

LAPORAN PENGANTAR KECERDASAN BUATAN

TUGAS PEMROGRAMAN 3



Oleh:

Kelompok 15

Anggota:

Khalilullah Al Faath - 1301204376 / IF-44-08

Mirai Tsuchiya - 1301203555 / IF-44-08

M Ivan Irsanto - 1301200467 / IF-44-08

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022

Daftar Isi

Table of Contents

Daftar Isi	2
1. Tinjauan Pustaka	4
1.1. Definisi Learning	4
1.1.1. Supervised Learning	4
1.1.2. Semi-supervised Learning.....	4
1.1.3. Unsupervised Learning	4
1.1.4. Reinforcement Learning	4
1.2. K-Nearest Neighbors	4
1.2.1. Karakteristik KNN	5
1.2.2. Tahapan-tahapan	5
1.2.3. Kelebihan dan Kekurangan KNN	5
1.2.4. Kekurangan dari algoritma KNN	5
2. Hal-hal yang Dapat Diobservasi	6
2.1. Pendahuluan.....	6
2.1.1. Mengimport library yang dibutuhkan	6
2.1.2. Membaca dataset.....	6
2.1.3. Hasil dataset	6
2.2. EDA (Explotory Data Analysist)	6
2.2.1. Deskripsi Dataset	7
2.2.2. Tampilan Tiga Data Teratas	7
2.2.3. Tampilan Tiga Data Terbawah	7
2.2.4. Tampilan 3 sample data	8
2.2.5. Drop Kolom yang Tidak Dibutuhkan	8
2.2.6. Korelasi Antar Variabel	9
2.3. Pre-Processing.....	11
2.3.1. Menghitung dan memepetkan nilai outliers.....	11
2.3.2. Scaling.....	12
2.4. KNN.....	16
2.4.1. Perhitungan Jarak.....	16
3. Hasil Output	19

3.1.	Import data testing	19
3.2.	Drop kolom yang tidak dibutuhkan	20
3.3.	Scaling dataset	21
3.4.	Output boxplot	22
3.5.	KNN pada data training untuk semua data terhadap data testing untuk mengisi kolom y target test yang masih kosong.....	23
3.6.	Mengisi dataset test dengan kolom prediksi	23
3.7.	Ekspor dataset	23
4.	Eksplorasi.....	24
4.1.	Pengecekan head training set terpilih dan validation set terpilih	24
4.2.	Algoritma KNN pada data random	25
4.3.	Hasil Akurasi pada data random	26
5.	Kesimpulan	27
6.	Resources	28
7.	Daftar Pustaka.....	28

1. Tinjauan Pustaka

1.1. Definisi Learning

Learning adalah salah satu blok bangunan dasar solusi kecerdasan buatan atau AI. Dari sudut pandang konseptual, learning adalah proses yang meningkatkan pengetahuan dari sebuah program AI dengan melakukan pengamatan terhadap lingkungannya, dalam hal ini adalah inputannya. Terdapat 4 tipe learning yang diketahui, yaitu supervised learning, semi-supervised learning, unsupervised learning dan reinforcement learning.

1.1.1. Supervised Learning

Model Supervised learning menggunakan umpan balik eksternal untuk fungsi pembelajaran yang memetakan masukan ke pengamatan keluaran. Dalam model tersebut, lingkungan eksternal bertindak sebagai "guru" dari algoritma AI.

1.1.2. Semi-supervised Learning

Semi-supervised Learning menggunakan kumpulan data yang dikuratori dan diberi label dan mencoba menyimpulkan label/atribut baru pada kumpulan data baru. Model Semi-supervised Learning adalah jalan tengah antara model supervised dan unsupervised.

1.1.3. Unsupervised Learning

Model Unsupervised berfokus pada mempelajari pola dalam data input tanpa umpan balik eksternal. Clustering adalah contoh klasik dari model unsupervised.

1.1.4. Reinforcement Learning

Model Reinforced Learning menggunakan dinamika yang berlawanan seperti penghargaan dan hukuman untuk "memperkuat" berbagai jenis pengetahuan. Jenis teknik pembelajaran ini menjadi sangat populer dalam solusi AI modern.

1.2. K-Nearest Neighbors

K-Nearest Neighbors atau disingkat sebagai KNN adalah adalah suatu metode yang menggunakan algoritma supervised learning dimana hasil dari sampel

uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Mengklasifikasi objek baru berdasarkan atribut dan sampel latih adalah tujuan dari algoritma ini.

1.2.1. Karakteristik KNN

- a) Klasifikasi langsung dari tetangga terdekat.
- b) Bekerja secara lokal.
- c) Bisa untuk data apapun.
- d) Bisa untuk klasifikasi dan regresi.
- e) Instance Based Learning (IBL).
- f) Lazy learner.
- g) Tidak melakukan proses belajar dari data latih.

1.2.2. Tahapan-tahapan

- a) Menentukan parameter k (jumlah tetangga terdekat).
- b) Menghitung jarak terhadap data training yang diberikan.
- c) Mengurutkan hasil perhitungan jarak.
- d) Mengambil sejumlah k data terdekat.

1.2.3. Kelebihan dan Kekurangan KNN

Terdapat kelebihan dari penggunaan algoritma KNN, berikut adalah kelebihan dari KNN :

- a) Algoritma k-NN kuat dalam mentraining data yang noisy.
- b) Algoritma k-NN sangat efektif jika datanya besar.
- c) Mudah diimplementasikan.

1.2.4. Kekurangan dari algoritma KNN

Sementara itu,terdapat juga kekurangan dari algoritma KNN :

- a) Algoritma K-NN perlu menentukan nilai parameter K.
- b) Sensitif pada data pencilan.
- c) Rentan pada variabel yang non-informatif.

2. Hal-hal yang Dapat Diobservasi

Hal-hal yang dapat diobservasi dari dataset, antara lain pendahuluan, EDA (Exploratory Data Analyst), pre-processing, KNN dengan k-Fold cross validation, dan KNN dengan data random.

