JNATIA Volume 3, Nomor 3, Mei 2025 p-ISSN: 2986-3929
Jurnal Nasional Teknologi Informasi dan Aplikasinya e-ISSN: 3032-1948

Implementasi Particle Swarm Optimization pada Sistem Rekomendasi Tanaman Hortikultura Berbasis Naïve Bayes

Kadek Belvanatha Gargita Satwikananda**,** I Gede Surya Rahayuda

Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Udayana
Jalan Raya Kampus UNUD, Bukit Jimbaran, Kuta Selatan, Badung, Bali, Indonesia
satwikananda.2208561048@student.unud.ac.id
igedesuryarahayuda@unud.ac.id

Abstract

Indonesia is an agrarian country with an environment that strongly supports agricultural processes. Because of this, crops are one of Indonesia's main commodities and choosing a right crop to cultivate becomes a crucial process. Nowadays, lots of recommendation system has been made to help with decision making. A crop recommendation system is one of them and prove to be helpful in helping farmers decide what crop to plant based on the condition of the environment. An idea of implementing particle swarm optimization on a crop recommendation system occurred. Particle swarm optimization can be implemented to choose the presumably best parameter for a Gaussian Naïve Bayes model. The result of implementing particle swarming optimization to find the best smoothing parameter is the accuracy of the model reaching 99.5%. However, this is not different to the result of the model without particle swarm optimization which reach the accuracy of around 99.5% too.

Keywords: *Recommendation System, Naïve Bayes, Particle Swarm Optimization*

1. Pendahuluan

Indonesia merupakan negara agrikultur dengan sumber daya yang melimpah. Karakteristik lingkungan di Indonesia sangat mendukung proses agrikultur seperti pertanian. Dengan begitu salah satu komoditas utama Indonesia adalah tanaman hortikultura yang merupakan tanaman hasil panen dari proses pertanian. Pemilihan tanaman hortikultura yang tepat dapat meningkatkan efisiensi dari proses agrikultur. Sebuah sistem rekomendasi dapat dibangun untuk membantu proses pemilihan jenis tanaman yang tepat berdasarkan kondisi lingkungan. Ada berbagai algoritma yang dapat digunakan untuk membangun sistem rekomendasi ini dan salah satunya adalah algoritma Naïve Bayes. Pada proses klasifikasinya, algoritma Naïve Bayes bekerja dengan asumsi bahwa keberadaan suatu fitur dalam suatu kelas tidak dipengaruhi oleh keberadaan fitur lainnya. Sebuah pengklasifikasi Naive Bayes mengasumsikan bahwa keberadaan suatu elemen tertentu dalam suatu kelas sama sekali tidak terkait dengan keberadaan elemen lainnya [1]. Dengan begitu, membangun sistem rekomendasi terutama menggunakan dataset dengan data yang memiliki fitur – fitur yang tidak terikat satu sama lain akan sangat mungkin dilakukan menggunakan algoritma Naïve Bayes.

