Learning Case Study : traintest.xlsx

Dibuat untuk Memenuhi Tugas Pemrograman 3 Mata Kuliah Pengantar Kecerdasan Buatan

Oleh Kelompok 18 :

1. Asfa Amalia Dinata - 1301204498
2. Kelvyn Lukito - 1301200104
3. Mohamad Zulistiyan - 1301204037

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022

Daftar Isi

A. Definisi Learning

Learning adalah salah satu teknik pemecahan masalah dalam cabang keilmuan dalam ilmu komputer, yaitu kecerdasan buatan. Dalam kecerdasan buatan, learning dibagi menjadi supervised dan unsupervised learning. Pada supervised learning, mesin digunakan untuk memetakan input kepada output yang diinginkan. Pada supervised learning mesin juga digunakan untuk menemukan pola dan relasi di antara atribut dan output. Supervised learning dibagi berdasarkan tipe output menjadi regresi dan klasifikasi. Pada regresi outputnya berupa nilai numerik, dan klasifikasi menghasilkan output berupa label (kategorikal). Supervised learning juga dibagi berdasarkan jumlah kelas, menjadi klasifikasi biner, multiclass, dan multilabel.

Sementara pada unsupervised learning mesin mempelajari hubungan antar data atau atribut.

B. Rumusan Masalah

Diberikan file traintest.xlsx yang terdiri dari dua sheet: train dan test, yang berisi dataset untuk problem klasifikasi biner. 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 digunakan. Adapun sheet test berisi 10 baris data, dengan output kelas (y) yang disembunyikan. Sheet ini akan digunakan untuk tahap pengujian model yang sudah dilatih, kemudian output program untuk data uji ini akan dicocokkan dengan target atau class sebenarnya.

C. Definisi KNN

Dalam studi kasus ini, akan digunakan metode K-Nearest Neighbour. K-Nearest Neighbour adalah salah satu algoritma supervised learning. Algoritma ini mengelompokkan data baru menggunakan kemiripan antara data baru dengan data sejumlah k dengan jarak terdekat dari data yang tersedia. K-Nearest Neighbour non parametrik, yang berarti KNN tidak membuat asumsi apapun terhadap data. Algoritma

KNN juga merupakan “lazy learner” karena KNN menerapkan “lazy learning” atau “instant based learning”. Ini artinya algoritma KNN tidak melakukan proses training dalam membangun model. KNN menyimpan data set training dan melakukan learning hanya pada saat membuat prediksi secara real-time. KNN dapat digunakan untuk kasus klasifikasi maupun regresi. Walaupun demikian, KNN lebih sering digunakan dalam proses klasifikasi. Pada K Nearest Neighbor terdapat dua jenis distance metric, yaitu numerik dan non-numerik.

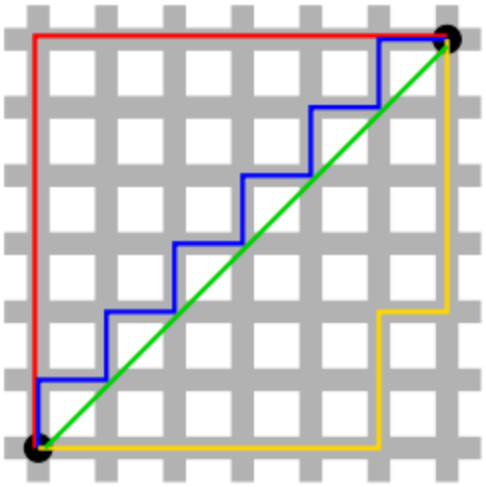
1. Jarak numerik
    a. Euclidean Distance

        Euclidean Distance adalah perhitungan jarak dari 2 buah titik dalam ruang euclidean. Ruang Euclidean diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. Euclidean ini berkaitan dengan Teorema Phytagoras dan biasanya diterapkan pada 1, 2 dan 3 dimensi. Rumus Euclidean distance dapat dinyatakan sebagai berikut ini :

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{in} - x_{jn})^2}$$

    b. Manhattan distance

        Manhattan Distance merupakan salah satu pengukuran yang paling banyak
        digunakan meliputi penggantian perbedaan kuadrat dengan menjumlahkan
        perbedaan absolute dari variable-variable. Fungsi ini hanya akan menjumlahkan
        selisih nilai x dan y dari dua buah titik



Mudahnya, Manhattan Distance dianalogikan seperti gambar diatas. Untuk menghitung Manhattan Distance menggunakan rumus :

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{in} - x_{jn}|$$

c. Minkowski distance

Minkowski distance merupakan sebuah metrik dalam ruang vektor di mana suatu norma didefinisikan (normed vector space) sekaligus dianggap sebagai generalisasi dari Euclidean distance dan Manhattan distance. Dalam pengukuran jarak objek menggunakan minkowski distance biasanya digunakan nilai p adalah 1 atau 2.

