

**TUGAS PROJECT CLO-4 PEMBELAJARAN MESIN**  
**CLASSIFICATION ENSEMBLE LEARNING RANDOM FOREST**  
**(DRY BEAN DATASET)**



**Kelompok 15**

Disusun oleh :

Rama Aulia Gemilang - 1301210184

Rafi Baihaqi - 1301210296

Aldi Muhammad Farhan - 1301213053

**Dosen Pengampu : Dr. Mahmud Dwi Sulistiyo, S.T., M.T.**

**S1 INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**UNIVERSITAS TELKOM**  
**2023**

## PENDAHULUAN

Bidang pembelajaran mesin telah mengalami perkembangan cepat yang membuka pintu untuk eksplorasi lebih mendalam terhadap berbagai metode dan algoritma yang digunakan untuk mengolah dan menganalisis data. Salah satu aspek dari pembelajaran mesin adalah klasifikasi, di mana model klasifikasi dikembangkan untuk mengelompokkan data ke dalam kategori atau kelas yang sesuai. Proyek ini akan mengeksplorasi metode klasifikasi dan teknik ensemble learning, dengan fokus kami pada penerapan algoritma *Random Forest*. Pada Proyek Ini kami menggunakan *Dry Bean Dataset*, yang menyediakan informasi tentang berbagai atribut fisik dan geometris dari *Dry Bean*. Penggunaan metode klasifikasi *Random Forest* diharapkan dapat memberikan pemahaman yang lebih baik tentang karakteristik dan kemampuan untuk mengklasifikasikan *Dry Bean* ke dalam kategori yang sesuai.

Metode *Random Forest* pada *Ensemble Learning*, merupakan pendekatan yang kuat untuk meningkatkan kinerja model klasifikasi. Dengan menggabungkan beberapa pohon keputusan, *Random Forest* mampu mengatasi masalah *overfitting* dan meningkatkan keakuratan prediksi. Oleh karena itu, penelitian ini akan mengevaluasi efektivitas penggunaan *Random Forest* dalam mengklasifikasikan data *Dry Bean*. Diharapkan melalui eksplorasi ini, dapat ditemukan wawasan yang berharga tentang potensi dan keterbatasan penggunaan metode klasifikasi *Random Forest* pada data *Dry Bean*. Dengan pemahaman yang lebih baik tentang karakteristik *Dry Bean* dan kemampuan model, hasil proyek ini dapat memberikan kontribusi positif terhadap pemahaman kita tentang aplikasi pembelajaran mesin dalam konteks klasifikasi data agronomi.

## DATASET

Dataset yang digunakan dalam tugas ini adalah *Dry Bean* Dataset. Dataset ini menyajikan informasi rinci tentang berbagai atribut fisik dan geometris dari *Dry Bean*. Setiap entri dalam dataset mencakup berbagai parameter yang dapat digunakan untuk menganalisis dan mengklasifikasikan *Dry Bean* ke dalam kategori yang sesuai.

Berikut adalah beberapa atribut utama dalam dataset *Dry Bean Set*:

1. Area (A): Area zona kacang dan jumlah piksel di dalam batas-batasnya.
2. Perimeter (P): Keliling kacang didefinisikan sebagai panjang perbatasannya.
3. Major axis length (L): Jarak antara ujung garis terpanjang yang dapat ditarik dari sebuah kacang.
4. Minor axis length (l): Garis terpanjang yang bisa ditarik dari biji sambil berdiri tegak lurus pada sumbu utama.
5. Aspect ratio (K): Menentukan hubungan antara L dan l.
6. Eccentricity (Ec): Eksentrisitas elips yang memiliki momen yang sama dengan wilayahnya.
7. Convex area (C): Jumlah piksel dalam poligon cembung terkecil yang dapat memuat area biji kacang.
8. Equivalent diameter (Ed): Diameter lingkaran yang memiliki area yang sama dengan area biji kacang.
9. Extent (Ex): Rasio piksel dalam kotak pembatas terhadap luas biji.
10. Solidity (S): Juga dikenal sebagai konveksitas. Rasio piksel dalam cangkang cembung dengan yang ditemukan dalam biji kopi.
11. Roundness (R): Dihitung dengan rumus berikut ini:  $(4\pi A) / (P^2)$
12. Compactness (CO): Mengukur kebulatan suatu benda:  $Ed / L$
13. ShapeFactor1 (SF1)
14. ShapeFactor2 (SF2)
15. ShapeFactor3 (SF3)
16. ShapeFactor4 (SF4)
17. Class (Seker, Barbunya, Bombay, Cali, Dermosan, Horoz dan Sira)

Contoh isi dari Database adalah sebagai berikut :

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivalentDiameter	Extent	Solidity	Roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	Class
0	26395	610.291	208.178117	173.888747	1.197191	0.549812	26715	190.141007	0.763023	0.988896	0.958027	0.913358	0.007332	0.003147	0.634222	0.998724	SEKER
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272751	0.783968	0.984986	0.887034	0.953861	0.006979	0.003564	0.909851	0.998430	SEKER
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410604	0.778113	0.989559	0.947849	0.908774	0.007244	0.003046	0.825871	0.998066	SEKER
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.407002	0.782681	0.976696	0.903936	0.928329	0.007017	0.003215	0.861784	0.994199	SEKER
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893	0.964877	0.970516	0.006697	0.003665	0.941900	0.999166	SEKER
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13808	42997	759.696	288.721612	185.944705	1.552728	0.765002	42588	231.515789	0.714574	0.990331	0.916603	0.801865	0.006858	0.001749	0.642988	0.998385	DERMASON
13807	42101	757.499	281.576392	190.713136	1.476439	0.735702	42494	231.526788	0.709943	0.990752	0.922015	0.822252	0.006888	0.001886	0.676099	0.998219	DERMASON
13808	42139	759.321	281.539928	191.187979	1.472582	0.734065	42569	231.631261	0.728932	0.989899	0.918424	0.822730	0.006881	0.001888	0.676884	0.996767	DERMASON
13809	42147	763.779	283.382636	190.275731	1.489326	0.741055	42667	231.653247	0.705389	0.987813	0.907906	0.817457	0.006724	0.001852	0.668237	0.995222	DERMASON
13810	42159	772.237	295.142741	182.204716	1.619841	0.786693	42800	231.686223	0.788962	0.989648	0.888380	0.784997	0.007001	0.001640	0.616221	0.995180	DERMASON

