**Laporan Praktikum**

**Big Data**



DISUSUN OLEH:

MARCHEL YUSUF RUMLAWNG ARPIPI

(535210039)

Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Tarumanagara

JakartaTahun 2022

DAFTAR ISI

**DAFTAR ISIi**

**PRAKTIKUM I1**

Regresi Linear Sederhana1

Regresi Linear & Polynomial2

**PRAKTIKUM II3**

Regresi Linear dengan Regularisasi3

Naïve Bayes4

Regresi Logistik6

**PRAKTIKUM III8**

Clustering8

Decision Tree10

**PRAKTIKUM IV11**

k-Nearest Neighbors11

Neural Network12

**PRAKTIKUM V13**

**PRAKTIKUM 1**

**REGRESI**

1. Regresi

* Cara kerja/Langkah-langkah dalam analisis Regresi:

1. Tentukan Tujuan dari melakukan Analisis Regresi Linear Sederhana.

2. Identifikasikan Variabel Faktor Penyebab (Predictor) dan Variabel Akibat (Response).

3. Lakukan Pengumpulan Data.

4. Hitung X², Y², XY dan total dari masing-masingnya

5. Hitung a dan b berdasarkan rumus diatas.

6. Buatkan Model Persamaan Regresi Linear Sederhana.

7. Lakukan Prediksi atau Peramalan terhadap Variabel Faktor Penyebab atau Variabel Akibat.

* Contoh penerapan Regresi:

Yang paling sering menjadi kegunaan dari algritma ini adalah untuk memperdiksi harga penjualan dan juga prediksi berapa kilometer yang bisa dicapai dengan sekian liter bensin berdasarkan data-data yang telah ada yang di oleh menggunakan algoritma ini dengan mesin yang dilatih terlebih dahulu kemudian di test dengan test dataset yang telah disiapkan.

* Listing Programnya dan hasil/outputnya

**Regresi Linear Sederhana**

# Kode ini hanya untuk mengeplot data pada Contoh 1

# Untuk kode regresi, dapat dilihat di kode selanjutnya

#import numpy dan matplotlib

import numpy as np

from matplotlib import pyplot as plt

#data harga rumah Tabel 13.1

x = np.array([1400,1600,1700,1875,1100,1550,2350,2450,1425,1700])

y\_aktual = np.array([245,312,279,308,199,219,405,324,319,255])

#prediksi regresi linear

y\_prediksi = 98.2483 + 0.1098 \* x

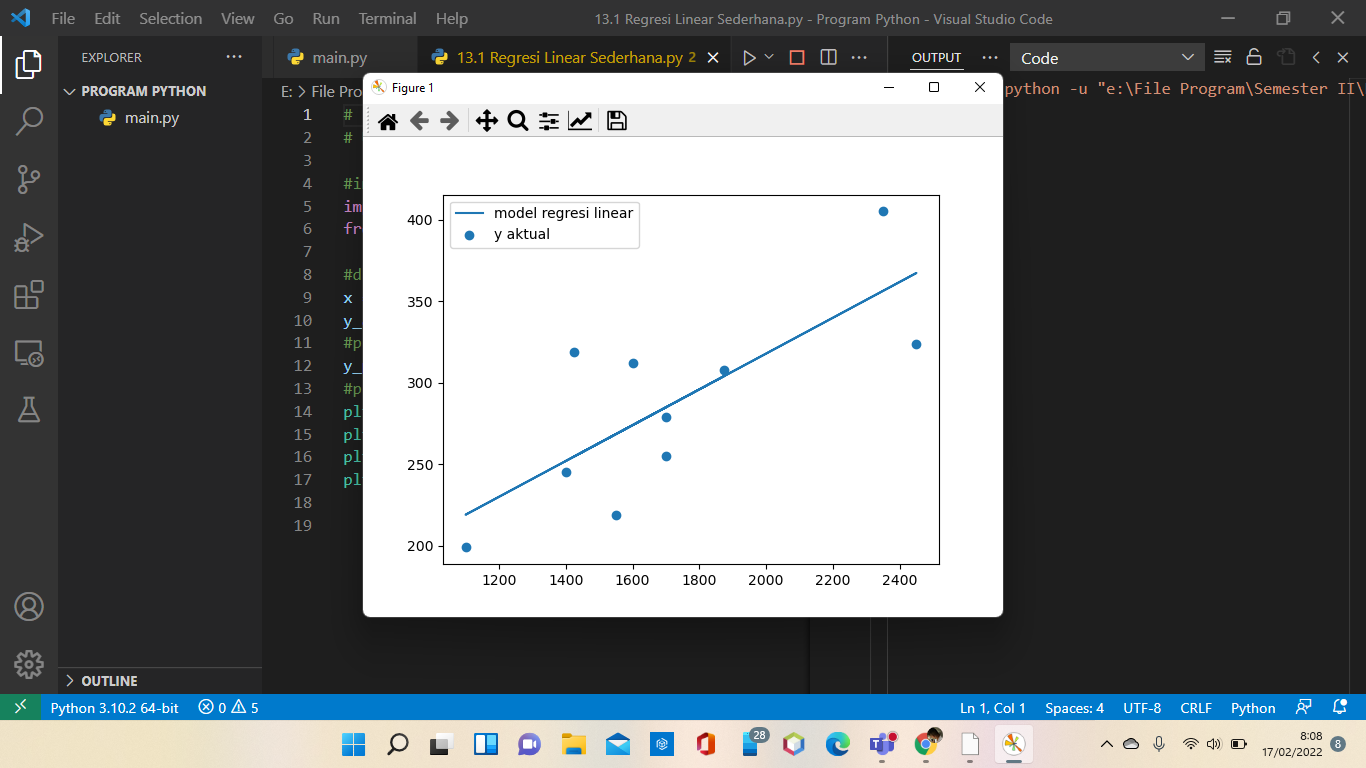
#plot ke grafik

plt.scatter(x, y\_aktual, marker="o", label = "y aktual")

plt.plot(x, y\_prediksi, label = "model regresi linear")

plt.legend()

plt.show()



**Regresi Linear & Polynomial**

#import numpy dan matplotlib

import numpy as np

from numpy.linalg import inv

from matplotlib import pyplot as plt

# data yang diberikan

x = np.array([0, 1.2, 2, 2.8, 4.3, 7, 7.8, 9])

y\_aktual = np.array(([3, 8, 10, 9.5, 1.5, -8, -3.4, 2.3]))

