DAFTAR 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 PENJUALAN BONEKA PADA F.R COLLECTION MENGGUNAKAN ARTIFICIAL NEURAL NETWORK (ANN) JENIS MULTI LAYER PERCEPTRON
Penulis	: Ahmad Fadhil Firmansyah
NIM	: **2021230048**
Tgl pemeriksaan	: 29 Juli 2025

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

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 29 Juli 2025

Ka.UPT Perpustakaan Unsada

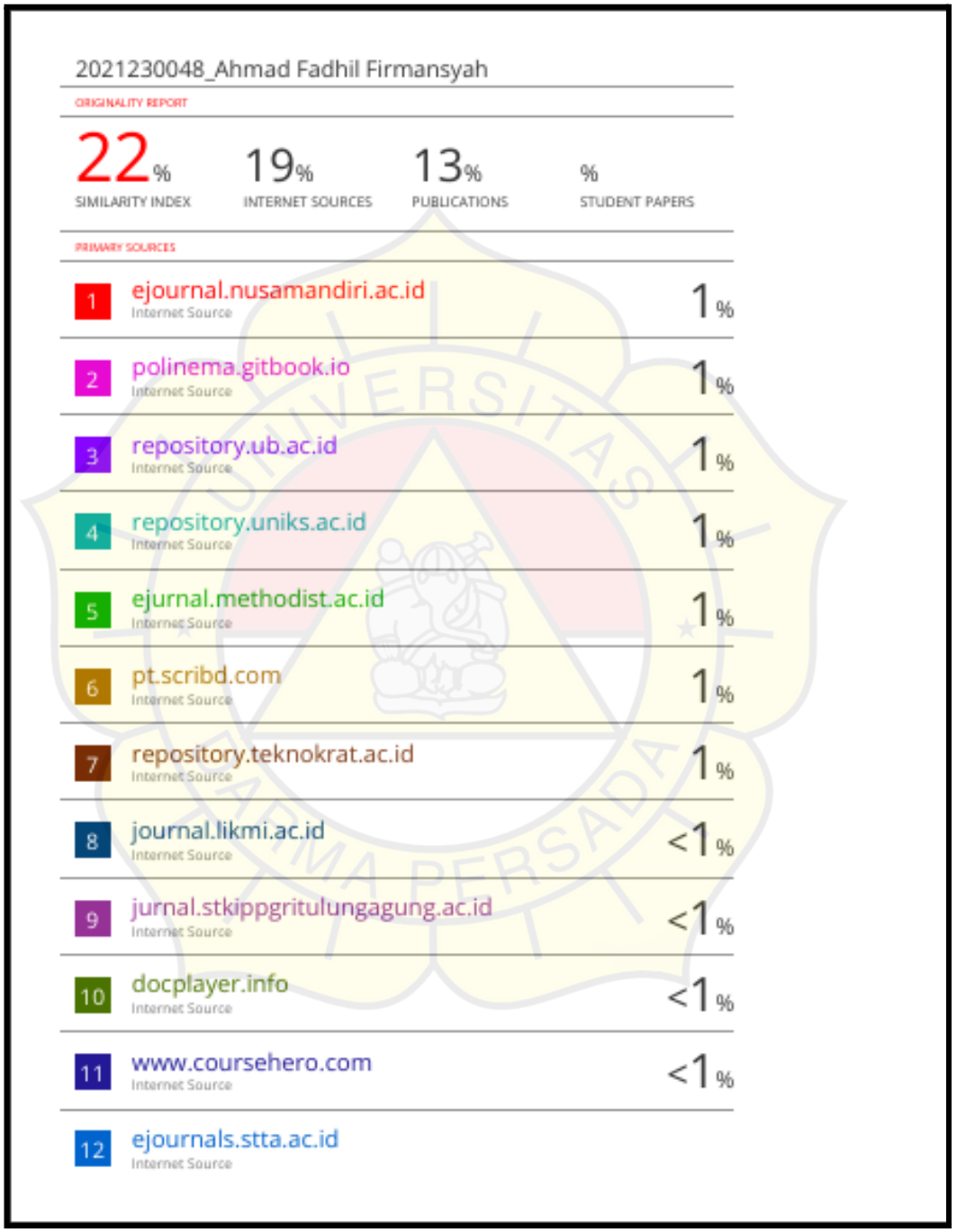
Yus Rusmiyati, SS., MM

Batas maksimal similarity 30% untuk Fakultas Sastra dan Ekonomi

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

Lampiran 2

Hasil Turnitin



2021230048_Ahmad Fadhil Firmansyah

ORIGINALITY REPORT

22%	19%	13%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ejournal.nusamandiri.ac.id Internet Source	1%
2	polinema.gitbook.io Internet Source	1%
3	repository.ub.ac.id Internet Source	1%
4	repository.uniks.ac.id Internet Source	1%
5	ejurnal.methodist.ac.id Internet Source	1%
6	pt.scribd.com Internet Source	1%
7	repository.teknokrat.ac.id Internet Source	1%
8	journal.likmi.ac.id Internet Source	<1%
9	jurnal.stkippgritulungagung.ac.id Internet Source	<1%
10	docplayer.info Internet Source	<1%
11	www.coursehero.com Internet Source	<1%
12	ejournals.stta.ac.id Internet Source	

