Media Nusa Creative
Anggota IKAPI (162/JTI/2015)
Bukit Cemara Tidar H5 No. 34 Malang
Telp : 0812 3334 0088
Email : mncpublishing.layout@gmail.com
Website : www.mncpublishing.com

CONTACT US

ISBN 978-602-462-907-6

MACHINE LEARNING METODE k-NEAREST NEIGHBORS KLASIFIKASI ANGKA BAHASA ISYARAT

Romy Budhi Widodo

MACHINE LEARNING

METODE k-NEAREST NEIGHBORS KLASIFIKASI ANGKA BAHASA ISYARAT

Romy Budhi Widodo

MACHINE LEARNING METODE *k-NEAREST NEIGHBORS*-KLASIFIKASI ANGKA BAHASA ISYARAT

Oleh:
Romy Budhi Widodo

MNC PUBLISHING
FUTURE BOOKS WITH PASSION
Media Nusa Creative

MACHINE LEARNING
METODE *k-NEAREST NEIGHBORS*- KLASIFIKASI ANGKA BAHASA ISYARAT

Penulis:
Romy Budhi Widodo

Desain Cover & Penata Isi
Tim Media Nusa Creative

Cetakan I, Maret 2022

Diterbitkan oleh :

Media Nusa Creative
Anggota IKAPI (162/JTI/2015)
Bukit Cemara Tidar H5 No. 34, Malang
Telp. : 0812.3334.0088
E-mail : mncpublishing.layout@gmail.com
Website : www.mncpublishing.com

ISBN 978-602-462-907-6

PENGANTAR PENULIS

Perkembangan permasalahan di dunia nyata semakin dekat dengan keberadaan ilmu pengetahuan itu sendiri. Model teori prediksi, desain algoritma, dan ekstrapolasi untuk peramalan kejadian terus berkembang. Saat ini pekerjaan di bidang machine learning memberikan suatu harapan keprofesionalan ilmu komputasi dalam membantu bidang kehidupan manusia di hampir semua aspek. Di aspek teknikal, perkembangan Python sejak diluncurkan oleh Guido van Rossum tahun 1991, banyak dipakai di bidang machine learning dan data science. Python memungkinkan konversi ke C dan C++, hal ini memberi harapan pengembang perangkat lunak menjalankan kode-nya pada graphic processing unit.

Buku ini menggunakan teori machine learning khususnya metode k-Nearest Neighbors (kNN) dan bahasa Python; untuk menyelesaikan klasifikasi bahasa isyarat. Data real yang digunakan diambil dari sarung tangan yang sudah dilengkapi dengan sensor tekuk, pada Pusat Studi Human-Machine Interaction (HMI), Program Studi Teknik Informatika, Universitas Ma Chung. Bagi Pembaca yang tertarik mempelajari dan berkolaborasi dapat mengunjungi pusat studi tersebut.

"Tiada gading yang tak retak", demikian juga dalam penulisan buku ini, oleh sebab itu masukan dari pembaca sangatlah bermanfaat dan dapat dilayangkan kepada penulis melalui email: romy.budhi@machung.ac.id

Malang, Maret 2022
Penulis

Romy Budhi Widodo

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

Teknologi Alat Bantu

Penyandang disabilitas pada usia 24 sampai 59 bulan di Indonesia cukup banyak. Di tahun 2018, jenis disabilitas tunawicara dilaporkan 0,15% dan tunarungu 0,11% dari populasi [1]. Menurut data disabilitas pada [2], data prevalensi tunarungu dan tunawicara berdasarkan sensus 2012 di Indonesia adalah 36.959 jiwa; dimana nilai tersebut mencapai 7,87% keseluruhan disabilitas di Indonesia.

Komunikasi antara penyandang dan masyarakat awam sering terbentur karena masyarakat awam tidak memiliki keterampilan berbahasa isyarat. Salah satu anggota GERKATIN (Gerakan untuk Kesejahteraan Tunarungu Indonesia) di Malang menyatakan tanggapan positif jika ada teknologi tepat guna yang dapat menerjemahkan bahasa isyarat menjadi Bahasa Indonesia.

Melihat kondisi bahwa sulitnya masyarakat awam berkomunikasi dengan penyandang tunarungu dan tunawicara, maka diperlukan sebuah sistem untuk menerjemahkan gerakan tangan menjadi bahasa sehari-hari melalui audio atau teks. Memandang hal tersebut, sarung tangan adalah kandidat yang cocok sebab posisinya berada di tangan pengguna bahasa isyarat. Diharapkan dengan teknologi tepat guna tersebut antara penyandang dan masyarakat awam dapat berkomunikasi.

Pada penelitian ini sistem yang ditargetkan menggunakan sarung tangan. Peneliti lain menggunakan kamera dan Leap Motion Controller sebagai alat atau instrumen dalam penelitiannya. Sarung tangan memiliki kelebihan karena merupakan benda yang dapat dipakai dan tidak terikat di depan sebuah komputer. Penerjemah bahasa isyarat menggunakan kamera membutuhkan jarak optimal dan kuantitas lux pencahayaan yang stabil; menilik faktor kepraktisannya dalam penggunaan sehari-hari maka kamera bukan pilihan dalam penelitian ini. Kekurangan penerjemah bahasa isyarat dengan instrumen sarung tangan yaitu tidak dapat menangkap dan mengenali ekspresi wajah, sedangkan dengan menggunakan kamera hal tersebut dapat dilakukan. Ekspresi wajah atau mimik saat melakukan bahasa isyarat dapat memperkuat makna setiap kata dalam komunikasi bahasa isyarat. Peneliti lain yang menggunakan Leap Motion Controller [3] perlu meletakkan sensor tersebut di atas meja atau area yang sesuai dan selaras dengan area gerakan tangan, konsekuensinya terdapat area buta yang menyebabkan sensor tidak dapat membaca gerakan bahasa

isyarat. Menilik hal tersebut pilihan sarung tangan sebagai media untuk menangkap gestur tangan diharapkan menjadi pilihan yang tepat.

Penghitungan jumlah prevalensi disabilitas adalah persoalan yang kompleks. Banyak faktor yang mempengaruhi pendekatan perhitungan jumlah atau prevalensi penyandang disabilitas tersebut, diantaranya pengaruh dari tujuan/pemanfaatan datanya, definisi dan konsep disabilitas yang digunakan, aspek disabilitas itu sendiri (misalnya: keterbatasan partisipasi, keterbatasan aktifitas, faktor lingkungan, kondisi kesehatan yang terkait) dan sumber datanya. *World Report on Disability* (WHO) tahun 2011, menggunakan hasil *World Health Survey* dan *Global Burden of Disease* dalam mengestimasi prevalensi disabilitas. Dalam laporan tersebut dikatakan bahwa semestinya data estimasi prevalensi disabilitas sebaiknya tidak dipandang sebagai angka definitif, namun sebaiknya sebagai ketersediaan data dan refleksi pengetahuan terkini.

Berikut ini ingin disajikan penelitian terdahulu berkaitan dengan *sign language* (bahasa isyarat). Penggunaan sensor flex, sensor tekanan, dan sensor inertial dilakukan oleh [4] dalam penelitiannya untuk menciptakan *wearable device* bagi penyandang tunawicara. Penggunaan bentuk sarung tangan (*handglove*) sudah banyak dilakukan oleh peneliti lain untuk menangkap sinyal dari gerakan jari atau tangan diantaranya juga dilakukan oleh [5][6][7][8]. Kebanyakan studi yang telah ada mendeskripsikan penggunaan berbagai sensor namun uji performansinya masih belum banyak dilakukan, utamanya pada penyandang tunawicara dan tunarungu. Berbagai jenis sensor digunakan namun metode klasifikasi yang digunakan terbuka luas untuk diteliti, untuk menentukan metode mana yang lebih efisien dan efektif jika nantinya diterapkan ke dalam *board* terpadu untuk menghasilkan produk di kemudian hari.

Buku ini membahas hasil penelitian dan ingin berkontribusi pada pengembangan *assistive technology*, di mana akan didasari oleh akuisisi data sarung tangan terlebih dahulu. Tujuan yang ingin dicapai adalah menentukan jenis sensor, metode pengambilan data gerakan, dan klasifikasi yang sesuai untuk aplikasi *real time.* Adapun bahasa isyarat standar di Indonesia adalah SIBI (Sistem Isyarat Bahasa Indonesia) yang akan digunakan dalam buku ini.

Tidak banyak masyarakat awam mengenal bahasa isyarat. Di samping itu prevalensi tunarungu dan tunawicara cukup banyak sesuai uraian di atas. Dengan demikian rumusan masalah yang bisa ditarik

adalah bagaimana supaya penyandang tunarungu dan tunawicara dapat berkomunikasi dengan masyarakat awam dengan mudah. Arti kata "mudah" lebih ke arah teknis dimana terdapat suatu piranti yang otomatis dapat menerjemahkan bahasa isyarat tangan menjadi teks.

Buku ini sejalan dengan perkembangan *assistive technology* yang sudah menggunakan berbagai metode kecerdasan buatan untuk mengolah hasil akuisisi data. Buku ini diharapkan mendasari langkah-langkah pengerjaan machine learning, khususnya metode *kNN* bagi pemula, praktisi, dan *machine learning engineer.*

BAB 2

Apa Itu Bahasa Isyarat dan Akuisisi Data?

2.1. SIBI dan BISINDO

Di masyarakat tunarungu dan tunawicara Indonesia terdapat dua bahasa isyarat Indonesia yang telah berkembang dan dipakai sehari-hari, yaitu Sistem Bahasa Isyarat Indonesia (SIBI) dan Bahasa Isyarat Indonesia (BISINDO).

Pemerintah telah membakukan Kamus Sistem Isyarat Bahasa Indonesia (SIBI) dan membakukan Sistem Bahasa Isyarat Indonesia (SIBI) pada tahun 1994 [9]. SIBI sendiri digunakan di sebagian besar Sekolah Luar Biasa (SLB) B, sekolah khusus untuk tunarungu. SIBI mengadopsi ASL (American Sign Language). Beberapa ciri khas SIBI seperti terdapat pada literatur adalah sebagai berikut: 1) Menggunakan imbuhan Bahasa Indonesia dalam kalimat seperti misalnya ber-, me-, pe-, -an, -nya, ke- dan awalan di-; 2) Kurang menggunakan bahasa isyarat alami, seperti misalnya ekspresi, gestur, kontak mata, posisi tubuh, dan gerakan tangan; 3) Konsep pada SIBI menekankan pada struktur kalimat [10].

