

به نام خدا

مبانی یادگیری ماشین

دکتر ابوالقاسمی - دکتر اعرابی

پروژه نهایی - فاز 2

سامان دوچی طوسی - 810101420

رضا چهرقانی - 810100401

علی دهقانزاده - 810101423

پیش پردازش داده و استخراج ویژگی:

۱. بارگذاری داده ها:

در این پروژه، مجموعه‌ای از فایل‌های صوتی شامل گفتار افراد مختلف مورد استفاده قرار گرفت. این داده‌ها در گوگل درایو ذخیره شده و با استفاده از Google Colab به محیط پردازشی وارد شده‌اند. برای این کار، ابتدا درایو به محیط Google Colab متصل شد:

```
from google.colab import drive  
drive.mount('/content/drive')
```

سپس، فایل‌های صوتی از مسیر مشخصی خوانده شدند:

```
import os  
  
folder_path      =      '/content/drive/MyDrive/Machine Learning Project/Audios'  
files = os.listdir(folder_path)  
files_path = [os.path.join(folder_path, file) for file in files]
```

استخراج اطلاعات مرتبط از نام فایل‌ها

نام فایل‌های صوتی شامل اطلاعاتی در مورد گوینده است، از جمله جنسیت و شناسه دانشجویی. برای استفاده از این اطلاعات در پردازش‌های بعدی، این ویژگی‌ها از نام فایل استخراج شدند.

استخراج جنسیت گوینده

نام فایل‌ها شامل کلمه "male" برای گویندگان مرد و "female" برای گویندگان زن است. تابع زیر با بررسی نام فایل‌ها جنسیت هر گوینده را استخراج می‌کند:

```
def extract_gender(file_name):  
    file_name_lower = file_name.lower()  
    if 'female' in file_name_lower:  
        return 'female'  
    elif 'male' in file_name_lower:  
        return 'male'  
    else:  
        return None  
  
genders = [extract_gender(file) for file in files]
```

استخراج شناسه دانشجویی

در فایل‌های صوتی، یک عدد ۹ رقمی به عنوان شناسه دانشجویی گوینده در نام فایل ذخیره شده است. با استفاده از عبارات منظم (Regular Expressions) این شناسه استخراج می‌شود:

```

import re

def extract_student_id(file_name, pattern=r"\d{9}"):
    match = re.search(pattern, file_name)
    if match:
        return match.group()
    else:
        return None

student_ids = [extract_student_id(file) for file in files]

```

۲. پیش پردازش داده ها صوتی

برای تحلیل داده های صوتی و استخراج ویژگی های مفید، ابتدا نیاز به پردازش اولیه سیگنال صوتی داریم. این مرحله شامل کاهش نویز، فیلتر گذاری، نرم افزاری، و تقسیم سیگنال به بخش های کوچکتر است.

بارگذاری فایل صوتی

ابتدا فایل های صوتی بارگذاری می شوند. برای اطمینان از یکسان بودن نرخ نمونه برداری (sampling rate) در تمام فایل ها، سیگنال ها به نرخ ۱۶ کیلوهرتز (kHz 16) بازنمونه برداری (resampling) می شوند:

```

import librosa

def load_audio(file_path, target_sr=16000):
    audio, sr = librosa.load(file_path, sr=target_sr, mono=True)
    return audio, sr

```

کاهش نویز با استفاده از کاهش طیفی (Spectral Subtraction)

نویز پس زمینه ممکن است کیفیت داده ها را کاهش دهد. برای حذف نویز، ابتدا طیف توان سیگنال محاسبه شده و سپس مقدار نویز که به صورت میانه ای از طیف توان تخمین زده شده است، از سیگنال کم می شود:

```

import scipy.signal as signal

def reduce_noise(audio, sr):
    freqs, times, Sxx = signal.spectrogram(audio, sr)
    noise_threshold = np.median(Sxx, axis=1)[:, np.newaxis]
    Sxx_denoised = np.maximum(Sxx - noise_threshold, 0)
    return librosa.istft(Sxx_denoised)

```

اعمال فیلتر میان گذر (Bandpass Filtering)

سیگنال گفتار معمولاً در بازه فرکانسی ۳۰۰ هرتز تا ۳۴۰۰ هرتز قرار دارد. بنابراین، برای حذف نویزهای فرکانس های پایین و بالا، از فیلتر باترورث (Butterworth Filter) استفاده می شود:

```

def bandpass_filter(audio, sr, lowcut=300, highcut=3400, order=6):

```

```

nyquist = 0.5 * sr
low = lowcut / nyquist
high = highcut / nyquist
b, a = signal.butter(order, [low, high], btype='band')
return signal.filtfilt(b, a, audio)

```

تقسیم سیگنال به پنجره‌های همپوشان

برای تحلیل بهتر سیگنال، آن را به بخش‌های کوچکتر (پنجره‌هایی با اندازه ۲۰۴۸ نمونه و همپوشانی ۷۵٪) تقسیم می‌کنیم. این کار باعث می‌شود ویژگی‌های استخراج شده دقیق‌تر باشند:

```

def frame_audio(audio, frame_size=2048, hop_size=512):
    return librosa.util.frame(audio, frame_length=frame_size,
hop_length=hop_size).T

```

نرمال‌سازی سیگنال صوتی

برای جلوگیری از تفاوت‌های ناشی از شدت صدای گویندگان، مقدار سیگنال به حداقل مقدار خود تقسیم شده و در بازه $[1-1]$ مقیاس‌بندی می‌شود:

```

def normalize_audio(audio):
    return audio / np.max(np.abs(audio))

```

پردازش کلی داده‌های صوتی

تمام مراحل فوق در یکتابع قرار می‌گیرند تا برای هر فایل صوتی به ترتیب اجرا شوند:

```

def preprocess_audio(file_path):
    audio, sr = load_audio(file_path)
    audio = reduce_noise(audio, sr)
    audio = bandpass_filter(audio, sr)
    frames = frame_audio(audio)
    frames = np.array([normalize_audio(frame) for frame in frames])
    return frames, sr

```

۳. استخراج ویژگی‌های صوتی

برای تجزیه و تحلیل سیگنال‌های صوتی، از ویژگی‌های مختلفی استفاده می‌شود که اطلاعات مفیدی درباره ماهیت صوت (مانند زیر و بمی، شدت، و الگوهای طیفی) ارائه می‌دهند. در این بخش، ویژگی‌های صوتی که در کد زیر استفاده شده‌اند، توضیح داده می‌شوند.

```

from librosa.feature import (
    mfcc,
    spectral_centroid,
    spectral_bandwidth,
    spectral_contrast,
    melspectrogram,

```

```

        zero_crossing_rate,
        chroma_stft
    )

def extract_features(frame, sr):
    features = {
        "MFCC": mfcc(y=frame, sr=sr, n_mfcc=8),
        "Spectral Centroid": spectral_centroid(y=frame, sr=sr),
        "Spectral Bandwidth": spectral_bandwidth(y=frame, sr=sr),
        "Spectral Contrast": spectral_contrast(y=frame, sr=sr),
        "Log Mel Spectrogram": librosa.power_to_db(melspectrogram(y=frame,
sr=sr, n_mels=10)),
        "Zero Crossing Rate": zero_crossing_rate(y=frame),
        "Chroma Features": chroma_stft(y=frame, sr=sr, n_chroma=6),
        "Energy": np.sum(frame ** 2) / len(frame)
    }
    return features

```

ضرایب فرکانس مل (MFCC - Mel Frequency Cepstral Coefficients)

MFCC یکی از پرکاربردترین ویژگی‌ها در پردازش صوت است که برای تشخیص گفتار، شناسایی گوینده و دسته‌بندی سیگنال‌های صوتی استفاده می‌شود.

ابتدا طیف توان سیگنال صوتی محاسبه شده و سپس روی یک مقیاس فرکانسی به نام مل (Mel Scale) نگاشت می‌شود. در ادامه، تبدیل MFCC (Discrete Cosine Transform) روی طیف مل اعمال شده و ضرایب استخراج می‌شوند. این ضرایب، اطلاعات مربوط به طیف فرکانسی را در قالب یک بردار فشرده نمایش می‌دهند. در کد، ۸ ضریب MFCC استخراج شده‌اند:

```
"MFCC": mfcc(y=frame, sr=sr, n_mfcc=8)
```

مرکز ثقل طیفی (Spectral Centroid)

مرکز ثقل طیفی نشان‌دهنده میانگین وزنی فرکانس‌های موجود در سیگنال است و بیانگر "درک انسان از زیر و بمی صدا" است. اگر فرکانس‌های بالاتر انرژی بیشتری داشته باشند، مقدار مرکز ثقل طیفی افزایش می‌یابد و برعکس. فرمول محاسبه به صورت میانگین وزنی فرکانس‌ها نسبت به دامنه توان طیف است:

$$SC = \frac{\sum_{i=1}^N f_i |X(f_i)|}{\sum_{i=1}^N |X(f_i)|}$$

```
"Spectral Centroid": spectral_centroid(y=frame, sr=sr)
```

پهناهی باند طیفی (Spectral Bandwidth)

پهنانی باند طیفی میزان پراکندگی انرژی فرکانسی را نسبت به مرکز ثقل طیفی اندازه‌گیری می‌کند. نشان‌دهنده تنوع طیفی یک سیگنال است و مقدار زیاد آن بیانگر نویز یا تنوع بالا در فرکانس‌ها است. مقدار آن با استفاده از انحراف معیار نسبت به مرکز ثقل طیفی محاسبه می‌شود:

$$SB = \sqrt{\sum_{i=1}^N (f_i - SC)^2 \cdot |X(f_i)|}$$

```
"Spectral Bandwidth": spectral_bandwidth(y=frame, sr=sr)
```

کنتراست طیفی (Spectral Contrast)

این ویژگی تفاوت بین نواحی با انرژی بالا و پایین در هر باند فرکانسی را اندازه‌گیری می‌کند. مقادیر بالای کنتراست طیفی نشان‌دهنده تمایز شدید بین فرکانس‌های قوی و ضعیف در طیف است. معمولاً در تشخیص موسیقی و گفتار استفاده می‌شود.

```
"Spectral Contrast": spectral_contrast(y=frame, sr=sr)
```

اسپکتروگرام مل (Log Mel Spectrogram)

اسپکتروگرام مل نمایش طیفی سیگنال در مقیاس فرکانسی مل است. ابتدا Mel Spectrogram با استفاده از فیلتر بانک‌های مل محاسبه شده و سپس باتابع لگاریتمی مقدار آن به واحد dB تبدیل می‌شود. این ویژگی نسخه‌ای پیشرفته از MFCC است که برای استخراج ویژگی‌های گفتاری استفاده می‌شود.

```
"Log Mel Spectrogram": librosa.power_to_db(melspectrogram(y=frame, sr=sr, n_mels=10))
```

نرخ عبور از صفر (Zero Crossing Rate - ZCR)

این ویژگی تعداد دفعاتی را که سیگنال از مقدار صفر عبور می‌کند، اندازه‌گیری می‌کند. سیگنال‌های گفتاری یا موسیقی با فرکانس‌های بالا مقدار ZCR بیشتری دارند.

$$ZCR = \frac{1}{N-1} \sum_{n=1}^{N-1} 1_{[x[n].x[n-1] < 0]}$$

```
"Zero Crossing Rate": zero_crossing_rate(y=frame)
```

ویژگی‌های رنگ‌آمیزی طیفی (Chroma Features)

کروماتوگرام (Chroma Features) میزان انرژی سیگنال در ۱۲ کلاس فرکانسی موسیقی را بررسی می‌کند. این ویژگی در پردازش موسیقی و تشخیص گام‌های موسیقی (Pitch) بسیار مهم است.

```
"Chroma Features": chroma_stft(y=frame, sr=sr, n_chroma=6)
```

انرژی سیگنال صوتی (Energy)

انرژی سیگنال مقدار توان سیگنال در طول زمان است و با مجموع مربعات دامنه سیگنال در هر فریم محاسبه می‌شود:

$$Energy = \frac{1}{N} \sum_{n=1}^N x[n]^2$$

مقدار بالاتر نشان‌دهنده شدت بیشتر صدا است.

```
"Energy": np.sum(frame ** 2) / len(frame)
```

اعمال پردازش داده‌های صوتی و استخراج ویژگی‌ها

این کد مسئول پردازش فایل‌های صوتی، استخراج ویژگی‌های آن‌ها و ذخیره داده‌های پردازش شده برای دو هدف احراز هویت کاربران و تشخیص جنسیت است.

در تابع process_audio_files فایل‌های صوتی انتخاب شده با preprocess_audio پردازش شده و به فریم‌های کوچکتر تقسیم می‌شود. سپس برای هر فریم، ویژگی‌های صوتی استخراج شده و در لیست all_features ذخیره می‌شوند. برچسب متناظر با هر فایل نیز در all_labels ذخیره می‌شود. در نهایت تمام ویژگی‌های هر فریم به یک بردار ویژگی تجمعی می‌شوند و ویژگی‌ها و برچسب‌ها در یک فایل ذخیره می‌شوند.

```
import pandas as pd

def process_audio_files(indices, labels, name):
    all_features, all_labels = [], []

    for i in tqdm(indices):
        frames, sr = preprocess_audio(files_path[i])
        if np.any(np.isnan(frames)):
            continue

        for frame in frames:
            features = extract_features(frame, sr)
            all_features.append(features)
            all_labels.append(labels[i])

    df = pd.DataFrame(all_features)

    features = df.drop(columns='Energy').map(lambda x: x.flatten())
    features['Energy'] = df['Energy']
    features = np.vstack(features.apply(np.hstack, axis=1).to_numpy())

    np.save(f'/content/drive/MyDrive/Machine Learning Project/features{name}.npy', features)
```

```

        np.savez_compressed(f'/content/drive/MyDrive/Machine Learning
Project/dataset {name}.npz',
        features=features,
        labels=np.array(all_labels)
)

```

ایجاد مجموعه داده برای احراز هویت

در هر تکرار، ۶ فایل صوتی از دانشجویان مختلف برای تست احراز هویت انتخاب می‌شوند. این انتخاب به صورت تصادفی انجام می‌شود. ویژگی‌های فایل‌های انتخاب شده استخراج و در مجموعه داده authentication {i + 1} ذخیره می‌شوند.

```

import random

for i in range(3):
    selected_indices = set()
    while len(selected_indices) < 6:
        index = random.randint(0, len(student_ids) - 1)
        if student_ids[index] != None:
            selected_indices.add(index)

    process_audio_files(selected_indices, student_ids,
f'authentication {i + 1}')

```

ایجاد مجموعه داده برای تشخیص جنسیت

به ازای هر جنسیت از یک هشتم داده‌ها استفاده می‌کنیم که در کل برای با ۲۵٪ کل داده‌ها می‌شود. از لیست فایل‌ها، به اندازه n فایل از هر جنسیت انتخاب می‌شود. ویژگی‌های این فایل‌ها استخراج شده و در مجموعه داده gender ذخیره می‌شوند.

```

n = len(files) // 8
male_count, female_count = 0, 0
selected_indices = []

for i in range(len(genders)):
    if genders[i] == 'male' and male_count < n:
        selected_indices.append(i)
        male_count += 1
    elif genders[i] == 'female' and female_count < n:
        selected_indices.append(i)
        female_count += 1

process_audio_files(selected_indices, genders, 'gender')

```

۴. بصری‌سازی داده‌های صوتی:

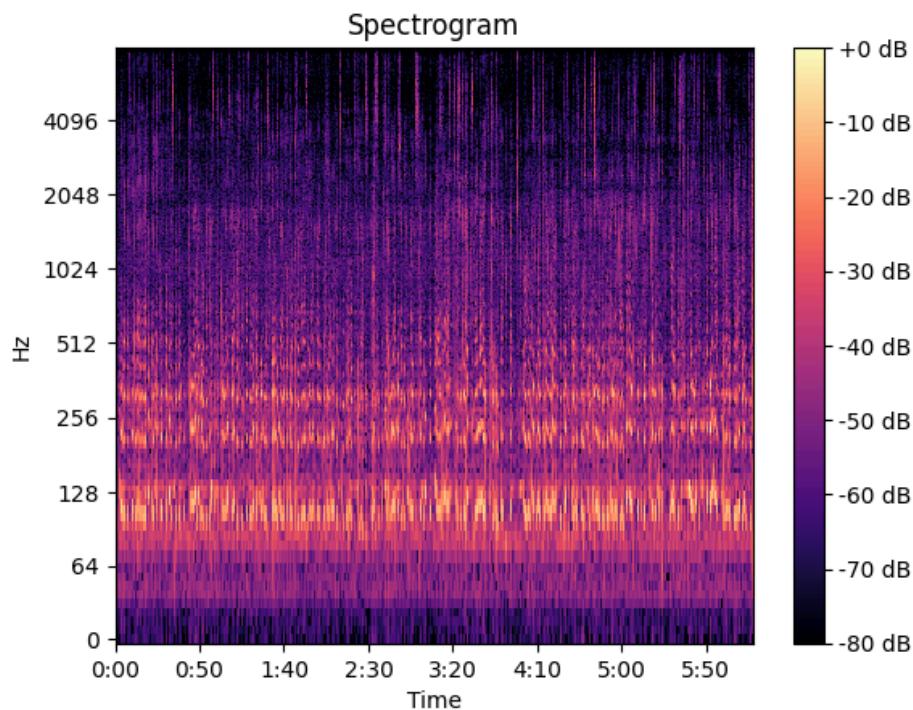
به بررسی و تحلیل داده‌های صوتی با استفاده از دو روش بصری‌سازی Mel-Spectrogram و Spectrogram می‌پردازیم. این روش‌ها به ما کمک می‌کنند تا ویژگی‌های فرکانسی و زمانی سیگنال‌های صوتی را بهتر درک کنیم.

نمایش اسپکتروگرام (Spectrogram) :

تبدیل فوریه کوتاه‌مدت (STFT) را روی سیگنال صوتی اعمال می‌کند؛ دامنه را به مقیاس دسی‌بل تبدیل می‌کند و نمایش بصری از طیف فرکانسی بر حسب زمان ارائه می‌دهد.

```
def plot_spectrogram(audio, sr):
    D = librosa.amplitude_to_db(np.abs(librosa.stft(audio)), ref=np.max)
    librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar(format='%.2f dB')
    plt.title('Spectrogram')
    plt.show()
```

این نمودار تغییرات دامنه سیگنال را در فرکانس‌های مختلف و در طول زمان نشان می‌دهد. محور عمودی نشان‌دهنده‌ی فرکانس (به صورت لگاریتمی) و محور افقی نشان‌دهنده‌ی زمان است. رنگ‌ها نیز نشان‌دهنده‌ی شدت دامنه (بر حسب دسی‌بل) هستند.



