

LAPORAN UJIAN AKHIR SEMESTER
ARTIFICIAL INTELLIGENCE

“Dry Bean Dataset”

Dosen Pengampu: Cakra Adipura Wicaksana S.T., M.T.



Disusun Oleh:

Hanny Destian Marzuliyanti (3337220057)

Rifdatun Nafi'ah (3337220075)

Nanda Eka Prihantoro (3337220091)

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS SULTAN AGENG TIRTAYASA
2023

Kata Pengantar

Dengan penuh rasa syukur dan rahmat-Nya, kami ingin menyampaikan ucapan terima kasih kepada Allah yang Maha Mulia atas karunia dan petunjuk-Nya yang telah diberikan kepada kami, sebagai kelompok, dalam menyelesaikan tugas akhir mata kuliah Pengantar Artificial Intelligence berjudul "Dry beam data set". Kiranya shalawat dan salam senantiasa tercurah kepada Rasulullah Shallallahu Alaihi Wasallam, yang menjadi sumber inspirasi dan teladan terbaik bagi seluruh umat manusia.

Tak lupa, kami mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Cakra Adipura Wicaksana, ST., MT, selaku dosen pengajar mata kuliah Pengantar Artificial Intelligence. Beliau telah memberikan tugas akhir ini kepada kami dan memberi kesempatan untuk memperluas pengetahuan kami.

Kami sadar bahwa kesempurnaan tidaklah ada, dan mungkin terdapat kesalahan dalam pengerjaan proyek dan penyusunan laporan tugas akhir kami. Oleh karena itu, dengan tulus dan sungguh-sungguh, kami memohon maaf yang mendalam atas segala kesalahan yang telah kami lakukan. Kebenaran berasal dari Allah, dan kesalahan adalah tanggung jawab kami sendiri. Semoga Allah SWT senantiasa melimpahkan rahmat dan ridha-Nya kepada kita semua.

DAFTAR ISI

BAB I.....	4
PENDAHULUAN	4
1.1 Latar Belakang	4
1.2. Rumusan masalah	5
1.3. Tujuan	5
1.4. Algoritma Genetika.....	5
1. 4.1. Bagan Alur Algoritma Genetika	7
1.5. Dry Bean Dataset	9
BAB II.....	11
PEMBAHASAN.....	11
2.1. Simulasi dan Penjelasan Source Code	11
2.2.. Analisa Program.....	29
BAB III	30
PENUTUP	30
3. Kesimpulan	30

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pertanian dan industri pangan memainkan peran penting dalam mendukung kehidupan manusia. Di dalam industri pangan salah satunya biji kering, merupakan komoditas utama yang memiliki nilai ekonomi tinggi. Identifikasi dan klasifikasi biji-bijian menjadi sangat penting untuk memastikan kualitas dan keragaman hasil pertanian. Meskipun terdapat berbagai varietas biji kering, seringkali sulit untuk membedakan antara jenis yang satu dengan yang lain secara visual.

Dalam konteks ini, pengembangan sistem visi komputer untuk mengklasifikasikan tujuh varietas biji kering menjadi langkah penting dalam meningkatkan efisiensi dan akurasi proses ini. Melalui penggunaan teknologi ini, diharapkan dapat meminimalkan kesalahan manusia dalam mengenali dan memisahkan varietas biji kering yang serupa, yang pada akhirnya dapat meningkatkan kualitas dan nilai jual hasil dari pertanian.

Dengan memahami tantangan ini, penelitian ini bertujuan untuk mengembangkan sistem visi komputer yang mampu membedakan antara tujuh varietas biji kering yang berbeda berdasarkan ciri-ciri visual. Melalui penggunaan analisis fitur biji, diharapkan sistem ini dapat memberikan kontribusi positif pada pemrosesan dan pengenalan biji kering dalam skala industri pertanian menggunakan dataset seimbang dengan 16 fitur (Area, Perimeter, MajorAxisLength, MinorAxisLength, AspectRatio, Eccentricity, ConvexArea, EquivDiameter, Extent, Solidity, roundness, Compactness, ShapeFactor1, ShapeFactor2, ShapeFactor3, ShapeFactor4) dan 7 varietas/class kacang kering (Dermason, Sira, Seker, CALi, Bombay, Horoz, dan Barbunya).

Dengan demikian, penelitian ini tidak hanya relevan dalam konteks teknologi informasi dan pertanian modern, tetapi juga memiliki dampak potensial pada peningkatan kualitas produk pertanian dan efisiensi proses industri.

1.2. Rumusan masalah

- a. Mencari akurasi prediksi dengan target serta parameter yang digunakan.
- b. Mencari berapa lama proses training data diukur.
- c. Mencari tahu kekurangan dan kelebihan dari percobaan program Algoritma Genetika.

1.3. Tujuan

- a. Menganalisis fitur-fitur yang mempengaruhi dataset biji kacang kering.
- b. Merancang dan mengimplementasikan model klasifikasi berbasis visi komputer menggunakan teknik segmentasi dan ekstraksi fitur.
- c. Menentukan akurasi prediksi dengan target serta parameter yang digunakan.
- d. Menentukan berapa lama proses data dapat diukur.
- e. Menemukan kelebihan dan kekurangan pada Algoritma Genetika.

1.4. Algoritma Genetika

Algoritma genetika, yang juga dikenal dengan Genetic Algorithm (GA), merupakan sebuah metode pencarian yang terinspirasi dari seleksi alam dalam proses evolusi. GA digunakan sebagai suatu pendekatan untuk mengatasi masalah local optimum. GA telah terbukti efektif dalam masalah pencarian dan optimasi. Dalam penelitian ini, Ga akan ditetapkan sebagai solusi untuk mengatasi masalah tersebut.

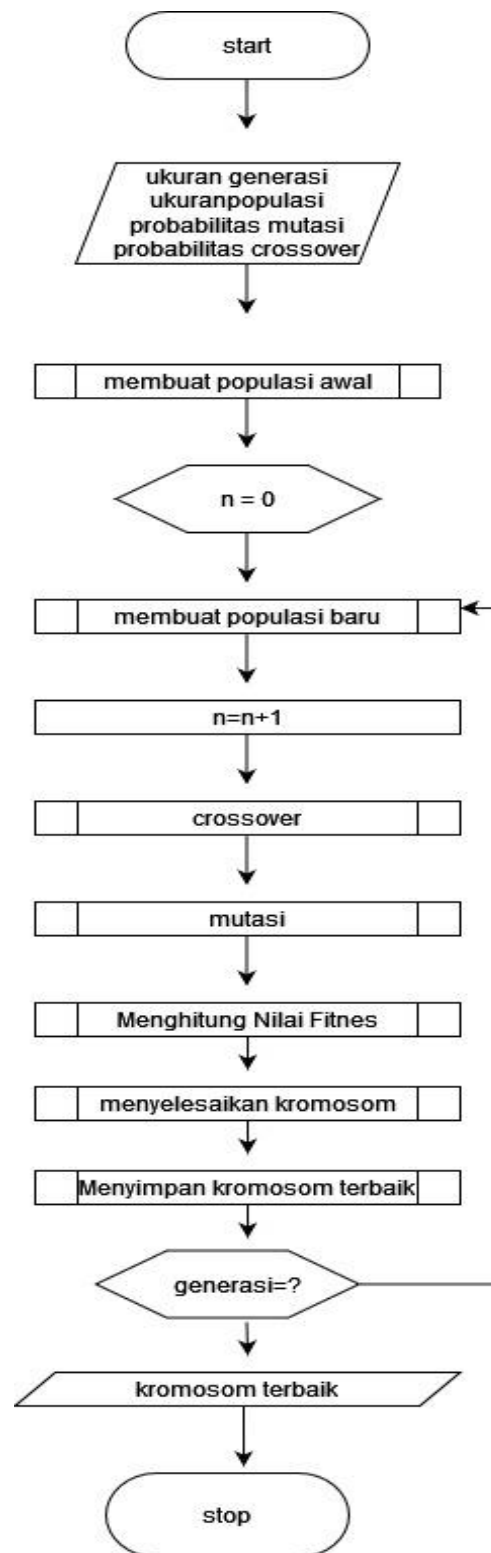
Namun, perlu diperhatikan bahwa dari segi waktu, algoritma genetika membutuhkan waktu yang lama dibandingkan dengan algoritma K-means. Proses komputasi yang lebih banyak dan kompleks pada algoritma genetika ini menjadi penyebab utamanya. Meskipun demikian, dengan hasil pengelompokkan yang lebih baik, trade-off ini mungkin masih dapat diterima tergantung pada kebutuhan dan kendala waktu yang ada.