Sedangkan BISINDO, merupakan bahasa yang juga diakui secara formal dalam penjelasan Undang-Undang Republik Indonesia Nomor 8 tahun 2016 [11]. Di Indonesia ada komunitas GERKATIN (Gerakan untuk Kesejahteraan Tunarungu Indonesia) yang menggunakan BISINDO. Dalam keseharian, BISINDO diminati tunarungu sebab seolah-olah seperti bahasa ibu bagi penyandang tunarungu. Namun BISINDO dapat mengalami modifikasi di wilayah lain. Dalam penelitian yang menggunakan seratus responden tunarungu di Jawa-Bali tahun 2015, disimpulkan bahwa dalam keseharian BISINDO digunakan oleh 91% responden [10]. Beberapa ciri khas BISINDO yang dirangkum dari hasil penelitian tersebut adalah sebagai berikut: 1) BISINDO menyesuaikan pemahaman bahasa tunarungu dari berbagai latarbelakang tunarungu tanpa memberikan imbuhan Bahasa Indonesia; 2) Banyak menekankan pada ekspresi, gerakan tangan, dan kontak mata; 3) Penyampaian suatu kalimat/kata lebih singkat dari SIBI.

Demikianlah sekilas dua bahasa isyarat yang digunakan di Indonesia. Penelitian ini menggunakan referensi SIBI yang banyak digunakan oleh penerjemah dan sekolah luar biasa. Harapan ke depan aplikasi yang dibuat tentunya dapat dikembangkan juga untuk BISINDO. Gambar 2.1 menunjukkan bahasa isyarat 1-20 menggunakan SIBI [12]. Adapun gerakannya secara visual dapat dilihat pada tutorial Youtube: https://www.youtube.com/watch?v=h681dhezQyw&t=11s, menit ke-1 detik ke-45 hingga menit ke-3 detik ke-10.

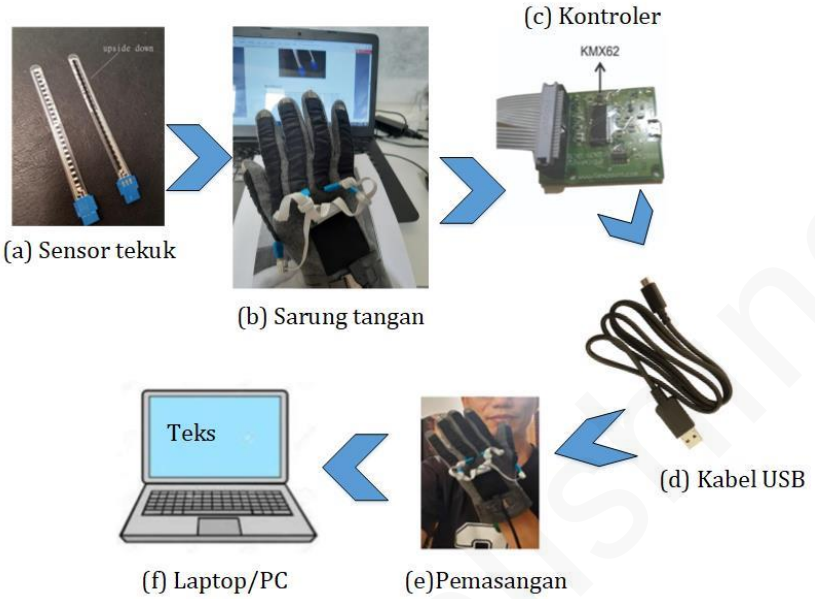
Gambar 2.1 Angka 1-20 dalam SIBI

2.2. Akuisisi Data

Proses akuisisi data terdiri atas beberapa peralatan yang diperlukan, yaitu:

a. Sensor tekuk (*bend sensor*)
b. Sarung tangan
c. Kontroler
d. Software akuisisi data

Gambar 2.2 mengilustrasikan blok diagram dari peralatan-peralatan tersebut.



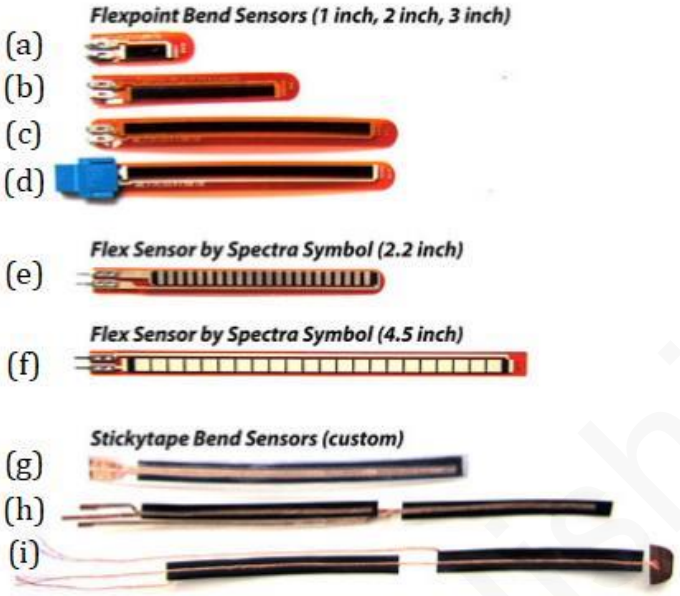
Gambar 2.2 Diagram pengambilan data gestur tangan

Berikut penjelasan keempat bagian utama dari Gambar 2.2 yaitu Sensor tekuk (*bend sensor*), Sarung tangan, Kontroler, dan Software akuisisi data di laptop/PC.

2.2.1. Sensor Tekuk (*bend sensor*)

Sensor tekuk ada berbagai merek, yang umum digunakan adalah flexpoint (https://flexpoint.com/bend-sensor/). Sensor tekuk bekerja berdasarkan perubahan konduktifitas bahan saat sensor tersebut ditekuk. Untuk mengamati seberapa besar perubahan tekukan umumnya dapat digunakan ohmmeter pada kaki-kaki sensor. Perubahan konduktifitas pada bahan pelapis sensor saat ditekuk, ditangkap oleh rangkaian elektronik atau kontroler dan merubahnya menjadi tegangan. Jika sudah menjadi tegangan maka akan diubah menjadi besaran digital yang siap diolah komputer.

Gambar 2.3 menunjukkan ilustrasi sensor tekuk dengan berbagai merek, ukuran, dan jenis konektor.



Gambar 2.3 Ilustrasi sensor tekuk (sumber: Internet)

Tabel 2.1 menunjukkan karakteristik sensor tekuk dan penjelasan singkat.

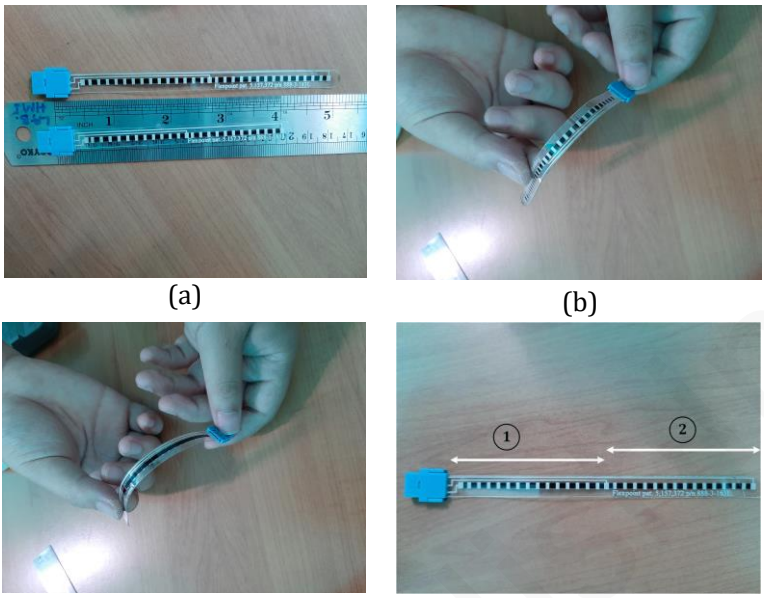
Tabel 2.1. Karakteristik Sensor Tekuk

Karakteristik		Keterangan
Jenis sensor	a. Standard	Sensor hanya bisa ditekuk ke satu sisi. Cocok untuk aplikasi yang ingin mendeteksi satu arah tekukan saja.
	b. Bi-Directional	Sensor bisa ditekuk ke atas atau ke bawah (dua sisi). Nilai hambatan (ohm) berbeda untuk dua arah tersebut.
Laminasi	a. Tidak ada	Tidak ada laminasi bahan.
	b. Polyester	Dilaminasi bahan Polyester.

Karakteristik		Keterangan
Jenis konektor	a. Solder tab	Ujung sensor untuk keperluan disolder ke piranti lain (seperti pada Gambar 2.3 a, b, dan c)
	b. Konektor female	Ada soket diujung sensor, memudahkan bongkar dan pasang dengan piranti lain (seperti pada Gambar 2.3 d)
Panjang	1” sampai 5”	Panjang sensor 1 inchi sampai 5 inchi, sesuai penggunaan dan pemesanan.

Sedangkan pada penelitian ini digunakan sensor tekuk dengan spesifikasi:

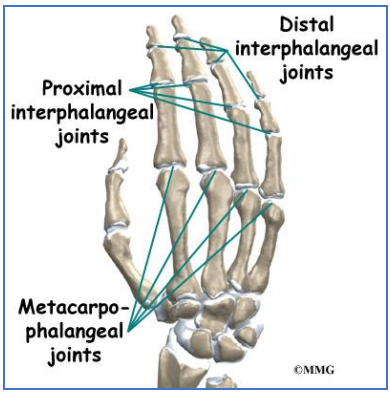
- Lima buah Flexpoint bend sensor dengan dua dimensi yaitu panjang 5” untuk jari telunjuk, jari tengah, dan jari manis; dan panjang 4” untuk ibu jari dan jari kelingking.
- Jenis sensor adalah standard, namun dirancang khusus. Satu sensor terdiri atas dua bagian yaitu atas dan bawah. Karakteristik sensor dan posisi bending pada Gambar 2.4.



Gambar 2.4 Karakteristik sensor yang digunakan: a) Panjang sensor 4" dan 5"; b) Posisi bending normal; c) Posisi bending terbalik; d) Dua bagian sensor, yaitu atas dan bawah (sumber: Pengujian)

Pada Gambar 2.4b) tekukan/bending satu arah menghasilkan nilai hambatan semakin besar seiring dengan bertambahnya sudut tekukan. Sedangkan Gambar 2.4c) merupakan tekukan yang tidak dapat berpengaruh terhadap nilai hambatan sebab sensor berjenis standard, artinya hanya dapat mengenali tekukan satu arah.

Di Gambar 2.4d) terlihat ada dua bagian sensor atas dan bawah. Sensor bagian atas (diberi nomor 2) mendeteksi tekukan jari dari *Proximal interphalangeal (PIP) joints* ke ujung jari. Sedangkan sensor bagian bawah (diberi nomor 1) mendeteksi tekukan jari dari *Metacarpo-phalangeal (MCP) joints* ke *Proximal interphalangeal (PIP) joints*. Istilah joints ini mengacu pada Gambar 2.5.



