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	: MODEL PREDIKSI BIAYA EKSPOR : STUDI PEMILIHAN FITUR DAN PERBANDINGAN ALGORITMA LINEAR REGRESSION DAN RANDOM FOREST REGRESSION
Penulis	: Muhammad Rafli Pratama
NIM	: **2021230063**
Tgl pemeriksaan	: 25 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **17%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

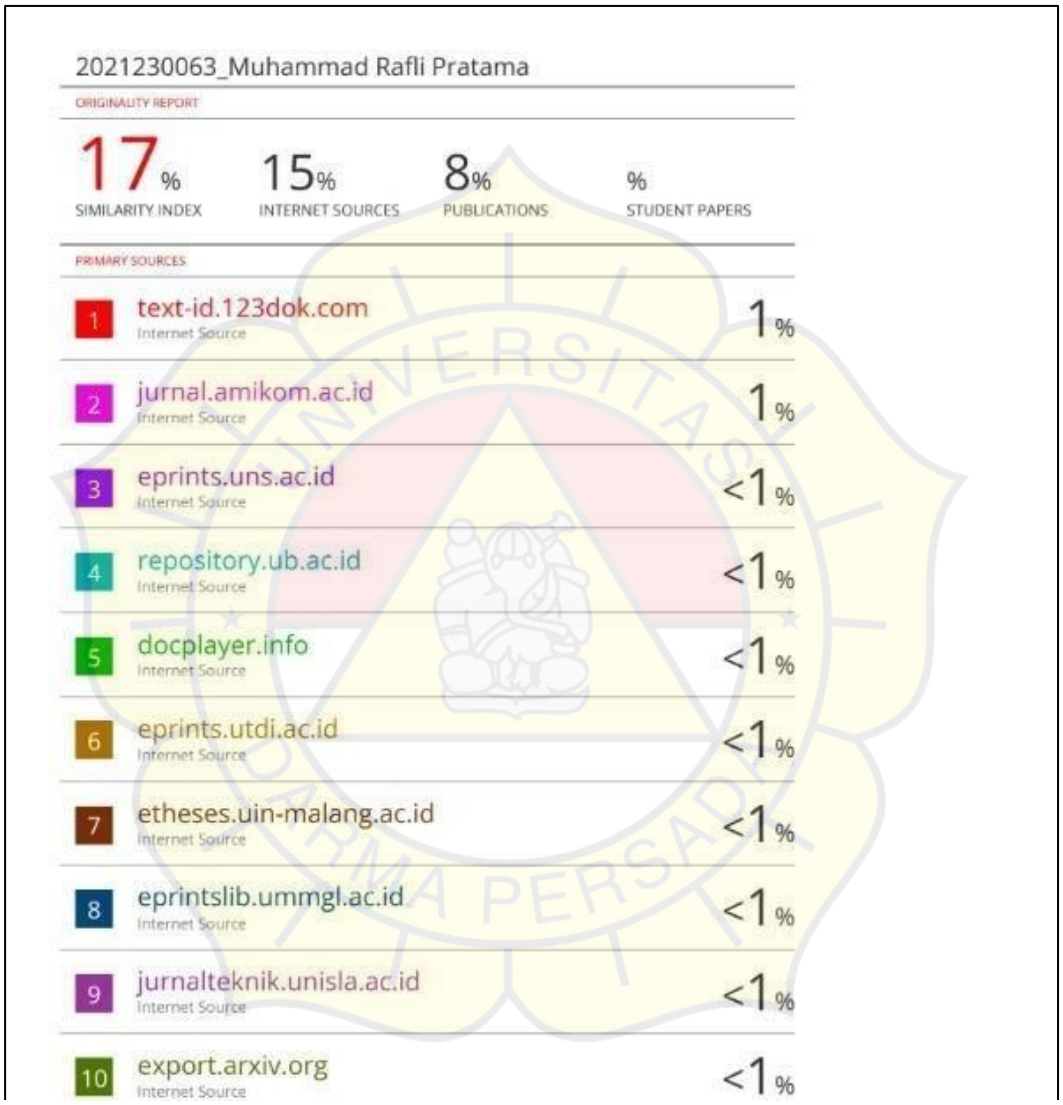
Jakarta, 25 Juli 2025

Ka.UPT Perpustakaan Unsada

Yus Rusmiyati, SS., MM

Lampiran 2

Hasil Turniti



2021230063_Muhammad Rafli Pratama

ORIGINALITY REPORT

17%	15%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	text-id.123dok.com Internet Source	1%
2	jurnal.amikom.ac.id Internet Source	1%
3	eprints.uns.ac.id Internet Source	<1%
4	repository.ub.ac.id Internet Source	<1%
5	docplayer.info Internet Source	<1%
6	eprints.utdi.ac.id Internet Source	<1%
7	etheses.uin-malang.ac.id Internet Source	<1%
8	eprintslib.ummgl.ac.id Internet Source	<1%
9	jurnalteknik.unisla.ac.id Internet Source	<1%
10	export.arxiv.org Internet Source	<1%

