



Fakultas Informatika

School of Computing

Telkom University

LAPORAN TUGAS PROGRAM 3

LEARNING-K NEAREST NEIGHBOR

DANI AGUNG PRASTIYO

1301154646

TELKOM UNIVERSITY

Daftar Isi

1.1	Deskripsi Program.....	2
1.2	Metode Rancangan Program.....	2
1.3	Cara Penggunaan Program.....	5
1.4	Cara Kerja Program	5
1.5	Pengujian Program.....	8
1.6	Kesimpulan	13
Daftar Pustaka.....		14

1.1 Deskripsi Program

Pada tugas Program 3 ini diberikan suatu himpunan data berisi 5000 berita dengan empat atribut: Jumlah Like, Emosi Komentar dan Provokasi yang bernilai 0 sampai 100, serta atribut kelas *Hoax* yang bernilai 1 yang berarti “Hoax” dan 0 yang berarti “Bukan Hoax”, seperti terdapat dalam file “Dataset Tugas 3.xlsx”. Dengan menggunakan 4000 data dalam sheet “DataTrain”, sebagai data latih untuk mendeteksi apakah 1000 berita yang belum diketahui kelasnya, dalam sheet “DataTest, adalah berita bohong (*hoax*) atau bukan. Metode klasifikasi yang digunakan dalam Tugas Program 3 ini adalah **k-Nearest Neighbor (kNN)**.

1.2 Metode Rancangan Program

A. Spesifikasi Program

1. Program **k-Nearest Neighbor** ini dibangun menggunakan Bahasa pemrograman python dengan interpreter 3.6
2. IDE yang digunakan dalam membangun program ini adalah PyCharm Prefisional 2017.1
3. Program ini dapat dijalankan di system operasi windows atau linux

B. Algoritma k-Nearest Neighbor (kNN).

Algoritma k-Nearest Neighbor (k-NN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan data uji. k-Nearest Neighbor termasuk kelompok instance-based learning. Algoritma ini juga merupakan salah satu tehnik lazy learning dikarenakan hanya menyimpan sebagian atau seluruh data latih, kemudian menggunakan data latih tersebut ketika proses prediksi. K-Nearest Neighbor dilakukan dengan mencari kelompok k objek dalam data training yang paling dekat (mirip) dengan objek pada data baru atau data testing (Wu, 2009). Adapun K-Nearest Neighbor termasuk kategori memory based method, yaitu seluruhnya atau sebagian dari training set tetap disimpan dan dipakai dalam proses klasifikasi.

Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi sampel dari uji yang baru. Pengukuran jarak ketetanggaan yang digunakan seperti : Euclidean Distance, Mikowski Distance, dan Manhattan Distance. Dan yang paling sering digunakan adalah Euclidean Distance.

Pada fase training, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data training sample. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor training sample dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut.

Nilai k yang terbaik untuk algoritma ini tergantung pada data, secara umumnya, nilai k yang lebih tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Secara umum nilai k optimal yang sering digunakan berkisar diantara 3-10 atau \sqrt{n} dimana n merupakan jumlah data latih. Itu akan menghasilkan hasil yang lebih baik dibandingkan dengan 1NN.

Adapun penerapan algoritma K-Nearest Neighbor terdiri dari empat langkah, yaitu :

1. Menentukan parameter k (jumlah ketetanggaan yang paling dekat).
2. Menghitung jarak ketetanggaan (similarity measure) masing-masing objek terhadap data sampel yang diberikan.
3. Mengurutkan objek-objek tersebut kedalam kelompok yang mempunyai jarak terkecil sampai terbesar.
4. Mengumpulkan kategori Y (Klasifikasi Nearest Neighbor).
5. Dengan menggunakan kategori Nearest Neighbor yang paling mayoritas maka dapat
6. diprediksi nilai query instance yang telah dihitung.

K-Nearest Neighbor memiliki beberapa kelebihan yaitu tangguh terhadap data template yang noisy dan efektif apabila jumlah data template besar. Sedangkan kelemahan dari metode ini adalah perlunya menentukan nilai parameter k (jumlah ketetanggaan terdekat), pembelajaran berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut mana yang harus digunakan untuk mendapatkan hasil yang terbaik, dan biaya komputasi yang cukup tinggi karena diperlukan perhitungan jarak dari tiap query instance pada keseluruhan training sample.



