JOBSHEET 2
Regression

1. Tujuan

- Mahasiswa memahami tentang konsep KNN
- Mahasiswa memahami konsep KNN untuk permasalahan klasifikasi dan regresi
- Mahasiswa mampu melakukan perhitungan manual KNN
- Mahasiswa mampu menerapkan KNN untuk permasalahan klasifikasi dan regresi

2. Materi

Pada bab ini akan dibahas mengenai algoritma *k-Nearest Neighbors (KNN)* untuk permasalahan klasifikasi dan regresi, contoh implementasi *k-Nearest Neighbors (KNN)* dengan menggunakan bahasa pemrograman *Python*.

2.1. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) merupakan salah satu algoritma *machine learning* yang dapat digunakan untuk memecahkan permasalahan klasifikasi dan regresi. KNN melakukan klasifikasi dan regresi berdasarkan jarak antar datanya. Perhitungan jarak yang bisa digunakan adalah *Euclidian distance, Manhattan distance, Minkowski distance,* dll. Pada perhitungan klasifikasi KNN dapat dilakukan pada data untuk ***binary*** ataupun ***multi-class classification*****.**

Langkah pertama pada *K-nearest neighbors* adalah menentukan nilai k, k merupakan jumlah tetangga terdekat yang akan menjadi dasar un/'tuk melakukan prediksi kelas/ nilai dari data training. Nilai k diberikan secara manual biasanya nilai k diberikan angka ganjil untuk menghindari angka yang imbang. K tetangga yang dipilih adalah tetangga terdekat dari data testing terhadap data training, yang diukur dengan perhitungan jarak.

Rumus perhitungan jarak dengan Euclidean distance:

$$d(xi, yi) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2}$$

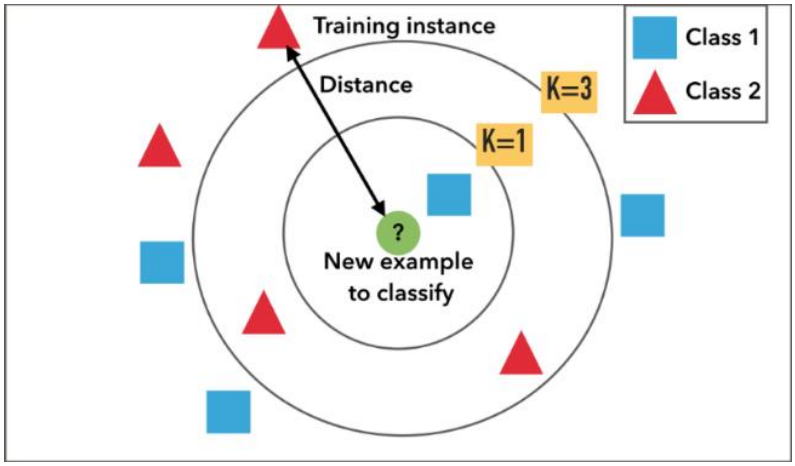
Dengan d adalah jarak antara titik pada data training (x) dan titik data testing (y) yang akan diklasifikasi, dimana x = x1, x2, x3, .. xi dan y = y1, y2, y3, y .. yi dan i mempresentasikan nilai atribut/fitur. Jika terdapat 7 buah fitur maka i mulai dari 1 sampai 7.

2.2. Klasifikasi menggunakan k-Nearest Neighbors (KNN)

Berikut ini merupakan langkah-langkah algoritma KNN untuk menyelesaikan permasalahan klasifikasi:

1. Input data training dan data testing
2. Tentukan nilai k
3. Hitung jarak data testing ke setiap data training dengan menggunakan rumus perhitungan jarak
4. Mengurutkan hasil perhitungan jarak data training dengan data testing (jika menggunakan perhitungan jarak eucliden maka jika nilai jarak mendekati nilai 0, maka dianggap kedua data tersebut mirip)
5. Ambil sejumlah k data (jumlah tetangga terdekat nilai Euclidian yang telah diurutkan dari nilai terkecil ke nilai terbesar)
6. Tentukan label kelas data training berdasarkan kelas yang datanya paling dominan dari k data latih yang diambil.

