

**Laporan tugas program 3**

**Learning-K Nearest Neighbor**

Dani Agung Prastiyo

1301154646

TELKOM UNIVERSITY

Daftar Isi

[1.1 Deskripsi Program 2](#_Toc500016677)

[1.2 Metode Rancangan Program 2](#_Toc500016678)

[1.3 Cara Penggunaan Program 5](#_Toc500016679)

[1.4 Cara Kerja Program 5](#_Toc500016680)

[1.5 Pengujian Program 8](#_Toc500016681)

[1.6 Kesimpulan 13](#_Toc500016682)

[Daftar Pustaka 14](#_Toc500016683)

## 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)**.

## Metode Rancangan Program

1. **Spesifikasi Program**
2. Program **k-Nearest Neighbor** ini dibangun menggunakan Bahasa pemograman python dengan interpreter 3.6
3. IDE yang digunakan dalam membangun program ini adalah PyChram Prefisional 2017.1
4. Program ini dapat dijalankan di system operasi windows atau linux
5. **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 ketetanggan yang digunakan seperti : Euclidean Distance, Mikwoski Distance, dan Manhattam Distance. Dan yang paling sering digunakan adalah Euclidean Distance.

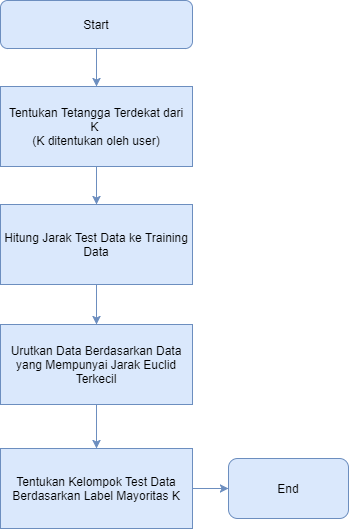
Pada fase training, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data training sample. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor training sample dihitung, 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 √ 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.



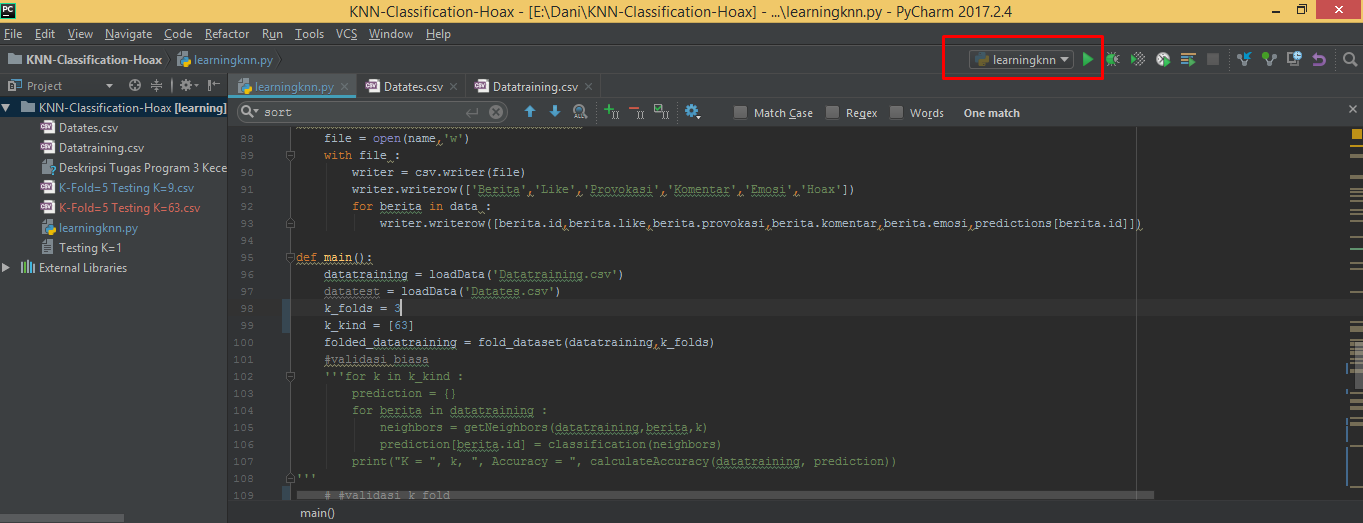
1. K-Fold Validation

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 Bk dengan ukuran yang sama dimana Bk merupakan himpunan bagian dari {1,...,n} sedemikian sehingga  dan . Setelah itu dilakukan iterasi sebanyak K kali. Pada iterasi ke k, subset Bk 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]

## Cara Penggunaan Program

1. Program ini dibagun dengan menggunakkan Bahasa pemograman 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. Lakukkan Running



## Cara Kerja Program

1. Library yang dipakai adalah seperti gambar dibawah

import csv  
import random  
import math  
import operator

1. Melakukkan Inisalisasi 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

1. 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

1. 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)

1. 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

1. 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

1. Melakukkan 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)

1. Melakukkan 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)

1. 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

1. 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]])

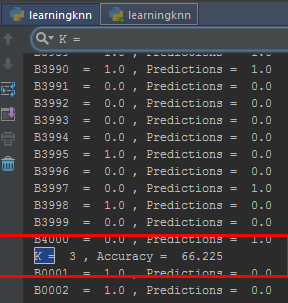
1. Fungsi inputan nilai K dan Nilai K-fold secara manual

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)

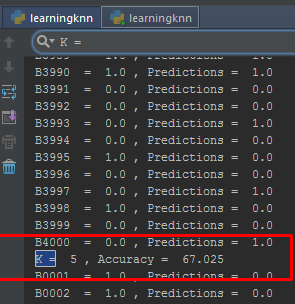
## Pengujian Program

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 √ dimana n merupakan jumlah data latih. Itu akan menghasilkan hasil yang lebih baik dibandingkan dengan 1NN.

