Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2023106196
Vol. 10, No. 2, April 2023, hlm. 245-254 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 38/E/KPT/2019 e-ISSN: 2528-6579

ANALISIS PERFORMA ALGORITMA MACHINE LEARNING PADA PERANGKAT EMBEDDED ATMEGA328P

Jeffry Atur Firdaus[1], Agung Setia Budi*[2], Eko Setiawan[3]

[1,2,3]Universitas Brawijaya, Malang
Email: [1]jeffryaf@student.ub.ac.id, [2]agungsetiabudi@ub.ac.id, [3]ekosetiawan@ub.ac.id
*Penulis Korespondensi

(Naskah masuk: 05 Maret 2022, diterima untuk diterbitkan: 10 April 2023)

Abstrak

Artificial intelligence (AI) merupakan sistem kompleks yang meniru kecerdasan manusia untuk melakukan tugas dan dapat mengembangkan kecerdasannya menggunakan informasi yang mereka kumpulkan. *Machine learning* yang merupakan bagian dari AI, sering ditemui pada perangkat *embedded*. Beberapa algoritma *machine learning* yang banyak dikembangkan pada perangkat *embedded* adalah *K-Nearest Neighbor, Naive Bayes* dan *SEFR*. Pada tahun 2019, evaluasi pasar *embedded system* mencapai $100 miliar dan akan diprediksi jumlahnya terus meningkat sebesar enam persen setiap tahunnya. Salah satu perangkat *embedded open source* yang sering ditemui di pasaran adalah Arduino Nano berbasis ATmega328P. Namun tidak seperti komputer, *embedded system* memiliki sumber daya yang terbatas. Dalam mengembangkan sistem pada perangkat *embedded* perlu diperhatikan faktor seperti waktu komputasi, daya yang diperlukan dan penggunaan memori. Penelitian ini mengkaji tiga algoritma tersebut untuk mencari algoritma yang paling sesuai di perangkat *embedded*. Penelitian ini menemukan bahwa dalam melakukan klasifikasi tiga *dataset,* algoritma *K-Nearest Neighbor* mendapatkan akurasi paling baik dan paling konsisten hingga 93.3% akurat. Penggunaan sumber daya SRAM paling sedikit didapatkan pada algoritma *Naive Bayes* dengan rata-rata 764 *Bytes*. Waktu komputasi paling cepat didapatkan oleh algoritma SEFR dengan waktu yang dibutuhkan untuk melakukan klasifikasi *dataset* dalam waktu rata-rata 1.16 mili sekon dan konsumsi daya 0.1 mili joule.

Kata kunci: *machine learning, embedded system, k-nearest neighbor, naive bayes, SEFR*

MACHINE LEARNING ALGORITHM PERFORMANCE ANALYSIS ON ATMEGA328P EMBEDDED DEVICES

Abstract

Artificial intelligence (AI) is a complex system that imitates human intelligence to perform tasks and can develop their intelligence using the information they collect. Machine learning, which is part of AI, is often encountered in embedded devices. Several machine learning algorithms that have been developed on embedded devices are K-Nearest Neighbor, Naive Bayes and SEFR. In 2019, the evaluation of the embedded systems market reached $100 billion and is predicted to continue to increase by six percent annually. One of the open source embedded devices that is often found in the market is the Arduino Nano based on the ATmega328P. However, unlike computers, embedded systems have limited resources. In developing systems on embedded devices, factors such as computing time, required power and memory usage must be considered. This study examines these three algorithms to find the most suitable for the embedded device. This study found that in classifying three datasets, the K-Nearest Neighbor algorithm got the best and most consistent accuracy up to 93.3% accurate. The least use of SRAM resources is found in the Naive Bayes algorithm with an average of 764 Bytes. The fastest computation time is obtained by the SEFR algorithm with the time required to classify the dataset in an average time of 1.16 milliseconds and a power consumption of 0.1 milli joules.

Keywords: *machine learning, embedded system, k-nearest neighbor, naive bayes, SEFR*

1. PENDAHULUAN

Artificial intelligence atau yang biasa disingkat AI merupakan sistem kompleks yang meniru kecerdasan manusia untuk melakukan tugas dan dapat mengembangkan kecerdasannya menggunakan informasi yang mereka kumpulkan (Russell & Norvig, 2010). Salah satu cabang pada *artificial intelligence* adalah *machine learning. machine*

learning merupakan fokus pada data dan algoritma supaya dapat berpikir dan belajar seperti manusia (IBM, 2020). Algoritma yang dikembangkan pada bidang *machine learning,* beberapa algoritma yang dikembangkan antara lain *K-Nearest Neighbor* (KNN)*, Naive Bayes* dan *Scalable, Efficient, and Fast classifieR* (SEFR)*.* Algoritma tersebut sering ditemui pada penelitian *embedded system* karena membutuhkan sumber daya yang minimum.

Embedded system merupakan sebuah sistem tertanam menggunakan mikroprosesor yang bertujuan untuk mengendalikan fungsi-fungsi dari sebuah sistem dengan tujuan yang spesifik (Heath, 2003). *Embedded system* terdiri dari prosesor untuk memproses data, memori untuk menyimpan data dan perangkat *input/output* untuk berinteraksi dengan *user* (Barr, 2007). Karena *embedded system* mempermudah user dalam menggunakan sebuah sistem atau perangkat, maka *embedded system* semakin banyak ditanamkan pada perangkat sehari-hari. Beberapa contoh penerapan *embedded system* dalam kehidupan sehari-hari terdapat pada mobil, kulkas, modern *rice cooker,* dan *smart lamp*. Pada beberapa kendaraan sudah dikembangkan teknologi kendaraan otonom atau yang biasa dikenal *autonomous vehicle* yang dapat mengendalikan sebuah kendaraan tanpa bantuan pengemudi dengan dukungan dari *embedded system* dan *artificial intelligence* (Marwedel, 2021). Perkembangan dalam bidang teknologi mendorong jumlah *embedded system* yang terus bertambah. Pada tahun 2019, evaluasi pasar *embedded system* mencapai $100 miliar dan akan diprediksi jumlahnya terus meningkat sebesar enam persen setiap tahunnya (Wadhwani, 2020). namun tidak seperti komputer, *embedded system* memiliki sumber daya yang terbatas. dalam mengembangkan sistem pada perangkat *embedded* perlu diperhatikan faktor seperti waktu komputasi dan daya yang diperlukan.