Berikut rumus yang digunakan menghitung jarak dalam metode ini:

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

d. Supremum distance

Supremum Distance adalah generalisasi untuk Minkowski Distance. Untuk menghitungnya, kita mencari atribut f yang memberikan perbedaan nilai maksimum antara dua objek

e. Hamming distance

mengukur jarak antara dua string yang ukurannya sama dengan membandingkan simbol-simbol yang terdapat pada kedua string pada posisi yang sama. Hamming distance dari dua string adalah jumlah

simbol dari kedua string yang berbeda. Sebagai contoh Hamming distance antara string ‘toned’ dan ‘roses’ adalah 3.

f. Cosine distance

2. Jarak Non-Numerik

a. Nominal

Perhitungan ini menghasilkan dissimilarity matrix atau matriks jarak. Matriks ini menjelaskan perbedaan pasangan antar m objek. Matriks ini adalah matrik berukuran mxm dengan elemen i,j yang sama dengan nilai ukuran perbedaan jarak antara objek ke i dan ke j. Rumus untuk nominal distance adalah :

$$d(i,j) = \frac{p-m}{p}$$

Untuk :

M = jumlah status yang bernilai sama pada kedua objek

P = jumlah atribut pada objek tersebut

b. Binary symmetric

Pada perhitungan ini juga menghasilkan dissimilarity matrix, hanya saja memperhitungkan atribut yang bernilai 1 untuk objek i atau objek bernilai 0. Persamaan binary symmetric dapat dinyatakan sebagai berikut

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

Untuk :

q = jumlah atribut yang bernilai 1 untuk kedua objek

r = jumlah atribut yang bernilai 1 untuk objek i dan bernilai 0 untuk objek j

s = jumlah atribut yang bernilai 0 untuk objek i dan bernilai 1 untuk objek j

t = jumlah atribut yang bernilai 0 untuk kedua objek

c. Binary asymmetric

Perhitungan ini pun juga menghasilkan dissimilarity matrix, hanya saja memperhitungkan atribut yang bernilai 1 untuk objek i dan bernilai 0 untuk objek j atau sebaliknya tanpa memperhitungkan yang keduanya bernilai 0. Sehingga persamaan untuk binary asymmetric dapat dinyatakan sebagai berikut :

$$d(i,j) = \frac{r+s}{q+r+s}$$

Untuk :

q = jumlah atribut yang bernilai 1 untuk kedua objek

r = jumlah atribut yang bernilai 1 untuk objek i dan bernilai 0 untuk objek j

s = jumlah atribut yang bernilai 0 untuk objek i dan bernilai 1 untuk objek j

D. Implementasi KNN

1. Load library

```
[ ]  import numpy as np
     import pandas as pd
     from scipy.spatial.distance import euclidean
     from google.colab import files
     import io
```

2. Load dataset

```
[ ]  #Digunakan untuk membaca data
     def readData(file_):
         df = file_
         train_ = pd.DataFrame(pd.read_excel(df, sheet_name='train'))
         test_ = pd.DataFrame(pd.read_excel(df, sheet_name='test'))
         return train_, test_
```

3. Distance Metric

```
[ ]  #Untuk menghitung Jarak
     def dist(Train, Test):
         distdmp = []
         for j in range(len(Test[0])):
             dmp = []
             for i in range(len(Train[0])):
                 a = (Train[0][i], Train[1][i], Train[2][i])
                 b = (Test[0][j], Test[1][j], Test[2][j])
                 dmp.append(euclidean(a,b))
             distdmp.append(dmp)
         return distdmp
```

4. Class determining

```
[ ]  #Untuk menentukan Class
     def pick_Class(data, Train, k):
         dataTrain = Train
         Class = []
         for i in range(len(data)):
             x = data[i]
             dmpList = [x for _, x in sorted(zip(x,dataTrain))]
             kelas = max(set(dmpList[0:k]), key = dmpList.count)
             Class.append(kelas)
         return Class
```