Singkatnya Algoritma genetika adalah algoritma optimasi yang terinspirasi dari konsep evolusi dalam biologi. Algoritma ini digunakan untuk mencari solusi optimal atau mendekati solusi optimal dari suatu masalah optimasi. Berikut adalah langkah-langkah umum dalam algoritma genetika:

- **Inisialisasi Populasi:**
Populasi awal yang terdiri dari individu-individu atau kromosom yang merepresentasikan solusi potensial dibuat secara acak. Setiap kromosom mewakili suatu solusi di dalam ruang pencarian.
- **Evaluasi Fungsi Fitness:**
Setiap individu dalam populasi dievaluasi menggunakan fungsi fitness yang mengukur sejauh mana solusi tersebut mendekati solusi optimal. Fungsi fitness mengukur kualitas relatif setiap solusi.
- **Seleksi:**
Individu-individu dipilih untuk reproduksi berdasarkan nilai fitness mereka. Individu dengan nilai fitness yang lebih tinggi memiliki peluang lebih besar untuk dipilih. Ada berbagai metode seleksi seperti roulette wheel selection, turnamen seleksi, atau metode lainnya.
- **Reproduksi:**
Individu yang dipilih bereproduksi untuk menghasilkan keturunan. Ini melibatkan proses penggabungan (crossover) genetik antara dua individu untuk menghasilkan satu atau lebih keturunan baru.
- **Mutasi:**
Dalam langkah ini, beberapa individu atau kromosom mungkin mengalami mutasi, yang merupakan perubahan acak dalam gen atau parameter solusi. Mutasi memberikan variasi genetik baru dalam populasi.
- **Iterasi:**
Populasi lama digantikan dengan populasi baru yang dihasilkan dari langkah-langkah reproduksi, crossover, dan mutasi. Proses ini menciptakan generasi baru individu dengan harapan meningkatkan nilai fitness secara keseluruhan.

Algoritma genetika dapat disesuaikan dengan berbagai jenis masalah optimasi dengan memodifikasi parameter dan operasi, seperti metode crossover, mutasi, dan seleksi. Selama iterasi berulang, algoritma genetika secara evolusioner mencari solusi yang semakin mendekati atau bahkan optimal terhadap suatu masalah.

1. 4.1. Bagan Alur Algoritma Genetika



Gambar 1.

Langkah pertama melibatkan inisialisasi parameter awal pada proses Algoritma Genetika ini adalah sebagai berikut :

- Penentuan tujuan dan batas waktu untuk masing-masing tujuan.
- Menetapkan ukuran individu dalam setiap populasi.
- Menetapkan ukuran generasi.
- Menentukan probabilitas crossover.
- Menetapkan probabilitas mutasi.
- Setelah parameter awal diatur, langkah berikutnya adalah menciptakan populasi awal, dengan panjang kromosom sesuai dengan jumlah tujuan yang akan dicapai, dan jumlah populasi sesuai dengan individu yang diinisialisasi sebelumnya.

Setelah mendapatkan populasi awal, langkah selanjutnya adalah melibatkan reproduksi melalui crossover dan mutasi. Proses crossover dan mutasi melibatkan pengambilan populasi sebagai calon induk, dengan pemilihan induk dilakukan secara acak untuk menghasilkan anak sejumlah probabilitas crossover dan mutasi.

Langkah berikutnya adalah menghitung nilai kebugaran (fitness) dari setiap kromosom dalam semua proses pada generasi ini, menggunakan Persamaan 1 dan Persamaan 2. Setelah semua kromosom dihitung nilai fitnessnya, langkah berikutnya adalah seleksi kromosom untuk diproses pada generasi berikutnya, serta menyimpan kromosom dengan nilai fitness terbaik. Seleksi kromosom terbaik melibatkan perbandingan nilai fitness terbaik dari setiap generasi, yang melibatkan kromosom dari generasi awal dan hasil crossover dan mutasi.

Hasil akhir dari algoritma genetika adalah menampilkan kromosom yang memiliki nilai fitness tertinggi dari semua generasi.

1.5. Dry Bean Dataset

Dataset ini berisi tentang informasi kacang kering, data yang telah tersedia diperuntukkan

untuk membuat sebuah program yang nantinya diharapkan dapat mempermudah pengklasifikasian kacang kering.

Dalam dataset kacang kering terdapat beberapa atribut yaitu :

1. Area : Area zona kacang dan jumlah piksel di dalam batas-batasnya.
2. Parimeter : Keliling kacang didefinisikan sebagai panjang perbatasannya.
3. MajorAxisLength : Jarak antara ujung garis terpanjang yang dapat ditarik dari sebuah kacang.
4. MinorAxisLength : Garis terpanjang yang bisa ditarik dari biji sambil berdiri tegak lurus pada sumbu utama.
5. AspectRation : Menentukan hubungan antara MajorAxisLenght dan Minor AxisLength.
6. Eccentricity : Eksentrisitas elips yang memiliki momen yang sama dengan wilayahnya.
7. ConvexArea : Area cembung yaitu jumlah piksel dalam poligon cembung terkecil yang dapat memuat area biji kacang.
8. EquivDiameter : Diameter lingkaran yang memiliki area yang sama dengan area biji kacang.
9. Extent : Rasio piksel dalam kotak pembatas terhadap luas biji.
10. Solidity : Dikenal sebagai konveksitas. Rasio piksel dalam cangkang cembung dengan yang ditemukan dalam biji kopi.
11. Roundness : Dihitung dengan rumus : $(4\pi A) / (P^2)$
12. Compactness : Mengukur kebulatan suatu benda: E_d/L
13. ShapeFactor1
14. ShapeFactor2
15. ShapeFactor3
16. ShapeFactor4

Dan 7 variates / class kacang kering :

1. Dermason
2. Sira
3. Seker
4. Cali
5. Bombay
6. Horoz
7. Barbunya.

BAB II

PEMBAHASAN

2.1. Simulasi dan Penjelasan Source Code

```
IMPORT LIBRARY

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt |
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

[22] ✓ 0.0s
```

- Melakukan import library yang diperlukan;
 - a. pandas digunakan untuk manipulasi dan analisis data.
 - b. numpy adalah pustaka untuk operasi numerik dan manipulasi array.
 - c. seaborn dan matplotlib.pyplot digunakan untuk visualisasi data.
 - d. train_test_split dari sklearn.model_selection digunakan untuk membagi dataset menjadi data latih dan data uji.
 - e. RandomForestClassifier dari sklearn.ensemble adalah model klasifikasi yang akan digunakan.

```
Membuat Dataset untuk Training dan Validasi

# Mengambil data dari file CSV
dataset = pd.read_csv('Dry Bean Dataset.csv')
dataset

[23] ✓ 0.0s
```

- Kode ini merupakan langkah awal dalam memuat dataset ke dalam lingkungan Python untuk analisis lebih lanjut.

