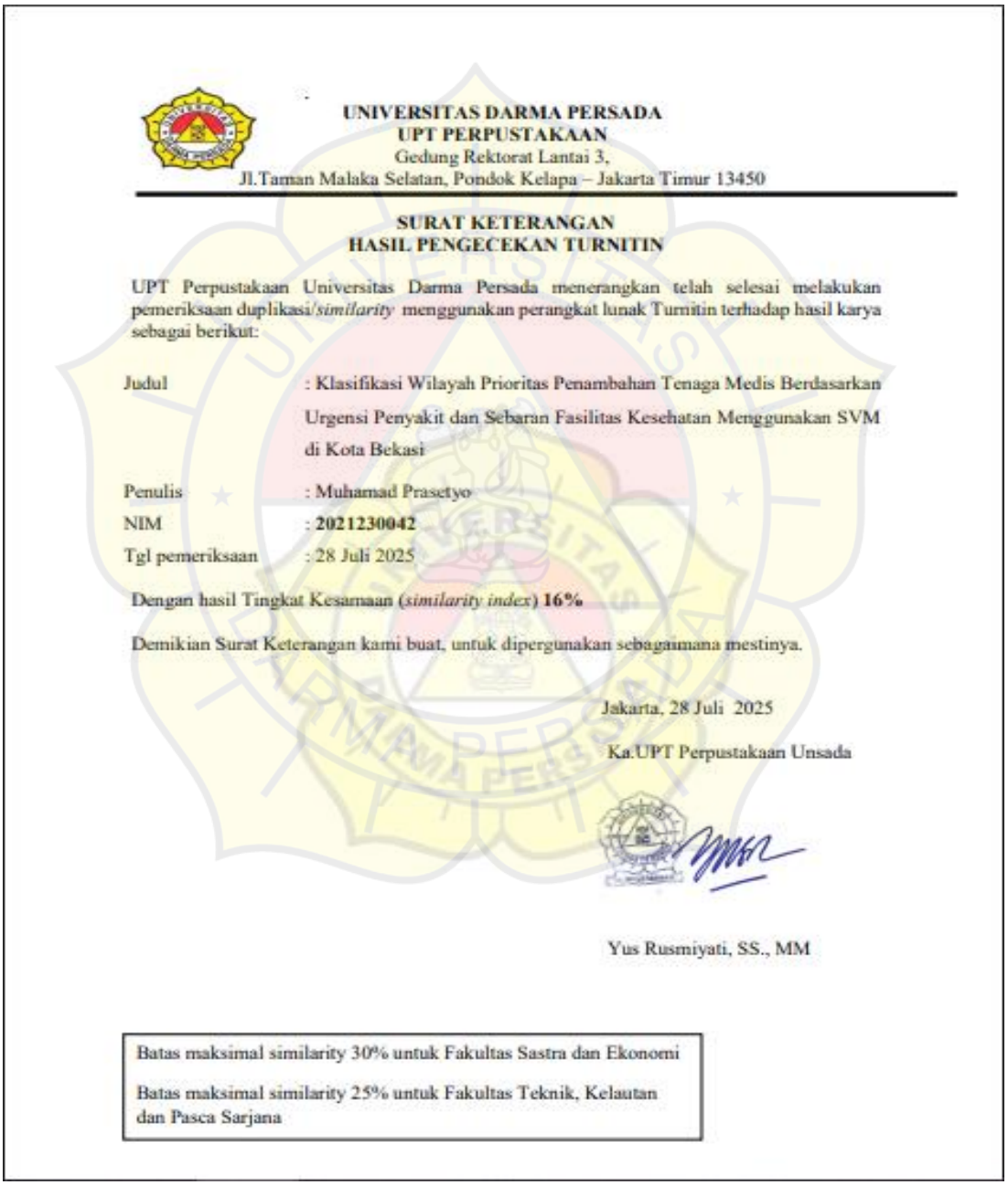
LAMPIRAN

Lampiran 1

Surat Keterangan Bebas Plagiat



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 : Klasifikasi Wilayah Prioritas Penambahan Tenaga Medis Berdasarkan Urgensi Penyakit dan Sebaran Fasilitas Kesehatan Menggunakan SVM di Kota Bekasi

Penulis : Muhamad Prasetyo
NIM : **2021230042**
Tgl pemeriksaan : 28 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **16%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 28 Juli 2025

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 Turnitin

2021230042_Muhamad Prasetyo

ORIGINALITY REPORT

16%	15%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	journal.arteii.or.id Internet Source	2%
2	kc.umn.ac.id Internet Source	1%
3	lmsspada.kemdiktisaintek.go.id Internet Source	1%
4	e-journal.hamzanwadi.ac.id Internet Source	<1%
5	ejournal.bsi.ac.id Internet Source	<1%
6	jurnal.radenfatah.ac.id Internet Source	<1%
7	journal.mediapublikasi.id Internet Source	<1%
8	docobook.com Internet Source	<1%
9	bpmpp.uma.ac.id Internet Source	<1%
10	journal.isas.or.id Internet Source	<1%
11	openjournal.unpam.ac.id Internet Source	<1%
12	ejournal.nusamandiri.ac.id Internet Source	<1%
13	mikrotekno.com Internet Source	<1%