Seiring berkembangnya jumlah *embedded system,* maka penelitian pada bidang tersebut juga semakin populer. Dalam melakukan penelitian *embedded* ada banyak pilihan perangkat dengan harga yang terjangkau dan tersedia di pasaran. Saat ini salah satu perangkat yang populer digunakan untuk penelitian adalah Arduino (Vasquez, 2020). Arduino sendiri memiliki berbagai model namun prosesor yang paling sering ditemui adalah ATmega328P, mikroprosesor ini banyak digunakan pada penelitian karena kemudahan dalam penggunaan dan sistem yang fleksibel (Hadi, 2018)

Beberapa penelitian terkini pada bidang *embedded machine learning* antara lain "Sistem Pengukur Kesegaran Daging Sapi menggunakan Metode *K-Nearest Neighbor* (K-NN) dengan Fitur Penambahan Data Latih berbasis EEPROM". Sistem pada penelitian ini mendapatkan akurasi hingga 85% dengan waktu komputasi yang dibutuhkan 117 milisekon untuk menguji kesegaran daging sapi (Firdaus, 2020).

Pada penelitian dengan judul "Sistem Deteksi Titik Kebakaran dengan Algoritme *K-Nearest Neighbor* (KNN) menggunakan Sensor Suhu dan Sensor Api". Sistem melakukan prediksi letak sumber api menggunakan algoritma *K-Nearet Neighbor* dengan akurasi 80.55% dan waktu pengambilan data dari sensor hingga klasifikasi adalah 1428 mili sekon (Firdaus, 2019).

Penelitian pada algoritma SEFR digunakan artikel "SEFR: A Fast Linear-Time Classifier for Ultra-Low Power Devices" Dilakukan percobaan akurasi dan konsumsi energi pada algoritma SEFR pada *dataset* yang besar menggunakan komputer. Diketahui bahwa algoritma SEFR membutuhkan energi yang lebih rendah dibandingkan dengan algoritma lain dan memiliki akurasi yang tidak buruk (Keshavarz, 2020).

Penelitian pada algoritma *Naive Bayes* adalah "Implementasi Algoritme *Naive Bayes* Menggunakan Arduino Uno untuk Otomatisasi Lampu Ruangan Berdasarkan Kebiasaan dari Penghuni Rumah". Sistem pada penelitian ini mendapatkan akurasi sebesar 84,61% untuk orang pertama dan 92,30% pada orang kedua dengan waktu komputasi yang dibutuhkan 0,25 detik dari 13 kali pengujian (Pratama, 2018).

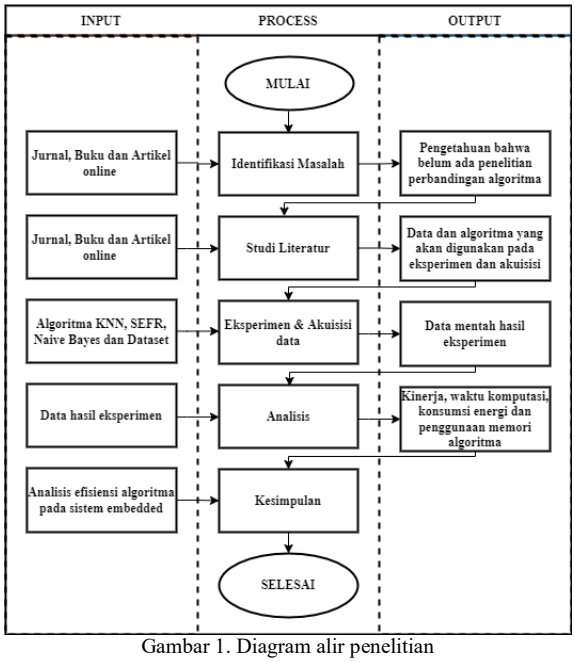
Pada penelitian dengan judul "Sistem Klasifikasi Rasa Buah Jeruk Menggunakan Metode Naïve Bayes Dengan Arduino Nano". Algoritma *naive bayes* dapat melakukan klasifikasi dengan akurasi hingga 80% dari 45 data uji dan 15 data latih (Sinaga, 2021).

Dari sekian banyak yang menggunakan *machine learning* pada perangkat *embedded*, masih belum diketahui algoritma mana yang paling optimum untuk perangkat *embedded* ATmega328P dengan sumber daya sistem yang terbatas. Penelitian ini akan membandingkan akurasi, penggunaan SRAM dan waktu komputasi pada beberapa algoritma *machine learning* untuk mengetahui algoritma yang paling efisien pada mikrokontroler ATmega328p.

2. METODE PENELITIAN

Pada bab ini akan dijelaskan mengenai metodologi atau uraian tentang langkah kerja yang dilakukan dalam penyusunan penelitian ini. Penelitian diawali dengan melakukan identifikasi masalah dengan bantuan dari jurnal, buku dan artikel daring. Identifikasi masalah menghasilkan pengetahuan bahwa belum ada penelitian yang pernah melakukan perbandingan algoritma pada *embedded system* ATmega328P. Selanjutnya dilakukan studi literatur dari jurnal, buku dan artikel daring untuk mempelajari data dan algoritma yang akan digunakan pada eksperimen dan akuisisi data. Selanjutnya dilakukan eksperimen dan akuisisi data terhadap algoritma KNN, SEFR dan *Naive Bayes* terhadap tiga *dataset*. Eksperimen dan akuisisi data menghasilkan data mentah dari penelitian. Kemudian data dari eksperimen dianalisis untuk mendapatkan

kinerja, waktu komputasi, konsumsi energi dan penggunaan memori pada algoritma kemudian ditarik kesimpulan dari penelitian ini. Diagram alir penelitian dapat diamati pada gambar 1.



Gambar 1. Diagram alir penelitian

3. LANDASAN KEPUSTAKAAN

Dalam bab ini akan dibahas mengenai landasan pustaka yang digunakan pada penyusunan penelitian ini. Landasan pustaka yang digunakan pada penelitian ini bersumber dari jurnal penelitian, laporan penelitian tugas akhir dan sumber lainnya yang dapat mendukung teori-teori yang digunakan dalam penyusunan penelitian ini. Teori-teori utama yang digunakan pada penyusunan penelitian ini meliputi *dataset*, ATmega328P, Algoritma *K-Nearest Neighbor,* Algoritma *Naive Bayes* dan Algoritma SEFR.

3.1. Dataset

Pada penelitian ini digunakan tiga jenis *dataset* untuk menguji algoritma *machine learning* pada perangkat ATmega328P. Pertama adalah Dataset Iris dari UCI *Machine Learning*, kedua *dataset* kesegaran daging sapi dari penelitian "Implementasi Sistem Penentuan Kesegaran Daging Sapi Lokal Berdasarkan Warna dan Kadar Amonia Dengan Metode Jaringan Saraf Tiruan Berbasis Embedded System" dan ketiga *dataset* kesegaran daging ikan gurami dari penelitian "Sistem Klasifikasi Kesegaran Daging Ikan Gurami berdasarkan Warna dan Gas Amonia menggunakan *K-Nearest Neighbor* (KNN) berbasis Arduino".

