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	: PREDIKSI JUMLAH KASUS PENYAKIT DBD, TBC, DAN DIARE BERDASARKAN DATA PER KECAMATAN DAN TAHUN DI KABUPATEN BEKASI MENGGUNAKAN ALGORITMA DECISION TREE REGRESSION
Penulis	: Muhammad Dzaki Zahirsyah
NIM	: **2021230043**
Tgl pemeriksaan	: 25 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **15%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 25 Juli 2025

Ka.UPT Perpustakaan Unsada

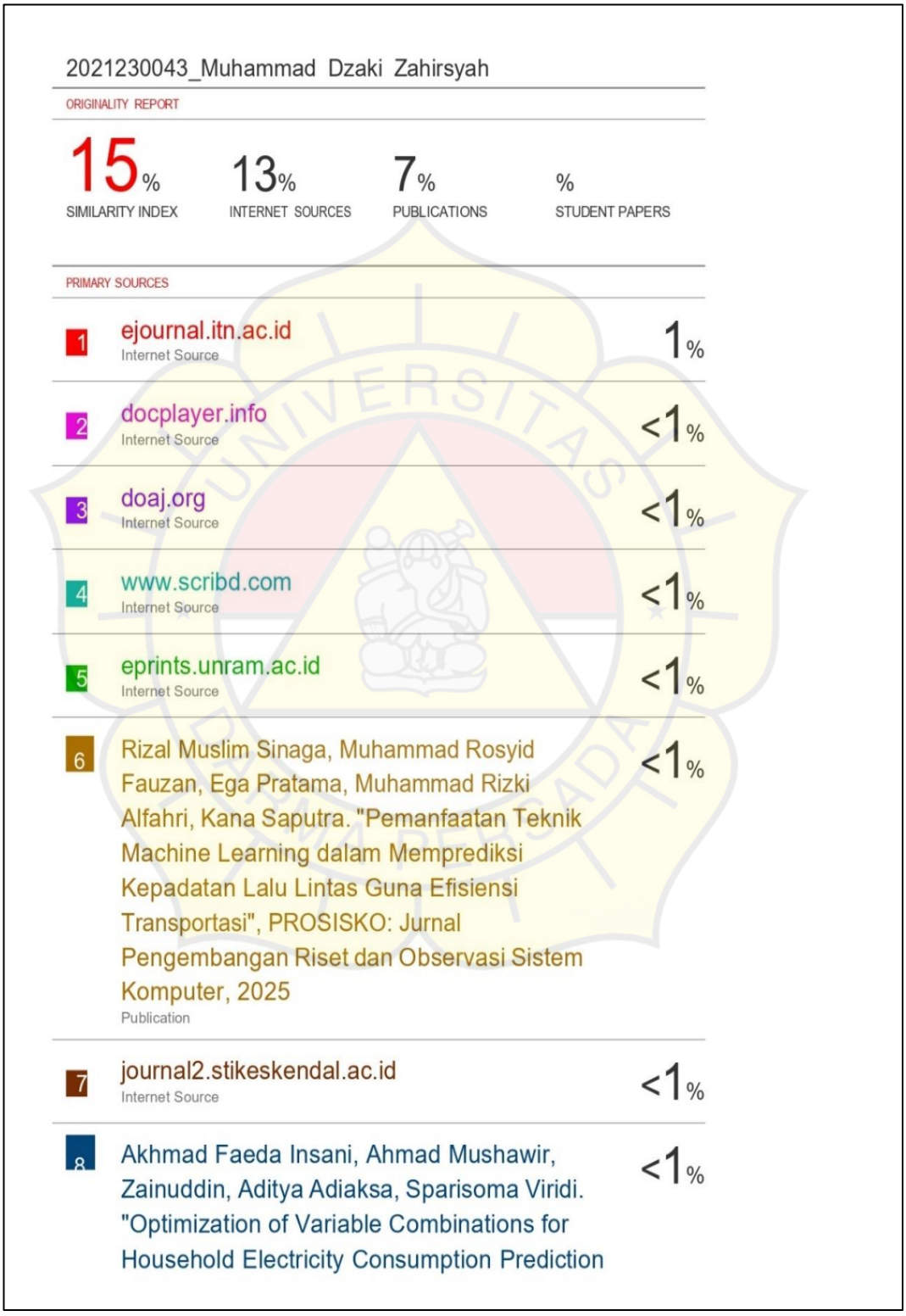
Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

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

Lampiran 2

Hasil Turnitin



2021230043_Muhammad Dzaki Zahirsyah

ORIGINALITY REPORT

15%	13%	7%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ejournal.itn.ac.id Internet Source	1%
2	docplayer.info Internet Source	<1%
3	doaj.org Internet Source	<1%
4	www.scribd.com Internet Source	<1%
5	eprints.unram.ac.id Internet Source	<1%
6	Rizal Muslim Sinaga, Muhammad Rosyid Fauzan, Ega Pratama, Muhammad Rizki Alfahri, Kana Saputra. "Pemanfaatan Teknik Machine Learning dalam Memprediksi Kepadatan Lalu Lintas Guna Efisiensi Transportasi", PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer, 2025 Publication	<1%
7	journal2.stikeskendal.ac.id Internet Source	<1%
8	Akhmad Faeda Insani, Ahmad Mushawir, Zainuddin, Aditya Adiaksa, Sparisoma Viridi. "Optimization of Variable Combinations for Household Electricity Consumption Prediction	<1%

Lampiran 3

Source Code App.py

```
import os
import json
import joblib
import pandas as pd
import numpy as np
import time
import re # Import library untuk regular expression
import matplotlib
matplotlib.use('Agg')  # Gunakan backend non-GUI agar bisa disimpan di server
import matplotlib.pyplot as plt
import seaborn as sns
from flask import Flask, request, jsonify
from flask_cors import CORS
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import BaggingRegressor
from sklearn.metrics import (mean_absolute_error, mean_squared_error, r2_score,
                             accuracy_score, precision_score, recall_score, f1_score,
                             mean_absolute_percentage_error)
from scipy.stats import ks_2samp
# --- IMPORT LIBRARY BARU ---
import dtreeviz
app = Flask(__name__)
CORS(app)