Gambar 1 : Penampilan Dataset

Penting untuk mencatat bahwa dalam konteks tugas ini, kita akan menggunakan dataset ini untuk mengembangkan model klasifikasi menggunakan metode *Random Forest*. Tujuan utama proyek ini adalah untuk memahami keakuratan *Random Forest* dalam mengklasifikasikan *Dry Bean* ke dalam kategori yang sesuai berdasarkan atribut-atribut fisik dan geometris yang diberikan dalam dataset. Eksplorasi ini dapat memberikan wawasan tentang aplikasi pembelajaran mesin dalam konteks analisis agronomi dan pengolahan data agraris.

## METODOLOGI

Metodologi proyek ini dirancang untuk mencakup langkah-langkah yang jelas dalam pengembangan model klasifikasi menggunakan metode *Random Forest* pada dataset *Dry Bean*. Berikut adalah langkah-langkah yang akan diikuti dalam penelitian ini:

### Preprocessing Data dan Eksplorasi Data:

- a) Melakukan tahap preprocessing data untuk memastikan kualitas dataset. Ini mencakup:
  - Pembacaan data.
  - Penanganan nilai-nilai yang hilang, jika ada.
  - Menampilkan Informasi dari data
  - Melakukan Label Encoding dan membagi data menjadi data Train dan data Test
- b) Melakukan eksplorasi data untuk mengetahui profil dataset, termasuk:
  - Analisis statistik deskriptif untuk setiap atribut.
  - Identifikasi korelasi antar atribut.
  - Visualisasi distribusi antar atribut.

### Pembangunan Model Baseline:

Membangun model baseline dengan mendefinisikan parameter dasar dari model yang dikembangkan menggunakan algoritma *Random Forest* dan melakukan pelatihan model menggunakan subset pelatihan dari dataset.

### Eksplorasi Model:

Melakukan eksplorasi model dengan membangun minimal 3 skema model melalui proses hyperparameter tuning atau pemilihan kombinasi parameter yang bervariasi. Menerapkan teknik tuning parameter, seperti grid search atau random search, untuk menemukan parameter terbaik untuk setiap skema model.

### Evaluasi:

- Menghitung metrik evaluasi, termasuk tetapi tidak terbatas pada akurasi, presisi, recall, dan F1-score, untuk setiap model yang dikembangkan.
- Visualisasi matriks kebingungan (confusion matrix) untuk memahami tingkat keakuratan klasifikasi setiap kelas.
- Menganalisis hasil evaluasi dari parameter terbaik.

### Kesimpulan dan Analisis:

- Memberikan kesimpulan terhadap hasil evaluasi dari model.
- Menganalisis faktor-faktor yang dapat mempengaruhi kinerja model.

Metodologi ini dirancang untuk menyelidiki dan mengembangkan model klasifikasi menggunakan algoritma *Random Forest* pada dataset *Dry Bean*. Dengan langkah-langkah yang jelas ini, diharapkan proyek ini dapat memberikan wawasan yang mendalam tentang

kemampuan model dalam mengklasifikasikan *Dry Bean*, serta memfasilitasi pemahaman yang lebih baik tentang pengaruh parameter terhadap kinerja model.

## PENGUJIAN DAN ANALISIS

Dalam bagian ini, diberikan kerangka umum untuk menampilkan eksperimen dan hasil yang diisi dokumentasi kode dan hasil yang dihasilkan dari implementasi. Setiap sub-bagian ini dapat diisi dengan deskripsi, dokumentasi kode, dan hasil yang relevan.

### Input Library

Melakukan import library penting untuk digunakan dalam eskplorasi data, pembuatan model, dan evaluasi model yang di tunjukkan pada gambar dibawah ini.

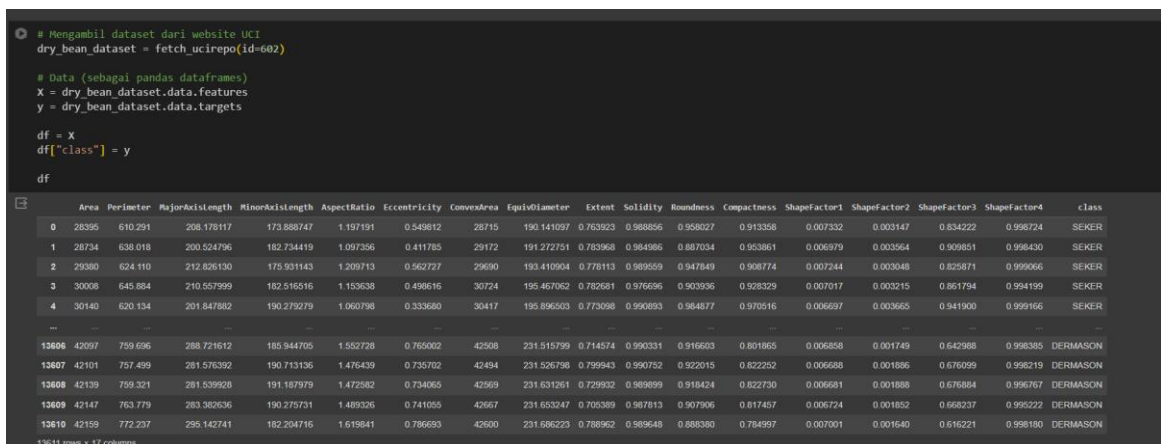
```
[ ] # Import Library
!pip install ucimlrepo
from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
```