Lampiran 3

Source Code api_svm.py

```
import pandas as pd
from flask import Flask, request, jsonify
import joblib
import numpy as np
import os

app = Flask(__name__)

# --- Lokasi File Model ---
MODEL_DIR = os.path.dirname(os.path.abspath(__file__))

model_filename = os.path.join(MODEL_DIR,
'svm_prioritas_wilayah_model.pkl')
label_encoder_filename = os.path.join(MODEL_DIR,
'label_encoder.pkl')
scaler_filename = os.path.join(MODEL_DIR, 'scaler.pkl')
feature_names_filename = os.path.join(MODEL_DIR,
'feature_names.pkl')

# --- Muat Model, Scaler, dan Label Encoder ---
try:
    svm_model = joblib.load(model_filename)
    le = joblib.load(label_encoder_filename)
    scaler = joblib.load(scaler_filename)
    feature_names = joblib.load(feature_names_filename)
    print("Model SVM dan komponen lainnya berhasil dimuat.")
except FileNotFoundError as e:
    print(f"Error: File model tidak ditemukan. Detail: {e}")
    exit()
except Exception as e:
    print(f"Error saat memuat model: {e}")
    exit()

# --- Definisi Kolom ---
medical_personnel_cols = [
    'Dr Spesialis', 'Dokter', 'Dokter Gigi', 'Dokter Gigi
Spesialis',
```

```
    'Tenaga Keperawatan', 'Tenaga Kebidanan', 'Tenaga Kesehatan
Masyarakat',
    'Tenaga Kesehatan Lingkungan', 'Tenaga Gizi', 'Ahli Teknologi
Laboratorium Medik',
    'Tenaga Teknik Biomedika Lainnya', 'Keterapian Fisik',
'Keteknisian Medis',
    'Tenaga Teknis Kefarmasian', 'Apoteker', 'Pejabat Struktural',
'Tenaga Dukungan Manajemen'
]

disease_cols = [
    'Hipertensi', 'Diabetes Mellitus', 'TBC', 'ISPA', 'Stroke',
'Pneumonia',
    'Malaria', 'Demam Berdarah', 'Hepatitis', 'Kanker', 'Gagal
Ginjal',
    'Asma', 'Osteoarthritis'
]

# --- Ambang batas (ganti dengan nilai dari training) ---
q1_burden_global = 1.5
q2_burden_global = 4.5

@app.route('/classify_faskes', methods=['POST'])
def classify_faskes():
    try:
        data = request.json
        if not data:
            return jsonify({'error': 'Tidak ada data JSON yang
diterima'}), 400

        input_df_raw = pd.DataFrame([data])
        processed_input =
input_df_raw.reindex(columns=feature_names)
        processed_input = processed_input.apply(pd.to_numeric,
errors='coerce').fillna(0)

        if processed_input.empty or processed_input.shape[0] == 0:
            return jsonify({'error': 'Gagal memproses data
input.'}), 400

        scaled_input = scaler.transform(processed_input)
```

```
        predicted_label_encoded = svm_model.predict(scaled_input)[0]
        predicted_label_original =
le.inverse_transform([predicted_label_encoded])[0]

        recommendations = []

        total_cases_agg =
processed_input[disease_cols].sum(axis=1).values[0]
        total_personnel_agg =
processed_input[medical_personnel_cols].sum(axis=1).values[0]
        current_burden_ratio = total_cases_agg /
(total_personnel_agg + 1)

        disease_high_thresholds = {
            'Hipertensi': 200, 'Diabetes Mellitus': 50, 'TBC': 10,
'ISPA': 200,
            'Stroke': 30, 'Pneumonia': 10, 'Malaria': 5, 'Demam
Berdarah': 5,
            'Hepatitis': 5, 'Kanker': 10, 'Gagal Ginjal': 10,
'Asma': 50, 'Osteoarthritis': 50
        }

        personnel_low_thresholds = {
            'Dr Spesialis': 1, 'Dokter': 2, 'Dokter Gigi': 1,
'Dokter Gigi Spesialis': 0,
            'Tenaga Keperawatan': 5, 'Tenaga Kebidanan': 2, 'Tenaga
Kesehatan Masyarakat': 1,
            'Tenaga Kesehatan Lingkungan': 1, 'Tenaga Gizi': 1,
'Ahli Teknologi Laboratorium Medik': 1,
            'Tenaga Teknik Biomedika Lainnya': 0, 'Keterapian
Fisik': 0, 'Keteknisian Medis': 0,
            'Tenaga Teknis Kefarmasian': 1, 'Apoteker': 1, 'Pejabat
Struktural': 0, 'Tenaga Dukungan Manajemen': 1
        }

        personnel_target_thresholds = {
            'Dr Spesialis': 2, 'Dokter': 10, 'Dokter Gigi': 2,
'Dokter Gigi Spesialis': 1,
            'Tenaga Keperawatan': 20, 'Tenaga Kebidanan': 5, 'Tenaga
Kesehatan Masyarakat': 3,
```

```
            'Tenaga Kesehatan Lingkungan': 2, 'Tenaga Gizi': 2,
'Ahli Teknologi Laboratorium Medik': 2,
            'Tenaga Teknik Biomedika Lainnya': 1, 'Keterapian
Fisik': 1, 'Keteknisian Medis': 1,
            'Tenaga Teknis Kefarmasian': 3, 'Apoteker': 2, 'Pejabat
Struktural': 1, 'Tenaga Dukungan Manajemen': 5
        }

        recommendations.append({
            "type": "summary",
            "message": f"Unit kerja ini memiliki prioritas
{predicted_label_original}. Rasio beban penyakit terhadap tenaga
medis {predicted_label_original.lower()}
({current_burden_ratio:.2f})."
        })

        if predicted_label_original == 'Tinggi':
            recommendations.append({
                "type": "advice",