# --- Konfigurasi dan Variabel Global ---
PATH_TO_MODELS = os.path.join(os.path.dirname(__file__), '..', 'models')
COLUMN_TAHUN = 'tahun'
COLUMN_BULAN = 'bulan'
COLUMN_PENDUDUK_RIBU = 'penduduk(ribu)'
# Variabel untuk menampung model dan data pendukung yang dimuat
model_dbd, model_tbc, model_diare = None, None, None
original_features, average_feature_values, disease_recommendations = None, None, {}
ALL_MODEL_FEATURES_NAMES = []
# --- Fungsi-fungsi Utama ---
def load_all_dependencies():
    """Fungsi untuk memuat (atau memuat ulang) semua model dan file pendukung."""
    global model_dbd, model_tbc, model_diare, original_features, average_feature_values, disease_recommendations, ALL_MODEL_FEATURES_NAMES
    try:
        model_dbd = joblib.load(os.path.join(PATH_TO_MODELS, 'decision_tree_model_dbd.joblib'))
        model_tbc = joblib.load(os.path.join(PATH_TO_MODELS, 'decision_tree_model_tbc.joblib'))
```

```
41.          model_diare                                          =
  joblib.load(os.path.join(PATH_TO_MODELS,
  'decision_tree_model_diare.joblib'))
42.          original_features                                    =
  joblib.load(os.path.join(PATH_TO_MODELS,
  'model_features.joblib'))
43.          average_feature_values                               =
  joblib.load(os.path.join(PATH_TO_MODELS,
  'average_feature_values.joblib'))
44.          with              open(os.path.join(PATH_TO_MODELS,
  'recommendations.json'), 'r', encoding='utf-8') as f:
45.              disease_recommendations = json.load(f)
46.          ALL_MODEL_FEATURES_NAMES   =   [col   for   col   in
  original_features if col not in [COLUMN_TAHUN, COLUMN_BULAN] and
  not col.startswith('Kecamatan_')]
47.          print("SUCCESS: Model dan semua dependensi berhasil
  dimuat/dimuat ulang.")
48.          return True
49.      except Exception as e:
50.          print(f"CRITICAL ERROR: Gagal memuat model atau data
  pendukung. Pastikan semua file .joblib dan .json ada di folder
  'models'. Error: {e}")
51.          return False
52.
53.  def        classify_spread_level(cases,        disease_type,
  population_thousands):
54.      """Mengklasifikasikan tingkat penyebaran berdasarkan
  jumlah kasus dan populasi."""
55.      level = "Tidak Diketahui"
56.      recommendation = "Konsultasikan otoritas kesehatan untuk
  pedoman spesifik."
57.
