LAMPIRAN

LAMPIRAN 1 SURAT KETERANGAN HASIL PENGECEKAN TURNITIN

UNIVERSITAS DARMA PERSADA
UPT PERPUSTAKAAN
Gedung Rektorat Lantai 3,
Jl.Taman Malaka Selatan, Pondok Kelapa – Jakarta Timur 13450

SURAT KETERANGAN
HASIL PENGECEKAN TURNITIN

UPT Perpustakaan Universitas Darma Persada menerangkan telah selesai melakukan pemeriksaan duplikasi/*similarity* menggunakan perangkat lunak Turnitin terhadap hasil karya sebagai berikut:

Judul : Implementasi Klasifikasi Tanaman Aglaonema Menggunakan Metode You Only Look Once (YOLO) Dan Single Shot Multibox Detector (SSD)

Penulis : Riska Larasati

NIM : 2018230099

Tgl pemeriksaan : 11 Juli 2024

Dengan hasil Tingkat Kesamaan (*similarity index*) 22%

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

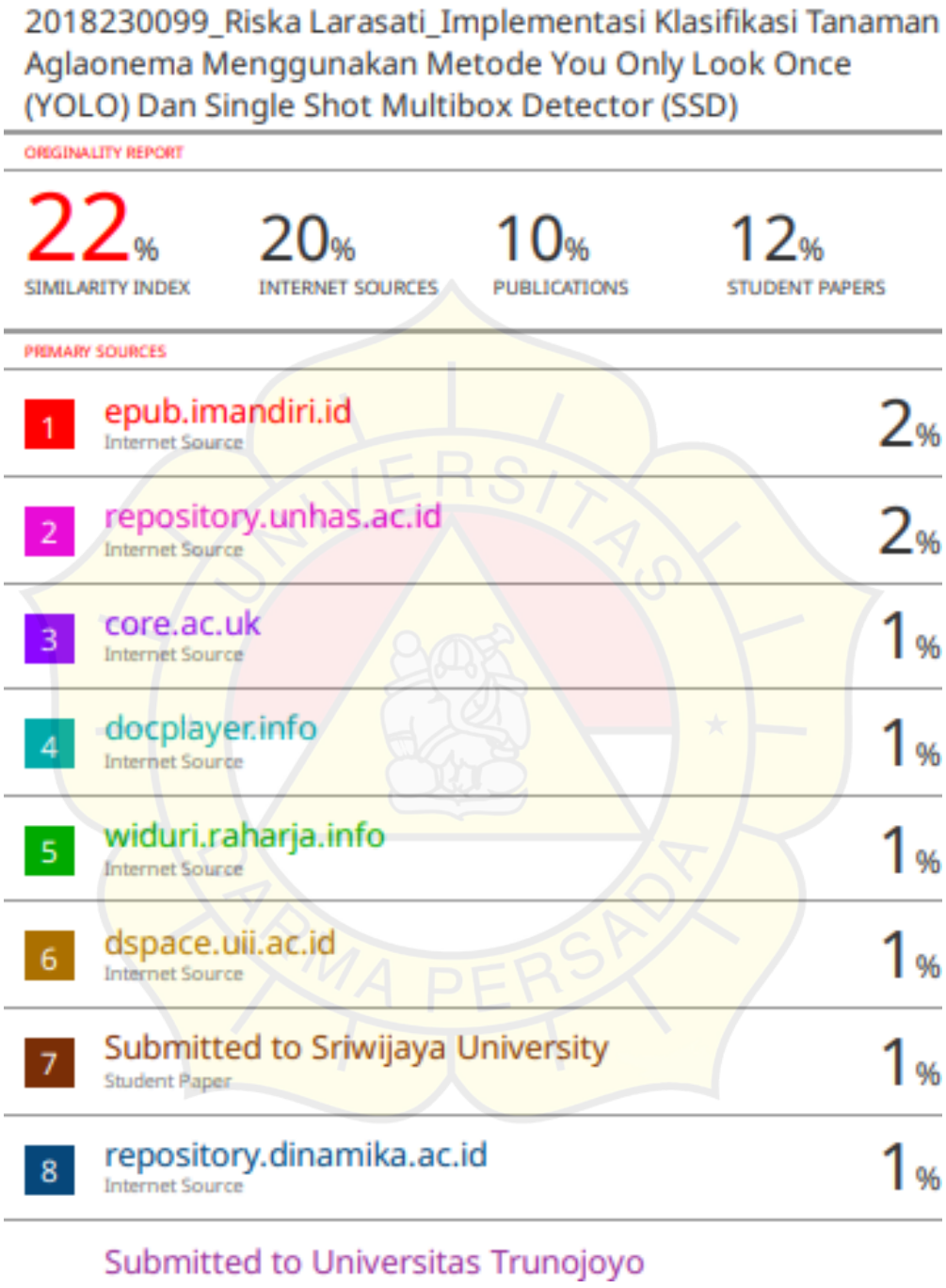
Jakarta, 11 Juli 2024

Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi
Batas maksimal similarity 25% untuk Fakultas Teknik, Kelautan dan Pasca Sarjana

LAMPIRAN 2 HASIL PENGECEKAN TURNITIN



2018230099_Riska Larasati_Implementasi Klasifikasi Tanaman Aglaonema Menggunakan Metode You Only Look Once (YOLO) Dan Single Shot Multibox Detector (SSD)

ORIGINALITY REPORT

22%	20%	10%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	epub.imandiri.id Internet Source	2%
2	repository.unhas.ac.id Internet Source	2%
3	core.ac.uk Internet Source	1%
4	docplayer.info Internet Source	1%
5	widuri.raharja.info Internet Source	1%
6	dspace.uii.ac.id Internet Source	1%
7	Submitted to Sriwijaya University Student Paper	1%
8	repository.dinamika.ac.id Internet Source	1%

Submitted to Universitas Trunojoyo

LAMPIRAN 3 DATABASE CONNECTION

```
<?php
$host = "localhost";
$username = "root";
$password = "";
$database = "db_esp";

