LAPORAN
TUGAS PEMROGRAMAN 3
PENGANTAR KECERDASAN BUATAN SEMESTER GENAP
2021/2022

Disusun oleh :

Muhammad Rafli Ramadhan - 1301200204

Shafa Diva Syahira - 1301200157

(IF 44 04)

S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

TAHUN 2022

Daftar isi

BAB I **1**

PENDAHULUAN **1**

1.1 Latar Belakang 1

1.2 Tujuan Observasi 1

BAB 2 **2**

ANALISIS DAN DESAIN **2**

2.1 Spesifikasi Tugas Besar 2

2.2 Metode Modeling 2

2.3 Normalisasi 3

2.4 Validasi 3

BAB 3 **4**

IMPLEMENTASI **4**

3.2 Library yang digunakan 4

3.3 Import Data 4

3.4 Model training 4

3.5 Pengujian Model 5

3.6 Evaluasi Model 6

3.7 Menuliskan File 7

BAB 4 **8**

HASIL DAN KESIMPULAN **8**

4.1 Hasil 8

4.2 Kesimpulan 8

BAB 5 **9**

LAMPIRAN **9**

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam laporan ini kami telah melakukan analisis dan mengimplementasikan machine learning menggunakan algoritma *K Nearest Neighbor* (KNN) yaitu algoritma klasifikasi yang mencari kemiripan dengan menghitung tetangga terdekat yang ada di dalam dataset sebanyak K, dengan K adalah banyaknya tertangga terdekat.

1.2 Tujuan Observasi

Tujuan laporan ini dibuat untuk memenuhi Tugas Pemrograman 3 pada mata kuliah Pengantar Kecerdasan Buatan. dan melakukan analisis terhadap algoritma KNN dengan metode euclidean dan manhattan terhadap data traintest.xlsx untuk mendapatkan akurasi dari modelnya.

BAB 2

ANALISIS DAN DESAIN

2.1 Spesifikasi Tugas Besar

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 (*x*1 sampai *x*3), 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).

id	x1	x2	x3	y
1	60	64	0	1
2	54	60	11	0
3	65	62	22	0
4	34	60	0	1
5	38	69	21	0

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.

2.2 Metode Modeling

a. Pendekatan euclidean
   Formula untuk menemukan jarak 2 titik pada ruang 2 Dimensi

$$d_1(x1, x2) = \sqrt{\sum_p (x1_p - x2_p)^2}$$

b. Pendekatan manhattan
   Formula untuk menemukan jarak antara 2 titik pada matriks berukuran N

$$d_1(x1, x2) = \sum_p |x1_p - x2_p|$$

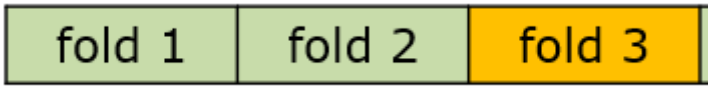
2.3 Normalisasi

Sebelum data dapat di proses, akan dilakukan Normalisasi terlebih dahulu untuk memastikan bahwa data sudah tepat dan dapat digunakan kedalam fungsi yang ada pada metode modeling. Metode normalisasi yang digunakan adalah metode min-max

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$$

2.4 Validasi

Sebelum melakukan testing pada training data, sebaiknya melakukan validasi terlebih dahulu agar K dan Fold dapat dilihat yang paling efektif dan cocok digunakan untuk testing. Metode validasi ini akan menghasilkan persenan akurasi pada setiap fold yang diukur oleh faktor K. Akurasi terbaik dan keunikan pada fold akan menjadi kesimpulan bahwa fold tersebut adalah yang terbaik untuk testing data.



BAB 3

IMPLEMENTASI

3.2 Library yang digunakan

Pada proses implementasi KNN, library yang akan digunakan adalah pandas untuk membaca data berformat .xlsx

3.3 Import Data

```
xls = pd.ExcelFile('https://github.com/raflidev/Tupro-KNN/blob/main/traintest.xlsx?raw=true')
df = pd.read_excel(xls, sheet_name='train')
df2 = pd.read_excel(xls, sheet_name='test').drop(['y'], axis=1)
print(df.head())
```

```
   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
```

3.4 Model training

Normalisasi data :

```
# normalisasi dengan metode min-max
def normalize(df, column, factor=1):
    result = df.copy()
    for col in column:
        max_value = df[col].max()
        min_value = df[col].min()
        result[col] = ((df[col] - min_value) / (max_value - min_value))*factor
    return result

print(normalize(df, df.columns.values[1:]))
print(normalize(df2, df2.columns.values[1:]))
```

```
TRAIN DATA :
      id        x1        x2        x3    y
0      1  0.566038  0.545455  0.000000  1.0
1      2  0.452830  0.181818  0.211538  0.0
2      3  0.660377  0.363636  0.423077  0.0
3      4  0.075472  0.181818  0.000000  1.0
4      5  0.150943  1.000000  0.403846  0.0
..   ...       ...       ...       ...  ...
291  292  0.547170  0.545455  0.019231  1.0
292  293  0.660377  0.818182  0.000000  1.0
293  294  0.433962  0.636364  0.230769  0.0
294  295  0.509434  0.545455  0.019231  0.0
295  296  0.452830  0.090909  0.134615  1.0

[296 rows x 5 columns]
TEST DATA :
    id    x1        x2        x3
0  297  0.04  0.090909  0.666667
1  298  1.00  0.727273  0.000000
2  299  0.64  0.181818  1.000000
3  300  0.28  0.454545  1.000000
4  301  0.12  0.181818  0.000000
5  302  0.48  0.000000  0.333333
6  303  0.56  0.727273  1.000000
```