(a)

(b)

Gambar 2.5 a) Sendi MCP (*metacarpo-phalangeal*) dan sendi PIP (*proximal interphalangeal*) (sumber: Internet); b) Sarung tangan dan sensor dalam penelitian (sumber: pengujian)

Beberapa pertanyaan sering muncul berkaitan dengan sensor ini (FAQ), diantaranya adalah sebagai berikut:

- Berapa suhu kerja Flexpoint bend sensor?
  Secara normal sensor dapat bekerja pada suhu 200^{o}C sampai -60^{o}C.
- Bagaimana sensor dibaca?
  Secara dasar sensor ini seperti variabel resistor, umumnya jika sensor hanya mengukur satu bagian tekukan, maka ada 2 pin yang diukur. Sensor ini umumnya diolah datanya dengan rangkaian elektronik pembagi tegangan (*voltage divider*), besar tegangan yang berubah-ubah akibat tekukan selanjutnya diumpankan ke ADC (*Analog to Digital Converter*). Hasil kuantisasi tegangan ke besaran digital oleh ADC akan diolah oleh mikroprosesor atau komputer. Cara kedua adalah mengalirkan arus ke sensor, kemudian membaca tegangan pada pin sensor, untuk cara kedua ini tidak diperlukan *voltage divider*.
- Berapa besar tegangan yang digunakan untuk sensor?
  Hal tersebut tergantung dari rangkaian elektronik yang akan digunakan. Disediakan tiga jenis desain tegangan, yaitu: 12V, 5V dan 3 VDC.
- Jenis material apa yang digunakan untuk print sensor ini?
  Saat ini Bend Sensor diprint pada material Polyester dan Polyimide (Kapton).

- Jenis gerakan apa yang menimbulkan perubahan resistansi pada sensor?
  Jenis gerakan yang mempengaruhi nilai resistansi sensor adalah radius tekukan dan defleksi sudut.
- Apakah kelembaban mempengaruhi sensor?
  Ya, kelembaban mempengaruhi sensor ini.
- Seberapa besar sensor dapat ditekuk?
  Ada batasan maksimal sensor bisa ditekuk, melebihi batas tersebut sensor bisa rusak. Melipat sensor tidak diperkenankan.

2.2.2. Sarung Tangan

Sarung tangan terbuat dari bahan kain atau wool. Di setiap ruas jari diberi kantong jahitan. Kegunaan kantong ini untuk menyisipkan sensor satu per satu ke setiap jari. Beberapa panduan pembuatan sarung tangan adalah sebagai berikut: [13]

- Saat sensor ditekuk akan membutuhkan ruang gerak yang lebih besar. Sehingga kantong jahitan perlu lebih longgar supaya tidak menimbulkan *wrinkle* (kerutan). Kerutan dapat mengakibatkan kesalahan pembacaan data.
- Jari kelingking dan ibu jari menggunakan sensor dengan panjang 4” dan jari yang lain menggunakan sensor 5”.

Gambar 2.6 menunjukkan cara pemasangan sensor pada sarung tangan (*handglove*).

(a) (b)

Gambar 2.6 a) Bentuk pemasangan sensor pada Glove; b) Kerutan (*wrinkle*) yang harus dihindari saat perubahan posisi tangan misalnya dari menggenggam ke lurus [13].

2.2.3. Kontroler

Kontroler merupakan board elektronik seperti terlihat pada Gambar 2.2c. Tugas kontroler:

1) Menerima nilai resistansi dari lima sensor tekuk, dimana ada sepuluh nilai, sebab setiap sensor memiliki dua bagian seperti dijelaskan pada Gambar 2.4d.
2) Dilengkapi sensor orientasi, yaitu accelerometer dan magnetometer untuk mendeteksi orientasi tangan. Sensor orientasi terdapat dalam IC KMX62 produksi perusahaan Kionix. Kedua sensor tersebut masing-masing menghasilkan tiga nilai dari sumbu-x, y, dan z. Sehingga ada enam data dari sensor orientasi.
3) Kini, sejumlah 16 data terkumpul, yaitu 10 data dari sensor tekuk dan 6 data dari sensor orientasi.

Dengan melihat Gambar 2.5 untuk sendi MCP dan PIP, maka Tabel 2.2 menunjukkan 17 data yang dikeluarkan kontroler untuk diolah pada komputer peneliti. Adapun transmisi data dari kontroler ke

komputer peneliti menggunakan kabel USB. Data ke-17 dengan index 16, bertuliskan "reserved" yang berarti masih belum digunakan atau dicadangkan untuk pengembangan berikutnya.

Tabel 2.2. Data Field pada Sensor Tekuk dan Sensor Orientasi di Sarung Tangan

Nomor/indeks field	Keterangan*
0	MCP kelingking
1	PIP kelingking
2	MCP jari manis
3	PIP jari manis
4	MCP jari tengah
5	PIP jari tengah
6	MCP jari telunjuk
7	PIP jari telunjuk
8	MCP ibu jari
9	PIP ibu jari
10	Data accelerometer X
11	Data accelerometer Y
12	Data accelerometer Z
13	Data magnetometer X
14	Data magnetometer Y
15	Data magnetometer Z
16	Reserved

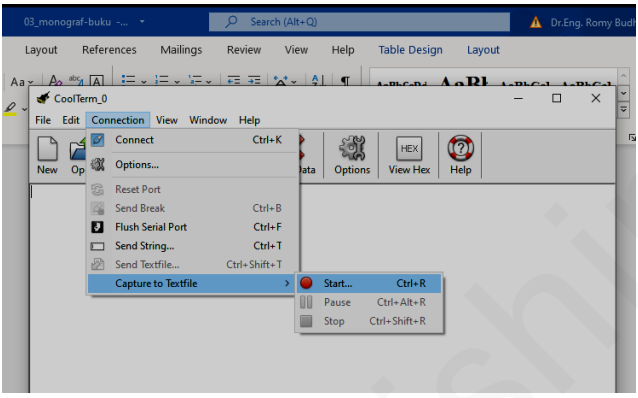
*MCP: sendi metacarpo-phalangeal kelingking
PIP: sendi proximal interphalangeal kelingking

2.2.4. Software Akuisisi Data Sarung Tangan

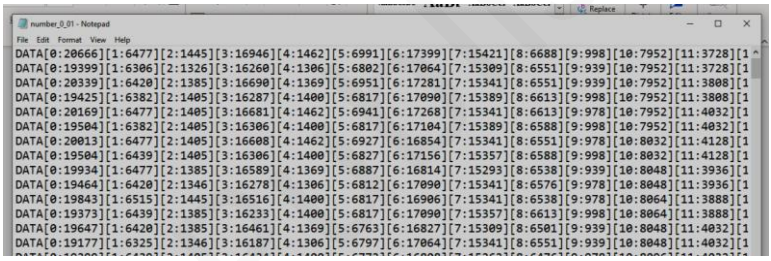
Data dari sensor melalui kontroler akan disalurkan ke komputer/laptop melalui USB port. Pengambilan data dapat dilakukan secara offline maupun realtime.

Perekaman data offline untuk keperluan training dapat menggunakan aplikasi freeware CoolTerm serial port terminal. Hasil rekaman file .txt dapat dikonversi ke .csv selanjutnya menjadi bahan untuk proses pelatihan di machine learning menggunakan Python.

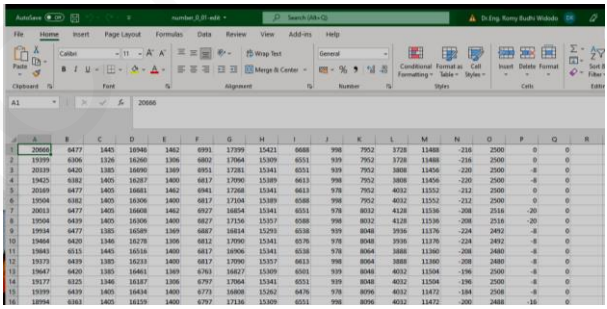
Gambar 2.7a menunjukkan cara bagaimana aplikasi CoolTerm serial port terminal menyimpan data ke .txt, melalui menu Connection→ Capture to Textfile.



(a)



(b)



(c)

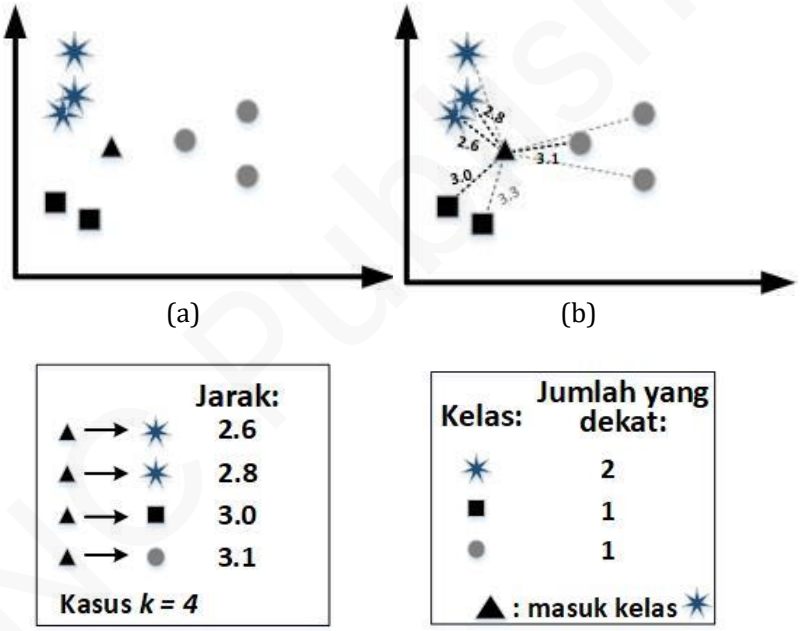
Gambar 2.7 a) Tampilan aplikasi CoolTerm untuk merekam data serial; b) Tampilan data teks (.txt); c) Hasil data diolah menjadi .csv file. (Sumber: Pengujian)

BAB 3

Metode k-Nearest Neighbors (kNN)

3.1. Cara Kerja Metode kNN

K-Nearest Neighbors (kNN) adalah metode jenis supervised machine learning yang dapat digunakan untuk klasifikasi dan regresi. Cara kerjanya menggunakan data latih (*training data*) dan mengklasifikasikan data tes berdasarkan jarak ke data latih tersebut. Tujuannya mencari sejumlah *k* tetangga terdekat antara data tes dan data latih. Kemudian hasil klasifikasi ditentukan melalui label kelas yang terbanyak pada jangkauan *k* tetangga terdekat tersebut. Ilustrasi metode ini seperti pada Gambar 3.1. Untuk kemudahan ilustrasi digunakan sumbu dua dimensi dengan koordinat (x_1, x_2), hal tersebut semisal mewakili dua nilai sensor. Dalam contoh ini ada tiga kelas yang menjadi target klasifikasi.



Gambar 3.1 Ilustrasi algortitma kNN.

Gambar 3.1a) Dalam contoh ini, akan dicari apakah data tes ▲ masuk ke suatu kelas, apakah masuk ke kelas ✶ atau ■ atau ● ?