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactnes
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856	0.958027	0.91335
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272751	0.783968	0.984986	0.887034	0.95386
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559	0.947849	0.90877
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.467062	0.782681	0.976696	0.903936	0.92832
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893	0.984877	0.97051
...
13606	42097	759.696	288.721612	185.944705	1.552728	0.765002	42508	231.515799	0.714574	0.990331	0.916603	0.80186
13607	42101	757.499	281.576392	190.713136	1.476439	0.735702	42494	231.526798	0.799943	0.990752	0.922015	0.82225
13608	42139	759.321	281.539528	191.187979	1.472582	0.734065	42569	231.631261	0.729932	0.989899	0.918424	0.82273
13609	42147	763.779	283.382636	190.275731	1.489326	0.741055	42667	231.653247	0.705389	0.987813	0.907906	0.81745
13610	42159	772.237	295.142741	182.204716	1.619841	0.786693	42600	231.686223	0.788962	0.989648	0.888380	0.78499

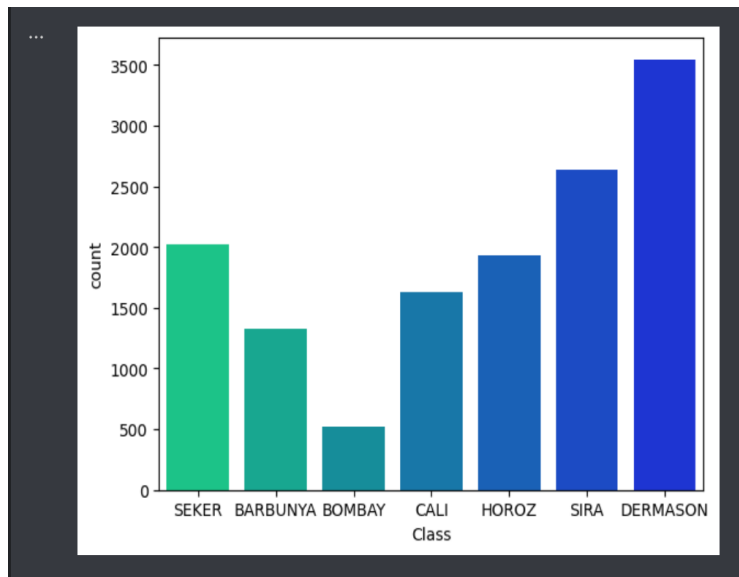
13611 rows x 13 columns

- Tampilan keseluruhan data, lima data awal dan lima data akhir.

```

> sns.countplot(x='Class',data=dataset,palette="winter_r")
[24] ✓ 0.2s
... C:\Users\Rifdatun Nafi'ah\AppData\Local\Temp\ipykernel_7500\179829218.py:1: FutureWarning:
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign
sns.countplot(x='Class',data=dataset,palette="winter_r")
... <Axes: xlabel='Class', ylabel='count'>

```



- Kode ini adalah contoh visualisasi sederhana yang berguna untuk memahami distribusi kategori pada suatu variabel dalam dataset

```

#digunakan untuk menghasilkan ringkasan statistik deskriptif dari dataset yang disimpan dalam variabel 'dataset'.
dataset.describe()
[62] ✓ 0.3s

```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity
count	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000
mean	53048.284549	855.283459	320.141867	202.270714	1.583242	0.750895	53768.200206	253.064220	0.749733	0.987143
std	29324.095717	214.289696	85.694186	44.970091	0.246678	0.092002	29774.915817	59.177120	0.049086	0.004660
min	20420.000000	524.736000	183.601165	122.512653	1.024868	0.218951	20684.000000	161.243764	0.555315	0.919246
25%	36328.000000	703.523500	253.303633	175.848170	1.432307	0.715928	36714.500000	215.068003	0.718634	0.985670
50%	44652.000000	794.941000	296.883367	192.431733	1.551124	0.764441	45178.000000	238.438026	0.759859	0.988283
75%	61332.000000	977.213000	376.495012	217.031741	1.707109	0.810466	62294.000000	279.446467	0.786851	0.990013
max	254616.000000	1985.370000	738.860154	460.198497	2.430306	0.911423	263261.000000	569.374358	0.866195	0.994677

- Digunakan untuk menghasilkan statistik deskriptif ringkas dari dataset, yang memberikan pemahaman awal tentang distribusi dan karakteristik dari setiap kolom numerik.

```
# menampilkan ringkasan statistik deskriptif khusus untuk kolom-kolom dengan tipe data objek (non-numeric)
dataset.describe(include='object')
[26] ✓ 0.0s
```

	Class
count	13611
unique	7
top	DERMASON
freq	3546

- Digunakan untuk menampilkan ringkasan statistik deskriptif khusus untuk kolom-kolom dengan tipe data objek (non-numeric) dalam dataset.

```
# Membuat pembagian antara feature (x) dan target (y)
feature = dataset.drop(columns=['Class'])
target = dataset['Class']
feature
[27] ✓ 0.0s
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactnes
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856	0.958027	0.91335
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272751	0.783968	0.984986	0.887034	0.95386
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559	0.947849	0.90877
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.467062	0.782681	0.976696	0.903936	0.92832
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893	0.984877	0.97051
...
13606	42097	759.696	288.721612	185.944705	1.552728	0.765002	42508	231.515799	0.714574	0.990331	0.916603	0.80186
13607	42101	757.499	281.576392	190.713136	1.476439	0.735702	42494	231.526798	0.799943	0.990752	0.922015	0.82225
13608	42139	759.321	281.539928	191.187979	1.472582	0.734065	42569	231.631261	0.729932	0.989899	0.918424	0.82273
13609	42147	763.779	283.382636	190.275731	1.489326	0.741055	42667	231.653247	0.705389	0.987813	0.907906	0.81745
13610	42159	772.237	295.142741	182.204716	1.619841	0.786693	42600	231.686223	0.788962	0.989648	0.888380	0.78499

13611 rows x 16 columns

- Potongan kode ini berfokus pada pembagian antara fitur (features) dan target dalam suatu dataset.

```
# Membagi dataset menjadi dua bagian, yaitu training dan validasi dengan rasio 80:20
x_train , x_valid , y_train , y_valid = train_test_split(feature , target , test_size=0.2)
[28] ✓ 0.0s
```

- Potongan kode ini bertujuan untuk membagi dataset menjadi dua bagian: satu untuk pelatihan (training) dan satu untuk validasi, dengan rasio 80% training dan 20 % validationset.

MENENTUKAN GEN DAN POPULASI INISIALISASI AWAL

```
# Disini kita akan mengoptimalkan parameter untuk model Random Forest
# Untuk jumlah populasi, kita akan menetapkan jumlahnya dengan 4 setiap generasi

# Menentukan jumlah pohon
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 100, num = 4)]
# Menentukan kedalaman maksimal suatu pohon
max_depth = [int(x) for x in np.linspace(10, 100, 4)]
```

- Kode ini fokus pada optimasi parameter untuk model Random Forest, yang merupakan salah satu jenis model ensambel dalam pembelajaran mesin.
- Optimasi parameter bertujuan untuk mencari kombinasi nilai parameter yang memberikan performa model terbaik.
- Jumlah populasi mengacu pada jumlah solusi yang dievaluasi pada setiap generasi dalam algoritma optimasi.
- Pada potongan kode ini, jumlah populasi ditetapkan sebanyak 4 setiap generasi.