Menyimpan model hasil training :

a. Pendekatan Euclidian

```
# Metode Modelling
def euclidean_distance(x, y):
  result = []
  for i in range(len(x)):
    res = ((x['x1'][i] - y['x1'])**2) + ((x['x2'][i] - y['x2'])**2) + ((x['x3'][i] - y['x3'])**2)
    result.append([res**0.5, x['y'][i]])
  return result
```

b. Pendekatan Manhattan

```
def manhattan_distance(x, y):
  result = []
  for i in range(len(x)):
    res = (abs(x['x1'][i] - y['x1']) + abs((x['x2'][i] - y['x2'])) + abs((x['x3'][i] - y['x3'])))
    result.append([res, x['y'][i]])
  return result
```

3.5 Pengujian Model

a. Pendekatan Euclidian

```
def k_euclidean(df, dfTest, k):
  result = []
  for i in range(len(dfTest)):
    distance = euclidean_distance(df, dfTest.iloc[i])
    distance.sort()
    resDist = distance[:k]

    a, b = assign_knn(resDist, k)
    if (a > b):
      result.append([dfTest.iloc[i]['id'], 1])
    else:
      result.append([dfTest.iloc[i]['id'], 0])
  return result
```

b. Pendekatan Manhattan

```
def k_manhattan(df, dfTest, k):
  result = []
  for i in range(len(dfTest)):
    distance = manhattan_distance(df, dfTest.iloc[i])
    distance.sort()
    resDist = distance[:k]
    a, b = assign_knn(resDist, k)
    if (a > b):
      result.append([dfTest.iloc[i]['id'], 1])
    else:
      result.append([dfTest.iloc[i]['id'], 0])
  return result
```

Fungsi ini untuk menentukan y pada metode Euclidian dan Manhattan 1 atau 0

```
# menentukan 1 atau 0
def assign_knn(res, k):
  a,b = 0, 0
  for i in range(k):
    if(res[i][1] == 1):
      a += 1
    else:
      b += 1
  return a,b
```

3.6 Evaluasi Model

```
# validasi dataset dengan rumus modelling
def validate(fold, k):
  euc = []
  man = []
  for i in range(len(fold)):
    euclidean, manhattan = knn(fold[i][0], fold[i][1], k)
    euc.append(check_accurracy(euclidean))
    man.append(check_accurracy(manhattan))
  return euc, man

# cek akurasi (masuk dalam validasi)
def check_accurracy(modeling):
  res = 0
  for i in range(len(modeling)):
    if(modeling[i][1] == df['y'][i]):
      res += 1
  return res/len(modeling)
```

```
# metode KNN dilakukan
def knn(df, dfTest, k):
  euclidean = k_euclidean(df, dfTest, k)
  manhattan = k_manhattan(df, dfTest, k)

  return euclidean, manhattan;
```

```
[11] # Start
    df = normalize(df, df.columns.values[1:])
    dfTest = normalize(df2, df2.columns.values[1:])

    # fold untuk kebutuhan validasi
    fold1 = (df.iloc[0:98],  df.iloc[98:].drop('y', axis=1))
    fold2 = (df.iloc[98:196].reset_index(), pd.concat([df.iloc[0:98].reset_index().drop('y', axis=1), df.iloc[98:].reset_index().drop('y', axis=1)]))
    fold3 = (df.iloc[196:].reset_index(), df.iloc[0:98].reset_index())

    # hasil validasi
    euclidean, manhattan = validate([fold1, fold2, fold3], 3)
    print(euclidean)
    print(manhattan)

    euclideanKNN, manhattanKNN = knn(df, dfTest, 3)
    dfEuc = pd.DataFrame(euclideanKNN, columns=['id', 'y'])
    dfMan = pd.DataFrame(manhattanKNN, columns=['id', 'y'])
```

```
[0.7070707070707071, 0.75, 0.6530612244897959]
[0.702020202020202, 0.75, 0.6122448979591837]
```

3.7 Menuliskan File

```
with pd.ExcelWriter('KNN.xlsx') as writer:
    dfEuc.to_excel(writer, sheet_name='euclidean', index=False)
    dfMan.to_excel(writer, sheet_name='manhattan', index=False)
```

BAB 4

HASIL DAN KESIMPULAN

4.1 Hasil

Hasil dari program yang telah di tulis kedalam file .xlsx

a. Hasil Euclidean

id	y
297	1
298	1
299	0
300	0
301	1
302	0
303	0
304	1
305	0
306	1

b. Hasil Manhattan

id	y
297	1
298	1
299	0
300	0
301	1
302	0
303	1
304	1
305	1
306	1

4.2 Kesimpulan

Setelah mempelajari dan menerapkan algoritma KNN pada tugas pemrograman 3 ini, kami simpulkan bahwa algoritma KNN merupakan algoritma yang berfungsi untuk mengklasifikasikan data berdasarakan tetangga terdekat sebanyak K, yang bisa menggunakan metode euclidian dan manhattan.

BAB 5

LAMPIRAN

Link Colab:
https://colab.research.google.com/drive/1UR5wT0EqB2tDxicxW2vBZWlRVo1Ik-B-?usp=sharing
Link github: https://github.com/raflidev/Tupro-KNN