Source Code

app.py

File: python_scripts/app.py

```
1. import os
2. import json
3. import joblib
4. import pandas as pd
5. from flask import Flask, request, jsonify
6. from flask_cors import CORS
7. import numpy as np
8. import sys
9. import subprocess
10.  from sklearn.ensemble import RandomForestRegressor
11.  from sklearn.utils import resample


12.  app = Flask(__name__)
13.  CORS(app)

14.  # --- Konfigurasi Lokasi Model ---
15.  PATH_TO_MODELS  =  os.path.join(os.path.dirname(__file__),
   '..', 'ml_models')

16.  # --- Muat Model dan Data Pendukung Sekali Saat Aplikasi
  Dimulai ---
17.  model = None
18.  encoders = None
19.  features_used_during_training = None
20.  model_algo = "Tidak Dikenal"
21.  model_r2 = None

22.  try:
23.  model_path_rf        =        os.path.join(PATH_TO_MODELS,
   'best_rf_model.pkl')
24.  model_path_lr        =        os.path.join(PATH_TO_MODELS,
   'best_lr_model.pkl')
25.  encoder_path         =        os.path.join(PATH_TO_MODELS,
   'encoders.pkl')
26.  features_path        =        os.path.join(PATH_TO_MODELS,
   'features.pkl')

27.  if os.path.exists(model_path_rf):
28.  model = joblib.load(model_path_rf)
```

```
29.  model_algo = "Random Forest Regression"
30.  elif os.path.exists(model_path_lr):
31.  model = joblib.load(model_path_lr)
32.  model_algo = "Linear Regression"

33.  if os.path.exists(encoder_path):
34.  encoders = joblib.load(encoder_path)

35.  if os.path.exists(features_path):
36.  features_used_during_training                                =
   joblib.load(features_path)

37.  metrics_path        =        os.path.join(PATH_TO_MODELS,
   'model_metrics.json')
38.  if os.path.exists(metrics_path):
39.  with open(metrics_path, 'r') as f:
40.  metrics = json.load(f)
41.  if "Best Model Summary" in metrics and "r_squared" in
   metrics["Best Model Summary"]:
42.  model_r2 = metrics["Best Model Summary"]["r_squared"]

43.  except Exception as e:
44.  model, encoders, features_used_during_training = None,
   None, None


45.  @app.route('/predict', methods=['POST'])
46.  def predict():
47.  global model, encoders, features_used_during_training,
   model_algo, model_r2

48.  if   model   is   None   or   encoders   is   None   or
   features_used_during_training is None:
49.  return jsonify({"status": "error", "message": "Model belum
   dimuat."}), 500

50.  if not request.is_json:
51.  return jsonify({"status": "error", "message": "Input harus
   JSON."}), 400

52.  data = request.get_json()

53.  required_fields      =      ['weight_kg',      'volume_m3',
   'destination_country',
54.  'commodity_type',   'vessel_type',   'insurance_cost_usd',
   'customs_fee_usd']
```

```
55.  for field in required_fields:
56.  if field not in data:
57.  return jsonify({"status": "error", "message": f"Field
   '{field}' wajib diisi."}), 400

58.  try:
59.  val_country = str(data['destination_country'])
60.  le_country = encoders['destination_country']
61.  if val_country not in le_country.classes_:
62.  le_country.classes_    =    np.append(le_country.classes_,
   val_country)
63.  country_encoded = le_country.transform([val_country])[0]

64.  val_commodity = str(data['commodity_type'])
65.  le_commodity = encoders['commodity_type']