Perhatikan gambar ilustrasi berikut ini:



Gambar 1. Ilustrasi K-NN

Pada gambar diatas merupakan ilustrasi dari algoritma K-NN. Data bulat warna hijau merupakan data testing yang akan dicari kelasnya, sedangkan data segitiga warna merah dan persegi warna biru adalah data training. Data segitiga menunjukkan titik-titik data pada class 2 dan data persegi menunjukkan titik-titik data pada class 1. Jika ditentukan k=1 maka tetangga yang terdekat dengan data testing adalah data persegi, maka data testing akan memiliki kelas = class1, sedangkan jika ditentukan k=3 maka akan terdapat 3 tetangga

terdekat, yaitu 2 buah data persegi dan 1 buah data segitiga. Karena pada k=3 data persegi lebih dominan maka kelas untuk data testing tersebut adalah kelas = class1.

Contoh perhitungan K-NN untuk klasifikasi data.

Tabel 1. Data training

Height	Weight	Label
158 cm	64 kg	male
170 cm	66 kg	male
183 cm	84 kg	male
191 cm	80 kg	male
155 cm	49 kg	female
163 cm	59 kg	female
180 cm	67 kg	female
158 cm	54 kg	female
178 cm	77 kg	female

Pada tabel 1 merupakan contoh data training dimana terdapat 2 fitur yaitu *height* (tinggi) dan *weight* (berat). Kedua fitur tersebut akan digunakan untuk menentukan fitur jenis kelamin. Diasumsikan bahwa akan dilakukan prediksi jenis kelamin seseorang dengan tinggi 155 cm dan berat 70 kg. Langkah-langkah yang dilakukan sebagai berikut:

1. Tentukan nilai k
2. Hitung jarak data testing terhadap setiap data training. Keseluruhan jarak dapat dilihat pada tabel 2. Rumus perhitungan jarak yang digunakan adalah Euclidean distance.
3. Urutkan data mulai dari nilai terendah sampai nilai tertinggi.
4. Ambil 3 data sebagai tetangga terdekat sehingga didapatkan data 1 dengan jarak 6.71, data 6 dengan jarak 13.6, data 8 dengan jarak 16.28
5. Dari ketiga tetangga tersebut didapatkan kelas *male*, *female*, *female*. Karena *female* lebih dominan sehingga disimpulkan bahwa data testing memiliki kelas *female.*

Tabel 2. Perhitungan jarak data testing terhadap data training

Height	Weight	Label	Distance from test instance
158 cm	64 kg	male	$\sqrt{(158-155)^2+(64-70)^2}=6.71$
170 cm	66 kg	male	$\sqrt{(170-155)^2+(64-70)^2}=21.93$
183 cm	84 kg	male	$\sqrt{(183-155)^2+(84-70)^2}=31.30$
191 cm	80 kg	male	$\sqrt{(191-155)^2+(80-70)^2}=37.36$
155 cm	49 kg	female	$\sqrt{(155-155)^2+(49-70)^2}=21.00$
163 cm	59 kg	female	$\sqrt{(163-155)^2+(59-70)^2}=13.60$
180 cm	67 kg	female	$\sqrt{(180-155)^2+(67-70)^2}=25.18$
158 cm	54 kg	female	$\sqrt{(158-155)^2+(54-70)^2}=16.28$
178 cm	77 kg	female	$\sqrt{(178-155)^2+(77-70)^2}=24.04$

Contoh Kode Program KNN tanpa *library Scikit-learn*:

1. Perhitungan Euclidean distance

```
import numpy as np
import matplotlib.pyplot as plt

X_train = np.array([
                    [158,64],
                    [170,86],
                    [183,84],
                    [191,80],
                    [155,49],
                    [163,59],
                    [180,67],
                    [158,54],
                    [170,67]
                    ])
y_train=['male','male','male','male','female','female','female','female','female']

X_test = np.array([[155,70]])
```

Pada gambar diatas x_train merupakan variabel data training yang dibuat dalam bentuk array, dan y_train adalah label dari masing-masing data training, x_test merupakan data yang akan diprediksi kelasnya.

```
# calculate euclidean distance
dis = np.sqrt(np.sum((X_train-X_test)**2, axis=1))
dis

array([ 6.70820393, 21.9317122 , 31.30495168, 37.36308338, 21.        ,
       13.60147051, 25.17935662, 16.2788206 , 15.29705854])
```

Pada gambar diatas adalah perhitungan *Euclidean distance* untuk data training ke setiap data testing. Untuk memastikan bahwa kode program yang dibuat benar maka lakukan pencocokan antara kode program dengan hasil jarak yang dihasilkan oleh program.