C. K-Fold Validation

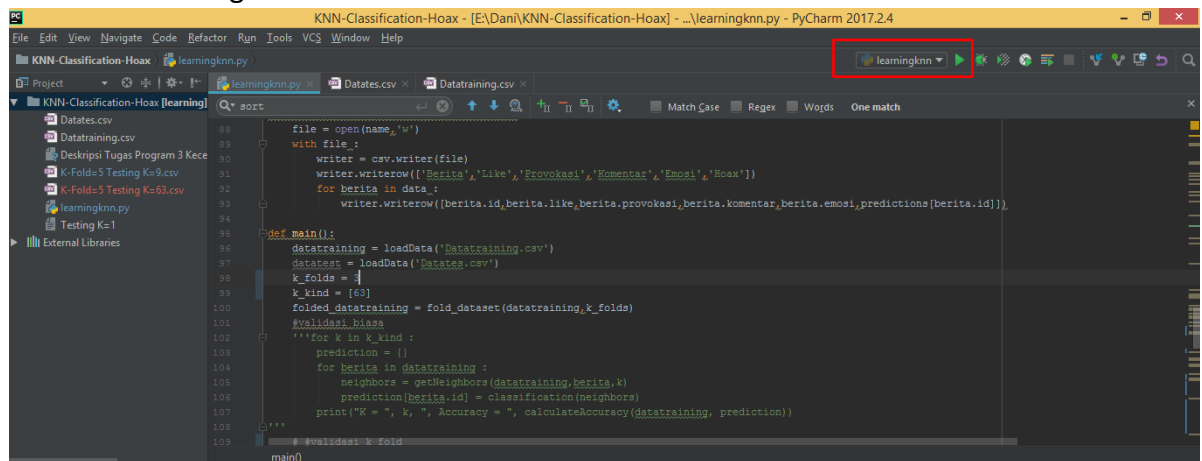
K-fold cross validation adalah sebuah teknik intensif komputer yang menggunakan keseluruhan data yang ada sebagai training set dan test set [BEN04]. Seluruh data secara acak dibagi menjadi K buah subset B_k dengan ukuran yang sama dimana B_k merupakan himpunan bagian dari $\{1, \dots, n\}$ sedemikian sehingga $\bigcup_{k=1}^K B_k = \{1, \dots, n\}$

dan $B_i \cap B_k = \emptyset \ (j \neq k)$. Setelah itu dilakukan iterasi sebanyak K kali. Pada iterasi ke k, subset B_k menjadi test set, sedangkan subset yang lain menjadi training set. Setelah itu dihitung nilai rata-rata error dengan menggunakan hasil dari K buah iterasi.

Kelebihan dari metode ini adalah tidak adanya masalah dalam pembagian data. Setiap data akan menjadi test set sebanyak satu kali dan akan menjadi training set sebanyak K-1 kali. Kekurangan dari metode ini adalah algoritma pembelajaran harus dilakukan sebanyak K kali yang berarti menggunakan K kali waktu komputasi.[SCH97]

1.3 Cara Penggunaan Program

1. Program ini dibangun dengan menggunakan Bahasa pemrograman python 3
2. Lakukan instalasi interpreter Python 3, anda dapat mendownload di <https://www.python.org/downloads/>
3. Buka Program menggunakan IDE JetBrains Pycharm atau sejenisnya yang mendukung Python, anda dapat mendownloadnya di <https://www.jetbrains.com/pycharm/download/>
4. Buka File Program learningknn.py
5. Lakukan Running



1.4 Cara Kerja Program

1. Library yang dipakai adalah seperti gambar dibawah

```
import csv
import random
import math
import operator
```

2. Melakukan Inisialisasi semua method

```
class Berita:
    def __init__(self, id, like, provokasi, komentar, emosi, hoax):
        self.id = id
        self.like = like
        self.provokasi = provokasi
        self.komentar = komentar
        self.emosi = emosi
        self.hoax = hoax
```

3. Fungsi Meload Data CSV

```
def loadData(file):
    outputSet = []
    with open(file, 'r') as csvfile:
        lines = csv.reader(csvfile)
        dataset = list(lines)
        for x in range(1, len(dataset)):
            hoax = 0
            if(dataset[x][5] == '?') :
                hoax = -1
            else :
                hoax = float(dataset[x][5])
            berita =
Berita(dataset[x][0],float(dataset[x][1]),float(dataset[x][2]),float(dataset[x][3]),float(dataset[x][4]),hoax)
            outputSet.append(berita)
    return outputSet
```

4. Fungsi Eulid Distance

```
def euclideanDistance(berita1, berita2):
    distance = 0
    distance += pow(berita1.like - berita2.like,2) + pow(berita1.provokasi -
berita2.provokasi,2) + pow(berita1.komentar - berita2.komentar,2) +
pow(berita1.emosi - berita2.emosi,2)
    return math.sqrt(distance)
```