Gambar 3.1b) Metode kNN menghitung jarak antara data tes ▲ dengan semua data latih. Pada gambar terlihat perhitungan yang menghasilkan jarak 2.6, 2.8, 3.0, 3.1, dan 3.3.
Gambar 3.1c) Pada contoh ini digunakan nilai *k*=4 sehingga diurutkan empat jarak yang dekat sesuai rankingnya. Terlihat bahwa jarak paling dekat adalah 2.6.
Gambar 3.1d) Pada langkah ini dihitung berapa jumlah label yang dekat pada kelas-kelas tersebut. Terlihat ada 2 label pada kelas ✸, 1 label pada kelas ■, dan 1 label pada kelas ●. Dari sini terlihat bahwa data tes diklasifikasikan ke kelas bintang ✸. Demikian gambaran metode kNN.

Metode kNN termasuk dalam *Instance-Based Learning* (IBL), yang merupakan teknik *machine learning* yang baru belajar ketika sebuah masukan diberikan untuk klasifikasi. Lain halnya dengan metode-metode *fast learner* seperti Decision Tree, Naïve Bayes, ANN, dan SVM; teknik IBL dinamakan *lazy learner* [14]. Kalau dikenal metode ANN (Artificial Neural Network) memiliki model hasil latihan yang berisi bobot-bobot tiap hubungan neuronnya, maka di kNN tidak ada model seperti itu.

Metode kNN secara umum terdiri atas dua langkah [14]. Langkah pertama adalah pelatihan untuk menyimpan setiap pola latih. Langkah kedua adalah klasifikasi. Pada saat klasifikasi sebuah pola, kNN memeriksa semua pola latih untuk menemukan sejumlah *k* pola terdekat.

Proses pelatihan pada kNN menghasilkan *k* yang memberikan akurasi tertinggi dalam menggeneralisasi data-data yang akan datang. Salah satu tantangan pada proses pelatihan adalah menentukan nilai *k* yang optimum. Nilai *k* bisa ditentukan berdasarkan jarak, kesamaan, dan ketidaksamaannya sesuai jenis fitur data. Pada kasus diatas penentuan *k* menggunakan jarak. Umumnya digunakan perhitungan jarak *Euclidean*, *Minkowski*, dan *Manhattan distance*.

Pada Gambar 3.1b) perhitungan jarak menggunakan *Euclidean distance* untuk dua titik, dengan rumus seperti pada persamaan (3.1).

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (3.1)$$

Langkah-langkah klasifikasi dirangkumkan sebagai berikut:

1) Menentukan nilai *k.*
2) Menghitung jarak antara data tes yang baru dan tetangga terdekat sejumlah *k* pada data latih.
3) Memeriksa dengan *voting*, di kelas manakah yang terbanyak jumlah tetangganya. Yang memiliki jumlah tetangga terdekat terbanyak akan dipilih sebagai hasil kelas klasifikasi.

Permasalahan yang sering muncul adalah bagaimana memilih nilai *k* yang optimum.

Pemilihan nilai *k* yang optimum dilakukan saat tahap pelatihan. Tidak ada metode yang pasti untuk menentukan nilai *k*, namun pendekatan berikut memberikan hasil yang baik:

- Mengawali *k* dengan angka acak.
- Menghindari nilai *k* yang terlalu kecil sebab akan menimbulkan hasil yang tidak stabil.
- *Error* adalah hasil klasifikasi yang dibandingkan dengan kelas sebenarnya. Membuat tabel/grafik yang berisi perbandingan *error* dengan berbagai nilai *k*. Memilih nilai *k* akhir yang memiliki *error rate* kecil.
- Sebaliknya dapat juga dibuat tabel/grafik perbandingan antara berbagai nilai *k* dengan akurasi. Memilih nilai *k* pada saat akurasi tertinggi.

Setelah nilai *k* optimum diperoleh maka nilai *k* tersebut digunakan untuk perhitungan selanjutnya jika ada data tes yang akan diklasifikasikan.

Perbaikan kNN diusulkan untuk mengatasi kelemahan-kelemahan. Sampai saat ini ada puluhan varian kNN hasil modifikasi para ahli seperti dijelaskan dalam [14]. Modifikasi dikelompokkan kedalam empat kategori:

1. Untuk mereduksi sensitivitas kNN terhadap fitur-fitur yang kurang relevan, dilakukan perbaikan dengan fungsi jarak atau lebih dikenal *feature selection*. Hanya fitur yang memiliki peran besar diambil sebagai fitur sebenarnya. Pada bab selanjutnya, penelitian ini menggunakan modifikasi kNN jenis ini pada saat pra pemrosesan data.
2. Perbaikan dengan ukuran ketetanggaan.

3. Perbaikan dengan estimasi probabilitas kelas untuk mereduksi sensitivitas terhadap data berderau dan pencilan data.
4. Perbaikan dengan struktur data untuk mereduksi kompleksitas waktu dan memori.

3.2. Mengukur Performansi

Performansi kerja kNN dan *machine learning* untuk klasifikasi pada umumnya diukur menggunakan metrik akurasi, *confusion matrix* (*error matrix*), dan *classification report.*

3.2.1. Akurasi

Skor akurasi dihitung dengan persamaan (3.2).

$$Akurasi = \frac{\text{Jumlah prediksi yang benar}}{\text{Jumlah seluruh prediksi}} \quad (3.2)$$

Skor akurasi adalah metrik yang paling sederhana, mengukur berapa kasus prediksi yang benar dibagi dengan jumlah seluruh kasus. Jika semua prediksi benar, akan menghasilkan skor akurasi = 1.0, dan jika semua prediksi salah maka skor akurasi = 0.

Namun skor akurasi mengandung kelemahan, sebab tidak memberikan informasi pengaruh ketidakseimbangan jumlah *false positive* dan *false negative*. Untuk itu dua metrik yang lain akan memberikan informasi yang lebih lengkap.

3.2.2. Confusion matrix

Ada jenis klasifikasi yang biner, dimana terdapat dua kelas, maupun klasifikasi yang *multiclass*, jika lebih dari dua kelas. Gambar 3.2 menunjukkan bentuk *confusion matrix* untuk dua kelas. Sedangkan Gambar 3.3 untuk *multiclass,* misalnya empat kelas.

		Nilai prediksi	
		1	**0**
Nilai aktual	**1**	*TP*	*FN*
	0	*FP*	*TN*

(a)

		Nilai prediksi	
		1	**0**
Nilai aktual	**1**	*120*	*8*
	0	*7*	*115*

(b)

Gambar 3.2 a)Confusion matrix untuk dua kelas (biner), misalnya kelas "1" dan "0" dimana TP=True Positive; TN=True Negative; FP=False Positive; dan FN=False Negative; b)Contoh dalam angka

Pada praktiknya, jika ada dua kelas yang akan diklasifikasikan, kelas mana yang dijadikan kelas 1 (positif) dan kelas 0 (negatif) tergantung kondisi klasifikasi yang ditentukan pendesain *machine learning*. Berikut pengertian TP, TN, FP dan FN:

True Positive: Hasil prediksi kelas 1 sama dengan hasil aktualnya.

True Negative: Hasil prediksi kelas 0 sama dengan hasil aktualnya.

False Positive: Untuk kelas 0, dikehendaki kelas 0 sebagai hasil prediksi, namun hasil model memprediksi 1.

False Negative: Untuk kelas 1, dikehendaki kelas 1 sebagai hasil prediksi, namun hasil model memprediksi 0.

Dari *confusion matrix* di atas dapat dihitung rasio false positive terhadap false negative dan skor akurasinya. Perhitungannya sebagai berikut: Jumlah false positive dan false negative adalah (7+8) = 15. Jika dibagi dengan total prediksi yang sudah dihasilkan, maka

$$\frac{(\text{False positive} + \text{False negative})}{\text{Total prediksi}} = \frac{15}{120+8+7+115} = 0.06 = 6\%$$

Artinya model gagal mengklasifikasikan 6% dari seluruh kasus. Skor akurasi dapat dihitung dari komplemen persentase kegagalan yaitu 94% atau menggunakan persamaan (3.2) sebagai berikut:

$$Akurasi = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{235}{120+115+7+8} = 0.94 = 94\%$$

		Nilai prediksi			
		1	**2**	**3**	**4**
Nilai aktual	**1**	*A*	*B*	*C*	*D*
	2	*E*	*F*	*G*	*H*
	3	*I*	*J*	*K*	*L*
	4	*M*	*N*	*O*	*P*

Gambar 3.3 Confusion matrix untuk multiclass, sebagai contoh ada empat kelas (kelas 1, 2, 3, dan 4), dimana TP=True Positive; TN=True Negative; FP=False Positive; dan FN=False Negative

Sedangkan untuk kasus multi kelas seperti contoh di Gambar 3.3; pengertian TP, TN, FP, dan FN agak berbeda. Pada multi kelas, fokus pembahasan dilakukan per kelas. Misal saat ini fokus pada kelas 1, maka TP adalah A. Berikut nilai TN, FP, dan FN jika fokus pada kelas 1.

True Positive: Jika kelas 1 maka TP=A. Jika fokus pembahasan kelas 2, maka TP=F. Jika kelas 3, maka TP=K, dan jika kelas 4, maka TP=P.

True Negative: Jumlah semua sel kecuali sel-sel pada kolom dan baris kelas 1, maka TN=F+G+H+J+K+L+N+O+P

False Positive: Jumlah semua nilai aktual selain kelas 1 yang diprediksi sebagai kelas 1, maka FP=E+I+M

False Negative: Jumlah semua nilai prediksi yang bukan kelas 1 saat nilai aktualnya 1, maka FN=B+C+D.

3.2.3. Classification report

Classification report menghasilkan tiga metrik evaluasi yaitu *precision*, *recall*, dan *f1-score*. Contoh tampilan classification report dengan library Python Scikit-learn untuk 20 kelas sebagai berikut:

Classification report:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	351
2	0.99	1.00	0.99	343
3	1.00	0.99	0.99	358
4	1.00	0.99	0.99	404
5	0.98	0.99	0.99	349
6	1.00	0.99	1.00	478

7	1.00	1.00	1.00	427
8	0.98	0.99	0.99	390
9	0.98	1.00	0.99	409
10	1.00	0.99	0.99	402
11	1.00	1.00	1.00	470
12	1.00	1.00	1.00	449
13	0.99	0.98	0.99	584
14	1.00	0.99	1.00	581
15	0.98	0.99	0.99	550
16	0.97	0.96	0.97	523
17	0.94	0.96	0.95	442
18	0.99	0.99	0.99	493
19	1.00	1.00	1.00	544
20	1.00	1.00	1.00	482
accuracy			0.99	9029
macro avg	0.99	0.99	0.99	9029
weighted avg	0.99	0.99	0.99	9029

Precision adalah rasio dari prediksi true positive terhadap total prediksi kasus positif. Persamaan (3.3) menunjukkan formula *precision*.

$$precision = \frac{TP}{FP + TP} \tag{3.3}$$

Semakin tinggi *precision* menandakan rendahnya kejadian false positive. Jika false positive nol, maka precision maksimal 1.0.

