Analisis dan Implementation Algoritma *K-Nearest Neighbor* (KNN) untuk biner classification

Fikhri Masri (1301164662)

Program Study S1 Teknik Informatika Fakultas Informatika, Telkom University

I. Analisis Masalah

K-Nearest Neighbor

K-Nearest Neighbor (KNN) adalah metode melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Metode ini bertujuan untuk mengklasifikasikan objek baru berdasarkan atribut dan training sample. Diberikan suatu titik query, selanjutnya akan ditemukan sejumlah K objek atau titik training yang paling dekat dengan titik query. Nilai prediksi dari query akan ditentukan berdasarkan klasifikasi tetanggaan (Tri, 2010).

Sebelum melakukan perhitungan dengan metode *K-Nearest Neighbor*, terlebih dahulu harus menentukan data latih dan data uji. Kemudian akan dilakukan proses perhitungan untuk mencari jarak menggunakan Euclidean. Setelah itu, akan dilakukan tahapan perhitungan dengan metode KNN.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Gambar I.1

Pada kasus disini *K-Nearest Neighbor* (KNN) digunakan untuk biner classification. Data train yang digunakan memiliki 4000 baris, 4 kolom untuk fitur, dan 1 kolom untuk classification. Datatrain dibagi menjadi 80% untuk train yang berjumlah = 3200, dan validation 20% yang berjumlah = 800.

II. Implementasi

Untuk mengimplementasikan algoritma *K-Nearest Neighbor* (KNN) disini menggunakan Bahasa pemrograman R, dengan IDE R Studio.

```
train <- read.csv("E:/Matkul kuliah/Semester 6/AI/Tugas 2 AI/Dataset/DataTrain Tugas 2 AI.csv", sep = ",")
test <- read.csv("E:/Matkul kuliah/Semester 6/AI/Tugas 2 AI/Dataset/DataTest_Tugas_2_AI.csv", sep = ",")
```

Gambar II.1

Gambar diatas merupakan pemanggilan data test dan data train yang berformat csv untuk digunakan sebagai data utama pada program.

```
#Fungsi dari distance untuk menghitung jarak antar attribut
distance <- function(x1, x2){
  temp = 0
  for(i in c(1:(length(x1)-1) ))
  {
    temp = temp + (x1[[i]]-x2[[i]])^2
  }
  temp = sqrt(temp)
  return(temp)
}
```

Gambar II.2

Pada gambar II.2 disamping merupakan pendeklarasian dari suatu fungsi untuk menghitung jarak antar titik dengan menggunakan *Euclidean Distance*.

```r
#Fungsi KNN
K.Nearest.Neighbor <- function(valid, train, k){
  # Inisialisasi vector untuk menyimpan prediksi kelas
  rowclass <- c()
  for(j in 1:nrow(valid)){
    # Gabungkan label asli dari datatest dengan distance menjadi satu dataframe
    rowvalue <- data.frame("dist" = distance(valid[j,], train), "kelas" = train$kelas)

    # Urutkan dataframe berdasarkan jarak secara ascending
    rowvalue <- rowvalue[order(rowvalue$dist),]

    # Ambil hasil urut dari 1 sampai k dan hitung frekuensi kemunculan label yang ada
    predict <- as.data.frame(table(rowvalue[1:k,]$kelas))
    # Urutkan dataframe label berdasarkan frekuensi secara DESCENDING
    predict <- predict[order(predict$Freq, decreasing = TRUE),]

    # Append ke vector label dengan frekuensi maksimal
    rowclass <- c(rowclass, as.numeric(as.character(predict$Var1[1])))
  }

  # Hitung akurasi antara vector prediksi label dengan label asli
  acc <- rowclass == valid$kelas
  return((length(acc[acc == TRUE]) / length(acc)) * 100)
}
```