#membuat fungsi untuk menghitung koefisien regresi

def hitungKoefisienRegresi(x, n): # x adalah variabel input dan n = orde model

#membuat desain matriks

X = np.ones((x.shape[0], n+1))

for i in range(0, x.shape[0]):

for j in range(0, n+1):

X[i, j] = pow(x[i],(j))

#hitung koefisien regresi b = (X^TX)^-1 X^Ty (Persamaan 13.15)

b = inv(np.transpose(X).dot(X)).dot(np.transpose(X).dot(y\_aktual))

return X, b

#hitung prediksi output y menggunakan koefisien b

#dengan regresi linear, yaitu orde n = 1

X, b = hitungKoefisienRegresi(x, 1)

y\_prediksi\_liear = X.dot(b)

#dengan regresi polinomial, misal orde n = 4

X, b = hitungKoefisienRegresi(x, 4)

y\_prediksi\_polim = X.dot(b)

#plot data

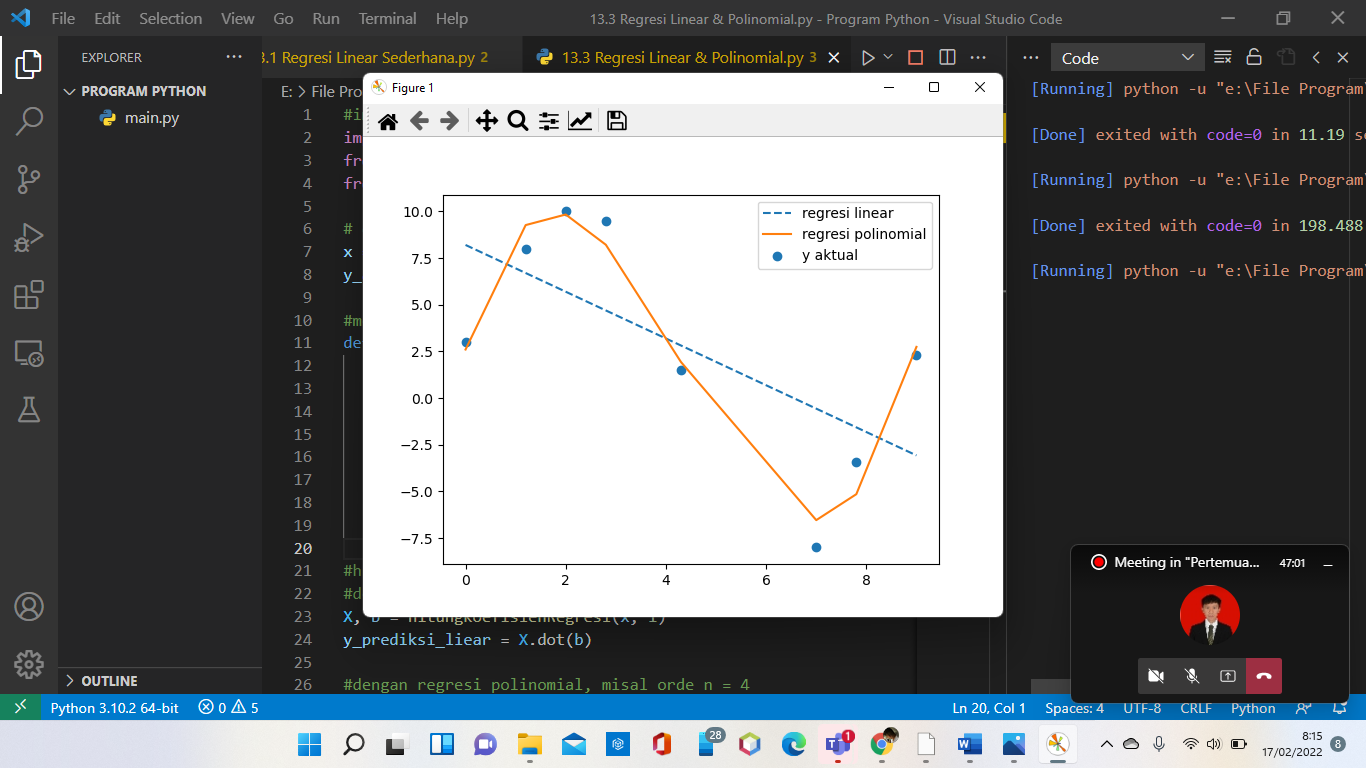
plt.scatter(x, y\_aktual, marker="o", label = "y aktual")

plt.plot(x, y\_prediksi\_liear, "--", label = "regresi linear")

plt.plot(x, y\_prediksi\_polim, label = "regresi polinomial")

plt.legend()

plt.show()



**PRAKTIKUM II**

**REGRESI (melanjutkan dari praktikum I)**

**Regresi dengan Regularisasi**

#import numpy dan matplotlib

import numpy as np

from numpy.linalg import inv

from matplotlib import pyplot as plt

# data yang diberikan

x = np.array([0, 1.2, 2, 2.8, 4.3, 7, 7.8, 9])

y\_aktual = np.array(([3, 8, 10, 9.5, 1.5, -8, -3.4, 2.3]))

#membuat fungsi untuk menghitung koefisien regresi

def hitungKoefisienRegresi(x, n, lamda): # x adalah variabel input dan n = orde model

#membuat desain matriks

X = np.ones((x.shape[0], n+1))

for i in range(0, x.shape[0]):

for j in range(0, n+1):

X[i, j] = pow(x[i],(j))

#hitung koefisien regresi b = (X'X+lamda\*I)^-1 X'y (persamaan 13.17)

b = inv(np.add(np.transpose(X).dot(X),

lamda\*np.identity(n+1))).dot(np.transpose(X).dot(y\_aktual))

return X, b

#prediksi output y dg regresi polinomial + regularisasi

# misal orde n = 4 dan konstanta regularisasi lamda = 1

X, b = hitungKoefisienRegresi(x, 4, 1)

y\_prediksi\_polim\_regu = X.dot(b)

#prediksi output y dg regresi polinomial tanpa regularisasi

# misal orde n = 4, sehingga konstanta regularisasi lamda = 0

X, b = hitungKoefisienRegresi(x, 4, 0)

y\_prediksi\_polim\_tanpa\_regu = X.dot(b)