Dataset UCI Iris merupakan data dari karakteristik bunga iris yang dikembangkan oleh Fisher pada tahu 1936. *Dataset* ini memiliki tiga kelas yaitu *Iris Setosa, Iris Versicolour* dan *Iris Virginica. dataset* ini memiliki 150 *instances* dan 4 fitur. fitur-fitur pada *dataset* ini antara lain "*sepal length in cm", "sepal width in cm", "petal length in cm"* dan *"petal width in cm"* (Fisher, 1936).

Dataset daging sapi merupakan kumpulan data mengenai kesegaran daging sapi yang dikembangkan oleh Firmansyah pada tahun 2019. *Dataset* ini memiliki tiga kelas yaitu segar, sedang dan busuk. *Dataset* ini memiliki empat fitur yaitu intensitas warna *red, green, blue* dan kualitas udara (Firmansyah, 2019).

Dataset ikan gurami merupakan sapi merupakan kumpulan data mengenai kesegaran daging sapi yang dikembangkan oleh Hidayatullah pada tahun 2022. *Dataset* ini memiliki tiga kelas yaitu segar, sedang dan busuk. *Dataset* ini memiliki empat fitur yaitu intensitas warna *red, green, blue* dan intensitas gas NH3 (Hidayatulah, 2022).

Pada artikel ini dipilih menggunakan ketiga *dataset* tersebut karena jumlah fitur dan jumlah *instance* menyerupai penelitian sebelumnya, sehingga dapat dipastikan mikrokontroler ATmega328P dapat memproses *dataset* tersebut. Gambar bunga dari UCI Iris *dataset* dapat dilihat pada gambar 2.


Gambar 2. Bunga iris UCI Dataset

3.2. ATmega328P

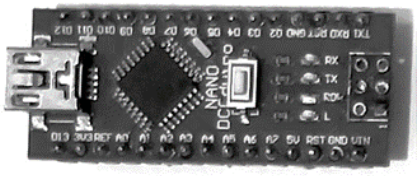
ATmega328P adalah mikrokontroler CMOS 8-bit berdaya rendah berdasarkan arsitektur RISC yang dikembangkan AVR. Dengan mengeksekusi instruksi yang kuat dalam satu siklus *clock*, ATmega328P mencapai *throughput* mendekati 1MIPS per MHz sehingga perancang sistem dapat mengoptimalkan konsumsi daya. Salah satu papan ATmega328P mudah didapatkan di pasaran, *low cost* dan *open source* adalah Arduino Nano.

Arduino Nano adalah papan elektronik *open source* yang di dalamnya terdapat komponen utama cip mikrokontroler Atmega328. Arduino Nano memiliki ukuran yang relatif lebih kecil serta harga lebih rendah dibandingkan dengan Arduino tipe lain. Arduino Nano memerlukan tegangan 5 volt untuk beroperasi. Arduino Nano dapat menggunakan USB sebagai sumber daya atau *pin* "+5V" dan "GND" dengan tegangan masuk sebesar 5 volt dari sumber daya eksternal (Arduino, 2008)

Papan Arduino pertama kali diperkenalkan pada tahun 2005 yang dihasilkan dari program pascasarjana "Interaction Design Institute Ivrea" di Ivrea, Italia. Papan elektronik ini menggunakan mikroprosesor Atmel dengan *interface* yang mudah

sehingga membantu peneliti dengan cepat mengembangkan proyek berbasis mikrokontroler.

Seiring berkembangnya jumlah *embedded system,* maka penelitian pada bidang tersebut juga semakin populer. Dalam melakukan penelitian *embedded* ada banyak pilihan perangkat dengan harga yang terjangkau dan tersedia di pasaran. Saat ini salah satu perangkat yang populer digunakan untuk penelitian adalah Arduino (Vasquez, 2020). Arduino sendiri memiliki berbagai model namun prosesor yang paling sering ditemui adalah ATmega328P, mikroprosesor ini banyak digunakan pada penelitian karena kemudahan dalam penggunaan dan sistem yang fleksibel (Hadi, 2018). Gambar Arduino Nano dapat dilihat pada gambar 3.



Gambar 3. Arduino Nano

3.3. Algoritma K-Nearest Neighbor

Pada tahun 1951, Fix dan Hodge mengembangkan sebuah metode pengenalan pola *non-parametric* yang saat ini kita kenal sebagai algoritma *k-Nearest Neighbor. K-Nearest Neighbor* adalah metode klasifikasi bersifat lazy learner dikarenakan algoritma ini menyimpan semua nilai data latih dan menunda proses pembentukan model klasifikasi sampai data uji diberikan untuk dilakukan prediksi (Mulak, & Talhar, 2015). Algoritma *K-Nearest Neighbor* biasanya menggunakan persamaan *euclidean distance* untuk mengukur jarak antar *neighbor*. Persamaan *euclidean distance* dapat diamati pada persamaan (1).

$$\delta(p,q) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} \quad (1)$$

Pada penelitian dengan judul "Sistem Pengukur Kesegaran Daging Sapi menggunakan Metode *K-Nearest Neighbor* (K-NN) dengan Fitur Penambahan Data Latih berbasis EEPROM". Dilakukan percobaan klasifikasi kesegaran daging sapi menggunakan mikrokontroler Arduino Nano ATmega328P. Penelitian ini menggunakan *dataset* daging sapi yang berdeskripsi 81 data latih, 27 data uji, 4 fitur dan 3 kelas. Sistem pada penelitian ini mendapatkan akurasi hingga 85% dengan waktu komputasi yang dibutuhkan 117 mili sekon untuk menguji kesegaran daging sapi (Firdaus, 2020).

Pada penelitian dengan judul "Sistem Deteksi Titik Kebakaran dengan Algoritme *K-Nearest Neighbor* (KNN) menggunakan Sensor Suhu dan Sensor Api". Dikembangkan sebuah sistem yang dapat mendeteksi titik kebakaran dengan sensor LM35 dan sensor *flame.* Sistem melakukan prediksi letak sumber api menggunakan algoritma *K-Nearet Neighbor* dengan akurasi 80.55% dan waktu pengambilan data dari sensor hingga klasifikasi adalah 1428 mili sekon (Firdaus, 2019). Kedua penelitian sudah menggunakan mikrokontroler namun belum melakukan percobaan pada beberapa algoritma *machine learning*.