2.1. Pendahuluan

Pada tahap ini dilakukan beberapa hal sebagai berikut.

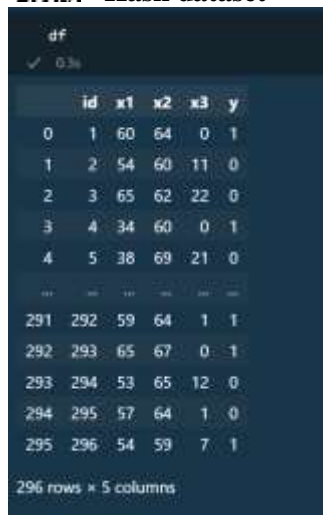
2.1.1. Mengimport library yang dibutuhkan

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
```

2.1.2. Membaca dataset

```
df = pd.read_excel("https://github.com/khalilullahalfaath/
AI_Tubes-03-Learning/blob/
19d11d6e2fa2d540afdc0cec3308b82c664d29f7/traintest.xlsx?
raw=true")
```

2.1.3. Hasil dataset



	id	x1	x2	x3	y
0	1	60	64	0	1
1	2	54	60	11	0
2	3	65	62	22	0
3	4	34	60	0	1
4	5	38	69	21	0
...
291	292	59	64	1	1
292	293	65	67	0	1
293	294	53	65	12	0
294	295	57	64	1	0
295	296	54	59	7	1

2.2. EDA (Explatory Data Analyst)

Exploratory Data Analysis (EDA) adalah pendekatan untuk menganalisis data menggunakan teknik visual. Ini digunakan untuk menemukan tren, pola, atau untuk memeriksa asumsi dengan bantuan ringkasan statistik dan representasi grafis. Dalam

pembuatan EDA, kami mengklasifikasikan info, deskripsi, tampilan 3 data teratas, tampilan 3 data terbawah, 3 data sample, drop kolom yang tidak dibutuhkan terkait datasetnya, dan korelasi antar variabel.

2.2.1. Deskripsi Dataset

Info dataset yang kami peroleh dari pembuatan tugas besar 3 adalah sebagai berikut:

```
df.describe()
```

	id	x1	x2	x3	y
count	296.000000	296.000000	296.000000	296.000000	296.000000
mean	148.500000	52.462838	62.881757	4.111486	0.736486
std	85.592056	10.896367	3.233753	7.291816	0.441285
min	1.000000	30.000000	58.000000	0.000000	0.000000
25%	74.750000	44.000000	60.000000	0.000000	0.000000
50%	148.500000	52.000000	63.000000	1.000000	1.000000
75%	222.250000	61.000000	65.250000	5.000000	1.000000
max	296.000000	83.000000	69.000000	52.000000	1.000000

2.2.2. Tampilan Tiga Data Teratas

Dengan syntax “df.head(3)”, kami menampilkan 3 dataset teratas.

```
df.head(3)
```

	id	x1	x2	x3	y
0	1	60	64	0	1
1	2	54	60	11	0
2	3	65	62	22	0

2.2.3. Tampilan Tiga Data Terbawah

Dengan syntax “df.tail(3)”, kami menampilkan 3 dataset terbawah.

```
df.tail(3)
```

	id	x1	x2	x3	y
293	294	53	65	12	0
294	295	57	64	1	0
295	296	54	59	7	1

2.2.4. Tampilan 3 sample data

Dengan syntax “df.sample(3), kami menampilkan 3 data sample yang terdapat dari dataset.

```
df.sample(3)
```

	id	x1	x2	x3	y
61	62	54	66	0	1
73	74	63	62	0	1
20	21	57	64	9	1

2.2.5. Drop Kolom yang Tidak Dibutuhkan

Kami melakukan drop pada kolom yang tidak dibutuhkan, yaitu kolom “id”. Dapat dilihat di gambar berikut:

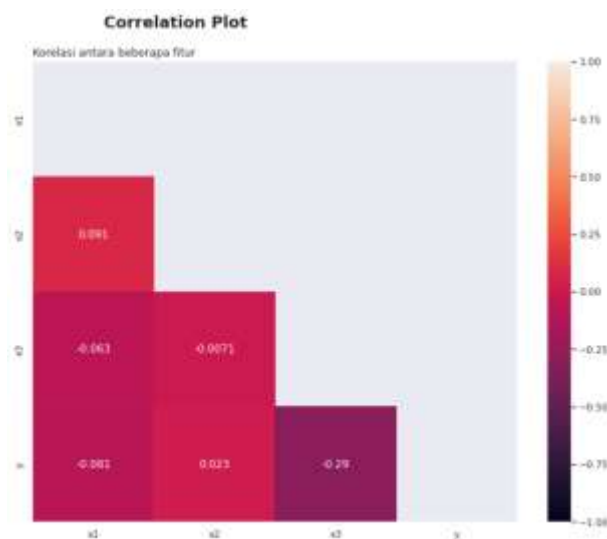
```
df.drop(columns = ['id'], axis = 1, inplace = True)
df
```

	x1	x2	x3	y
0	60	64	0	1
1	54	60	11	0
2	65	62	22	0
3	34	60	0	1
4	38	69	21	0

2.2.6. Korelasi Antar Variabel

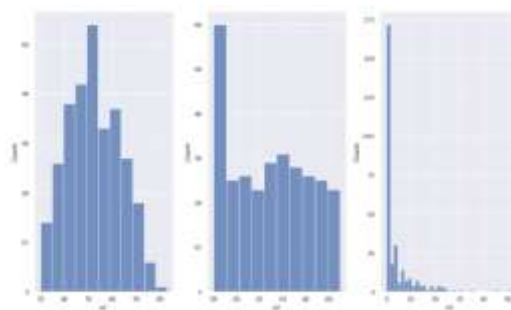
Correlation plot