2. Penentuan nilai k = 3, pengurutan nilai jarak, pencarian tetangga terdekat, serta pencarian kelas dominan.

```
21 dis = np.sqrt(np.sum((X_train-X_test)**2, axis=1))
22 print(dis)
23
24 nn_sort=dis.argsort()[:3]
25 print(nn_sort)
26 nn_gender = np.take(y_train,nn_sort)
27 print(nn_gender)
28
29 from collections import Counter
30 b= Counter(np.take(y_train,dis.argsort()[:3]))
31 b.most_common(1)[0][0]
```

```
[ 6.70820393 21.9317122  31.30495168 37.36308338 21.         13.60147051
 25.17935662 16.2788206  23.19482701]
[0 5 7]
['male' 'female' 'female']
'female'
```

Pengurutan nilai jarak dilakukan dengan menggunakan fungsi *argsort* dan selanjutnya diambil 3 nilai jarak terendah. *Np.take* digunakan untuk mengembalikan elemen dari array di sumbu y dan indeks dari *nn_sort* sehingga didapatkan label kelas dari 3 tetangga terdekat. Selanjutnya dari ketiga label tersebut akan dicari kelas dominan menggunakan fungsi *counter* dan *most_common*.

Contoh Kode Program KNN menggunakan *library Scikit-learn*:

```
1 from sklearn.preprocessing import LabelBinarizer
2 from sklearn.neighbors import KNeighborsClassifier
3
4 lb=LabelBinarizer()
5 y_trainbin = lb.fit_transform(y_train)
6 print(y_trainbin)
7
8 k=3
9 clf =KNeighborsClassifier(n_neighbors=k)
10 clf.fit(X_train,y_trainbin.reshape(-1))
11 pred_bin = clf.predict(np.array([X_test]).reshape(1,-1))[0]
12 pred_label = lb.inverse_transform(pred_bin)
13 print(pred_label)
```

```
[[1]
 [1]
 [1]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]]
['female']
```

LabelBinarizer merupakan salah satu class yang ada pada *Scikit-learn* yang dapat digunakan untuk merubah label awal yang berupa data kategorik (*string*) menjadi bilangan

biner 0 dan 1. Nilai biner 0 dan 1 keluaran dari *LabelBinarizer* merupakan *numpy array*. Selanjutnya *class KNeighborsClassifier* digunakan untuk melakukan prediksi terhadap data testing. Tentukan nilai k sebagai parameter untuk *class KNeighborsClassifier*. Selanjutnya, fit model pada data training menggunakan method *fit()* dan kemudian prediksi pada data training menggunakan method *predict().*

Evaluasi Model

Terdapat data testing seperti pada tabel 3. Data tersebut yang akan digunakan untuk melakukan evaluasi terhadap model yang telah dibuat.

Tabel 3. Data testing

Height	Weight	Label
168 cm	65 kg	male
170 cm	61 kg	male
160 cm	52 kg	female
169 cm	67 kg	female

Pada kode program yang telah dipaparkan sebelumnya tambahkan *variable x_test* dan *y_test* sebagai data testing. *X_test* menyimpan nilai tinggi dan berat sedangkan *y_test* menyimpan label untuk data testing. Label ini yang akan digunakan sebagai *actual data* untuk nantinya menghitung performa dari klasifikasi berupa *accuracy*, *precision*, dan *recall*.

```
from sklearn.preprocessing import LabelBinarizer
from sklearn.neighbors import KNeighborsClassifier
X_test = np.array([[168,65],
                   [190,96],
                   [160,52],
                   [169,67]
                   ])
y_test =['male','male','female','female']

#lb=LabelBinarizer()
y_testbin = lb.transform(y_test)
print('label biner:%s' %y_testbin.T[0])

k=3
clf =KNeighborsClassifier(n_neighbors=k)
clf.fit(X_train,y_trainbin.reshape(-1))
pred_bin = clf.predict(X_test)
pred_label = lb.inverse_transform(pred_bin)
print(pred_label)

