LAMPIRAN

Surat Hasil Turnitin

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 WILAYAH RAWAN BANJIR BERDASAKAN DAMPAK KORBAN DAN KERUSAKAN MENGGUNAKAN ALGORITMA K-MEANS

Penulis : **KAREEN HERMINTIA KAHAR**

NIM : **2021230030**

Tgl pemeriksaan : 5 Februari 2025

Dengan hasil Tingkat Kesamaan (*similarity index*) **18%**

Demikian Surat Keterangan kami buat, untuk dipergunakan sebagaimana mestinya.

Jakarta, 5 Februari 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

Kode program cluster.py

```
1. import streamlit as st
2. import pandas as pd
3. import numpy as np
4. import matplotlib.pyplot as plt
5. from sklearn.cluster import KMeans
6. from sklearn.preprocessing import StandardScaler
7. from sklearn.metrics import silhouette_score
8. from io import BytesIO
9. from mpl_toolkits.mplot3d import Axes3D
10.      from utils import create_connection  # Impor fungsi
   koneksi ke database
11.      # Fungsi untuk memuat data dari database
12.      def load_data_from_database():
13.      conn = create_connection()  # Membuka koneksi
14.      query = "SELECT * FROM wilayah_banjir"
15.      data = pd.read_sql(query, conn)
16.      conn.close()  # Menutup koneksi setelah pengambilan
   data
17.      data['Tahun'] = data['Tahun'].astype(str)  # Pastikan
   kolom Tahun ditampilkan sebagai string
18.      return data
19.      def show_cluster():
20.      st.title("Clustering Wilayah Berdasarkan Dampak
   Banjir")
21.      # Mengambil data dari database
22.      data = load_data_from_database()
23.      st.write("Dataset wilayah:")
24.      st.dataframe(data)  # Menampilkan seluruh data dalam
   bentuk tabel interaktif
25.      # Langkah 3: Data Understanding
26.      st.subheader("Info Dataset")
27.      st.write(data.info())
28.      st.subheader("Statistik Deskriptif")
29.      st.write(data.describe())
30.      # Langkah 4: Data Preparation
31.      features = data[['Jumlah_Kejadian',
   'Menderita_dan_Mengungsi', 'Rumah_Terendam']]
32.      data['Menderita_dan_Mengungsi_log'] =
   np.log1p(data['Menderita_dan_Mengungsi'])
33.      features_transformed = data[['Jumlah_Kejadian',
   'Menderita_dan_Mengungsi_log', 'Rumah_Terendam']]
34.      # Normalisasi data
35.      scaler = StandardScaler()
36.      scaled_features =
   scaler.fit_transform(features_transformed)
37.      # Langkah 5: Menentukan jumlah cluster optimal dengan
   metode Elbow
38.      st.subheader("Metode Elbow untuk Menentukan Jumlah
   Cluster")
```

```
39.       sse = []
40.       k_values = range(1, 11)
41.       for k in k_values:
          kmeans = KMeans(n_clusters=k, random_state=42)
          kmeans.fit(scaled_features)
          sse.append(kmeans.inertia_)
42.       fig, ax = plt.subplots(figsize=(6, 4))
43.       ax.plot(k_values, sse, marker='o')
44.       ax.set_title('Metode Elbow untuk Menentukan Jumlah
   Cluster')
45.       ax.set_xlabel('Jumlah Cluster')
46.       ax.set_ylabel('SSE')
47.       ax.grid(True)
48.       st.pyplot(fig)
49.       st.write("Metode Elbow digunakan untuk menentukan
   jumlah cluster optimal dengan melihat titik siku pada grafik
   SSE (Sum of Squared Errors). Jumlah cluster optimal adalah
   saat terjadi penurunan signifikan sebelum stabil. Dari
   grafik berikut, kita memilih jumlah cluster optimal sebanyak
   3 karena setelah titik ini, penurunan SSE mulai melambat.")
50.       # Berdasarkan plot elbow, pilih jumlah cluster optimal
   (misalnya 3)
51.       optimal_k = 3
52.       # Langkah 6: Modeling - K-Means Clustering
53.       kmeans = KMeans(n_clusters=optimal_k, random_state=42)
54.       kmeans.fit(scaled_features)
55.       data['Cluster'] = kmeans.labels_
56.       # Ubah indeks cluster dari 0,1,2 menjadi 1,2,3
57.       data['Cluster'] = data['Cluster'] + 1
58.       # Memberikan nama pada cluster
59.       cluster_names = {
          1: 'Wilayah dengan dampak korban dan kerusakan
          ringan',
          2: 'Wilayah dengan dampak korban dan kerusakan
          sedang',
          3: 'Wilayah dengan dampak korban dan kerusakan berat'
60.       }
61.       data['Cluster_Name'] =
   data['Cluster'].map(cluster_names)
62.       st.subheader("Hasil Clustering")
63.       st.dataframe(data[['Provinsi','KabKota','Jumlah_Kejadi