```
fig, ax = plt.subplots(figsize=(10, 8))
mask = np.triu(np.ones_like(df.corr(), dtype=bool))
heatmap = sns.heatmap(df.corr(), mask=mask, vmin=-1, vmax=1, annot=True)
plt.suptitle("Correlation Plot", ha='left', x=0.155, y=1.04, fontsize=18, fontweight='bold')
plt.title("Korelasi antara beberapa fitur", loc='left', fontsize=12)
plt.tight_layout()
plt.show()
```



dapat dilihat bahwa dataset di atas memiliki variabel yang tidak berkorelasi satu sama lain

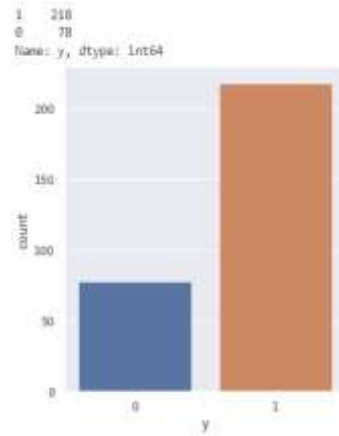
Histogram



```
fig, axes = plt.subplots(ncols=3)
sns.set(rc={'figure.figsize':(50,10)})
sns.histplot(df['x1'], ax=axes[0])
sns.histplot(df['x2'], ax=axes[1])
sns.histplot(df['x3'], ax=axes[2])
```

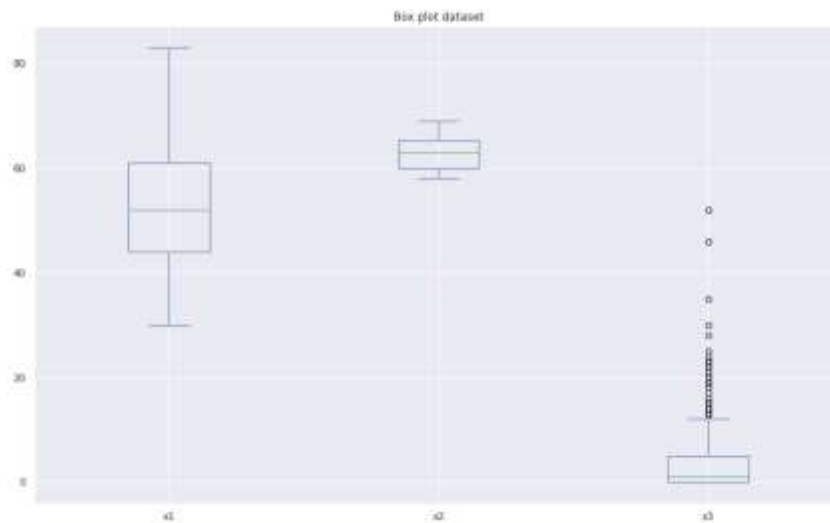
Countplot untuk nilai y

```
print(df['y'].value_counts())
sns.catplot(x='y', data=df, kind='count')
plt.show()
```



Boxplot

```
plt.rcParams["figure.figsize"] = [13, 8]
plt.rcParams["figure.autolayout"] = True
ax = df[['x1', 'x2', 'x3']].plot(kind='box', title='Box plot dataset')
```



Dapat dilihat bahwa terdapat banyak data pencilan. Yang akan diproses selanjutnya di tahap pre-processing.

2.3. Pre-Processing

Pre-processing kami melakukan beberapa hal sebagai berikut.

2.3.1. Menghitung dan memepetkan nilai outliers

```
def hitungOutliers(df):  
    q1 = df.quantile(0.25)  
    q3 = df.quantile(0.75)  
    IQR = q3 - q1  
    outliers = df[((df < (q1-1.5*IQR)) | (df > (q3+1.5*IQR)))]  
    return outliers  
  
outliers = hitungOutliers(df["x3"])  
percentage = len(outliers)/len(df["x3"])*100  
print("number of outliers: " + str(len(outliers)))  
print("max outlier value: " + str(outliers.max()))  
print("min outlier value: " + str(outliers.min()))  
print("Outliers percentage: " + str(float(f'{percentage:.2f}'))  
      + "%")
```

```
number of outliers: 34  
max outlier value: 52  
min outlier value: 13  
Outliers percentage: 11.49%
```

```
#comment code di bawah ketika tidak ingin memepetkan nilai  
outliers  
  
Q1 = df["x3"].quantile(0.25)  
Q3 = df["x3"].quantile(0.75)  
  
IQR = Q3 - Q1  
  
LB = Q1 - (IQR * 1.5)  
UB = Q3 + (IQR * 1.5)  
  
df.loc[df['x3'] > UB, "x3"] = UB  
df.loc[df['x3'] < LB, "x3"] = LB
```

```
plt.rcParams["figure.figsize"] = [13, 8]  
plt.rcParams["figure.autolayout"] = True  
ax = df[['x1', 'x2', 'x3']].plot(kind='box', title='box plot  
data setelah memepetkan nilai outliers')
```



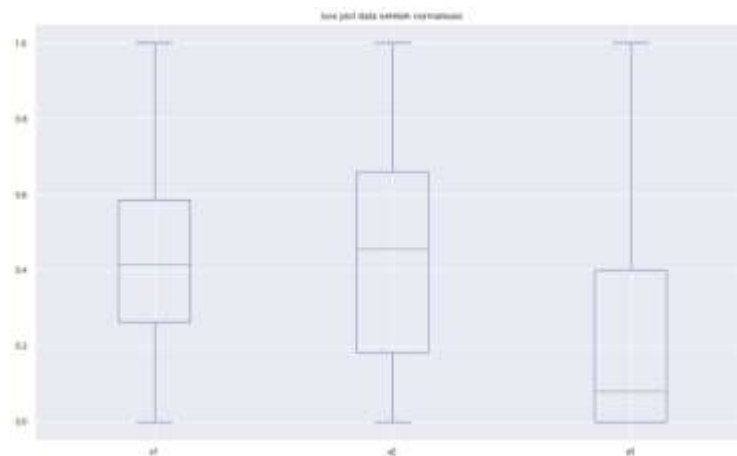
2.3.2. Scaling

Terdapat tiga metode scaling yang ada pada kelompok kami:

2.3.2.1. Normalisasi