3.4. Algoritma Naive Bayes

Naive bayes merupakan jenis algoritma dalam *machine learning* untuk melakukan klasifikasi. teknik klasifikasi *naive bayes* sangat cepat dan sederhana dalam menyelesaikan *dataset* multidimensi karena memiliki sedikit parameter yang dapat diubah. Klasifikasi *naive bayes* menggunakan teorema *bayes* yang merupakan persamaan hubungan probabilitas bersyarat dalam statistik dan persamaan distribusi normal (Zhang, 2004). Persamaan distribusi normal dapat diamati pada persamaan (2).

$$p(x = u|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{\frac{(u-\mu k)^2}{2\pi\sigma_k^2}} \quad (2)$$

Pada penelitian dengan judul "Implementasi Algoritme *Naive Bayes* Menggunakan Arduino Uno untuk Otomatisasi Lampu Ruangan Berdasarkan Kebiasaan dari Penghuni Rumah". Dilakukan percobaan pengklasifikasian waktu nyala atau mati lampu berdasarkan kebiasaan pengguna dalam menyalakan dan mematikan lampu menggunakan mikrokontroler Arduino UNO. penelitian ini menghasilkan otomatisasi waktu nyala dan mati pada lampu. Dengan 40 data latih dan 13 data uji, sistem pada penelitian ini mendapatkan akurasi sebesar 84,61% untuk orang pertama dan 92,30% pada orang kedua dengan waktu komputasi yang dibutuhkan 0,25 detik dari 13 kali pengujian (Ramadan, 2019). Namun penelitian ini belum melakukan pengujian penggunaan energi yang dibutuhkan sistem untuk melakukan klasifikasi.

Pada penelitian dengan judul "Sistem Klasifikasi Rasa Buah Jeruk Menggunakan Metode *Naïve Bayes* Dengan Arduino Nano". Dikembangkan sebuah sistem yang mengklasifikasi buah jeruk dengan rasa manis atau asam. Sistem ini menggunakan sensor warna RGB dan sensor berat *load cell.* Algoritma *naive bayes* dapat melakukan klasifikasi dengan akurasi hingga 80% dari 45 data uji dan 15 data latih (Sinaga, 2021). Penelitian ini sudah menggunakan mikrokontroler tetapi masih belum mengukur waktu komputasi yang dibutuhkan.

3.5. Algoritma SEFR

Scalable, Efficient, and Fast classifieR atau yang biasa dikenal dengan SEFR merupakan algoritma yang bekerja menyerupai SVM linier. SEFR didesain untuk menyelesaikan masalah biner (benar atau salah) namun dapat dikembangkan untuk menyelesaikan permasalahan *multiclass dataset* menggunakan model *hyperplane* seperti persamaan (3). Algoritma ini juga membutuhkan sumber daya energi dan memori yang sangat minimal (Keshavarz, 2020).

$$f(x) = w^t x + b \quad (3)$$

Pada tahap *training* algoritma SEFR dilakukan komputasi bobot (*weight*) dan bias masing-masing fitur dari data latih. Pada fitur *training* pertama dilakukan perhitungan jumlah sampel negatif dan sampel positif. Selanjutnya dihitung nilai rata-rata pada sampel positif dan sampel negatif. Dari keempat nilai tersebut dapat diukur *weight* dan bias dari data uji. Selanjutnya program akan membentuk *decision boundary* menggunakan rata-rata dari skor data uji.

Pada tahap klasifikasi dilakukan perhitungan skor data uji terhadap data latih mengalikan data uji dengan *weight* kemudian ditambah dengan *bias* seperti pada persamaan (4). Skor dengan nilai paling baik akan dipilih sebagai kelas prediksi..

$$score = example \times weight + bias \quad (4)$$

Pada penelitian dengan judul "SEFR: A Fast Linear-Time Classifier for Ultra-Low Power Devices" Dilakukan percobaan akurasi dan konsumsi energi pada algoritma SEFR pada *dataset* yang besar menggunakan komputer. Diketahui bahwa algoritma SEFR membutuhkan energi yang lebih rendah dibandingkan dengan algoritma lain dan memiliki akurasi yang tidak buruk (Keshavarz, 2020). Penelitian ini sudah membandingkan dengan beberapa algoritma lain namun belum diterapkan pada perangkat *embedded* ATmega328P.

4. PENGUJIAN

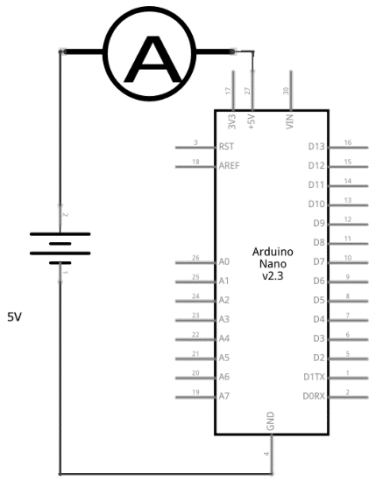
Bab pengujian menjelaskan langkah-langkah yang ditempuh dalam penelitian ini untuk mengimplementasikan masing-masing algoritma ke dalam perangkat *embedded*. Prinsip kerja algoritma di dalam perangkat *embedded* dan cara melakukan pengukuran konsumsi energi juga akan dijelaskan secara detail di bab ini. Kode dan *dataset* yang digunakan pada penelitian ini dapat di akses pada repositori berikut.

Pada tahap pengujian dilakukan proses mendesain kode k-NN, *naive bayes* dan SEFR pada Arduino Nano berbasis ATmega328p serta rangkaian sistem yang akan digunakan. Tahap eksperimen akan dibantu dengan perangkat lunak pada komputer yang dapat melakukan sketsa rangkaian elektronika secara digital serta aplikasi "Arduino IDE".

Tahap pertama pada pengujian adalah perancangan kode. Akan didesain kode k-NN, *naive bayes* dan SEFR pada Arduino Nano menggunakan bantuan aplikasi "Arduino IDE". Kode sistem akan didesain sesuai dengan kriteria penelitian. Kriteria-kriteria kode pada sistem yang diperlukan pada penelitian ini adalah sistem dapat melakukan pengukuran waktu komputasi algoritma *machine learning* pada Arduino Nano, sistem membaca *dataset*, dan sistem dapat menampilkan hasil dari klasifikasi algoritma *machine learning*.

Selanjutnya di desain rangkaian menggunakan perangkat lunak pada komputer yang dapat melakukan sketsa rangkaian elektronika secara digital. Ada dua jenis rangkaian yang akan didesain pada penelitian ini, yaitu rangkaian pengukur arus dan rangkaian pengukur tegangan. Rangkaian ini selanjutnya digunakan sebagai referensi perancangan sistem pengukuran arus dan tegangan sistem.

Pada rangkaian pengukur arus ada tiga komponen utama, yaitu sumber daya 5 volt, Arduino Nano dan amperemeter. Pada rangkaian pengukur arus, kutub positif pada sumber daya 5 volt dihubungkan pada *pin* "5v" di Arduino Nano dan kutub negatif pada sumber daya 5 volt dihubungkan pada *pin* "GND" di Arduino Nano. Selanjutnya komponen amperemeter akan dihubungkan secara seri pada salah satu jalur dari sumber daya 5 volt menuju Arduino Nano. Rangkaian pengukur arus dapat diamati pada gambar 4.



