 | 0.19 | 1 |
| 0.54 | 0.2 | 2 |
| 0.19 | 0.21 | 1 |
| 0.6 | 0.24 | 2 |
| 0.66 | 0.25 | 2 |
| 0.33 | 0.26 | 2 |
| 0.17 | 0.27 | 1 |
| 0.52 | 0.29 | 2 |
| 0.52 | 0.33 | 2 |
| 0.49 | 0.35 | 2 |
| 0.68 | 0.36 | 2 |
| 0.55 | 0.43 | 2 |
| 0.37 | 0.44 | 2 |
| 0.52 | 0.47 | 2 |
| 0.58 | 0.5 | 2 |
|  |  |  |
| X | Y | Kelas |
| 0.1 | 0.2 | ? |
| 0.4 | 0.1 | ? |

1. Buka Jupyter Notebook
2. Mengimport module

from random import seed

from random import randrange

from csv import reader

from math import sqrt

1. Melakukan load file CSV

def load\_csv(filename):

dataset = list()

with open(filename, 'r') as file:

csv\_reader = reader(file)

for row in csv\_reader:

if not row:

continue

dataset.append(row)

return dataset

1. Mengubah string menajdi float

def str\_column\_to\_float(dataset, column):

for row in dataset:

row[column] = float(row[column].strip())

1. Mengubah string menjadi integer

def str\_column\_to\_int(dataset, column):

class\_values = [row[column] for row in dataset]

unique = set(class\_values)

lookup = dict()

for i, value in enumerate(unique):

lookup[value] = i

for row in dataset:

row[column] = lookup[row[column]]

return lookup

1. Mencari nilai min dan max untuk setiap kolom

def dataset\_minmax(dataset):

minmax = list()

for i in range(len(dataset[0])):

col\_values = [row[i] for row in dataset]

value\_min = min(col\_values)

value\_max = max(col\_values)

minmax.append([value\_min, value\_max])

return minmax

8. Mengubah skala data

def normalize\_dataset(dataset, minmax):

for row n dataset:

for i in range(len(row)):

row[i] = (row[i] - minmax[i][0]) / (minmax[i][1] - minmax[i][0])

9. Melakukan split data

def cross\_validation\_split(dataset, n\_folds):

dataset\_split = list()

dataset\_copy = list(dataset)

fold\_size = int(len(dataset) / n\_folds)

for \_ in range(n\_folds):

fold = list()

while len(fold) < fold\_size:

index = randrange(len(dataset\_copy))

fold.append(dataset\_copy.pop(index))

dataset\_split.append(fold)

return dataset\_split

10. Menghitung akurasi

def accuracy\_metric(actual, predicted):

correct = 0

for i in range(len(actual)):

if actual[i] == predicted[i]:

correct += 1

return correct / float(len(actual)) \* 100.0

11. Mengevaluasi algoritma menggunakan cross validation split

def evaluate\_algorithm(dataset, algorithm, n\_folds, \*args):

folds = cross\_validation\_split(dataset, n\_folds)

scores = list()

for fold in folds:

train\_set = list(folds)

train\_set.remove(fold)

train\_set = sum(train\_set, [])

test\_set = list()

for row in fold:

row\_copy = list(row)

test\_set.append(row\_copy)

row\_copy[-1] = None

predicted = algorithm(train\_set, test\_set, \*args)

actual = [row[-1] for row in fold]

accuracy = accuracy\_metric(actual, predicted)

scores.append(accuracy)

return scores

12. Menghitung jarak Euclidean

def euclidean\_distance(row1, row2):

distance = 0.0

for i in range(len(row1)-1):

distance += (row1[i] - row2[i])\*\*2

return sqrt(distance)

13. Menemukan neighors yang paling mirip

def get\_neighbors(train, test\_row, num\_neighbors):

distances = list()

for train\_row in train:

dist = euclidean\_distance(test\_row, train\_row)

distances.append((train\_row, dist))

distances.sort(key=lambda tup: tup[1])

neighbors = list()

for i in range(num\_neighbors):

neighbors.append(distances[i][0])

return neighbors

14. Membuat prediksi dengan neighbors

def predict\_classification(train, test\_row, num\_neighbors):

neighbors = get\_neighbors(train, test\_row, num\_neighbors)

output\_values = [row[-1] for row in neighbors]

prediction = max(set(output\_values), key=output\_values.count)

return prediction

15. Algoritma KNN

def k\_nearest\_neighbors(train, test, num\_neighbors):

predictions = list()

for row in test:

output = predict\_classification(train, row, num\_neighbors)

predictions.append(output)

return(predictions)

16. Menjalankan KNN

# Test the kNN on the Iris Flowers dataset

seed(1)

filename = 'knn.csv'

dataset = load\_csv(filename)

for i in range(len(dataset[0])-1):

str\_column\_to\_int(dataset, i)

# convert class column to integers

str\_column\_to\_int(dataset, len(dataset[0])-1)

# evaluate algorithm

n\_folds = 5

num\_neighbors = 5

scores = evaluate\_algorithm(dataset, k\_nearest\_neighbors, n\_folds, num\_neighbors)

print('Scores: %s' % scores)

print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))

# define model parameter

num\_neighbors = 5

# define a new record

row = [0.4, 0.1]

# predict the label

label = predict\_classification(dataset, row, num\_neighbors)

print('Data=%s, Predicted: %s' % (row, label))

**Tugas**

**Gunakan algorima k-NN pada R untuk kasus berikut ini dengan k=3 :**

|  |  |  |
| --- | --- | --- |
| sepal.length | sepal.width | species |
| 5.4 | 3.7 | setosa |
| 7.2 | 3.2 | virginica |
| 5.4 | 3.4 | setosa |
| 5.1 | 3.3 | setosa |
| 5.4 | 3.9 | setosa |
| 7.4 | 2.8 | virginica |
| 6.1 | 2.8 | versicolor |
| 7.2 | 2.9 | virginica |
| 6.1 | 2.7 | versicolor |
| 5.8 | 2.8 | virginica |
| 6.4 | 2.3 | versicolor |
| 5.1 | 2.7 | versicolor |
| 6.3 | 2.4 | versicolor |
| 5.4 | 2.3 | versicolor |

Data baru:

|  |  |  |
| --- | --- | --- |
| sepal.length | sepal.width | species |
| 5.2 | 3.7 | ? |

Script R :

|  |
| --- |
|  |

Output :

|  |
| --- |
|  |

**CEK LIST**

|  |  |  |  |
| --- | --- | --- | --- |
| Elemen Kompetensi | No Latihan | Penyelesaian | |
| Selesai | Tidak selesai |
| 1 | 1.1.1 |  |  |
| 2 | 1.2.1 |  |  |

**FORM UMPAN BALIK**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Elemen Kompetensi** | **Tingkat Kesulitan** | | | **Tingkat Ketertarikan** | | | **Waktu Penyelesaian dalam menit** |
| Menampilkan hasil Klasifikasi dari kasus yang diberikan.  yang diberikan. |  |  |  |  |  |  |  |
|  |  | Sangat Mudah |  |  | Tidak Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Mudah |  |  | Cukup Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Biasa |  |  | Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sulit |  |  | Sangat Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sangat Sulit |  |  |  |  |
|  |  |  |  |  |  |  |
| Melakukan perhitungan manual menggunakan excel. |  |  |  |  |  |  |  |
|  |  | Sangat Mudah |  |  | Tidak Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Mudah |  |  | Cukup Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Biasa |  |  | Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sulit |  |  | Sangat Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sangat Sulit |  |  |  |  |
|  |  |  |  |  |  |  |