```
# Membuat variabel populasi awal
population = []
# Menambahkan nilai n_estimators ke dalam variabel populasi
population.append(n_estimators)
# Menambahkan nilai max_depth ke dalam variabel populasi
population.append(max_depth)
# Mengubah tipe variabel menjadi numpy array
population = np.array(population)
# Men-traspose array sehingga bentuknya menjadi array dengan baris sebagai individu (kromosom) dan kolom sebagai parameter
population = np.transpose(population)
population
```

```
... array([[ 10,  10],
          [ 40, 340],
          [ 70, 670],
          [100, 1000]])
```

- Kode ini memulai dengan membuat variabel population yang akan digunakan untuk menyimpan populasi awal.
- Populasi awal dalam konteks ini merujuk pada sekumpulan individu (kromosom) yang mewakili kombinasi parameter parameter yang akan dioptimalkan.

MELAKUKAN PENERAPAN ALGORITMA GENETIKA

jumlah data banyak dan jumlah variasi/kolom banyak menyebabkan lamanya proses

```
# Mengimpor modul yang berisi fungsi-fungsi pada algoritma genetika, seperti mating pool, crossover, mutation
import GA

# Menentukan jumlah generasi (loop)
num_generations = 9
# Menentukan jumlah parent setiap populasi
num_parents_mating = 2
# Menentukan ukuran populasi (individu x jumlah parameter)
pop_size = (4,2)

# Membuat 2 variabel untuk menyimpan nilai terbaik dari masing-masing training setiap generasi
train_accuracies = []
valid_accuracies = []

# Fungsi loop untuk algoritma genetika (Best result ditentukan oleh akurasi validasi)
for generation in range(num_generations):
    print("Generation : ", generation+1)

    # Melakukan fitness untuk mencari nilai masing-masing individu
    fitness = GA.cal_pop_fitness(population, x_train, y_train, x_valid, y_valid)
```

```
# Memilih individu terbaik untuk dijadikan parent
parents = GA.select_mating_pool(population, fitness,
                                num_parents_mating)

# Melakukan crossover atau keturunan dari parent yang dipilih sebelumnya
offspring_crossover = GA.crossover(parents,
                                    offspring_size=(pop_size[0]-parents.shape[0], 2))

# Melakukan mutasi pada individu baru lahir
offspring_mutation = GA.mutation(offspring_crossover)

# Membuat populasi baru dengan isi parent dan keturunannya
population[0:parents.shape[0], :] = parents
population[parents.shape[0]:, :] = offspring_mutation

# Mencetak generasi dan individu yang terbaiknya
print("Best result : ")
valid_fitness_last = []
training_fitness_last = []
for x in range(4):
    model_fit = RandomForestClassifier(n_estimators=population[x][0],
                                      max_depth=population[x][1])
    model_fit.fit(x_train, y_train)
    valid_accuracy = model_fit.score(x_valid, y_valid)
    train_accuracy = model_fit.score(x_train, y_train)
    valid_fitness_last.append(valid_accuracy)
    training_fitness_last.append(train_accuracy)
```

- Mengimpor modul yang berisi fungsi-fungsi yang dibutuhkan untuk algoritma genetika, seperti mating pool, crossover, dan mutation. Modul ini disebut GA.
- Menentukan jumlah generasi (iterasi), jumlah parent yang akan melakukan mating, dan ukuran populasi dalam bentuk tuple (jumlah individu, jumlah parameter).
- Membuat dua variabel untuk menyimpan nilai akurasi terbaik dari setiap training dan validasi pada setiap generasi.
- Melakukan loop sebanyak generasi yang ditentukan.
- Di setiap generasi, melakukan langkah-langkah algoritma genetika seperti fitness calculation, parent selection, crossover, mutation, dan pembentukan populasi baru.
- Menggunakan fungsi cal_pop_fitness dari modul GA untuk menghitung fitness setiap individu dalam populasi berdasarkan model RandomForestClassifier.
- Menggunakan fungsi select_mating_pool dari modul GA untuk memilih individu terbaik sebagai parent untuk mating.

- Menggunakan fungsi crossover dan mutation dari modul GA untuk melakukan crossover dan mutation pada individu-parent yang telah dipilih.
- Memperbarui populasi dengan memasukkan parent dan offspring yang telah melalui crossover dan mutation.
- Menggunakan model RandomForestClassifier dengan parameter terbaik dari populasi saat ini.
- Menyimpan akurasi terbaik dari setiap generasi untuk training dan validasi.
- Mencetak akurasi terbaik dan generasi terbaik

```
for x in range(4):
    model_fit = RandomForestClassifier(n_estimators=population[x][0],
                                     max_depth=population[x][1])
    model_fit.fit(x_train, y_train)
    valid_accuracy = model_fit.score(x_valid, y_valid)
    train_accuracy = model_fit.score(x_train, y_train)
    valid_fitness_last.append(valid_accuracy)
    training_fitness_last.append(train_accuracy)
    print(np.max(valid_fitness_last))
    train_accuracys.append(np.max(training_fitness_last))
    valid_accuracys.append(np.max(valid_fitness_last))

# Mencetak best fitness dan best generation
best_fitness = np.max(valid_accuracys)
best_generation = np.argmax(valid_accuracys) + 1
print("Best Fitness:", best_fitness)
print("Best Generation:", best_generation)
```

✓ 10m 9.2s

```
... Generation : 1
Best result :
0.9247153874403232
Generation : 2
Best result :
0.9236136614028645
Generation : 3
Best result :
0.9247153874403232
Generation : 4
Best result :
0.9247153874403232
Generation : 5
Best result :
0.9254498714652957
Generation : 6
Best result :
0.9265515975027543
Generation : 7
Best result :
0.9243481454278369
Generation : 8
Best result :
0.9265515975027543
Generation : 9
Best result :
0.9250826294528094
```

Best Fitness: 0.9265515975027543
Best Generation: 6

- berfungsi untuk membuat grafik perkembangan hasil terbaik (akurasi validasi maksimum) dari setiap generasi dalam algoritma genetika.

MEMBUAT GRAFIK NILAI KUALITAS TERBAIK (BEST FITNESS) PER GENERASI.

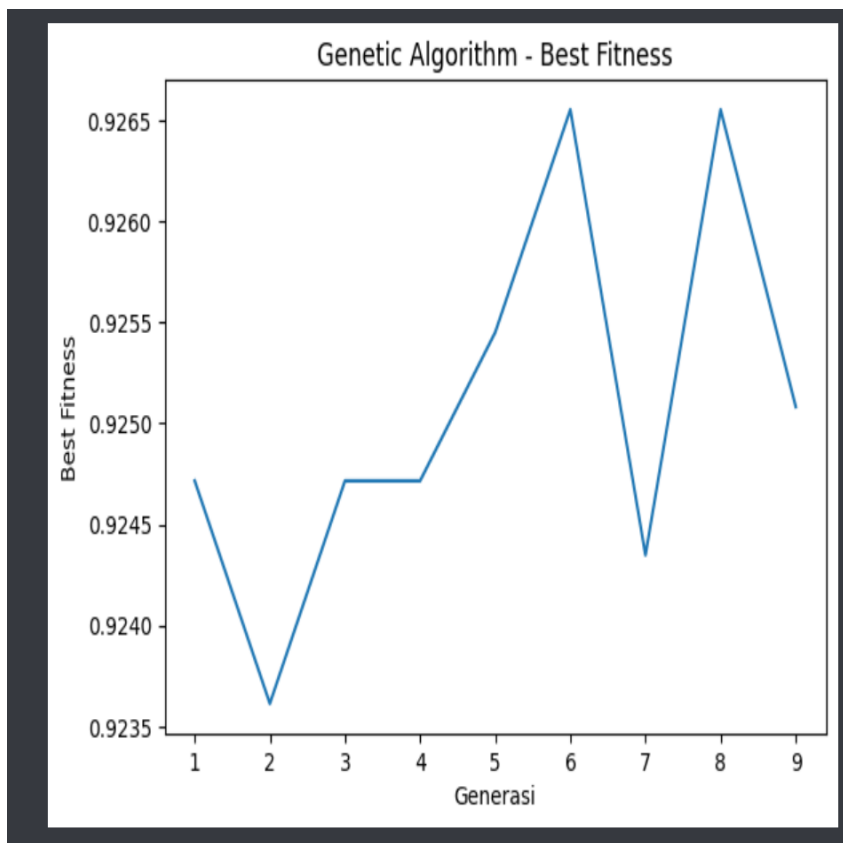
```
Run and Debug (Ctrl+Shift+D) ay dengan nilai x dari 1 hingga num_generations
generation_numbers = np.arange(1, num_generations+1)
# Menggunakan valid_accuracys sebagai nilai y
best_fitness_values = np.array(valid_accuracys)