5. Fungsi K-fold untuk membagi dataset dengan kfold

```
def fold_dataset(dataset, kfold) :
    folded_dataset = []
    folded_n_data = len(dataset) / kfold
    for x in range(0, kfold):
        folded_dataset.append(dataset[int(x * folded_n_data):int(((x + 1) *
folded_n_data))])
    return folded_dataset
```

6. Sekarang kita memiliki ukuran kesamaan, kita dapat menggunakannya untuk mengumpulkan contoh yang paling mirip untuk contoh yang tak terlihat. Ini adalah proses lurus ke depan untuk menghitung jarak untuk semua contoh dan memilih subset dengan nilai jarak terkecil. Berikut adalah fungsi getNeighbors yang mengembalikan sekumpulan tetangga yang paling mirip dari set pelatihan untuk

contoh uji yang diberikan (menggunakan fungsi euclideanDistance yang sudah ditentukan)

```
def getNeighbors(dataset, berita, k):
    distances = []
    for x in range(len(dataset)):
        distance = euclideanDistance(berita, dataset[x])
        distances.append((dataset[x], distance))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

7. Melakukan Klasifikasi jika Hoax 1 dan jika 0 tidak hoax

```
def classification(neighbors):
    hoax = 0
    not_hoax = 0
    for neighbor in neighbors:
        if neighbor.hoax == 1:
            hoax += 1
        else:
            not_hoax += 1
    if hoax > not_hoax:
        return float(1)
    else:
        return float(0)
```

8. Melakukan proses klasifikasi K-fold berdasarkan kelasnya

```
def fold_classification(fold_class):
    hoax = 0
    not_hoax = 0
    for x in fold_class:
        if x == 1:
            hoax += 1
        else:
            not_hoax += 1
    if hoax > not_hoax:
        return float(1)
    else:
        return float(0)
```

9. Cara mudah untuk mengevaluasi keakuratan model adalah menghitung rasio total prediksi yang benar dari semua prediksi yang dibuat, yang disebut keakuratan klasifikasi. Berikut adalah fungsi calculateAccuracy yang menghitung total prediksi yang benar dan mengembalikan keakuratannya sebagai persentase dari klasifikasi yang benar.

```
def calculateAccuracy(dataset, predictions):
    correct = 0
    for x in range(len(dataset)):
        print(dataset[x].id, " = ", dataset[x].hoax, ", Predictions = ",
              predictions[dataset[x].id])
        if dataset[x].hoax == predictions[dataset[x].id]:
            correct += 1
    return (correct / float(len(dataset))) * 100.0
```

10. Fungsi Mengekspor Ke File CSV


```
def writeToCsv(name, data, predictions) :
    file = open(name, 'w')
    with file :
        writer = csv.writer(file)

    writer.writerow(['Berita', 'Like', 'Provokasi', 'Komentar', 'Emosi', 'Hoax'])
    for berita in data :

    writer.writerow([berita.id, berita.like, berita.provokasi, berita.komentar, berita.emosi, predictions[berita.id]])
```

11. Fungsi inputan nilai K dan Nilai K-fold secara manual

```
def main():
    datatraining = loadData('Datatraining.csv')
    datatest = loadData('Datates.csv')
    k_folds = 7
    k_kind = [3, 5, 7, 9, 15, 17, 77]
    folded_datatraining = fold_dataset(datatraining, k_folds)
```

1.5 Pengujian Program

Cara Pengujian program dengan cara membrute force nilai K dengan hasil yang paling optimal, dari paper yang kami baca Nilai k yang terbaik untuk algoritma ini tergantung pada data, secara umumnya, nilai k yang lebih tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Secara umum nilai k optimal yang sering digunakan berkisar diantara 3-10 atau \sqrt{n} dimana n merupakan jumlah data latih. Itu akan menghasilkan hasil yang lebih baik dibandingkan dengan 1NN.

1. Pengujian K= 3 dengan K-Fold = 5 diperoleh akurasi 66%

learningknn	
Q K =	
B3989	1.0 , Predictions = 1.0
B3990	= 1.0 , Predictions = 1.0
B3991	= 0.0 , Predictions = 0.0
B3992	= 0.0 , Predictions = 0.0
B3993	= 0.0 , Predictions = 0.0
B3994	= 0.0 , Predictions = 0.0
B3995	= 1.0 , Predictions = 0.0
B3996	= 0.0 , Predictions = 0.0
B3997	= 0.0 , Predictions = 1.0
B3998	= 1.0 , Predictions = 0.0
B3999	= 0.0 , Predictions = 0.0
B4000	= 0.0 , Predictions = 1.0
K = 3 , Accuracy = 66.225	
B0001	= 1.0 , Predictions = 0.0
B0002	= 1.0 , Predictions = 0.0

2. Pengujian K = 5 dan K-Fold = 5 diperoleh akurasi 67%

learningknn	
Q K =	
B3989	1.0 , Predictions = 1.0
B3990	= 1.0 , Predictions = 1.0
B3991	= 0.0 , Predictions = 0.0
B3992	= 0.0 , Predictions = 0.0
B3993	= 0.0 , Predictions = 1.0
B3994	= 0.0 , Predictions = 0.0
B3995	= 1.0 , Predictions = 0.0
B3996	= 0.0 , Predictions = 0.0
B3997	= 0.0 , Predictions = 1.0
B3998	= 1.0 , Predictions = 0.0
B3999	= 0.0 , Predictions = 0.0
B4000	= 0.0 , Predictions = 1.0
K = 5 , Accuracy = 67.025	
B0001	= 1.0 , Predictions = 0.0
B0002	= 1.0 , Predictions = 0.0

3. Pengujian dengan K = 7 dan K-Fold = 5 diperoleh akurasi 68%