Lampiran 3

Source code

run.py

```
from flask import Flask, render_template, request, session,
    redirect, url_for, flash, send_file
import pandas as pd
import joblib
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import os
import math
from datetime import datetime
from sklearn.metrics import mean_squared_error,
    mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.neural_network import MLPRegressor
from functools import wraps

app = Flask(__name__)
app.secret_key = 'rahasia_super_aman'

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
STATIC_DIR = os.path.join(BASE_DIR, 'static')
DATA_FILE = os.path.join(BASE_DIR, 'data', 'dataset.csv')
MODEL_FILE = os.path.join(BASE_DIR, 'model.pkl')

def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if 'user' not in session:
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function

@app.route('/')
@login_required
def dashboard():
    return render_template('dashboard.html')

@app.route('/produk')
def produk():
    return render_template('produk.html')

@app.route('/landing')
def landing():
    return render_template('landing.html')

@app.route('/training')
```

```
@login_required
def training():
    df = pd.read_csv(DATA_FILE)
    df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce')
    df['Tahun'] = df['Tanggal'].dt.year
    df['Bulan_Angka'] = df['Tanggal'].dt.month
    df['Hari'] = df['Tanggal'].dt.day
    df['Bulan'] = df['Tanggal'].dt.strftime('%B')
    df.drop(columns=['Tanggal'], inplace=True)

    X = df.drop(columns=['Target Penjualan'])
    y = df['Target Penjualan']

     categorical_cols = ['Kategori', 'Jenis', 'Musim/Event', 'Jenis
    Transaksi', 'Bulan']
     numeric_cols = ['Harga Satuan', 'Jumlah Unit', 'Total Harga',
    'Diskon', 'Stok Awal', 'Tahun', 'Bulan_Angka', 'Hari']

    for col in numeric_cols:
        X[col] = pd.to_numeric(X[col], errors='coerce')

    X = X.dropna()
    y = y[X.index]

      X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

    preprocessor = ColumnTransformer([
        ('num', StandardScaler(), numeric_cols),
                    ('cat',  OneHotEncoder(handle_unknown='ignore'),
    categorical_cols)
    ])

    pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', MLPRegressor(
        hidden_layer_sizes=(64, 32),   # 2 hidden layer: 64 neuron,
    lalu 32 neuron
        activation='relu',             # Fungsi aktivasi default
        solver='adam',                 # Optimizer yang efisien
        alpha=0.0001,                  # L2 regularization rate
        learning_rate='constant',     # Learning rate tidak berubah
    sepanjang training
         learning_rate_init=0.001,     # Besarnya langkah pembaruan
    bobot
        max_iter=1000,                # Iterasi maksimum training
        random_state=42               # Agar hasil konsisten setiap
    run
    ))
])


    pipeline.fit(X_train, y_train)
    joblib.dump(pipeline, MODEL_FILE)
    score_float = pipeline.score(X_test, y_test)
```

```
    score = f"{score_float:.2f}"  # format sebagai string, 2 angka
    desimal


    session['training_score'] = score

    return render_template('training.html', score=score)

@app.route('/prediksi', methods=['GET', 'POST'])
@login_required
def prediksi():
    prediction = None
    saran_stok = None

    if request.method == 'POST':
        try:
            data = {
                'Tanggal': request.form['Tanggal'],
                'Kategori': request.form['Kategori'],
                'Jenis': request.form['Jenis'],
                    'Harga Satuan': float(request.form['Harga
    Satuan']),
                'Jumlah Unit': int(request.form['Jumlah Unit']),
                'Total Harga': float(request.form['Total Harga']),
                'Diskon': float(request.form['Diskon']),
                'Musim/Event': request.form['Musim/Event'],
                'Stok Awal': int(request.form['Stok Awal']),
                'Jenis Transaksi': request.form['Jenis Transaksi']
            }

            df = pd.DataFrame([data])
            df['Tanggal'] = pd.to_datetime(df['Tanggal'])
            df['Hari'] = df['Tanggal'].dt.day
            df['Bulan'] = df['Tanggal'].dt.strftime('%B')
            df['Bulan_Angka'] = df['Tanggal'].dt.month