58.      if cases is None or population_thousands is None or
  population_thousands <= 0:
59.          return  "Data  Tidak  Tersedia",  "Data  kasus  atau
  populasi tidak valid.", 0.0
60.       rate_per_thousand_pop = (cases / population_thousands)
  if population_thousands else 0
61.      percentage_rate = round((cases / (population_thousands *
  1000)) * 100, 2)
62.      # Ambang batas ini bisa Anda sesuaikan jika perlu
63.      threshold_rendah_rate = 0.5
64.      threshold_sedang_rate = 1.5
65.      if rate_per_thousand_pop < threshold_rendah_rate: level
  = "Rendah"
66.      elif   rate_per_thousand_pop   <   threshold_sedang_rate:
  level = "Sedang"
67.      else: level = "Tinggi"
68.      if disease_recommendations:
69.          try:
70.              rec_template                                     =
  disease_recommendations.get(disease_type,        {}).get(level,
  recommendation)
71.              recommendation                                   =
  rec_template.replace("{rate_per_thousand_pop}",
```

```
  f"{rate_per_thousand_pop:.2f}").replace("{percentage_rate}",
  f"{percentage_rate:.2f}")
72.           except Exception as e:
73.               print(f"Error format rekomendasi: {e}")
74.               recommendation = "Rekomendasi default."
75.       return level, recommendation, percentage_rate
76.
77.   # BAGIAN UNTUK PREDIKSI
78.   def    make_all_predictions(year,    month,    kecamatan_name,
  user_factor_overrides):
79.       """Fungsi utama untuk membuat prediksi berdasarkan input
  pengguna."""
80.       if    not    all([model_dbd,    model_tbc,    model_diare,
  original_features, average_feature_values]):
81.           return None, None, "Model atau data pendukung belum
  dimuat di server. Lakukan pelatihan."
82.
83.       input_data = average_feature_values.copy()
84.       for factor, value in user_factor_overrides.items():
85.           if value is not None and value != '':
86.               try:
87.                   input_data[factor] = float(value)
88.               except ValueError:
89.                   print(f"Warning:   Nilai   '{value}'   untuk
  '{factor}' tidak valid, menggunakan nilai rata-rata.")
90.       input_data[COLUMN_TAHUN] = year
91.       input_data[COLUMN_BULAN] = month
92.       kecamatan_col_name = f'kecamatan_{kecamatan_name}'
93.       if kecamatan_col_name not in original_features:
94.           return  None,  None,  f"Kecamatan  '{kecamatan_name}'
  tidak dikenal oleh model."
95.       for col in original_features:
96.           if col.startswith('kecamatan_'):
97.               input_data[col] = 1 if col == kecamatan_col_name
  else 0
98.       population = float(input_data.get(COLUMN_PENDUDUK_RIBU,
  1))
99.       if population <= 0: population = 1
100.      try:
101.          input_df                                          =
  pd.DataFrame([input_data])[original_features]
102.      except Exception as e:
103.          return  None,  None,  f"Gagal  membuat  DataFrame.  Cek
  fitur: {e}"
104.      try:
105.          results = {}
106.          # DBD
107.          pred_dbd                    =                    max(0,
  round(model_dbd.predict(input_df)[0]))
108.          level_dbd,         rec_dbd,         perc_dbd         =
  classify_spread_level(pred_dbd, 'DBD', population)
109.          results['DBD']    =    {"predicted_cases":    pred_dbd,
  "spread_level":    level_dbd,    "recommendation":    rec_dbd,
  "percentage": perc_dbd}
110.          # TBC
```

```
111.         pred_tbc                    =                   max(0,
  round(model_tbc.predict(input_df)[0]))
112.         level_tbc,         rec_tbc,         perc_tbc         =
  classify_spread_level(pred_tbc, 'TBC', population)
113.         results['TBC']    =    {"predicted_cases":    pred_tbc,
  "spread_level":    level_tbc,    "recommendation":    rec_tbc,
  "percentage": perc_tbc}
114.         # Diare
115.         pred_diare                  =                   max(0,
  round(model_diare.predict(input_df)[0]))
116.         level_diare,       rec_diare,       perc_diare       =
  classify_spread_level(pred_diare, 'Diare', population)
117.         results['Diare']  =  {"predicted_cases":  pred_diare,
  "spread_level":   level_diare,   "recommendation":   rec_diare,
  "percentage": perc_diare}
118.
119.         final_inputs_used    =    {k:    v    for    k,    v    in
  input_data.items()  if  k  in  ALL_MODEL_FEATURES_NAMES  or  k  in
  [COLUMN_TAHUN, COLUMN_BULAN]}
120.         return results, final_inputs_used, None
121.     except Exception as e:
122.         return None, None, f"Gagal melakukan prediksi: {e}"
123.