$conn = new mysqli($host, $username, $password, $database);

if ($conn->connect_error) {
    die("Koneksi database gagal: " . $conn->connect_error);
}

$nama_sensor = isset($_GET['nama_sensor']) ? mysqli_real_escape_string($conn,
$_GET['nama_sensor']) : '';
$nilai_sensor = isset($_GET['nilai_sensor']) ?
mysqli_real_escape_string($conn, $_GET['nilai_sensor']) : '';

if (!empty($nama_sensor) && !empty($nilai_sensor)) {
    // Insert data ke tabel data_sensor
    $sql_insert = "INSERT INTO data_sensor (nama_sensor, nilai_sensor) VALUES
('$nama_sensor', '$nilai_sensor')";
    $conn->query($sql_insert);

    // Update atau insert data ke tabel data_akumulasi
    $sql_check = "SELECT * FROM data_akumulasi WHERE nama_sensor =
'$nama_sensor'";
    $result_check = $conn->query($sql_check);

    if ($result_check->num_rows > 0) {
        // Jika sudah ada entri, update nilai_akumulasi
        $sql_update = "UPDATE data_akumulasi SET nilai_akumulasi =
nilai_akumulasi + $nilai_sensor WHERE nama_sensor = '$nama_sensor'";
        $conn->query($sql_update);
    } else {
        // Jika belum ada entri, tambahkan entri baru
        $sql_insert_akumulasi = "INSERT INTO data_akumulasi (nama_sensor,
nilai_akumulasi) VALUES ('$nama_sensor', $nilai_sensor)";
        $conn->query($sql_insert_akumulasi);
    }

    echo "Data berhasil disimpan ke database.";
} else {
    echo "Endpoint API tidak valid, gagal disimpan ke database.";
}

// Menutup koneksi database
$conn->close();
?>
```

LAMPIRAN 4 CODINGAN YOLO

```
!nvidia-smi
```

```
import os
HOME = os.getcwd()
print(HOME)
```

```
# Pip install method (recommended)

!pip install ultralytics

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

```
# Git clone method (for development)