```
#normalisasi
def minMaxScaling(df) :
    return (df.iloc[:,4] - df.iloc[:,4].min()) / (df.iloc[:,
:4].max() - df.iloc[:,4].min())
```

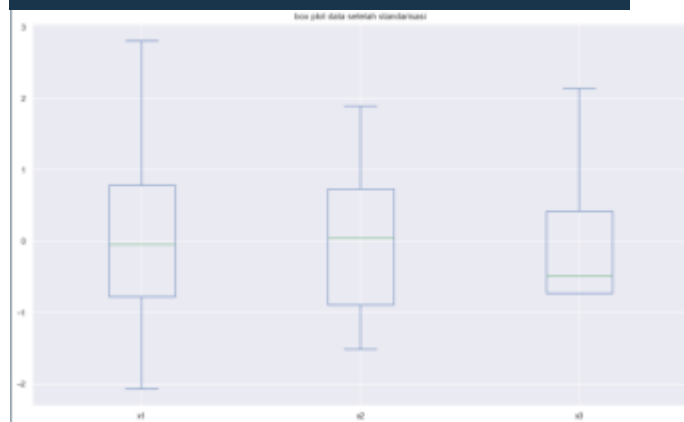
dataNormalized				
✓ 0.4s				
	x1	x2	x3	y
0	0.566038	0.545455	0.00	1.0
1	0.452830	0.181818	0.88	0.0
2	0.660377	0.363636	1.00	0.0
3	0.075472	0.181818	0.00	1.0
4	0.150943	1.000000	1.00	0.0
...
291	0.547170	0.545455	0.08	1.0
292	0.660377	0.818182	0.00	1.0
293	0.433962	0.636364	0.96	0.0
294	0.509434	0.545455	0.08	0.0
295	0.452830	0.090909	0.56	1.0



2.3.2.2. Standarisasi

```
def standardScaling(df) :  
    return ((df.iloc[:,4] - df.iloc[:,4].mean()) / df.iloc[:,4].std())
```

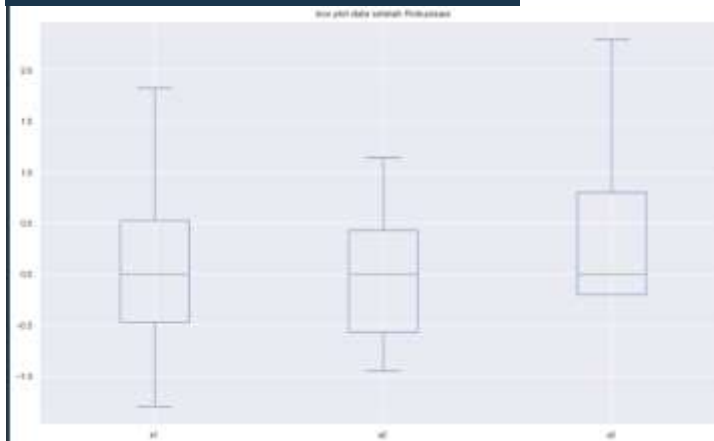
dataStandarized				
✓	0.8s			
	x1	x2	x3	y
0	0.691713	0.345804	-0.722610	0.597151
1	0.141071	-0.891149	1.801873	-1.668961
2	1.150582	-0.272673	2.146121	-1.668961
3	-1.694403	-0.891149	-0.722610	0.597151
4	-1.327308	1.891995	2.146121	-1.668961
...
291	0.599940	0.345804	-0.493112	0.597151
292	1.150582	1.273518	-0.722610	0.597151
293	0.049297	0.655042	2.031372	-1.668961
294	0.416392	0.345804	-0.493112	-1.668961
295	0.141071	-1.200388	0.883879	0.597151



2.3.2.3. Robust Scaling

```
def robustScaling(df):  
    return ((df.iloc[:,4] - df.iloc[:,4].median()) / (df.  
   .iloc[:,4].quantile(0.75) - df.iloc[:,4].quantile(0.25)))
```

dataRobusted				
✓ 0.8s				
	x1	x2	x3	y
0	0.470588	0.190476	-0.2	0.0
1	0.117647	-0.571429	2.0	-1.0
2	0.764706	-0.190476	2.3	-1.0
3	-1.058824	-0.571429	-0.2	0.0
4	-0.823529	1.142857	2.3	-1.0
...
291	0.411765	0.190476	0.0	0.0
292	0.764706	0.761905	-0.2	0.0
293	0.058824	0.380952	2.2	-1.0
294	0.294118	0.190476	0.0	-1.0
295	0.117647	-0.761905	1.2	0.0



2.3.2.4. Split Dataset

Berikut adalah tampilan split dataset kelompok kami. Dataset akan dipisah menjadi training set dan validation set untuk menghitung akurasi tiap nilai k.

```
def splitDependent(df):  
    x = df.iloc[:, [0,1,2]].values  
    y = df.iloc[:, [3]].values  
    return x,y
```

```
x,y = splitDependent(dataNormalized.head(10))  
print(x)  
print()  
print(y)  
✓ 0.4s  
[[0.56603774 0.54545455 0.      ]  
 [0.45283019 0.18181818 0.88    ]  
 [0.66037736 0.36363636 1.      ]  
 [0.0754717  0.18181818 0.      ]  
 [0.1509434  1.          1.      ]  
 [0.05660377 0.          0.8     ]  
 [0.62264151 0.27272727 0.      ]  
 [0.50943396 0.54545455 0.      ]  
 [0.30188679 0.          0.24    ]  
 [0.24528302 0.63636364 0.      ]]
```

```
[[1.]  
 [0.]  
 [0.]  
 [1.]  
 [0.]  
 [1.]  
 [1.]  
 [1.]  
 [1.]  
 [1.]]
```

2.4. KNN

2.4.1. Perhitungan Jarak

Fungsi untuk sort nilai training

```
def sortNilaiTraining(d):  
    return d[0]
```

✓ 0.4s

Terdapat tiga metode perhitungan jarak pada program kami, yaitu:

2.4.1.1. Euclidean