```
learningknn learningknn
Q K =
B3996 = 0.0 , Predictions = 0.0
B3997 = 0.0 , Predictions = 1.0
B3998 = 1.0 , Predictions = 0.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 7 , Accuracy = 68.125
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
B0003 = 0.0 , Predictions = 0.0
```

4. Pengujian dengan K = 9 dan K-Fold = 5 diperoleh akurasi 68,67%

```
learningknn learningknn
Q K =
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 9 , Accuracy = 68.675
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
B0003 = 0.0 , Predictions = 0.0
B0004 = 1.0 , Predictions = 1.0
B0005 = 0.0 , Predictions = 0.0
```

5. Pengujian dengan K = 15 dan K-Fold = 5 diperoleh akurasi 69,5%

```
learningknn learningknn
Q K =
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 15 , Accuracy = 69.525
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
B0003 = 0.0 , Predictions = 0.0
```

6. Pengujian dengan K=17 dan K-Fold = 5 diperoleh akurasi 69,22 %

```
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 17 , Accuracy = 69.22500000000001
Process finished with exit code 0
```

7. Pengujian dengan K=3 dan K-Fold = 7 diperoleh akurasi 67%

```
learningknn learningknn
K =
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 3 , Accuracy = 67.375
B0001 = 1.0 , Predictions = 0.0
B0002 = 1.0 , Predictions = 0.0
B0003 = 0.0 , Predictions = 0.0
```

8. Pengujian dengan K=5 dan K-Fold = 7 diperoleh akurasi 68,77%

```
Run: learningknn learningknn
K =
B3995 = 1.0 , Predictions = 0.0
B3996 = 0.0 , Predictions = 0.0
B3997 = 0.0 , Predictions = 1.0
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 5 , Accuracy = 68.77499999999999
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
```

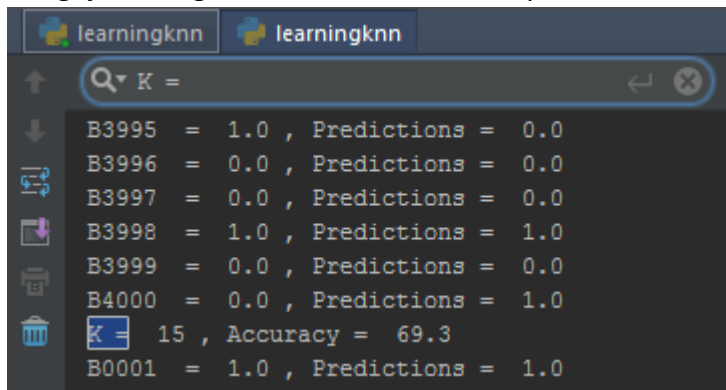
9. Pengujian dengan K=7 dan K-Fold = 7 diperoleh akurasi 69,225%