124. @app.route('/predict', methods=['POST'])
125. def predict():
126.     if not request.is_json:
127.         return    jsonify({"status":    "error",    "message":
  "Permintaan harus dalam format JSON."}), 400
128.      data = request.get_json()
129.     year = data.get('tahun')
130.     month = data.get('bulan')
131.     kecamatan = data.get('kecamatan')
132.     user_factor_overrides = data.get('faktor_penyebab', {})
133.
134.     if not (isinstance(year, int) and isinstance(month, int)
  and isinstance(kecamatan, str) and
135.             year >= 2020 and 1 <= month <= 12 and kecamatan):
136.         return jsonify({"status": "error", "message": "Input
  'tahun', 'bulan', dan 'kecamatan' tidak valid."}), 400
137. predictions_data,       inputs_used,       error_msg       =
  make_all_predictions(year,          month,          kecamatan,
  user_factor_overrides)
138.     if error_msg:
139.         return    jsonify({"status":    "error",    "message":
  error_msg}), 400
140.     else:
141.         return   jsonify({"status":   "success",   "results":
  predictions_data, "final_inputs_used": inputs_used}), 200
142.
143. # BAGIAN UNTUK PELATIHAN MODEL
144. def clean_column_names(df):
145.     """Membersihkan semua nama kolom di DataFrame."""
146.     new_columns = []
147.     for col in df.columns:
148.         new_col = str(col).lower()
149.         new_col = re.sub(r'[^a-z0-9_]+', '_', new_col)
```

```
150.         new_col = new_col.strip('_')
151.         new_columns.append(new_col)
152.     df.columns = new_columns
153.     return df
154.  def calculate_smape(y_true, y_pred):
155.     """Menghitung Symmetric Mean Absolute Percentage
  Error."""
156.     denominator = (np.abs(y_true) + np.abs(y_pred))
157.     denominator = np.where(denominator == 0, 1e-9,
  denominator)
158.     return np.mean(2 * np.abs(y_pred - y_true) /
  denominator) * 100
159.
160. def train_and_evaluate_from_file(file_path):
161.     global model_dbd, model_tbc, model_diare,
  original_features, average_feature_values
162.
163.     try:
164.         if file_path.endswith('.xlsx'):
165.             df = pd.read_excel(file_path, engine='openpyxl')
166.         else:
167.             df = pd.read_csv(file_path)
168.     except Exception as e:
169.         return {"error": f"Gagal membaca file: {str(e)}"}
170.     df = clean_column_names(df)
171.     TARGET_DBD = 'jumlah_kasus_dbd'
172.     TARGET_TBC = 'jumlah_kasus_tbc'
173.     TARGET_DIARE = 'jumlah_kasus_diare'
174.
175.     df.drop(columns=['id', 'uploaded_by', 'uploaded_at'],
  inplace=True, errors='ignore')
176.     df.fillna(0, inplace=True)
177.
178.     if 'kecamatan' in df.columns:
179.         df = pd.get_dummies(df, columns=['kecamatan'],
  prefix='kecamatan')
180.     all_targets = [TARGET_DBD, TARGET_TBC, TARGET_DIARE]
181.     current_features = [col for col in df.columns if col not
  in all_targets]
182.     X = df[current_features]
183.     joblib.dump(current_features,
  os.path.join(PATH_TO_MODELS, 'model_features.joblib'))
184.     numeric_features = [f for f in current_features if not
  f.startswith('kecamatan_')]
185.     new_average_values =
  X[numeric_features].mean().to_dict()
186.     joblib.dump(new_average_values,
  os.path.join(PATH_TO_MODELS, 'average_feature_values.joblib'))
187.
188.     all_results = {}
189.     diseases = {"DBD": TARGET_DBD, "TBC": TARGET_TBC,
  "Diare": TARGET_DIARE}
190.     rng = np.random.default_rng(seed=42)
191.
192.     for name, target_col in diseases.items():
193.         if target_col not in df.columns:
```

```
194.             continue
195.         # ... di dalam loop for ...
196.         print(f"INFO: Memulai pelatihan dan tuning untuk
  {name}...")
197.         y = df[target_col]
198.         X_train,     X_test,     y_train,     y_test     =
  train_test_split(X, y, test_size=0.2, random_state=42)
199.