```
def euclidean(xTrain,yTrain,xVal):  
    result = []  
    for x in xVal:  
        resultPerTraining = []; indexTraining = 0  
        for j in xTrain:  
            d = (x[0]-j[0])**2 + (x[1]-j[1])**2 + (x[2]-j[2])**2  
            #print(x)  
            #print(yTrain)  
            hasilPerTrainingData = [math.sqrt(d),yTrain[indexTraining]]  
            #print(resultPerTraining)  
            resultPerTraining.append(hasilPerTrainingData)  
            indexTraining = indexTraining + 1  
        resultPerTraining.sort(key=sortNilaiTraining)  
        result.append(resultPerTraining)  
    return(result)
```

2.4.1.2. Manhattan

```
def manhattan(xTrain,yTrain,xVal):  
    result = []  
    for x in xVal:  
        resultPerTraining = []; indexTraining = 0  
        for j in xTrain:  
            d = abs(x[0]-j[0]) + abs(x[1]-j[1]) + abs(x[2]-j[2])  
            #print(x)  
            hasilPerTrainingData = [d,yTrain[indexTraining]]  
            resultPerTraining.append(hasilPerTrainingData)  
            indexTraining = indexTraining + 1  
        resultPerTraining.sort(key=sortNilaiTraining)  
        result.append(resultPerTraining)  
        #print(result)  
    return(result)
```

2.4.1.3. Minkowski

```
def minkowski(xTrain,yTrain,xVal,h=9):  
    result = []  
    for x in xVal:  
        resultPerTraining = []; indexTraining = 0  
        for j in xTrain:  
            d = (abs(x[0]-j[0]))**h + (abs(x[1]-j[1]))**h + (abs(x[2]-j[2]))**h  
            #print(x)  
            hasilPerTrainingData = [d**(1/h),yTrain[indexTraining]]  
            resultPerTraining.append(hasilPerTrainingData)  
            indexTraining = indexTraining + 1  
        resultPerTraining.sort(key=sortNilaiTraining)  
        result.append(resultPerTraining)  
        #print(result)  
    return(result)
```


2.4.1.4. Pemilihan k tetangga terdekat

```
def pilihTetangga(result,k):
    tetangga = []
    #print(result)
    for data in result:
        selected = data[:k]
        tetangga.append(selected)
    #print(tetangga)
    return tetangga
```

2.4.1.5. Memilih data prediksi pada tetangga sebanyak k Nilai unik y untuk setiap scaling data:

```
np.unique(dataRobusted["y"])
```

✓ 0.3s

Python

```
array([-1., 0.])
```

```
np.unique(dataNormalized["y"])
```

✓ 0.4s

Python

```
array([0., 1.])
```

```
np.unique(dataStandarized["y"])
```

✓ 0.4s

Python

```
array([-1.66896065, 0.59715106])
```

Ubah kondisi sesuai data unique pada metode skaling if dataVal[1] ==
<"ubah di sini">:

```
def vote(result):
    hasil = []
    #print(result)
    for data in result:
        count0 = 0; count1 = 0;
        for dataVal in data:
            if dataVal[1] == 0:
                count0 = count0 + 1
            elif dataVal[1] == 1:
                count1 = count1 + 1
            if count0 > count1:
                hasil.append(0)
            else:
                hasil.append(1)
    #print(hasil)
    return hasil
```

✓ 0.4s

Python

2.4.1.6. Menghitung nilai akurasi

```
def akurasi(hasil,yVal):
    akurasi = 0
    for i in range(len(hasil)):
        if hasil[i] == yVal[i]:
            akurasi = akurasi + 1
    return akurasi
```

✓ 0.4s

Python

2.4.1.7. Membuat plot nilai akurasi untuk memilih k terbaik

```
def makePlot(akurasiList,k):
    listAngka = []
    for i in range(1,k,2):
        listAngka.append(i)
    plt.plot(listAngka,akurasiList)
    plt.title("Hasil Akurasi")
    plt.xlabel("Nilai k")
    plt.ylabel("Akurasi")
    plt.show()
```

2.4.1.8. Main program KNN untuk training data dan validasi dengan cara cross k-fold validation

```
def knn(xTrain,yTrain,xVal,yVal,k):
    resultEuclidean = euclidean(xTrain,yTrain,xVal)
    resultManhattan = manhattan(xTrain,yTrain,xVal)
    resultMinkowski = minkowski(xTrain,yTrain,xVal)

    resultEuclidean = pilihTetangga(resultEuclidean,k)
    resultManhattan = pilihTetangga(resultManhattan,k)
    resultMinkowski = pilihTetangga(resultMinkowski,k)

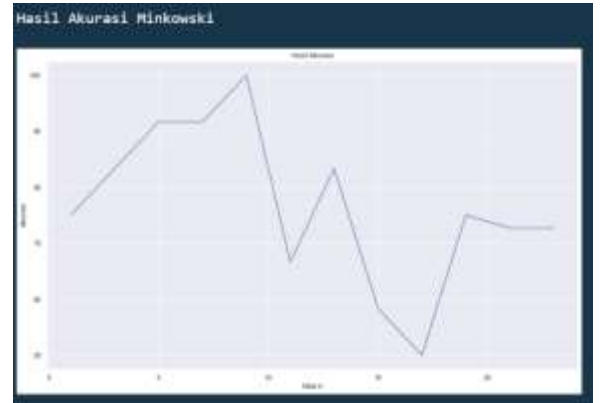
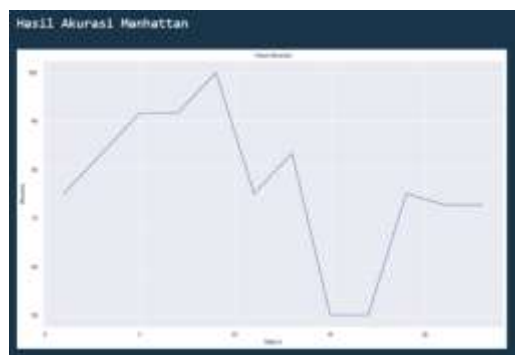
    finalEuclidean = vote(resultEuclidean)
    finalManhattan = vote(resultManhattan)
    finalMinkowski = vote(resultMinkowski)