                "message": "Sangat disarankan untuk peningkatan
sumber daya atau intervensi segera."
            })

            for col in disease_cols:
                val = processed_input[col].values[0]
                if val > disease_high_thresholds.get(col,
float('inf')):
                    recommendations.append({
                        "type": "specific_recommendation",
                        "category": "Penyakit",
                        "item": col,
                        "action": "Prioritaskan penanganan",
                        "value": int(val),
                        "unit": "kasus",
                        "detail": f"Kasus {col} ({int(val)}) sangat
tinggi."
                    })

            for col in medical_personnel_cols:
                val = processed_input[col].values[0]
                target = personnel_target_thresholds.get(col, 0)
```

```
                if val < personnel_low_thresholds.get(col, 0):
                    if target > val:
                        recommendations.append({
                            "type": "specific_recommendation",
                            "category": "Tenaga Medis",
                            "item": col,
                            "action": "Tambahkan",
                            "value": int(target - val),
                            "unit": "orang",
                            "detail": f"Tenaga {col} ({int(val)})
sangat rendah."
                        })
                    else:
                        recommendations.append({
                            "type": "specific_recommendation",
                            "category": "Tenaga Medis",
                            "item": col,
                            "action": "Perlu dipertimbangkan
penambahan",
                            "value": int(val),
                            "unit": "orang",
                            "detail": f"Tenaga {col} ({int(val)})
sangat rendah."
                        })

        elif predicted_label_original == 'Rendah':
            recommendations.append({
                "type": "advice",
                "message": "Pertahankan standar pelayanan yang ada
dan fokus pada program preventif."
            })

        else:  # Sedang
            recommendations.append({
                "type": "advice",
                "message": "Lakukan pemantauan rutin dan optimalkan
alokasi sumber daya yang efisien."
            })

            for col in disease_cols:
                val = processed_input[col].values[0]
```

```
                if val > disease_high_thresholds.get(col,
float('inf')) * 0.75:
                    recommendations.append({
                        "type": "specific_recommendation",
                        "category": "Penyakit",
                        "item": col,
                        "action": "Kasus dengan tingkat kewaspadaan
tinggi",
                        "value": int(val),
                        "unit": "kasus",
                        "detail": f"Kasus {col} ({int(val)})
termasuk tingkat kewaspadaan tinggi. Ada potensi peningkatan kasus."
                    })

            for col in medical_personnel_cols:
                val = processed_input[col].values[0]
                target = personnel_target_thresholds.get(col, 0)
                if 0 < val < personnel_low_thresholds.get(col, 0) *
1.5:
                    if target > val:
                        recommendations.append({
                            "type": "specific_recommendation",
                            "category": "Tenaga Medis",
                            "item": col,
                            "action": "Direkomendasikan penambahan
tenaga medis",
                            "value": int(target - val),
                            "unit": "orang",
                            "detail": f"Tenaga {col} ({int(val)})
direkomendasikan untuk penambahan."
                        })
                    else:
                        recommendations.append({
                            "type": "specific_recommendation",
                            "category": "Tenaga Medis",
                            "item": col,
                            "action": "Perlu diawasi",
                            "value": int(val),
                            "unit": "orang",
                            "detail": f"Tenaga {col} ({int(val)})
perlu diawasi."
```

```
                        })

        return jsonify({
            'prioritas_wilayah': predicted_label_original,
            'recommendations': recommendations
        }), 200

    except Exception as e:
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```