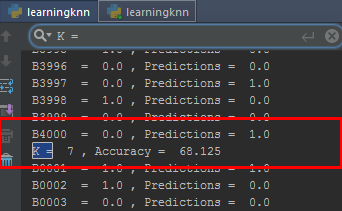
1. Pengujian K= 3 dengan K-Fold = 5 diperoleh akurasi 66%



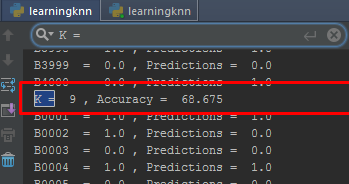
1. Pengujian K = 5 dan K-Fold = 5 diperoleh akurasi 67%



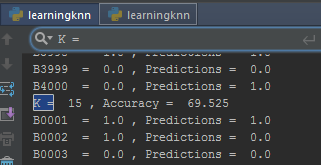
1. Pengujian dengan K = 7 dan K-Fold = 5 diperoleh akurasi 68%



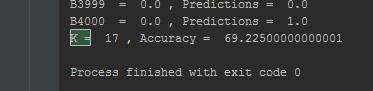
1. Pengujian dengan K = 9 dan K-Fold = 5 diperoleh akurasi 68,67%



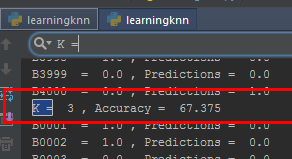
1. Pengujian dengan K = 15 dan K-Fold = 5 diperoleh akurasi 69,5%



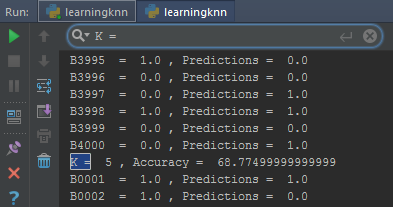
1. Pengujian dengan K=17 dan K-Fold = 5 diperoleh akurasi 69,22 %



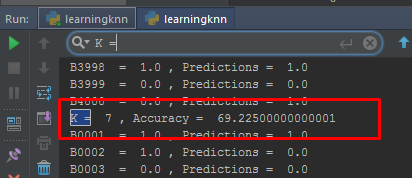
1. Pengujian dengan K=3 dan K-Fold = 7 diperoleh akurasi 67%



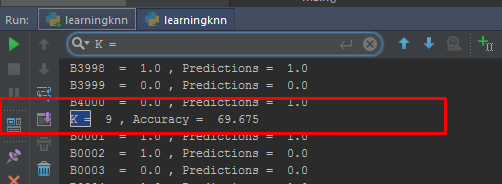
1. Pengujian dengan K=5 dan K-Fold = 7 diperoleh akurasi 68,77%



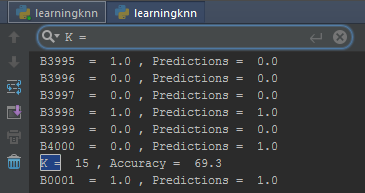
1. Pengujian dengan K=7 dan K-Fold = 7 diperoleh akurasi 69,225%



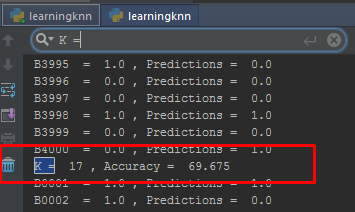
1. Pengujian dengan K=9 dan K-Fold = 7 diperoleh akurasi 69,675%



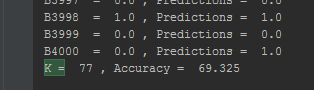
1. Pengujian dengan K=15dan K-Fold = 7 diperoleh akurasi 69,3%



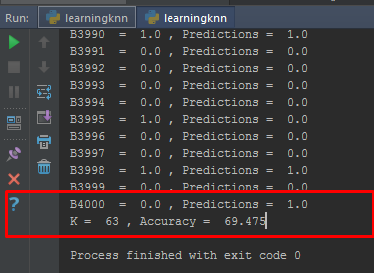
1. Pengujian dengan K=17 dan K-Fold = 7 diperoleh akurasi 69,675%



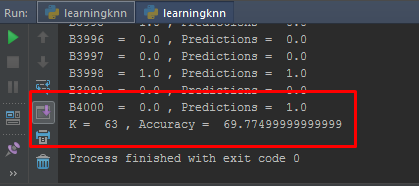
1. Pengujian dengan K=77 dan K-Fold = 7 diperoleh akurasi 69,325%



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



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



## Kesimpulan

Setelah dilakukkan 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 √ 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.