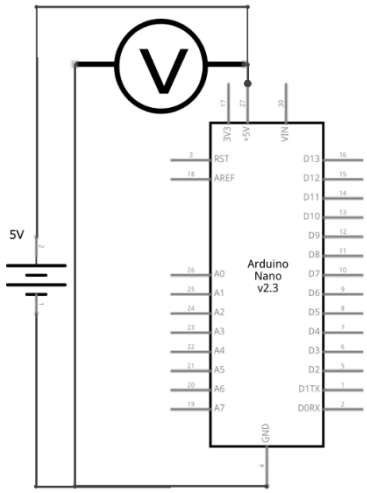
Gambar 4. Rangkaian pengukur arus

Pada rangkaian pengukur tegangan ada tiga komponen utama, yaitu sumber daya 5 volt, Arduino Nano dan voltmeter. Pada rangkaian pengukur tegangan, kutub positif pada sumber daya 5 volt dihubungkan pada *pin* "5v" di Arduino Nano dan kutub negatif pada sumber daya 5 volt dihubungkan pada *pin* "GND" di Arduino Nano. Selanjutnya komponen volt akan dihubungkan secara paralel pada kedua jalur dari sumber daya 5 volt menuju Arduino Nano. Rangkaian pengukur tegangan dapat diamati pada gambar 5.

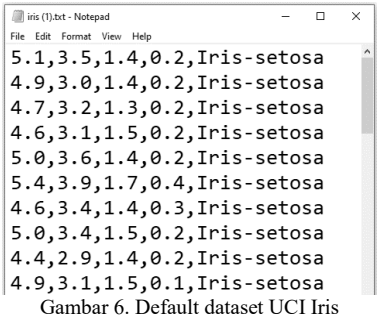
4.1 Persiapan Dataset

Sebelum dilakukan percobaan, perlu dilakukan *preprocessing* pada *dataset* supaya dapat diproses pada mikrokontroler. Pada dataset UCI Iris dilakukan konversi kelas *dataset* yang sebelumnya *string* "Iris-setosa", "Iris-versicolor" dan "Iris-virginica" menjadi angka "1", "2" dan "3". Selanjutnya *dataset* dibagi menjadi dua bagian yaitu data uji dan data latih. *Dataset* UCI Iris memiliki 150 *instances* dengan komposisi 50 *instances* bunga *Iris Setosa,* 50 *instances* bunga *Iris Versicolour* dan 50 *instances* bunga *Iris Virginica.* Dari 150 *instances,* dipilih 15 data uji secara acak dengan komposisi 5 *instances* bunga *Iris Setosa,* 5 *instances* bunga *Iris Versicolour*

dan 5 *instances* bunga *Iris Virginica* sehingga tersisa 135 data yang digunakan sebagai data latih. Tampilan *dataset* sebelum dilakukan *preprocessing* dapat diamati pada gambar 6.



Gambar 5. Rangkaian pengukur tegangan



Gambar 6. Default dataset UCI Iris

Pada *dataset* daging sapi dilakukan konversi kelas *dataset* yang sebelumnya *string* "segar", "sedang" dan "busuk" menjadi angka "1", "2" dan "3". Selanjutnya *dataset* dibagi menjadi dua bagian yaitu data uji dan data latih. *Dataset* daging sapi memiliki 108 *instances,* dipilih 27 data uji secara acak sehingga tersisa 81 *instances* yang digunakan sebagai data latih.

Pada *dataset* daging ikan gurami dilakukan konversi kelas *dataset* yang sebelumnya *string* "segar", "sedang" dan "busuk" menjadi angka "1", "2" dan "3". Selanjutnya *dataset* dibagi menjadi dua bagian yaitu data uji dan data latih. *Dataset* daging ikan gurami memiliki 100 *instances,* dipilih 20 data uji secara acak sehingga tersisa 80 *instances* yang digunakan sebagai data latih.

4.2. Akuisisi Data

Perangkat keras yang sebelumnya di desain pada komputer secara digital di implementasikan dengan komponen utama Arduino Nano, Power Supply dengan tegangan 5 volt dan multimeter digital sebagai alat ukur. Selain itu terdapat komponen pendukung seperti kabel dan timah untuk menghubungkan antara komponen utama. Dari perangkat keras yang telah dirancang, tegangan dan arus sistem diukur. Tegangan dan arus kemudian dikalikan dengan waktu komputasi algoritma untuk mengetahui konsumsi energi yang dibutuhkan oleh sistem.

Pada tahap akuisisi data dilakukan pengambilan data-data yang diperlukan dalam melakukan penelitian. Data-data yang diperlukan antara lain akurasi sistem, waktu komputasi, konsumsi energi dan penggunaan memori. kode dan rangkaian yang sebelumnya telah di desain pada tahap perancangan diterapkan pada perangkat keras. Pada kode program, bagian pertama dengan kutipan kode "waktu = millis();" berfungsi untuk mengukur waktu yang dibutuhkan sistem untuk melakukan klasifikasi. Kemudian hasil klasifikasi ditampilkan pada Arduino IDE melalui fitur "*Serial.monitor*". Hasil dari klasifikasi data uji kemudian dibandingkan dengan data uji pada aplikasi *spreadsheet* untuk mengetahui akurasi dari algoritma *machine learning.* Konsumsi SRAM dapat diamati pada informasi yang tampil pada *console* Arduino IDE.

4.3. K-Nearest Neighbor

Pada pengujian algoritma *K-Nearest Neighbor,* pertama dilakukan penyimpanan data uji pada *array* berjenis *float* yang dapat menyimpan bilangan pecahan. Kemudian kode dilanjutkan dengan tahap pertama klasifikasi yaitu menghitung *euclidean distance* array data uji terhadap seluruh data latih pada EEPROM. Setelah diketahui nilai semua *euclidean distance*, kemudian diurutkan dari nilai terkecil hingga terbesar menggunakan *bubble sort.* Dari hasil *bubble sort* kemudian dipilih tiga nilai *euclidean* terkecil. Dari tiga nilai *euclidean* terkecil kemudian dipilih kelas yang paling dominan sebagai penentuan hasil klasifikasi *K-Nearest Neighbor.*

Pada algoritma *K-Nearest Neighbor*, pengujian waktu komputasi hanya dilakukan pada proses klasifikasi, karena algoritma *K-Nearest Neighbor* tidak memerlukan proses *training*.