66.  if val_commodity not in le_commodity.classes_:
67.  le_commodity.classes_ = np.append(le_commodity.classes_,
   val_commodity)
68.  commodity_encoded                                    =
   le_commodity.transform([val_commodity])[0]

69.  val_vessel = str(data['vessel_type'])
70.  le_vessel = encoders['vessel_type']
71.  if val_vessel not in le_vessel.classes_:
72.  le_vessel.classes_     =     np.append(le_vessel.classes_,
   val_vessel)
73.  vessel_encoded = le_vessel.transform([val_vessel])[0]

74.  input_dict = {
75.  'weight_kg': float(data['weight_kg']),
76.  'volume_m3': float(data['volume_m3']),
77.  'destination_country_encoded': country_encoded,
78.  'commodity_type_encoded': commodity_encoded,
79.  'vessel_type_encoded': vessel_encoded,
80.  'insurance_cost_usd': float(data['insurance_cost_usd']),
81.  'customs_fee_usd': float(data['customs_fee_usd'])
82.  }

83.  except Exception as e:
84.  return jsonify({"status": "error", "message": f"Encoding
   gagal: {e}"}), 400

85.  X_input     =     [input_dict[feat]     for     feat     in
   features_used_during_training]

86.  try:
```

```
87.  pred = model.predict([X_input])[0]

88.  # Tambahkan Confidence Interval (jika model Random Forest)
89.  ci_lower = ci_upper = None
90.  if isinstance(model, RandomForestRegressor):
91.  preds = [estimator.predict([X_input])[0] for estimator in
  model.estimators_]
92.  std_dev = np.std(preds)
93.  ci_lower = pred - 1.96 * std_dev
94.  ci_upper = pred + 1.96 * std_dev

95.  return jsonify({
96.  "status": "success",
97.  "predicted_cost": float(pred),
98.  "confidence_interval":          [round(ci_lower,          2),
  round(ci_upper, 2)] if ci_lower and ci_upper else None,
99.  "model_used": model_algo,
100. "model_r2": model_r2
101. })
102. except Exception as e:
103. return jsonify({"status": "error", "message": f"Prediksi
  gagal: {e}"}), 500



104. @app.route('/rectangle_area', methods=['POST'])
105. def rectangle_area():
106. if not request.is_json:
107. return jsonify({"status": "error", "message": "Input harus
  berupa JSON."}), 400

108. data = request.get_json()
109. if 'length' not in data or 'width' not in data:
110. return  jsonify({"status":  "error",  "message":  "Field
  'length' dan 'width' wajib diisi."}), 400

111. try:
112. length = float(data['length'])
113. width = float(data['width'])
114. area = length * width
115. return jsonify({
116. "status": "success",
117. "length": length,
118. "width": width,
119. "area": area
120. })
```

```
except Exception as e:
return jsonify({"status": "error", "message": f"Perhitungan gagal: {e}"}), 400

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

# --- Endpoint Tambahan: Treatment per Komoditas ---
@app.route('/analyze/product_treatment', methods=['GET'])
def analyze_product_treatment():
try:
treatment_data = [
{
"commodity_type": "Elektronik",
"treatment": (
"Barang elektronik sangat sensitif terhadap suhu ekstrem, kelembaban tinggi, dan getaran selama pengiriman. "
"Dibutuhkan kemasan pelindung anti-statis, pengganjal busa EVA, serta kontainer dengan suhu terkendali dan shock absorber "
"untuk mencegah kerusakan fisik atau gangguan kelistrikan. Pemantauan suhu dan kelembaban menggunakan sensor juga disarankan."