5. Model evaluation

```
[ ]  #Evaluasi Model
     def accurate(k):
         data = pd.ExcelFile('traintest(1).xlsx')
         latih = pd.read_excel(data, 'train')
         x1 = latih['x1'].tolist()
         x2 = latih['x2'].tolist()
         x3 = latih['x3'].tolist()
         y = latih['y'].tolist()
         print("Uji coba Evaluasi dengan tetangga K:", k)
         print("Evaluasi Pertama")
         x1Eva1 = x1[0:74]
         x2Eva1 = x2[0:74]
         x3Eva1 = x3[0:74]
         x1Eva1U1 = x1[74:148]
         x1Eva1U2 = x2[74:148]
         x1Eva1U3 = x3[74:148]
         yEva1 = y[74:148]
         arrayEva1 = np.array((x1Eva1, x2Eva1, x3Eva1), dtype = int)
         arrayEva2 = np.array((x1Eva1U1, x1Eva1U2, x1Eva1U3), dtype = int)
         dmp1 = dist(arrayEva1, arrayEva2)
```

```
[ ]      Kelas1 = pick_Class(dmp1, yEva1, k)
         count = 0
         for i in range(len(dmp1)):
             if Kelas1[i] == yEva1[i]:
                 count += 1
         print("Tingkat Error Evaluasi: ", 1 - (count/len(yEva1)))

         print("Evaluasi Kedua")
         x1Eva1 = x1[148:222]
         x2Eva1 = x2[148:222]
         x3Eva1 = x3[148:222]
         x1Eva1U1 = x1[222:296]
         x1Eva1U2 = x2[222:296]
         x1Eva1U3 = x3[222:296]
         yEva1 = y[222:296]
         arrayEva1 = np.array((x1Eva1, x2Eva1, x3Eva1), dtype = int)
         arrayEva2 = np.array((x1Eva1U1, x1Eva1U2, x1Eva1U3), dtype = int)
         dmp1 = dist(arrayEva1, arrayEva2)
         Kelas1 = pick_Class(dmp1, yEva1, k)
         count = 0
```

```
         for i in range(len(dmp1)):
             if Kelas1[i] == yEva1[i]:
                 count += 1
         print("Tingkat Error Evaluasi: ", 1 - (count/len(yEva1)))
```

6. Output file

```
[ ]  #Digunakan untuk menulis output file
     def writeData(nama, output):
         inp = pd.ExcelFile(nama)
         data = pd.read_excel(inp, 'test')
         id = data['id'].tolist()
         x1 = data['x1'].tolist()
         x2 = data['x2'].tolist()
         x3 = data['x3'].tolist()

         df = pd.DataFrame({'id': id,'x1': x1, 'x2': x2, 'x3': x3, 'y': output })
         df.to_excel('output.xlsx', sheet_name='hasil', index=False)
```

7. Main function

```
def main():
    uploaded = files.upload()
    df = pd.ExcelFile(io.BytesIO(uploaded.get('traintest(1).xlsx')))
    xTrain, yTest = readData(df)
    arrayTrain = np.array((xTrain['x1'].tolist(),xTrain['x2'].tolist(),xTrain['x3'].tolist()), dtype= int)
    arrayTest = np.array((yTest['x1'].tolist(),yTest['x2'].tolist(),yTest['x3'].tolist()), dtype= int)
    dmp = dist(arrayTrain, arrayTest)
    k = int(input("Masukkan banyak k(Tetannga) yang diinginkan: "))
    dataTrain = xTrain['y'].tolist()
    Kelas = pick_Class(dmp, dataTrain, k)
    accurate(k)
    writeData('traintest(1).xlsx', Kelas)
```

8. Main program

```
if __name__ == '__main__':
    print("Tugas=03-Learning")
    print("====================")
    main()
    print("====================")
    print("Program Selesai")
```

9. Result (output.xlsx)

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

E. Daftar Pustaka

- Slide Mata Kuliah Kecerdasan Buatan ADF - Nearest Neighbour
- Slide Study Group Laboratorium Kecerdasan Buatan
- Slide Mata Kuliah Kecerdasan Buatan - Introduction to Learning
- https://www.statistics.com/glossary/dissimilarity-matrix/#:~:text=The%20dissimilarity%20matrix%20