نمایش مل اسپکتروگرام (Mel-Spectrogram) :

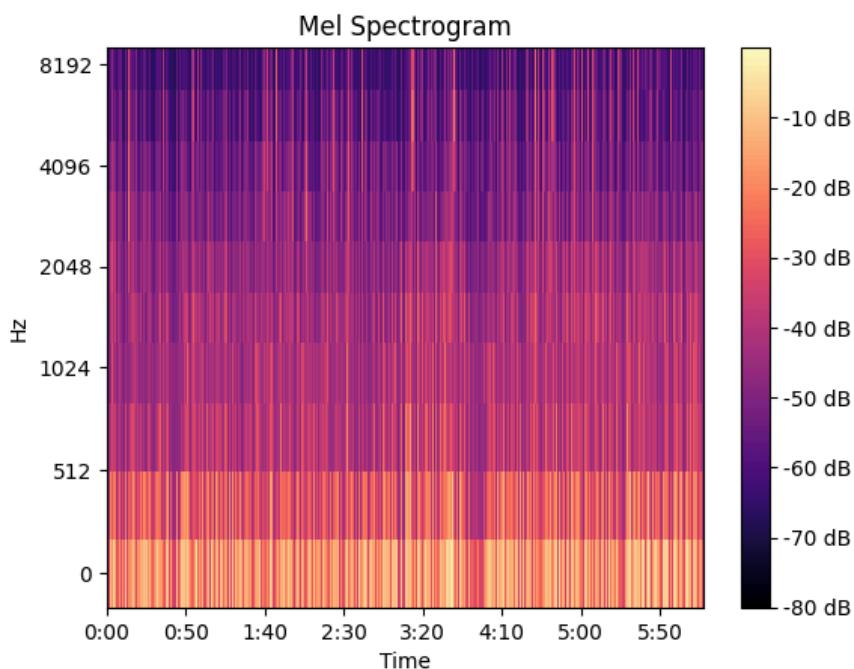
اسپکتروگرام را در مقیاس فرکانسی Mel محاسبه می‌کند؛ مقدار توان را به مقیاس دسی‌بل تبدیل می‌کند و نمایش بصری از طیف Mel بر حسب زمان ارائه می‌دهد.

```

def plot_mel_spectrogram(audio, sr):
    S = librosa.feature.melspectrogram(y=audio, sr=sr, n_mels=10)
    S_DB = librosa.power_to_db(S, ref=np.max)
    librosa.display.specshow(S_DB, sr=sr, x_axis='time', y_axis='mel')
    plt.colorbar(format='%.2f dB')
    plt.title('Mel Spectrogram')
    plt.show()

```

این نمودار مشابه Spectrogram است، اما فرکانس‌ها بر اساس مقیاس مل (Mel scale) نمایش داده می‌شوند. این مقیاس به نحوی طراحی شده است که بهتر با شنوایی انسان مطابقت دارد.



این کد داده‌های صوتی را دریافت کرده و دو نوع نمایش بصری (Mel-Spectrogram و Spectrogram) برای تحلیل ویژگی‌های صوتی ارائه می‌دهد.

```

audio, sr = load_audio(files_path[0])
plot_spectrogram(audio, sr)
plot_mel_spectrogram(audio, sr)

```

نمایش ویژگی‌های استخراج شده از داده‌های صوتی

تابع `visualize_features` بر اساس انتخاب روش، ویژگی‌ها را به دو بعد کاهش می‌دهد و سپس آنها را در نمودار پراکندگی نمایش می‌دهد.

```

from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

def visualize_features(features, labels, method):

```

```

if method == 'pca':
    reducer = PCA(n_components=2)
elif method == 'tsne':
    reducer = TSNE(n_components=2, perplexity=30, learning_rate=200)
else:
    raise ValueError("Method must be 'pca' or 'tsne'")

reduced_features = reducer.fit_transform(features)

unique_labels = np.unique(labels)
colors = plt.get_cmap('tab10', len(unique_labels))

for i, label in enumerate(unique_labels):
    mask = labels == label
    plt.scatter(reduced_features[mask, 0], reduced_features[mask, 1],
                label=label, alpha=0.6, color=colors(i))

plt.title(f'Feature Visualization using {method.upper()}')
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.legend()
plt.show()

```

۶ فایل صوتی به صورت تصادفی انتخاب می‌شوند. برای هر فایل صوتی، ویژگی‌های آن با استفاده از پیش‌پردازش و استخراج ویژگی‌های صوتی از آن محاسبه می‌شود. سپس ویژگی‌ها برای هر فریم صوتی استخراج و در یک آرایه جمع‌آوری می‌شوند.

```

all_features, all_labels = [], []
for _ in range(6):
    index = random.randint(0, len(student_ids) - 1)
    file_path = files_path[index]
    frames, sr = preprocess_audio(file_path)
    for frame in frames:
        features = extract_features(frame, sr)
        features = np.hstack([feature.flatten() for feature in features.values()])
        all_features.append(features)
        all_labels.append(student_ids[index])
all_features, all_labels = np.array(all_features), np.array(all_labels)

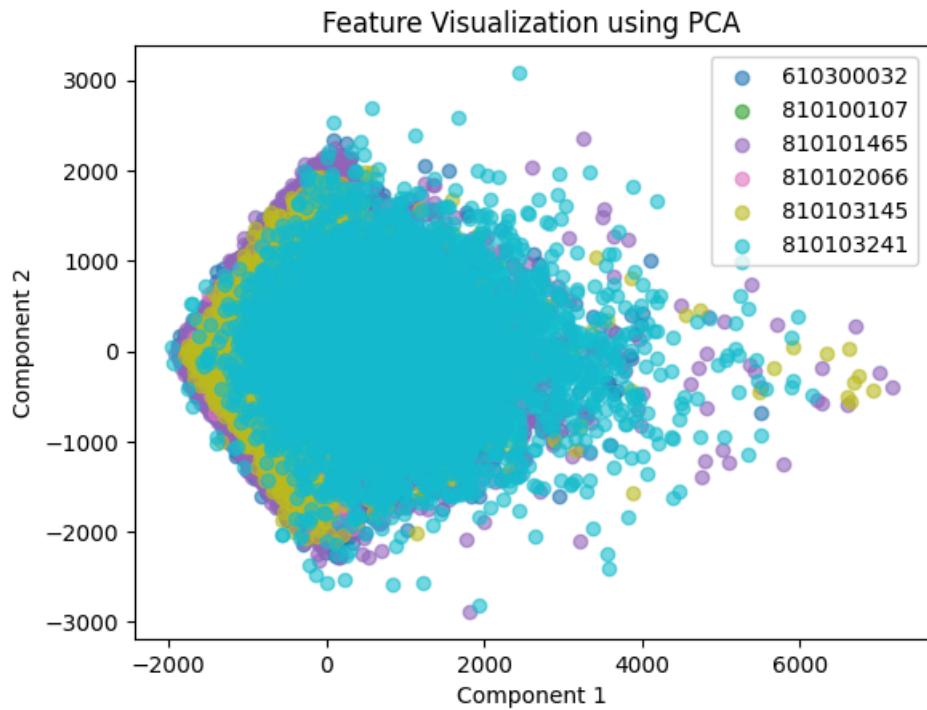
visualize_features(all_features, all_labels, 'pca')
visualize_features(all_features, all_labels, 'tsne')

```

با اجرای کد، دو نمودار زیر تولید می‌شوند:

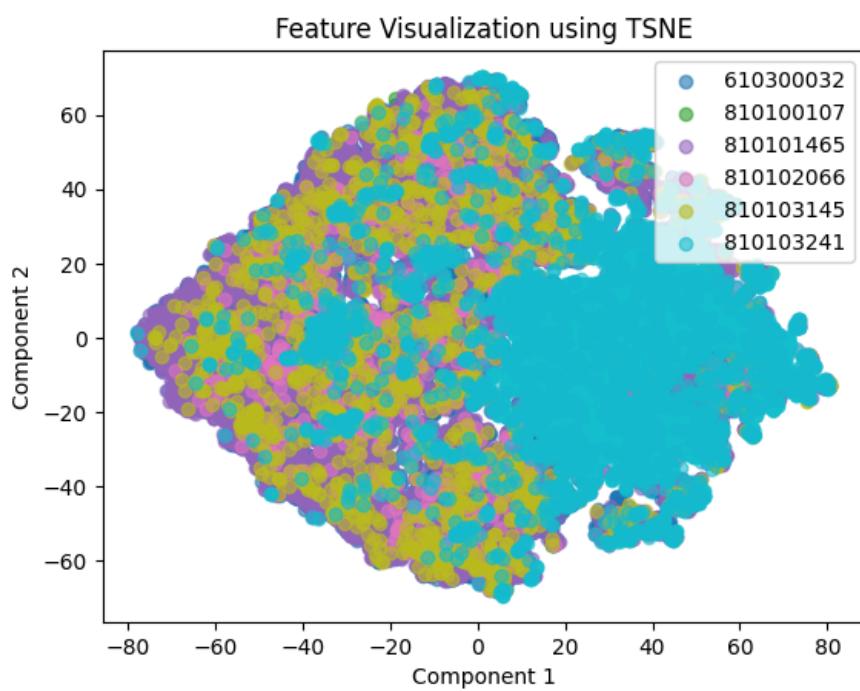
:PCA نمودار

داده‌ها در امتداد محورهای اصلی (Component 1 و Component 2) پراکنده شده‌اند. ممکن است همپوشانی بین برخی برچسبها وجود داشته باشد (نشان‌دهنده شباهت ویژگی‌ها). تغییرات کلی داده‌ها در جهت مولفه اول (Component 1) بیشتر است.



:t-SNE نمودار

داده‌ها در خوشه‌های مجزا گروه‌بندی شده‌اند (نسبت به PCA جداسازی بهتری دارد). فاصله بین خوشه‌ها نشان‌دهنده تفاوت در ویژگی‌های صوتی مربوط به هر برچسب است.



طبقه بندی:

تشخیص هویت:

کد:

```
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

from google.colab import drive
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc

drive.mount('/content/drive')

dataset_path = "/content/drive/MyDrive/Machine Learning Project/"

datasets = ["dataset authentication 1.npz", "dataset authentication 2.npz",
"dataset authentication 3.npz"]

models = {
    "SVM": SVC(kernel='rbf', probability=True, random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "MLP": MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=500,
random_state=42)
}

def load_dataset(file_path):
    try:
        data = np.load(file_path)
        print("Keys in file:", data.files)

        key_X = "features"
        key_y = "labels"

        if key_X not in data or key_y not in data:
            raise ValueError(f"Dataset {file_path} does not contain '{key_X}' and '{key_y}'.")
    except Exception as e:
        print(f"Error loading {file_path}: {e}")

    X, y = data[key_X], data[key_y]
    print(f"Loaded {file_path}: X shape = {X.shape}, y shape = {y.shape}")
    return X, y
```

```

        return None, None

def train_and_evaluate(X, y, dataset_name):
    if X is None or y is None:
        print(f"Skipping {dataset_name} due to loading error.")
        return

    # Normalize features
    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    # Split into 75% train and 25% test
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42, stratify=y)

    print(f"\n{dataset_name}: Training size = {X_train.shape[0]}, Test size =
{X_test.shape[0]}")

    joblib.dump(scaler, f"{dataset_name}_scaler.pkl")

    for model_name, model in models.items():
        print(f"\nTraining {model_name} on {dataset_name}...")

        try:
            model.fit(X_train, y_train)

            y_pred = model.predict(X_test)
            y_proba = model.predict_proba(X_test)[:, 1] if hasattr(model,
"predict_proba") else None

            print(f"\n--- {model_name} Classification Report ---")
            print(classification_report(y_test, y_pred))

            cm = confusion_matrix(y_test, y_pred)
            plt.figure(figsize=(5, 4))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
            plt.xlabel("Predicted")
            plt.ylabel("Actual")
            plt.title(f"Confusion Matrix - {model_name} ({dataset_name})")
            plt.show()

            if y_proba is not None:
                fpr, tpr, _ = roc_curve(y_test, y_proba)
                roc_auc = auc(fpr, tpr)
                plt.figure(figsize=(6, 5))
                plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc:.2f})')
                plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
                plt.xlabel('False Positive Rate')
                plt.ylabel('True Positive Rate')
                plt.title(f'ROC Curve - {model_name} ({dataset_name})')
                plt.legend()
                plt.show()

        joblib.dump(model, f"{dataset_name}_{model_name}.pkl")
    except Exception as e:
        print(f"Error training {model_name} on {dataset_name}: {e}")

```

```
for dataset in datasets:  
    file_path = os.path.join(dataset_path, dataset)  
    X, y = load_dataset(file_path)  
    train_and_evaluate(X, y, dataset.split('.')[0])
```

توضیحات کد:

۱. بارگذاری داده‌ها از Google Drive

ابتدا درایو گوگل به Colab متصل می‌شود تا فایل‌های `npz` که شامل ویژگی‌ها و برچسب‌ها هستند، خوانده شوند.

۲. بررسی محتويات فایل‌های `npz`.

چون مجموعه داده‌های شما شامل `labels.npy` و `features.npy` هستند، ابتدا بررسی می‌شود که آیا این کلیدها در فایل وجود دارند یا نه. اگر نبود، خطای داده و ادامه نمی‌دهد.

۳. نرمال‌سازی ویژگی‌ها

بعد از خواندن داده‌ها، ویژگی‌ها با `StandardScaler` نرمال‌سازی می‌شوند تا مدل‌ها بهتر یاد بگیرند.

۴. تقسیم داده‌ها به ۷۵٪ آموزش و ۲۵٪ تست

داده‌ها به دو بخش تقسیم می‌شوند:

- ۷۵٪ برای آموزش مدل
- ۲۵٪ برای تست مدل

تقسیم‌بندی به شکلی انجام می‌شود که تعادل برچسب‌ها حفظ شود (`stratify=y`).

۵. آموزش مدل‌ها

سه مدل یادگیری ماشین روی داده‌ها آموزش داده می‌شوند:
SVM (Support Vector Machine)
KNN (K-Nearest Neighbors)
(MLP (Multi-Layer Perceptron Neural Network
هر مدل ابتدا روی داده‌های آموزشی **fit()** شده و سپس روی داده‌های تست ارزیابی می‌شود.

۶. ارزیابی مدل‌ها

هر مدل با معیارهای زیر ارزیابی می‌شود:
→ نمایش تعداد پیش‌بینی‌های درست و غلط در قالب جدول **Confusion Matrix**
→ شامل دقت (Precision)، بازخوانی (Recall) و F1-score → **Classification Report**
→ نمایش عملکرد مدل در تشخیص کلاس‌های مختلف → **ROC Curve & AUC Score**

۷. ذخیره مدل‌های آموزش‌دیده و مقیاس‌ساز (Scaler)

- مدل‌های آموزش‌دیده در **Google Drive** ذخیره می‌شوند (**joblib.dump**)
 - مقیاس‌ساز (**StandardScaler**) هم ذخیره می‌شود تا بعداً روی داده‌های جدید اعمال شود.
-

۸. نمایش خروجی‌ها

✓ ماتریس درهم‌ریختگی (Confusion Matrix) به صورت نمودار رنگی نمایش داده می‌شود.
✓ نمودار **ROC Curve** برای بررسی عملکرد کلی مدل‌ها رسم می‌شود

نتایج:

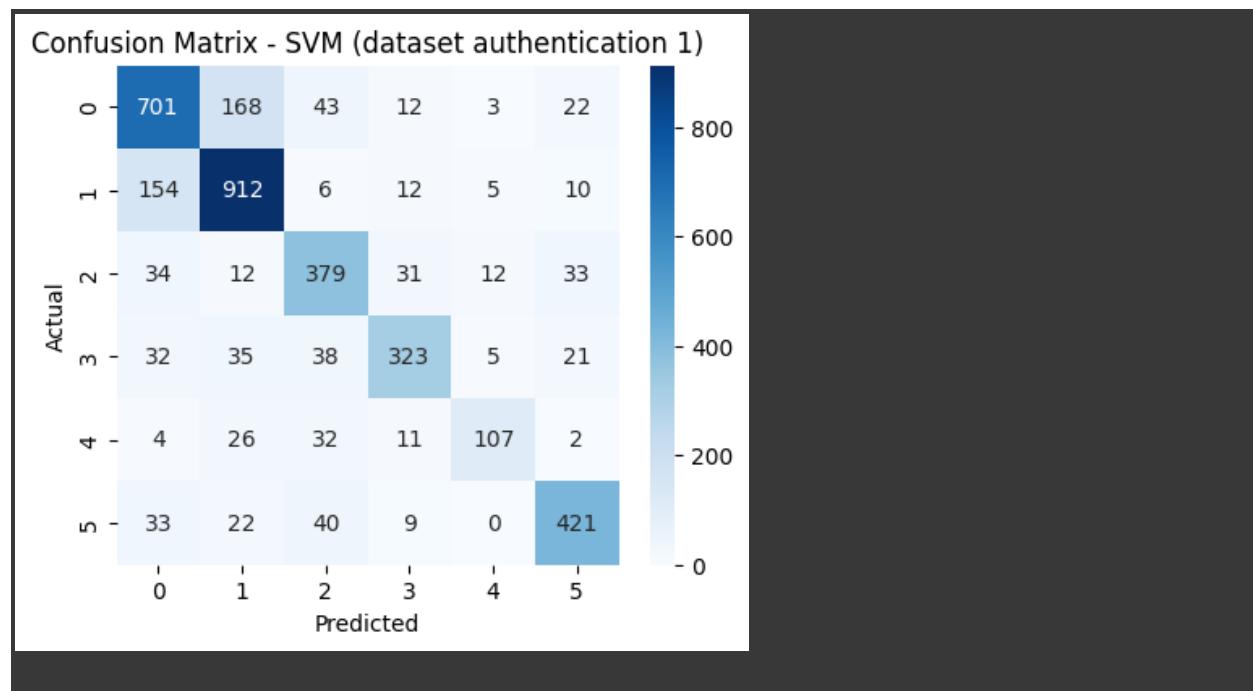
Keys in file: ['features', 'labels']