)
},
{
"commodity_type": "Tekstil",
"treatment": (
"Tekstil rentan terhadap kelembapan yang dapat menyebabkan jamur atau perubahan warna. "
"Dibutuhkan kemasan tahan air, pengering desiccant silica gel, dan kontainer yang kering dan berventilasi baik. "
"Penyimpanan dalam suhu ruangan yang stabil (18-24°C) penting agar tidak merusak serat kain."
)
},
{
"commodity_type": "Makanan Beku",
"treatment": (
"Memerlukan rantai dingin (cold chain logistics) sepanjang perjalanan. Harus dikirim menggunakan kontainer berpendingin "
"(reefer container) dengan suhu di bawah -18°C. Pengemasan juga menggunakan insulated box dengan gel ice packs atau dry ice, "
"dan harus cepat untuk menghindari thawing (pencairan)."
```

```
152. )
153. },
154. {
155. "commodity_type": "Otomotif",
156. "treatment": (
157. "Komponen otomotif perlu perlindungan dari getaran dan
   goresan, serta pengemasan khusus seperti foam padding, shrink
   wrap, dan pelindung logam. "
158. "Selain itu, harus ada label berat dan arah posisi untuk
   mencegah kerusakan saat bongkar muat. Kontainer harus
   tertutup rapat dan aman dari debu."
159. )
160. },
161. {
162. "commodity_type": "Farmasi",
163. "treatment": (
164. "Produk farmasi sangat sensitif terhadap cahaya, suhu,
   dan kontaminasi. Pengiriman dilakukan dengan kondisi suhu
   terkendali "
165. "(2-8°C atau 15-25°C) tergantung jenis obat. Dibutuhkan
   kontainer termal berinsulasi, data logger suhu, dan kemasan
   anti-radiation & anti-bakteri. "
166. "Waktu pengiriman harus sesingkat mungkin."
167. )
168. },
169. {
170. "commodity_type": "NATURAL RUBBER",
171. "treatment": (
172. "Karet alami harus dilindungi dari kelembapan berlebih
   dan suhu tinggi yang bisa menyebabkan oksidasi atau
   penggumpalan. "
173. "Gunakan liner plastik PE, pallet non-kayu, dan simpan di
   ruang yang kering, teduh dan berventilasi. Hindari sinar
   matahari langsung."
174. )
175. },
176. {
177. "commodity_type": "PALM OIL",
178. "treatment": (
179. "Minyak sawit perlu pengiriman dalam kondisi hangat (35–
   45°C) agar tetap cair. Gunakan isotank atau flexitank untuk
   pengiriman massal "
180. "dan drum baja untuk volume kecil. Seluruh proses harus
   menjaga kebersihan tinggi dan menghindari oksidasi atau
   pencemaran bahan kimia lain."
181. )
```

```
182. },
183. {
184. "commodity_type": "COFFEE",
185. "treatment": (
186. "Kopi (biji atau bubuk) sangat mudah menyerap bau dan
   lembap. Diperlukan kemasan kedap udara berbahan aluminium
   foil atau vakum, "
187. "serta disimpan di ruang kering dan sejuk (15–20°C).
   Hindari sinar matahari langsung dan fluktuasi suhu. "
188. "Bisa juga dikirim dalam kontainer dengan dehumidifier."
189. )
190. },
191. {
192. "commodity_type": "WOODEN FURNITURE",
193. "treatment": (
194. "Mebel kayu sensitif terhadap kelembapan tinggi dan
   benturan. Perlu pengeringan oven (kiln-dried) sebelum
   pengemasan, "
195. "serta lapisan bubble wrap dan karton berlapis kayu.
   Kontainer harus kedap air dan diberi pengganjal untuk
   menghindari gesekan antar barang."
196. )
197. },
198. {
199. "commodity_type": "Shrimp (Udang)",
200. "treatment": (
201. "Udang harus dikirim segera setelah pembekuan dalam suhu
   - 18°C atau lebih rendah. Gunakan dry ice, insulated boxes,