            df['Tahun'] = df['Tanggal'].dt.year
            df.drop(columns=['Tanggal'], inplace=True)

            model = joblib.load(MODEL_FILE)
            pred = model.predict(df)[0]
            prediction = math.ceil(pred)
            session['last_prediction'] = prediction

            stok_awal = data['Stok Awal']
            selisih = prediction - stok_awal
            ideal_stok_maks = math.ceil(prediction * 1.5)

            if stok_awal < prediction:
                saran_stok = (
                    f"⚠️ Stok tidak mencukupi.\n"
                    f"📦 Prediksi penjualan: {prediction} unit.\n"
                    f"📦 Stok tersedia: {stok_awal} unit.\n"
                     f"➕ Tambahkan setidaknya {selisih} unit untuk
    mencukupi permintaan."
                )
            elif stok_awal > ideal_stok_maks:
```

```
                saran_stok = (
                    f"⚠️ Stok terlalu banyak.\n"
                    f"📦 Prediksi penjualan: {prediction} unit.\n"
                    f"📦 Stok tersedia: {stok_awal} unit.\n"
                       f"📉 Stok ideal maksimal: {ideal_stok_maks}
    unit untuk efisiensi."
                )
            else:
                saran_stok = (
                    f"✅ Stok mencukupi.\n"
                    f"📦 Prediksi penjualan: {prediction} unit.\n"
                    f"📦 Stok tersedia: {stok_awal} unit."
                )

        except Exception as e:
            prediction = f'Terjadi kesalahan: {e}'

      return  render_template('index.html',  prediction=prediction,
    saran_stok=saran_stok)

@app.route('/evaluasi')
@login_required
def evaluasi():
    if not os.path.exists(MODEL_FILE):
         return render_template('evaluasi.html', error="Model belum
    ditraining.")

    df = pd.read_csv(DATA_FILE)
    df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce')
    df['Tahun'] = df['Tanggal'].dt.year
    df['Bulan_Angka'] = df['Tanggal'].dt.month
    df['Hari'] = df['Tanggal'].dt.day
    df['Bulan'] = df['Tanggal'].dt.strftime('%B')
    df.drop(columns=['Tanggal'], inplace=True)

    X = df.drop(columns=['Target Penjualan'])
    y = df['Target Penjualan']
     numeric_cols = ['Harga Satuan', 'Jumlah Unit', 'Total Harga',
    'Diskon', 'Stok Awal', 'Tahun', 'Bulan_Angka', 'Hari']

    for col in numeric_cols:
        X[col] = pd.to_numeric(X[col], errors='coerce')

    X = X.dropna()
    y = y[X.index]

      X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,
    test_size=0.2, random_state=42)

    model = joblib.load(MODEL_FILE)
    y_pred = model.predict(X_test)

    mse = round(mean_squared_error(y_test, y_pred), 2)
    mae = round(mean_absolute_error(y_test, y_pred), 2)
    r2 = round(r2_score(y_test, y_pred), 2)
```

```
    plt.figure(figsize=(10, 5))
    plt.plot(y_test.values[:50], label='Aktual', marker='o')
    plt.plot(y_pred[:50], label='Prediksi', marker='x')
    plt.title('Aktual vs Prediksi')
    plt.xlabel('Index')
    plt.ylabel('Penjualan')
    plt.legend()
    plot_path = os.path.join(STATIC_DIR, 'error_plot.png')
    plt.savefig(plot_path)
    plt.close()

    session['eval_result'] = {'mse': mse, 'mae': mae, 'r2': r2}

       return  render_template('evaluasi.html',  mse=mse,  mae=mae,
    r2=r2, plot_path='static/error_plot.png')

@app.route('/catat', methods=['GET', 'POST'])
@login_required
def catat():
    message = None

    if request.method == 'POST':
        try:
            form_data = {
                'Tanggal': request.form['Tanggal'],
                'Kategori': request.form['Kategori'],
                'Jenis': request.form['Jenis'],
                        'Harga  Satuan':  float(request.form['Harga
    Satuan']),
                'Jumlah Unit': int(request.form['Jumlah Unit']),
                'Total Harga': float(request.form['Total Harga']),
                'Diskon': float(request.form['Diskon']),
                'Musim/Event': request.form['Musim/Event'],
                'Stok Awal': int(request.form['Stok Awal']),
                'Jenis Transaksi': request.form['Jenis Transaksi'],
                    'Target  Penjualan':  float(request.form['Target
    Penjualan']),
                                                        'Bulan':