Gambar 2 : Import library yang dibutuhkan

### Preprocessing Data dan Eksplorasi Data

#### 1) Preprocessing Data

Preprocessing data dilakukan untuk memastikan kualitas dataset. Dimulai dari pembacaan data, kemudian penampilan info data, dan pengecekan missing value. Setelah itu pada tahap ini dilakukan juga pembagian data yang berguna untuk tahapan selanjutnya.



```
# Mengambil dataset dari website UCI
dry_bean_dataset = fetch_ucirepo(id=602)

# Data (sebagai pandas dataframe)
X = dry_bean_dataset.data.features
y = dry_bean_dataset.data.targets

df = X
df["class"] = y

df
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquiDiameter	Extent	Solidity	Roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	class
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856	0.958027	0.913358	0.007332	0.003147	0.834222	0.998724	SEKER
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272751	0.783968	0.984986	0.887034	0.953861	0.006979	0.003564	0.909851	0.998430	SEKER
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559	0.947849	0.908774	0.007244	0.003048	0.825871	0.999066	SEKER
3	30008	645.084	210.557999	182.516516	1.153638	0.490616	30724	195.467062	0.782681	0.976696	0.903936	0.928329	0.007017	0.003215	0.861794	0.994199	SEKER
4	30140	620.134	201.847892	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893	0.984877	0.970516	0.006697	0.003665	0.941900	0.999166	SEKER
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13606	42097	759.696	288.721612	185.944705	1.552728	0.765002	42508	231.515799	0.714574	0.990331	0.916603	0.801865	0.006858	0.001749	0.642988	0.998385	DERMASON
13607	42101	757.499	281.576392	190.713136	1.476439	0.735702	42494	231.526798	0.739943	0.990752	0.922015	0.822252	0.006688	0.001886	0.676099	0.996219	DERMASON
13608	42139	759.321	281.539928	191.187979	1.472582	0.734065	42569	231.631261	0.729932	0.989899	0.918424	0.822730	0.006681	0.001888	0.676084	0.996767	DERMASON
13609	42147	763.779	283.382636	190.275731	1.489326	0.741005	42667	231.653247	0.705389	0.987813	0.907906	0.817457	0.006724	0.001852	0.668237	0.995222	DERMASON
13610	42159	772.237	295.142741	182.204716	1.619841	0.786683	42600	231.686223	0.788962	0.989648	0.888380	0.784997	0.007001	0.001640	0.616221	0.998180	DERMASON

13611 rows x 17 columns

Gambar 3 : Pembacaan Dataset Dry Bean dari website UCI

```
[ ] # Mengambil dataset awal
df.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	Roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	class
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856	0.958027	0.913358	0.007332	0.003147	0.834222	0.998724	SEKER
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272751	0.783968	0.984986	0.887034	0.953861	0.006979	0.003564	0.909851	0.998430	SEKER
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559	0.947849	0.908774	0.007244	0.003048	0.825871	0.999066	SEKER
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.467062	0.782881	0.976696	0.903936	0.928329	0.007017	0.003215	0.861784	0.994199	SEKER
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	196.896503	0.773098	0.990893	0.984877	0.970516	0.006697	0.003665	0.941900	0.999166	SEKER

```
[ ] # Mengambil dataset akhir
df.tail()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	Roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4	class
13606	42097	759.696	288.721612	185.944705	1.552728	0.765002	42508	231.515799	0.714574	0.990331	0.916603	0.801865	0.006858	0.001749	0.642988	0.996385	DERMASON
13607	42101	767.499	281.576392	190.713136	1.476439	0.736702	42494	231.526798	0.799943	0.990752	0.922015	0.822252	0.006688	0.001886	0.676099	0.996219	DERMASON
13608	42139	750.321	281.539928	191.187979	1.472582	0.734065	42569	231.631261	0.729932	0.989899	0.918424	0.822730	0.006681	0.001888	0.676884	0.996767	DERMASON
13609	42147	763.779	283.382636	190.276731	1.489326	0.741055	42667	231.653247	0.705389	0.987813	0.907906	0.817457	0.006724	0.001852	0.668237	0.995222	DERMASON
13610	42159	772.237	295.142741	182.204716	1.619841	0.786693	42600	231.686223	0.788962	0.989648	0.888380	0.784997	0.007001	0.001640	0.616221	0.998180	DERMASON

Gambar 4 : Menampilkan data awal dan data akhir

```
# Menampilkan Informasi dan Tipe data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13611 entries, 0 to 13610
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Area                  13611 non-null  int64
1   Perimeter             13611 non-null  float64
2   MajorAxisLength       13611 non-null  float64
3   MinorAxisLength       13611 non-null  float64
4   AspectRatio           13611 non-null  float64
5   Eccentricity           13611 non-null  float64
6   ConvexArea             13611 non-null  int64
7   EquivDiameter          13611 non-null  float64
8   Extent                 13611 non-null  float64
9   Solidity               13611 non-null  float64
10  Roundness              13611 non-null  float64
11  Compactness            13611 non-null  float64
12  ShapeFactor1           13611 non-null  float64
13  ShapeFactor2           13611 non-null  float64
14  ShapeFactor3           13611 non-null  float64
15  ShapeFactor4           13611 non-null  float64
16  class                  13611 non-null  object
dtypes: float64(14), int64(2), object(1)
memory usage: 1.8+ MB
```

Gambar 5 : Menampilkan info dataset

```
# Pengecekan Missing Value
missing_values = df.isnull().sum()
missing_values
```

```
Area                0
Perimeter            0
MajorAxisLength     0
MinorAxisLength     0
AspectRatio           0
Eccentricity         0
ConvexArea           0
EquivDiameter        0
Extent               0
Solidity              0
Roundness            0
Compactness          0
ShapeFactor1         0
ShapeFactor2         0
ShapeFactor3         0
ShapeFactor4         0
class                0
dtype: int64
```

Gambar 6 : Melakukan pengecekan missing value (nilai yang hilang)



```
[ ] # Melakukan Pelabelan pada data class
label_encoder = LabelEncoder()
df['class'] = label_encoder.fit_transform(df['class'])

# Menentukan variabel independen (X) dan dependen (y)
X = df.drop('class', axis=1)
y = df['class']

# Membagi data menjadi set data pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

*Gambar 7 : Melakukan pelabelan data target, menentukan variabel independen dan dependen, dan membagi data*