Loaded /content/drive/MyDrive/Machine Learning Project/dataset authentication 1.npz: X shape = (14838, 171), y shape = (14838,)

dataset authentication 1: Training size = 11128, Test size = 3710

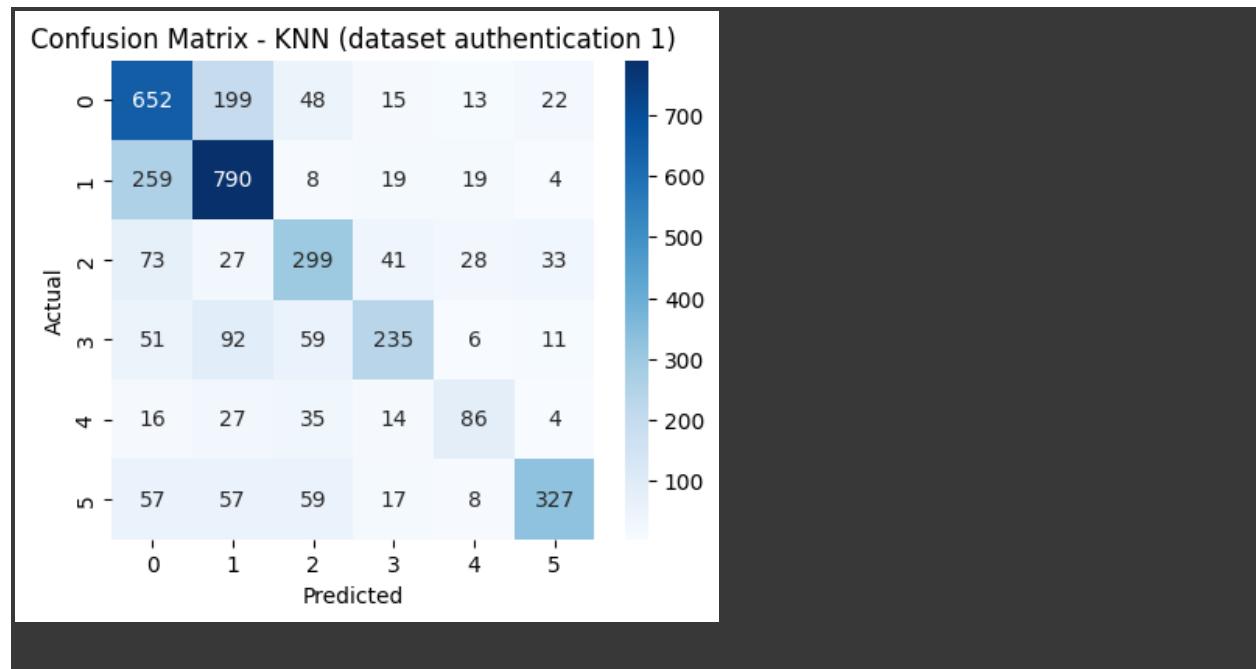
--- SVM Classification Report ---

	precision	recall	f1-score	support
610399182	0.73	0.74	0.74	949
810101575	0.78	0.83	0.80	1099
810102187	0.70	0.76	0.73	501
810103197	0.81	0.71	0.76	454
810103224	0.81	0.59	0.68	182
810199569	0.83	0.80	0.81	525
accuracy		0.77		3710
macro avg	0.78	0.74	0.75	3710
weighted avg	0.77	0.77	0.77	3710



--- KNN Classification Report ---

	precision	recall	f1-score	support
610399182	0.59	0.69	0.63	949
810101575	0.66	0.72	0.69	1099
810102187	0.59	0.60	0.59	501
810103197	0.69	0.52	0.59	454
810103224	0.54	0.47	0.50	182
810199569	0.82	0.62	0.71	525
accuracy		0.64		3710
macro avg	0.65	0.60	0.62	3710
weighted avg	0.65	0.64	0.64	3710

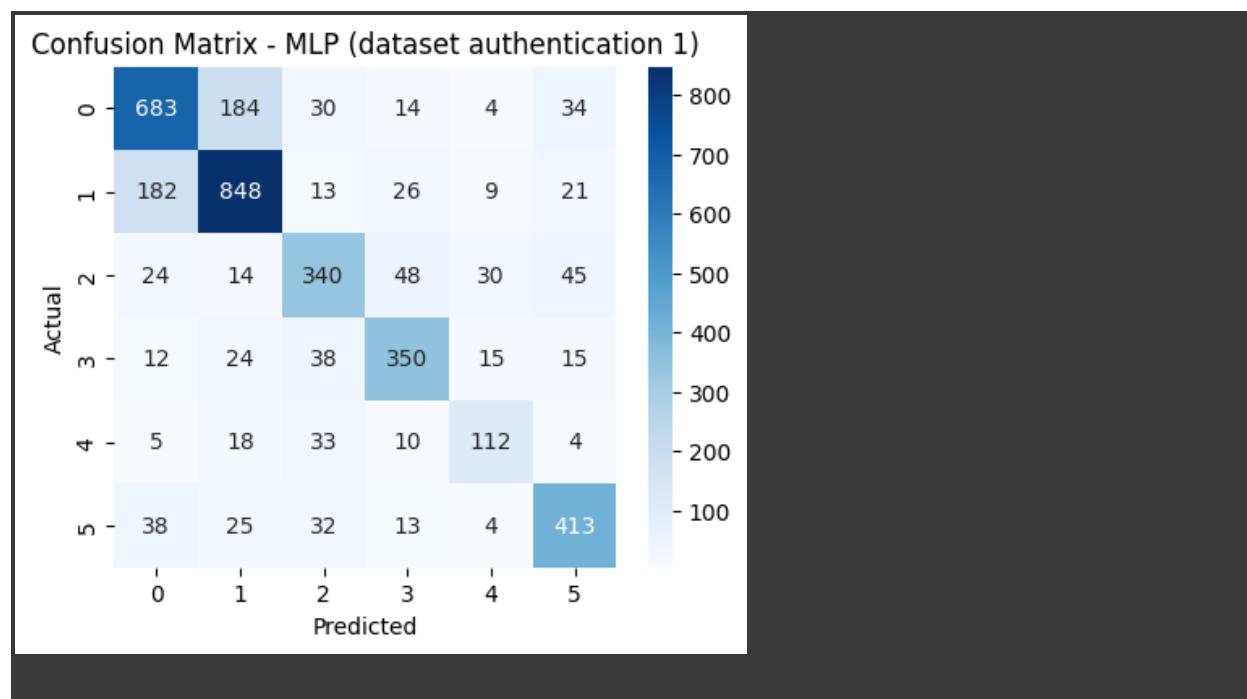


--- MLP Classification Report ---

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

610399182	0.72	0.72	0.72	949
810101575	0.76	0.77	0.77	1099
810102187	0.70	0.68	0.69	501
810103197	0.76	0.77	0.77	454
810103224	0.64	0.62	0.63	182
810199569	0.78	0.79	0.78	525

accuracy	0.74	3710
macro avg	0.73	0.72
weighted avg	0.74	0.74



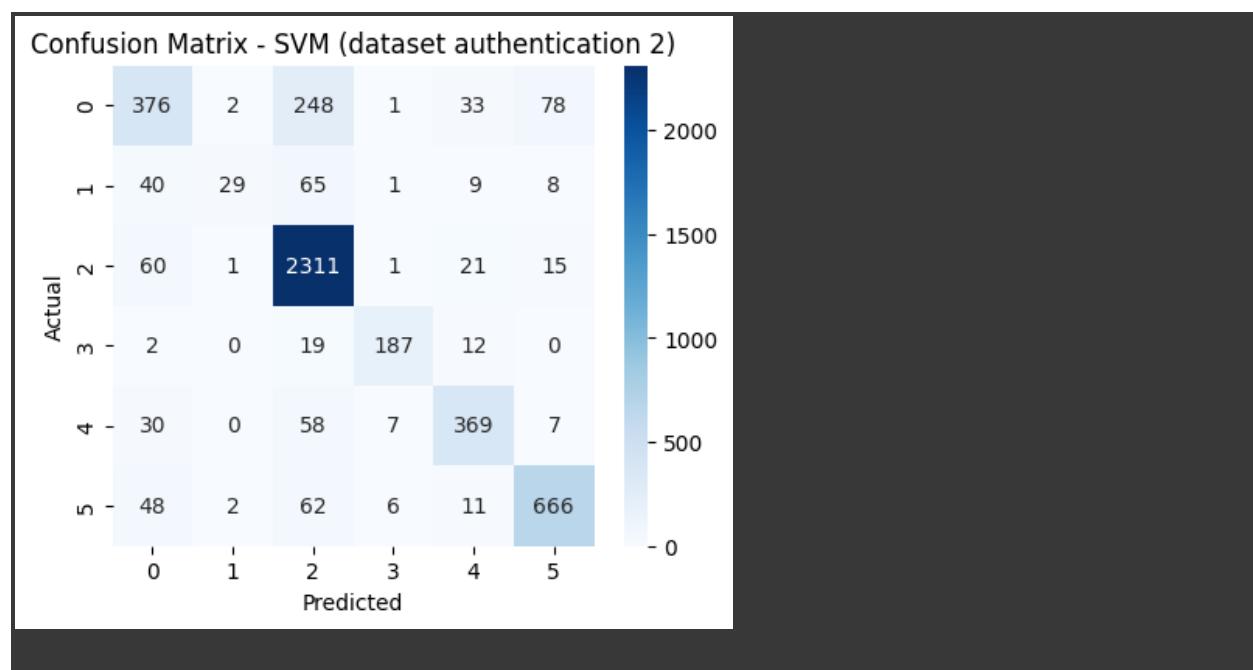
Keys in file: ['features', 'labels']

Loaded /content/drive/MyDrive/Machine Learning Project/dataset authentication 2.npz: X shape = (19138, 171), y shape = (19138,)

dataset authentication 2: Training size = 14353, Test size = 4785

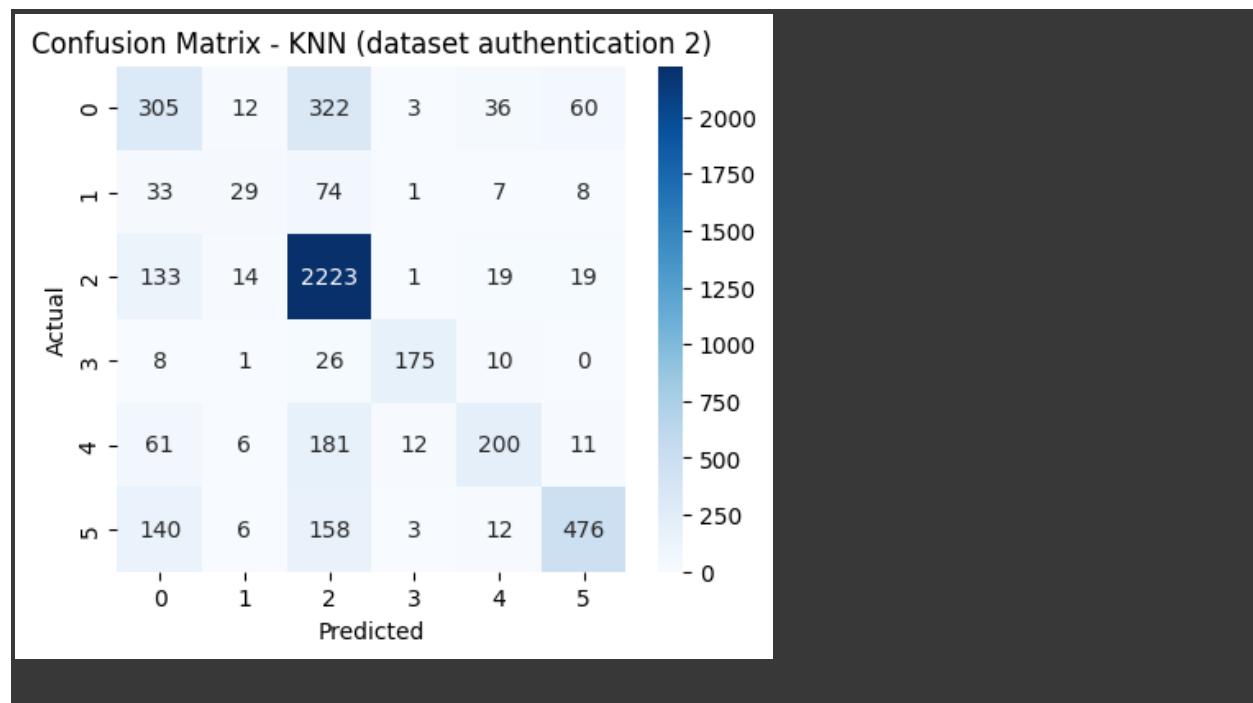
--- SVM Classification Report ---

	precision	recall	f1-score	support
810100217	0.68	0.51	0.58	738
810102345	0.85	0.19	0.31	152
810103054	0.84	0.96	0.89	2409
810103123	0.92	0.85	0.88	220
810103262	0.81	0.78	0.80	471
810199319	0.86	0.84	0.85	795
accuracy		0.82		4785
macro avg	0.83	0.69	0.72	4785
weighted avg	0.82	0.82	0.81	4785



--- KNN Classification Report ---

	precision	recall	f1-score	support
810100217	0.45	0.41	0.43	738
810102345	0.43	0.19	0.26	152
810103054	0.74	0.92	0.82	2409
810103123	0.90	0.80	0.84	220
810103262	0.70	0.42	0.53	471
810199319	0.83	0.60	0.70	795
accuracy		0.71		4785
macro avg	0.68	0.56	0.60	4785
weighted avg	0.71	0.71	0.70	4785

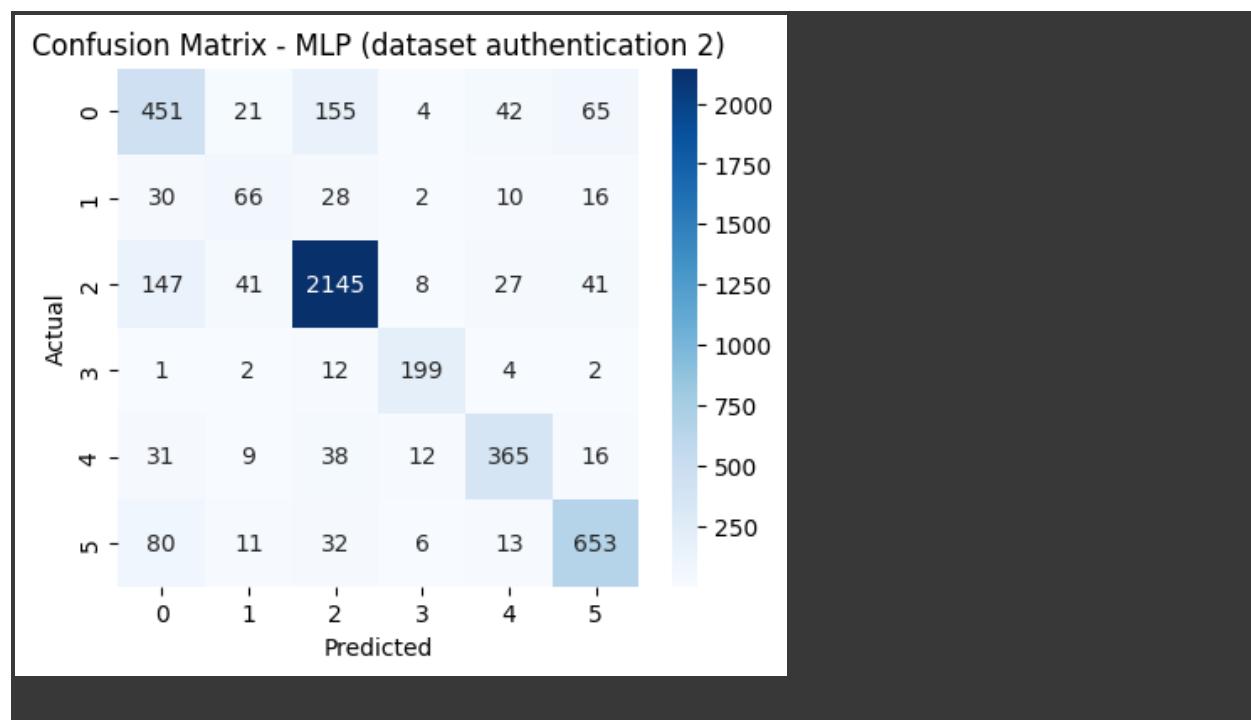


--- MLP Classification Report ---

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

810100217	0.61	0.61	0.61	738
810102345	0.44	0.43	0.44	152
810103054	0.89	0.89	0.89	2409
810103123	0.86	0.90	0.88	220
810103262	0.79	0.77	0.78	471
810199319	0.82	0.82	0.82	795

accuracy	0.81	4785
macro avg	0.74	0.74
weighted avg	0.81	0.81



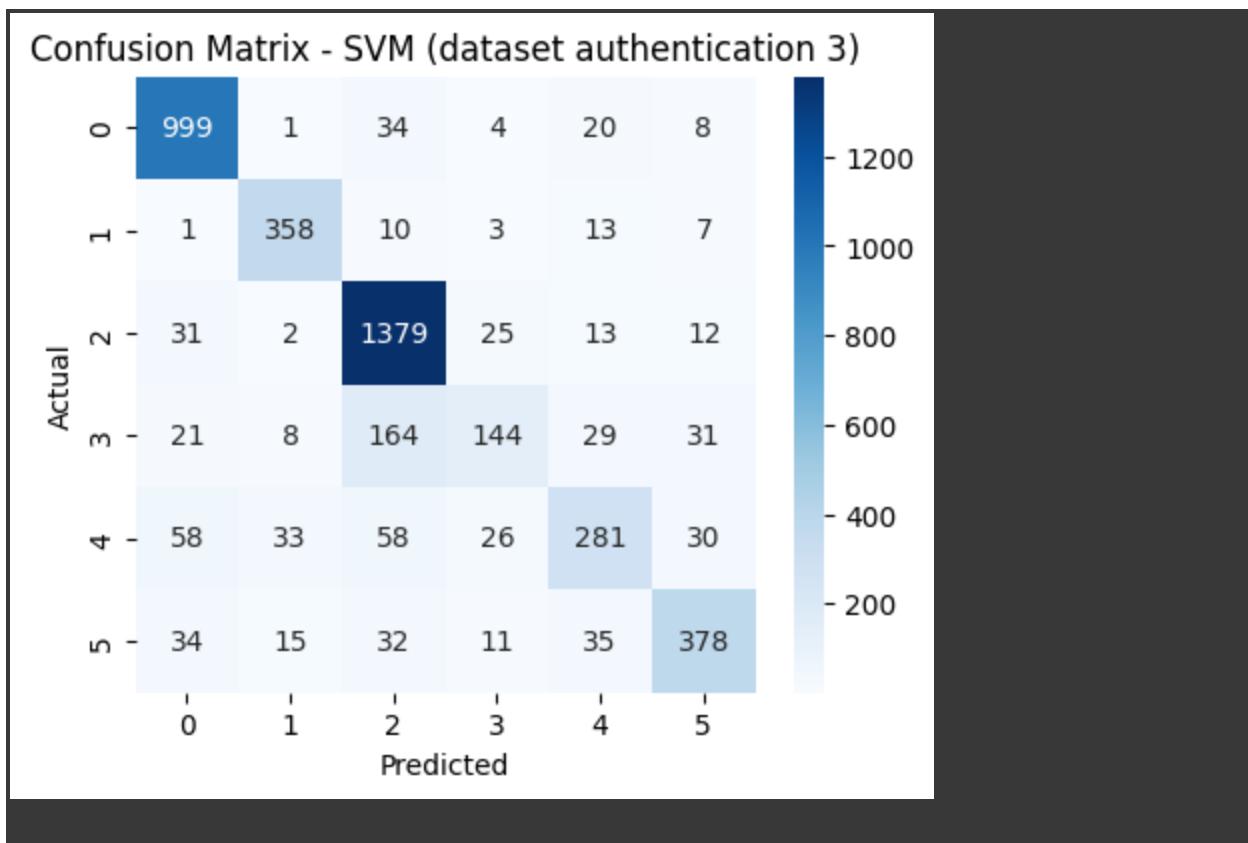
Keys in file: ['features', 'labels']