   an', 'Menderita_dan_Mengungsi', 'Rumah_Terendam', 'Cluster',
   'Cluster_Name']])
64.       output_file = 'hasil_clustering_banjir.xlsx'
65.       # Menyimpan ke dalam BytesIO object dengan openpyxl
66.       towrite = BytesIO()
67.       with pd.ExcelWriter(towrite, engine='openpyxl') as
   writer:
          data.to_excel(writer, index=False,
          sheet_name='Cluster')
68.       towrite.seek(0)  # Kembali ke awal stream
```

```
69.       st.write(f"Hasil clustering disimpan dalam file:
   {output_file}")
70.       # Tombol download
71.       st.download_button("Download hasil clustering",
   towrite, file_name=output_file, mime="application/vnd.ms-
   excel")
72.       # Langkah 7: Evaluasi - Perhitungan Silhouette Score
   untuk 10 percobaan
73.       st.subheader("Evaluasi Model - Silhouette Score")
74.       sil_scores = []  # List untuk menyimpan hasil
   Silhouette Score dari setiap percobaan
75.       for i in range(10):
          kmeans_test = KMeans(n_clusters=optimal_k,
          random_state=i)
          kmeans_test.fit(scaled_features)
          score = silhouette_score(scaled_features,
          kmeans_test.labels_)
          sil_scores.append(score)
          st.write(f"Percobaan {i+1} - Silhouette Score:
          {score:.4f}")  # Menampilkan Silhouette Score setiap
          percobaan
76.       # Menghitung rata-rata Silhouette Score
77.       average_sil_score = np.mean(sil_scores)
78.       st.write(f"Rata-rata Silhouette Score dari 10
   percobaan: {average_sil_score:.4f}")
79.       # Langkah 8: Visualisasi hasil clustering
80.       # # Scatter plot untuk visualisasi cluster (2D)
81.       # st.subheader("Visualisasi Cluster (2D)")
82.       # fig, ax = plt.subplots(figsize=(8, 6))
83.       # scatter = ax.scatter(scaled_features[:, 2],
   scaled_features[:, 1],
84.       #                      c=data['Cluster'],
   cmap='viridis', label='Cluster')
85.       # centroids = kmeans.cluster_centers_
86.       # ax.scatter(centroids[:, 2], centroids[:, 1], s=200,
   c='red', marker='X', label='Centroid')
87.       # ax.set_xlabel('Rumah Terendam (scaled)')
88.       # ax.set_ylabel('Menderita dan Mengungsi (scaled)
   log')
89.       # ax.set_title('Clustering Wilayah Berdasarkan Dampak
   Banjir (2D)')
90.       # fig.colorbar(scatter, label='Cluster')
91.       # ax.legend()
92.       # st.pyplot(fig)
93.       # Scatter plot untuk visualisasi cluster (3D)
94.       # Tab untuk visualisasi
95.       tab1, tab2, tab3 = st.tabs(["Visualisasi 3D",
   "Barchart Per Wilayah", "Pie Chart Per Cluster"])
96.       # Tab 1: Visualisasi 3D
97.       with tab1:
          st.subheader("Visualisasi Cluster (3D)")
          st.write("""
```

```
        Di tab ini, kita memvisualisasikan hasil
        clustering dalam bentuk grafik tiga dimensi
        (3D),
        yang memungkinkan kita untuk melihat bagaimana
        wilayah-wilayah dikelompokkan berdasarkan tiga
        faktor utama:
        Jumlah Kejadian, Menderita dan Mengungsi, serta
        Rumah Terendam.
        Setiap titik di dalam grafik ini mewakili satu
        wilayah, dengan warna yang menunjukkan cluster
        yang berbeda.
        Visualisasi ini membantu kita memahami sebaran
        geografis wilayah-wilayah yang mengalami dampak
        banjir dengan intensitas yang berbeda-beda.
    """)
    st.write("""
       i.  **Keterangan warna:**
             -  🟣 **Ungu** → Cluster 1 (**Dampak
                Ringan**)
             -  🟢 **Hijau Kebiruan** → Cluster 2
                (**Dampak Sedang**)
             -  🟡 **Kuning** → Cluster 3 (**Dampak
                Berat**)
             -  ❌ **Merah X** → **Centroid (Titik Pusat
                Cluster)**
      ii.  """)

fig1 = plt.figure(figsize=(6, 4))

    ax1 = fig1.add_subplot(111, projection='3d')


    scatter3d = ax1.scatter(scaled_features[:, 0],
    scaled_features[:, 1], scaled_features[:, 2],
                        i.  c=data['Cluster'],
                            cmap='viridis',
                            label='Cluster')
    centroids = kmeans.cluster_centers_
    ax1.scatter(centroids[:, 0], centroids[:, 1],
    centroids[:, 2], s=200, c='red', marker='X',
    label='Centroid')
    ax1.set_xlabel('Jumlah Kejadian', fontsize=8)