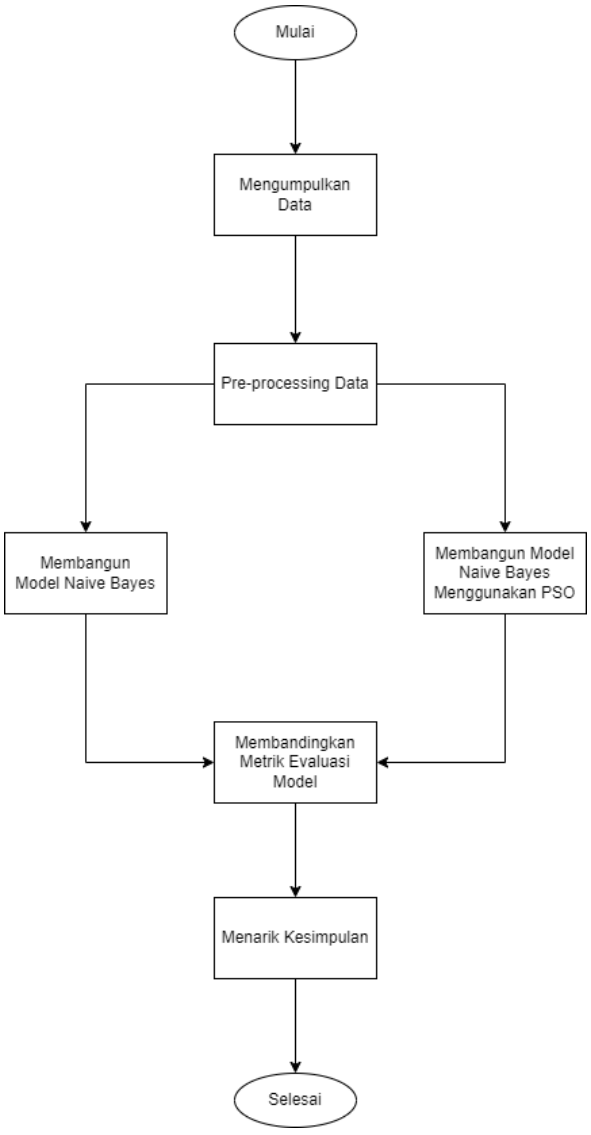
Sistem rekomendasi tanaman hortikultura menggunakan metode Naïve Bayes sesungguhnya sudah pernah dibangun dan menghasilkan akurasi yang cukup baik yaitu sebesar 85.71% pada penelitian [2]. Dengan melihat hasil ini, penulis ingin melihat pengaruh penerapan Particle Swarm Optimization (PSO) pada sistem rekomendasi berbasis Naïve Bayes untuk kasus rekomendasi tanaman hortikultura. PSO merupakan salah satu metode metaheuristik untuk menyelesaikan masalah optimasi yang terinspirasi oleh perilaku sosial dan dinamika gerakan dari gerombolan hewan seperti ikan atau burung [3]. Beberapa penelitian sebelumnya juga sudah mencoba mengaplikasikan PSO ke sistem berbasis Naïve Bayes. Pada penelitian [4] PSO digunakan untuk

mengoptimasi Naïve Bayes untuk menentukan kelayakan kredit Bumdes. Sementara pada penelitian [5] PSO digunakan untuk mengoptimasi seleksi fitur pada algoritma Naïve Bayes.

Pada penelitian ini penulis akan menggunakan PSO untuk mencari nilai parameter yang digunakan oleh algoritma Naïve Bayes. Diharapkan nantinya dengan menggunakan metode PSO ini sistem rekomendasi yang dibangun akan memiliki perbedaan yang cukup signifikan ke arah yang lebih baik sehingga dapat membantu pegiat agrikultur dalam memilih tanaman hortikultura yang tepat sesuai kondisi lingkungannya.

2. Metode Penelitian

Adapun tahapan yang dilakukan penulis dalam melakukan penelitian ini ditunjukkan oleh diagram alur pada gambar 1.



Gambar 1. Diagram Alur Tahapan Panelitian

Seperti yang ditunjukkan pada gambar satu, proses penelitian dimulai dengan pengumpulan data untuk melatih model. Setelah dataset berhasil didapatkan, kemudian akan dilakukan proses pre-processing data supaya data dapat digunakan untuk melatih model. Setelah data bersih dan dapat digunakan untuk melatih model barulah dilakukan proses membangun model. Proses

p-ISSN: 2986-3929
e-ISSN: 3032-1948

membangun model ini dibagi menjadi dua yaitu membangun model Naïve Bayes tanpa menerapkan PSO dan membangun model Naïve Bayes dengan mengimplementasikan PSO untuk memilih parameternya. Setelah model selesai dibangun, kemudian metrik evaluasi kedua model dibandingkan untuk melihat perbedaannya dan akhirnya kesimpulan pun dapat ditarik dari penelitian ini.

