LAPORAN TUGAS PROGRAM 3

Artificial Intelligent

k-Nearest Neighbour (k-NN)

Disusun Oleh:
Aditya Alif Nugraha (1301154183)
IF 39-01

PRODI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2017

Deskripsi Masalah

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". Gunakan 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)**. Anda dapat membagi kembali data latih tersebut menjadi data latih dan validasi (dapat menggunakan metode cross validation) untuk mengukur seberapa akurat system klasifikasi yang sudah anda bangun. Dataset yang digunakan dalam Tugas Program 3 dapat diunduh pada link berikut: https://drive.google.com/file/d/0BwM-1JKzh7XgV3kyanFiLUhjUFE/view?usp=sharing.

Rancangan Metode

Untuk melakukan klasifikasi dengan dengan data *training* yang memiliki label, maka algoritma *supervised learning* dapat digunakan. Salah satu algoritma supervised learning yaitu k-*Nearest Neighbor*.

Definisi k-*Nearest Neighbor* (k-NN)

Algoritma k-NN adalah sebuah metode untuk melakukan klasifikasi terhadap objek beradasarkan data *training* yang jaraknya paling dekat dengan objek tersebut. Tujuan dari algoritma ini yaitu untuk mengklasifikasikan objek baru berdasarkan atribut dan sampel-sampel dari data training.

Tahapan algoritma k-NN yaitu:

1. Menentukan parameter k (jumlah tetangga terdekat).
2. Menghitung kuadrat jarak eucliden objek *test* terhadap data *training* yang diberikan.
3. Mengurutkan hasil tahapan ke-2 dari terkecil hingga terbesar.
4. Mengumpulkan kelas dari data *training* sejumlah parameter k (tetangga terdekat).
5. Dengan menggunakan kelas dari tetangga terdekat yang paling mayoritas, maka dapat diprediksikan kategori objek.

Rancangan Input

Input dari dataset yang digunakan dalam tugas ini memiliki 4 atribut/*features* yang terdiri dari Like, Emosi Komentar dan Provokasi yang bernilai 0 hingga 100. Dan terdapat 1 atribut kelas Hoax yang bernilai 1 yang berarti "Hoax" dan 0 yang berarti "Bukan Hoax".

Total data pada dataset sebanyak 5000 baris. Dimana 4000 baris terdapat pada *sheet* "DataTrain" dan 1000 baris pada *sheet* "DataTest". Pada *sheet "*DataTest*"*, kelas Hoax dari data tersebut belum diketahui. Kelas dari data tersebut akan dicari berdasarkan data *train* dengan menggunakan algoritma k-NN.

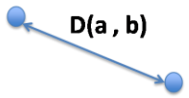
Proses Klasifikasi

Proses klafisifikasi menggunakan *k*-NN sangatlah sederhana. Sesuatu objek tak dikenal dinyatakan tergabung dalam suatu kelas berdasarkan jumlah kelas mayoritas dari tetangga terdekatnya.

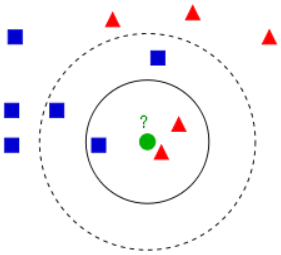
Perhitungan jarak terdekat dapat dilakukan dengan rumus *Euclidean* dengan menghitung jarak setiap data *train* dari data *test*-nya.

