LAMPIRAN 1

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	: PENGELOMPOKAN KATEGORI MINUMAN KOPI BERDASARKAN KOMPOSISI BAHAN DAN KALORI DENGAN ALGORITMA PRINCIPAL COMPONENT ANALYSIS (PCA) DAN SUPPORT VECTOR MACHINE (SVM)
Penulis	: Nur Ariffiyatul Furaida
NIM	: **2021230027**
Tgl pemeriksaan	: 28 Juli 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **21%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

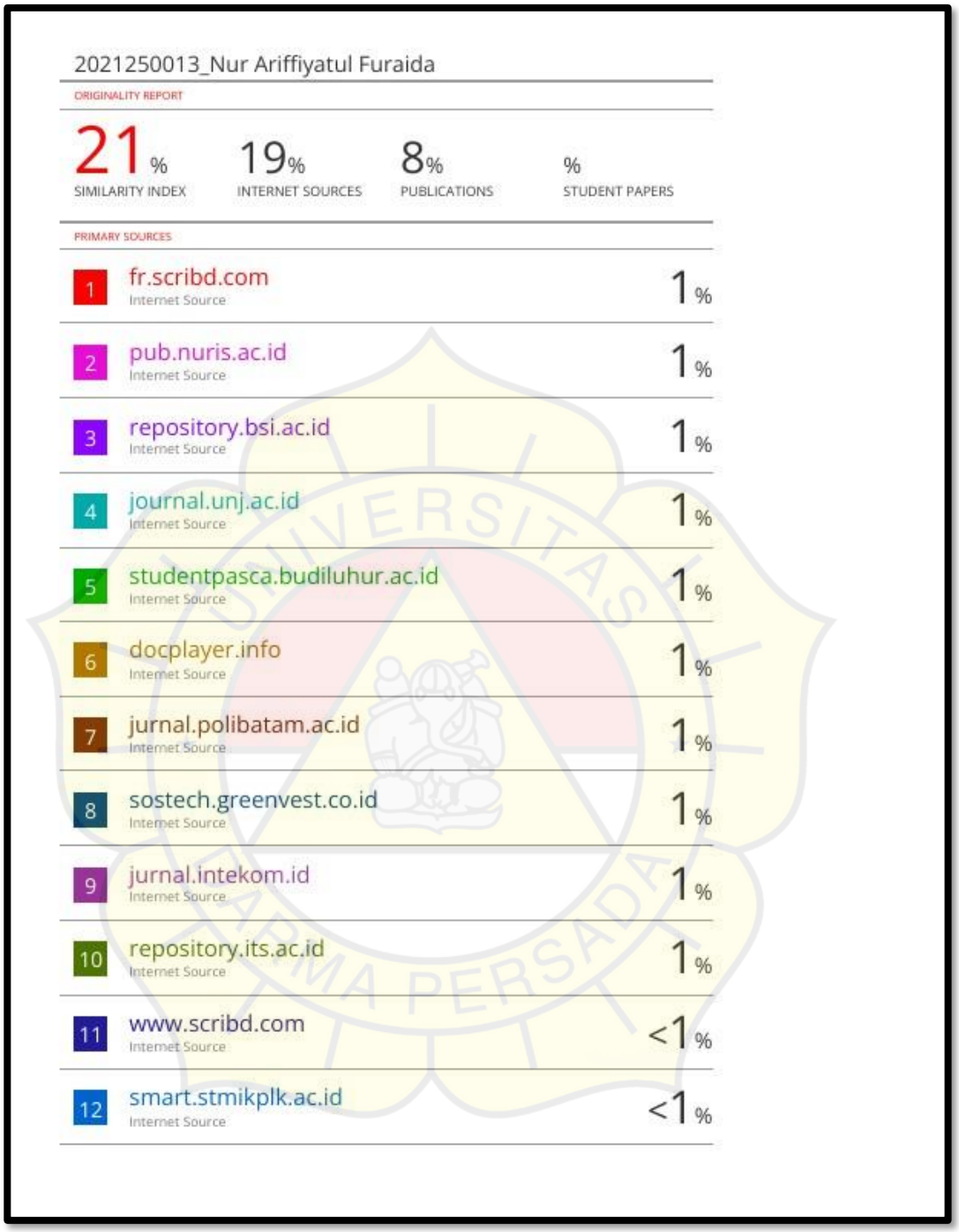
Jakarta, 28 Juli 2025

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



2021250013_Nur Ariffiyatul Furaida

ORIGINALITY REPORT

21%	19%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	fr.scribd.com Internet Source	1%
2	pub.nuris.ac.id Internet Source	1%
3	repository.bsi.ac.id Internet Source	1%
4	journal.unj.ac.id Internet Source	1%
5	studentpasca.budiluhur.ac.id Internet Source	1%
6	docplayer.info Internet Source	1%
7	jurnal.polibatam.ac.id Internet Source	1%
8	sostech.greenvest.co.id Internet Source	1%
9	jurnal.intekom.id Internet Source	1%
10	repository.its.ac.id Internet Source	1%
11	www.scribd.com Internet Source	<1%
12	smart.stmikplk.ac.id Internet Source	<1%