200.         # 1. Tentukan parameter grid untuk di-tuning
201.         param_grid = {
202.             'n_estimators': [100, 150, 200],        # Jumlah
  pohon
203.             'max_samples':   [0.8,   1.0],                 #
  Persentase data untuk tiap pohon
204.             'estimator__max_depth':   [7,   10,   15],      #
  Kedalaman pohon
205.             'estimator__min_samples_leaf': [1, 2, 3],# Kunci
  untuk sensitivitas
206.             'estimator__min_samples_split':  [2,  5,  8]  #
  Kunci untuk sensitivitas
207.         }
208.        # 2. Siapkan GridSearchCV dengan BaggingRegressor
209.         grid_search = GridSearchCV(
210.             estimator=BaggingRegressor(
211.                 #   Tetap   gunakan   DecisionTreeRegressor
  sebagai model dasar
212.                 estimator=DecisionTreeRegressor(random_state
  =42),
213.                 random_state=42
214.             ),
215.             # Sesuaikan parameter grid untuk BaggingRegressor
216.             param_grid=param_grid,
217.             cv=5,
218.             scoring='neg_mean_squared_error',
219.             n_jobs=-1
220.         )
221.
222.         # 3. Lakukan fitting untuk mencari parameter terbaik
223.         grid_search.fit(X_train, y_train)
224.
225.         # 4. Ambil model terbaik hasil tuning
226.         model = grid_search.best_estimator_
227.
228.         #   (Opsional)   Tampilkan   parameter   terbaik   yang
  ditemukan
229.         print(f"INFO:   Parameter   terbaik   untuk   {name}:
  {grid_search.best_params_}")
230.
231.         # 5. Simpan model TERBAIK yang sudah di-tuning
232.         joblib.dump(model,      os.path.join(PATH_TO_MODELS,
  f'decision_tree_model_{name.lower()}.joblib'))
233.         print(f"INFO: Model terbaik untuk {name} berhasil
  disimpan.")
234.
235.         # Sisa kode di bawah ini TIDAK PERLU DIUBAH.
```

```
        # Variabel 'model' sudah berisi model terbaik, sehingga plot dan evaluasi akan berjalan seperti biasa.
        y_pred = model.predict(X_test)
# ... sisa kode untuk evaluasi dan plot ...
        charts_dir = os.path.join(os.path.dirname(__file__), 'static', 'charts')
        os.makedirs(charts_dir, exist_ok=True)
        timestamp = int(time.time())
# --- PEMBUATAN GAMBAR-GAMBAR VISUALISASI ---
        scatter_filename = f'scatter_{name.lower()}_{timestamp}.png'
        importance_filename = f'importance_{name.lower()}_{timestamp}.png'
        tree_plot_filename = f'tree_plot_{name.lower()}_{timestamp}.png' # Nama file untuk plot_tree

        # 1. Feature Importance
        # Feature Importance (perlu penyesuaian untuk Bagging)
        # Kita akan ambil rata-rata importance dari semua pohon di ensemble
        importances_df = pd.DataFrame([tree.feature_importances_ for tree in model.estimators_], columns=X_train.columns)
        mean_importances = importances_df.mean().sort_values(ascending=False)

        plt.figure(figsize=(10, 7))
        sns.barplot(x=mean_importances.head(10).values, y=mean_importances.head(10).index, palette='viridis', hue=mean_importances.head(10).index, legend=False)
        plt.title(f'Top 10 Fitur Penting - Model {name} (Bagging)')
        plt.xlabel('Tingkat Kepentingan Rata-rata'); plt.ylabel('Fitur')
        plt.savefig(os.path.join(charts_dir, importance_filename), bbox_inches='tight')
        plt.close()
# 2. Scatter Plot
        r2 = r2_score(y_test, y_pred)
        plt.figure(figsize=(8, 6))
        sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
        plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
        plt.title(f"Aktual vs. Prediksi - Model {name} (R²: {r2:.2f})")
        plt.xlabel("Kasus Aktual"); plt.ylabel("Kasus Diprediksi"); plt.grid(True)
        plt.savefig(os.path.join(charts_dir, scatter_filename), bbox_inches='tight')
        plt.close()

        # 3. Visualisasi Pohon Keputusan dengan plot_tree (PENGGANTI DTREEVIZ)
```

```
270.         # Perubahan 3: Penyesuaian Visualisasi Pohon Keputusan
271.         try:
272.             # Ambil satu pohon dari ensemble untuk divisualisasikan
273.             single_tree_to_plot = model.estimators_[0]
274.             plt.figure(figsize=(30, 15))
275.             plot_tree(
276.                 single_tree_to_plot,
277.                 feature_names=current_features,
278.                 filled=True,
279.                 rounded=True,
280.                 fontsize=8,
281.                 max_depth=5  # Batasi kedalaman plot agar terbaca
282.             )
283.             plt.title(f"Visualisasi Pohon Perwakilan - Model {name}", fontsize=16)
284.             plt.savefig(os.path.join(charts_dir, tree_plot_filename), bbox_inches='tight')
285.             plt.close()
286.             print(f"INFO: Visualisasi plot_tree untuk {name} berhasil disimpan.")