2.1. Data

Data yang digunakan dalam penelitian ini adalah data rekomendasi tanaman hortikultura tersedia di situs Kaggle pada tautan:

https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset/data.

Dataset merupakan sebuah *file* dengan format .csv yang berisi sebuah tabel dengan 8 kolom dan 2201 baris. Terdapat 7 kolom yang merupakan fitur dan 1 kolom yang merupakan label dengan 2200 baris data. Terdapat 22 label unik pada dataset ini dengan persebaran yang rata yaitu 100 data untuk tiap labelnya. Sampel dari dataset yang penulis gunakan dapat dilihat pada tabel 1.

Tabel 1. Sampel Data

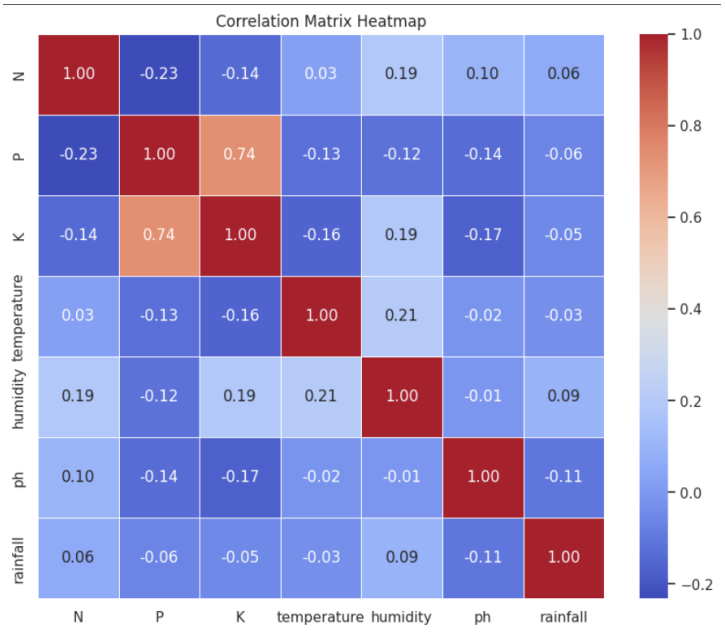
N	P	K	temperature	humidity	ph	rainfall	label
90	42	43	20.87974	82.00274	6.502985	202.9355	rice
85	58	41	21.77046	80.31964	7.038096	226.6555	rice
60	55	44	23.00446	82.32076	7.840207	263.9642	rice
74	35	40	26.4911	80.15836	6.980401	242.864	rice
78	42	42	20.13017	81.60487	7.628473	262.7173	rice
69	37	42	23.05805	83.37012	7.073454	251.055	rice
69	55	38	22.70884	82.63941	5.700806	271.3249	rice
94	53	40	20.27774	82.89409	5.718627	241.9742	rice
89	54	38	24.51588	83.53522	6.685346	230.4462	rice
68	58	38	23.22397	83.03323	6.336254	221.2092	rice
91	53	40	26.52724	81.41754	5.386168	264.6149	rice
90	46	42	23.97898	81.45062	7.502834	250.0832	rice
78	58	44	26.8008	80.88685	5.108682	284.4365	rice
93	56	36	24.01498	82.05687	6.984354	185.2773	rice
94	50	37	25.66585	80.66385	6.94802	209.587	rice

Setelah data didapatkan, penulis melakukan *exploratory data analysis* dan melihat bagaimana kondisi data. Penulis menggunakan bahasa pemrograman python untuk membuat beberapa grafik dan diagram. Tapi sebelumnya dilakukan pemisahan antara fitur dan label dari tabel pada dataset. Kolom N, P, K, temperature, kelembaban, ph, dan curah hujan akan digunakan sebagai fitur untuk menentukan rekomendasi tanaman sementara pada kolom label akan dijadikan target rekomendasi dari model yang dibangun.