LAMPIRAN 2

Source Code Latih Algoritma PCA dan SVM

```
import streamlit as st

import pandas as pd

import mysql.connector

from sklearn.decomposition import PCA

from sklearn.svm import SVR

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, precision_score,
recall_score, confusion_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np


def create_connection():

    connection = mysql.connector.connect(

        host="localhost",

        user="root",

        password="",

        database="coffee_db"

    )

    return connection


def model_algorithm_page():

    st.subheader("Latih Model PCA dan SVM untuk Prediksi Kalori")
```

```
    # Upload file CSV opsional
    uploaded_file = st.file_uploader("Unggah File CSV (opsional)",
type=["csv"])

    if uploaded_file is not None:
        st.success("File berhasil diunggah.")
        coffee_data = pd.read_csv(uploaded_file)
        st.info("Data yang digunakan adalah hasil unggahan
file.")
    else:
        # Ambil data dari database jika tidak ada file yang
diunggah
        connection = create_connection()
        coffee_data = pd.read_sql("SELECT * FROM menu_coffee",
connection)
        connection.close()
        st.info("Data yang digunakan adalah dari database.")

    # Tampilkan data
    st.subheader("Data Menu Coffee")
    if not coffee_data.empty:
        st.dataframe(coffee_data)

        # Proses data dan latih model
        pca, svm, accuracy, precision, recall, conf_matrix,
predictions, coffee_data_clean, X, y = train_model(coffee_data)
```

```
        # Tombol untuk menampilkan hasil prediksi
        if st.button("Tampilkan Hasil Prediksi"):
            st.subheader("Hasil Prediksi untuk Semua Item Kopi")

            # Threshold untuk klasifikasi
            calorie_threshold =
coffee_data_clean['calories'].median()

            # Buat label berdasarkan threshold
            labels = ['Kalori Rendah' if p < calorie_threshold
else 'Kalori Tinggi' for p in predictions]
            true_labels = ['Kalori Rendah' if c <
calorie_threshold else 'Kalori Tinggi' for c in
coffee_data_clean['calories']]

            # Tabel hasil prediksi
            prediction_df = pd.DataFrame({
                'Item': coffee_data_clean['item'],
                'Prediksi Kalori': predictions,
                'Kategori': labels,
                'Prediksi Label': true_labels
            })

            st.write(prediction_df)

            # Visualisasi PCA
```

```
            st.subheader("Visualisasi PCA Berdasarkan Prediksi
Kalori (SVM) dengan Hyperplane")

            plot_svm_pca(coffee_data_clean, pca, predictions,
calorie_threshold, svm)


            # Evaluasi Model

            st.subheader("Evaluasi Model")

            evaluation_df = pd.DataFrame({

                'Metric': ['Accuracy', 'Precision', 'Recall'],

                'Value': [f"{accuracy:.2f}", f"{precision:.2f}",
f"{recall:.2f}"]

            })

            st.table(evaluation_df)


            # Confusion Matrix

            st.subheader("Confusion Matrix")

            ConfusionMatrixDisplay(confusion_matrix=conf_matrix,
display_labels=['Kalori Rendah', 'Kalori
Tinggi']).plot(cmap='Blues')

            st.pyplot(plt)


            # Analisis Otomatis

            st.subheader("Analisis Otomatis Dataset dan Model")

            jumlah_data = len(coffee_data_clean)

            jumlah_fitur = X.shape[1]

            jumlah_kategori = len(set(true_labels))

            tn, fp, fn, tp = conf_matrix.ravel()
```

```
            st.write(f"""
            **Jumlah Data Saat Ini:** {jumlah_data} item kopi
            **Jumlah Fitur Input:** {jumlah_fitur} fitur komposisi
kopi
            **Jumlah Kategori Kalori (Berdasarkan Threshold
Median):** {jumlah_kategori} kategori ('Kalori Rendah' & 'Kalori
Tinggi')