287.         except Exception as e:
288.             print(f"ERROR saat membuat visualisasi plot_tree untuk {name}: {e}")
289.             tree_plot_filename = None
290. # Kalkulasi metrik
291.         n, p = X_test.shape
292.         residuals = y_test - y_pred
293.         comparison_dist = rng.normal(0, np.std(residuals), len(residuals))
294.         ks_statistic, ks_pvalue = ks_2samp(residuals, comparison_dist)
295.
296.         all_results[name] = {
297.             'mae': mean_absolute_error(y_test, y_pred),
298.             'mse': mean_squared_error(y_test, y_pred),
299.             'rmse': np.sqrt(mean_squared_error(y_test, y_pred)),
300.             'r2_score': r2,
301.             'adjusted_r2': 1 - (1 - r2) * (n - 1) / (n - p - 1) if (n - p - 1) > 0 else r2,
302.             'mape': mean_absolute_percentage_error(y_test, y_pred) * 100,
303.             'smape': calculate_smape(y_test.values, y_pred),
304.             'ks_stat': ks_statistic,
305.             'ks_pvalue': ks_pvalue,
306.             'scatter_plot_path': f'charts/{scatter_filename}',
307.             'feature_importance_plot_path': f'charts/{importance_filename}',
308.             'tree_visualization_path': f'charts/{tree_plot_filename}' if tree_plot_filename else None,
309.             'best_params': grid_search.best_params_
310.         }
```

```
311.
312.     load_all_dependencies()
313.     return all_results
314.
315. @app.route('/train', methods=['POST'])
316. def train_model_endpoint():
317.     try:
318.         data = request.get_json(force=True)
319.         if not data: raise ValueError("Data JSON kosong atau
  tidak valid.")
320.     except Exception as e:
321.         return jsonify({
322.             "status": "error",
323.             "message": f"Server gagal mem-parsing permintaan
  sebagai JSON. Mimetype: '{request.mimetype}'. Error: {e}"
324.         }), 400
325.
326.     if 'file_path' not in data:
327.         return jsonify({"status": "error", "message": "Kunci
  'file_path' tidak ditemukan dalam data JSON."}), 400
328.
329.     file_path    =    os.path.join(os.path.dirname(__file__),
  data['file_path'])
330.     if not os.path.exists(file_path):
331.         return jsonify({"status": "error", "message": f"File
  tidak ada di path server: {file_path}"}), 400
332.     results = train_and_evaluate_from_file(file_path)
333.     if "error" in results:
334.         return   jsonify({"status":   "error",   "message":
  results["error"]}), 500
335.     try:
336.         os.remove(file_path)
337.     except Exception as e:
338.         print(f"Gagal menghapus file sementara: {e}")
339.     return jsonify({"status": "success", "results": results})
340.
341. @app.route('/reload-recommendations', methods=['POST'])
342. def reload_recommendations_endpoint():
343.     """
344.     Endpoint  untuk  memuat  ulang  file  rekomendasi  (dan
  dependensi lainnya)
345.     tanpa perlu me-restart server Flask.
346.     """
347.     print("INFO:  Menerima  permintaan  untuk  memuat  ulang
  rekomendasi...")
348.     if load_all_dependencies():
349.         return   jsonify({"status":   "success",   "message":
  "Rekomendasi berhasil dimuat ulang."}), 200
350.     else:
351.         return jsonify({"status": "error", "message": "Gagal
  memuat ulang rekomendasi."}), 500
352. # ----------------------------------
353. if __name__ == '__main__':
354.     load_all_dependencies()
355.     app.run(host='0.0.0.0', port=5000, debug=True)
```

Lampiran 4

Source Code Process_Prediction.php

```
<!?php
require_once '../config.php'; // Include config.php
ensure_logged_in(); // Pastikan user sudah login dan sesi dimulai
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tahun = filter_input(INPUT_POST, 'tahun', FILTER_VALIDATE_INT);
    $bulan = filter_input(INPUT_POST, 'bulan', FILTER_VALIDATE_INT);
    $kecamatan = filter_input(INPUT_POST, 'kecamatan', FILTER_SANITIZE_STRING);
    $faktor_inputs = $_POST['factors'] ?? []; // Input dari form (bisa kosong)

    if ($tahun === false || $tahun < 2020 || $bulan === false || !($bulan -->= 1 && $bulan <= 12) || empty($kecamatan)) {
        $_SESSION['message'] = "Parameter prediksi (Tahun, Bulan, Kecamatan) tidak lengkap atau tidak valid. Harap isi dengan benar.";
        $_SESSION['message_type'] = "error";
        session_write_close();
        redirect(BASE_URL . 'pages/predict_form.php');
    }
  // --- Logika Penentuan Faktor Penyebab dan Sumber Data ---
    $final_factors_for_api = []; // Ini akan berisi nilai numerik yang dikirim ke API
    $input_factors_used_for_db_display = []; // Ini akan berisi string penjelasan (untuk disimpan ke DB & ditampilkan)

    // Dapatkan semua nama fitur yang mungkin dibutuhkan model dari API (fallback ke daftar gabungan)
    // Ini penting agar kita tahu fitur apa saja yang perlu diisi, bahkan jika user tidak menginputnya.