# %cd {HOME}
# !git clone github.com/ultralytics/ultralytics
# %cd {HOME}/ultralytics
# !pip install -e .

# from IPython import display
# display.clear_output()

# import ultralytics
# ultralytics.checks()
```

```
from ultralytics import YOLO

from IPython.display import display, Image
```

```
%cd {HOME}
Image(filename='/content/runs/classify/train3/val_batch0_pred.jpg',
height=600)
```

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets

!pip install roboflow --quiet

!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="5fbsx9JHZdIBoA9wAGBP")
project = rf.workspace("mathtechstudio").project("aglaonema-l65wh")
dataset = project.version(2).download("folder")
```

```
%cd {HOME}

!yolo task=classify mode=train model=yolov8n-cls.pt data={dataset.location}
epochs=50 imgsz=128 val=True
```

```
from torch.utils.data import Dataset
from torchvision import transforms
from PIL import Image
import os
```

```
class YourDatasetClass(Dataset):
    def _init_(self, root, transform=None):
        self.root = root
        self.transform = transform
        self.image_paths = [os.path.join(root, img) for img in
        os.listdir(root)]

    def _len_(self):
        return len(self.image_paths)

    def _getitem_(self, idx):
        img_path = self.image_paths[idx]
        image = Image.open(img_path).convert('RGB')

        if self.transform:
            image = self.transform(image)

        # Misalnya, Anda dapat mengembalikan label sebagai indeks gambar
        label = idx

        return image, label
```

```
from torch.utils.data import DataLoader
from torchvision import transforms

# Load data validasi
transform = transforms.Compose([
    # Tambahkan transformasi yang sesuai jika diperlukan
])

dataset = YourDatasetClass(root='/content/datasets/Aglaonema-2/valid/',
transform=transform)
dataloader = DataLoader(dataset, batch_size=1, shuffle=False, num_workers=2)
```

```
from torch.utils.data import Dataset
from torchvision import transforms
from PIL import Image
import os

class CustomDataset(Dataset):
    def_init_(self, root, transform=None):
        self.root = root
        self.transform = transform
        self.image_paths = [os.path.join(root, img) for img in
        os.listdir(root)]

    def___len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        image = Image.open(img_path).convert('RGB')

        # Misalnya, kita anggap anotasi objek tersimpan dalam file
        txt dengan format:
        # label_index x_center y_center width height
```

```
        # (Contoh: 0 0.5 0.5 0.2 0.3)
        # Load anotasi dari file txt
        annotation_path = img_path.replace('.jpg', '.txt')
        with open(annotation_path, 'r') as file:
            annotation = file.readline().strip().split(' ')
            label = int(annotation[0])
            bbox = list(map(float, annotation[1:]))

        sample = {'image': image, 'label': label, 'bbox': bbox}

        if self.transform:
             sample['image'] = self.transform(sample['image'])

        return sample
```

```
from ultralytics import YOLO
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix

# Tentukan path dataset validasi
validation_data_path = '/path/to/your/validation/'

# Inisialisasi model YOLO
model = YOLO(model='yolov5s', pretrained=True)

# Evaluasi model pada dataset validasi
results = model.evaluate(dataset=validation_data_path)
# Peroleh hasil prediksi dan label sebenarnya
y_true = results.xyxy[0][:, 5].cpu().numpy()
y_pred = results.xyxy[0][:, 6].cpu().numpy()

# Hitung metrik
accuracy = accuracy_score(y_true, y_pred)
conf_matrix = confusion_matrix(y_true, y_pred)
classification_report_str = classification_report(y_true, y_pred)

# Tampilkan hasil
print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Classification Report:\n{classification_report_str}")
```

```
from datasets import CustomDataset
dataset = CustomDataset(root='/content/datasets/Aglaonema-2/valid',
transform=None)
```

```
from torch.utils.data import DataLoader