#plot data

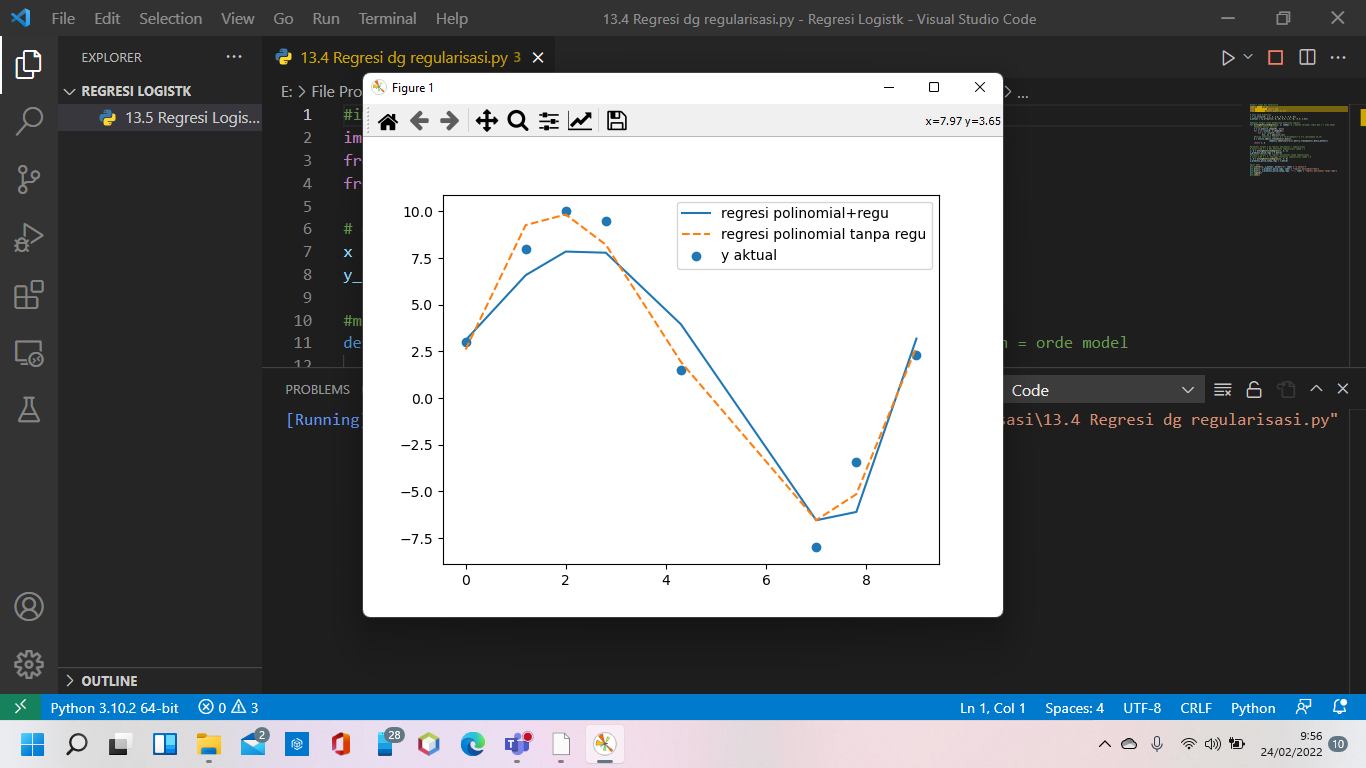
plt.scatter(x, y\_aktual, marker="o", label = "y aktual")

plt.plot(x, y\_prediksi\_polim\_regu, label = "regresi polinomial+regu")

plt.plot(x, y\_prediksi\_polim\_tanpa\_regu, "--", label = "regresi polinomial tanpa regu")

plt.legend()

plt.show()



**Naïve Bayes**

#mengimport numpy, pandas dan scikit-learn

import numpy as np

import pandas as pd

from sklearn.naive\_bayes import GaussianNB

#membaca dataset dari file ke pandas dataFrame

irisDataset=pd.read\_csv('dataset/klasifikasi\_dataset\_iris.csv',delimiter=',',header=0)

#mengubah kelas (kolom "Species") dari string ke unique-integer

irisDataset["Species"] = pd.factorize(irisDataset.Species)[0]

#menghapus kolom "Id"

irisDataset = irisDataset.drop(labels="Id", axis=1)

#mengubah dataframe ke array numpy

irisDataset = irisDataset.to\_numpy()

#membagi dataset, 40 baris data untuk training dan 20 baris data untuk testing

dataTraining = np.concatenate((irisDataset[0:40,:],irisDataset[50:90,:]), axis=0)

dataTesting = np.concatenate((irisDataset[40:50,:], irisDataset[90:100,:]), axis=0)

#memecah dataset ke input dan label

inputTraining = dataTraining[:,0:4]

inputTesting = dataTesting[:,0:4]

labelTraining = dataTraining[:,4]

labelTesting = dataTesting[:, 4]

#mendefinisikan klasifier Naive Bayes Gaussian

#Gaussian dipilih karena dataset kita kontinyu

model = GaussianNB()

#mentraining model

model = model.fit(inputTraining, labelTraining)

#memprediski input data testing

hasilPrediksi = model.predict(inputTesting)

print("label sebenarnya ", labelTesting)

print("hasil prediksi: ", hasilPrediksi)

