Farhan Anas
1301183427
IF-42-01

Perhitungan Ukuran Jarak

Perhitungan jarak atau distance menggunakan rumus Euclidean Distance yaitu :

L2 (Euclidean) distance

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

Maka dibuat dalam bentuk fungsi eucDistance sebagai berikut :

```
def eucDistance(x1, x2):
  dist = 0.0
  for i in range(len(x1)-1):
    dist += (x1[i] - x2[i])**2
  return sqrt(dist)
```

Fungsi untuk menghitung jarak antara 1 baris data train dan 1 baris data test dimana data train dan data test digunakan sebagai inputnya dan menggunakan loop untuk menghitung masing-masing kolom dari kedua baris data tersebut

Prapemrosesan Data

Prapemrosesan data menggunakan 2 buah fungsi yaitu :

- Fungsi replaceZero

```
def replaceZero(dataset):
  colZero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
  for column in colZero:
    dataset[column] = dataset[column].replace(0, np.NaN)
    mean = int(dataset[column].mean(skipna=True))
    dataset[column] = dataset[column].replace(np.NaN, mean)
  return dataset
```

  Fungsi ini digunakan untuk mengubah kolom-kolom di dataset yang memiliki nilai 0 menjadi nilai rata-rata yang terdapat pada kolom tersebut. Kolom-kolom yang diubah nilai 0 nya adalah kolom Glucose, BloodPressure, SkinThickness, Insulin dan BMI. Pertama-tama nilai 0 diganti sementara menjadi nilai null agar tidak mempengaruhi perhitungan rata-rata nilai kolom lalu selanjutnya nilai null tersebut diganti menjadi nilai rata-rata yang sudah dihitung, hal ini diloop sebanyak jumlah kolom yang memiliki nilai 0

- Fungsi normalize

```
def normalize(dataset):
  dataset = (dataset - dataset.min()) / (dataset.max() - dataset.min())
  return dataset
```

  Fungsi ini digunakan untuk mengubah nilai-nilai yang ada pada dataset menjadi skala 0 sampai 1 dengan cara data pada masing-masing kolom dikurangi dengan data minimum pada kolom tersebut lalu dibagi dengan selisih dari data maksimum dan data minimum

Klasifikasi KNN

Farhan Anas
1301183427
IF-42-01
Klasifikasi kNN menggunakan 2 buah fungsi yaitu :

- Fungsi nearestNeighbors

```
def nearestNeighbors(train, test, k):
  listDistances = list()
  for train_row in train:
    dist = eucDistance(test, train_row)
    listDistances.append((train_row, dist))
  listDistances.sort(key=lambda tup: tup[1])
  neighbors = list()
  for i in range(k):
    neighbors.append(listDistances[i][0])
  return neighbors
```

  Fungsi ini digunakan untuk mencari sejumlah k neighbors atau tetangga yang dalam hal ini adalah baris pada data train yang memiliki distance atau jarak paling kecil, input dari fungsi ini sendiri adalah data train, data test dan jumlah k. Pertama-tama dibuat list kosong bernama listDistances yang nantinya akan berisi jarak antara 1 data test dengan semua data train, kemudian dilakukan loop sebanyak jumlah data train dan menggunakan fungsi yang sebelumnya sudah didefinisikan untuk menghitung jarak dan memasukkan hasil dari fungsi tersebut ke dalam list yang telah dibuat sebelumnya. Setelah mendapatkan list yang berisi jarak, dilakukan sort ascending untuk list tersebut dengan key untuk sort adalah kolom jarak tersebut. Terakhir dibuat list baru yang akan berisi sebanyak K tetangga dengan jarak terkecil berdasarkan listDistances yang sudah disort sebelumnya

- Fungsi predictClass

```
def predictClass(train, test, k):
  neighbors = nearestNeighbors(train, test, k)
  output_values = [row[-1] for row in neighbors]
  prediction = max(set(output_values), key=output_values.count)
  return prediction
```

  Fungsi ini digunakan untuk memprediksi outcome dari sebuah data test dengan cara mencari outcome yang paling dominan dari tetangga data test tersebut. Pertama-tama dicari sejumlah K tetangga dengan jarak terkecil menggunakan fungsi nearestNeighbors yang sudah didefinisikan sebelumnya, lalu mencari nilai outcome dari masing-masing tetangga terdekat untuk selanjutnya dicari outcome yang paling dominan dan mengembalikan outcome tersebut

Pemilihan Nilai K Terbaik

Pemilihan Nilai K Terbaik adalah dengan membuat dictionary kosong terlebih dahulu yang nantinya akan digunakan untuk menampung nilai K

```
dict_K = {}
```

Setelah dilakukan perhitungan rata-rata akurasi, tiap-tiap nilai rata-rata ini akan dimasukkan kedalam dictionary dict_K dan akan diberikan index sesuai nilai K nya