from torchvision import transforms
from custom_dataset import ClassificationDataset

# Load data validasi
transform = transforms.Compose([
    # Tambahkan transformasi yang sesuai jika diperlukan
])

dataset = ClassificationDataset(root='/content/datasets/Aglaonema-2/valid/',
transform=transform)
dataloader = DataLoader(dataset, batch_size=1, shuffle=False, num_workers=2)
```

```
transform = transforms.Compose([
    # Tambahkan transformasi yang sesuai jika diperlukan
])

dataset = YourDatasetClass(root='/content/datasets/Aglaonema-2/valid/',
transform=transform)
dataloader = DataLoader(dataset, batch_size=1, shuffle=False, num_workers=2)
```

```
from models.experimental import attempt_load
from torch.utils.data import DataLoader
from torchvision import transforms

model = attempt_load('/content/runs/classify/train/weights/best.pt',
map_location='cuda')
model = model.to('cuda').eval()

transform = transforms.Compose([
    # Tambahkan transformasi yang sesuai jika diperlukan
])

dataset = YourDatasetClass(root='/content/datasets/Aglaonema-2/valid/',
transform=transform)
dataloader = DataLoader(dataset, batch_size=1, shuffle=False, num_workers=2)
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

# Model evaluation
model.eval()
y_true = []
y_pred = []

with torch.no_grad():
    for batch in dataloader:
        inputs, labels = batch['image'], batch['label']
        outputs = model(inputs)

        _, predicted = torch.max(outputs, 1)

        y_true.extend(labels.cpu().numpy())
        y_pred.extend(predicted.cpu().numpy())

# Calculate metrics
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='macro') # atau 'micro',
'weighted', 'binary' tergantung pada kasus Anda
recall = recall_score(y_true, y_pred, average='macro')
f1 = f1_score(y_true, y_pred, average='macro')

print(f"Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, F1:
{f1}")
```

```
import torch
from sklearn.metrics import classification_report

all_preds = []
all_labels = []
```

```
with torch.no_grad():
    for images, labels in dataloader:
        images = images.to('cuda')
        preds = model(images)
        all_preds.append(preds.argmax(dim=1).cpu().numpy())
        all_labels.append(labels.cpu().numpy())
```

```
from sklearn.metrics import confusion_matrix

# Contoh confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred)

# Ekstrak nilai True Positives, True Negatives, False Positives, dan False
Negatives
TP = conf_matrix[1, 1]
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]
```

```
from torch.utils.data import DataLoader
from torchvision import transforms
from custom_dataset import CustomDataset

# Load data validasi
transform = transforms.Compose([
    # Tambahkan transformasi yang sesuai jika diperlukan
])

dataset = CustomDataset(root='/content/datasets/Aglaonema-2/valid/',
transform=transform)
dataloader = DataLoader(dataset, batch_size=1, shuffle=False, num_workers=2)
```

```
from models.experimental import attempt_load
from torch.utils.data import DataLoader
from datasets import Aglaonem  # Ganti dengan modul dan kelas dataset yang
sesuai

model = attempt_load('/content/path/to/your/model.pt', map_location='cuda')
model = model.to('cuda').eval()

transform = transforms.Compose([
    # Tambahkan transformasi yang sesuai jika diperlukan
])

dataset = YourDatasetClass(root='/content/path/to/your/validation/images/',
transform=transform)
dataloader = DataLoader(dataset, batch_size=1, shuffle=False, num_workers=2)
```

```
!ls -la {HOME}/runs/classify/train/
```

```
!cat {HOME}/runs/classify/train/results.csv | head -50
```

```
%cd {HOME}

!yolo task=classify mode=val model={HOME}/runs/classify/train/weights/best.pt
data={dataset.location}
```

```
%cd {HOME}
!yolo task=classify mode=predict
model={HOME}/runs/classify/train/weights/best.pt conf=0.25
source={dataset.location}/test/overripe
```

```
import glob
from IPython.display import Image, display

for image_path in
glob.glob(f'{HOME}/runs/classify/train/val_batch2_pred.jpg')[:3]:
      display(Image(filename=image_path, width=600))
      print("\n")
```

```
from yolov8.models.experimental import attempt_load
from yolov8.utils.torch_utils import select_device

# Menggunakan device CUDA jika tersedia, jika tidak, menggunakan CPU
device = select_device('0')  # Ganti '0' sesuai dengan GPU yang ingin Anda
gunakan atau 'cpu' untuk CPU

# Memuat model YOLOv8 yang telah dilatih
model = attempt_load('/content/runs/classify/train/weights/best.pt',
map_location=device)

# Selanjutnya, Anda dapat menggunakan model tersebut untuk membuat prediksi
dan menghitung metrik klasifikasi.
# Pastikan untuk menyesuaikan dengan cara model YOLOv8 Anda menerima input
dan mengeluarkan prediksi.
```

```
!git clone https://github.com/ultralytics/yolov5.git
```

```
%cd yolov5
```

```
!pip install -U -r requirements.txt
```

```
# Install dependencies
!pip install pycocotools
!pip install git+https://github.com/ultralytics/yolov5.git

# Import necessary modules
import json
from pycocotools.coco import COCO
from pycocotools.cocoeval import COCOeval
import torch

# Load the YOLOv8 model
model = torch.hub.load('ultralytics/yolov5:v6.0', 'yolov5s')
model.load_state_dict(torch.load('/content/yolov8n.pt'))  # Replace with the
path to your model
model = model.to('cuda').eval()

# Load COCO validation dataset
coco = COCO('/content/path/to/your/coco_annotation.json')  # Replace with the
path to your COCO annotations

# Run evaluation on the validation dataset
predictions = []
image_ids = coco.getImgIds()

for image_id in image_ids:
    img_info = coco.loadImgs(image_id)[0]
    image_path = '/content/path/to/your/validation/images/' +
img_info['file_name']  # Replace with the path to your validation images
```

```
    img =
torch.from_numpy(model.img2tensor(image_path)[0]).to('cuda').unsqueeze(0)
    prediction = model(img)[0]

    # Convert predictions to COCO format
    prediction_coco = model.results2json(prediction, img_info)
    predictions.extend(prediction_coco)

# Save predictions to a JSON file
with open('/content/path/to/your/coco_results.json', 'w') as f:
    json.dump(predictions, f)

# Load predictions using COCO API
coco_results = coco.loadRes('/content/path/to/your/coco_results.json')  #
Replace with the path to your results JSON file

# Initialize COCO evaluator
coco_eval = COCOeval(coco, coco_results, 'bbox')
coco_eval.evaluate()
coco_eval.accumulate()
coco_eval.summarize()

# Get evaluation metrics
precision = coco_eval.stats[0]
recall = coco_eval.stats[1]
f1_score = coco_eval.stats[2]

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1_score:.4f}')
```

```
import torch

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)
```

```
from yolov8.models.experimental import attempt_load
from yolov8.utils.general import non_max_suppression
import torch

# Load model
model = attempt_load('/content/yolov8n.pt', map_location='cpu')

# Set model ke mode evaluasi
model.eval()


# Inisialisasi metrik
true_positives = 0
false_positives = 0
false_negatives = 0

# Loop melalui data_loader
for images, labels in data_loader:
    # Forward pass untuk mendapatkan prediksi
    with torch.no_grad():
```

```
        predictions = model(images)

    # Post-process prediksi menggunakan non-maximum suppression (NMS)
    predictions = non_max_suppression(predictions, conf_thres=0.5,
iou_thres=0.5)

    # Perhitungan metrik
    for pred, label in zip(predictions, labels):
        # Misalnya, label[0] adalah tensor kelas ground truth
        # pred[:, -1] adalah tensor kelas prediksi
        true_positives += torch.sum((pred[:, -1] ==
label[0]).to(torch.int)).item()
        false_positives += torch.sum((pred[:, -1] !=
label[0]).to(torch.int)).item()
        false_negatives += torch.sum((pred[:, -1] !=
label[0]).to(torch.int)).item()

# Menghitung precision, recall, dan F1 score
precision = true_positives / (true_positives + false_positives)
recall = true_positives / (true_positives + false_negatives)