#menghitung akurasi

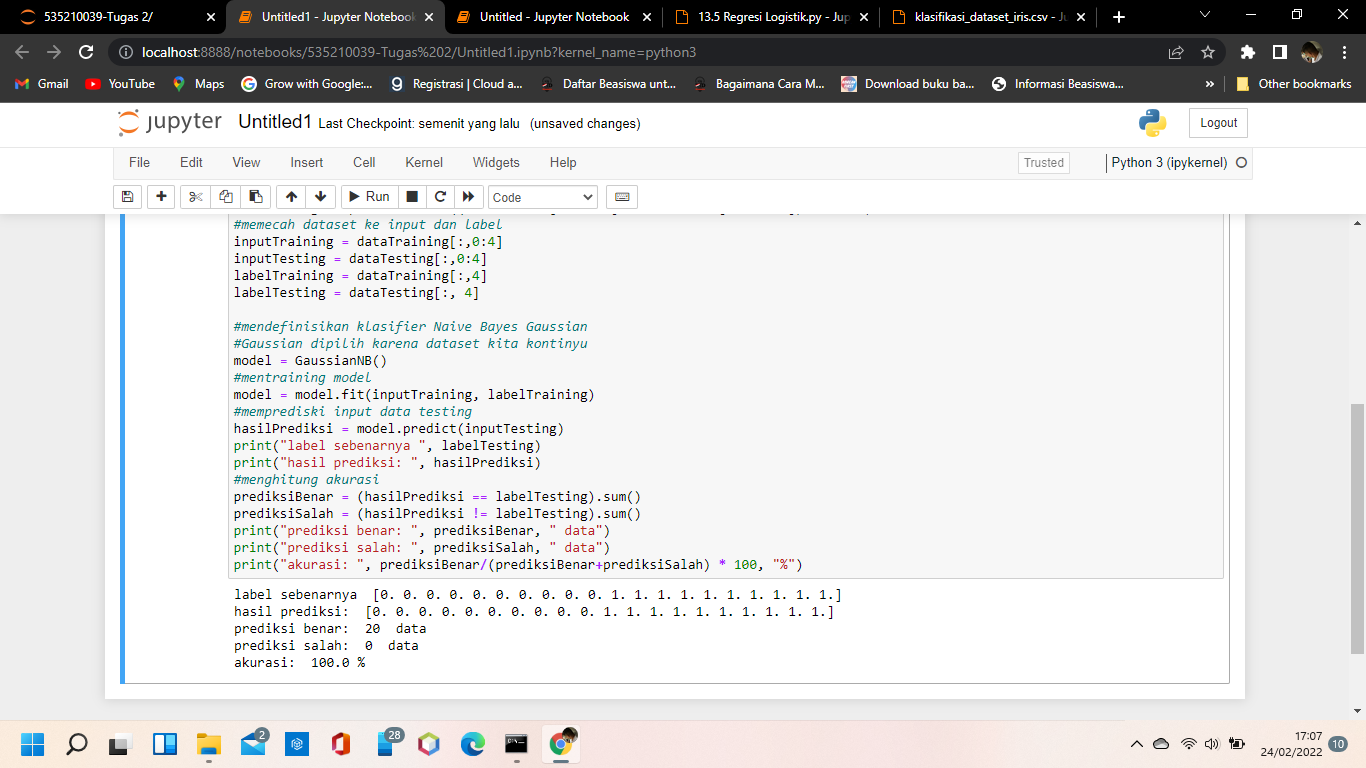
prediksiBenar = (hasilPrediksi == labelTesting).sum()

prediksiSalah = (hasilPrediksi != labelTesting).sum()

print("prediksi benar: ", prediksiBenar, " data")

print("prediksi salah: ", prediksiSalah, " data")

print("akurasi: ", prediksiBenar/(prediksiBenar+prediksiSalah) \* 100, "%")



**Regresi Logistik**

#import numpy dan pandas

import numpy as np

import pandas as pd

#membaca dataset dari file ke pandas dataFrame

irisDataset=pd.read\_csv('dataset/klasifikasi\_dataset\_iris.csv',delimiter=',',header=0)

#mengubah kelas (kolom "Species") dari string ke unique-integer

irisDataset["Species"] = pd.factorize(irisDataset.Species)[0]

#menghapus kolom "Id"

irisDataset = irisDataset.drop(labels="Id", axis=1)

#mengubah dataframe ke array numpy

irisDataset = irisDataset.to\_numpy()

#membagi dataset, 40 baris data untuk training dan 20 baris data untuk testing

dataTraining = np.concatenate((irisDataset[0:40,:],irisDataset[50:90,:]), axis=0)

dataTesting = np.concatenate((irisDataset[40:50,:], irisDataset[90:100,:]), axis=0)

#memecah dataset ke input dan label

inputTraining = dataTraining[:,0:4]

inputTesting = dataTesting[:,0:4]

labelTraining = dataTraining[:,4]

labelTesting = dataTesting[:, 4]

#mengubah array ke bentuk matriks

inputTraining = np.matrix(inputTraining)

inputTesting = np.matrix(inputTesting)

#fungsi untuk membuat desain matriks X

def buatMatriksDesainX(inputDataset):

bias = np.full((inputDataset.shape[0],1),1.0); bias = np.matrix(bias)

X = np.concatenate((bias, inputDataset), axis=1)

X = np.matrix(X)

return X

#fungsi sigmoid pada regresi logistik, lihat Persamaan (12.24)

def outputSigmoid(W, inputDataset)

hasilSigmoid = np.ndarray(shape = (0, 1))

for i in range(0,inputDataset.shape[0]):

singleBias = np.full((1,1),1.0)

x = np.concatenate((singleBias,inputDataset[i,:].T), axis=0)

fungsiSigmoid = 1.0/(1.0 + np.exp(-np.transpose(W).dot(x)))

hasilSigmoid=np.concatenate((hasilSigmoid,fungsiSigmoid) axis=0)

hasilSigmoid = np.matrix(hasilSigmoid)

return hasilSigmoid

#menginisiasi nilai W dan F(W'x)

initial\_W = np.full((inputTesting.shape[1] + 1, 1), 0.1)

outSigm = outputSigmoid(initial\_W, inputTraining)

#fungsi gradient descent

def gradDescent(initialWeight, inputDataset, label, step, iteration, tolerance):

w\_lama = initialWeight

designMat = buatMatriksDesainX(inputDataset)

label = np.matrix(label).T

for i in range(0, iteration):

outSigm = outputSigmoid(w\_lama, inputDataset)

#update W menggunakan Persamaan (12.23)

w\_baru=w\_lama-step\*np.transpose(designMat).dot(np.subtract(outSigm,label))

deltaW = np.subtract(w\_baru, w\_lama)

delta = np.abs(deltaW[:,:].sum())

if delta < tolerance:

break

w\_lama = w\_baru

return w\_baru

#definisikan hyperparameter dari gradient descentnya

tolerance = 0.00001

konstantaStep = 0.3

banyaknyaIterasi = 400

w = gradDescent(initial\_W, inputTraining, labelTraining, konstantaStep, banyaknyaIterasi, tolerance)

#testing: masukkan nilai W dan input x-nya ke fungsi sigmoid kembali

#kemudian bulatkan (agar <0,5 masuk ke kelas 1 (label=0) dan >0,5 masuk ke kelas 2 (label=1))

hasilPrediksi = np.round(outputSigmoid(w, inputTesting))

#mengubah matriks ke 1D array

hasilPrediksi = np.asarray(hasilPrediksi).ravel()

#menghitung akurasi

print("label sebenarnya ", labelTesting)

print("hasil prediksi ", hasilPrediksi)