Loaded /content/drive/MyDrive/Machine Learning Project/dataset authentication 3.npz: X shape = (17229, 171), y shape = (17229,)

dataset authentication 3: Training size = 12921, Test size = 4308

--- SVM Classification Report ---

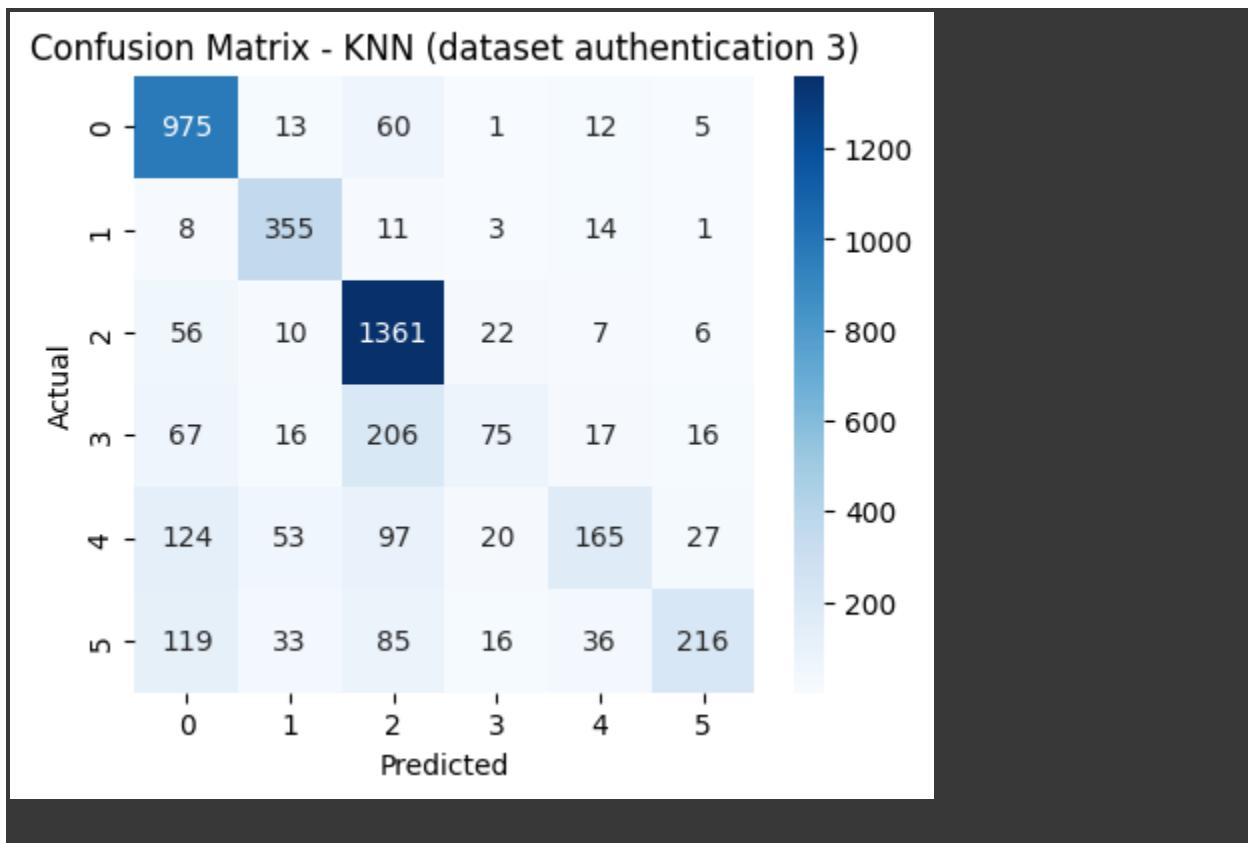
	precision	recall	f1-score	support
610399206	0.87	0.94	0.90	1066
810101471	0.86	0.91	0.89	392
810103317	0.82	0.94	0.88	1462
810199423	0.68	0.36	0.47	397
810600065	0.72	0.58	0.64	486
810600088	0.81	0.75	0.78	505
accuracy		0.82		4308
macro avg	0.79	0.75	0.76	4308
weighted avg	0.81	0.82	0.81	4308



-- KNN Classification Report --

	precision	recall	f1-score	support
610399206	0.72	0.91	0.81	1066
810101471	0.74	0.91	0.81	392
810103317	0.75	0.93	0.83	1462
810199423	0.55	0.19	0.28	397
810600065	0.66	0.34	0.45	486
810600088	0.80	0.43	0.56	505

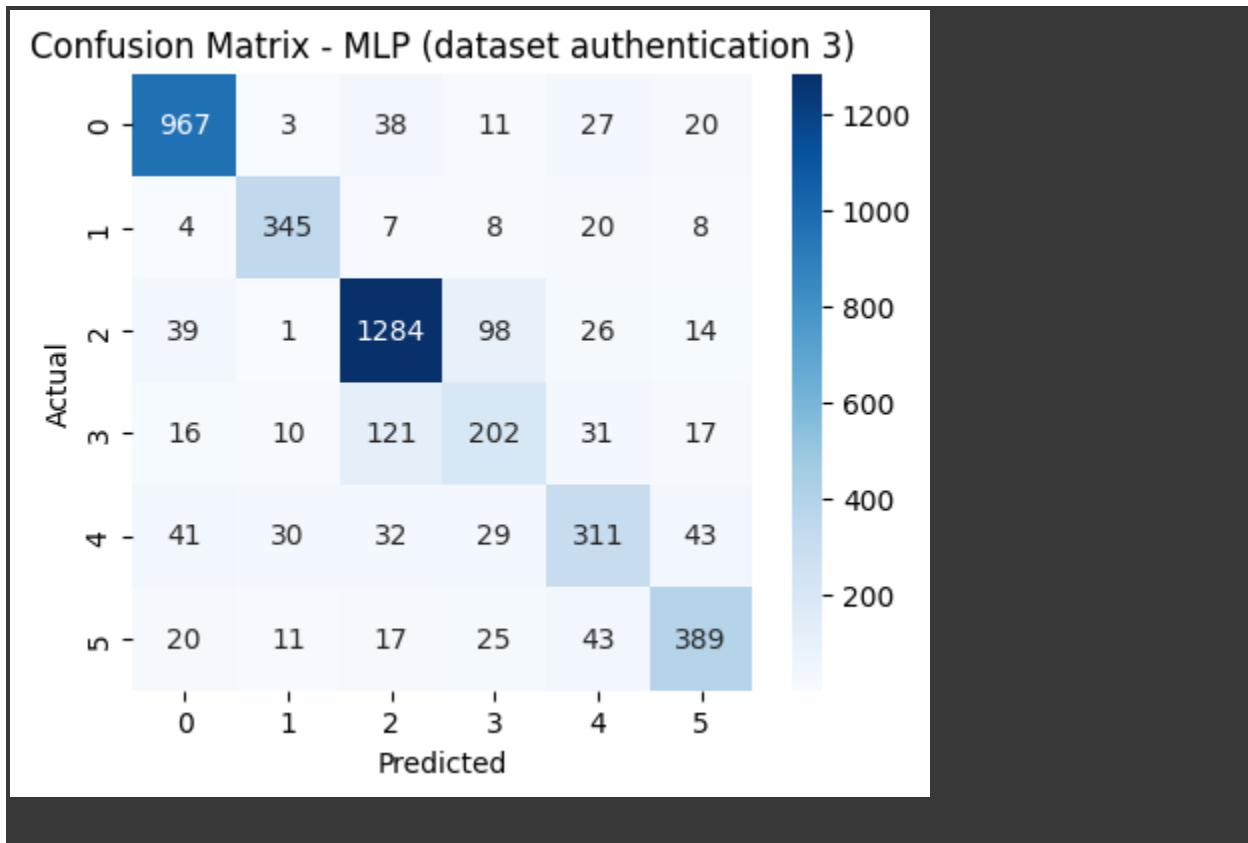
accuracy	0.73	4308
macro avg	0.70	0.62
weighted avg	0.72	0.73



--- MLP Classification Report ---

	precision	recall	f1-score	support
610399206	0.89	0.91	0.90	1066
810101471	0.86	0.88	0.87	392
810103317	0.86	0.88	0.87	1462
810199423	0.54	0.51	0.52	397
810600065	0.68	0.64	0.66	486
810600088	0.79	0.77	0.78	505
accuracy			0.81	4308
macro avg	0.77	0.76	0.77	4308

weighted avg 0.81 0.81 0.81 4308



توضیحات نتایج:

ما دقت های هر یک از 3 تا مدل svm, knn, MLP برای همه دیتاست به دست آورده ایم.

مجموعه داده 1

- تعداد کل نمونه ها: 14838 •
- تعداد ویژگی ها: 171 •
- تعداد داده های آموزشی: 11128 •
- تعداد داده های آزمایشی: 3710 •

نتایج مدل ها:

:SVM

دقت کلی Accuracy): 77% •

:KNN

دقت کلی Accuracy): 64% •

:MLP

دقت کلی Accuracy): 74% •

مجموعه داده 2

تعداد کل نمونه‌ها: 19138 •

تعداد ویژگی‌ها: 171 •

تعداد داده‌های آموزشی: 14353 •

تعداد داده‌های آزمایشی: 4785 •

نتایج مدل‌ها:

:SVM

دقت کلی Accuracy): 82% •

:KNN

دقت کلی Accuracy): 71% •

:MLP

دقت کلی Accuracy): 81% •

مجموعه داده 3

تعداد کل نمونه‌ها: 17229 •

تعداد ویژگی‌ها: 171 •

تعداد داده‌های آموزشی: 12921 •

تعداد داده‌های آزمایشی: 4308 •

نتایج مدل‌ها:

:SVM

دقت کلی 82% • Accuracy:

:KNN

دقت کلی 73%) • Accuracy):

:MLP

دقت کلی 81%) • Accuracy):

تشخیص جنسیت:

:کد:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import f1_score, recall_score, precision_score,
confusion_matrix, roc_curve, auc

from google.colab import drive
drive.mount('/content/drive')

file_path = "/content/drive/MyDrive/Machine Learning Project/dataset
gender.npz"
data = np.load(file_path)

X = data["features"]
y = data["labels"]

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42, stratify=y)

models = {
    "SVM": SVC(probability=True, random_state=42),
    "KNN": KNeighborsClassifier(n_neighbors=5),
```

```

    "MLP": MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=500,
random_state=42)
}

for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1] if hasattr(model,
"predict_proba") else None

    f1 = f1_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

    print(f"\n{name} Results:")
    print(f"F1 Score: {f1:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"Precision: {precision:.4f}")
    print("Confusion Matrix:")
    print(cm)

    if y_prob is not None:
        fpr, tpr, _ = roc_curve(y_test, y_prob)
        roc_auc = auc(fpr, tpr)