```

```
label biner:[1 1 0 0]
['female' 'male' 'female' 'female']
```

Berdasarkan output yang ada pada gambar diatas dapat dilihat bahwa terdapat label yang tidak sesuai dengan *actual data*. Yaitu label untuk data pertama dimana label actualnya adalah *male* sedangkan hasil prediksi oleh model adalah *female*. Pada pembahasan sebelumnya telah dijelaskan bahwa error pada klasifikasi yaitu *false positives* dan *false negatives*. Ada banyak ukuran performa untuk klasifikasi diantaranya adalah *accuracy*, *precision*, dan *recall*.

		actual / manual	
		positif	negatif
predicted	positif	TP	FP
	negatif	FN	TN

Male: positif; Female: negatif

Prediksi Sistem	Actual	TP/TN/FN/FP
Female	Male	FN
Male	Male	TP
Female	Female	TN
Male	Female	FP

Akurasi adalah proporsi data testing yang diklasifikasikan dengan benar dibagi keseluruhan jumlah data testing. Model yang telah dibuat salah mengklasifikasikan salah satu dari 4 data training sehingga akurasi yang didapatkan adalah 75%:

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP+FP+FN+TN})} = \frac{3}{4} = 0.75 * 100\% = 75\%$$

Kode program untuk menghitung akurasi:

```
1 from sklearn.metrics import accuracy_score
2 print('akurasi: %s' % accuracy_score(y_testbin,pred_bin))
```

```
akurasi: 0.75
```

Presisi merupakan proporsi data testing yang diprediksi positif dan benar-benar positif sesuai data actual. Dalam contoh ini, kelas positifnya ditentukan pada kelas *male*. Tidak ada aturan tertentu manakah label kelas yang menjadi kelas positif. Penetapan ini bisa dibolak-balik, bisa saja male menjadi kelas positif atau female yang menjadi kelas positif.

$$\text{Precision} = \frac{\text{TP}}{(\text{TP+FP})} = \frac{1}{1} = 1 = 100\%$$

Kode program untuk menghitung presisi:

```
1 from sklearn.metrics import precision_score
2 print('presisi: %s' % precision_score(y_testbin,pred_bin))

presisi: 1.0
```

Recall adalah proporsi data testing yang benar-benar positif (actual positif) dan diperkirakan positif. Model yang dibuat memprediksi bahwa salah satu dari 2 contoh pengujian adalah positif. Sehingga didapatkan nilai recall 50%:

$$\text{Recall} = \frac{\text{TP}}{(\text{TP+FN})} = \frac{1}{(2)} = 0.5 = 50\%$$

Kode program untuk menghitung recall:

```
1 from sklearn.metrics import recall_score
2 print('recall: %s' % recall_score(y_testbin,pred_bin))

recall: 0.5
```

2.3. Regresi dengan menggunakan KNN

Regresi merupakan salah satu model pembelajaran *supervised learning*. Berbeda dengan klasifikasi yang melakukan prediksi kelas / label data pada regresi, ciri utamanya adalah prediksi yang dihasilkan berupa angka dan bersifat *continuous*. Tabel 4 merupakan karakteristik dari regresi dan klasifikasi.

Tabel 4. Karakteristik Regresi dan Klasifikasi

Karakteristik	Regresi	Klasifikasi
Output Prediksi	Angka	Label / Kelas
Tipe Output	Continuous	Diskrit
Metrik Evaluasi	MAPE, MSE, MAE, dll	Akurasi, Presisi, Recall, dll

Berikut ini merupakan langkah algoritma KNN untuk menyelesaikan permasalahan regresi:

1. Input data training dan data testing
2. Tentukan nilai k
3. Hitung jarak data testing ke setiap data training dengan menggunakan rumus perhitungan jarak
4. Mengurutkan hasil perhitungan jarak data training dengan data testing (jika menggunakan perhitungan jarak *eucliden* maka jika nilai jarak mendekati nilai 0 maka dianggap kedua data tersebut mirip)

5. Ambil sejumlah k data (jumlah tetangga terdekat nilai *Euclidian* yang telah diurutkan dari kecil ke besar)
6. Tentukan nilai data training berdasarkan rata-rata tetangga terdekatnya.