Untuk melihat mengenai independensi fitur pada dataset ini, penulis juga membuat matriks korelasi. Pada matriks korelasi dengan bentuk *heatmap*, nilai 1 akan menandakan fitur memiliki korelasi yang berbanding lurus sementara -1 akan menandakan fitur memiliki korelasi yang berbanding terbalik. Dapat dilihat pada gambar 2 bahwa hanya fitur P dan K yang memiliki

p-ISSN: 2986-3929
e-ISSN: 3032-1948

korelasi yang signifikan dengan nilai mendekati 1 sementara fitur lainnya memiliki Tingkat korelasi yang rendah.



Gambar 2. Matriks Korelasi Fitur

Setelah itu dilakukan juga proses pre-processing data yang merupakan salah satu proses yang penting dalam membangun sebuah model *machine learning*. Proses ini bertujuan untuk mengubah data mentah menjadi data yang berkualitas dan dapat diolah lebih lanjut [5]. Pada penelitian ini fase pre-processing tidak begitu banyak dilakukan karena data yang sudah bersih dan data sangat cocok untuk model berbasis Naïve Bayes. Pada dataset ini tidak terdapat data yang kosong atau bernilai null. Kemudian setiap fitur juga tidak memiliki ketergantungan yang tinggi terhadap satu sama lain sehingga sangat cocok digunakan untuk model berbasis Naïve Bayes.

2.2. Model Naïve Bayes

Naïve Bayes dikembangkan oleh seorang fisikawan inggris bernama Thomas Bayes. Teorema Bayes merupakan algoritma yang memprakirakan hasil dari sebuah kemungkinan berdasarkan dari performa di masa lalu. Teorema ini kemudian digabungkan dengan Naïve, yang mengasumsikan bahwa setiap kriteria yang mengatur karakteristik bersifat independen dan tidak bergantung satu sama lain. Dengan begitu, pada metode Naïve Bayes dibuat asumsi bahwa keberadaan atau sebaliknya, tidak adanya sebuah fitur kelas tidak akan mempengaruhi ciri dari kelas lainnya. Berdasarkan uraian ini juga dapat disimpulkan bahwa pada dataset dengan fitur yang tidak terlalu bergantung satu sama lain, metode Naïve Bayes akan sangat efektif. Adapun persamaan dari teorema Bayes adalah seperti pada persamaan 1.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \quad (1)$$

Variabel E merupakan data dengan kelas yang belum diketahui, yang ingin kita prediksi menggunakan model Naïve Bayes. Di sisi lain, variabel H adalah hipotesis yang menyatakan bahwa data E termasuk ke dalam kelas tertentu. Selanjutnya, pada persamaan ini juga terdapat P(H|E), yang merupakan probabilitas dari hipotesis H mengingat kondisi E, dikenal sebagai probabilitas posterior. Probabilitas ini menggambarkan tingkat kebenaran hipotesis H berdasarkan kondisi E. Selanjutnya, P(H) adalah probabilitas dari hipotesis H sebelum data E dipertimbangkan, dikenal sebagai probabilitas prior, yang digunakan untuk menilai kemungkinan terjadinya hipotesis H tanpa data E. Lebih lanjut, P(E|H) menunjukkan probabilitas kemunculan

data E berdasarkan kondisi hipotesis H, memberikan wawasan mengenai hubungan antara data E dan hipotesis H serta seberapa sering data E muncul dalam konteks tersebut. Terakhir, P(E) adalah probabilitas keseluruhan dari data E, menunjukkan frekuensi kemunculan data E dalam dataset. Pemahaman mendalam tentang setiap variabel dalam persamaan teorema Bayes sangat penting untuk pengembangan model Naïve Bayes yang efektif dan akurat, baik dengan menggunakan maupun tanpa menggunakan metode optimasi seperti Particle Swarm Optimization.