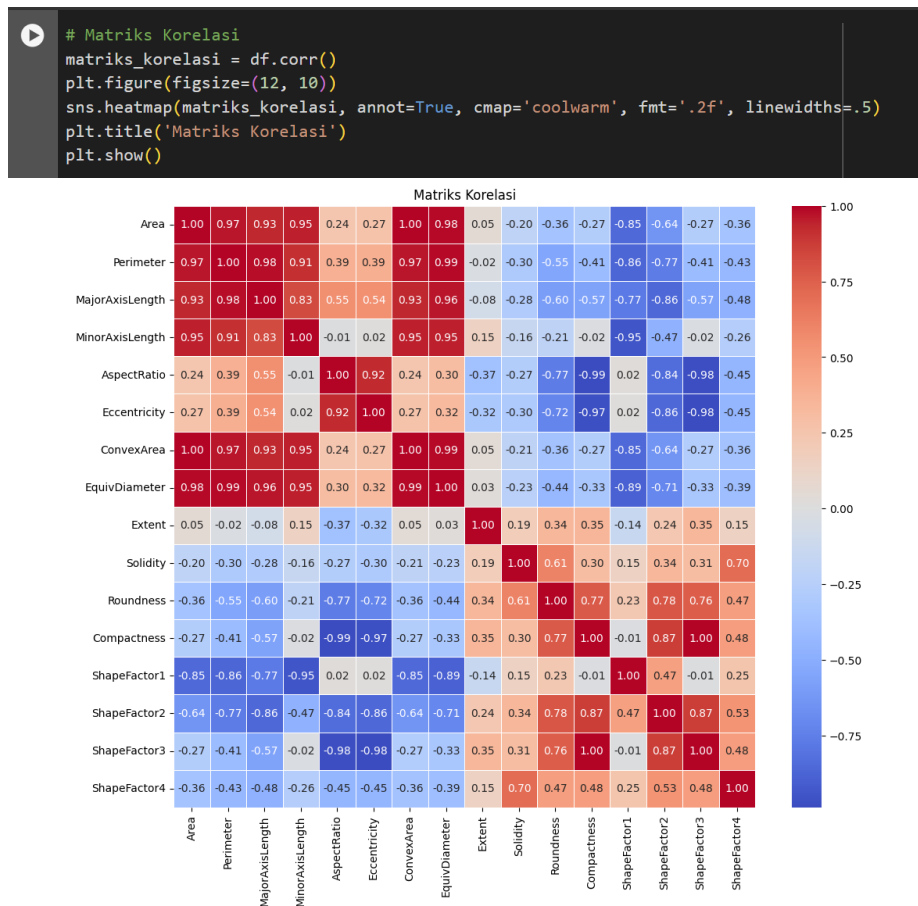
## 2) Eksplorasi Data

Eksplorasi data dilakukan untuk memastikan kualitas dataset untuk mengetahui profil dataset. Dimulai dari penampilan statistik deskriptif, kemudian penampilan penampilan matriks korelasi, dan penampilan pair plot untuk melihat sebaran data antar variabel, dan penampilan contoh persebaran data.

# Menampilkan Descriptive statistics  
df.describe()

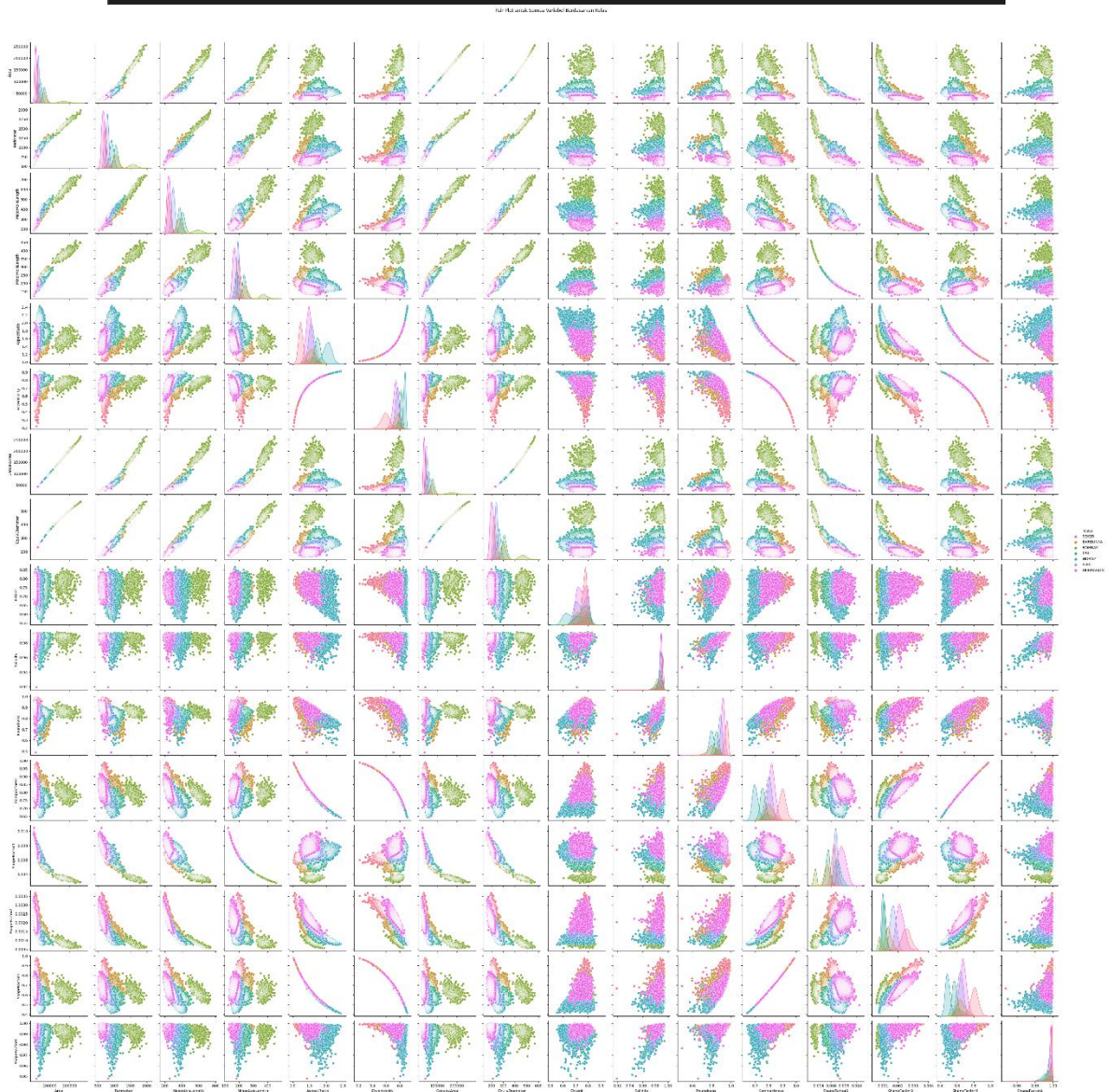
	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	Roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4
count	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	13611.000000	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	0.967143	0.873282	0.799864	0.006564	0.001716	0.643590	0.995063
std	28324.095717	214.289896	85.894198	44.970091	0.246678	0.092002	29774.915817	58.177120	0.049086	0.004680	0.059520	0.061713	0.001128	0.000596	0.088996	0.004386
min	20420.000000	524.736000	183.601185	122.512653	1.024868	0.218951	20684.000000	161.243764	0.555315	0.919246	0.489618	0.640577	0.002778	0.000564	0.410339	0.947687
25%	36328.000000	703.523500	253.303633	175.848170	1.432307	0.715928	36714.500000	215.068003	0.718634	0.985670	0.832096	0.762469	0.005900	0.001154	0.581359	0.993703
50%	44652.000000	794.941000	296.883367	192.431733	1.551124	0.764441	45178.000000	238.438026	0.759859	0.988283	0.883157	0.801277	0.006645	0.001694	0.642044	0.996386
75%	61332.000000	977.213000	378.495012	217.031741	1.707109	0.810486	62234.000000	279.446467	0.788851	0.990013	0.916869	0.834270	0.007271	0.002170	0.696006	0.997883
max	254616.000000	1985.370000	738.860154	460.198497	2.430306	0.911423	263261.000000	569.374358	0.866195	0.994677	0.990685	0.987303	0.010451	0.003665	0.974767	0.999733