# Membuat grafik menggunakan matplotlib
plt.plot(generation_numbers, best_fitness_values)

# Memberi label pada sumbu x dan y serta judul grafik
plt.xlabel('Generasi')
plt.ylabel('Best Fitness')
plt.title('Genetic Algorithm - Best Fitness')

# Menampilkan grafik
plt.show()
```

[32] ✓ 0.3s



- Berfungsi untuk membuat grafik perkembangan hasil terbaik (akurasi validasi maksimum) dari setiap generasi dalam algoritma genetika.

MENCARI AKURASI MENGGUNAKAN PARAMETER TERBAIK DARI HASIL ALGORITMA GENETIKA

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping
import numpy as np
import matplotlib.pyplot as plt

# Preprocessing
sc = StandardScaler()
sc.fit(x_train)
x_train = sc.transform(x_train)
x_valid = sc.transform(x_valid)
```

[33] ✓ 11.9s

- Selanjutnya mencari akurasi menggunakan parameter terbaik dari hasil algoritma
Pertama mengimport library dan fungsi metode yang diperlukan :
Berikut adalah penjelasan singkat dari setiap fungsi pada kode yang Anda berikan:
 1. `StandardScaler` dari `sklearn.preprocessing`: Ini adalah kelas yang digunakan untuk melakukan penskalaan fitur. Metode `fit` digunakan untuk menghitung mean dan standar deviasi dari data pelatihan, sedangkan metode `transform` digunakan untuk mengubah data menggunakan mean dan standar deviasi yang dihitung sebelumnya. Dalam kode ini, `StandardScaler` digunakan untuk melakukan penskalaan pada `x_train` dan `x_valid` agar memiliki mean 0 dan standar deviasi 1.
 2. `OneHotEncoder` dari `sklearn.preprocessing`: Ini adalah kelas yang digunakan untuk melakukan one-hot encoding pada variabel kategorikal. Dalam kode yang Anda berikan, tidak ada variabel kategorikal yang disebut dalam kode tersebut.
 3. `Sequential` dari `keras.models`: Ini adalah kelas yang digunakan untuk membuat model neural network secara berurutan. Model ini memungkinkan kita untuk menambahkan lapisan-lapisan (layers) secara berurutan.
 4. `Dense` dari `keras.layers`: Ini adalah lapisan (layer) yang terdiri dari neuron-neuron yang terhubung sepenuhnya. Setiap neuron dalam lapisan ini menerima input dari setiap neuron dalam lapisan sebelumnya dan menghasilkan output yang terhubung ke setiap neuron dalam lapisan berikutnya.
 5. `Dropout` dari `keras.layers`: Ini adalah lapisan (layer) yang digunakan untuk menghindari overfitting dalam model neural network. Lapisan `Dropout` secara acak mengabaikan sebagian unit-neuron dalam lapisan sebelumnya selama proses pelatihan, sehingga mendorong pembelajaran yang lebih umum dan mengurangi overfitting.

6. ``Adam`` dari ``keras.optimizers``: Ini adalah algoritma optimisasi yang digunakan untuk mengoptimalkan model neural network. Algoritma ini menggabungkan metode Stochastic Gradient Descent (SGD) dengan algoritma lain untuk meningkatkan kecepatan dan performa optimisasi.
7. ``ReduceLROnPlateau`` dari ``keras.callbacks``: Ini adalah callback yang digunakan untuk mengurangi laju pembelajaran (learning rate) saat suatu metrik berhenti meningkat. Dalam kode ini, callback ini mungkin digunakan untuk mengurangi laju pembelajaran jika tidak ada peningkatan dalam validasi setelah beberapa epoch.
8. ``ModelCheckpoint`` dari ``keras.callbacks``: Ini adalah callback yang digunakan untuk menyimpan model dengan kinerja terbaik selama pelatihan. Dalam kode ini, callback ini mungkin digunakan untuk menyimpan model dengan akurasi validasi terbaik.
9. ``EarlyStopping`` dari ``keras.callbacks``: Ini adalah callback yang digunakan untuk menghentikan pelatihan jika suatu metrik berhenti meningkat. Dalam kode ini, callback ini mungkin digunakan untuk menghentikan pelatihan jika tidak ada peningkatan dalam validasi setelah beberapa epoch.
10. ``numpy`` (diimpor sebagai ``np``): Ini adalah pustaka Python yang digunakan untuk manipulasi array dan operasi numerik. Dalam kode ini, ``numpy`` digunakan untuk melakukan beberapa operasi numerik pada data.
11. ``matplotlib.pyplot`` (diimpor sebagai ``plt``): Ini adalah pustaka Python yang digunakan untuk membuat visualisasi grafik dan plot. Dalam kode ini, ``matplotlib.pyplot`` digunakan untuk membuat plot.

Selanjutnya bagian Preprocessing :

Berikut adalah penjelasan singkat dari setiap fungsi pada bagian preprocessing yang Anda berikan:

1. ``StandardScaler()`` adalah sebuah objek dari kelas ``StandardScaler`` yang digunakan untuk melakukan penskalaan fitur.
2. ``sc.fit(x_train)`` digunakan untuk menghitung mean dan standar deviasi dari data pelatihan (``x_train``). Proses ini akan mengestimasi nilai mean dan standar deviasi dari setiap fitur pada ``x_train``.
3. ``x_train = sc.transform(x_train)`` digunakan untuk mentransformasikan data pelatihan (``x_train``) menggunakan mean dan standar deviasi yang telah dihitung sebelumnya. Proses ini akan mengubah setiap fitur pada ``x_train`` menjadi memiliki mean 0 dan standar deviasi 1.
4. ``x_valid = sc.transform(x_valid)`` digunakan untuk mentransformasikan data validasi

(`x_valid`) menggunakan mean dan standar deviasi yang telah dihitung dari data pelatihan (`x_train`). Hal ini dilakukan agar skala fitur pada data validasi serupa dengan skala fitur pada data pelatihan.

```
# Mengambil parameter terbaik dari hasil algoritma genetika
best_fitness = np.max(valid_accuracys)
best_generation = np.argmax(valid_accuracys) + 1
```

[34] ✓ 0.0s

- Mengambil parameter terbaik dari hasil algoritma genetika, dimana disini mengambil best_fitness dan best_generation untuk dijadikan parameter.

```
# Mengubah y_train dan y_valid ke dalam format one-hot encoding
num_classes = len(np.unique(y_train))
encoder = OneHotEncoder(categories='auto')
y_train_encoded = encoder.fit_transform(y_train.values.reshape(-1, 1)).toarray()
y_valid_encoded = encoder.transform(y_valid.values.reshape(-1, 1)).toarray()
```

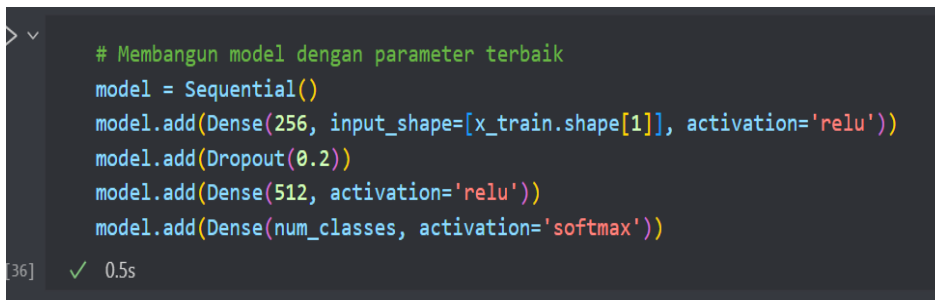
[35] ✓ 0.0s

- Menubah y_train dan y_valid ke dalam format one-hot encoding : hal ini berfungsi untuk mengubah label Class menjadi representasi biner.