2.3. Particle Swarm Optimization (PSO)

Algoritma PSO dikembangkan untuk mengatasi masalah optimasi tanpa batasan. Berkat kesederhanaan dan efisiensi pencariannya yang menakjubkan, keefektifan PSO telah terbukti di berbagai bidang, termasuk optimasi fungsi, optimasi rute kendaraan, dan pemrosesan gambar. Cara kerja algoritma ini yang meniru perilaku kawanan makhluk hidup seperti burung atau ikan. PSO merupakan algoritma iteratif yang menggunakan pendekatan berbasis populasi. Populasi ini terdiri dari sejumlah partikel yang digunakan untuk menyelesaikan masalah optimasi, diawali dengan kumpulan solusi acak. Dalam proses pencarian, partikel-partikel ini secara bertahap mengarah ke lokasi yang dinilai lebih efektif dalam pencarian solusi.

Algoritma ini memulai prosesnya dengan menempatkan N partikel secara acak dalam ruang pencarian. Setiap partikel dalam populasi akan memiliki vector posisi yang unik disimbolkan dengan $x_{id}(t)$ dan vector kecepatan dilambangkan dengan $v_{id}(t)$. Posisi dan kecepatan partikel ke-i akan diperbarui sesuai dengan persamaan berikut tiap iterasi.

$$c_1 r_1[p_{id}(t) - x_{id}(t)] + c_2 r_2\left[p_{gd}(t) - x_{id}(t)\right] + v_{id}(t) = v_{id}(t+1) \quad (2)$$

$$v_{id}(t+1) + x_{id}(t) = x_{id}(t+1) \quad (3)$$

Dengan d adalah dimensi ke-d, $x_{id}(t)$ dan $v_{id}(t)$ secara berurutan adalah posisi dan kecepatan partikel ke-i pada iterasi ke-t. Kemudian $p_{id}(t)$ adalah posisi terbaik yang pernah ditemukan partikel ke-i sementara $p_{gd}(t)$ adalah posisi optimal secara global yang ditemukan oleh semua partikel pada populasi. Adapun c_1 dan c_2 adalah koefisien percepatan yang merupakan konstanta dengan nilai 1.49. Lalu r_1 dan r_2 adalah dua angka acak yang dibangkitkan dari rentang $[0, 1]$. Pada konteks ini, posisi optimal adalah hyper-parameter optimal yang muncul pada posisi dimana nilai fitness seminimal mungkin dan bersifat konstan [6].

2.4. Metrik Evaluasi

Dalam membandingkan performa kedua sistem yang ada, akan digunakan metrik evaluasi berupa akurasi/ Penulis akan mencari nilai akurasi dari sistem dengan membagi data menjadi data latih dan data uji. Kemudian data pada kolom fitur dari data uji akan dimasukkan ke dalam kedua sistem dan sistem akan memberikan hasil berupa rekomendasi tanaman. Hasil yang diberikan ini kemudian dibandingkan dengan data pada kolom labelnya dan didapatkanlah akurasi dari model yang telah dibangun.

3. Hasil dan Diskusi

Penulis membangun dua sistem yaitu sistem rekomendasi berbasis Naïve Bayes yang tidak menggunakan PSO dan yang menggunakan PSO menggunakan bahasa pemrograman Python pada *platform* Google Colaboratory.

3.1 Model Naïve Bayes Tanpa PSO

Dalam membangun model ini, pertama penulis membagi dataset menjadi fitur dan target. Lalu penulis membagi data yang ada menjadi data latih dan data uji untuk validasi hasil. Setelah itu penulis mengimport library sklearn. naive_bayes untuk mengambil fungsi GaussianNB yang merupakan algoritma Naïve Bayes lebih tepatnya Gaussian Naïve Bayes. Kemudian penulis memasukkan data training ke model dan dihasilkanlah model Naïve Bayes yang dapat memberikan rekomendasi tanaman hortikultur. Adapun implementasinya dalam bahasa pemrograman Python adalah sebagai berikut.