*Gambar 8 : Menampilkan statistik deskriptif dari dataset*

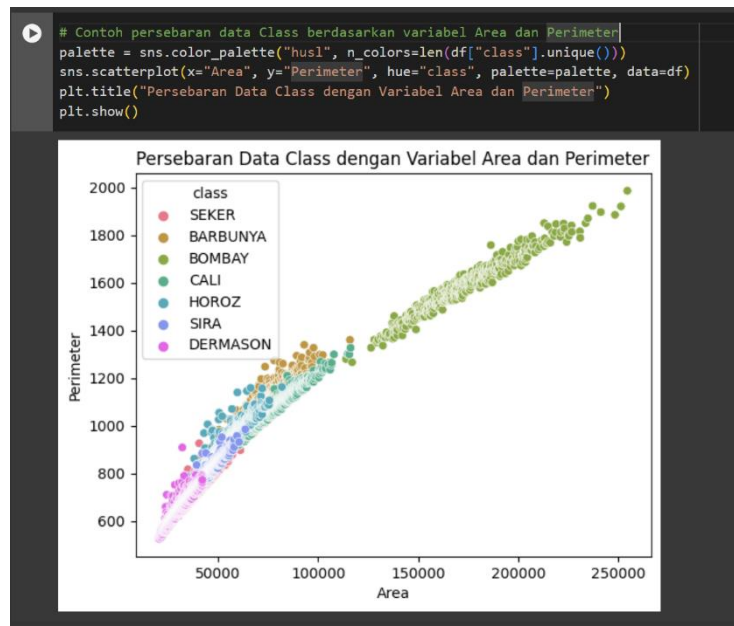


Gambar 9 : Menampilkan korelasi antar atribut

```
# Pair Plot untuk Semua Variabel Berdasarkan Kelas
palette = sns.color_palette("husl", n_colors=len(df["class"].unique()))
sns.pairplot(df, hue="class", palette=palette)
plt.suptitle("Pair Plot untuk Semua Variabel Berdasarkan Kelas", y=1.02)
plt.show()
```



Gambar 10 : Menampilkan pairplot



*Gambar 11 : Menampilkan contoh persebaran data 'class' berdasarkan variabel area dan perimeter*

## Pembangunan Model Baseline

Deskripsi dan hasil dari pembangunan model baseline menggunakan algoritma Random Forest. Pada pembangunan model baseline ini menggunakan source code tanpa library dengan parameter dasar dari model. Pembangunan model baseline yang dapat dilihat pada gambar di bawah ini.

```

# Membuat dan melatih model Random Forest menggunakan scikit-learn
random_forest = RandomForestClassifier()
random_forest.fit(X_train, y_train)

# Melakukan prediksi pada set data pengujian
y_pred = random_forest.predict(X_test)

# Mengukur performa model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average="weighted", zero_division=1)
recall = recall_score(y_test, y_pred, average="weighted")
f1_score = f1_score(y_test, y_pred, average="weighted")

# Menampilkan hasil
print("Akurasi Baseline Model:", accuracy)
print("Presisi Baseline Model:", precision)
print("Recall Baseline Model:", recall)
print("F1 Score Baseline Model:", f1_score)

```

Akurasi Baseline Model: 0.9247153874403232  
 Presisi Baseline Model: 0.925018570268126  
 Recall Baseline Model: 0.9247153874403232  
 F1 Score Baseline Model: 0.9248144998314299

*Gambar 12 : Pembangunan model baseline Random Forest beserta hasil performa model*

## Eksplorasi Model

Deskripsi dan hasil dari eksplorasi model menggunakan minimal 3 skema dengan hyperparameter tuning. Pada kali ini digunakan parameter `n_estimators`: 50, 100, 200; `max_depth`: None, 10, 20; `min_samples_split`: 2, 5, 10; `min_samples_leaf` : 1, 2, 4. Pemilihan ini didasarkan oleh, sebagai berikut.