Jadi *precision* ingin menjawab pertanyaan seberapa tepat model ketika memprediksi *positive outcome* (dalam hal ini kelas 1)? Atau kemampuan model untuk tidak memberi label negatif pada kasus yang positif. Untuk kasus multi kelas, nilai *precision* dapat diperoleh dari rata-rata *precision* pada setiap kelas.

Recall adalah rasio prediksi true positive terhadap total kasus positif aktual. Persamaan (3.4) menunjukkan formula *recall*.

$$recall = \frac{TP}{FN + TP} \tag{3.4}$$

Metrik *recall* ingin menjawab pertanyaan: berapa luaran positif secara benar diklasifikasikan sebagai positif? Sehingga dapat dipahami juga sebagai kemampuan model untuk identifikasi semua kasus positif. Untuk kasus multi kelas, nilai *recall* dapat diperoleh dari rata-rata *recall* pada setiap kelas.

f1-score adalah rata-rata harmonik dari *precision* dan *recall*. Artinya rata-rata dengan bobot pada *precision* dan *recall*. Persamaan (3.5) merupakan persamaan harmonik yang sama maknanya dengan persamaan (3.6).

$$\frac{1}{f_1} = \frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right) \tag{3.5}$$

$$f_1 = 2\left(\frac{precision\ x\ recall}{precision + recall}\right) \tag{3.6}$$

f1-score lebih banyak dipakai untuk membandingkan antar model daripada asesmen performansi dalam suatu model.

3.2.4. MAE dan RMSE

Pada penggunaan *machine learning* untuk regresi, yaitu memprediksi variabel kontinu, dua metrik yang sering digunakan adalah *mean absolute error (MAE)* dan *Root Mean Square Error (RMSE)*. MAE mengukur error rata-rata pada suatu himpunan prediksi, yakni seberapa jauh garis regresi menyimpang dari data poin sebenarnya. Sedangkan RMSE mengukur standar deviasi dari error prediksi. RMSE menginformasikan seberapa menyebarnya atau berkumpulnya error prediksi dalam hubungannya dengan optimal fit.

Pada hitungan RMSE, error yang akan dihitung dikuadratkan, sehingga RMSE lebih sensitif terhadap error yang besar dibandingkan dengan MAE. RMSE tidak mudah diinterpretasikan dibandingkan dengan MAE. RMSE lebih banyak digunakan untuk memberikan umpan balik apakah suatu prediksi baik atau buruk. Sedangkan MAE memberikan asesmen error rata-rata pada tiap prediksi.

BAB 4

Bagaimana Mendesain dan Menggunakan kNN?

Langkah-langkah dalam perancangan *machine learning* mengacu pada saran dari Theobald dkk [15]. Disarikan terdapat sepuluh tahapan, yaitu: (1) import library, (2) import dataset, (3) *exploratory data analysis*, (4) data scrubbing, (5) algoritma pre-model, (6) split data, (7) menentukan algoritma machine learning, (8) prediksi, (9) evaluasi, dan (10) optimasi.

Tahap persiapan hardware dan penjelasan sensor telah diuraikan pada Bab 2 dan pada [16]. Klasifikasi yang diharapkan adalah angka 1-20. Jika diamati dari bahasa isyarat pada Gambar 2.1, isyarat angka 10 dan angka belasan (10-20) membutuhkan ayunan punggung tangan ke bawah jika dibandingkan isyarat angka 1-9 yang statis dan tidak membutuhkan ayunan tangan. Karena bahasa isyarat 1-20 membutuhkan ayunan tangan maka di penelitian ini digunakan data sensor accelerometer selain sensor tekuk. Untuk klasifikasi bahasa isyarat angka, kelima sensor yang terpasang memiliki sepuluh fitur dominan dan enam fitur tambahan dari sensor accelerometer dan magnetometer [13]; sehingga total ada 16 fitur, yaitu :

x_1: MCP (*metacarpo-phalangeal*) kelingking
x_2: PIP (*proximal interphalangeal*) kelingking
x_3: MCP jari manis
x_4: PIP jari manis
x_5: MCP jari tengah
x_6: PIP jari tengah
x_7: MCP jari telunjuk
x_8: PIP jari telunjuk
x_9: MCP ibu jari
x_{10}: PIP ibu jari
x_{11}: Accelerometer sumbu x
x_{12}: Accelerometer sumbu y
x_{13}: Accelerometer sumbu z
x_{14}: Magnetometer sumbu x
x_{15}: Magnetometer sumbu y
x_{16}: Magnetometer sumbu z

Perangkat lunak yang digunakan adalah Python dengan notebook Jupyter. Instalasi Python menggunakan pre-packaged Python Distribution Anaconda. Adapun library Python yang digunakan adalah Pandas, NumPy, Sklearn (preprocessing, feature_selection, decomposition, dan model_selection)

Pada bagian ini dijelaskan hasil beberapa tahapan dalam desain machine learning. Dataset terdiri atas 16 kolom (x_1 hingga x_{16}).

4.1. Tahap 1: Import library/package

Salah satu kelebihan Python untuk machine learning karena dilengkapi package yang lengkap. Gambar 4.1 menunjukkan potongan program import library.

```
import pandas as pd
import sklearn.preprocessing as sklp
import numpy as np
import seaborn as seab
```

Gambar 4.1. Import library

Library Pandas banyak digunakan mengelola dan presentasi data termasuk tabulasi data. Dataset akan dipanggil dalam bentuk dataframe Pandas.

Library NumPy digunakan untuk mengolah multi-dimensional array dan matrik, menghitung beberapa fungsi matematika.

Library Scikit-learn (Sklearn) merupakan library inti pada machine learning. Sklearn tergolong library untuk *shallow algorithm* (model diprediksi langsung dari fitur input, berbeda dengan deep learning dimana output tergantung dari hasil layer sebelumnya). Didalam Sklearn memuat logistic regression, decision trees, linear regression, gradient boosting, dan lain-lain. Sklearn juga digunakan untuk evaluasi seperti menghitung mean absolute error, serta metode partisi data, yaitu split dan cross validation. Fungsi penting Sklearn dapat melakukan proses pelatihan (model train) dan menggunakan model hasil pelatihan untuk prediksi data tes.

Masih ada library lain yang mendukung visualisasi data misalnya Matplotlib dan Seaborn. Keduanya menghasilkan visualisasi data yang berkualitas tinggi. Sedangkan library TensorFlow cocok untuk deep learning dan artificial neural network.

4.2. Tahap 2: Import dataset

Folder kerja yang digunakan secara default pada C:/users/..; pada folder kerja disiapkan .csv file. File dengan format csv (comma-separated values) memiliki format data dalam teks dan separatornya koma, lain dengan file excel dimana format dalam tabular. Bentuk file csv pada data gestur tangan seperti pada Gambar 4.2. Terdapat 17 fitur dalam satu baris, fitur ke-1 hingga ke-16 berasal dari 16 sensor, sedangkan data ke-17 adalah kelas yang diberikan oleh peneliti. Kelas 1 artinya bahasa isyarat 1, kelas 20 berarti gestur tangan untuk bahasa isyarat 20.

11001,7893,2554,11935,1322,12992,21177,7970,6217,1785,13040,-480,6560,160,2176,388,1

11073,8064,2554,11721,1028,12695,20553,8038,6105,1785,13040,-480,6560,160,2176,388,1

:

:

8446,3723,2554,7801,280,9528,13428,3851,1871,3993,12752,5120,6144,-404,2416,296,20

9125,3780,2613,8386,342,9807,13816,3899,1896,4032,12816,4944,5888,-408,2420,292,20

Gambar 4.2. Cuplikan data csv gestur tangan

Library Pandas memungkinkan import csv file dan melakukan manipulasi atau pengolahan data csv tanpa merubah file asli, untuk itu Pandas merubahnya menjadi dataframe sehingga dapat dikelola dalam *development environment*. Cuplikan pada Gambar 4.3 adalah tahapan import dataset, set presisi 2 digit, menampilkan dimensi dataframe, dan menampilkan contoh dua baris dari dataframe.

```
# library pada Gambar 4.1 di sini ….
path = r"~/data_Python_study/gabunganAngka-train-label.csv"

#membuat header kolom sebab data tidak memiliki header
headernames =
['mcpPinky','pipPinky','mcpRing','pipRing','mcpMid',
'pipMid','mcpIndex','pipIndex','mcpThumb','pipThumb',
'accX','accY','accZ','magX','magY','magZ','output']

dataframe = pd.read_csv(path, names=headernames)
pd.set_option('precision',2)

print(dataframe.shape)     #menampilkan dimensi data
print(dataframe.head(n=2))#menampilkan 2 baris data pertama
```

Gambar 4.3. Cuplikan import dataset, set presisi, dimensi data, dan tampilan isi dataframe

Sedangkan Gambar 4.4 menunjukkan hasil eksekusi Gambar 4.3.

```
(45147, 17)
   mcpPinky  pipPinky  mcpRing  pipRing  mcpMid  pipMid  mcpIndex  pipIndex \
0     11001      7893     2554    11935    1322   12992     21177      7970
1     11073      8064     2554    11721    1028   12695     20553      8038

   mcpThumb  pipThumb  accX  accY  accZ  magX  magY  magZ  output
0      6217      1785  13040  -480  6560   160  2176   388       1
1      6105      1785  13040  -480  6560   160  2176   388       1
```

Gambar 4.4. Hasil import dataset berupa dimensi data dan tampilan isi dataframe

4.3. Tahap 3: Exploratory data analysis (EDA)

Tahap EDA bertujuan mempersiapkan dataset untuk pemrosesan dan analisis lebih lanjut. Didalamnya terdapat pemahaman bentuk dan distribusi data, scan *missing value*, dan analisis korelasi untuk melihat fitur mana yang relevan.

Untuk mengamati jumlah data yang kosong (null) digunakan perintah **.isnull ( ).sum( )**. Gambar 4.5 menunjukkan hasil eksekusi pengecekan *missing value*, diperoleh tidak ada data yang hilang pada semua fitur.

```
# library pada Gambar 4.1 di sini ….
# list program import dataset pada Gambar 4.3 di sini….

  dataframe.isnull().sum()
```

#hasil eksekusi pada dataframe

```
mcpPinky     0
pipPinky     0
mcpRing      0
pipRing      0
mcpMid       0
pipMid       0
mcpIndex     0
pipIndex     0
mcpThumb     0
pipThumb     0
accX         0
accY         0
accZ         0
magX         0
magY         0
magZ         0
output       0
dtype: int64
```

Gambar 4.5. Cuplikan untuk melihat *missing value* pada dataframe

Jika pada Gambar 4.5, dijumpai ada data yang hilang maka beberapa teknik bisa digunakan untuk proses selanjutnya pada Tahap 5 (*data scrubbing*), yaitu:

1) Jika variabel berjenis *float* maka dapat dihitung nilai rata-rata (*mean*). Data kosong dapat diisi dengan *mean*, menggunakan perintah:

```
dataframe['nama_variabel'].fillna((dataframe['nama_
variabel'].mean()), inplace=True)
```

2) Jika variabel berjenis *string* maka dapat diganti dengan *string*, contoh perintah:

```
dataframe['nama_variabel'].fillna("string-yang-
akan-diisikan", inplace=True)
```

3) Pilihan terakhir adalah menghapus data dengan perintah **.dropna()**, namun beberapa pertimbangan dalam penghapusan data perlu dilakukan.