Berikut adalah penjelasan singkat dari setiap fungsi pada kode yang Anda berikan:

1. `num_classes = len(np.unique(y_train))`: Fungsi ini menghitung jumlah kelas yang ada dalam data target `y_train`. `np.unique(y_train)` mengembalikan nilai-nilai unik dari `y_train`, dan `len()` digunakan untuk menghitung jumlah nilai unik tersebut. Hasilnya disimpan dalam variabel `num_classes`.
2. `encoder = OneHotEncoder(categories='auto')`: Objek `encoder` dibuat dari kelas `OneHotEncoder` dengan parameter `categories='auto'`. `OneHotEncoder` digunakan untuk melakukan one-hot encoding pada variabel kategorikal. Parameter `categories='auto'` mengindikasikan bahwa kategori akan ditentukan secara otomatis berdasarkan nilai unik dalam data.
3. `y_train_encoded = encoder.fit_transform(y_train.values.reshape(-1, 1)).toarray()`: Fungsi ini melakukan one-hot encoding pada data target `y_train`. `encoder.fit_transform()` digunakan untuk mengubah `y_train` menjadi representasi one-hot encoded. `y_train.values.reshape(-1, 1)` digunakan untuk mengubah dimensi `y_train` menjadi bentuk matriks 2D yang diperlukan oleh `fit_transform()`. Hasilnya disimpan dalam variabel `y_train_encoded`.

4. ``y_valid_encoded = encoder.transform(y_valid.values.reshape(-1, 1)).toarray()`:` Fungsi ini melakukan one-hot encoding pada data target ``y_valid`` menggunakan ``encoder`` yang telah di-fit pada data pelatihan. ``encoder.transform()`` digunakan untuk mengubah ``y_valid`` menjadi representasi one-hot encoded. ``y_valid.values.reshape(-1, 1)`` digunakan untuk mengubah dimensi ``y_valid`` menjadi bentuk matriks 2D yang diperlukan oleh ``transform()``. Hasilnya disimpan dalam variabel ``y_valid_encoded``.



```
> ✓  
# Membangun model dengan parameter terbaik  
model = Sequential()  
model.add(Dense(256, input_shape=[x_train.shape[1]], activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(512, activation='relu'))  
model.add(Dense(num_classes, activation='softmax'))  
[36] ✓ 0.5s
```

- Membangun model dengan parameter terbaik, ini adalah fungsi aktivasi dimana model dibangun dengan menggunakan parameter terbaik yang diambil dari hasil algoritma genetika sebelumnya.

Berikut adalah penjelasan singkat dari setiap fungsi pada kode :

1. ``model = Sequential()`:` Fungsi ini digunakan untuk membuat objek model neural network. Model ini akan dibangun secara berurutan dengan lapisan-lapisan yang ditambahkan satu per satu.
2. ``model.add(Dense(256, input_shape=[x_train.shape[1]], activation='relu'))`:` Fungsi ini menambahkan lapisan Dense (sepenuhnya terhubung) pertama ke model. Lapisan ini memiliki 256 unit-neuron dengan fungsi aktivasi ReLU. Parameter ``input_shape=[x_train.shape[1]]`` menentukan dimensi input yang diharapkan oleh model berdasarkan jumlah fitur pada ``x_train``.
3. ``model.add(Dropout(0.2))`:` Fungsi ini menambahkan lapisan Dropout ke model. Dropout digunakan untuk menghindari overfitting dengan secara acak mengabaikan sebagian unit-neuron dalam lapisan sebelumnya selama proses pelatihan. Angka ``0.2`` menunjukkan tingkat dropout, yaitu 20% dari unit-neuron akan diabaikan secara acak.
4. ``model.add(Dense(512, activation='relu'))`:` Fungsi ini menambahkan lapisan Dense kedua ke model. Lapisan ini memiliki 512 unit-neuron dengan fungsi aktivasi ReLU.
5. ``model.add(Dense(num_classes, activation='softmax'))`:` Fungsi ini menambahkan lapisan Dense terakhir ke model. Lapisan ini memiliki jumlah unit-neuron yang sama dengan jumlah kelas (``num_classes``) dalam tugas klasifikasi. Fungsi aktivasi yang digunakan adalah softmax, yang menghasilkan probabilitas untuk setiap kelas.

```
model.summary()
```

✓ 0.0s

- Fungsi `model.summary()` digunakan untuk menampilkan ringkasan (summary) dari arsitektur model neural network yang telah dibangun.

Output :

```
Model: "sequential"
Layer (type)                Output Shape              Param #
=====
dense (Dense)                (None, 256)              4352
dropout (Dropout)            (None, 256)              0
dense_1 (Dense)              (None, 512)              131584
dense_2 (Dense)              (None, 7)                3591
=====
Total params: 139527 (545.03 KB)
Trainable params: 139527 (545.03 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

✓ 0.0s

- Fungsi `loss`

```
lrd = ReduceLROnPlateau(monitor='val_loss', patience=10, verbose=1, factor=0.75, min_lr=1e-10)
mcp = ModelCheckpoint('model.h5')
es = EarlyStopping(verbose=1, patience=100)
```

✓ 0.0s

- Mengatur callback untuk mengoptimalkan pelatihan model dan menghindari overfitting.

Berikut adalah penjelasan singkat dari setiap objek yang didefinisikan dalam kode :

1. ``lrd = ReduceLROnPlateau(monitor='val_loss', patience=10, verbose=1, factor=0.75, min_lr=1e-10)``: Objek ``lrd`` merupakan instance dari kelas ``ReduceLROnPlateau`` yang digunakan untuk mengurangi tingkat learning rate (LR) saat suatu metrik (dalam hal ini ``val_loss``, yaitu loss pada data validasi) tidak mengalami perbaikan selama beberapa epoch (`patience`). Parameter ``monitor`` menentukan metrik yang akan dipantau, ``patience`` menentukan jumlah epoch yang diizinkan tanpa perbaikan sebelum mengurangi LR, ``verbose`` menentukan apakah

pesan peringatan akan ditampilkan, `factor` menentukan faktor pengurangan LR, dan `min_lr` menentukan batas bawah dari LR yang diizinkan.

2. `mcp = ModelCheckpoint('model.h5')`: Objek `mcp` merupakan instance dari kelas `ModelCheckpoint` yang digunakan untuk menyimpan model dengan performa terbaik selama pelatihan. Parameter `'model.h5'` menentukan nama file tempat model akan disimpan. Secara default, `ModelCheckpoint` akan menyimpan model dengan performa terbaik berdasarkan metrik yang dipantau, dalam hal ini `val_loss`.

3. `es = EarlyStopping(verbose=1, patience=100)`: Objek `es` merupakan instance dari kelas `EarlyStopping` yang digunakan untuk menghentikan pelatihan jika suatu metrik (misalnya, `val_loss`) tidak mengalami perbaikan dalam jangka waktu tertentu (patience). Parameter `verbose` menentukan apakah pesan peringatan akan ditampilkan, dan `patience` menentukan jumlah epoch yang diizinkan tanpa perbaikan sebelum pelatihan dihentikan.