4.4. Naive Bayes

Pada pengujian algoritma *naive bayes* terdapat dua tahap yaitu *training* dan *testing*. Tahap *training* dilakukan pada ATmega328P. Pada tahap *training* dibentuk *float* sebanyak 24 variabel. 24 variabel *float* ini digunakan untuk menyimpan 12 nilai rata-rata dan 12 standar deviasi dari masing-masing 4 fitur dan 3 kelas. Penghitungan rata-rata rata data latih digunakan fitur *loop* dari mikrokontroler untuk menjumlahkan nilai masing-masing fitur kelas. Setelah diketahui rata-rata masing-masing fitur kelas, dilanjutkan dengan menghitung standar deviasi dari fitur kelas. Persamaan matematis standar deviasi dapat diamati pada persamaan (5). Setelah proses *training* selesai, dilanjutkan dengan fitur "millis()" untuk mengetahui waktu yang diperlakukan dalam proses *training*.

$$s = \sqrt{\frac{\sum(X-\bar{X})^2}{n-1}} \quad (5)$$

Pada tahap testing, pertama dilakukan penyimpanan data uji pada *array* berjenis *float* yang dapat menyimpan bilangan pecahan. Kemudian dihitung probabilitas data uji terhadap data latih menggunakan rumus persamaan distribusi normal atau yang biasa dikenal dengan distribusi *gauss*. Dari tiga hasil kalkulasi distribusi *gauss* kemudian dipilih dengan kelas probabilitas paling tinggi sebagai hasil prediksi kelas.

4.5. SEFR

Pada algoritma *Scalable, Efficient, and Fast classifieR* atau yang biasa dikenal dengan SEFR. Pertama dilakukan inisialisasi matriks sebanyak data latih dikalikan dengan jumlah fitur untuk menyimpan fitur data latih dan *array* berukuran sebanyak data latih untuk menyimpan kelas data latih. Data uji dimasukkan ke dalam kode secara manual dengan mengetik ke dalam kode untuk mengubahnya. Pada kode SEFR dibentuk dua fitur utama yaitu fitur untuk melakukan *training* data latih dan fitur untuk melakukan klasifikasi data uji.

Pada fungsi *training* dilakukan komputasi bobot (*weight*) dan bias masing-masing fitur dari data latih. Pada fitur *training* pertama dilakukan perhitungan jumlah sampel negatif dan sampel positif. Selanjutnya dihitung nilai rata-rata pada sampel positif dan sampel negatif. Dari keempat nilai tersebut dapat diukur *weight* dan bias dari data uji. Selanjutnya program akan membentuk *decision boundary* menggunakan rata-rata dari skor data uji.

Pada fungsi klasifikasi dilakukan perhitungan skor data uji terhadap data latih menggunakan persamaan (6). Skor dengan nilai paling baik akan dipilih sebagai kelas prediksi. Algoritma SEFR hanya dapat memprediksi benar atau salah saja, karena dataset ketiga dataset memiliki tiga kelas, maka proses klasifikasi dilakukan secara berikut.

- Kelas A atau (Kelas B dan C)
- Kelas B atau (Kelas A dan C)
- Kelas C atau (Kelas A dan B)

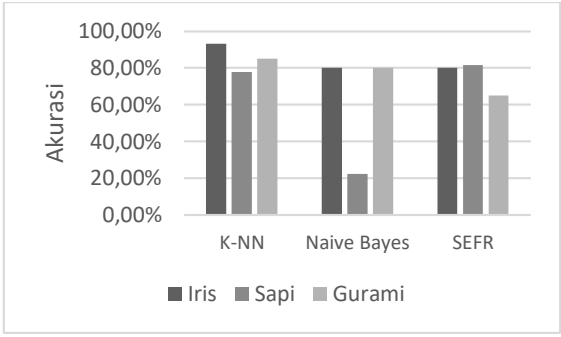
$$score = example \times weight + bias \quad (6)$$

5. HASIL DAN ANALISIS

Pada hasil dan analisis penelitian ini, dilakukan pengumpulan data dari hasil pengujian algoritma *machine learning*. Data yang dikumpulkan adalah akurasi klasifikasi, SRAM yang dibutuhkan dalam klasifikasi dan waktu komputasi yang dibutuhkan untuk melakukan *training* dan prediksi pada ketiga *dataset*.

5.1. Akurasi

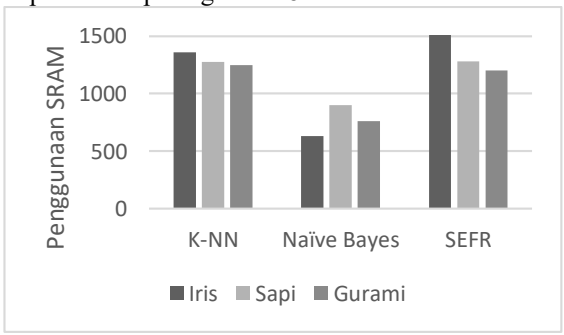
Akurasi merupakan persentase seberapa besar sistem dapat memprediksi kebenaran hasil klasifikasi. Hasil dari pengujian algoritma *machine learning* pada *dataset* UCI Iris, diketahui bahwa akurasi terbaik didapatkan oleh algoritma *K-Nearest Neighbor* dengan akurasi hingga 93,3% dan rata-rata 85%. Dilanjutkan dengan SEFR dengan rata-rata akurasi 80% dan terakhir *Naive Bayes* dengan rata-rata 61%. Pada *dataset* Iris dan Gurami, *Naive Bayes* dapat melakukan prediksi dengan akurasi 80%. Namun pada kasus tertentu, algoritma *Naive Bayes* mendapatkan akurasi yang buruk. Akurasi yang buruk disebabkan karena sifat dari *algoritma Naive Bayes* yang menganggap tiap fitur bersifat independen (tidak berhubungan satu sama lain). Perbandingan akurasi algoritma *machine learning* dapat dilihat pada gambar 7.



Gambar 7. Akurasi *machine learnig*

5.2. Pengunaan SRAM

SRAM merupakan singkatan dari *Static random-access memory*. SRAM digunakan mikrokontroler untuk menyimpan variabel-variabel yang akan dimanipulasi nilainya. ATmega328P memiliki SRAM sebesar 2048 *bytes*. Hasil dari pengujian algoritma *machine learning* pada *dataset* UCI Iris diketahui bahwa penggunaan SRAM paling sedikit didapatkan oleh algoritma *Naive Bayes* dengan rata-rata SRAM yang dibutuhkan 764 *bytes*. Dilanjutkan dengan *K-Nearest Neighbor* yang membutuhkan rata-rata SRAM sebesar 1296 *bytes* dan SEFR membutuhkan rata-rata SRAM sebesar 1380 *bytes*. Konsumsi SRAM pada algoritma K-NN dan SEFR paling dipengaruhi oleh banyaknya data latih di sistem (Iris 135, Sapi 81 dan Gurami 80), sedangkan pada algoritma Naive Bayes Konsumsi SRAM paling dipengaruhi oleh banyaknya data uji pada sistem (Iris 15, Sapi 27 dan Gurami 20). Perbandingan SRAM algoritma *machine learning* dapat dilihat pada gambar 8.