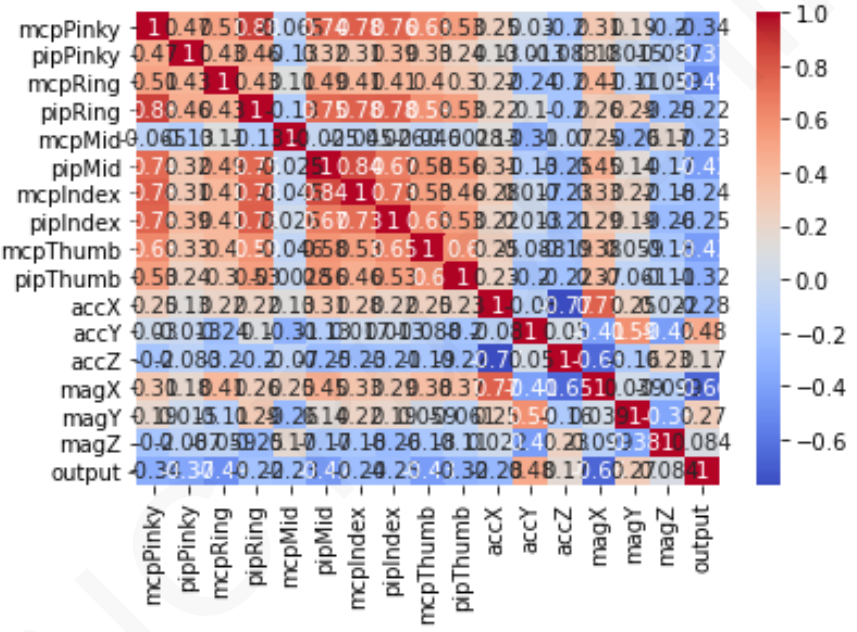
Selanjutnya korelasi antar fitur bisa dicari menggunakan Pearson's correlation coefficient, semakin mendekati 1 atau -1 korelasi semakin menguat dengan arah yang sama atau berbeda. Perintah korelasi menggunakan library Seaborn, jika dikehendaki tampilan dalam heatmap untuk visualisasi dapat ditambahkan perintah heatmap. Gambar 4.6 menunjukkan perintah dan tampilan heatmap dari korelasi antar fitur. Tampilan angka/nilai korelasi pada heatmap tidak jelas karena banyaknya jumlah fitur, sehingga pengamatan melalui warna lebih sesuai untuk heatmap. Sedangkan jika diinginkan pengamatan dalam angka, penggunaan perintah **print** lebih sesuai.

```
# library pada Gambar 4.1 di sini ….
# list program import dataset pada Gambar 4.3 di sini….

#jika menggunakan Pearson's correlation
df_corr = dataframe.corr(method='pearson')

#membuat heatmap dan print hasil korelasi
seab.heatmap(df_corr, annot=True, cmap='coolwarm')
print(df_corr)
```

#hasil eksekusi heatmap (tampilan terlalu kecil sebab ada 16 fitur)



Gambar 4.6. Cuplikan perhitungan korelasi antar fitur dan penggambaran heatmap.

4.4. Tahap 4: Split data

Klasifikasi tergolong *supervised machine learning* dimana input data set dan target diberikan bersama-sama. Dalam klasifikasi, kita perlu menyediakan minimal dua set data, yaitu: data latih dan data uji. Data latih digunakan untuk membangun model prediksi. Sedangkan data uji digunakan untuk menentukan tingkat akurasi model yang sudah dihasilkan. Supaya model valid, maka kuncinya adalah data uji

tidak boleh dipakai untuk proses latih, membangun model, dan mengoptimasi model.

Selain kedua jenis data tersebut, beberapa peneliti memiliki sekelompok data yang disebut data validasi. Namun umumnya keberadaan data validasi bukanlah suatu keharusan dalam proses *machine learning*. Urutannya menjadi sebagai berikut: 1) Model awal dibangun dengan data latih; 2) Data validasi dicobakan pada model, digunakan untuk memberi masukan dan mengoptimasi parameter model; 3) Data uji digunakan untuk menghitung error model dalam memprediksi luaran. Namun dalam kNN tidak mengenal istilah model melainkan pola-pola latih yang dibentuk saat proses pelatihan untuk menemukan nilai *k* yang optimum.

Dalam implementasi klasifikasi, langkah pertama adalah melatih *classifier*. Sehingga diperlukan data latih. Porsi ketiga jenis data dalam penelitian ini adalah 60% untuk data latih, 20% data validasi, dan 20% data uji. Teknik split data, mula-mula data dibagi 60%:40%, 60% data akan menjadi data latih. Data yang 40% akan dibagi menjadi 20%:20% untuk data validasi dan data uji. Gambar 4.7 adalah proses split data menggunakan class **sklearn.model_selection.train_test_split()**. Sedangkan Gambar 4.8 adalah hasil eksekusi program.

```
# library pada Gambar 4.1 di sini ….
import sklearn.model_selection as sklms
# list program import dataset pada Gambar 4.3 di sini….

#memisahkan tahap I, 60%-40% untuk training dan testing
#data 1

X_train, X_test1, y_train, y_test1 =
sklms.train_test_split (X, y, test_size = 0.4,
random_state = 8, shuffle = True)
#menggunakan seed number 8

print("\nukuran X train: ", X_train.shape)
print("ukuran y train: ", y_train.shape)

print("\nukuran X test sementara/tahap 1: ", X_test1.shape)
print("ukuran y test sementara/tahap 1: ", y_test1.shape)

#menyimpan ke excel file
df_test_excel=X_test1.to_excel("df_test_excel.xlsx")
#untuk menyimpan X_test1 ke excel

#---------------------------------
#memisahkan tahap II, 50%-50% untuk validation dan testing dat

X_test, X_validation, y_test, y_validation =
sklms.train_test_split (X_test1, y_test1, test_size = 0.5,
random_state = 8, shuffle = True)
#menggunakan seed number 8

print("\nukuran X validation: ", X_validation.shape)
print("ukuran y validation: ", y_validation.shape)

print("\nukuran X test: ", X_test.shape)

print("ukuran y test: ", y_test.shape)

#---------------------------------
```

Gambar 4.7. Proses split data menjadi data latih, data validasi, dan data uji.

```
#hasil eksekusi

ukuran X train:  (27088, 16)
ukuran y train:  (27088, 1)

ukuran X test sementara/tahap 1:  (18059, 16)
ukuran y test sementara/tahap 1:  (18059, 1)

ukuran X validation:  (9030, 16)
ukuran y validation:  (9030, 1)

ukuran X test:  (9029, 16)
ukuran y test:  (9029, 1))
```

Gambar 4.8. Hasil eksekusi program Gambar 4.7

4.5. Tahap 5: Data scrubbing

Data scrubbing adalah istilah umum yang mewakili "manipulasi data untuk keperluan analisis, termasuk mempersiapkan jenis data." Beberapa algoritma memerlukan jenis data yang tepat, misalnya regresi linier menggunakan variabel kontinu, Gradient Descent dan k-Nearest Neighbors membutuhkan data yang sudah di-*rescaling*.

Beberapa teknik *data scrubbing* seperti pernah diuraikan sedikit di Tahap 3, yaitu 1) penghapusan data, 2) penanganan *missing value.* Teknik lain adalah 3) *one-hot encoding*, digunakan untuk merubah variabel kategorikal menjadi bentuk biner, jika data kategorikal tersebut tidak dapat dipakai untuk algoritma clustering dan regresi, misalnya pengubahan gender dan nama kota menjadi biner. Jenis *data scrubbing* yang lain adalah: 4) Scaling, 5) Normalisasi, 6) Binerisasi, 7) Standardisasi, 8) Data labelling.

Scaling, secara umum digunakan untuk rescaling data menjadi 0 sampai 1. Gradient Descent dan k-Nearest Neighbors membutuhkan scaling. Gambar 4.9 menunjukkan proses rescaling dengan MinMaxScaler pada data gestur tangan, sedangkan Gambar 4.10 menunjukkan hasil eksekusi program.

```
# library pada Gambar 4.1 di sini ….
# list program import dataset pada Gambar 4.3 di sini….
# list program split data pada Gambar 4.7 di sini….

#membuat penskala 0 sampai 1
data_scaler = sklp.MinMaxScaler(feature_range = (0,1))
np.set_printoptions(precision=1)

print("X train sebelum scaling: ")
print(X_train[0:3])

X_train = data_scaler.fit_transform(X_train)
X_validation = data_scaler.fit_transform(X_validation)
X_test = data_scaler.fit_transform(X_test)

print("X train setelah scaling: ")
print(X_train[0:3])
```

Gambar 4.9. Proses rescaling pada data gestur tangan

#hasil eksekusi proses rescaling (tampilan 3 baris pertama saja)

```
X train sebelum scaling:
    mcpPinky pipPinky mcpRing pipRing mcpMid pipMid mcpIndex \
30994   2650      360      178      983      93     203       558
14626   9660     3286     3747    16261    1601   14458     20250
32947  12115     7674      950    11331     280    7229     12777

      pipIndex mcpThumb pipThumb  accX  accY accZ magX magY magZ
30994      308     2108       191  8976  4880  9888 -736  2304   392
14626    11409     4592      6152 13840   128  6256   40  2328   280
32947     9023     3880      1800 11568  4224  6176 -456  2336   396

X train setelah scaling:
[[0.1 0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.7 0.4 0.6 0.4 0.8 0.8]
 [0.3 0.1 0.1 0.5 0.  0.4 0.6 0.3 0.1 0.2 0.7 0.3 0.5 0.8 0.8 0.7]
 [0.4 0.2 0.  0.3 0.  0.2 0.4 0.3 0.1 0.1 0.7 0.4 0.5 0.6 0.8 0.8]]
```

Gambar 4.10. Hasil eksekusi program

Normalisasi, tahap ini bermanfaat jika data tergolong *Sparse dataset*, artinya data banyak yang bernilai nol. Tujuan normalisasi data: menormalisasi masing-masing data ke unit norm, untuk rescaling tiap baris data supaya panjangnya = 1. Normalisasi banyak digunakan untuk klasifikasi dan clustering teks. Normalisasi menggunakan library scikit-learn preprocessing yang dipakai adalah "Normalizer" class. Ada dua teknik normalisasi: L1 normalization dan L2 normalization

Binerisasi/Thresholding, konsep binerisasi data adalah menjadikan data 0 jika dibawah nilai *threshold*. Data menjadi 1 jika nilai data sama dengan atau diatas *threshold*. Penggunaan preprocessing ini lebih kepada tujuannya untuk mendeteksi ada atau tidak suatu fitur, bukan kepada nilainya yang dipentingkan. Binerisasi menggunakan library scikit-learn preprocessing, yang dipakai adalah "Binarizer" class.