Pada kode di atas, penulis juga melakukan validasi dengan menggunakan metode k-fold cross-validation dengan fold sebanyak lima. Adapun hasil akurasi dari lima fold tersebut adalah [0.99545455 0.99090909 0.99545455 0.99545455 0.99772727] dan skor akurasinya secara umum sebesar 99.5%. Penulis juga membuat confusion matrix menggunakan Python dengan kode seperti di bawah.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score, KFold

df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/CropRecommendation.csv')
X = df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = GaussianNB()
k_folds = 5
cv = KFold(n_splits=k_folds, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')

print ("Cross-Validation Scores:", cv_scores)
```

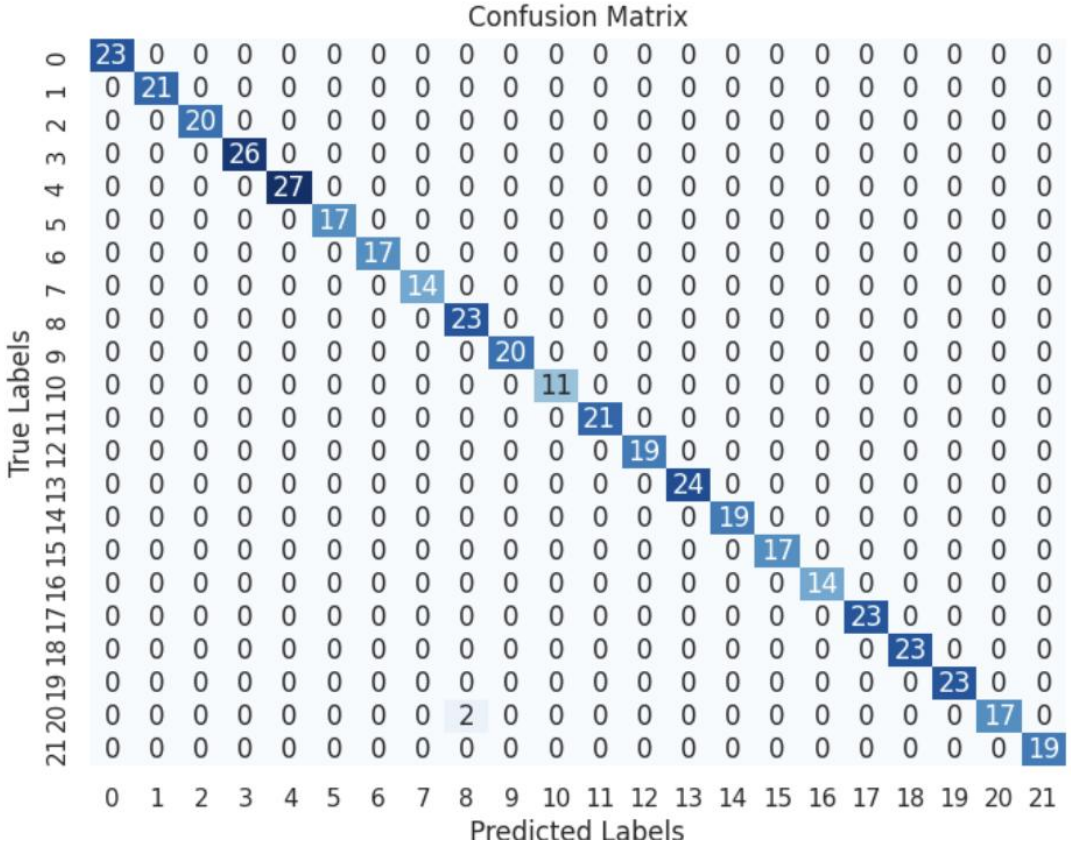
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

p-ISSN: 2986-3929
e-ISSN: 3032-1948

Dengan menjalankan kode di atas, akan dibuat sebuah confusion matrix yang menunjukkan model salah dalam memprediksi label apa dan seberapa banyak. Confusion matrix yang dibuat penulis ditunjukkan oleh gambar 3.



Gambar 3. Confusion Matrix Hasil Prediksi Model Naïve Bayes

3.2 Model Naïve Bayes Dengan PSO

Dalam membangun model Naïve Bayes dengan menerapkan PSO akan mirip seperti pada saat membangun model Naïve Bayes biasa, yang berbeda hanyalah diterapkannya metode PSO untuk mencari nilai hyperparameter var_smoothing yang terbaik. Kemudian hyperparameter ini digunakan untuk membangun model Naïve Bayes yang baru dan setelah itu bisa dilihat seberapa baik akurasinya. Adapun tahap mengimplementasikan PSO menggunakan Python adalah seperti berikut.

a. Mendefinisikan Partikel
   Penulis membuat sebuah kelas partikel yang memiliki beberapa metode yaitu metode untuk menginisiasi posisi awal kecepatan awal partikel, mengubah posisi dan kecepatan di tiap iterasi, dan metode untuk mengambil posisi partikel di tiap iterasi.

b. Membuat Fungsi PSO
   Pada bagian ini pertama penulis membuat sebuah looping untuk melihat posisi terbaik tiap partikel yang ada pada populasi. Jika ada partikel yang memiliki posisi lebih baik dari posisi global terbaik saat ini, posisi partikel tersebut akan dijadikan sebagai posisi global terbaik yang baru. Kemudian dilakukan iterasi sebanyak jumlah iterasi yang didefinisikan untuk PSO. Di setiap iterasi itu metode untuk mengubah posisi dan kecepatan tiap partikel akan dipanggil sehingga di tiap iterasinya posisi dan kecepatan semua partikel akan diperbarui. Setelah semua iterasi dilakukan, posisi global terbaik yang ada akan menjadi nilai paling

p-ISSN: 2986-3929
e-ISSN: 3032-1948

optimal dari hyperparameter yang kit acari, dalam hal ini adalah nilai dari hyperparameter var_smoothing.