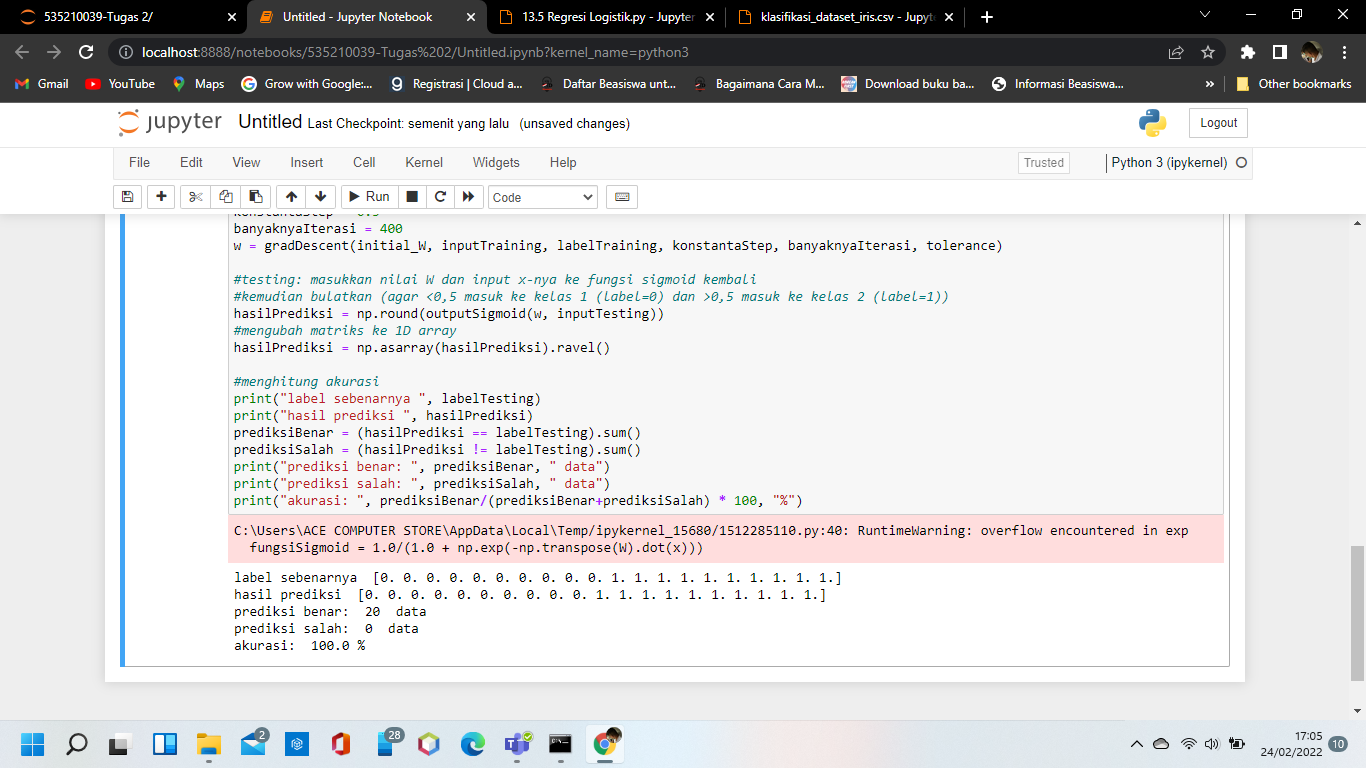
prediksiBenar = (hasilPrediksi == labelTesting).sum()

prediksiSalah = (hasilPrediksi != labelTesting).sum()

print("prediksi benar: ", prediksiBenar, " data")

print("prediksi salah: ", prediksiSalah, " data")

print("akurasi: ", prediksiBenar/(prediksiBenar+prediksiSalah) \* 100, "%")



**PRAKTIKUM III**

**Clustering dan Decision Tree (untuk Regresi dengan regularisasi dan Naïve Bayes telah dijelaskan pada praktikum II)**

1. Clustering

* Cara kerjanya : Dengan mengambil salah satu contoh metode

clustering yang paling mudah dijelaskan untuk langkah-langkah atau cara kerjanya agar dapat dimengerti secara umum adalah hierarchical clustering. Berikut cara kerjanya:

-Identifikasi item dengan jarak terdekat

-Gabungkan item itu kedalam satu cluster

-Hitung jarak antar cluster

-Ulangi dari awal sampai semua terhubung.

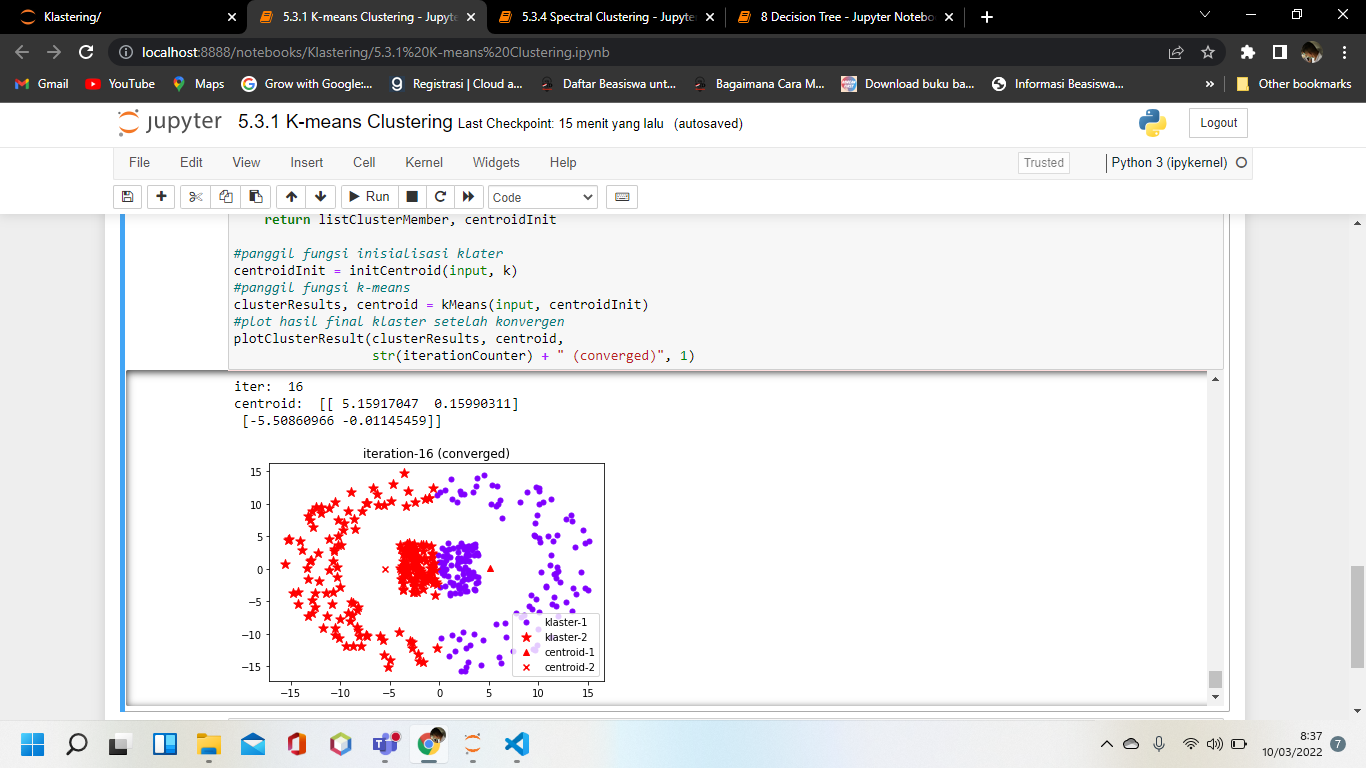
* Contoh penerapan Clustering:

a). Clustering merupakan metode segmentasi data yang sangat berguna dalam prediksi dan analisa masalah bisnis tertentu. Misalnya Segmentasi pasar, marketing dan pemetaan zonasi wilayah.

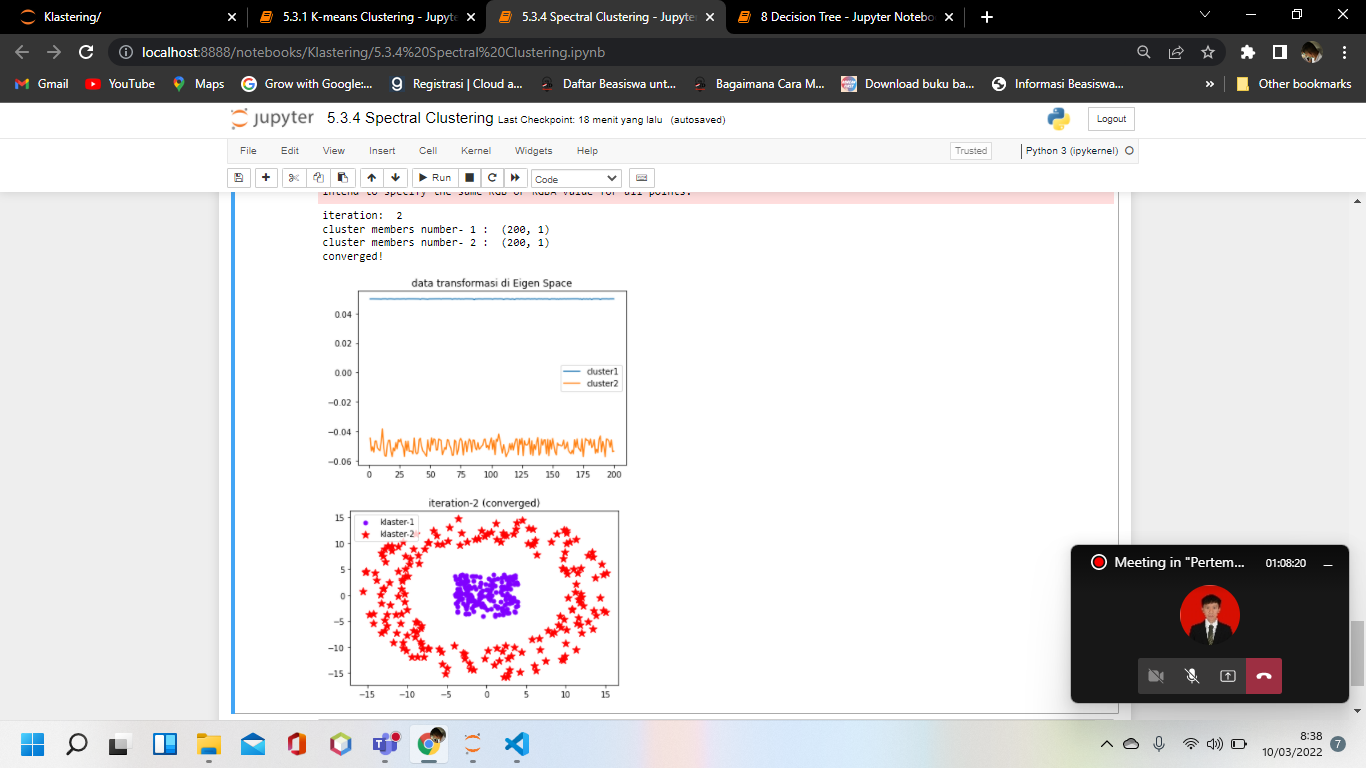
b). Identifikasi obyek dalam bidang berbagai bidang seperti computer vision dan image processing.