    // Untuk ini, kita bisa hardcode daftar faktor yang lengkap dari predict_form.php / app.py
    $all_possible_features_php = [
        'penduduk(ribu)', 'kepadatan', 'jumlah faskes', 'Jarak Faskes (km)', 'pola Hidup bersih dan sehat',
        'jumlah curah hujan(mm)', 'RH_avg kelembapan rata-rata(%)', 'Tavg: Temperatur rata rata(°C)', 'Panjang Drainase(m)',
        'ventilasi layak %', 'Polusi PM2.5(µg/m³)', 'akses air bersih%', 'tempat buang air besar sehat %', 'perilaku cuci tangan %'
    ];

    // Ambil data historis dari database sesuai hierarki baru
    $historical_factors_row = get_historical_factors_for_prediction($pdo, $kecamatan, $tahun, $bulan);

    // Debugging - lihat apa yang didapat dari database
```

```
33.         error_log("Historical data for $kecamatan, $tahun-
    $bulan: " . print_r($historical_factors_row, true));
34.         foreach ($all_possible_features_php as $feature_name)
    {
35.             $manual_input = $faktor_inputs[$feature_name] ??
    null;
36.
37.             if ($manual_input !== null && $manual_input !==
    '') {
38.                 $final_factors_for_api[$feature_name]       =
    floatval($manual_input);
39.                 $input_factors_used_for_db_display[$feature_n
    ame] = "Manual: " . htmlspecialchars($manual_input);
40.             }     elseif     ($historical_factors_row     &&
    isset($historical_factors_row[$feature_name])) {
41.                 $final_factors_for_api[$feature_name]       =
    floatval($historical_factors_row[$feature_name]);
42.                 $input_factors_used_for_db_display[$feature_n
    ame]      =      "Dari     DB     Historis:      "      .
    htmlspecialchars($historical_factors_row[$feature_name]);
43.
44.                 // Debugging
45.                 error_log("Using    historical    data    for
    $feature_name: " . $historical_factors_row[$feature_name]);
46.             } else {
47.                 $final_factors_for_api[$feature_name] = null;
48.                 $input_factors_used_for_db_display[$feature_n
    ame] = "Rata-rata Historis";
49.
50.                 // Debugging
51.                 error_log("Using average for $feature_name -
    no historical data found");
52.             }
53.         }
54.     // --- Konfigurasi Endpoint API Python Flask ---
55.     $api_url = 'http://127.0.0.1:5000/predict';
56.
57.     $post_data = [
58.         'tahun' => $tahun,
59.         'bulan' => $bulan,
60.         'kecamatan' => $kecamatan,
61.         'faktor_penyebab'    =>    $final_factors_for_api    //
    Mengirim nilai numerik (atau null) ke API
62.     ];
63.
64.     $json_post_data = json_encode($post_data);
65.
66.     $ch = curl_init($api_url);
67.     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
68.     curl_setopt($ch, CURLOPT_POST, true);
69.     curl_setopt($ch, CURLOPT_POSTFIELDS, $json_post_data);
70.     curl_setopt($ch,   CURLOPT_HTTPHEADER,   [   'Content-Type:
    application/json',         'Content-Length:         '         .
    strlen($json_post_data) ]);
71.     curl_setopt($ch, CURLOPT_TIMEOUT, 60);
72.
```

```
73.     $response = curl_exec($ch);
74.     $http_status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
75.     $curl_error = curl_error($ch);
76.     curl_close($ch);
77.