Standardisasi, kegunaan standardisasi adalah untuk mentransformasi atribut data ke distribusi normal (normally distributed Gaussian). Dimana mean=0 dan SD=1. Preprocessing ini bermanfaat untuk metode *machine learning* seperti Regresi Linier dan Regresi Logistik; dimana data diasumsikan memiliki distribusi normal. Sehingga bila data sudah di-*rescale* dengan teknik ini, maka harapannya metode *machine learning* tersebut menghasilkan performansi yang tinggi. Untuk standardisasi ke mean=0 dan SD=1, digunakan library scikit-learn preprocessing, yang dipakai adalah "StandardScaler" class. Perintah yang digunakan adalah

```
data_scaler = sklp.StandardScaler()

X_train = data_scaler.fit_transform(X_train)
```

Data Labelling, data yang dikirim ke *machine learning* harus sesuai dengan tipe data yang diharapkan oleh metode *machine learning* itu sendiri dan class yang digunakan. Sebagian besar fungsi pada scikit learn membutuhkan label angka. Jika data masih memiliki label dalam *words* (kata/string) maka perlu proses "label encoding" untuk merubah menjadi label angka. Untuk data labeling digunakan library scikit-learn preprocessing, yang dipakai adalah "LabelEncoder" class.

4.6. Tahap 6: Algoritma pre-model

Kadang tidak semua fitur digunakan. Fitur yang paling relevan untuk prediksi akan dipilih dalam proses ini. Proses memilih fitur yang

akan dipakai untuk *machine learning* dinamakan *Feature Selection* atau *Attribute Selection* atau algoritma pre-model. Kelebihan penggunaan *feature selection* sebelum pemodelan data dengan *machine learning* adalah 1) mengurangi overfitting, 2) meningkatkan akurasi *machine learning* utamanya yang menggunakan regresi linier dan logistik, 3) mengurangi waktu pelatihan data.

Beberapa teknik *feature selection*: 1) menggunakan *automatic feature*: *univariate selection*, 2) menggunakan *recursive feature elimination*, 3) menggunakan *feature importance*, 4) menggunakan *unsupervised learning*, yaitu PCA (Principal Component Analysis) dan *k-Means Clustering*.

Pada kesempatan ini digunakan *feature importance*, teknik *feature importance* digunakan untuk memilih fitur yang penting. Pemilihannya berdasarkan skor, semakin tinggi skor maka atribut tersebut tergolong penting. Ekstrak fitur menggunakan class **sklearn.ensemble.ExtraTreesClassifier**. Gambar 4.11 menunjukkan proses ekstrak fitur dan Gambar 4.12 menunjukkan hasil eksekusi program.

```
# library pada Gambar 4.1 di sini ….
import sklearn.ensemble as skle
# list program import dataset pada Gambar 4.3 di sini….

array = dataframe.values   #two dimension tabular data
#memisahkan INPUT dan OUTPUT
X = array[:, 0:16]   #mengambil indeks ke-0 sampai 15
Y = array[:,16]      #mengambil indeks ke-16
print("\n X size: \n", X.shape)
print("\n Y size: \n", Y.shape)
#-----------------------------------------------------

# Extract feature dari dataset
model = skle.ExtraTreesClassifier()
model.fit(X,Y)
print("\nScores for each attributes:\n",
model.feature_importances_)
```

Gambar 4.11. Proses pemilihan fitur dengan *feature importance*.

#hasil eksekusi proses pemilihan fitur

X size:
(45147, 16)

Y size:
(45147,)

Scores for each attributes:
[0.05971364 **0.10022071 0.07325942 0.07230683 0.06274054 0.07278559** 0.0554361 **0.06361204 0.06370255 0.08837127** 0.02399304 0.04091566 0.02997139 **0.07177275** 0.06101359 **0.06018487**]

Gambar 4.12. Hasil eksekusi program pemilihan fitur dengan *feature importance*.

Dari hasil *feature importance* diperoleh 10 fitur penting yaitu: fitur 2 (PIP kelingking), 10 (PIP ibu jari), 14 (Magnetometer x), 3 (MCP jari manis), 4 (PIP jari manis), 6 (PIP jari tengah), 5 (MCP jari tengah), 8 (PIP jari telunjuk), 9 (MCP ibu jari), 16 (Magnetometer z).

Terlihat bahwa sensor pada kelima jari merupakan fitur penting yang tidak dapat disederhanakan. Berikutnya adalah data dari magnetometer x dan z yang merupakan sebagian dari 10 fitur penting. Dalam praktik data magnetometer akan mudah berubah sesuai arah subjek/partisipan saat percobaan dilakukan, sehingga fitur magnetometer dalam praktik akan dihilangkan dan digantikan fitur accelerometer. Fitur accelerometer (x_{11}, x_{12}, dan x_{13}) digunakan sebab angka 10 hingga 20 menggunakan anggukan tangan yang dapat dideteksi oleh sensor tersebut.

Berdasarkan analisis di atas, 11 fitur akan digunakan dalam klasifikasi, yaitu: fitur 2 (PIP kelingking), 10 (PIP ibu jari), 3 (MCP jari manis), 4 (PIP jari manis), 6 (PIP jari tengah), 5 (MCP jari tengah), 8 (PIP jari telunjuk), 9 (MCP ibu jari), 11 (Accelerometer x), 12 (Accelerometer y), dan 13 (Accelerometer z).

4.7. Tahap 7: Menentukan algoritma Machine Learning

Langkah ini meskipun dari awal sudah kita rencanakan, utamanya pada saat melakukan Tahap 5 (*data scrubbing*) di bagian scaling; perlu adanya pemeriksaan ulang. Adapun beberapa algoritma machine learning menurut [15] yang sesuai digunakan dalam penelitian ini adalah *k-Nearest Neighbors* (KNN). Beberapa karakteristik kNN yang

sesuai dengan sumber daya penelitian ini yaitu: 1) Output target: diskrit; 2) Sifat data: tidak ada nilai data yang hilang, dimensi data terbatas; 3) Metodologi: supervised; 4) *Computing resources*: sedang (*medium*); dan 5) Akurasi: sedang.

4.8. Tahap 8: Prediksi atau Klasifikasi

Algoritma kNN dapat digunakan untuk kasus prediksi (*regressor*) maupun klasifikasi. Pada penelitian ini kasusnya adalah klasifikasi dengan 20 kelas.

Langkah awal adalah menentukan nilai *k* pada algoritma kNN. Tabel 4.1 menunjukkan hasil percobaan penggunaan beberapa kandidat nilai *k* (ganjil) yang sering digunakan dalam berbagai penelitian.

Tabel 4.1. Hasil pemilihan nilai *k* pada algoritma kNN

k	***False Positive***	***False Negative***	**Total**	**Akurasi**
3	106	106	212	0.988
5	127	127	254	0.986
7	148	148	296	0.984

Langkah berikutnya akan menggunakan *k*=3 dengan pertimbangan akurasi yang lebih tinggi dan jumlah *false* paling kecil diantara ketiga uji coba, yaitu bernilai 212. Gambar 4.13 memperlihatkan proses mendapatkan *confusion matrix* untuk evaluasi nilai *k*, data yang digunakan adalah data validasi.

```
# library pada Gambar 4.1 di sini ….

import sklearn.model_selection as sklms
import sklearn.neighbors as sklneigh
import sklearn.metrics as sklmet
# list program import dataset pada Gambar 4.3 di sini….
# list program split data pada Gambar 4.7 di sini….
# list program rescaling pada Gambar 4.9 di sini….

## SET algorithm
# k-NN, dengan k=3
classifier = sklneigh.KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train,y_train.values.ravel( ))
# contiguous flattened array for y_train
#-----------------------------------
## Evaluate k
#menggunakan data validation untuk menentukan k
y_prediction = classifier.predict(X_validation)

result1 =
sklmet.confusion_matrix(y_validation, y_prediction)
print("Confusion matrix:")
print(result1)

result2 =
sklmet.classification_report(y_validation, y_prediction)
print("")
print("Classification report:")
print(result2)

result3 =
sklmet.accuracy_score(y_validation, y_prediction)
print("")
print("Accuracy:")
print(result3)
#---------------------------------
```

Gambar 4.13. Proses mencari nilai *k*, memanfaatkan *confusion matrix* dan *accuracy score*

4.9. Tahap 9: Optimasi

Langkah optimasi digunakan untuk *tuning* hiperparameter. Untuk kasus *clustering* langkah optimasi dilakukan dengan kembali pada langkah sebelumnya dan memodifikasi jumlah kluster, atau merubah nilai hiperparameter dari teknik *tree-based learning*.

Model optimasi dapat diperoleh dengan cara manual melalui *trial and error* atau metode otomatis seperti *grid search.* Penggunaan *grid search* memiliki pembahasan tersendiri.

Penggunaan data validasi pada tahap sebelumnya bisa dilakukan untuk keperluan optimasi ini. Cara yang digunakan adalah menguji pola latihan dengan *k* yang sudah ditemukan sebelumnya pada data validasi. Apabila akurasi masih baik, maka bisa dilanjutkan ke tahap 10, namun jika akurasi menurun drastis maka dapat dilakukan memilih nilai *k* yang lain. Artinya nilai *k* sebelumnya yang dipilih belum dapat menggeneralisasi persoalan.

4.10. Tahap 10: Evaluasi

Langkah evaluasi model dilakukan dengan menerapkan model pada data uji (data tes). Kemudian menguji klasifikasi, misalnya 10 baris pertama dari data uji, dilihat hasil klasifikasinya. Pada kNN berarti menerapkan pola latih sebelumnya dengan menggunakan nilai *k* optimum pada data tes. Gambar 4.14 adalah program yang sama dengan Gambar 4.13 namun perhitungan *confusion matrix*, *classification report*, dan *accuracy score* menggunakan y_test. Bagian akhir dari program di Gambar 4.14 adalah cara menggunakan pola latih yang telah dihasilkan untuk klasifikasi.

```
# library pada Gambar 4.1 di sini ….
import sklearn.model_selection as sklms
import sklearn.neighbors as sklneigh
import sklearn.metrics as sklmet
# list program import dataset pada Gambar 4.3 di sini….
# list program split data pada Gambar 4.7 di sini….
# list program rescaling pada Gambar 4.9 di sini….
#-----------------------------------
## Evaluate k
#menggunakan data validation untuk menentukan k
y_prediction = classifier.predict(X_test)

result1 = sklmet.confusion_matrix(y_test, y_prediction)
print("Confusion matrix:")
print(result1)

result2 =
sklmet.classification_report(y_test, y_prediction)
print("")
print("Classification report:")
print(result2)

result3 =
sklmet.accuracy_score(y_test, y_prediction)
print("")
print("Accuracy:")
print(result3)
#-----------------------
# Klasifikasi dengan memeriksa terhadap k pola latih

Xnew = X_test [0:10]  #misal evaluasi terhadap 10 data
ynew = classifier.predict(Xnew)
# print input dan hasil klasifikasi
print(Xnew)
print(ynew)
```

Gambar 4.14. Bagian evaluasi menggunakan data uji dan bagian menerapkan pola latih *kNN* untuk klasifikasi.

