LAPORAN TUGAS PEMROGRAMAN LEARNING
PENGANTAR KECERDASAN BUATAN

Telkom University

Disusun Oleh :
Muhammad Nur Iqbal Wariesky (1301204044)
Muh Thoriq Akhdan (1301204031)

Program Studi S1 Informatika – Fakultas Informatika Universitas Telkom
Jalan Telekomunikasi Terusan Buah Batu, Bandung Indonesia

Bab I

Analisis Masalah

Diberikan file traintest.xlsx yang terdiri dari dua sheet: train dan test, yang berisi dataset untuk problem klasifikasi biner (binary classification). Setiap record atau baris data dalam dataset tersebut secara umum terdiri dari nomor baris data (id), fitur input (x1 sampai x3), dan output kelas (y). Fitur input terdiri dari nilai-nilai integer dalam range tertentu untuk setiap fitur. Sedangkan output kelas bernilai biner (0 atau 1)

Sheet train berisi 296 baris data, lengkap dengan target output kelas (y). Gunakan sheet ini untuk tahap pemodelan atau pelatihan (training) model sesuai metode yang Anda gunakan. Adapun sheet test berisi 10 baris data, dengan output kelas (y) yang disembunyikan. Gunakan sheet ini untuk tahap pengujian (testing) model yang sudah dilatih. Nantinya output program Anda untuk data uji ini akan dicocokkan dengan target atau kelas sesungguhnya

Proses yang harus Anda implementasikan ke dalam program (bisa berbentuk fungsi/prosedur):

- Membaca data latih/uji
- Pelatihan atau training model
- Menyimpan model hasil training
- Pengujian atau testing model
- Evaluasi model
- Menyimpan output ke file

Bab II

Implementasi

A. Membaca data latih/uji

```
data =
pd.read_excel('https://github.com/thoriqakhdn/KNN/blob/main/traintest.xlsx?raw=true', index_col='id')
test_data =
pd.read_excel('https://github.com/thoriqakhdn/KNN/blob/main/traintest.xlsx?raw=true', sheet_name='test', index_col='id')[['x1', 'x2', 'x3']]
```

B. Training Model dan Menyimpan Model Hasil Training

Disini kita membagi ¼ atau 20% dari data sebagai validasi data dengan menggunakan metode K-Fold Cross Validation

```
#Membuat K-Fold Cross Validation
validation_data = data.sample(frac=.2)
train_data = data.drop(validation_data.index)
```

Kemudian kita training data dan untuk memudahkan ketika menguji validasi data kita membagi data nya menjadi 2 bagian, satu untuk menyimpan nilai x dan yang satunya untuk menyimpan nilai y

```
#training dataset
train_data_x = train_data[['x1', 'x2', 'x3']].values.tolist()
train_data_y = train_data['y'].values.tolist()

#training validation dataset
validation_data_x = validation_data[['x1', 'x2', 'x3']].values.tolist()
validation_data_y = validation_data['y'].values.tolist()
```

C. Pengujian atau testing model

Ada berbagai macam cara untuk menyelesaikan permasalahan untuk binary classification. Pada tugas besar ini kami menggunakan metode K-NN

K-nearest neighbors atau knn adalah algoritma yang berfungsi untuk melakukan klasifikasi suatu data berdasarkan data pembelajaran (*train data sets*), yang diambil dari k tetangga terdekatnya (*nearest neighbors*).

Pada metode K-NN ini kami membuat 5 komponen fungsi yang memiliki fungsi masing - masing :

1. eucl_dist : Menghitung jarak antar titik (titik baru dengan titik yang dimasukan)

```
#Euclidean Distance untuk menghitung jarak
def eucl_dist(x1, x2):
      distance = ((x1[0] - x2[0])**2) + ((x1[1] - x2[1])**2) +
((x1[2] - x2[2])**2)
    return sqrt(distance)
```

2. klasifikasi : Untuk mengklasifikasikan nilai y dari setiap id pada data train

```
#Mengklasifikasikan y
def klasifikasi(train_list):
    y_0 = 0
    y_1 = 0
    i = 0

        #proses perulangan dari awal list sampai akhir untuk
mengklasifikasikan y dari setiap id
    while i < len(train_list):
        val = train_list[i][1]
        if val == 0:
            y_0 += 1
        elif val == 1:
            y_1 += 1
        i += 1

    if y_0 > y_1:
        return 0
    elif y_0 < y_1:
        return 1
```