```
Run: learningknn learningknn
K =
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 7 , Accuracy = 69.22500000000001
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
B0003 = 0.0 , Predictions = 0.0
```

10. Pengujian dengan K=9 dan K-Fold = 7 diperoleh akurasi 69,675%

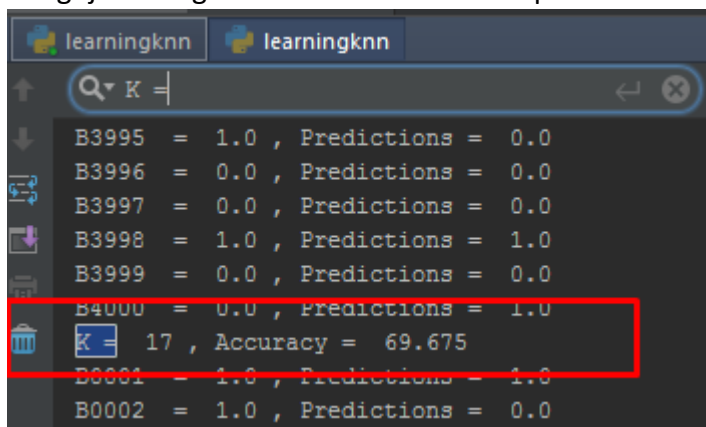
```
Run: learningknn learningknn
K =
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 9 , Accuracy = 69.675
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
B0003 = 0.0 , Predictions = 0.0
```

11. Pengujian dengan K=15 dan K-Fold = 7 diperoleh akurasi 69,3%



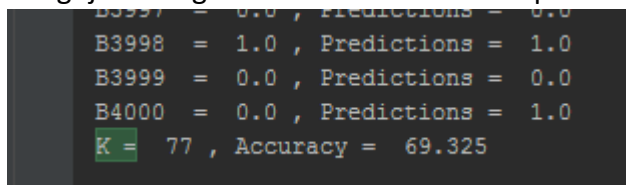
```
learningknn learningknn
K =
B3995 = 1.0 , Predictions = 0.0
B3996 = 0.0 , Predictions = 0.0
B3997 = 0.0 , Predictions = 0.0
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 15 , Accuracy = 69.3
B0001 = 1.0 , Predictions = 1.0
```

12. Pengujian dengan K=17 dan K-Fold = 7 diperoleh akurasi 69,675%



```
learningknn learningknn
K =
B3995 = 1.0 , Predictions = 0.0
B3996 = 0.0 , Predictions = 0.0
B3997 = 0.0 , Predictions = 0.0
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 17 , Accuracy = 69.675
B0001 = 1.0 , Predictions = 1.0
B0002 = 1.0 , Predictions = 0.0
```

13. Pengujian dengan K=77 dan K-Fold = 7 diperoleh akurasi 69,325%



```
B3997 = 0.0 , Predictions = 0.0
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 77 , Accuracy = 69.325
```

14. Pengujian dengan K=63 dan K-Fold = 7 diperoleh akurasi 69,47%

```
Run: learningknn learningknn
B3990 = 1.0 , Predictions = 1.0
B3991 = 0.0 , Predictions = 0.0
B3992 = 0.0 , Predictions = 0.0
B3993 = 0.0 , Predictions = 0.0
B3994 = 0.0 , Predictions = 0.0
B3995 = 1.0 , Predictions = 0.0
B3996 = 0.0 , Predictions = 0.0
B3997 = 0.0 , Predictions = 0.0
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 63 , Accuracy = 69.475
Process finished with exit code 0
```

15. Pengujian dengan K=63 dan K-Fold = 4 diperoleh akurasi 69,77%

```
Run: learningknn learningknn
B3996 = 0.0 , Predictions = 0.0
B3997 = 0.0 , Predictions = 0.0
B3998 = 1.0 , Predictions = 1.0
B3999 = 0.0 , Predictions = 0.0
B4000 = 0.0 , Predictions = 1.0
K = 63 , Accuracy = 69.77499999999999
Process finished with exit code 0
```

1.6 Kesimpulan

Setelah dilakukan pengujian, diperoleh nilai K yang paling Optimal yaitu K = 63 dan K-Fold = 4 dengan akurasi 69,77 yang paling tinggi diantara yang lain dan karena menurut paper untuk menentukan nilai \sqrt{n} dimana n merupakan jumlah data latih. Itu akan menghasilkan hasil yang lebih baik, karena data training 4000, jadi $\sqrt{4000}$ adalah 63 jadi saya memutuskan untuk memilih K = 63 dan K-Fold = 7.

Daftar Pustaka

- Dudani, S. A. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*. Diambil kembali dari The Distance-Weighted k-Nearest-Neighbor Rule
- Sukma, A. (2017, 12 2). *K-NEAREST-NEIGHBOR Informational Retrival*. Diambil kembali dari Unair: http://web.unair.ac.id/admin/file/f_41382_STKI-KEL-2_K-NEAREST-NEIGHBOR.pdf
- Suyanto (2014). *Artificial Intelligence Searching-Reasoning-Planning-Learning*. Informatika, Bandung.