**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 hasil prediksi dengan confusion matriks. | Melakukan perhitungan hasil prediksi dengan confusion matriks berdasarkan dataset pada studi kasus. | 1 |  |

**TUGAS PENDAHULUAN**

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

1. Menginstal Python pada PC masing-masing praktikan.

2. Menginstal Jupyter notebook pada PC masing-masing praktikan.

3. Menginstal library scikit-learn.

**DAFTAR PERTANYAAN**

1. Apa itu algoritma K-Nearest Neighbor?

algoritma untuk melakukan klasifikasi terhadap objek dengan data pembe- lajaran yang jaraknya paling dekat dengan objek tersebut.

2. Apa kegunaan K-Nearest Neighbor?

memiliki keunggulan dapat mengklasifikasikan data calon pegawai yang tidak diketahui dengan adanya data latih dan data uji. KNN dapat menprosedur yang berbasis matematis untuk mengevaluasi nilai kriteria-kriteria tersebut menjadi sebuah keterangan klasifikasi.

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

Langkah-1: Pilih nilai banyaknya tetangga K.

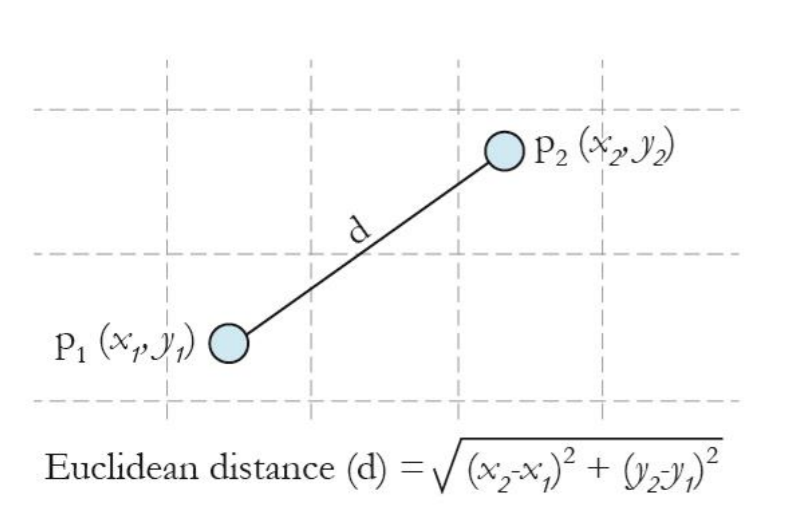
Langkah-2: Hitung jarak dari jumlah tetangga K (bisa menggunakan salah satu metrik jarak, misalnya Euclidean distance)

Langkah-3: Ambil tetangga terdekat K sesuai jarak yang dihitung.

**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 Jupyter notebook.

**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. import library yang dibutuhkan

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

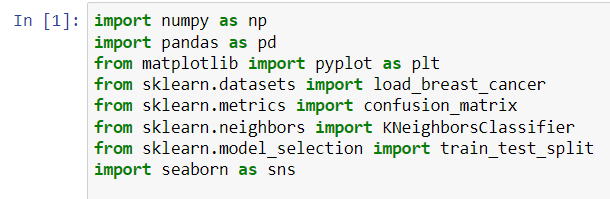
from sklearn.datasets import load\_breast\_cancer

from sklearn.metrics import confusion\_matrix

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

import seaborn as sns



2. Menampilkan column features

breast\_cancer.feature\_names

Text

Description automatically generated

3. Menampilkan class target

breast\_cancer.target\_names

Graphical user interface, text, application

Description automatically generated

4. Menyeseleksi features column dan mngubah class target menjadi categorical

X = pd.DataFrame(breast\_cancer.data, columns=breast\_cancer.feature\_names)

X = X[['mean area', 'mean compactness']]

y = pd.Categorical.from\_codes(breast\_cancer.target, breast\_cancer.target\_names)

y = pd.get\_dummies(y, drop\_first=True)

Text

Description automatically generated

5. Split dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)



6. Memanggil dan menjalankan algoritma KNN

knn = KNeighborsClassifier(n\_neighbors=5, metric='euclidean')

knn.fit(X\_train, y\_train)

A picture containing timeline

Description automatically generated

7. Melakukan prediksi dan menampilkan hasil prediksi

y\_pred = knn.predict(X\_test)

y\_pred

A picture containing text

Description automatically generated

8. Visualisasi dengan sns

sns.scatterplot(

x='mean area',

y='mean compactness',

hue='benign',

data=X\_test.join(y\_test, how='outer')

)

Chart, scatter chart

Description automatically generated

9. visualisasi dengan matplotlib

plt.scatter(

X\_test['mean area'],

X\_test['mean compactness'],

c=y\_pred,

cmap='coolwarm',

alpha=0.7

)

Chart, scatter chart

Description automatically generated

10. Menampilkan hasil Confusion matriks hasil prediksi

confusion\_matrix(y\_test, y\_pred)

Graphical user interface, text, application

Description automatically generated

Interpretasi Confusion matriks :

DIISISI

Interpretasi yang bisa dijelaskan adalah banyaknya data yang didapat dari berbagai sumber, maka pemrosesan dan analisis data yang efisien menjadi sulit, dikarenakan distribusi data yang tidak merata antar kelas.

**Elemen Komptensi 1.2.1**

**Tugas Laporan :**

1. Siapkan data iris.csv
2. Buka Jupyter Notebook
3. 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 in 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))

Screenshot tiap cell code:

|  |
| --- |
|  |

**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 hasil prediksi dengan confusion matriks. |  |  |  |  |  |  |  |
|  |  | Sangat Mudah |  |  | Tidak Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Mudah |  |  | Cukup Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | * Biasa |  |  | Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sulit |  |  | * Sangat Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sangat Sulit |  |  |  |  |
|  |  |  |  |  |  |  |