    hasilAkurasiEuclidean = (akurasi(finalEuclidean,yVal)/len(yVal))*100
    hasilAkurasiManhattan = (akurasi(finalManhattan,yVal)/len(yVal))*100
    hasilAkurasiMinkowski = (akurasi(finalMinkowski,yVal)/len(yVal))*100
    #print(hasilAkurasiEuclidean,hasilAkurasiManhattan,
    #      hasilAkurasiMinkowski)
    #print(str(hasilAkurasiEuclidean)+"%")
    return hasilAkurasiEuclidean,hasilAkurasiManhattan,
    hasilAkurasiMinkowski
```

2.4.1.9. Implementasi k-fold cross validation

Untuk menghindari saat jumlah data count0 dan count1 sama, maka tetangga yang diterima hanya tetangga yang ganjil.

```
def kFolds(arrayOf,k):  
    # membagi data menjadi k folds  
    folds = np.array_split(arrayOf,k)  
    listEuclidean, listManhattan, listMinkowski = [],[],[]  
  
    for i in range(1,k,2):  
        #print("Folds: ",i)  
        trainingSet = folds.copy()  
        validationSet = folds[i]  
        del trainingSet[i]  
        trainingSet = pd.concat(trainingSet, sort=False)  
  
        xTrain,yTrain = splitDependent(trainingSet)  
        xVal,yVal = splitDependent(validationSet)  
        #print("training set: \n",trainingSet)  
        #print("validation set: \n", validationSet)  
        hEuclidean, hManhattan, hMinkowski = knn(xTrain,  
        yTrain,xVal,yVal,k)  
        listEuclidean.append(hEuclidean)  
        listManhattan.append(hManhattan)  
        listMinkowski.append(hMinkowski)
```



Dapat dilihat kalau data dengan k=9 adalah tetangga terbaik menurut k-fold cross validation.

3. Hasil Output

Untuk data testing. Dilakukan hal-hal berikut:

3.1. Import data testing

```
dfTest = pd.read_excel("https://github.com/khalilullahalfaath/  
AI_Tubes-03-Learning/blob/  
19d11d6e2fa2d540afdc0cec3308b82c664d29f7/traintest.xlsx?  
raw=true", "test")
```

dfTest

✓ 0.4s

	id	x1	x2	x3	y
0	297	43	59	2	?
1	298	67	66	0	?
2	299	58	60	3	?
3	300	49	63	3	?
4	301	45	60	0	?
5	302	54	58	1	?
6	303	56	66	3	?
7	304	42	69	1	?
8	305	50	59	2	?
9	306	59	60	0	?

3.2. Drop kolom yang tidak dibutuhkan

```
dfTest.drop(columns = ['id'], axis = 1, inplace = True)
```

dfTest

✓ 0.6s Python

	x1	x2	x3	y
0	43	59	2	?
1	67	66	0	?
2	58	60	3	?
3	49	63	3	?
4	45	60	0	?
5	54	58	1	?
6	56	66	3	?

3.3. Scaling dataset

```
#normalizestest
def minMaxScalingTest(df) :
    return (df.iloc[:, :3] - df.iloc[:, :3].min()) / (df.iloc[:, :3].max() - df.iloc[:, :3].min())

def standardScalingTest(df) :
    return (df.iloc[:, :3] - df.iloc[:, :3].mean()) / df.iloc[:, :3].std()

def robustScalingTest(df):
    return ((df.iloc[:, :3] - df.iloc[:, :3].median()) / (df.iloc[:, :3].quantile(0.75) - df.iloc[:, :3].quantile(0.25)))

dataTestNormalized = minMaxScalingTest(dfTest)
dataTestRobusted = robustScalingTest(dfTest)
dataTestStandarized = standardScalingTest(dfTest)