    ax1.set_ylabel('Menderita dan Mengungsi', fontsize=8)
    ax1.set_zlabel('Rumah Terendam', fontsize=8)
    ax1.set_title('Clustering Wilayah Berdasarkan Dampak
    Banjir (3D)')
    ax1.legend()
    st.pyplot(fig1)
    # Definisikan data per cluster
    cluster1_data = data[data['Cluster'] == 1]
    cluster2_data = data[data['Cluster'] == 2]
    cluster3_data = data[data['Cluster'] == 3]
```

```
98.     # Tab 2: Barchart per wilayah berdasarkan cluster
99.     with tab2:
        st.subheader("Barchart Dampak Wilayah Berdasarkan
        Cluster")
        st.write("""
              Pada tab ini, kita menunjukkan bar chart untuk
              masing-masing cluster yang menggambarkan jumlah
              kejadian banjir di setiap provinsi.
              Setiap cluster merepresentasikan wilayah dengan
              tingkat dampak yang berbeda, mulai dari ringan
              hingga berat.
              Dengan melihat bar chart ini, pengguna dapat
              membandingkan secara langsung jumlah kejadian
              banjir di wilayah yang termasuk dalam setiap
              kategori dampak banjir,
              serta melihat perbedaan kejadian antarprovinsi
              dalam setiap cluster.
        """)
        fig2, (ax2_1, ax2_2, ax2_3) = plt.subplots(1, 3,
        figsize=(18, 6))
        # Cluster 1
        ax2_1.bar(cluster1_data['Provinsi'],
        cluster1_data['Jumlah_Kejadian'], color='lightblue')
        ax2_1.set_title('Cluster 1: Wilayah Dengan Dampak
        Ringan')
        ax2_1.set_xlabel('Provinsi')
        ax2_1.set_ylabel('Jumlah Kejadian')
        ax2_1.tick_params(axis='x', rotation=90)
        # Cluster 2
        ax2_2.bar(cluster2_data['Provinsi'],
        cluster2_data['Jumlah_Kejadian'], color='lightgreen')
        ax2_2.set_title('Cluster 2: Wilayah Dengan Dampak
        Sedang')
        ax2_2.set_xlabel('Provinsi')
        ax2_2.set_ylabel('Jumlah Kejadian')
        ax2_2.tick_params(axis='x', rotation=90)
        # Cluster 3
        ax2_3.bar(cluster3_data['Provinsi'],
        cluster3_data['Jumlah_Kejadian'], color='lightcoral')
        ax2_3.set_title('Cluster 3: Wilayah Dengan Dampak
        Berat')
        ax2_3.set_xlabel('Provinsi')
        ax2_3.set_ylabel('Jumlah Kejadian')
        ax2_3.tick_params(axis='x', rotation=90)
        st.pyplot(fig2)
100.    # Tab 3: Pie chart per cluster
101.    with tab3:
        st.subheader("Pie Chart Per Cluster")
        st.write("""
          i.  Tab ini menunjukkan pie chart yang menggambarkan
              persentase distribusi wilayah berdasarkan
              cluster dampak banjir yang telah terbentuk.
```

```
                Setiap bagian dari pie chart mewakili proporsi
                wilayah yang termasuk dalam kategori dampak
                banjir ringan, sedang, atau berat.
                Visualisasi ini memberikan gambaran umum tentang
                bagaimana wilayah-wilayah tersebut terbagi dalam
                tiga cluster,
                sehingga kita bisa dengan cepat melihat sebaran
                dampak banjir di seluruh dataset.
            """)
            cluster_sizes = data['Cluster_Name'].value_counts()
            fig3, ax3 = plt.subplots(figsize=(7, 7))
            ax3.pie(cluster_sizes, labels=cluster_sizes.index,
            autopct='%1.1f%%', startangle=90,
            colors=plt.cm.Paired.colors)
            ax3.set_title('Distribusi Wilayah Berdasarkan Dampak
            Banjir', fontsize=14)
            st.pyplot(fig3)
102.        # # Langkah 9: Simpan hasil ke dalam BytesIO
103.        # output_file = 'hasil_clustering_banjir.xlsx'
104.        # # Menyimpan ke dalam BytesIO object dengan openpyxl
105.        # towrite = BytesIO()
106.        # with pd.ExcelWriter(towrite, engine='openpyxl') as
    writer:
107.        #     data.to_excel(writer, index=False,
    sheet_name='Cluster')
108.        # towrite.seek(0)  # Kembali ke awal stream
109.        # st.write(f"Hasil clustering disimpan dalam file:
    {output_file}")
110.        # # Tombol download
111.        # st.download_button("Download hasil clustering",
    towrite, file_name=output_file, mime="application/vnd.ms-
    excel")
112.        if __name__ == "__main__":
113.        show_cluster()
```