Contoh data untuk permasalahan regresi:

Tabel 5. Data Training

Height	Sex	Weight
158 cm	male	64 kg
170 cm	male	66 kg
183 cm	male	84 kg
191 cm	male	80 kg
155 cm	female	49 kg
163 cm	female	59 kg
180 cm	female	67 kg
158 cm	female	54 kg
178 cm	female	77 kg

Tabel 6. Data Testing

Height	Sex	Weight
168 cm	male	65 kg
170 cm	male	61 kg
160 cm	female	52 kg
169 cm	female	67 kg

2.4. Evaluasi Performa Regresi

Mean Absolute Error (MAE) adalah rata-rata nilai absolut dari kesalahan prediksi.

$$MAE = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i|$$

Dimana y adalah nilai actual sedangkan $\hat{y}_i$ adalah nilai prediksi dan n adalah banyaknya data

Mean Squared Error (MSE) atau Mean Squared Deviation (MSD), adalah rata-rata dari selisih kesalahan prediksi yang dikuadratkan.

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2$$

Dimana y adalah nilai actual sedangkan $\hat{y}_i$ adalah nilai prediksi dan n adalah banyaknya data.

Contoh implementasi Regresi menggunakan algoritma KNN:

```
import numpy as np
import matplotlib.pyplot as plt

X_train = np.array([
                    [158,1],
                    [170,1],
                    [183,1],
                    [191,1],
                    [155,0],
                    [163,0],
                    [180,0],
                    [158,0],
                    [178,0]
                    ])

y_train =[64,66,84,80,49,59,67,54,77]
X_test=np.array([[168,1],
                 [170,1],
                 [160,0],
                 [169,0]
])
y_test=[65,61,52,67]
```

```
from sklearn.preprocessing import LabelBinarizer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_erro
k=3
clf = KNeighborsRegressor(n_neighbors=k)
clf.fit(X_train,y_train)
predictions=clf.predict(X_test)
print('prediksi : %s' %predictions)
print('MAE: %s' % mean_absolute_error(y_test,predictions))
print('MSE : %s ' % mean_squared_error(y_test,predictions))
```

Output:

```
prediksi : [63.         67.33333333 59.         67.33333333]
MAE: 3.9166666666666643
MSE : 23.305555555555543
```

Berdasarkan kode program diatas dapat diketahui bahwa variabel jenis kelamin harus diubah menjadi angka terlebih dahulu agar bisa diproses ke dalam model machine learning. Pada contoh diatas, *male* diubah menjadi 1, lalu *female* diubah menjadi 0. Untuk melakukan regresi menggunakan KNN digunakan *class KNeighborsRegressor*. Selanjutnya fit model pada data training menggunakan *method fit()* dan kemudian prediksi pada data training menggunakan *method predict().* Berdasarkan hasil prediksi kemudian dilakukan perhitungan MAE menggunakan *class mean_absolute_error* dan MSE dengan menggunakan *class mean_squared_error*. Berikut merupakan perhitungan manual MAE dan MSE. Tabel 7 merupakan perbandingan data actual dengan hasil prediksi.

Tabel 7. Selisih prediksi dengan data actual

Data Actual	Hasil Prediksi	Selisih
65	63	2
61	67.333	6.333
52	59	7
67	67.333	0.333

$$MAE = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i| = \frac{1}{4}(2 + 6.333 + 7 + 0.333) = 3.9$$

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 = \frac{1}{4}\,(2^2 + 6{,}333^2 + 7^2 + 0{,}333^2) = 23{,}3$$

3. Tugas Praktikum

Gunakan metode KNN untuk menyelesaikan permasalahan klasifikasi berikut:

a) Training Dataset

Temperature	UV Index	Decision
26	8	Go out
26	11	Go out
29	8	Go out
34	10	Stay at home
36	11	Stay at home
33	8	Go out
30	9	Go out
27	10	Go out
27	11	Go out
26	11	Go out
31	13	Stay at home
34	8	Go out

b) Testing Dataset

Temperature	UV Index	Decision
30	10	Stay at home
28	13	Go out
29	9	Go out
31	12	Stay at home
27	8	Go out