Rumus *Euclidean* dapat dilihat dibawah ini:

$$D(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$$



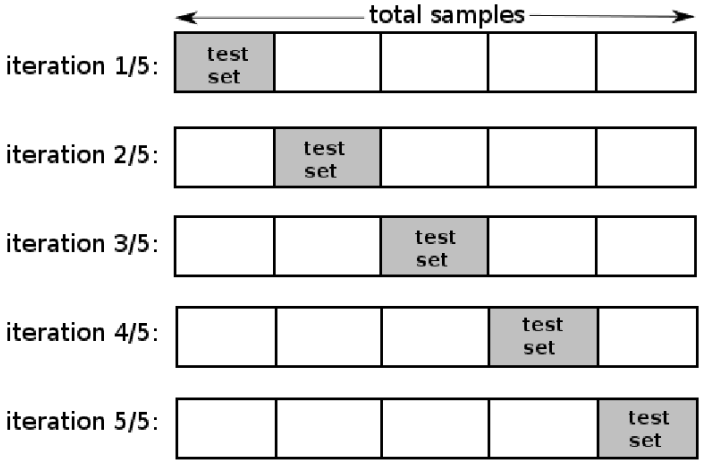
Dari hasil perhitungan jarak tersebut, akan diurutkan dari hasil yang terkecil hingga terbesar. Kemudian dari hasil pengurutan tersebut, akan diambil kelas dari hasil terkecil sebanyak nilai k yang telah ditentukan. Setelah itu akan dilihat mayoritas dari kelas terdekatnya dan kelas objek yang tak dikenal tersebut dapat ditentukan dari kelas mayoritas objek terdekatnya.



Menentukan Nilai K

Menentukan nilai k yang paling optimum dalam k-NN tidaklah mudah. Nilai k terbaik tergantung pada data yang digunakan. Jika nilai k tinggi, akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi kabur. Nilai k yang terbaik dapat dicari dengan menggunakan *cross validation*.

Salah satu metode *cross validation* adalah k-*fold cross validation*. k-*fold cross validation* adalah metode yang paling popular untuk menentukan nilai k dalam k-NN. Dalam teknik ini, data *train* dibagi menjadi sejumlah k-buah partisi. Misalnya terdapat 4000 data dibuat menjadi 5 partisi, sehingga 800 data menjadi data *test* dan 3200 data menjadi data *train*. Kemudian dilakukan sejumlah k-iterasi eksperimen untuk menentukan akurasi dari k yang akan digunakan untuk algoritma k-NN.



Dari metode k-fold, didapatkan nilai k untuk k-NN dengan akurasi terbaik. Nilai k tersebut kemudian akan menjadi nilai k yang terbaik dan akan digunakan dalam menentukan data *test* yang belum terklasifikasi.

Untuk kasus ini, pencarian nilai k dilakukan dengan melakukan percobaan dimulai dengan nilai k=1 hingga k=3200 dengan menggunakan 5-*fold cross validation*. Kemudian *k* yang diambil untuk klasifikasi menggunakan *k*-NN adalah *k* yang memiliki akurasi tertinggi.

Referensi

https://informatikalogi.com/algoritma-k-nn-k-nearest-neighbor/
https://medium.com/@piyut.dyoni/machine-learning-buat-yang-ngerasa-bodo-e37bc5b26d9d
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Output

- Baris kode untuk mencari nilai k

```
if __name__ == '__main__':
    for k_value in range(0,3200,2):
        print(k_value)
        sum_accuracy = 0
        start_val = 0
        finish_val = 800
        while (start_val < 4000):
            accuracy = 0
            # i for data train, j for data test
            for i in range(start_val,finish_val):
                like = df_train.loc[i, "Like"]
                provokasi = df_train.loc[i, "Provokasi"]
                komentar = df_train.loc[i, "Komentar"]
                emosi = df_train.loc[i, "Emosi"]

                euclid_val = []
                for j in range(len(df_train)):
                    if (not (j>=start_val and j<=finish_val)):
                        value = math.sqrt(((like - df_train.loc[j, "Like"]) ** 2) + (
                                          (provokasi - df_train.loc[j, "Provokasi"]) ** 2) + (
                                          (komentar - df_train.loc[j, "Komentar"]) ** 2) + (
                                          (emosi - df_train.loc[j, "Emosi"]) ** 2))
                        euclid_val.append(value)
                euclid_val_sorted = sorted(euclid_val)
                nearest_neighbor = []
                for k in range(k_value):
                    nearest_neighbor.append(df_train.loc[euclid_val.index(euclid_val_sorted[k]), "Hoax"])
                counter = Counter(nearest_neighbor)
                if (counter.most_common()[0][0] == df_train.loc[i, "Hoax"]):
                    accuracy += 1
            sum_accuracy += ((accuracy/800) * 100)
            start_val += 800
            finish_val += 800
        print("Akurasi dengan k-"+str(k_value)+": "+str(sum_accuracy/k_fold))
```