            ### **Analisis Hyperplane SVM dan Visualisasi PCA**
            - Model SVM dilatih menggunakan PCA dengan 2 komponen
utama.
            - Hyperplane (garis putus-putus hitam) menunjukkan
batas pemisah antara kategori kalori.
            - Distribusi titik akan berubah jika data berubah
(misal unggah file baru).

            ### **Analisis Confusion Matrix**
            - **True Positive (TP):** {tp}
            - **False Positive (FP):** {fp}
            - **False Negative (FN):** {fn}
            - **True Negative (TN):** {tn}

            ### **Kesimpulan**
            - **Akurasi:** {accuracy:.2f}
            - **Precision:** {precision:.2f}
            - **Recall:** {recall:.2f}
            """)
```

```
    else:
        st.warning("Tidak ada data yang tersedia untuk
pelatihan.")


def train_model(data):
    data_clean = data.dropna(subset=['calories'])
    X = data_clean.drop(columns=['item', 'tanggal_transaksi',
'customer'])
    y = data_clean['calories']


    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)


    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X_scaled)


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


    param_grid = {
        'C': [0.1, 1, 10, 100],
        'epsilon': [0.01, 0.1, 0.5],
        'kernel': ['linear', 'rbf']
    }


    svm = SVR()
```

```
    grid_search = GridSearchCV(svm, param_grid, cv=5,
scoring='neg_mean_squared_error')
    grid_search.fit(X_train, y_train)

    best_svm = grid_search.best_estimator_
    y_pred = best_svm.predict(X_test)

    calorie_threshold = y.median()
    y_pred_labels = ['Kalori Rendah' if p < calorie_threshold else
'Kalori Tinggi' for p in y_pred]
    y_test_labels = ['Kalori Rendah' if c < calorie_threshold else
'Kalori Tinggi' for c in y_test]

    accuracy = accuracy_score(y_test_labels, y_pred_labels)
    precision = precision_score(y_test_labels, y_pred_labels,
pos_label='Kalori Tinggi', average='binary')
    recall = recall_score(y_test_labels, y_pred_labels,
pos_label='Kalori Tinggi', average='binary')
    conf_matrix = confusion_matrix(y_test_labels, y_pred_labels)

    # Prediksi untuk semua data
    predictions = best_svm.predict(X_pca)

    return pca, best_svm, accuracy, precision, recall,
conf_matrix, predictions, data_clean, X, y

def plot_svm_pca(data, pca, predictions, calorie_threshold,
svm_model):
```

```
    # Transformasi data menggunakan PCA
    pca_components =
pca.transform(StandardScaler().fit_transform(data.drop(columns=['i
tem', 'tanggal_transaksi', 'customer'])))


    # Tentukan label berdasarkan threshold kalori
    labels = ['Kalori Rendah' if p < calorie_threshold else
'Kalori Tinggi' for p in predictions]


    # Tentukan grid untuk visualisasi
    x_min, x_max = pca_components[:, 0].min() - 1,
pca_components[:, 0].max() + 1
    y_min, y_max = pca_components[:, 1].min() - 1,
pca_components[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
np.linspace(y_min, y_max, 100))


    # Prediksi pada grid untuk menggambar hyperplane
    Z = svm_model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)


    # Plotting
    plt.figure(figsize=(10, 6))


    # Plot hyperplane sebagai garis pemisah
    plt.contour(xx, yy, Z, levels=[calorie_threshold],
colors='black', linestyles='--', linewidths=1.5,
label='Hyperplane')
```

```
    # Plot data points
    sns.scatterplot(
        x=pca_components[:, 0],
        y=pca_components[:, 1],
        hue=labels,
        palette={'Kalori Rendah': 'blue', 'Kalori Tinggi': 'red'},
        edgecolor='w',
        linewidth=0.5
    )

    plt.title("PCA Komponen Berdasarkan Prediksi Kalori (SVM) dengan Hyperplane")
    plt.xlabel("Komponen PCA 1")
    plt.ylabel("Komponen PCA 2")
    plt.legend(title="Kategori")
    st.pyplot(plt)

# Jalankan di halaman Streamlit
if __name___ == "__main__":
    model_algorithm_page()
```