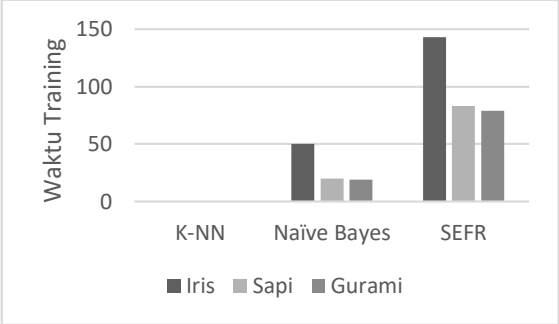


Gambar 8. Penggunaan SRAM *machine learning*

5.3. Waktu Komputasi

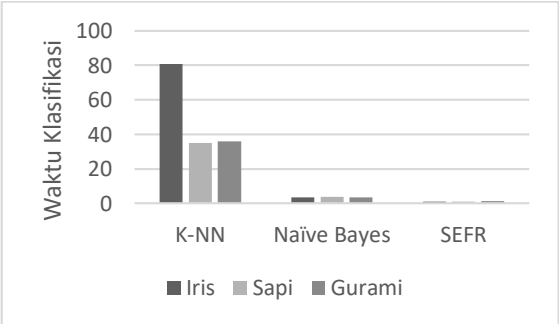
Waktu komputasi merupakan waktu yang dibutuhkan sistem untuk menjalankan sebuah program. Waktu komputasi pada pengujian dibagi menjadi dua, yaitu waktu *training* dan waktu klasifikasi.

Sebelum melakukan klasifikasi perlu dilakukan proses *training* untuk membentuk model matematika data *training*. Algoritma *K-Nearest Neighbor* tidak membutuhkan proses *training* karena pembentukan model matematika dilakukan pada proses klasifikasi, sehingga waktu *training* yang diperlukan adalah nol mili sekon. Rata-rata waktu *training* tercepat kedua dilakukan oleh algoritma *Naive Bayes* dengan 24 mili sekon dan waktu *training* tercepat ketiga dilakukan oleh algoritma SEFR dengan 102 mili sekon. Perbandingan waktu *training* algoritma *machine learning* dapat diamati pada gambar 9.



Gambar 9. Waktu *training machine learning*

Hasil dari pengujian algoritma *machine learning* pada *dataset* UCI Iris, diketahui bahwa rata-rata penggunaan waktu komputasi paling sedikit didapatkan oleh algoritma SEFR dengan waktu komputasi satu mili sekon. Dilanjutkan dengan *Naive bayes* yang membutuhkan waktu komputasi selama 3.4 mili sekon dan *K-Nearest Neighbor* yang membutuhkan waktu komputasi selama 51 mili sekon. Perbandingan waktu klasifikasi algoritma *machine learning* dapat diamati pada gambar 10.



Gambar 10. Waktu klasifikasi *machine learning*

Dengan diketahui spesifikasi kebutuhan ATmega328P sebesar tegangan 5 volt, arus 18.5 mili ampere dan waktu komputasi masing-masing algoritma, dapat dilakukan kalkulasi energi yang dibutuhkan masing-masing algoritma menggunakan persamaan (7).

$$energy(j) = power(w)\ x\ time(s) \quad (7)$$

5.4. Analisis Kompleksitas Waktu

Algoritma *K-Nearest Neighbor* tidak membutuhkan proses *training*, maka kompleksitasnya adalah Θ(1).

Proses klasifikasi algoritma *K-Nearest Neighbor* lebih lama dibandingkan dua algoritma lainnya karena prosesnya lebih kompleks. Klasifikasi diawali pada bagian *loop* "while (datake < X)", merupakan baris yang digunakan untuk mengukur *euclidean distance* data latih terhadap seluruh data uji. Karena baris ini melakukan perulangan sebanyak data latih, maka bagian kode ini memiliki kompleksitas Θ(n) dimana n merupakan jumlah data latih.

Kode dilanjutkan dengan mengurutkan (*sorting*) *euclidean distance* terkecil menuju terbesar. Proses mengurutkan data dilakukan menggunakan algoritma *bubble sort.* Karena *bubble sort* merupakan *nested loop,* maka *bubble sort* memiliki kompleksitas $\Theta(n^2)$ dimana n merupakan jumlah data latih. Maka klasifikasi algoritma *K-Nearest Neighbor* memiliki konstanta (Θn) + $\Theta(n^2)$, sehingga kompleksitasnya adalah $\Theta(n^2)$.

Pada baris *training* algoritma *Naive Bayes,* terdapat fungsi untuk menghitung rata-rata dan fungsi menghitung standar deviasi. Dalam mengukur nilai rata-rata dan standar deviasi, diperlukan *loop* sebanyak data latih (n). Lalu komputasi ini dilakukan sebanyak fitur yang ada pada *dataset* sistem (m). Sehingga kompleksitasnya dapat ditulis dalam bentuk Θ(mn)

Pada baris kode klasifikasi algoritma *Naive Bayes* peneliti tidak ada *loop*, hanya rumus matematis untuk menghitung probabilitas distribusi normal. Namun secara teori penghitungan distribusi normal dilakukan sebanyak fitur (m) dikalikan dengan banyaknya kelas (c) dataset, sehingga kompleksitasnya dapat ditulis dengan Θ(mc).

Pada baris *training* algoritma SEFR dilakukan pada fungsi "fit()" dua *loop* yaitu, sebuah tiga *nested loop* yang dijalankan sebanyak kelas(c)*fitur(m)*dataLatih(n) dan sebuah *loop* sebanyak data latih untuk mengukur *weighted score.* Maka kompleksitas pada baris kode ini adalah Θ(mnc).

Pada baris klasifikasi algoritma SEFR dilakukan komputasi skor untuk menghitung probabilitas masing-masing kelas sebanyak kelas(c)*fitur(m) dengan bentuk *nested loop.* Kompleksitas pada baris kode ini adalah Θ(mc). Secara praktik SEFR dapat melakukan klasifikasi lebih cepat daripada *Naive Bayes* karena klasifikasi pada *Naive Bayes* menggunakan persamaan distribusi normal yang lebih kompleks dibandingkan penjumlahan pada SEFR. Hasil kompleksitas waktu komputasi dapat diamati pada tabel 1.