        plt.figure()
        plt.plot(fpr, tpr, color="blue", lw=2, label=f"ROC curve (area = {roc_auc:.2f})")
        plt.plot([0, 1], [0, 1], color="gray", linestyle="--")
        plt.xlabel("False Positive Rate")
        plt.ylabel("True Positive Rate")
        plt.title(f"ROC Curve - {name}")
        plt.legend(loc="lower right")
        plt.show()

```

توضیحات کد:

مانند کردن گوگل درایو: ابتدا از گوگل درایو فایل داده‌ها را بارگذاری می‌کند.

بارگذاری داده‌ها: داده‌ها از فایل `gender.npz` بارگذاری می‌شوند که شامل ویژگی‌ها (`features`) و برچسب‌ها (`labels`) است.

نرمال‌سازی ویژگی‌ها: ویژگی‌های ورودی با استفاده از `StandardScaler` نرمال‌سازی می‌شوند تا مقیاس آن‌ها یکسان شود.

تقسیم داده‌ها: داده‌ها به دو بخش تقسیم می‌شوند: 75% برای آموزش و 25% برای آزمون.

تعریف مدل‌ها: سه مدل مختلف (MLP، SVM، KNN) برای آموزش و ارزیابی آماده می‌شوند.

آموزش و ارزیابی مدل‌ها: مدل‌ها روی داده‌های آموزشی آموزش داده می‌شوند و سپس روی داده‌های آزمایشی ارزیابی می‌شوند. ارزیابی شامل معیارهای زیر است:

- F1 امتیاز
- حساسیت (Recall)
- دقیقت (Precision)
- ماتریس اشتباهات (Confusion Matrix)
- نمودار ROC (برای مدل‌هایی که قابلیت پیش‌بینی احتمال دارند)

نتایج:

...Training KNN

:KNN Results

F1 Score: 0.7363

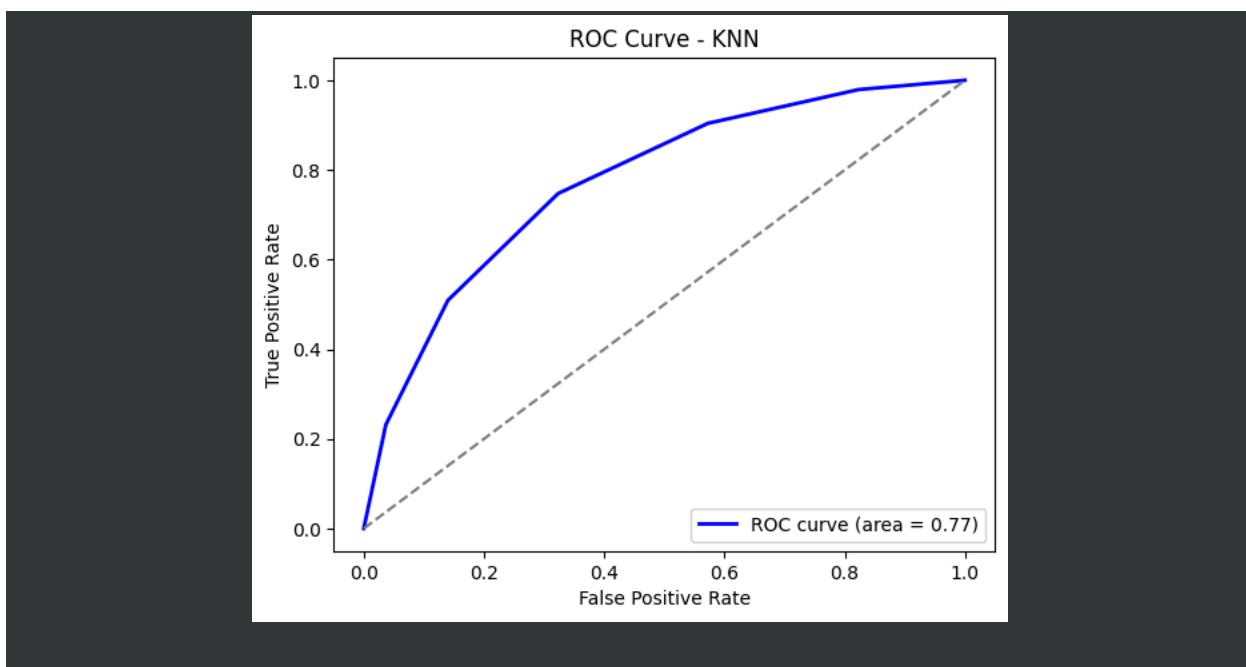
Recall: 0.7473

Precision: 0.7256

:Confusion Matrix

[2911 6093]]

[[7699 2603]



...Training SVM

:SVM Results

F1 Score: 0.8361

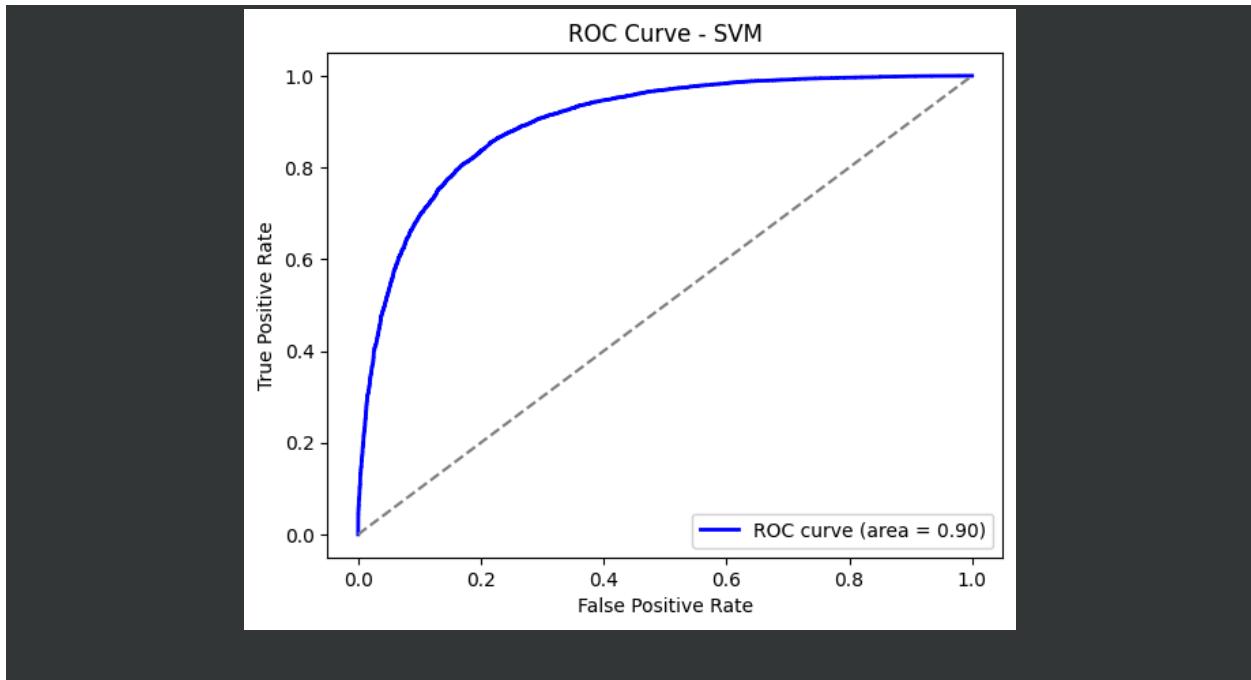
Recall: 0.8535

Precision: 0.8194

:Confusion Matrix

[1938 7066]]

[[8793 1509]



...Training MLP

:MLP Results

F1 Score: 0.8259

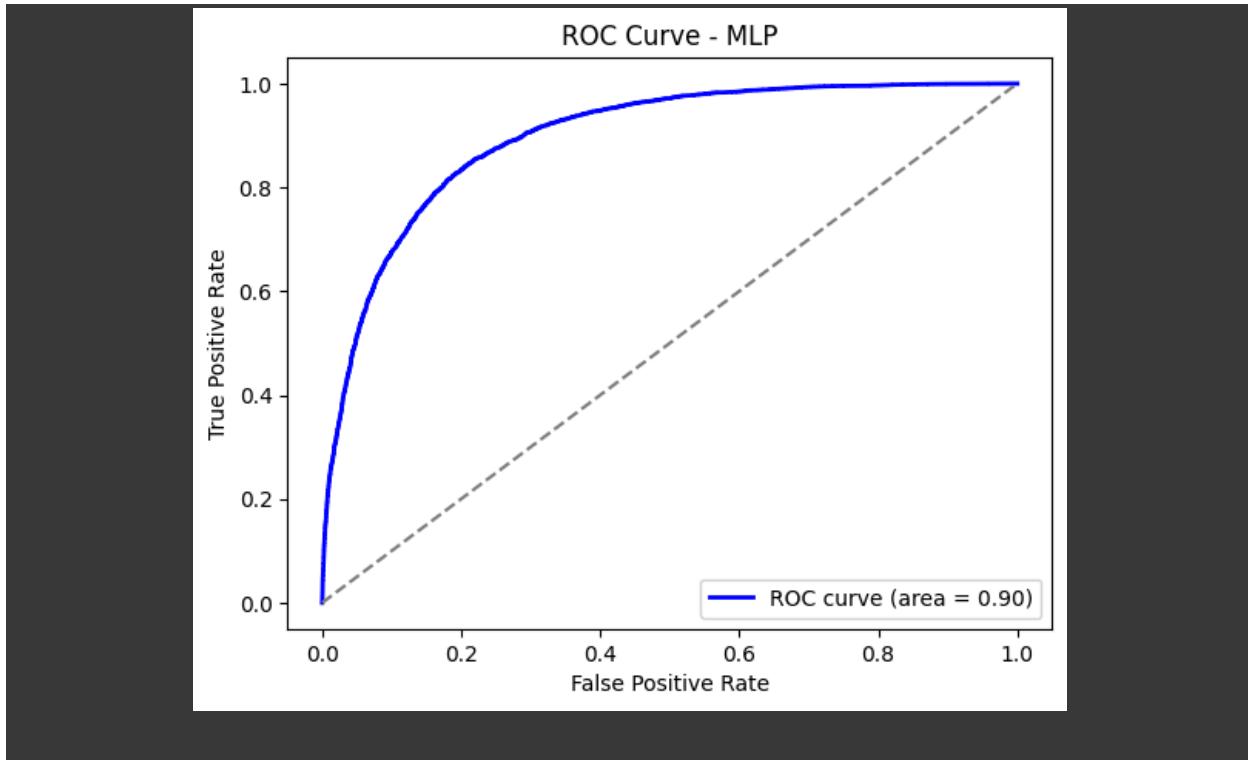
Recall: 0.8160

Precision: 0.8362

:Confusion Matrix

[1647 7357]]

[[8406 1896]



توضیحات نتایج:

نتایج به دست آمده از آموزش و ارزیابی سه مدل KNN، SVM و MLP روی مجموعه داده‌ی تشخیص جنسیت بررسی می‌شود. معیارهای ارزیابی شامل F1 Score، Recall، Precision و ماتریس اشتباهات (Confusion Matrix) هستند.

۱. مدل (KNN (K-Nearest Neighbors

F1 Score: 0.7363 •

Recall: 0.7473 •

Precision: 0.7256 •

• ماتریس اشتباهات:

[2911 6093] •
[[7699 2603] •

تحلیل:

مدل KNN در مقایسه با سایر مدل‌ها عملکرد ضعیفتری دارد. دقت (Precision) و یادآوری (Recall) آن پایین‌تر از سایر مدل‌ها است، که نشان می‌دهد احتمال خطا طبقه‌بندی در این مدل بالاتر است. به خصوص، مقدار بالای (2911 FP نمونه نادرست در دسته‌ی اول) و (2603 FN نمونه نادرست در دسته‌ی دوم) نشان می‌دهد که این مدل در تشخیص صحیح جنسیت با مشکل مواجه است.

۲. مدل (SVM (Support Vector Machine

F1 Score: 0.8361 •
Recall: 0.8535 •
Precision: 0.8194 •
ماتریس اشتباهات: •

[1938 7066] •
[[8793 1509] •

تحلیل:

مدل SVM عملکرد بهتری نسبت به KNN دارد. مقدار F1 Score بالاتر (0.8361) نشان می‌دهد که تعادل خوبی بین دقت و یادآوری ایجاد شده است. مقدار FP (1938 نمونه نادرست) و FN (1509 نمونه نادرست) در مقایسه با مدل KNN کاهش یافته است، که نشان دهنده‌ی بهبود کلی دقت مدل است.

۳. مدل (MLP (Multi-Layer Perceptron

F1 Score: 0.8259 •
Recall: 0.8160 •
Precision: 0.8362 •

• ماتریس اشتباهات:

[1647 7357] •
[[8406 1896] •

تحلیل:

مدل MLP عملکردی مشابه با SVM دارد. مقدار Precision (0.8362) آن کمی بالاتر از SVM است، اما مقدار Recall (0.8160) کمی پایین‌تر است. این مدل در مقایسه با SVM تعداد بیشتری FN (1896 نمونه نادرست) دارد، اما در مقابل FP (1647 نمونه نادرست) کمتری تولید کرده است. به همین دلیل، دقیق آن در مقایسه با SVM کمی بالاتر، اما قابلیت تشخیص نمونه‌های مثبت کمی ضعیفتر است.

جمع‌بندی و مقایسه مدل‌ها

SVM بهترین مدل در این آزمایش است زیرا بالاترین مقدار Recall (0.8535) و F1 Score (0.8361) را دارد، که نشان می‌دهد به طور کلی در طبقه‌بندی بهتر عمل می‌کند.

1. MLP نیز عملکرد خوبی دارد، اما مقدار Recall آن از SVM کمتر است، به این معنی که ممکن است برخی نمونه‌ها را به درستی شناسایی نکند.
2. KNN ضعیفترین عملکرد را دارد، زیرا مقادیر F1 Score، Precision و Recall آن از دو مدل دیگر کمتر است.

نتیجه نهایی: مدل SVM به عنوان بهترین مدل برای این مجموعه داده پیشنهاد می‌شود.

5. خوشبندی

در این قسمت، به کمک ویژگی‌های استخراج شده در بخش قبلی و بدون استفاده از ستون هدف (در تفکیک جنسیت، مرد یا زن بودن، و در تشخیص هویت، شماره دانشجویی 6 نفر) سعی می‌کنیم داده‌ها را خوشبندی کنیم.

تشخیص هویت:

در قسمت قبل، مجموعه داده 6 نفر در سه فایل مجزا به نام‌های 1, 2, 3 Feature authentication ذخیره شده اند که به فرمت npy در آمده‌اند. به کمک کد زیر، فایل‌ها را می‌خوانیم:

```
data_1 = np.load('/content/drive/MyDrive/Machine Learning Project/features  
('authentication 1.npy
```

```
data_2 = np.load('/content/drive/MyDrive/Machine Learning Project/features authentication 2.npy')
data_3 = np.load('/content/drive/MyDrive/Machine Learning Project/features authentication 3.npy')
```

در قسمت بعدی، برای انجام عمل کاهش بعد (برای نمایش داده ها) و انجام بهتر محاسبات، ستون ها را normalize می کنیم. کد زیر این کار را برای هر سه مجموعه داده انجام می دهد.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

def normalizer(data):
    scaler = StandardScaler()
    data_normalized = scaler.fit_transform(data)
    return data_normalized

full_data = normalizer(full_data)
                                         به مجموعه داده مربوط به تفکیک جنسیت اشاره دارد)

data_1 = normalizer(data_1)
data_2 = normalizer(data_2)
data_3 = normalizer(data_3)
```

در کد زیر جهت اطمینان خاطر ابعاد هر کدام از مجموعه داده ها را نمایش می دهیم.

```
print("Gender features shape:", full_data.shape)
print("Auth 1 features shape:", data_1.shape)
print("Auth 2 features shape:", data_2.shape)
print("Auth 3 features shape:", data_3.shape)
```

```
Gender features shape: (154552, 171)
Auth 1 features shape: (14838, 171)
Auth 2 features shape: (19138, 171)
Auth 3 features shape: (17229, 171)
```

برای خوش بندی، از دو روش kmeans و agglomerative hierarchical استفاده می کنیم. کد زیر تابع مربوط به خوش بندی به کمک kmeans را نمایش می دهد:

```
from sklearn.cluster import DBSCAN, KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
```

```

def clustering_with_kmeans(data, max_cluster=10):
    inertia = []
    silhouette = []
    pca = PCA(n_components=0.95)
    pca_data = pca.fit_transform(data)

    for n_clusters in range(2, max_cluster + 1):
        print("number of clusters: ", n_clusters)
        kmeans = KMeans(n_clusters = n_clusters, random_state = 42)
        clustered_kmeans = kmeans.fit_predict(pca_data)
        inertia.append(kmeans.inertia_)
        silhouette.append(silhouette_score(pca_data, clustered_kmeans))

    data_visualization = TSNE(n_components = 2).fit_transform(pca_data)
    plt.scatter(data_visualization[:, 0], data_visualization[:, 1],
c=clustered_kmeans, cmap='tab10')
    plt.title(f"number of clusters: {n_clusters}")
    plt.show()

    plt.plot(range(2, max_cluster + 1), inertia, marker='o')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Inertia')
    plt.title('Elbow Method for Optimal k')
    plt.show()

    plt.plot(range(2, max_cluster + 1), silhouette, marker='o')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Silhouette Score')
    plt.title('Silhouette Score for Optimal k')
    plt.show()

```

در قسمت اول تابع، برای یافتن بهترین تعداد خوشه برای مجموعه داده مان، دو آرایه به نام های `inertia` و `silhouette` می سازیم (در حقیقت به دنبال پیاده سازی روش های elbow و silhouette scoring هستیم) برای اینکه محاسبات در قسمت خوشه بندی پیچیدگی محاسباتی نسبتا کمتری داشته باشد، به کمک PCA ابعاد مجموعه داده را طوری کاهش می دهیم که تنها 5 درصد اطلاعات را از دست داده باشیم (در حقیقت، 5 درصد واریانس داده ها).

سپس به کمک تابع `KMeans`، داده ها را خوشه بندی کرده و `label` ها را در یک آرایه به نام `clustered_kmeans` قرار می دهیم. به ازای خوشه بندی برای تعداد خوشه های مختلف (از 2 تا 10)، `inertia` و `silhouette score` را محاسبه می کنیم

به کمک الگوریتم زیر حساب می شود:
به ازای هر نقطه:

فاصله آن نقطه از نقاط درون خوشه ای که درونش است را محاسبه کن و میانگین بگیر (a_i)

فاصله آن نقطه از نقاط بیرون خوشه را محاسبه کن و میانگین بگیر (b_i)

$$\text{مقدار} \frac{(b_i - a_i)}{\max(a_i, b_i)} \text{ را محاسبه کن}$$

میانگین مقدار بدست آمده برای همه نقاط را به عنوان نتیجهٔ نهایی گزارش بده

برای محاسبهٔ inertia، از روش زیر استفاده می شود:

به ازای هر نقطه:

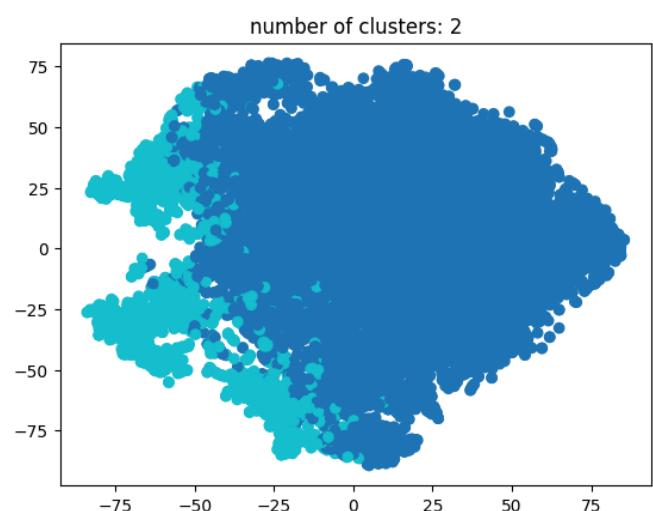
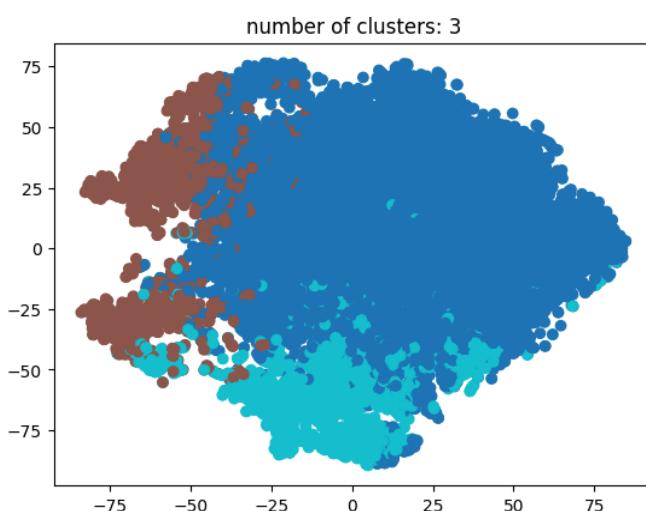
فاصله آن نقطه از مرکز خوشه ای که درونش است را محاسبه کن و به توان دو برسان

حاصل جمع مقدار بدست آمده برای همه نقاط را به عنوان نتیجهٔ نهایی گزارش بده

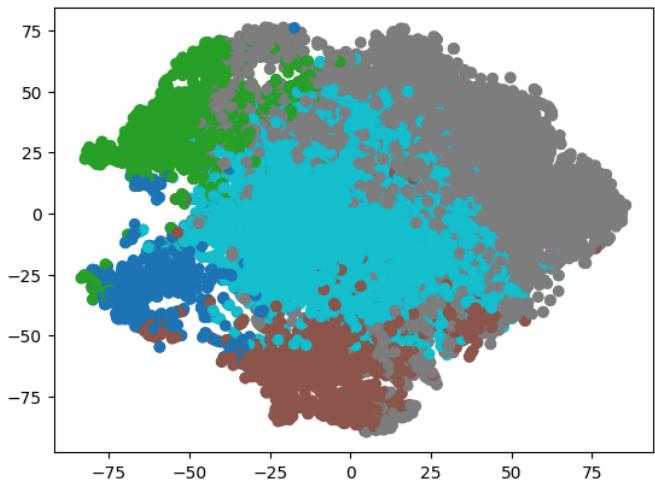
برای جلوگیری از پیچیدگی بیش از اندازهٔ کد و سرعت بهتر، از توابع آماده استفاده کرده ایم.

برای تصویر سازی این خوشه بندی، به کمک تابع TSNE تعداد ابعاد را به دو کاهش می دهیم (یکی از دلایل دیگر استفاده از PCA، توصیه شدن به کاهش تعداد ابعاد قبل از استفاده از TSNE بود). سپس به ازای هر خوشه بندی، تصویر سازی را انجام می دهیم. در نهایت، شکل توابع inertia و silhouette score را رسم می کنیم. نتایج را در آرایهٔ خود می ریزیم.