```
# Melatih model dengan parameter terbaik
history = model.fit(x=x_train, y=y_train_encoded, epochs=best_generation, callbacks=[lrd, mcp, es], batch_size=32, validation_split=0.1)
✓ 19.2s
```

- Melatih model dengan parameter terbaik

```
... Epoch 1/6
WARNING:tensorflow:From c:\Users\Rifdatun Nafi'ah\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:492: The n
WARNING:tensorflow:From c:\Users\Rifdatun Nafi'ah\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\base_layer_utils.py:3

307/307 [=====] - 5s 10ms/step - loss: 0.3248 - accuracy: 0.8846 - val_loss: 0.2035 - val_accuracy: 0.9192 - lr: 0.0010
Epoch 2/6
1/307 [.....] - ETA: 3s - loss: 0.1053 - accuracy: 1.0000
c:\Users\Rifdatun Nafi'ah\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving
saving_api.save_model(
307/307 [=====] - 3s 11ms/step - loss: 0.2262 - accuracy: 0.9185 - val_loss: 0.1988 - val_accuracy: 0.9238 - lr: 0.0010
Epoch 3/6
307/307 [=====] - 3s 9ms/step - loss: 0.2198 - accuracy: 0.9221 - val_loss: 0.1876 - val_accuracy: 0.9293 - lr: 0.0010
Epoch 4/6
307/307 [=====] - 1s 4ms/step - loss: 0.2121 - accuracy: 0.9206 - val_loss: 0.1899 - val_accuracy: 0.9376 - lr: 0.0010
Epoch 5/6
307/307 [=====] - 3s 10ms/step - loss: 0.2054 - accuracy: 0.9257 - val_loss: 0.1812 - val_accuracy: 0.9339 - lr: 0.0010
Epoch 6/6
307/307 [=====] - 3s 11ms/step - loss: 0.1993 - accuracy: 0.9252 - val_loss: 0.1811 - val_accuracy: 0.9357 - lr: 0.0010
```

```
> ✓
# Evaluasi model pada data latih dan data validasi
scores_train = model.evaluate(x_train, y_train_encoded)
print("Training Accuracy: %.2f%%\n" % (scores_train[1] * 100))
scores_valid = model.evaluate(x_valid, y_valid_encoded)
print("Testing Accuracy: %.2f%%\n" % (scores_valid[1] * 100))
[41] ✓ 1.8s
```

- Evaluasi model pada data latih dan data validasi
Penjelasan kode di atas:

1. `scores_train = model.evaluate(x_train, y_train_encoded, verbose=0)`: Metode `evaluate()` digunakan untuk menghitung loss dan akurasi model pada data latih. `x_train` adalah input data latih, `y_train_encoded` adalah target yang telah di-encode, dan

verbose=0 digunakan untuk tidak menampilkan output selama evaluasi.

2. `print("Training Accuracy: %.2f%%" % (scores_train[1] * 100))`: Menampilkan nilai akurasi pada data latih dengan format persentase.

3. `scores_valid = model.evaluate(x_valid, y_valid_encoded, verbose=0)`: Metode `evaluate()` juga digunakan untuk menghitung loss dan akurasi model pada data validasi. `x_valid` adalah input data validasi, `y_valid_encoded` adalah target yang telah di-encode, dan `verbose=0` digunakan untuk tidak menampilkan output selama evaluasi.

4. `print("Validation Accuracy: %.2f%%" % (scores_valid[1] * 100))`: Menampilkan nilai akurasi pada data validasi dengan format persentase.

Output :

```
341/341 [=====] - 1s 4ms/step - loss: 0.1906 - accuracy: 0.9300
Training Accuracy: 93.00%

86/86 [=====] - 0s 3ms/step - loss: 0.2161 - accuracy: 0.9251
Testing Accuracy: 92.51%
```

```
# Melakukan prediksi pada data validasi
y_pred = np.argmax(model.predict(x_valid), axis=-1)
✓ 0.4s

86/86 [=====] - 0s 2ms/step
```

- Melakukan prediksi pada data validasi

```
# Menampilkan hasil prediksi dan label aktual
print("Predicted Labels:", y_pred)
print("Actual Labels:", y_valid)
✓ 0.0s
```

- Menampilkan hasil prediksi dan label aktual : menampilkan label dimana label berisi masing – masing sampel dalam prediksi.

Outout :

```
Predicted Labels: [4 5 4 ... 5 0 0]
Actual Labels: 5652      HOROZ
237          SEKER
5641         HOROZ
4525         CALI
6930         HOROZ
...
939          SEKER
10034        SIRA
369          SEKER
2546        BARBUNYA
3117        BARBUNYA
Name: Class, Length: 2723, dtype: object
```



```
# Menampilkan prediksi pada contoh data ke-10
print("Prediction for example 10:", y_pred[10])
print("Actual label for example 10:", y_valid.iloc[10])
```

✓ 0.0s

- Menampilkan prediksi dimana dalam contoh ini kami mengambil data ke-10 untuk prediksi.

Output :

```
Prediction for example 10: 3
Actual label for example 10: DERMASON
```

```
# Menampilkan evaluasi model pada data validasi secara keseluruhan
evaluations = model.evaluate(x_valid, y_valid_encoded)
print("Evaluation on validation data:", evaluations)
```

✓ 0.7s

- Menampilkan evaluasi model pada data validasi secara keseluruhan.

Output :

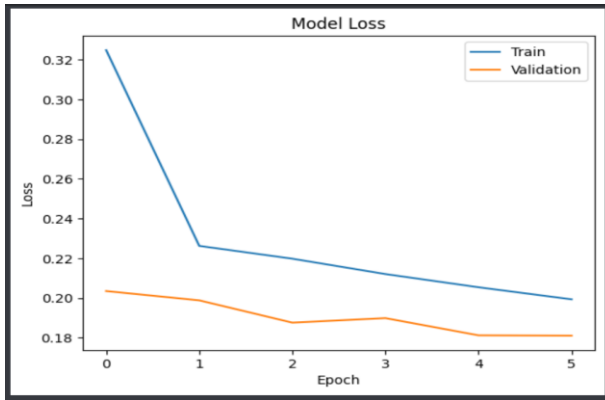
```
86/86 [=====] - 1s 6ms/step - loss: 0.2161 - accuracy: 0.9251
Evaluation on validation data: [0.21614713966846466, 0.9250826239585876]
```

```
# Plot loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```

✓ 0.1s

- Kode diatas berfungsi untuk membuat grafik pada fungsi loss.