```
class Particle:
  def __init__ (self, lower_bounds, upper_bounds, dimensions):
    self.position = np.random.uniform(low=lower_bounds, high=upper_bounds,
size=dimensions)
    self.velocity = np.zeros(dimensions)
    self.pbest = self.position.copy()
    self.pbest_fitness = float("-inf")

  def update (self, gbest, w, c1, c2):
    self.velocity = w * self.velocity + c1 * np.random.randn(*self.position.shape) *
(self.pbest - self.position) + c2 * np.random.randn(*self.position.shape) * (gbest -
self.position)
    self.position += self.velocity
    self.position = np.clip(self.position, lower_bounds, upper_bounds)
    fitness = fitness_function(X_train, y_train, self.get_params())
    if fitness > self.pbest_fitness:
      self.pbest = self.position.copy()
      self.pbest_fitness = fitness

  def get_params(self):
    return {key: value for key, value in zip(hyperparameter_ranges.keys(),
self.position)}

def pso(X_train, y_train, lower_bounds, upper_bounds, swarm_size, max_iter, w, c1,
c2):
  swarm = [Particle (lower_bounds, upper_bounds, len(lower_bounds)) for _ in
range(swarm_size)]
  gbest = None
  gbest_fitness = float("-inf")

  for particle in swarm:
    fitness = fitness_function(X_train, y_train, particle.get_params())
    if fitness > gbest_fitness:
      gbest = particle.position.copy()
      gbest_fitness = fitness

  for iter in range(max_iter):
    for particle in swarm:
      particle.update(gbest, w, c1, c2)

    for particle in swarm:
      fitness = fitness_function(X_train, y_train, particle.get_params())
      if fitness > gbest_fitness:
        gbest = particle.position.copy()
        gbest_fitness = fitness