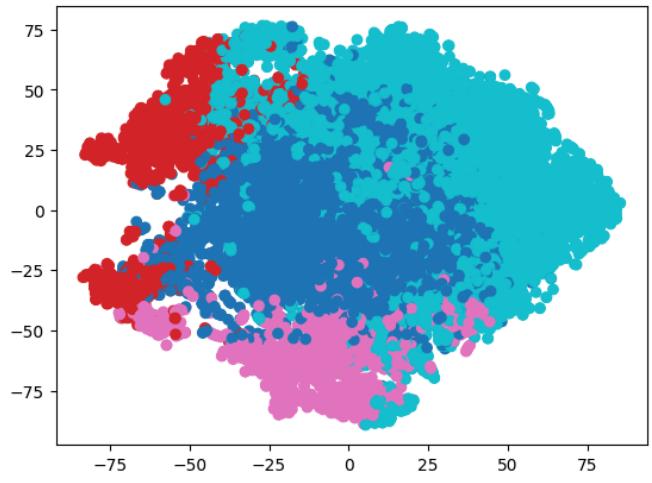
برای تصویر سازی این خوشه بندی، به کمک تابع TSNE تعداد ابعاد را به دو کاهش می دهیم (یکی از دلایل دیگر استفاده از PCA، توصیه شدن به کاهش تعداد ابعاد قبل از استفاده از TSNE بود). سپس به ازای هر خوشه بندی، تصویر سازی را انجام می دهیم. در نهایت، شکل توابع inertia و silhouette score را رسم می کنیم. نتیجه برای مجموعه داده های اول و دوم به شکل زیر است:



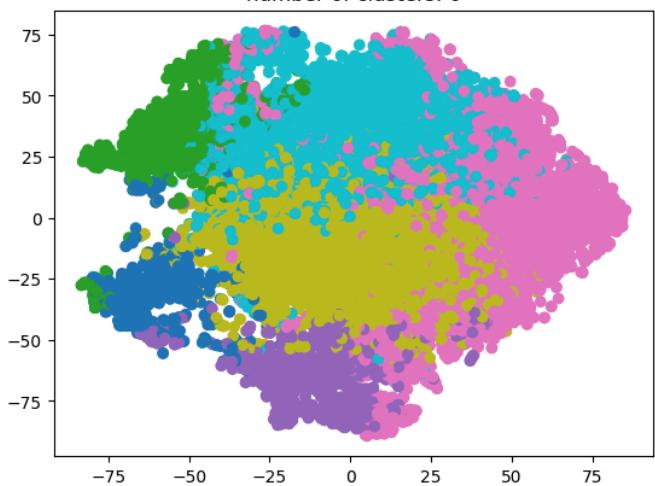
number of clusters: 5



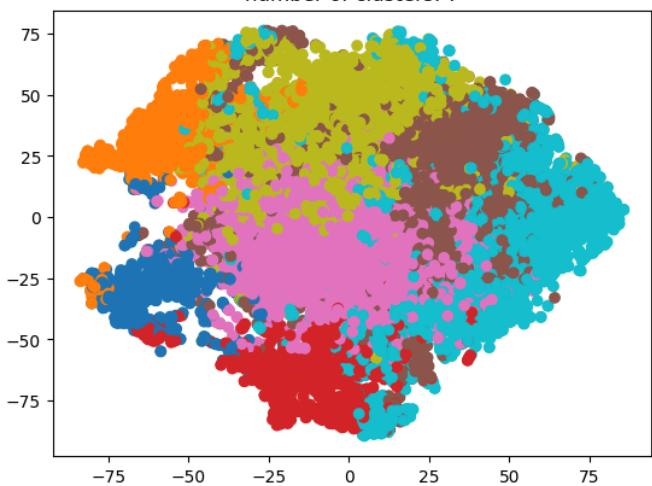
number of clusters: 4



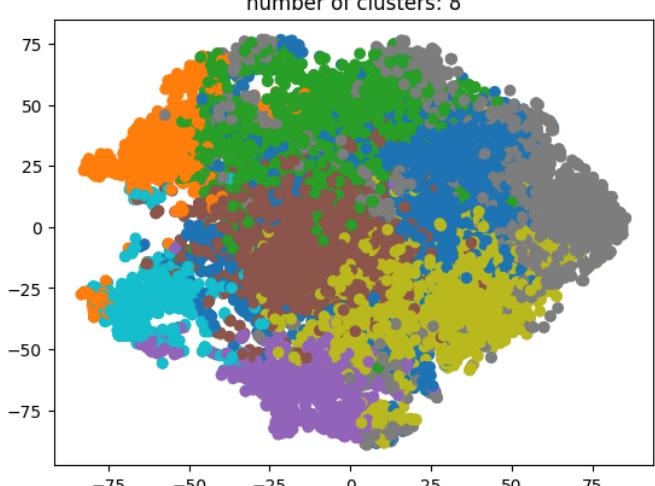
number of clusters: 6



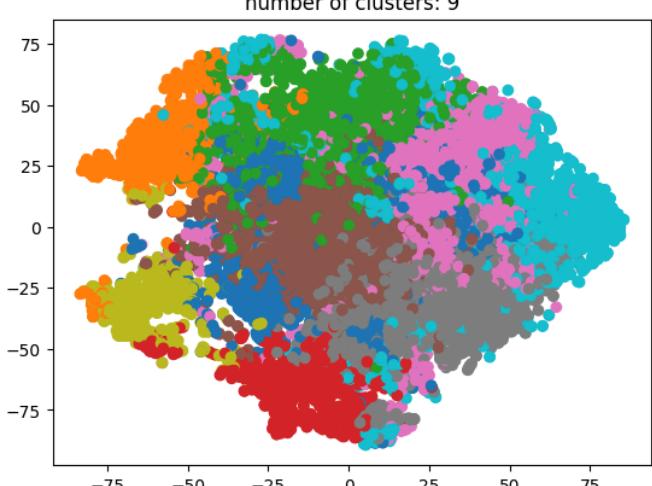
number of clusters: 7



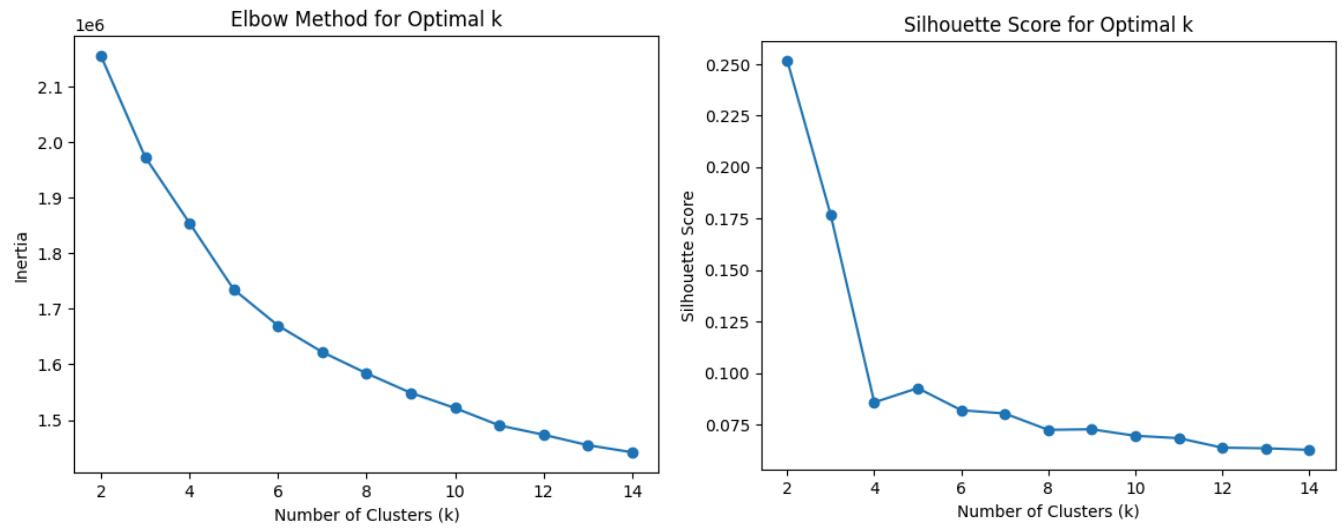
number of clusters: 8



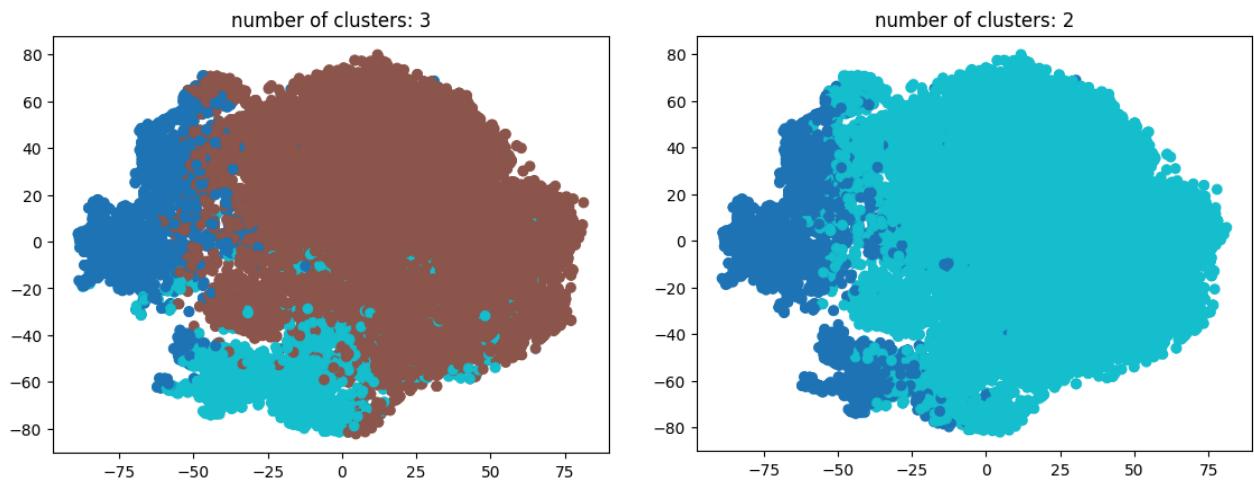
number of clusters: 9

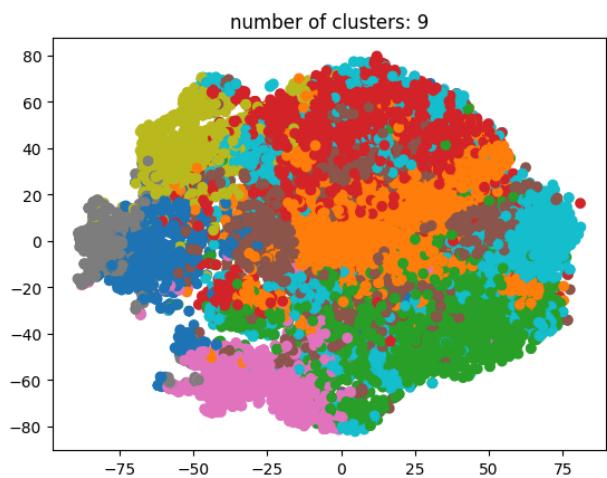
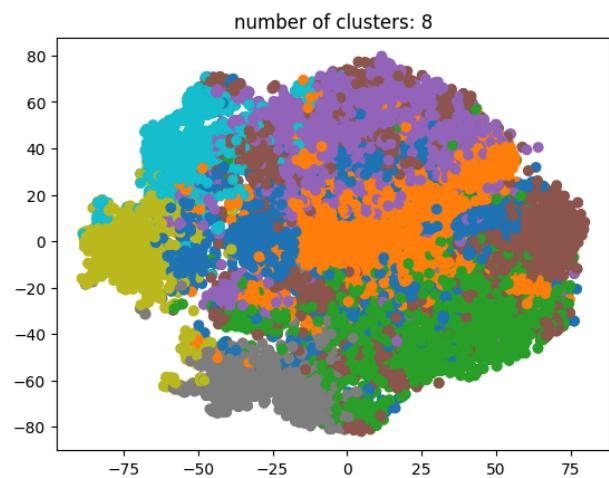
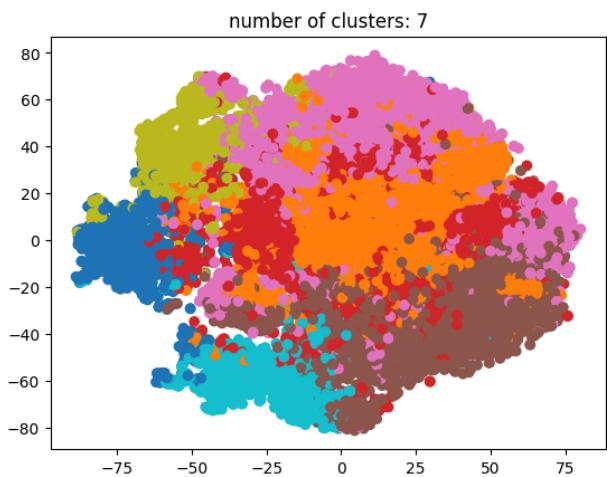
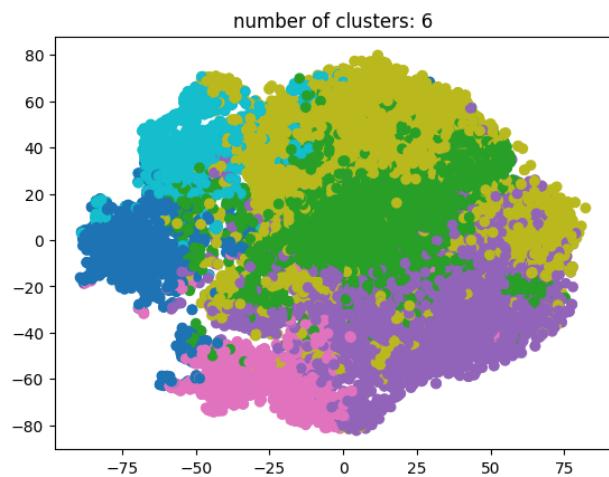
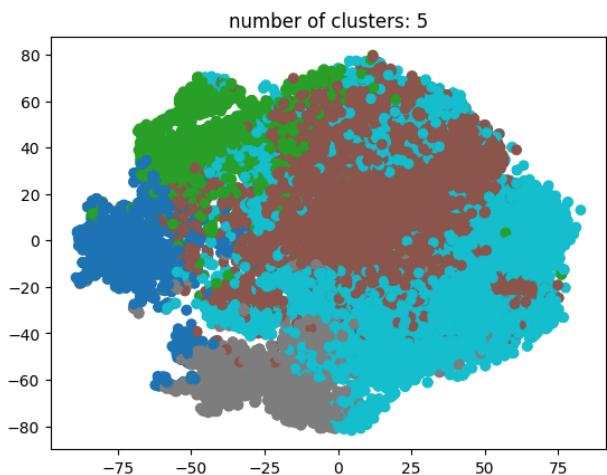
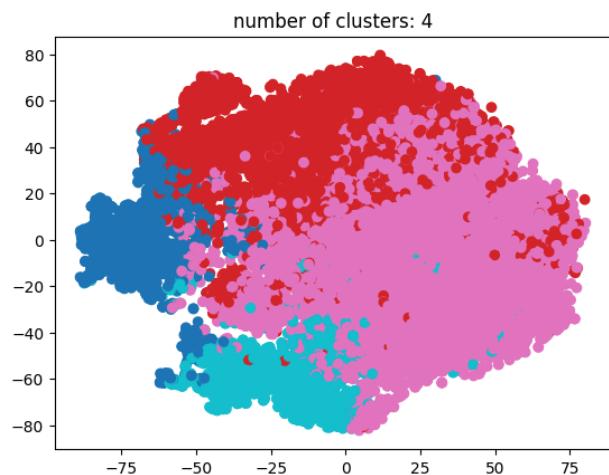


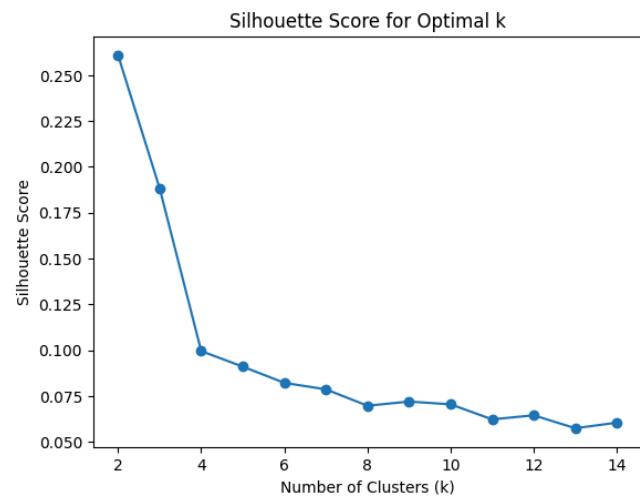
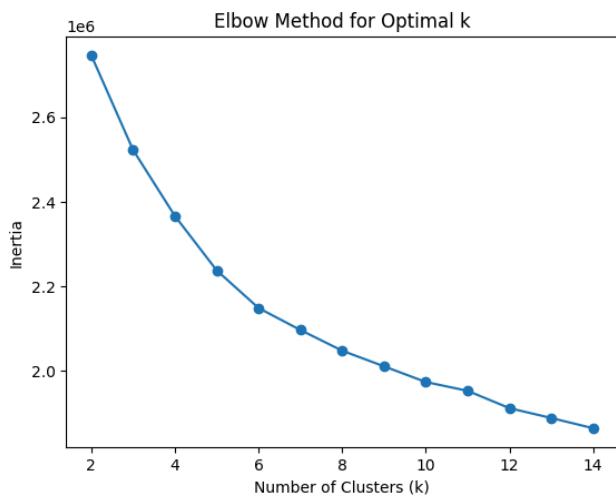
این نتایج برای مجموعه داده اول است.



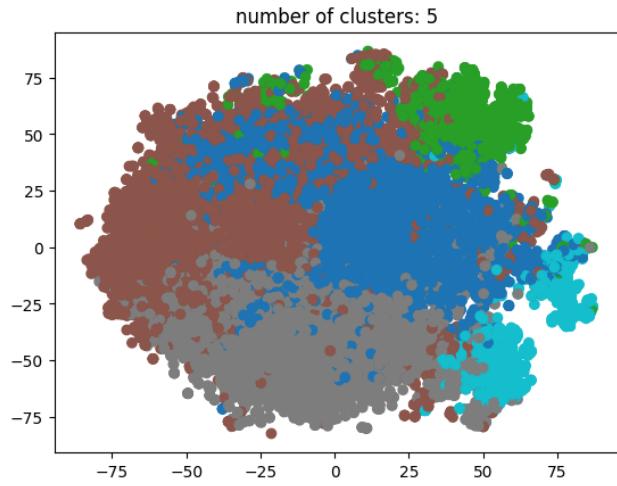
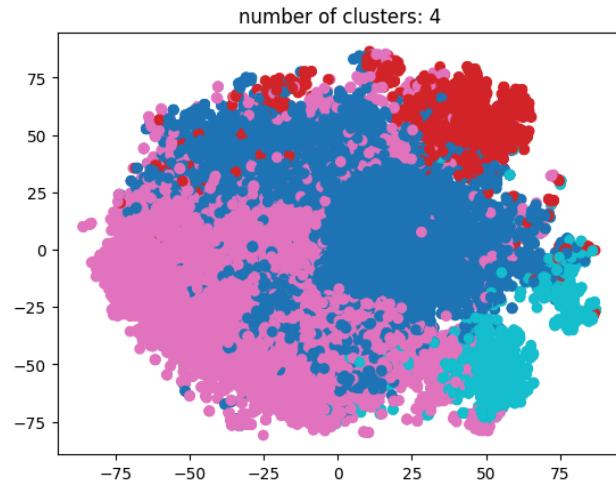
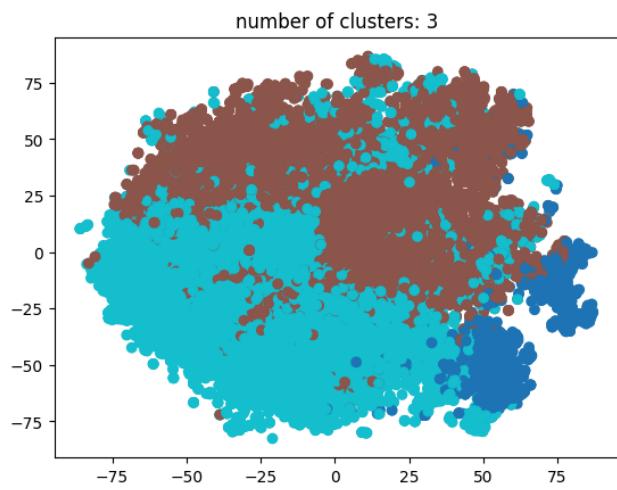
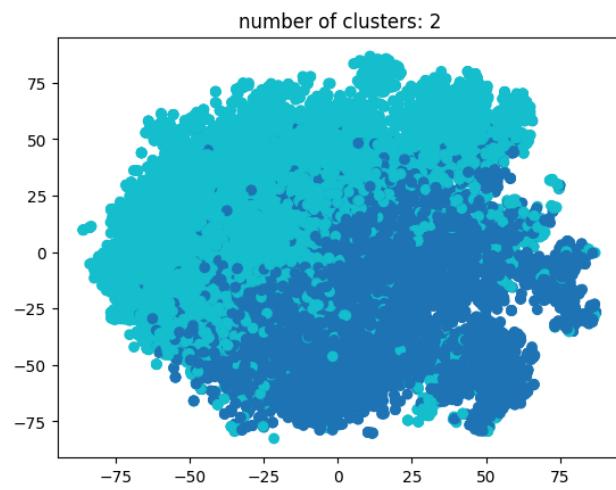
شکل های زیر، نتیجه برای مجموعه داده دوم را نشان می دهد.