Hasil pengujian kNN terhadap semua data uji memiliki akurasi 0,989 dengan false positive=94 dan false negative=94; dimana nilai ketiganya lebih baik dibandingkan hasil validasi di Tabel 4.1.

Sedangkan hasil percobaan klasifikasi dengan 10 data masukan, diperoleh akurasi 100%. Gambar 4.15 merupakan hasil perhitungan *confusion matrix* dan klasifikasi 10 data masukan.

a)
Classification report:

	precision	recall	f1-score	support
1	1.00	1.00	1.00	351
2	0.99	1.00	0.99	343
3	1.00	0.99	0.99	358
4	1.00	0.99	0.99	404
5	0.98	0.99	0.99	349
6	1.00	0.99	1.00	478
7	1.00	1.00	1.00	427
8	0.98	0.99	0.99	390
9	0.98	1.00	0.99	409
10	1.00	0.99	0.99	402
11	1.00	1.00	1.00	470
12	1.00	1.00	1.00	449
13	0.99	0.98	0.99	584
14	1.00	0.99	1.00	581
15	0.98	0.99	0.99	550
16	0.97	0.96	0.97	523
17	0.94	0.96	0.95	442
18	0.99	0.99	0.99	493
19	1.00	1.00	1.00	544
20	1.00	1.00	1.00	482
accuracy			0.99	9029
macro avg	0.99	0.99	0.99	9029
weighted avg	0.99	0.99	0.99	9029

Accuracy: 0.9895891017831432
b)
Sepuluh baris data uji:
[[0. 0.7 0. 0. 0. 0. 0. 0. 0. 0.1 0.8 0.2 0.4 0.7 0.4 0.9]
[0. 0. 0. 0. 0. 0. 0. 0. 0.1 0.1 0.8 0.4 0.4 0.6 0.9 0.7]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.5 0.3 0.7 0.2 0.6 0.7]
[0.1 0.1 0. 0.1 0. 0. 0.2 0.1 0. 0.1 0.8 0.4 0.5 0.5 0.8 0.8]
[0.7 0.3 0.1 0.4 0. 0.5 0.5 0.5 0.6 0. 0.8 0.2 0.4 0.9 0.5 0.8]
[0.6 0.4 0.2 0.2 0. 0.8 0.8 0.4 0.3 0.4 0.8 0.2 0.4 0.8 0.4 0.9]
[0.3 0.9 0.2 0.2 0. 0. 0. 0. 0. 0. 0.7 0.3 0.5 0.7 0.7 0.8]
[0.3 0.1 0.1 0.4 0.1 0.6 0.5 0.3 0.1 0.3 0.8 0.3 0.5 0.7 0.8 0.8]
[0. 0.1 0. 0. 0. 0. 0. 0. 0. 0. 0.7 0.3 0.4 0.5 0.6 0.9]
[0.4 0.2 0.1 0.5 0. 0.2 0.3 0.4 0.2 0.1 0.7 0.4 0.5 0.4 0.8 0.7]]
Hasil klasifikasi sesuai dengan kelas sebenarnya, sebagai berikut:
[6 14 15 19 4 2 3 7 13 15]

Gambar 4.15. a) Tampilan hasil perhitungan confusion matrix dan akurasi terhadap seluruh data uji. b) Percobaan klasifikasi terhadap 10 data.

BAB 5
PENUTUP

Buku ini mendasari tahap klasifikasi gestur tangan yang dapat dikembangkan untuk klasifikasi huruf dan kata pada bahasa isyarat. Bersama dengan tujuan tersebut pada buku ini telah diketengahkan urutan perancangan machine learning difokuskan pada metode k-Nearest Neighbors. Dari tahapan ini telah dihasilkan 11 fitur dari 16 fitur sebelumnya melalui proses *feature importance*, yang dapat menjadi pertimbangan untuk memperkecil *resource* komputasi. Data telah dibagi menjadi data latih, data validasi, dan data uji, dengan perbandingan 60%:20%:20%. Hasil akurasi menggunakan data uji adalah 98.9%. Pembaca yang tertarik untuk mengembangkan klasifikasi bahasa isyarat untuk kata dapat berkolaborasi dengan penulis. Beberapa tantangan ke depan dalam pengembangan klasifikasi kata adalah modifikasi kNN. Seperti diketahui banyak variasi kNN oleh para peneliti kelas dunia yang berusaha mengurangi kelemahan kNN yaitu sifat solusi lokal dan mengurangi komputasi real time.

DAFTAR PUSTAKA

[1] Kementerian Kesehatan Republik Indonesia, *Laporan Nasional Riskesdas*. 2018.

[2] F.A. Prasetyo, "Disability and Health Issues: Evolution Concepts, Human Rights, Complexity of Problems, and Challenges (in Indonesian)," Jakarta, 2014. doi: 10.1007/s13398-014-0173-7.2.

[3] T. W. Chong and B. G. Lee, "American sign language recognition using leap motion controller with machine learning approach," *Sensors (Switzerland)*, vol. 18, no. 10, 2018, doi: 10.3390/s18103554.

[4] B. G. Lee and S. M. Lee, "Smart Wearable Hand Device for Sign Language Interpretation System with Sensors Fusion," *IEEE Sens. J.*, vol. 18, no. 3, 2018, doi: 10.1109/JSEN.2017.2779466.

[5] B. S. Lin, P. C. Hsiao, S. Y. Yang, C. S. Su, and I. J. Lee, "Data glove system embedded with inertial measurement units for hand function evaluation in stroke patients," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 11, 2017, doi: 10.1109/TNSRE.2017.2720727.

[6] J. D. Lemos, A. M. Hernandez, and G. Soto-Romero, "An instrumented glove to assess manual dexterity in simulation-based neurosurgical education," *Sensors (Switzerland)*, vol. 17, no. 5, 2017, doi: 10.3390/s17050988.

[7] L. Sbernini, L. R. Quitadamo, F. Riillo, N. Di Lorenzo, A. L. Gaspari, and G. Saggio, "Sensory-Glove-Based Open Surgery Skill Evaluation," *IEEE Trans. Human-Machine Syst.*, vol. 48, no. 2, 2018, doi: 10.1109/THMS.2017.2776603.

[8] Y. Zheng, Y. Peng, G. Wang, X. Liu, X. Dong, and J. Wang, "Development and evaluation of a sensor glove for hand function assessment and preliminary attempts at assessing hand coordination," *Meas. J. Int. Meas. Confed.*, vol. 93, 2016, doi: 10.1016/j.measurement.2016.06.059.

[9] "Online SIBI dictionary," 2020. https://pmpk.kemdikbud.go.id/sibi/ (accessed Jan. 28, 2022).

[10] R. A. Mursita, "Respon Tunarungu Terhadap Penggunaan Sistem Bahasa Isyarat Indonesa (SIBI) dan Bahasa Isyarat Indonesia (Bisindo) Dalam Komunikasi," *Inklusi*, vol. 2, no. 2, p. 221, 2015, doi: 10.14421/ijds.2202.

[11] Y. H. Laoly, *Undang-Undang Republik Indonesia Nomor 8 Tahun 2016 Tentang Penyandang Disabilitas*. 2016, pp. 1–102.

[12] Anonymous, *Kamus Sistem Isyarat Bahasa Indonesia & Bahasa Murni*. 2006.

[13] F. S. Systems, "Bend Sensor ® USB Glove Kit User Guide," 2016.

[14] Suyanto, *Machine Learning Tingkat Dasar dan Lanjut*. Informatika Bandung, 2018.
[15] O. Theobald, *Machine Learning with Python: A Practical Beginners' Guide*. Scatterplot Press, 2019.
[16] R. B. Widodo, W. Swastika, and A. B. Haryasena, "Studi Sensor dan Akuisisi Data Hand Gesture dengan Sarung Tangan," in *Conference on Innovation and Application of Science and Technology (CIASTECH)*, 2020, pp. 561–568, [Online]. Available: http://publishing-widyagama.ac.id/ejournal-v2/index.php/ciastech/article/view/1949.

GLOSARIUM

Assistive technology	:	Istilah umum untuk sistem atau layanan untuk membantu manusia supaya tidak tergantung orang lain.
BISINDO	:	Bahasa Isyarat Indonesia, bahasa isyarat yang digunakan sebagian besar komunitas tunawicara dan tunarungu di Indonesia, bersifat lokal dan banyak digunakan di pergaulan non formal.
Feature selection	:	Salah satu tahapan untuk menentukan fitur-fitur yang akan digunakan sebagai input machine learning.
Gestur tangan	:	Gerakan atau simbol-simbol dengan tangan.
Leap motion	:	Alat pendeteksi gerakan tangan dengan menggunakan sejenis sensor magnetis, dihubungkan ke USB komputer, jarak leap motion dengan tangan kurang lebih 5-10 cm.
Library	:	Sekumpulan fungsi yang siap pakai pada suatu bahasa pemrograman. Umumnya disediakan untuk memudahkan pengembangan perangkat lunak oleh programmer.
Prevalensi	:	Jumlah keseluruhan kasus disabilitas pada waktu tertentu.
SIBI	:	Sistem Isyarat Bahasa Indonesia, bahasa isyarat untuk tunawicara dan tunarungu di Indonesia, bersifat nasional dan diakui pemerintah.
Voltage divider	:	Istilah untuk rangkaian pembagi tegangan, umumnya terdiri atas resistor-resistor.
Wearable device	:	Peralatan atau devais yang dipasang di tubuh manusia untuk keperluan khusus, misalnya mengukur suhu tubuh, mengukur gerakan tubuh, dan sebagainya; lebih diutamakan kepraktisannya.

INDEKS

T

U

V

W

BIOGRAFI PENULIS

Romy Budhi Widodo, adalah dosen program studi Teknik Informatika Universitas Ma Chung Malang sejak 2007 hingga sekarang. Pendiri Pusat Studi Human-Machine Interaction Universitas Ma Chung.

Romy B. Widodo mendalami *human-computer interaction*, *embedded system*, dan saat ini mengaplikasikan *machine learning* untuk *human-machine interaction*. Pendidikan terakhir tahun 2017 menamatkan Doktoral dari Kyushu Institute of Technology, Japan pada Graduate School of Life Science and Systems Engineering.