Tabel 1. Kompleksitas Waktu

Order Of Growth	K-NN	Naive Bayes	SEFR
Training	Θ(1)	Θ(mn)	Θ(mnc)
Klasifikasi	$\Theta(n^2)$	Θ(mc)	Θ(mc)

6. KESIMPULAN DAN SARAN

Berdasarkan pada bab hasil dan pembahasan, penerapan algoritma *machine learning* yang berbeda pada *embedded system* memiliki kelebihan dan kekurangannya masing-masing. Ketika dibutuhkan sistem dengan akurasi paling baik pada perangkat *embedded system*, maka sistem dapat menggunakan algoritma *machine learning K-Nearest Neighbor.* Karena akurasinya paling baik dalam melakukan klasifikasi *dataset* hingga 93% akurat dan hasilnya lebih konsisten dari salah satu algoritma *machine learning* yang diuji. Selain akurat dan konsisten, algoritma *K-Nearest Neighbor* juga dapat menambah dan mengurangi data latih tanpa melakukan proses *training* terlebih dahulu. Ketika dibutuhkan pemrosesan *dataset* yang besar dan kompleks, maka sistem dapat menggunakan algoritma *machine learning Naive Bayes. Naive Bayes* membutuhkan SRAM paling sedikit dibandingkan algoritma lain sehingga dapat memproses *dataset* yang lebih besar. Ketika dibutuhkan sistem dengan konsumsi daya paling sedikit, maka sistem dapat menggunakan algoritma *machine learning* SEFR. Algoritma SEFR hanya membutuhkan 0.1 mili joule untuk melakukan satu kali klasifikasi. Sistem baterai 9v (1000 Joule) dapat melakukan hingga 10 juta kali klasifikasi pada algoritma SEFR. Hasil dari analisis dapat diamati pada Tabel 2.

Tabel 2. Hasil Analisis

Rata-rata	K-NN	Naive Bayes	SEFR
Akurasi	85%	61%	75%
Waktu *Training*	0 ms	24 ms	102 ms
Waktu Klasifikasi	51 ms	9 ms	1.16 ms
Konsumsi SRAM	1296 B	764 B	1380 B
Tanpa *Training*	Ya	Tidak	Tidak
Konsumsi Energi	5.1 mJ	0.8 mJ	0.1 mJ
Daya tahan (1000 J)	196.078	1.250.000	10.000.000

DAFTAR PUSTAKA

ARDUINO, 2008. Arduino Nano (V2.3) User Manual. [online] Tersedia di: < https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf> [Diakses 29 Mei 2021]

BARR, M. 2007. Embedded Systems Glossary. Neutrino Technical Library. [online] Tersedia di: < https://barrgroup.com/Embedded-Systems/Glossary> [Diakses 10 Januari 2022]

BELLAL, F., ELGHAZEL, H. AND AUSSEM, A., 2012. A semi-supervised feature ranking method with ensemble learning. Pattern Recognition Letters, 33(10), pp.1426-1433.

FIRDAUS, A., SYAUQY, D., & MAULANA, R. 2019. Sistem Deteksi Titik Kebakaran dengan Algoritme K-Nearest Neighbor (KNN) menggunakan Sensor Suhu dan Sensor Api. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 3(9), 8656-8663.

FIRDAUS, J., SETIAWAN, E., & SYAUQY, D. 2020. Sistem Pengukur Kesegaran Daging Sapi menggunakan Metode K-Nearest Neighbor (K-NN) dengan Fitur Penambahan Data Latih berbasis EEPROM. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 4(5), 1555-1562.

FISHER, R.A., MARSHALL, M., 1936, Iris Data Set. [online] Tersedia di: < https://archive.ics.uci.edu/ml/datasets/iris> [Diakses 12 Januari 2022]

FIRMANSYAH, H., SYAUQY, D., & ICHSAN, M. 2019. Implementasi Sistem Penentuan Kesegaran Daging Sapi Lokal Berdasarkan Warna dan Kadar Amonia Dengan Metode Jaringan Saraf Tiruan Berbasis Embedded System. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 3(4), 3955-3962.

HADI, M.S., AFANDI, A.N., WIBAWA, A.P., AHMAR, A.S., SAPUTRA, K.H., 2018. Stand-alone data logger for solar panel energy system with RTC and SD card. Journal of Physics Conference Series, 1028

HEATH, S., 2003. Embedded systems design second edition. [online] Tersedia di: < https://archive.org/details/embeddedsystemsd0000heat/page/n3/mode/2up?q=embedded+system> [Diakses 13 Januari 2022]

HIDAYATULAH, M., FITRIYAH, H., & UTAMININGRUM, F. 2022. Sistem Klasifikasi Kesegaran Daging Ikan Gurami berdasarkan Warna dan Gas Amonia menggunakan K-Nearest Neighbor (KNN) berbasis Arduino. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 6(2), 824-829.

IBM, 2020. Machine Learning. IBM Cloud Education. [online] Tersedia di: < https://www.ibm.com/cloud/learn/machine-learning> [Diakses 10 Januari 2022]

KESHAVARZ, H., ABADEH, M.S., RAWASSIZADEH, R., 2020. SEFR: A Fast Linear-Time Classifier for Ultra-Low Power Devices. arXiv preprint arXiv:2006.04620.

MARWEDEL, P., 2021. Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things (p. 433). Springer Nature.

MULAK, P. & TALHAR, N. 2015. Analysis of Distance Measures Using K-Nearest Neighbor Algorithm on KDD Dataset. International Journal of Science and Research (IJSR) 4(7) : 2101-2104.

PRATAMA, S., KURNIAWAN, W., & FITRIYAH, H, 2018. Implementasi Algoritme Naive Bayes Menggunakan Arduino Uno untuk Otomatisasi Lampu Ruangan Berdasarkan Kebiasaan dari Penghuni Rumah. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 2(9), 2485-2490.

RUSSEL, S.J., NORVIG, P., 2010. Artificial Intelligence A Modern Approach Third Edition 2010.

SINAGA, V., SETIAWAN, E., & HANAFI ICHSAN, M. 2021. Sistem Klasifikasi Rasa Buah Jeruk Menggunakan Metode Naive Bayes Dengan Arduino Nano. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 5(5), 1853-1859.

VASQUEZ, R.K.A., COLE, A.M., YORDANOVA, D., SMITH, R., KIDWELL, N.M., 2020. AIRduino: On-demand atmospheric secondary organic aerosol measurements with a mobile arduino multisensor. Journal of Chemical Education 2020, 97 (3).

WADHWANI, P., YADAV S., 2020. Embedded Systems Market Size By Component. [online] Tersedia di: <https://www.gminsights.com/industry-analysis/embedded-system-market> [Diakses 30 Mei 2021]

ZHANG, H., 2004. The Optimality of Naive Bayes Aa, 1(2), p.3.