dataTestNormalized
```

dataTestNormalized

✓ 0.6s

	x1	x2	x3
0	0.04	0.090909	0.666667
1	1.00	0.727273	0.000000
2	0.64	0.181818	1.000000
3	0.28	0.454545	1.000000
4	0.12	0.181818	0.000000
5	0.48	0.000000	0.333333
6	0.56	0.727273	1.000000
7	0.00	1.000000	0.333333
8	0.32	0.090909	0.666667
9	0.68	0.181818	0.000000

dataTestRobusted

✓ 0.5s

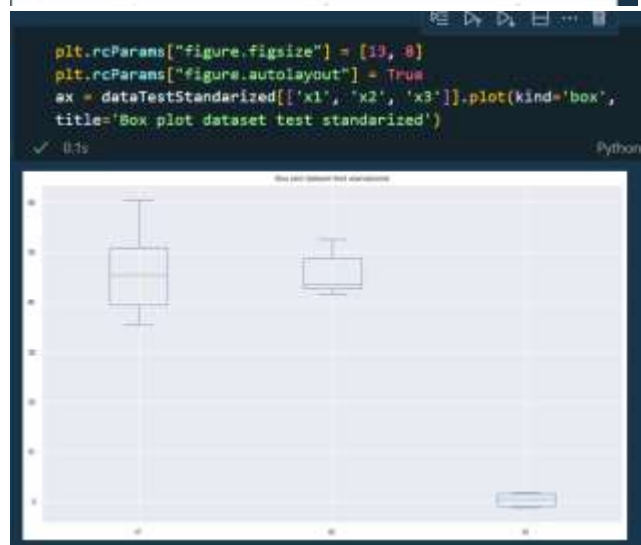
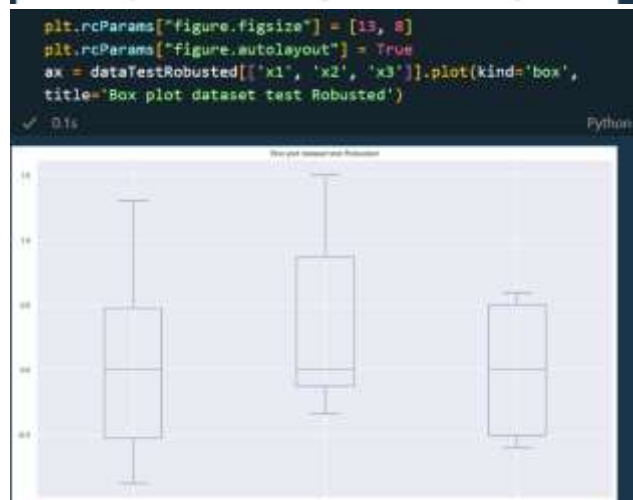
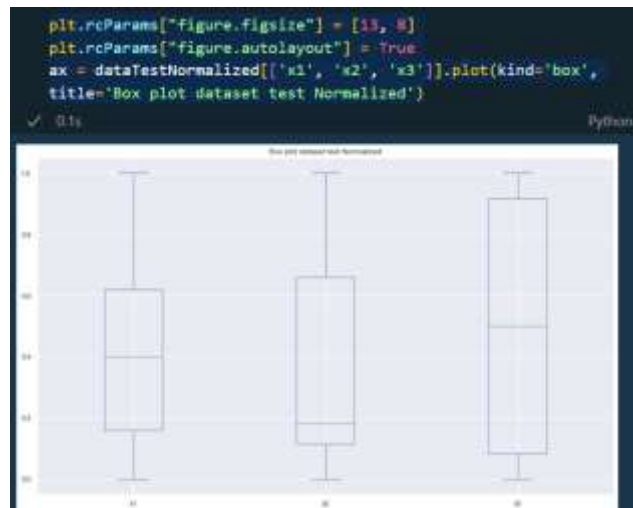
	x1	x2	x3
0	-0.782609	-0.166667	0.2
1	1.304348	1.000000	-0.6
2	0.521739	0.000000	0.6
3	-0.260870	0.500000	0.6
4	-0.608696	0.000000	-0.6
5	0.173913	-0.333333	-0.2
6	0.347826	1.000000	0.6
7	-0.869565	1.500000	-0.2
8	-0.173913	-0.166667	0.2
9	0.608696	0.000000	-0.6

dataTestStandarized

✓ 0.6s

	x1	x2	x3
0	36.440255	42.559767	0.818242
1	60.440255	49.559767	-1.181758
2	51.440255	43.559767	1.818242
3	42.440255	46.559767	1.818242
4	38.440255	43.559767	-1.181758
5	47.440255	41.559767	-0.181758
6	49.440255	49.559767	1.818242
7	35.440255	52.559767	-0.181758
8	43.440255	42.559767	0.818242
9	52.440255	43.559767	-1.181758

3.4. Output boxplot



3.5. KNN pada data training untuk semua data terhadap data testing untuk mengisi kolom y target test yang masih kosong

```
# resultEuclidean, resultManhattan, resultMinkowski = [],[],[]
# resultEuclideanFix, resultManhattanFix, resultMinkowskiFix
# finalEuclidean, finalManhattan, finalMinkowski = [],[],[]

resultEuclidean = euclidean(xTrain,yTrain,xTest)
#print(resultEuclidean)
resultManhattan = manhattan(xTrain,yTrain,xTest)
resultMinkowski = minkowski(xTrain,yTrain,xTest)

resultEuclideanFix = pilihTetangga(resultEuclidean,9)
resultManhattanFix = pilihTetangga(resultManhattan,9)
resultMinkowskiFix = pilihTetangga(resultMinkowski,9)

finalEuclidean = vote(resultEuclideanFix)
finalManhattan = vote(resultManhattanFix)
finalMinkowski = vote(resultMinkowskiFix)

print(finalEuclidean)
print(finalManhattan)
print(finalMinkowski)
```

0.9s Python

```
[1, 1, 0, 0, 1, 1, 0, 1, 1, 1]
[1, 1, 0, 0, 1, 1, 0, 1, 1, 1]
[1, 1, 0, 0, 1, 1, 0, 1, 1, 1]
```

3.6. Mengisi dataset test dengan kolom prediksi

```
xResult = pd.DataFrame(dfTest, columns=['x1', 'x2', 'x3'])
yResult = pd.DataFrame(finalManhattan, columns=['y'])
result = pd.merge(xResult, yResult, left_index=True,
right_index=True)
result
```

0.6s Python

	x1	x2	x3	y
0	43	59	2	1
1	67	66	0	1
2	58	60	3	0
3	49	63	3	0
4	45	60	0	1
5	54	58	1	1
6	56	66	3	0
7	42	69	1	1
8	50	59	2	1
9	59	60	0	1

3.7. Ekspor dataset

ekspor dataset

```
dataResult = pd.ExcelWriter('dataResult.xlsx')
result.to_excel(dataResult)
dataResult.save()
```

0.1s Python

4. Eksplorasi

Di sini kami mengetes akurasi algoritma KNN kami dengan dataset yang sama yang training data dan validation datanya dipilih sama random dengan perbandingan 20% dan 80%.