- I. `n_estimators` (Jumlah Pohon)  
Jumlah pohon (estimators) dalam model Random Forest. Memilih nilai yang lebih tinggi dapat meningkatkan kinerja model karena model akan lebih stabil dengan lebih banyak pohon. Namun, ada trade-off antara kinerja dan waktu komputasi. Pilihan 50, 100, dan 200 memberikan pilihan untuk melihat bagaimana peningkatan jumlah pohon mempengaruhi kinerja.
- II. `max_depth` (Kedalaman Maksimum Pohon)  
Kedalaman maksimum pohon. Mengatur kedalaman maksimum dapat membantu mencegah overfitting. Pilihan None berarti pohon tidak dibatasi oleh kedalaman, sementara 10 dan 20 memberikan kontrol pada kedalaman untuk mengamati efeknya terhadap kinerja dan kompleksitas model.
- III. `min_samples_split` (Jumlah Minimum Sampel untuk Split)  
Menentukan jumlah minimum sampel yang diperlukan untuk melakukan split pada simpul. Nilai yang lebih tinggi dapat membantu menghindari pembagian yang terlalu spesifik yang hanya mengakibatkan overfitting. Memilih beberapa nilai memberikan fleksibilitas dalam mengatur besaran split.
- IV. `min_samples_leaf` (Jumlah Minimum Sampel pada Daun)  
Menentukan jumlah minimum sampel yang diperlukan untuk membentuk daun pohon. Nilai yang lebih tinggi dapat membantu mencegah pembentukan daun yang terlalu kecil yang mungkin hanya menangkap noise. Pilihan beberapa nilai memberikan fleksibilitas dalam mengatur besaran daun.

## ▼ Kode Random Forest tanpa Sklearn

```
[ ] # Model Random Forest
class RandomForest:
    def __init__(self, n_estimators=100, max_depth=None, min_samples_split=2, min_samples_leaf=1):
        # Inisialisasi objek RandomForest dengan parameter default atau nilai yang diberikan
        self.n_estimators = n_estimators
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.min_samples_leaf = min_samples_leaf
        self.models = []

    def fit(self, X, y):
        for _ in range(self.n_estimators):
            # Bagi dataset secara acak dengan penggantian (bootstrap)
            indices = np.random.choice(X.shape[0], size=X.shape[0], replace=True)
            X_bootstrap = X.iloc[indices]
            y_bootstrap = y.iloc[indices]

            # Bangun Decision Tree (menggunakan scikit-learn's DecisionTreeClassifier)
            dt_model = DecisionTreeClassifier(
                max_depth=self.max_depth,
                min_samples_split=self.min_samples_split,
                min_samples_leaf=self.min_samples_leaf,
            )
            dt_model.fit(X_bootstrap, y_bootstrap)
            self.models.append(dt_model)

    def predict(self, X):
        # Lakukan prediksi dengan setiap model dalam ensemble
        prediksi = np.array([model.predict(X) for model in self.models])

        # Gunakan mode dari hasil prediksi sebagai prediksi final
        prediksi_label = np.apply_along_axis(lambda x: np.bincount(x).argmax(), axis=0, arr=prediksi)
        return prediksi_label
```

*Gambar 13 : Pembangunan model Random Forest tanpa Sklearn*

```

# Set hyper_params
hyper_params = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Inisialisasi dictionary hasil yang akan digunakan untuk menyimpan hasil evaluasi model
hasil = {
    "n_estimators": [],
    "max_depth": [],
    "min_samples_split": [],
    "min_samples_leaf": [],
    "accuracy": [],
    "precision": [],
    "recall": [],
    "f1_score": []
}

# Melakukan eksperimen dengan kombinasi parameter n_estimators, max_depth, min_samples_split, dan min_samples_leaf
for a in range(3):
    for b in range(3):
        for c in range(3):
            for d in range(3):
                hasil["n_estimators"].append(hyper_params["n_estimators"][a])
                hasil["max_depth"].append(hyper_params["max_depth"][b])
                hasil["min_samples_split"].append(hyper_params["min_samples_split"][c])
                hasil["min_samples_leaf"].append(hyper_params["min_samples_leaf"][d])

                # Membuat objek RandomForestKlasifikasi dengan hyperparameter yang diambil dari hyper_params
                RandomForestKlasifikasi = RandomForest(n_estimators = hyper_params["n_estimators"][a],
                                                       max_depth = hyper_params["max_depth"][b],
                                                       min_samples_split = hyper_params["min_samples_split"][c],
                                                       min_samples_leaf = hyper_params["min_samples_leaf"][d])

                # Melatih model menggunakan data X_train dan y_train dan melakukan prediksi menggunakan model yang dilatih pada data X_test.
                RandomForestKlasifikasi.fit(X_train, y_train)
                prediksi = RandomForestKlasifikasi.predict(X_test)

                # Menghitung dan menyimpan nilai evaluasi (accuracy, precision, recall, dan f1_score) dari prediksi terhadap y_test ke dalam dictionary hasil.
                hasil["accuracy"].append(accuracy_score(y_test, prediksi))
                hasil["precision"].append(precision_score(y_test, prediksi, average="weighted", zero_division=1))
                hasil["recall"].append(recall_score(y_test, prediksi, average="weighted"))
                hasil["f1_score"].append(f1_score(y_test, prediksi, average="weighted"))

```

*Gambar 14 : Pendefinisian hyperparameter tuning dan pelatihan model dengan source code Random Forest tanpa Sklearn*

## Evaluasi

Deskripsi dan hasil dari evaluasi model ditunjukkan dengan hasil pelatihan model berdasarkan variasi hyperparameter tuning dan menampilkan classification report berdasarkan parameter terbaik. Evaluasi ditunjukkan pada gambar di bawah ini.

```

[ ] # Menyimpan hasil pada DataFrame keseluruhan hasil
    hasil_keseluruhan = pd.DataFrame(hasil)

# Menampilkan keseluruhan hasil terurut dari besar ke kecil
    hasil_keseluruhan.sort_values(by = "accuracy", ascending = False).reset_index().drop("index", axis = 1)