3. neighbours : Mengeluarkan hasil klasifikasi dari titik baru ke K tetangga yang ada berdasarkan perbandingan jaraknya

```
#Mencari Neighbors
def neighbours(train_data_x, train_data_y, titik_baru, k):
    train_list = []
    i = 0
    while i != len(train_data_x):
        train_list.append([eucl_dist(train_data_x[i], titik_baru),
train_data_y[i]])
        i += 1
    train_list.sort(key=lambda x: x[0])
    return klasifikasi(train_list[:k])
```

4. prediction : Melakukan pengujian terhadap hasil prediksi untuk menentukan TP, FP , TN dan FN

```
#Menguji hasil aktual dari prediksi
def prediction(k, train_data_x, train_data_y, validation_data_x,
nilai_y=None):
    prediction_x_actual = []
    banyak_data = len(validation_data_x)
    for i in range (0, banyak_data, 1):
        if nilai_y == None:
            prediction_x_actual.append(neighbours(train_data_x,
train_data_y, validation_data_x[i], k))
        else:
            prediction_x_actual.append([neighbours(train_data_x,
train_data_y, validation_data_x[i], k), validation_data_y[i]])
    return prediction_x_actual
```

5. Confusion matrix : Menggunakan confusion matrix untuk menghitung accuracy, precision,recall dan f1 score berdasarkan hasil dari prediction yang sudah dibandingkan dengan hasil aktual, lalu diurutkan untuk menemukan k terbaik

```
#Confusion matrix
def confusion_matrix(train_data_x, train_data_y,
validation_data_x, validation_data_y):
    nilai_k = range(1, 10, 2)
    hasil = []
    for k in nilai_k:
        TP = 0
        TN = 0
        FP = 0
        FN = 0
            prediction_actual = prediction(k, train_data_x,
train_data_y, validation_data_x, validation_data_y)
```

```
        i = 0
        while i != len(prediction_actual):
                            if  prediction_actual[i][0]  ==  1  and
prediction_actual[i][1] == 1:
                TP += 1
                            elif  prediction_actual[i][0]  ==  1  and
prediction_actual[i][1] == 0:
                FP += 1
                            elif  prediction_actual[i][0]  ==  0  and
prediction_actual[i][1] == 1:
                FN += 1
                            elif  prediction_actual[i][0]  ==  0  and
prediction_actual[i][1] == 0:
                TN += 1
            i += 1
        accuracy = (TP+TN)/(TP+TN+FP+FN)
        precision = TP/(TP+FP)
        recall = TP/(TP+FN)
        f1_score = (2*precision*recall)/(precision+recall)
        hasil.append([k, accuracy, f1_score])
    x = 0
    while x < len(hasil):
        print(hasil[x])
        x += 1
    hasil.sort(key=lambda x: (-x[1], x[2]))
     print("K terbaik adalah", hasil[0][0], "dengan hasil F-1 ",
hasil[0][2])
    return hasil[0]
```

Selanjutnya dari k terbaik yang sudah didapatkan menggunakan confusion matrix pada training model akan dilakukan prediction terhadap dataset test untuk mendapatkan nilai y berdasarkan hasil dari training model yang sudah didapatkan.

D. Evaluasi Model dan Menyimpan Output Ke File

Setelah kita membuat metode K-NN dan metode untuk mencari nilai K terbaik, maka selanjutnya kita akan menggunakan metode ini untuk data real berdasarkan data training sebelumnya.

Kemudian hasil dari testing ke real data akan disimpan kedalam file bernama ‘Hasil_testing.xlsx’.

```
def main(data, test_data, train_data_x, train_data_y, validation_data_x,
validation_data_y):

    data_x = data[['x1', 'x2', 'x3']].values.tolist()
    data_y = data['y'].values.tolist()
    test = test_data.values.tolist()

    #Menggunakan training model untuk mendapatkan hasil dari testing
    k_terbaik = confusion_matrix(train_data_x, train_data_y, validation_data_x,
    validation_data_y)[0]
    result = prediction(k_terbaik, data_x, data_y, test)

    #Menyimpan hasil data hasil Testing ke variabel untuk di output file xlsx
    test_result = test_data
    test_result['y'] = result
    print(test_result)
    test_result.to_excel('Hasil_testing.xlsx', sheet_name='test')
    return;
```

Bab III

Kesimpulan

Dari hasil analisis yang kami lakukan dengan menggunakan metode Euclidean Distance untuk menghitung jarak dalam metode K-NN, dan juga menggunakan metode confusion matrix untuk mendapatkan nilai K terbaik, kami menemukan hasil dengan menggunakan dataset training yang tersedia dan kemudian melakukan testing ke dataset real. Hasil yang kami temukan sebagai berikut :

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