* Listing Programnya dan hasil outputnya

**K-Means Clustering**



**Spectral Clustering**



1. Decision Tree

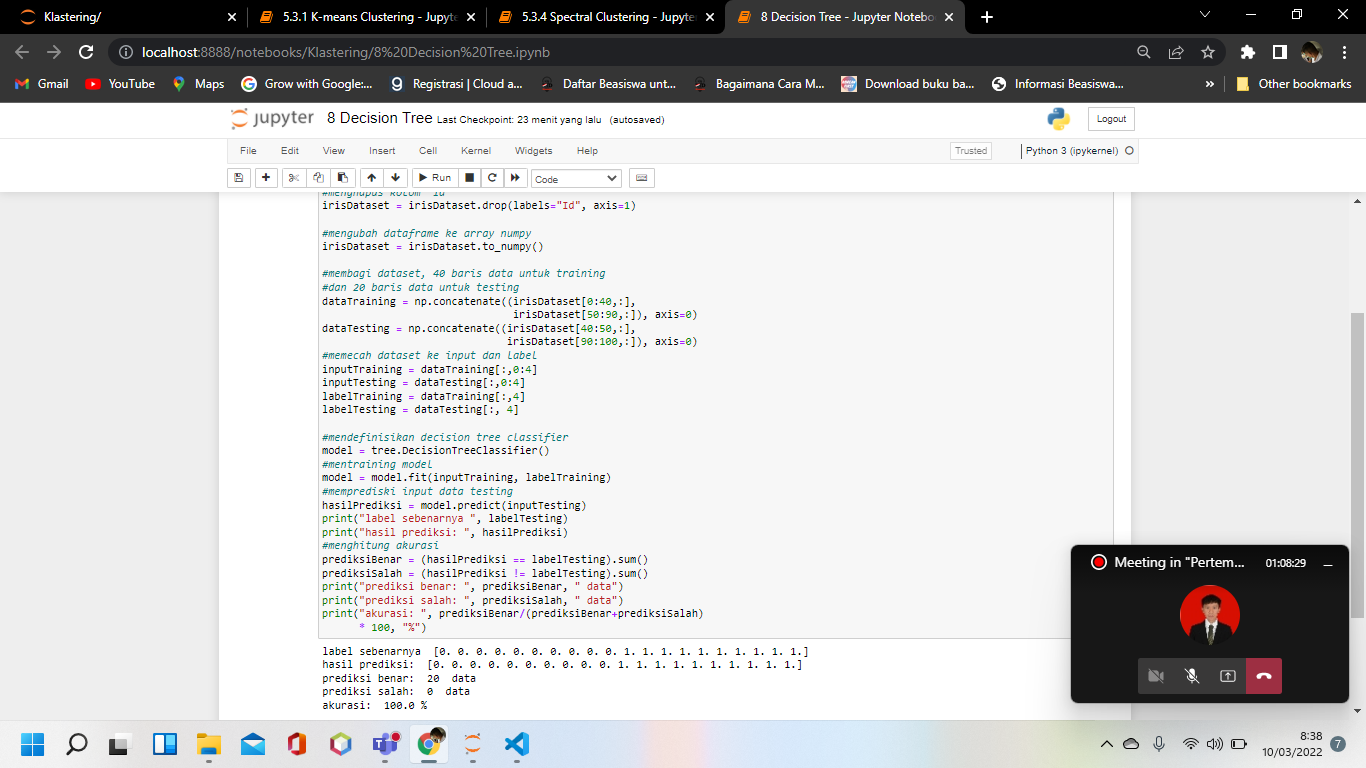
* Cara kerjanya : Umumnya, decision tree dimulai dengan satu

node atau simpul. Kemudian, node tersebut bercabang untuk menyatakan pilihan-pilihan yang ada. Selanjutnya, setiap cabang tersebut akan memiliki cabang-cabang baru. Oleh karenanya, metode ini disebut 'tree' karena bentuknya menyerupai pohon yang memiliki banyak cabang.

* Contoh penerapan Decision Tree

Secara singkat, decision tree untuk membuat model prediksi *binary classification* (“ya” dan “tidak”), bisa juga :dipakai untuk menentukan output yang sifatnya *continuous*. Selain itu decision tree juga menjadi tools yang populer untuk machine learning dan AI ketika digunakan sebagai algaritma, misalnya untuk mengkategorikan data maupun hal lainnya.

* Listing Program dan hasil outputnya:



**PRAKTIKUM IV**

**k-Nearest Neighbors (kNN) dan Neural Network**

1. k-Nearest Neightbors (kNN)

* Cara kerjanya : Prinsip kerja K-Nearest Neighbor (KNN) adalah

mencari jarak terdekat antara data yang akan dievaluasi dengan k tetangga (neighbor) terdekatnya dalam data pelatihan(training) . Dengan k merupakan banyaknya tetangga terdekat. Tujuan dari algoritma ini adalah untuk mengklasifikasikan obyek baru berdasarkan atribut dan sample-sample dari training data.

- Tentukan jumlah tetangga (K) yang akan digunakan untuk pertimbangan penentuan kelas.

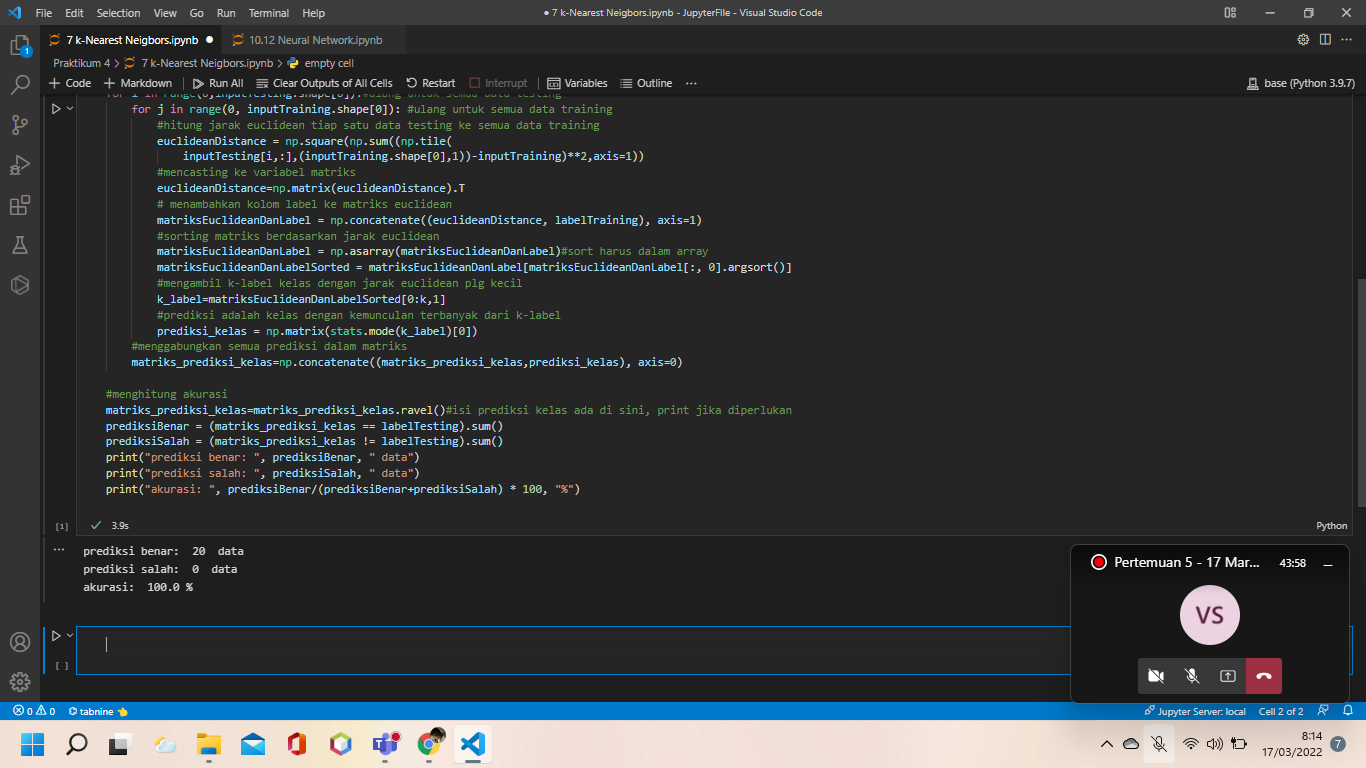
- Hitung jarak dari data baru ke masing-masing data point di dataset.