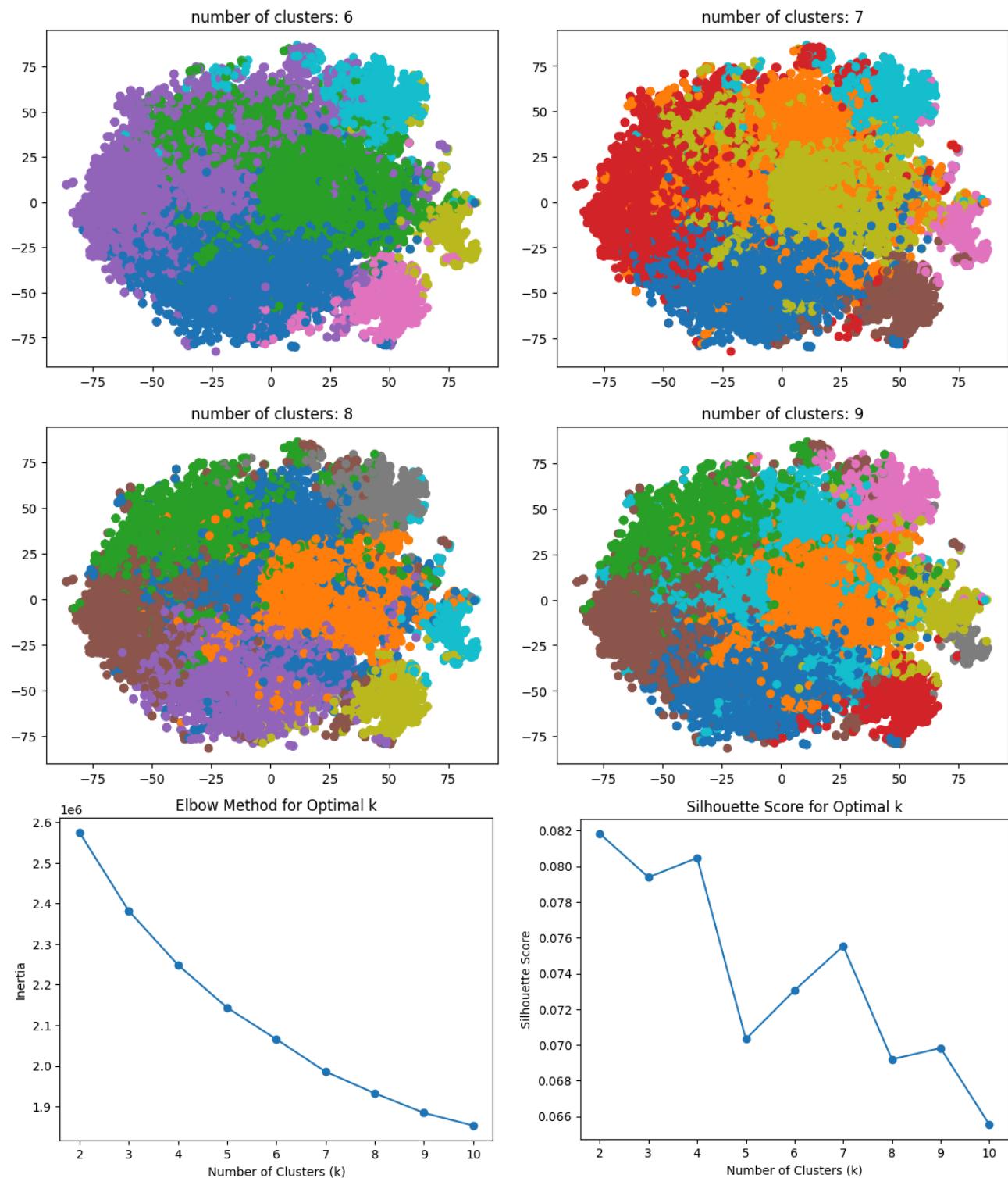






و این نتیجه برای مجموعه داده سوم است.





```

from sklearn.cluster import AgglomerativeClustering

def clustering_with_hierarchical(data, max_cluster):
    inertia = []
    silhouette = []
    pca = PCA(n_components=0.95)
    pca_data = pca.fit_transform(data)

    pca_data.shape
    for n_clusters in range(2, max_cluster + 1):
        print("number of clusters: ", n_clusters)
        hierarchical = AgglomerativeClustering(n_clusters = n_clusters)
        clustered_hierarchical = hierarchical.fit_predict(pca_data)
        silhouette.append(silhouette_score(pca_data,
clustered_hierarchical))

    data_visualization = TSNE(n_components = 2).fit_transform(pca_data)
    plt.scatter(data_visualization[:, 0], data_visualization[:, 1],
c=clustered_hierarchical, cmap='tab10')
    plt.title(f"number of clusters: {n_clusters}")
    plt.show()

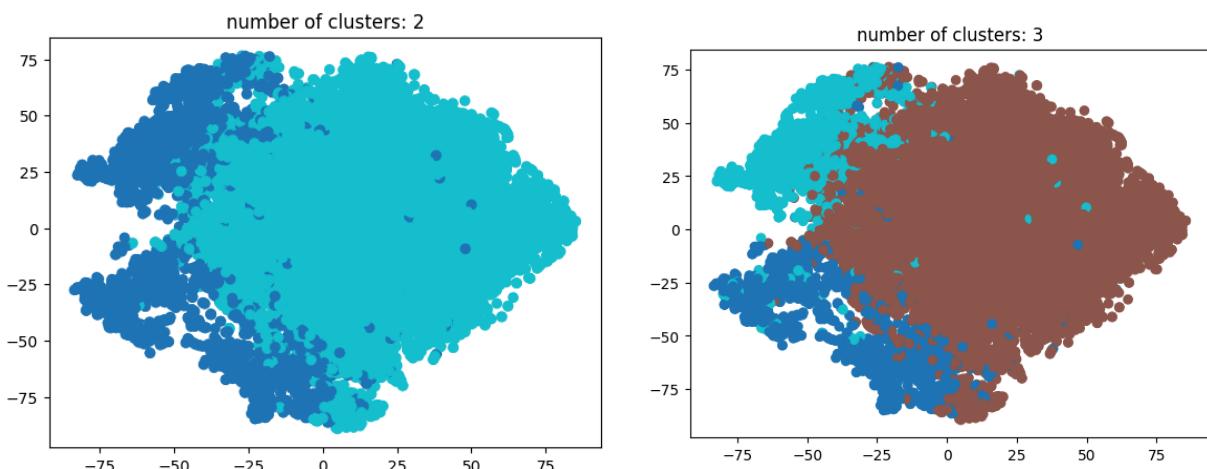
    plt.plot(range(2, max_cluster + 1), silhouette, marker='o')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Silhouette Score')
    plt.title('Silhouette Score for Optimal k')
    plt.show()

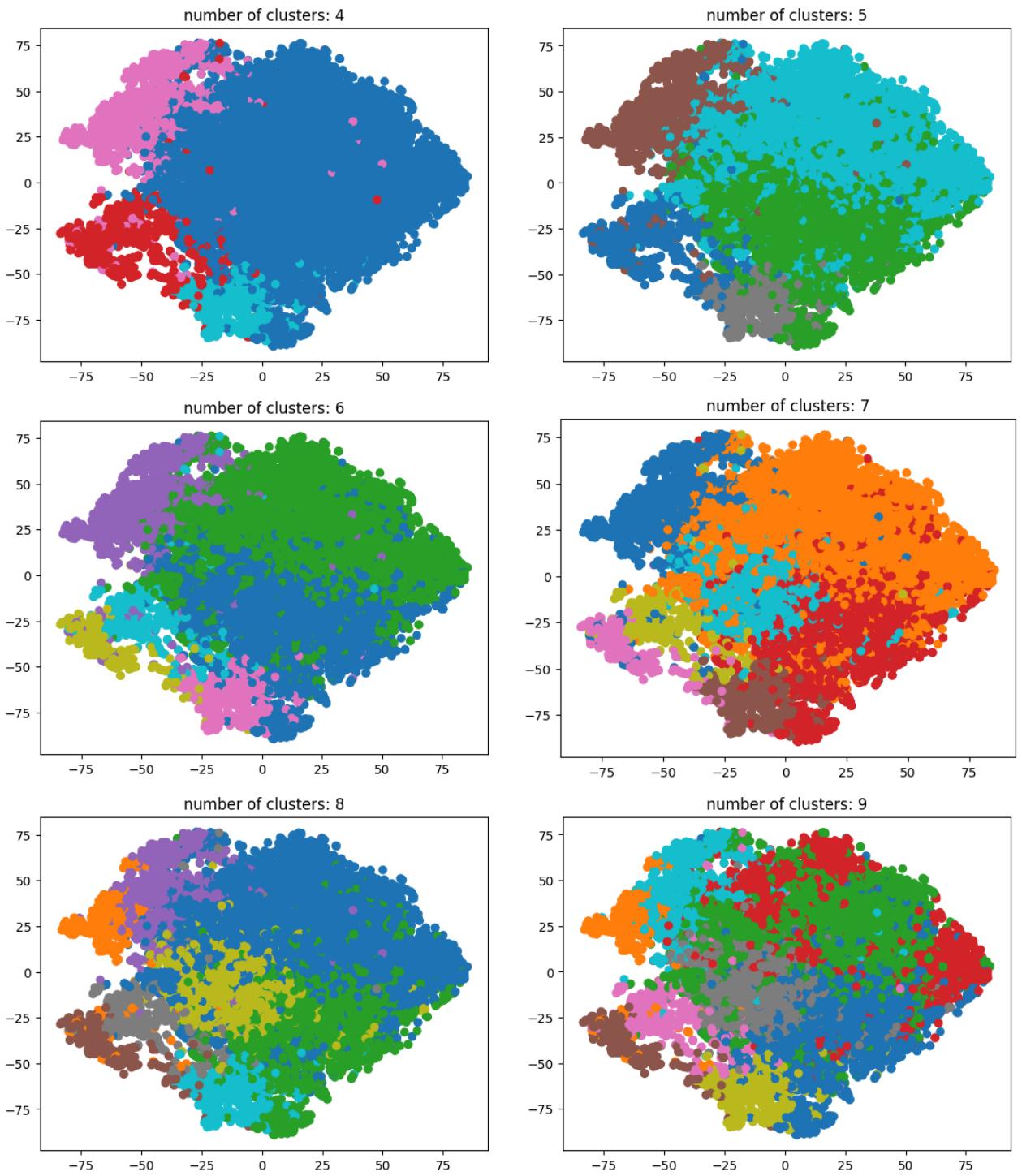
```

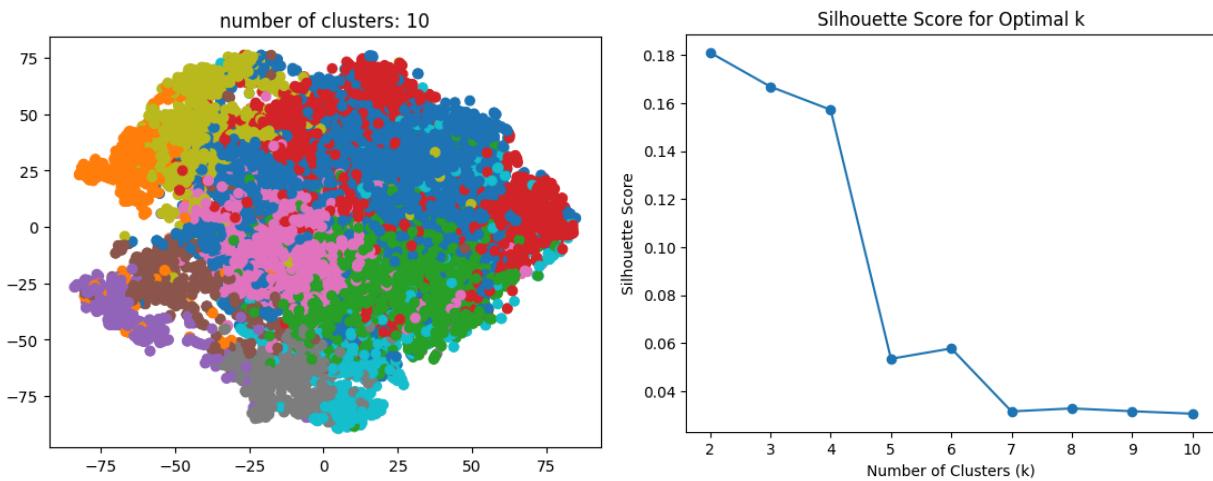
تابع بالا همانند تابع پیشین است اما اینجا به جای استفاده از kmeans آنچایی که روش elbow برای این روش خوش بندی کاربردی نیست، تنها از silhouette score استفاده کردیم.

چون توضیحات تابع همانند تابع پیشین است، صرفا سراغ نتیجه می رویم.

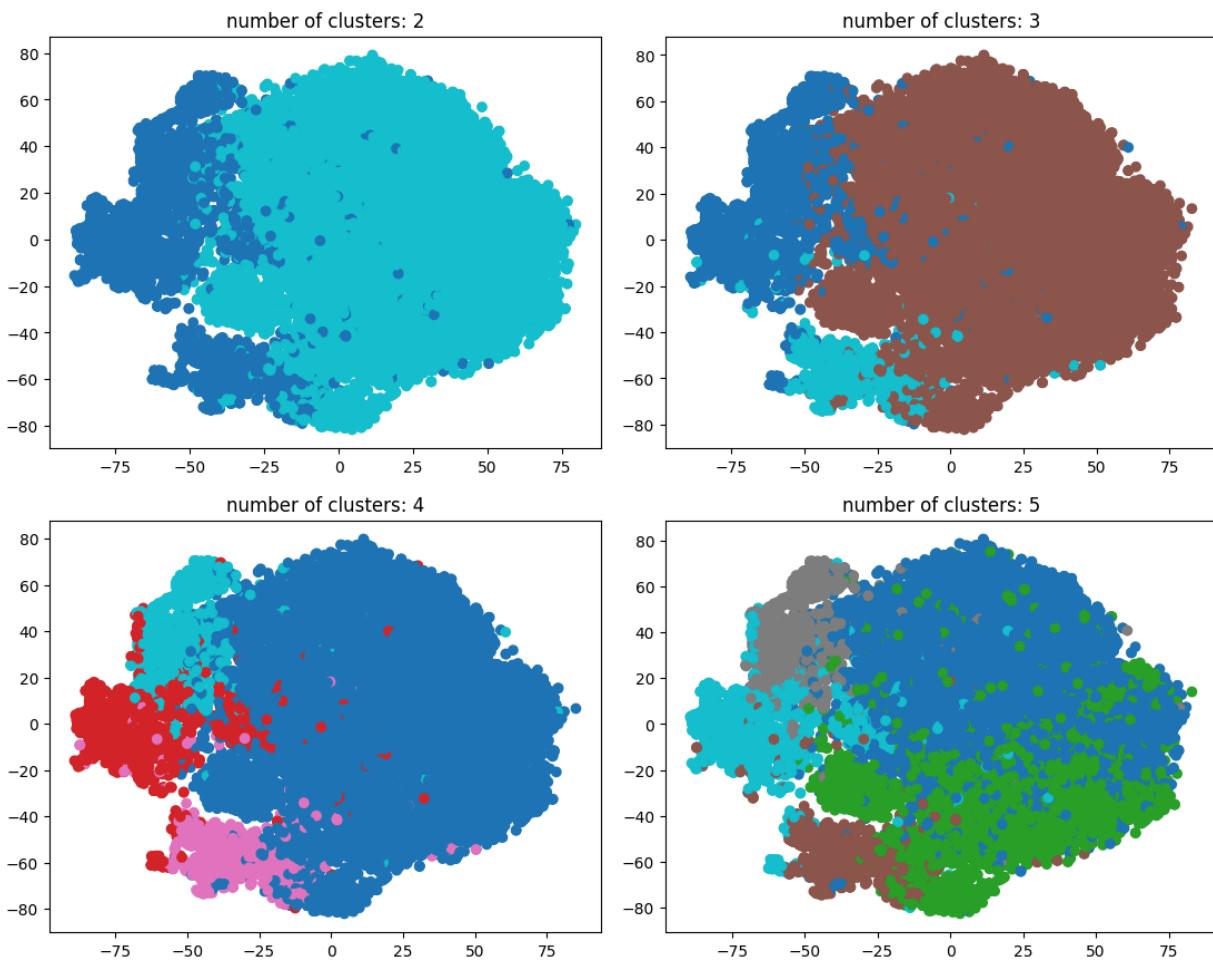
برای مجموعه داده اول:

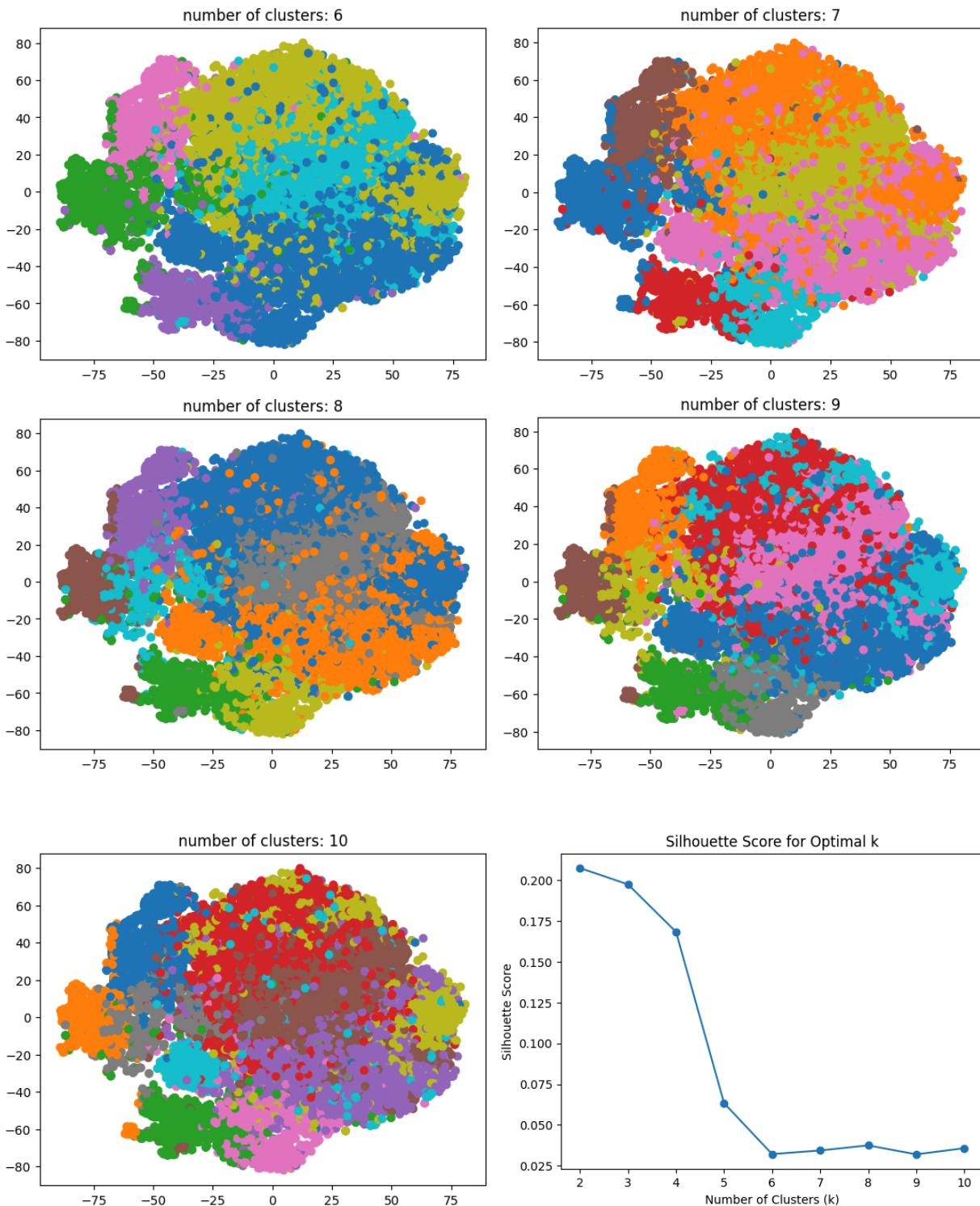




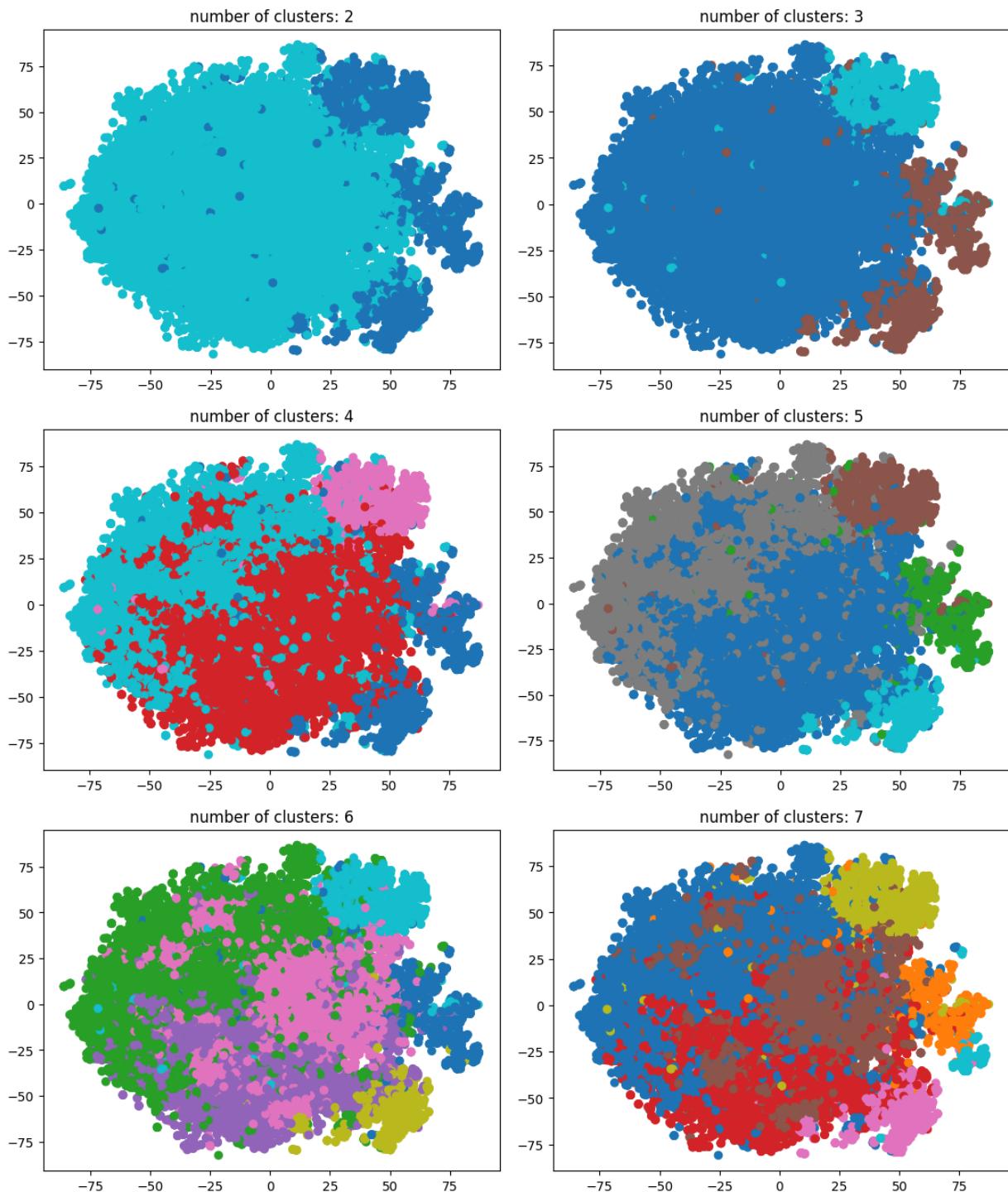


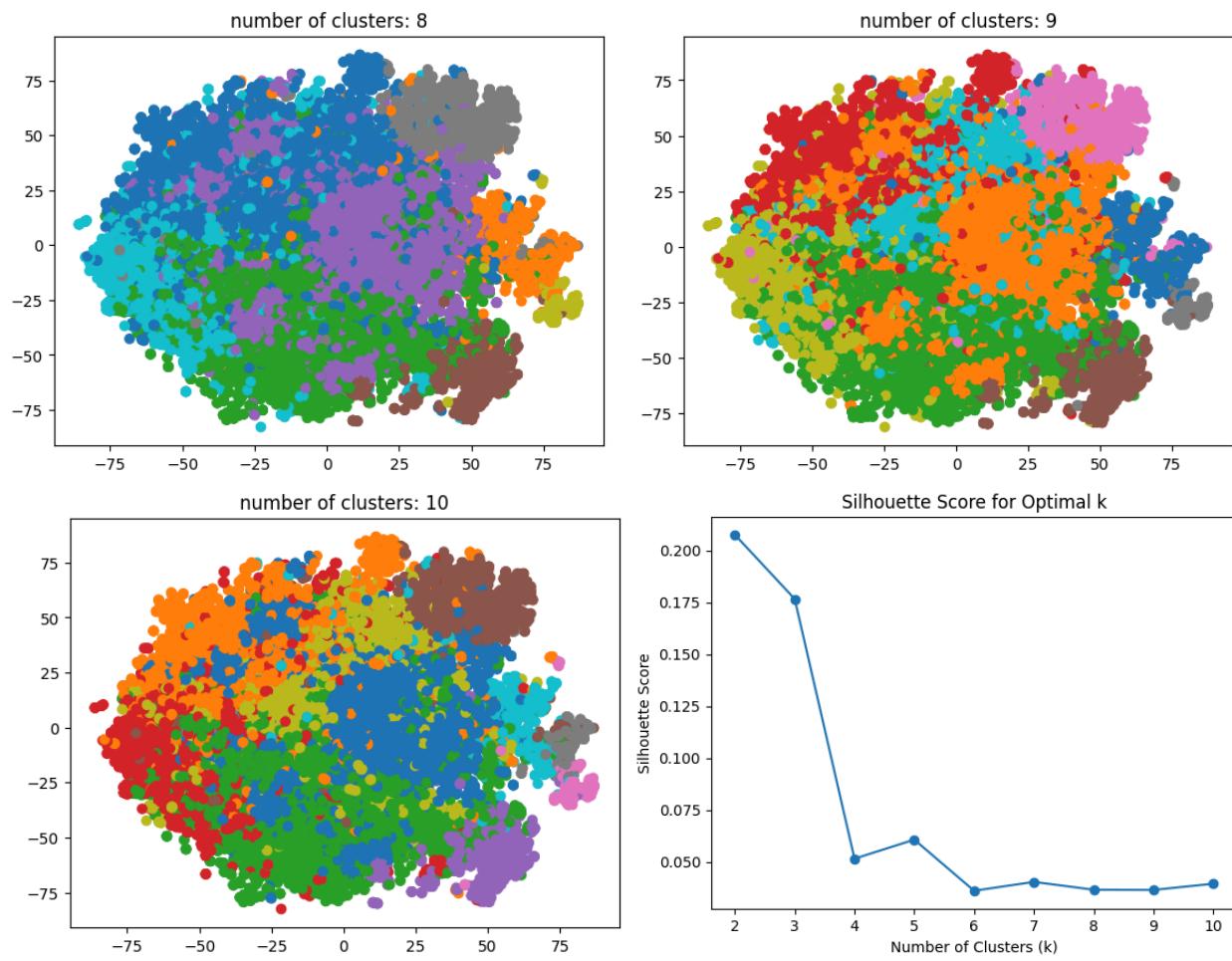
برای مجموعه داده دوم





و برای مجموعه داده سوم





تحلیل:

در هر دو مدل خوش بندی برای داده ها، بهترین تعداد خوش ها دو تا یا سه تا هستند. مقدار silhouette score انتظار می رفت برای 6 بیشترین امتیاز را داشته باشد، اما برای داده هایی کمترین امتیاز را گرفته است.

در روش elbow نیز زانوی قابل توجهی برای تعداد خوش ها دیده نمی شود و عملاً بهترین تعداد خوش پیشنهادی دو تا یا سه تا است.

همچنین تصویر سازی شماره گذاری شده ها نشان می دهد که میان خوش ها همپوشانی قابل توجه مشاهده می شود و یکی از اصلی ترین دلایل عدم تشخیص تعداد درست خوش ها می تواند همین مورد باشد.

برای تحلیل بیشتر این مورد و تحقیق در مورد علت ضعف در تشخیص 6 خوش، می توان گفت که میان ویژگی های نمونه ها همپوشانی قابل توجهی دیده می شود و همچنین تعداد ویژگی ها شاید به اندازه کافی نباشد و لازم باشد تعداد ویژگی ها را افزایش دهیم (به عنوان مثال اگر تعداد ضرایب MFCC و رزولوشن Chroma

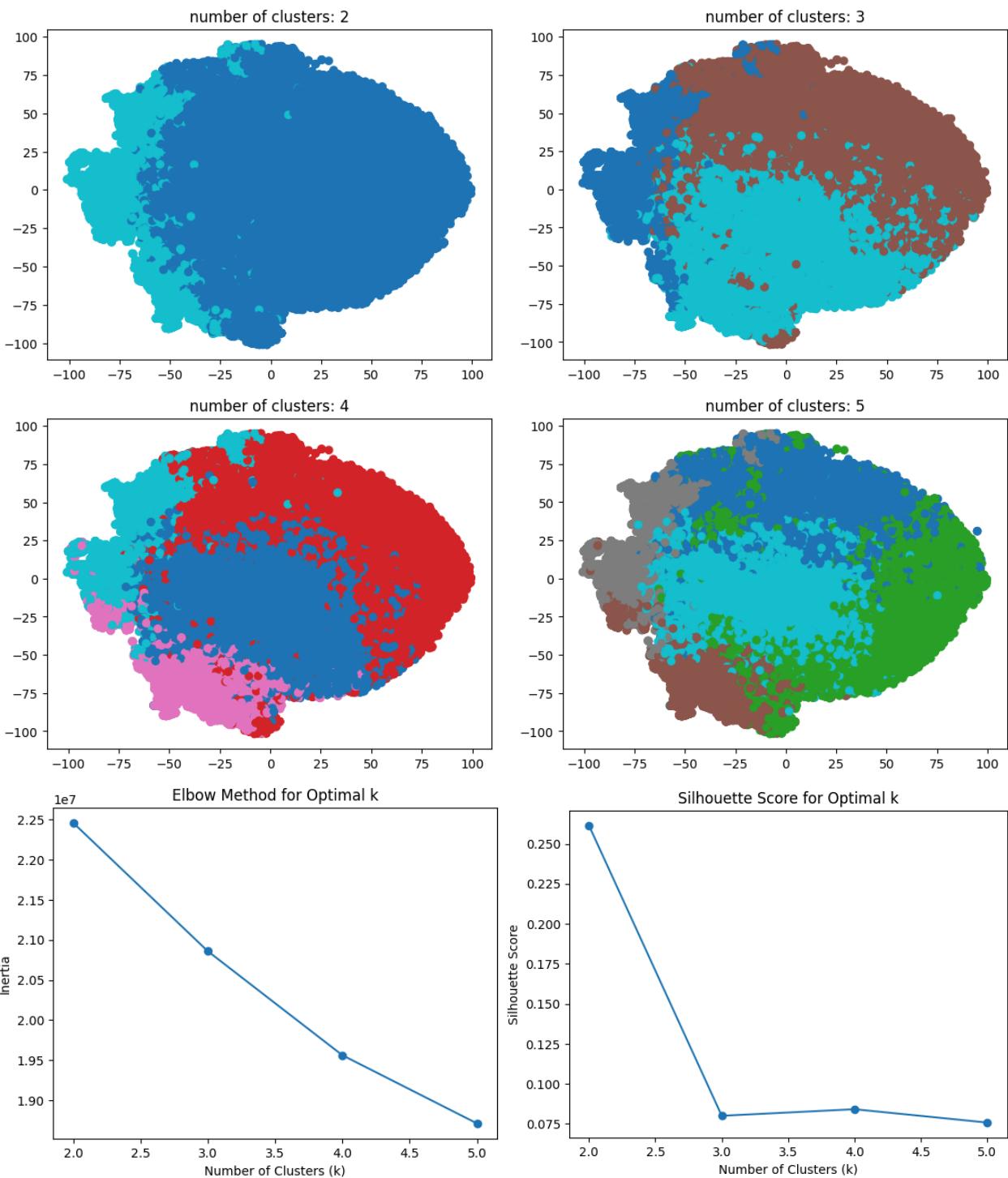
را افزایش دهیم، شاید بهتر می توانستیم تفکیک انجام دهیم اما این موجب افزایش قابل توجه تعداد ویژگی ها و پیچیدگی محاسباتی می شد).

به عنوان مثال در هر فایل صوتی، بخشی از صدا زیر است و بخشی دیگر بم. در clustering قسمت بم همه فایل ها نزدیک یکدیگر قرار می گیرد و عملا به ترتیب نمونه ها توجهی نمی کند. شاید بهتر بود برای تشخیص هویت از یک مدل Recurrent استفاده کرد تا نمونه های پیشین را در حافظه خود نگه داشته باشد.

از DBSCAN به خاطر نزدیکی نمونه ها (به خاطر نرمالیزه کردن ویژگی ها) استفاده نکردیم چون تنظیم کردن eps که حداقل فاصله بین نمونه هاست و n_samples که حداقل تعداد نمونه ها برای اینکه بگوییم یک خوش جدید یافت شده است کار بسیار سخت و زمان بربی بود.

تفکیک جنسیت:

به کمک توابع بخش پیشین، داده های full_data را به تابع clustering_with_kmenas می دهیم (به دلیل محدودیت حافظه رم، قادر به اجرای الگوریتم hierarchical نبودیم). نتایج به شرح زیر است:



در تحلیل این قسمت، ما به دنبال تفکیک جنسیت بودیم و عملاً تنها دو خوشه برای هدفمان کافی بود. همانطور که در دو شکل بالا دیده می‌شود، بهترین تعداد خوشه برای دو تا پیشنهاد می‌شود. در اینجا، با توجه به ویژگی‌هایی که استخراج کرده‌ایم، تحلیل هر کدام از این نمونه‌ها به تنها‌یی برای تشخیص مرد یا زن بودن کافیست. با آزمایش کردن یک ورودی تست و دادن آن به مدل، با یافتن اینکه بیشترین نمونه‌های تست در کدام دسته قرار گرفته‌اند، می‌توانیم مرد یا زن بودن آن را تشخیص دهیم. در قسمت قبل ترتیب این نمونه‌ها حائز اهمیت بود.

به این گونه که هنگامی که فردی در حال صحبت کردن است، بخشی از صدایش ناخودآگاه نازک یا کلفت می‌شود. عملاین خوشه بندی همه قسمت‌های نازک یا کلفت را کنار همیگر می‌بیند.