78.     // --- Tangani Respons dari API ---
79.     if ($curl_error) {
80.         $_SESSION['message'] = "Kesalahan saat menghubungi
    server prediksi: " . htmlspecialchars($curl_error);
81.         $_SESSION['message_type'] = "error";
82.     } elseif ($http_status != 200) {
83.         $_SESSION['message'] = "Server prediksi mengembalikan
    status " . htmlspecialchars($http_status) . ". Respons: " .
    htmlspecialchars($response);
84.         $_SESSION['message_type'] = "error";
85.     } else {
86.         $api_response = json_decode($response, true);
87.
88.         if (json_last_error() !== JSON_ERROR_NONE) {
89.             $_SESSION['message'] = 'Error saat mendekode
    respons JSON dari API: ' . json_last_error_msg() . '. Respons
    mentah: ' . htmlspecialchars($response);
90.             $_SESSION['message_type'] = "error";
91.         }     elseif     (isset($api_response['status'])     &&
    $api_response['status'] == 'error') {
92.             $_SESSION['message']   =   "API   Prediksi:   "   .
    htmlspecialchars($api_response['message']);
93.             $_SESSION['message_type'] = "error";
94.         } else {
95.             $dbd_data = $api_response['results']['DBD'];
96.             $tbc_data = $api_response['results']['TBC'];
97.             $diare_data = $api_response['results']['Diare'];
98.             //            $final_inputs_from_api           =
    $api_response['final_inputs_used']; // Ini tidak digunakan
    lagi karena PHP yang membangun stringnya
99.
100.            // --- Logika untuk Menyimpan Hasil Prediksi ke
    Database ---
101.            try {
102.                $pdo->beginTransaction();
103.                $stmt_summary = $pdo->prepare("INSERT INTO
    predictions_summary (user_id, tahun_prediksi, bulan_prediksi,
    kecamatan_prediksi, input_factors_used_json) VALUES (?, ?, ?,
    ?, ?)");
104.                $stmt_summary->execute([
105.                    $_SESSION['user_id'],
106.                    $tahun,
107.                    $bulan,
108.                    $kecamatan,
109.                    json_encode($input_factors_used_for_db_d
    isplay) // Simpan string penjelasan dari PHP
110.                ]);
111.                $summary_id = $pdo->lastInsertId();
112.
113.                $stmt_details = $pdo->prepare("INSERT INTO
    prediction_details (prediction_summary_id, jenis_penyakit,
```

```
    predicted_cases,          klasifikasi,          rekomendasi,
    predicted_percentage) VALUES (?, ?, ?, ?, ?, ?)");
114.                 $stmt_details->execute([$summary_id,  'DBD',
    $dbd_data['predicted_cases'],      $dbd_data['spread_level'],
    $dbd_data['recommendation'], $dbd_data['percentage']]);
115.                 $stmt_details->execute([$summary_id,  'TBC',
    $tbc_data['predicted_cases'],      $tbc_data['spread_level'],
    $tbc_data['recommendation'], $tbc_data['percentage']]);
116.                 $stmt_details->execute([$summary_id,
    'Diare',                        $diare_data['predicted_cases'],
    $diare_data['spread_level'],    $diare_data['recommendation'],
    $diare_data['percentage']]);
117.                 $pdo->commit();
118.                 $_SESSION['message']  =  "Prediksi  berhasil
    disimpan ke database!";
119.                 $_SESSION['message_type'] = "success";
120.
121.                 // Simpan SEMUA hasil prediksi ke SESSION
    untuk ditampilkan
122.                 $_SESSION['prediction_output'] = [
123.                     'id_summary' => $summary_id,
124.                     'tahun' => $tahun,
125.                     'bulan' => $bulan,
126.                     'kecamatan' => $kecamatan,
127.                     'dbd' => $dbd_data,
128.                     'tbc' => $tbc_data,
129.                     'diare' => $diare_data,
130.                     'input_factors'                          =>
    $input_factors_used_for_db_display  //  BARU:  Kirim  string
    penjelasan ke output
131.                 ];
132.             } catch (PDOException $e) {
133.                 $pdo->rollBack();
134.                 $_SESSION['message'] = "Error saat menyimpan
    prediksi ke database: " . $e->getMessage();
135.                 $_SESSION['message_type'] = "error";
136.             }
137.             session_write_close();
138.             redirect(BASE_URL                                .
    'pages/prediction_result.php');
139.         }
140.     }
141.     session_write_close();
142.     redirect(BASE_URL . 'pages/predict_form.php');
143. } else {
144.     session_write_close();
145.     redirect(BASE_URL . 'pages/predict_form.php');
146. }
```