- Ambil sejumlah K data dengan jarak terdekat, kemudian tentukan kelas dari data baru tersebut.

* Contoh penerapan kNN:

k-Nearest Neighbors dapat digunakan untuk klasifikasi terhadap objek dengan cara menghitung jarak antar objek dan tetangganya dengan Euclidean Distance. Algoritma kNN dapat diterapkan juga pada Data Mining dan dapat melakukan prediksi pada hal-hal dalam kehidupan sehari-hari seperti memprediksi harga cabai rawit di suatu daerah atau dalam dunia ekonomi kNN bisa dipakai untuk penentuan risiko kredit kepemilikan kendaraan bermotor bahkan banyak hal lain akan membutuhkan kNN sebagai solusi penyelesaiannya.

* Listing Program dan hasil outputnya:



1. Neural Network

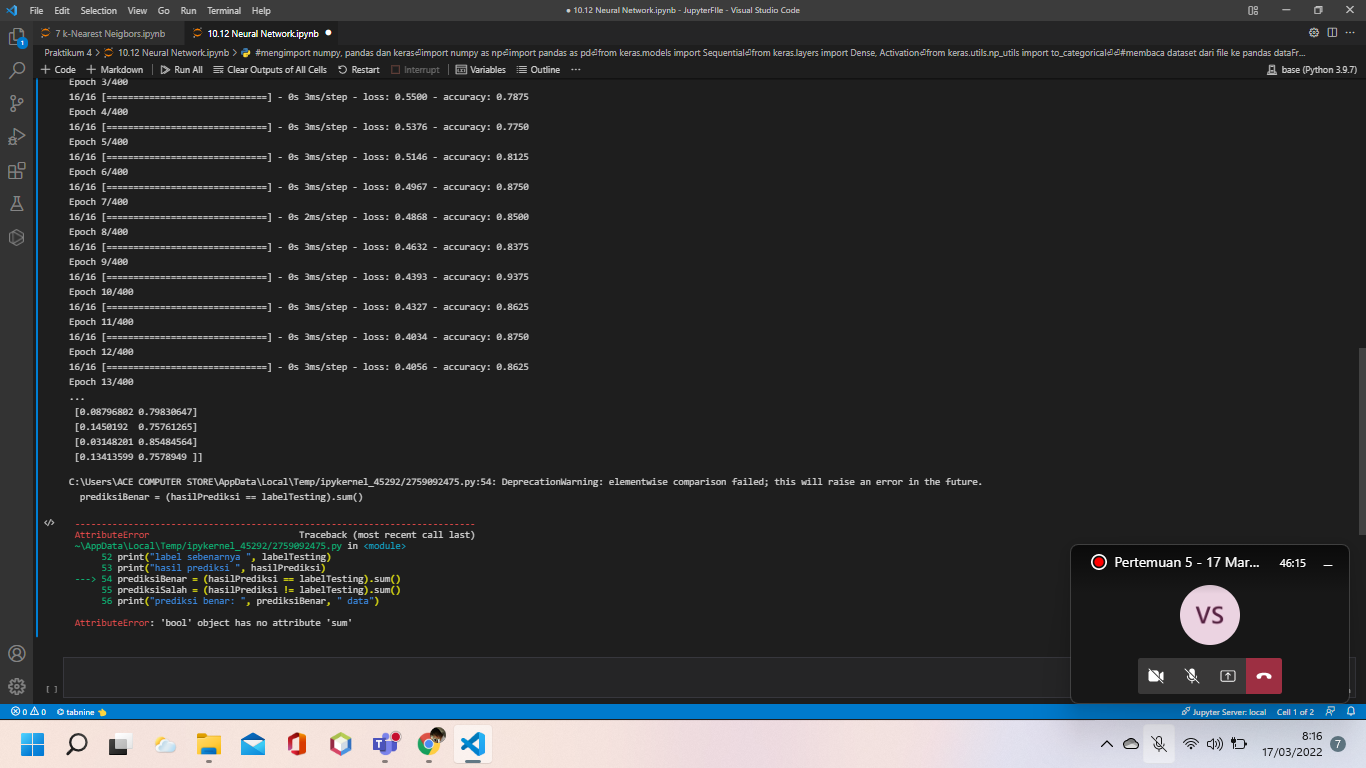
* Cara kerjanya : Cara kerja Neural Network dapat dianalogikan

sebagaiman halnya manusia belajar dengan mengunakan contoh atau yang disebut sebagai supervised learning. Sebuah Neural Network dikonfigurasi untuk aplikasi tertentu, seperti pengenalan pola atau klasifikasi data, dan kemudian disempurnakan melalui proses pembelajaran.

* Contoh penerapan Neural Network:

Banyak sekali program atau aplikasi yang menerapkan jaringan neural network ini, misalnya dalam bidang keuangan, analitik bisnis, hingga pemeliharaan produk. Ada juga yang memakai neural network sebagai alat prediksi seperti untuk memberi tahu calon investor mana instrumen investasi yang cocok untuk dan juga mengadopsinya untuk memperkirakan gerak entitas saham.

* Listing Program dan hasil outputnya:



**PRAKTIKUM V**

**(dikumpulkan dalam bentuk link dokumen dari Ms Teams)**

Link dokumen:

<https://untarid.sharepoint.com/:b:/s/GN2122-TK13025-BigDataA-BagusMulyawan/EWgHPv2CzKhAhAADZiwfspIB1BYX7AE9NovQXcKbx2T_kg?e=5GsfTg>