Gambar II.3

```r
# Pencarian K terbaik
# KFold = 5
K <-60
fold <- 5
split.data <- nrow(train) / fold


result <- c()
for(i in 1:fold){
  # Ambil data validation dan train baru untuk validasi
  test.validation <- train[c((1 + split.data*(i-1)) : (split.data*i)),]
  train.validation <- train[-c((1 + split.data*(i-1)) : (split.data*i)),]

  # inisialisasi vector untuk Tunning Hyperparameter K
  accuracy.per.test <- c()
  for(k in 1:K){
    # Menghitung akurasi tiap K
    accuracy.per.test <- c(accuracy.per.test, K.Nearest.Neighbor(test.validation, train.validation, k))
  }

  result <- rbind(result, accuracy.per.test)
}
train.validation
test.validation
colnames(result) <- 1:K

# Menghitung rata - rata dari semua lipatan setiap K
result <- colSums(result) / nrow(result)
result

# Ambil K dengan akurasi terbaik
K.Terbaik <- as.numeric(names(result[result == max(result)]))
K.Terbaik
```

Gambar II.4

Pada gambar II.3 merupakan pendeklarasian dari suatu fungsi untuk KNN yang memanggil fungsi dari distance untuk mencari tetangga terdekat dari datatrain dan datavalidation.

Pada gambar II.4 merupakan pendeklarasian dari suatu fungsi untuk mencari hasil dari sebuah K terbaik yang di looping dari 1 sampai 60, dan membagi sebuah trainvalidation menjadi 3200, testvalidation menjadi 800 data.

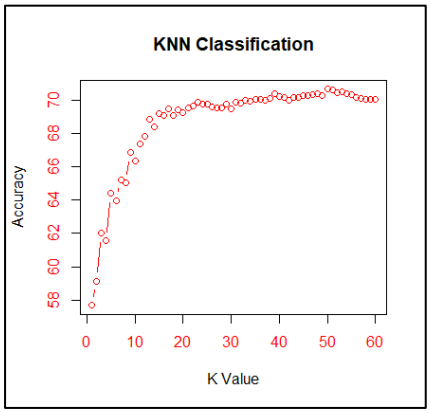
```r
plot(result, type = "b", col = "red", col.axis="red", main  = "KNN Classification", xlab = "K Value", ylab = "Accuracy")

rowclass <- c()
for(j in 1:nrow(test)){
  rowvalue <- data.frame("dist" = distance(test[j,], train), "kelas" = train$kelas)
  rowvalue <- rowvalue[order(rowvalue$dist),]

  class.predict <- as.data.frame(table(rowvalue[1:optimum.K,]$kelas))
  class.predict <- class.predict[order(class.predict$Freq, decreasing = TRUE),]

  rowclass <- c(rowclass, as.numeric(as.character(class.predict$Var1[1])))
}

rowclass
write.table(rowclass, file = "E:/Matkul kuliah/Semester 6/AI/Tugas 2 AI/Predicttugas2.csv", sep = ",", row.names = FALSE, col.names = "kelas")
```

Gambar II.5

Pada gambar II.5 menampilkan plot untuk melihat hasil K beserta akurasi dan predict data test. Hasil dari predict data test di tampung di dalam vector kemudian di ubah ke csv dan disimpan dalam directory local.

III. Hasil

Pada gambar dibawah merupakan hasil dari runningan program mencari K terbaik mencari dari 1 sampai 60 dan medapatkan hasil runningan K terbaik yaitu 50 dengan akurasi 70.650%.



Gambar III.1

```
      1       2       3       4       5       6       7       8       9      10      11      12      13      14      15      16
57.675 59.150 62.000 61.550 64.400 63.975 65.225 65.025 66.850 66.375 67.375 67.825 68.850 68.400 69.175 69.100
     17      18      19      20      21      22      23      24      25      26      27      28      29      30      31      32
69.475 69.075 69.425 69.250 69.550 69.625 69.875 69.775 69.750 69.600 69.550 69.550 69.775 69.475 69.875 69.825
     33      34      35      36      37      38      39      40      41      42      43      44      45      46      47      48
69.975 69.950 70.025 70.025 69.975 70.075 70.400 70.200 70.150 69.975 70.150 70.175 70.250 70.250 70.350 70.400
     49      50      51      52      53      54      55      56      57      58      59      60
70.275 70.650 70.625 70.425 70.475 70.375 70.300 70.150 70.075 70.025 70.050 70.025
>
> # Ambil K dengan akurasi terbaik
> K.Terbaik <- as.numeric(names(result[result == max(result)]))
> K.Terbaik
[1] 50
```

Gambar III.2