f1_score = 2 * (precision * recall) / (precision + recall)

print(f'True Positives: {true_positives}')
print(f'False Positives: {false_positives}')
print(f'False Negatives: {false_negatives}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1_score:.2f}')
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
import torch

# Contoh hasil prediksi dan label sebenarnya dari model YOLOv8
y_pred = model.predict(data_loader)  # Perlu disesuaikan dengan cara model
YOLOv8 memprediksi
y_true = get_true_labels()  # Perlu disesuaikan dengan cara Anda mendapatkan
label sebenarnya

# Menghitung metrik-metrik klasifikasi
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
```

```
pip install matplotlib pandas
```

```
PERHITUNGAN MANUAL UNTUK YOLO

import matplotlib.pyplot as plt
import pandas as pd
```

```
A11, A12, A13, A14, A15 = 30, 0, 0, 0, 0
A21, A22, A23, A24, A25 = 0, 30, 0, 0, 0
A31, A32, A33, A34, A35 = 0, 0, 30, 0, 0
A41, A42, A43, A44, A45 = 0, 0, 0, 30, 0
A51, A52, A53, A54, A55 = 0, 0, 0, 0, 11

# Menghitung metrik keseluruhan
total_TP = A11 + A22 + A33 + A44 + A55
total_FP = sum([A21, A31, A41, A51, A12, A32, A42, A52, A13, A23, A43, A53,
A14, A24, A34, A44, A54, A15, A25, A35, A45, A55]) - total_TP
total_FN = sum([A12, A13, A14, A15, A21, A23, A24, A25, A31, A32, A34, A35,
A41, A42, A43, A45, A51, A52, A53, A54]) - total_TP
total_TN = sum([A11, A12, A13, A14, A15, A21, A22, A23, A24, A25, A31, A32,
A33, A34, A35, A41, A42, A43, A44, A45, A51, A52, A53, A54, A55]) - (total_TP
+ total_FP + total_FN)

# Akurasi keseluruhan
accuracy = (total_TP + total_TN) / (total_TP + total_TN + total_FP +
total_FN)

# Recall keseluruhan
recall = total_TP / (total_TP + total_FN) if (total_TP + total_FN) != 0 else
0

# Precision keseluruhan
precision = total_TP / (total_TP + total_FP) if (total_TP + total_FP) != 0
else 0

# F1-score keseluruhan
f1_score = (2 * precision * recall) / (precision + recall) if (precision +
recall) != 0 else 0

# Membuat tabel
metrics_dict = {
    'Metrik': ['Akurasi', 'Recall', 'Precision', 'F1-score'],
    'Nilai': [accuracy, recall, precision, f1_score]
}

metrics_df = pd.DataFrame(metrics_dict)
display(metrics_df)

# Membuat grafik
fig, ax = plt.subplots(figsize=(8, 6))
metrics_df.plot(kind='bar', x='Metrik', y='Nilai', ax=ax, legend=False)
plt.title('Metrik Keseluruhan')
plt.ylabel('Nilai')
plt.show()
```

```
PERHITUNGAN MANUAL UNTUK SSD

import matplotlib.pyplot as plt
import pandas as pd

A11, A12, A13, A14, A15 = 29, 0, 0, 0, 0
A21, A22, A23, A24, A25 = 0, 30, 0, 0, 0
```

```
A31, A32, A33, A34, A35 = 0, 0, 30, 0, 0
A41, A42, A43, A44, A45 = 0, 0, 0, 30, 0
A51, A52, A53, A54, A55 = 0, 0, 0, 0, 10

# Menghitung metrik keseluruhan
total_TP = A11 + A22 + A33 + A44 + A55
total_FP = sum([A21, A31, A41, A51, A12, A32, A42, A52, A13, A23, A43, A53,
A14, A24, A34, A44, A54, A15, A25, A35, A45, A55]) - total_TP
total_FN = sum([A12, A13, A14, A15, A21, A23, A24, A25, A31, A32, A34, A35,
A41, A42, A43, A45, A51, A52, A53, A54]) - total_TP
total_TN = sum([A11, A12, A13, A14, A15, A21, A22, A23, A24, A25, A31, A32,
A33, A34, A35, A41, A42, A43, A44, A45, A51, A52, A53, A54, A55]) - (total_TP
+ total_FP + total_FN)

# Akurasi keseluruhan
accuracy = (total_TP + total_TN) / (total_TP + total_TN + total_FP +
total_FN)

# Recall keseluruhan
recall = total_TP / (total_TP + total_FN) if (total_TP + total_FN) != 0 else
0

# Precision keseluruhan
precision = total_TP / (total_TP + total_FP) if (total_TP + total_FP) != 0
else 0

# F1-score keseluruhan
f1_score = (2 * precision * recall) / (precision + recall) if (precision +
recall) != 0 else 0

# Membuat tabel
metrics_dict = {
'Metrik': ['Akurasi', 'Recall', 'Precision', 'F1-score'],
'Nilai': [accuracy, recall, precision, f1_score]
}

metrics_df = pd.DataFrame(metrics_dict)
display(metrics_df)

# Membuat grafik
fig, ax = plt.subplots(figsize=(8, 6))
metrics_df.plot(kind='bar', x='Metrik', y='Nilai', ax=ax, legend=False)
plt.title('Metrik Keseluruhan')
plt.ylabel('Nilai')
plt.show()
```