```
# Split dataset to train and test data
test_size = 0.2
trainingSet = dataNormalized.sample(frac = 1-test_size,
random_state = 74)
validationSet = dataNormalized.drop(trainingSet.index)

trainingSet.reset_index(drop=True, inplace=True)
validationSet.reset_index(drop=True, inplace=True)

print(f"No. of training examples: {trainingSet.shape[0]}")
print(f"No. of testing examples: {validationSet.shape[0]}")
```

✓ 0.6s Python

No. of training examples: 237
No. of testing examples: 59

4.1. Pengecekan head training set terpilih dan validation set terpilih

trainingSet.head(3)				
✓	0.4s			
	x1	x2	x3	y
0	0.094340	0.454545	0.0	1.0
1	0.320755	0.363636	0.0	0.0
2	0.716981	0.909091	0.0	1.0

validationSet.head(3)				
✓	0.6s			
	x1	x2	x3	y
0	0.566038	0.545455	0.00	1.0
1	0.075472	0.181818	0.00	1.0
2	0.754717	0.090909	0.64	1.0

4.2. Algoritma KNN pada data random

```
xTrain,yTrain = splitDependent(trainingSet)
xVal,yVal = splitDependent(validationSet)

resultEuclidean, resultManhattan,
resultMinkowski = [],[],[]
resultEuclideanFix, resultManhattanFix,
resultMinkowskiFix = [],[],[]
finalEuclidean, finalManhattan, finalMinkowski
= [],[],[]
listEuclidean, listManhattan,listMinkowski = [],
[],[]

xTrain = np.array(xTrain)
#print(xTrain)
yTrain = np.array(yTrain)
xVal = np.array(xVal)
yVal = np.array(yVal)

for i in range(1,50,2):
    resultEuclidean = euclidean(xTrain,yTrain,
xVal)
    resultManhattan = manhattan(xTrain,yTrain,
xVal)
    resultMinkowski = minkowski(xTrain,yTrain,
xVal)

    resultEuclideanFix = pilihTetangga
(resultEuclidean,i)
    resultManhattanFix = pilihTetangga
(resultManhattan,i)
    resultMinkowskiFix = pilihTetangga
(resultMinkowski,i)

    finalEuclidean = vote(resultEuclideanFix)
    finalManhattan = vote(resultManhattanFix)
    finalMinkowski = vote(resultMinkowskiFix)

    hasilAkurasiEuclidean = (akurasi
(finalEuclidean,yVal)/len(yVal))*100
    hasilAkurasiManhattan = (akurasi
(finalManhattan,yVal)/len(yVal))*100
    hasilAkurasiMinkowski = (akurasi
(finalMinkowski,yVal)/len(yVal))*100
```

4.3. Hasil Akurasi pada data random

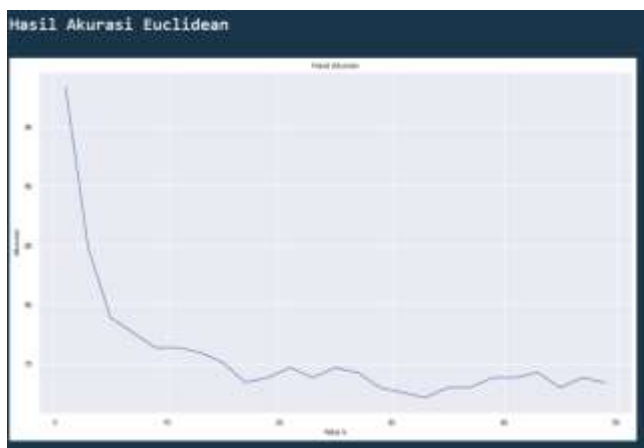


5. Kesimpulan

Algoritma KNN adalah algoritma yang sangat mudah untuk diimplementasikan. Namun, akurasi sangat bergantung dengan bagusnya dataset. Pada data di atas, data cenderung menjorok ke target bernilai 1 sehingga keumuman target pada testing adalah bernilai 1.

Algoritma kami di atas menyediakan beberapa metode dalam pencarian jarak, skaling, dan jumlah tetangga. Kami memilih pencarian jarak dengan euclidean, manhattan, dan minkowski, scaling dengan minkowski, dan tetangga berjumlah 9.

Akurasi akhir yang kami dapatkan tertinggi pada kfold cross validation adalah 100% untuk k=9. Namun, berbeda pada pemilihan data random = 74, akurasi maksimal adalah kurang dari 80 pada k = 50. Namun, pada data random = 36, akurasi maksimal sampai dengan 98 untuk k=1 kemudian turun secara ekstrem untuk k selanjutnya.



Berikut adalah hasil akhir untuk data testing

	x1	x2	x3	y
0	43	59	2	1
1	67	66	0	1
2	58	60	3	0
3	49	63	3	0
4	45	60	0	1
5	54	58	1	1
6	56	66	3	0
7	42	69	1	1
8	50	59	2	1
9	59	60	0	1

6. Resources

Link Github
https://github.com/khalilullahalfaath/AI_Tubes-03-Learning
Link google colabs
https://colab.research.google.com/drive/1ipKZRgajuXTQIVbwp1FCGqx7y2PAzh9d?usp=sharing
Link video presentasi
https://youtu.be/6XX67xnB3-c

7. Daftar Pustaka

- A. (2020, December 29). *Pengertian dan Cara Kerja Algoritma K-Nearest Neighbors (KNN)*. Advernesia. Retrieved June 17, 2022, from <https://www.advernesia.com/blog/data-science/pengertian-dan-cara-kerja-algoritma-k-nearest-neighbours-knn/>
- Arifianto, A. (2020a). *Introduction to Learning* [Slides]. LMS Telkom University. <https://lms.telkomuniversity.ac.id/mod/resource/view.php?id=1788959>
- Arifianto, A. (2020b). *Nearest Neighbor* [Slides]. LMS Telkom University. <https://lms.telkomuniversity.ac.id/mod/resource/view.php?id=1788959>
- Arifianto, A. [Anditya Arifianto]. (2020, November 24). *CII2M3_ADF06 / 10 - Nearest Neighbor* [Video]. YouTube. <https://www.youtube.com/watch?v=tb2gIc7Ug6c&list=PLcYqQ2VpYNYsJD4oBKf1NHZQNjm6jItBq&index=17&t=3969s>
- Osiński, B., & Budek, K. (2022, February 15). *Cookie and Privacy Settings*. Deepsense.Ai. Retrieved June 17, 2022, from <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/>

Rodriguez, J. (2018, June 13). *Types of Artificial Intelligence Learning Models* - Jesus Rodriguez. Medium. Retrieved June 17, 2022, from <https://jrodthoughts.medium.com/types-of-artificial-intelligence-learning-models-814e46eca30e#:~:text=Learning%20is%20one%20of%20the,making%20observations%20about%20its%20environment>