   dan kontainer reefer "
202. "dengan sistem pemantauan suhu. Hindari pembukaan
   berulang yang bisa memecah rantai dingin. Waktu pengiriman
   maksimal 48–72 jam."
203. )
204. },
205. {
206. "commodity_type": "SPICES (Rempah)",
207. "treatment": (
208. "Rempah-rempah rentan terhadap kelembaban, jamur, dan
   kehilangan aroma. Gunakan kemasan aluminium foil atau plastik
   multi-layer yang kedap udara. "
209. "Simpan dalam suhu ruang sejuk dan kering. Hindari cahaya
   langsung dan kontaminasi silang. Perlu kontainer bersih
   dengan humidity absorber."
210. )
211. }
212. ]
```

```
213. return jsonify({
214. "status": "success",
215. "data": treatment_data
216. })

217. except Exception as e:
218. return jsonify({
219. "status": "error",
220. "message": f"Gagal memproses treatment: {e}"
221. }), 500

222. @app.route('/train', methods=['POST'])
223. def train_model():
224. try:
225. script_path = os.path.join(os.path.dirname(__file__),
   'train_model.py')
226. data_path = 'data/data_coba_coba.csv'
227. models_dir = 'ml_models'

228. result = subprocess.run(
229. ['python', script_path,
230. '--data_path', data_path,
231. '--ml_models_path', models_dir,
232. '--DB_SERVER', 'localhost',
233. '--DB_USERNAME', 'root',
234. '--DB_PASSWORD', '',
235. '--DB_NAME', 'prediksi_db'],
236. capture_output=True,
237. text=True
238. )

239. if result.returncode == 0:
240. output = result.stdout.strip()
241. return jsonify({"status": "success", "message": "Model
   berhasil dilatih ulang", "output": output})
242. else:
243. return jsonify({"status": "error", "message": "Gagal
   melatih model", "output": result.stderr}), 500

244. except Exception as e:
245. return jsonify({"status": "error", "message": str(e)}),
   500

246. # --- Endpoint Baru: Prediksi Durasi Keterlambatan ---
247. @app.route('/predict_delay', methods=['POST'])
```

```
def predict_delay():
try:
# Lokasi folder khusus model delay
model_dir = os.path.join(os.path.dirname(__file__), '..',
'ml_models_delay')

# Gunakan nama model sesuai file Anda (misal:
best_delay_model.pkl)
delay_model_path = os.path.join(model_dir,
'best_delay_model.pkl')
encoder_path = os.path.join(model_dir,
'encoders_delay.pkl')
features_path = os.path.join(model_dir,
'features_delay.pkl')

# Validasi keberadaan file model, encoder, dan fitur
if not (os.path.exists(delay_model_path) and
os.path.exists(encoder_path) and
os.path.exists(features_path)):
return jsonify({"status": "error", "message": "Model
delay atau encoder tidak ditemukan."}), 500

# Load model dan pendukungnya
delay_model = joblib.load(delay_model_path)
delay_encoders = joblib.load(encoder_path)
delay_features = joblib.load(features_path)

# Validasi request JSON
if not request.is_json:
return jsonify({"status": "error", "message": "Input
harus berupa JSON."}), 400

data = request.get_json()
processed_input = {}

# Proses input data sesuai fitur training
for feature in delay_features:
if feature.endswith('_encoded'):
col = feature.replace('_encoded', '')
if col in data and col in delay_encoders:
le = delay_encoders[col]
val = str(data[col])
if val not in le.classes_:
le.classes_ = np.append(le.classes_, val)
processed_input[feature] = le.transform([val])[0]
else:
```