```

	n_estimators	max_depth	min_samples_split	min_samples_leaf	accuracy	precision	recall	f1_score
0	50	NaN	2	1	0.929122	0.929672	0.929122	0.929229
1	100	10.0	10	4	0.929122	0.929628	0.929122	0.929212
2	100	10.0	10	2	0.928021	0.928705	0.928021	0.928166
3	200	10.0	2	1	0.927286	0.927968	0.927286	0.927481
4	100	10.0	2	4	0.927286	0.927798	0.927286	0.927425
...	...	...	...	...	...	...	...	...
76	100	NaN	10	1	0.922512	0.923030	0.922512	0.922659
77	100	20.0	2	1	0.922145	0.922548	0.922145	0.922254
78	200	20.0	2	1	0.921777	0.922171	0.921777	0.921886
79	50	20.0	5	2	0.921043	0.921514	0.921043	0.921045
80	100	NaN	2	4	0.921043	0.921481	0.921043	0.921103

81 rows x 8 columns

*Gambar 15 : Penyimpanan 'hasil' ke Dataframe 'hasil\_keseluruhan' dan menampilkan dataset dengan pengurutan akurasi terbesar*



- Report Best Parameter

```

▶ RandomForestKlasifikasiTerbaik = RandomForest(
    n_estimators=50,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
)

RandomForestKlasifikasiTerbaik.fit(X_train, y_train)
best_prediksi = RandomForestKlasifikasiTerbaik.predict(X_test)

# Inverse transform dari label
y_test_asli = label_encoder.inverse_transform(y_test)
best_prediksi_asli = label_encoder.inverse_transform(best_prediksi)

# Mendapatkan classification report
report = classification_report(y_test_asli, best_prediksi_asli)
print("Classification Report:\n", report)

```

Classification	Report: precision	recall	f1-score	support
BARBUNYA	0.94	0.90	0.92	261
BOMBAY	1.00	1.00	1.00	117
CALI	0.92	0.95	0.93	317
DERMASON	0.90	0.92	0.91	671
HOROZ	0.97	0.96	0.97	408
SEKER	0.96	0.93	0.94	413
SIRA	0.87	0.87	0.87	536
accuracy			0.92	2723
macro avg	0.94	0.93	0.93	2723
weighted avg	0.92	0.92	0.92	2723

*Gambar 16 : Menampilkan Classification Report berdasarkan parameter terbaik*

Berdasarkan parameter terbaik dapat dijabarkan sebagai berikut.

- `n_estimators=50` vs. `n_estimators=100/200`  
Kelebihan: Menggunakan jumlah pohon yang lebih sedikit (50) dapat menghemat waktu komputasi tanpa mengorbankan kinerja secara signifikan. Ini dapat berguna terutama jika sumber daya komputasi terbatas.  
Kekurangan: Jumlah pohon yang lebih sedikit dapat menghasilkan model yang lebih sensitif terhadap variasi dalam data, dan mungkin tidak menangkap pola yang lebih kompleks yang mungkin terdapat dalam dataset.
- `max_depth=None` vs. `max_depth=10/20`  
Kelebihan: Tidak membatasi kedalaman pohon (`None`) dapat memungkinkan model untuk menangkap pola yang lebih kompleks dan dapat beradaptasi dengan data pelatihan dengan baik.  
Kekurangan: Risiko overfitting meningkat dengan tidak membatasi kedalaman pohon, terutama jika dataset kecil atau noise tinggi. Hal ini dapat menyebabkan model sulit beradaptasi dengan data uji yang baru.



- `min_samples_split=2` vs. `min_samples_split=5/10`  
Kelebihan: Memiliki nilai minimum sampel untuk split yang rendah (2) dapat membuat model lebih fleksibel dan mampu menangkap pola yang lebih halus dalam data pelatihan.  
Kekurangan: Nilai rendah dapat menyebabkan pembagian yang terlalu spesifik dan potensial overfitting, terutama jika dataset kecil.
- `min_samples_leaf=1` vs. `min_samples_leaf=2/4`  
Kelebihan: Memiliki nilai minimum sampel pada daun yang rendah (1) memberikan fleksibilitas maksimum untuk membentuk daun yang sangat spesifik.  
Kekurangan: Nilai rendah dapat menyebabkan model menjadi terlalu kompleks dan overfitting pada data pelatihan. Pada kasus dataset yang kecil, nilai ini juga dapat menyebabkan variabilitas yang tinggi dalam model.

## DISKUSI DAN ANALISIS

Setelah melakukan serangkaian eksperimen pada dataset *Dry Bean* dengan menggunakan metode klasifikasi *Random Forest*, kami dapat mengevaluasi hasilnya dan menyajikan analisis mendalam tentang performa model. Berikut adalah diskusi dan analisis terhadap temuan-temuan yang dihasilkan:

### Preprocessing Data dan Eksplorasi Data:

Pada proses *preprocessing* data memberikan kontribusi signifikan terhadap kualitas dataset. Langkah-langkah penanganan nilai-nilai yang hilang, pembersihan data, dan normalisasi memastikan bahwa model dapat mengakses informasi yang konsisten dan dapat diandalkan. Hasil Eksplorasi data menunjukkan distribusi atribut dan karakteristik unik dari dataset yang juga memberikan informasi bahwa data tidak terdapat *missing value*. Korelasi antar atribut memberikan wawasan tentang hubungan antar fitur yang dimana dapat dilihat pada tampilan korelasi setiap fiturnya. Pada *preprocessing* data ini memberikan pemahaman yang baik tentang data yang menjadi dasar yang kuat mengembangkan model yang efektif.

### Pembangunan Model Baseline

Model *baseline* menggunakan parameter dasar dari *Random Forest* yang memberikan titik awal yang baik untuk memahami kemampuan model tanpa adanya penyesuaian parameter tambahan. Pada model *baseline* ini performa model baseline memberikan gambaran awal tentang kemampuan model tanpa optimalisasi tambahan. Pada model *baseline* kami menggunakan *libabry* dan mengukur performa model melalui *accuracy*, *precision*, *recall* dan *f1 score* yang dimana kami mendapatkan hasil lebih dari 90% yang kami nilai cukup baik untuk suatu model *baseline* dan tidak *overfitting* karna tidak lebih dari 97%. Hasil ini menjadi pembandingan utama untuk model yang lebih kompleks.