Output :

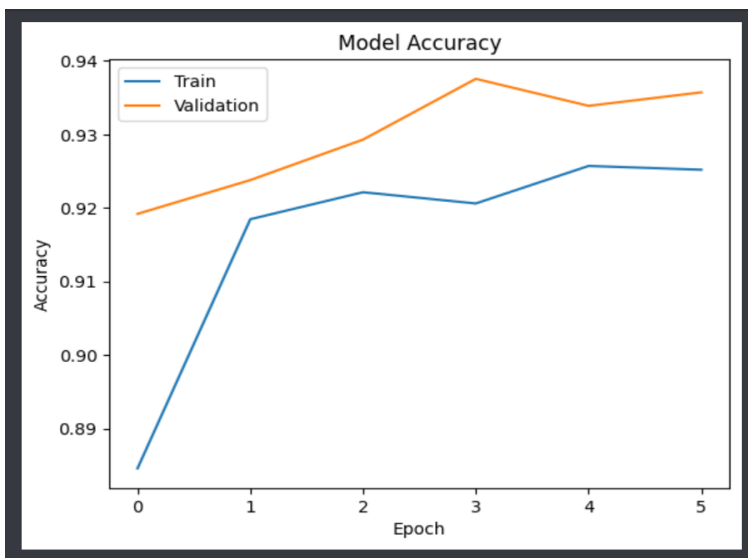


```
# Plot accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

✓ 0.2s

- Kode ini berfungsi untuk menampilkan grafik plot accuracy.

Output :



```
#MATRIKS VALIDATION

from sklearn.preprocessing import LabelEncoder

# Mengonversi label string menjadi angka
label_encoder = LabelEncoder()
y_valid_encoded = label_encoder.fit_transform(y_valid)

# Membuat classification report
report = get_classification_report(y_valid_encoded, y_pred_labels)
print(report)
```

✓ 0.0s

- Matriks validation : Classification report

Output :

	precision	recall	f1-score	support
0	0.89	0.93	0.91	276
1	1.00	1.00	1.00	88
2	0.94	0.91	0.92	322
3	0.89	0.94	0.92	690
4	0.98	0.94	0.96	378
5	0.95	0.96	0.95	452
6	0.91	0.85	0.88	517
accuracy			0.93	2723
macro avg	0.94	0.93	0.94	2723
weighted avg	0.93	0.93	0.92	2723

```

#CONFUSION MATRIKS

def plot_confusion_matrix(y_true, y_pred):
    # Menghitung confusion matrix
    cm = confusion_matrix(y_true, y_pred)

    # Menampilkan confusion matrix dalam bentuk heatmap
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Labels')
    plt.ylabel('Actual Labels')
    plt.show()

# Mengonversi label string menjadi angka
label_encoder = LabelEncoder()
y_valid_encoded = label_encoder.fit_transform(y_valid)

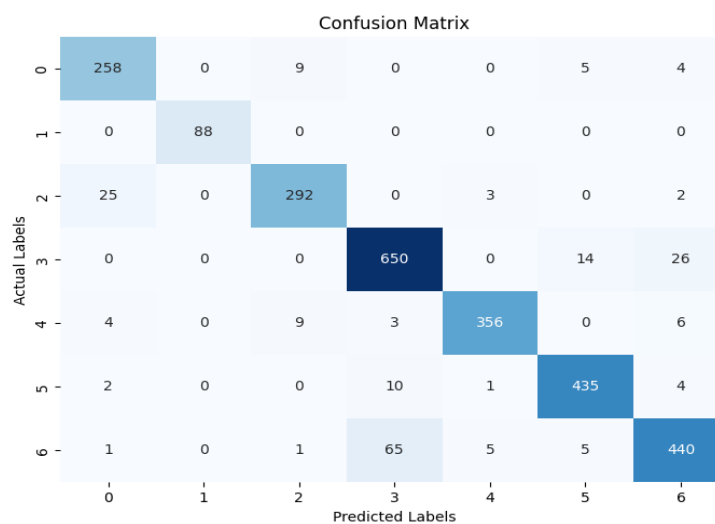
# penggunaan fungsi
y_pred = model.predict(x_valid) #hasil prediksi model
y_pred_labels = np.argmax(y_pred, axis=-1) # pemrosesan label prediksi
plot_confusion_matrix(y_valid_encoded, y_pred_labels)

```

✓ 0.3s

- Confussion Matriks : warna terang yaitu diagonal utama yang menunjukkan jumlah prediksi benar untuk setiap kelas, warna lebih gelap menunjukkan jumlah prediksi salah.

Output :



2.2.. Analisa Program

Pada pertemuan minggu pertama kami mengalami kesulitan, dimana adanya error. Sebab proses mencari best fitness dan best generation sangat lama dimana dalam 15 jam kami baru mendapat 2 generasi yang terlihat. Yang seharusnya muncul beberapa generasi yang kita atur yaitu 9 generasi dan mendapatkan bestfitness dan best generation. kami medapat beberapa solusi yaitu : mengganti source code, mencoba dengan device lain, memakai Google Colab, atau mencoba dengan bertahap, seperti 100 data, 200 data, 300, 1000, 2000, dst.

Solusi yang kami upayakan yaitu membuat menggunakan Google colab dan membagi data menjadi 100, 200, 300 ,1000 dan seterusnya sampai data yang kita gunakan yaitu 13611. Namun masih gagal karena proses masih sangat lama, dan mengalami kendala saat mencari fungsi loss, aktivasi, evaluasi, matric validation dan confussion matriks. selanjutnya mengubah source code secara total dimana dalam penelitian yang dilakukan didapat hasil bahwa algoritma genetika untuk seleksi beberapa fitur pada mungkin tidak efisien atau efektif secara umum.

Pada Algoritma Genetika disini dapat digunakan untuk mempelajari fitur atau representasi data yang optimal, yang kemudian dapat digunakan sebagai input bagi model pembelajaran mesin seperti Jaringan Saraf Tiruan (Artificial Neural Networks/ANN) atau Convolutional Neural Networks/CNN). ANN atau CNN adalah metode pembelajaran mesin yang mampu mengekstraksi pola yang kompleks dari data dan melakukan klasifikasi. dalam hal kami menggunakan ann untuk membantu proses klasifikasi.

Dari hasil program yang kami buat, dapat kami simpulkan metode Algoritma genetika bukan lah yang tepat, untuk data ini. Namun setelahnya kami memiliki sebuah pengalaman baru. Adapun kekurangan dan kelebihan pada project kali ini sebagai berikut :

- **Kekurangan:**

Program ini hanya memberikan contoh sederhana dan mungkin tidak mencakup semua kebutuhan atau kondisi khusus.

Penggunaan algoritma genetika untuk seleksi beberapa fitur pada mungkin tidak efisien atau efektif secara umum.

- **Kelebihan:**

Memberikan gambaran tentang bagaimana algoritma genetika dapat diimplementasikan untuk seleksi fitur.

BAB III

PENUTUP

3. Kesimpulan

membahas peran penting pertanian dan industri pangan dalam kehidupan manusia, khususnya pada biji kering sebagai komoditas utama. Identifikasi dan klasifikasi biji-bijian menjadi krusial untuk memastikan kualitas hasil pertanian. Pengembangan sistem visi komputer untuk mengklasifikasikan varietas biji kering dianggap penting untuk meningkatkan efisiensi dan akurasi. Penelitian ini memiliki tujuan mengembangkan sistem visi komputer berdasarkan analisis fitur biji untuk membedakan tujuh varietas biji kering.

Selanjutnya, pembahasan melibatkan penggunaan algoritma genetika sebagai solusi untuk masalah pemilihan fitur. Meskipun algoritma genetika efektif dalam masalah pencarian dan optimasi, kelemahannya adalah waktu komputasi yang lebih lama dibandingkan dengan algoritma K-means. Penggunaan algoritma genetika dalam penelitian ini dijelaskan sebagai upaya untuk meningkatkan kualitas produk pertanian dan efisiensi proses.

Langkah-langkah umum dalam algoritma genetika, seperti inisialisasi populasi, evaluasi fungsi fitness, seleksi, reproduksi, mutasi, dan iterasi. Terdapat pula perbandingan antara kelebihan dan kekurangan algoritma genetika dalam konteks penelitian ini. Meskipun tidak sepenuhnya cocok untuk data yang ada, penggunaan algoritma genetika memberikan gambaran tentang implementasinya dalam seleksi fitur.

Dari sini kami menyimpulkan bahwa penggunaan algoritma genetika dalam kasus ini mungkin tidak tepat, tetapi memberikan pengalaman baru. Kelebihannya mencakup memberikan gambaran implementasi algoritma genetika untuk seleksi fitur, sementara kekurangannya termasuk waktu komputasi yang lama dan keterbatasan efisiensi dalam kondisi tertentu.