    pd.to_datetime(request.form['Tanggal']).month
            }

            session['last_entry'] = form_data

            new_row = pd.DataFrame([form_data])
            if os.path.exists(DATA_FILE):
                df = pd.read_csv(DATA_FILE)
                new_row = new_row[df.columns]
                            final_df  =  pd.concat([df,  new_row],
    ignore_index=True)
            else:
                final_df = new_row

            final_df.to_csv(DATA_FILE, index=False)
            message = "✅ Data berhasil dicatat."
        except Exception as e:
            message = f"❌ Gagal mencatat data: {e}"
```

```
    return render_template('catat.html', message=message)

@app.route('/resume')
@login_required
def resume():
     df = pd.read_csv(DATA_FILE) if os.path.exists(DATA_FILE) else
    pd.DataFrame()
    total_data = len(df)
      last_data = df.tail(1).to_dict(orient='records')[0] if not
    df.empty else {}

    return render_template('resume.html',
                           total_data=total_data,
                           last_data=last_data,

    prediction=session.get('last_prediction'),

    training_score=session.get('training_score'),
                           eval_result=session.get('eval_result'),
                           last_entry=session.get('last_entry'))

@app.route('/dataset')
def lihat_dataset():
    df = pd.read_csv(DATA_FILE)
    return render_template(
        'dataset.html',
                  tables=[df.to_html(classes='table  table-bordered
    table-striped', index=False)],
        titles=df.columns.values
    )

@app.route('/download-dataset')
def download_dataset():
    return send_file('static/dataset.csv' , as_attachment=True)

@app.route('/tes', methods=['GET', 'POST'])
@login_required
def tes():
    message = None

    if request.method == 'POST':
        try:
            form_data = {
                'Tanggal': request.form['Tanggal'],
                'Kategori': request.form['Kategori'],
                'Jenis': request.form['Jenis'],
                         'Harga Satuan': float(request.form['Harga
    Satuan']),
                'Jumlah Unit': int(request.form['Jumlah Unit']),
                'Total Harga': float(request.form['Total Harga']),
                'Diskon': float(request.form['Diskon']),
                'Musim/Event': request.form['Musim/Event'],
                'Stok Awal': int(request.form['Stok Awal']),
                'Jenis Transaksi': request.form['Jenis Transaksi'],
```

```
                    'Target Penjualan': float(request.form['Target
    Penjualan']),
                                                            'Bulan':
    pd.to_datetime(request.form['Tanggal']).month
            }

            session['last_entry'] = form_data

            new_row = pd.DataFrame([form_data])
            if os.path.exists(DATA_FILE):
                df = pd.read_csv(DATA_FILE)
                new_row = new_row[df.columns]
                            final_df = pd.concat([df, new_row],
    ignore_index=True)
            else:
                final_df = new_row

            final_df.to_csv(DATA_FILE, index=False)
            message = "✅ Data berhasil dicatat."
        except Exception as e:
            message = f"❌ Gagal mencatat data: {e}"

    return render_template('tes.html', message=message)

@app.route('/perbandingan')
def perbandingan():
    df = pd.read_csv('hasil_perbandingan_mlp.csv')
                    return    render_template('perbandingan.html',
    tables=df.to_dict(orient='records'))

@app.route('/bandingkan_model')
def bandingkan_model():
    os.system("python compare_mlp_configs.py")
    flash("Perbandingan model berhasil diperbarui!", "success")
    return redirect(url_for('perbandingan'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
                if  request.form['username']  ==  'admin'  and
    request.form['password'] == 'admin123':
            session['user'] = 'admin'
            flash('Login berhasil.', 'success')
            return redirect(url_for('dashboard'))
        flash('Login gagal.', 'danger')
    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    flash('Logout berhasil.', 'info')
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```