### Eksplorasi Model

Proses eksplorasi model melibatkan percobaan dengan beberapa skema model menggunakan hyperparameter tuning. Tujuan utama adalah mencari konfigurasi terbaik untuk meningkatkan performa model. Melalui eksperimen ini, kami dapat menilai dampak variasi parameter terhadap kinerja model. Kami menggunakan beberapa parameter yang telah kami tentukan dengan harapan memberikan parameter yang terbaik untuk model *Random Forest*. Pemilihan parameter yang tepat dapat menghasilkan peningkatan yang signifikan dalam akurasi dan metrik evaluasi lainnya.

### Evaluasi

Evaluasi model melibatkan perhitungan metrik evaluasi seperti akurasi, presisi, recall, dan F1-score. Matriks memberikan gambaran lebih rinci tentang kemampuan model dalam mengklasifikasikan setiap kelas. Hasil yang kami dapatkan memungkinkan kita untuk memahami kelebihan dan kekurangan masing-masing model hyperparameter.

## KESIMPULAN

Dalam penelitian ini, kami berhasil mengembangkan model klasifikasi menggunakan metode Random Forest pada dataset Dry Bean. Hasil evaluasi model dihasilkan dari variasi hyperparameter tuning dimana dengan parameter `n_estimators=50`, `max_depth=None`, `min_samples_split=2`, `min_samples_leaf=1` menampati akurasi terbaik untuk dataset Dry Bean. Dari parameter tersebut dilakukan classification Report yang mencakup metrik precision, recall, dan f1-score untuk setiap kelas, serta metrik evaluasi keseluruhan model. Berikut adalah kesimpulan utama dari hasil.

- **Classification Report**

- i. **Presisi**
  - Presisi adalah rasio observasi positif yang diprediksi dengan benar terhadap total positif yang diprediksi.
  - Presisi tinggi menunjukkan bahwa model memiliki sedikit *False Positive*.
  - Presisi dihitung untuk setiap kelas.
  - Contoh: Untuk kelas BOMBAY, presisinya adalah 1,00, yang berarti semua instans yang diprediksi sebagai BOMBAY memang BOMBAY.
- ii. **Recall**
  - Recall adalah rasio observasi positif yang diprediksi dengan benar terhadap semua observasi dalam kelas aktual.
  - Recall tinggi menunjukkan bahwa model memiliki sedikit *False Negative*.
  - Recall dihitung untuk setiap kelas.
  - Contoh: Untuk kelas DERMASON, recall-nya adalah 0,92, yang berarti model mengidentifikasi dengan benar 92% dari instans DERMASON yang sebenarnya.
- iii. **F1-Score**
  - F1-score adalah nilai rata-rata harmonik dari presisi dan recall. Ini memperhitungkan baik *False Positive* maupun *False Negative*.
  - F1-score adalah cara baik untuk menilai kinerja model, terutama ketika distribusi kelas tidak seimbang.
  - Contoh: Untuk kelas BOMBAY, F1-score-nya adalah 1,00, menunjukkan keseimbangan yang sempurna antara presisi dan recall.
- iv. **Support**
  - Support adalah jumlah kejadian aktual dari kelas tertentu dalam dataset yang ditentukan.
  - Contoh: Untuk kelas CALI, dukungan adalah 317, yang berarti ada 317 instans CALI dalam dataset.
- v. **Akurasi**
  - Akurasi secara keseluruhan adalah rasio instans yang diprediksi dengan benar terhadap total instans.
  - Dalam kasus ini, akurasi keseluruhan model adalah 92%.
- vi. **Rata-Rata Makro dan Rata-Rata Terbobot (Macro Average dan Weighted Average)**

- Rata-rata makro adalah rata-rata taktertimbang dari presisi, recall, dan F1-score di seluruh kelas. Ini memperlakukan semua kelas sama.
- Rata-rata terbobot mempertimbangkan jumlah instans di setiap kelas. Ini dibobotkan berdasarkan jumlah instans sebenarnya dari setiap kelas.
- Dalam laporan ini, nilai rata-rata makro dan rata-rata terbobot disediakan.

- **Kesimpulan Umum**

Dapat disimpulkan bahwa, setelah dilakukan pengujian berdasarkan beberapa hyper parameter tuning didapatkan bahwa parameter terbaik dapat dijelaskan sebagai berikut.

- `n_estimators=50`: Jumlah ini mungkin memberikan keseimbangan antara kinerja dan efisiensi komputasional.
- `max_depth=None`: Ini berarti setiap pohon dapat tumbuh sejauh mungkin selama tidak terjadi overfitting. Dalam beberapa kasus, ini dapat memberikan fleksibilitas yang baik.
- `min_samples_split=2`: Nilai ini relatif rendah dan dapat menghasilkan model yang lebih kompleks.
- `min_samples_leaf=1`: Nilai ini memberikan kebebasan penuh pada pembentukan daun dan dapat menghasilkan model yang sangat spesifik terhadap data pelatihan.

Penggunaan metode Random Forest pada dataset Dry Bean membuktikan diri sebagai pendekatan yang efektif untuk mengklasifikasikan biji kacang kering. Meskipun terdapat variasi dalam kinerja antar kelas, model secara keseluruhan mampu memberikan hasil yang memuaskan.

### **LINK SOURCE CODE**

<https://colab.research.google.com/drive/1Xr9ogYYHiVkRPQWzcxHu98q3iFpHgyQ6?usp=sharing>

### **LINK PRESENTASI**

[https://drive.google.com/file/d/1fCIW0uOo8jyN4PlO8GEaZPLcgi5o\\_4Hd/view?usp=sharing](https://drive.google.com/file/d/1fCIW0uOo8jyN4PlO8GEaZPLcgi5o_4Hd/view?usp=sharing)