  best_particle = swarm [np.argmax([particle.pbest_fitness for particle in swarm])]
  return best_particle.get_params()
```

Adapun representasi kode dari tahapan di atas adalah sebagai berikut.

Pada implementasi PSO di atas, populasinya adalah 40 dan dilakukan iterasi sebanyak 100 kali. Dengan menggunakan kode di atas, didapatkan nilai terbaik untuk hyperparameter var_smoothing pada model Gaussian Naïve bayes adalah 1e-09. Adapun hasil yang didapatkan penulis setelah menggunakan nilai optimal tersebut pada hyperparameter model dan melakukan validasi dengan menggunakan metode k-fold cross-validation dengan fold sebanyak lima. Adapun hasil akurasi dari lima fold tersebut adalah [0.99545455 0.99090909 0.99545455 0.99545455 0.99772727] dan akurasi secara umum sebesar 99.5%. Hasil ini sama seperti yang didapatkan saat penulis membangun model tanpa menggunakan PSO. Ini berarti penggunaan PSO pada kasus ini tidak memberikan perubahan.

4. Kesimpulan

Dari penelitian ini, penulis mencoba menerapkan metode particle swarm optimization untuk mencari nilai hyperparameter var_smoothing dari model machine learning Gaussian Naïve Bayes. Didapatkan hasil 1e-09 sebagai nilai terbaik dari hyperparameter var_smoothing ini. Setelah menggunakan nilai ini sebagai hyperparameter, penulis melatih model dan mendapatkan hasil skor akurasi secara umum sebesar 99.5%. Adapun penulis juga membangun model Gaussian Naïve Bayes tanpa menerapkan PSO dan mendapatkan hasil skor akurasi model secara umum sebesar 99.5%. Dari perbandingan hasil ini dapat dilihat bahwa tidak ada perbedaan dari skor akurasi model yang menggunakan PSO dan tidak menggunakan PSO pada kasus yang diangkat penulis.

Daftar Pustaka

[1] D. Gosai, C. Raval, R. Nayak, H. Jayswal, and A. Patel, “Crop Recommendation System using Machine Learning,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 558–569, Jun. 2021, doi: 10.32628/CSEIT2173129.

[2] T. Setiadi, F. Noviyanto, H. Hardianto, A. Tarmuji, A. Fadlil, and M. Wibowo, “Implementation of Naïve Bayes Method in Food Crops Planting Recommendation,” *International Journal of Scientific & Technology Research*, vol. 9, p. 2, 2020, [Online]. Available: www.ijstr.org

[3] A. S. Talita, O. S. Nataza, and Z. Rustam, “Naïve Bayes Classifier and Particle Swarm Optimization Feature Selection Method for Classifying Intrusion Detection System Dataset,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1742-6596/1752/1/012021.

[4] Y. K. Putra, Fathurrahman, and M. Sadali, “Comparison of Pso-Based Naive Bayes and Naive Bayes Algorithm in Determining the Feasibility of Bumdes Credit,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1742-6596/1539/1/012030.

[5] Shalehah, Muhammad Itqan Mazdadi, Andi Farmadi, Dwi Kartini, and Muliadi, “Implementation of Particle Swarm Optimization Feature Selection on Naïve Bayes for Thoracic Surgery Classification,” *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 5, no. 3, pp. 150–158, Jul. 2023, doi: 10.35882/jeemi. v5i3.305.

[6] Y. Dai, M. Khandelwal, Y. Qiu, J. Zhou, M. Monjezi, and P. Yang, “A hybrid metaheuristic approach using random forest and particle swarm optimization to study and evaluate backbreak in open-pit blasting,” *Neural Comput Appl*, vol. 34, no. 8, pp. 6273–6288, Apr. 2022, doi: 10.1007/s00521-021-06776-z.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

Halaman ini sengaja dibiarkan kosong