- Output pencarian nilai k

```
Akurasi dengan k-191: 66.075
Akurasi dengan k-193: 66.2
Akurasi dengan k-195: 66.375
Akurasi dengan k-197: 66.325
Akurasi dengan k-199: 66.35
Akurasi dengan k-201: 66.45
Akurasi dengan k-203: 66.225
Akurasi dengan k-205: 66.35
Akurasi dengan k-207: 66.15
Akurasi dengan k-209: 65.95
Akurasi dengan k-211: 65.925
Akurasi dengan k-213: 65.9
Akurasi dengan k-215: 65.8
Akurasi dengan k-217: 65.875
Akurasi dengan k-219: 65.925
Akurasi dengan k-221: 66.025
Akurasi dengan k-223: 66.05
Akurasi dengan k-225: 66.1
Akurasi dengan k-227: 66.1
Akurasi dengan k-229: 66.05
Akurasi dengan k-231: 65.725
Akurasi dengan k-233: 65.725
Akurasi dengan k-235: 65.7
Akurasi dengan k-237: 65.85
Akurasi dengan k-239: 65.675
Akurasi dengan k-241: 65.825
Akurasi dengan k-243: 66.025
Akurasi dengan k-245: 65.9
Akurasi dengan k-247: 66.0
Akurasi dengan k-249: 65.95
Akurasi dengan k-251: 66.15
Akurasi dengan k-253: 65.9
Akurasi dengan k-255: 65.775
Akurasi dengan k-257: 65.775
Akurasi dengan k-259: 66.1
Akurasi dengan k-261: 66.05
Akurasi dengan k-263: 65.9
Akurasi dengan k-265: 65.85
Akurasi dengan k-267: 65.8
```

- Baris kode untuk menjawab sheet “DataTest”

```
df_train = pd.read_excel("dataset_fix.xlsx", "DataTrain")
df_test = pd.read_excel("dataset_fix.xlsx", "DataTest")

if __name__ == '__main__':
    k_value = 201
    accuracy = 0
    result = []
    print(df_test.head())
    # i for data train, j for data test
    for i in range(len(df_test)):
        euclid_val = []
        like = df_test.loc[i, "Like"]
        provokasi = df_test.loc[i, "Provokasi"]
        komentar = df_test.loc[i, "Komentar"]
        emosi = df_test.loc[i, "Emosi"]

        for j in range(len(df_train)):
            value = math.sqrt(((like - df_train.loc[j, "Like"]) ** 2) + (
                    (provokasi - df_train.loc[j, "Provokasi"]) ** 2) + (
                                    (komentar - df_train.loc[j, "Komentar"]) ** 2) + (
                                    (emosi - df_train.loc[j, "Emosi"]) ** 2))
            euclid_val.append(value)
        euclid_val_sorted = sorted(euclid_val)
        nearest_neighbours = []
        for j in range(k_value):
            nearest_neighbours.append(df_train.loc[euclid_val.index(euclid_val_sorted[j]), "Hoax"])
        counter = Counter(nearest_neighbours)
        df_test.loc[i, "Hoax"] = counter.most_common()[0][0]

    print(df_test.head())
    writer = pd.ExcelWriter("output.xlsx", engine="xlsxwriter")
    df_test.to_excel(writer, index=False, sheet_name="Result")
    writer.save()
```

NB: Output dapat dilihat di file output.xlsx