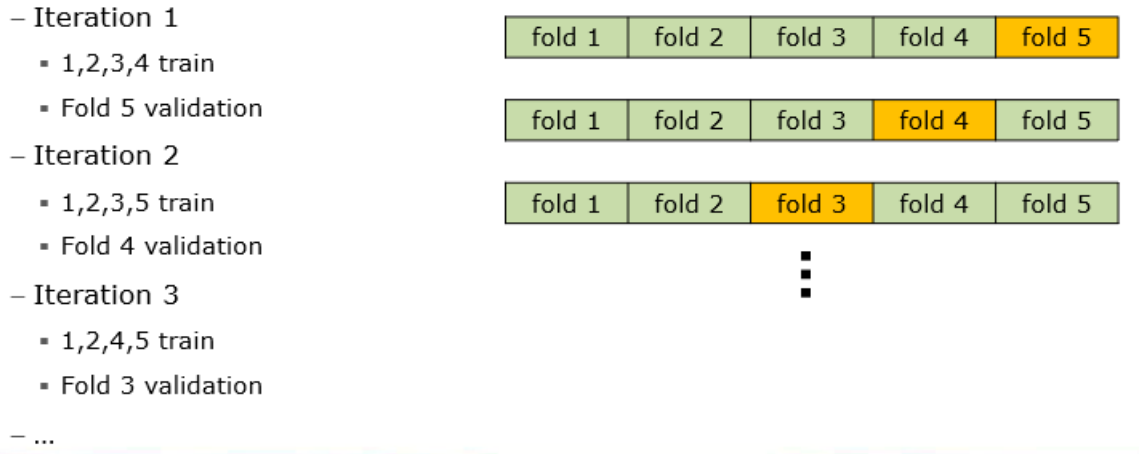
```
dict_K[K] = avg_accuracy
```

Farhan Anas
1301183427
IF-42-01
Setelah dilakukan loop untuk semua nilai K yang ingin dicari rata-rata akurasinya, dibuat sebuah variable best_K untuk mencari nilai K terbaik dengan cara melakukan fungsi max terhadap dictionary dict_K yang sudah terisi

```
best_K = max(dict_K, key = lambda k : dict_K.get(k))
```

Perhitungan Rata-Rata Akurasi kNN Menggunakan 5-Fold Cross-Validation

Perhitungan Rata-Rata adalah dengan cara dataset yang ada displit atau dibagi terlebih dahulu menjadi 5 bagian karena menggunakan 5-Fold Cross-Validation dengan ilustrasi sebagai berikut :



Setelah dibagi menjadi 5, dibuatlah prosedur sebagai berikut :

```
for K in range(1,51):
  total_accuracy = []
  for i in range(len(semua_fold)):
    list_predict = []
    for j in range(len(semua_fold[i][1])):
      results = predictClass(semua_fold[i][0], semua_fold[i][1][j], K)
      list_predict.append(results)
    total_accuracy.append(accuracy(semua_fold[i][2], list_predict))
  avg_accuracy = sum(total_accuracy) / len(semua_fold)
```

Pertama-tama melakukan loop sebanyak K yang ingin dicari yang dimana dalam contoh tersebut dicari dari K = 1 sampai dengan 50, K adalah banyaknya tetangga terdekat yang ingin dicari ketika melakukan testing. Dalam setiap K yang dicari, dibuat sebuah list total_accuracy yang nantinya akan menampung accuracy dari 5 iterasi yang nantinya akan dijalankan lalu selanjutnya dilakukan loop sebanyak jumlah fold (sebanyak 5 kali) dan dibuat list list_predict untuk menampung outcome dari masing-masing data test yang telah diprediksi dan nantinya akan digunakan untuk dibandingkan dengan outcome yang ada pada data aslinya. Maka dilakukan loop sebanyak data test yang dimana akan dilakukan prediksi outcome untuk setiap data test yang ada dan hasilnya akan dimasukkan ke list_predict yang sebelumnya sudah dibuat. Setiap selesai sebuah fold diprediksi, maka akan dihitung akurasinya dan dimasukkan ke total_accuracy dan setiap selesai sebuah nilai K diproses, maka akan dihitung rata-rata akurasi untuk nilai K tersebut sehingga didapat rata-rata akurasi dari masing-masing nilai K

Farhan Anas
1301183427
IF-42-01

Parameter KNN paling Optimum

Terdapat 2 hyperparameters untuk KNN yaitu rumus jarak yang digunakan dan nilai K terbaik seperti pada materi berikut :

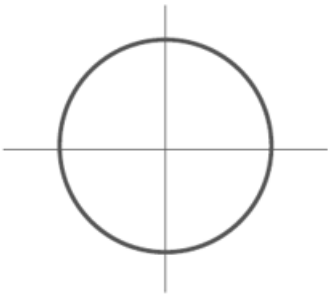
k-Nearest Neighbors - Hyperparameters

- What is the best **distance** to use?
- What is the best value of **k** to use?

Maka rumus jarak yang digunakan adalah Euclidean Distance :

L2 (Euclidean) distance

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



Dan setelah melakukan pengujian dari nilai K = 1 sampai dengan K = 50 menggunakan rumus jarak tersebut, diperoleh nilai K terbaik yaitu 23 dengan accuracy 76,62% dibuktikan dengan screenshot hasil running program sebagai berikut :

```
best K :  23
accuracy :  76.62337662337663
```

Link Video Penjelasan Algoritma :

https://drive.google.com/file/d/15R5sSvyF2IgzHBp0_kbNurHViT_tDe8L/view?usp=sharing