ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Програмні засоби кластеризації даних з використанням машинного навчання

Кваліфікаційна робота

Виконав:
студент гр. КІУКІ-21-4
Руденко О.Д.

Керівник:
доцент каф. ЕОМ
Сорокін А.Р

Аналіз проблеми

Метою кваліфікаційної роботи є створення програмного інструменту для обробки даних з використанням алгоритмів кластерного аналізу, побудованих на основі штучних нейронних мереж.

Завдання:

- порівняльний аналіз існуючих алгоритмів кластеризації даних за допомогою нейронних мереж;
- програмна реалізація розглянутих алгоритмів;
- застосування методів кластерного аналізу даних для створення груп схожих об’єктів.

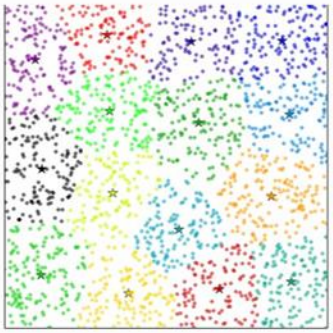
Основні задачі інтелектуального аналізу даних

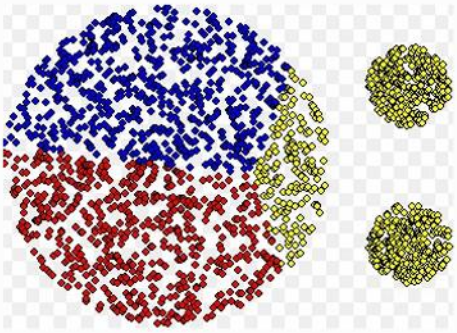
3

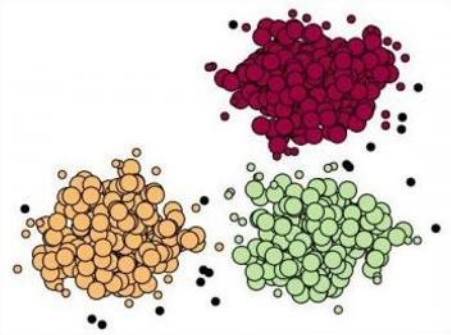
- Класифікація
- Кластеризація
- Асоціація
- Послідовність
- Прогнозування
- Визначення відхилень або викидів
- Оцінювання
- Аналіз зв'язків
- Візуалізація
- Підведення підсумків

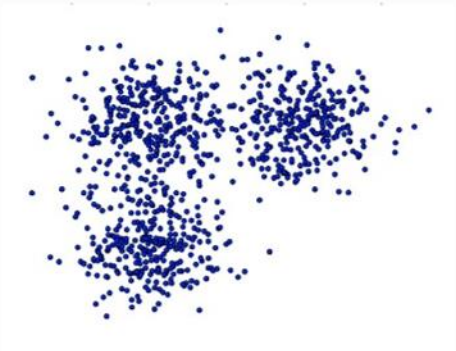
Кластеризація даних

4









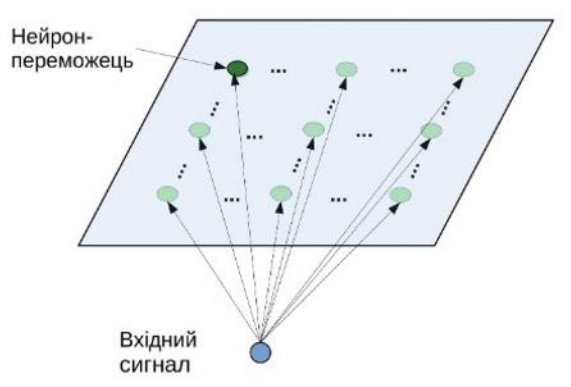
Етапи застосування кластерного аналізу в загальному вигляді

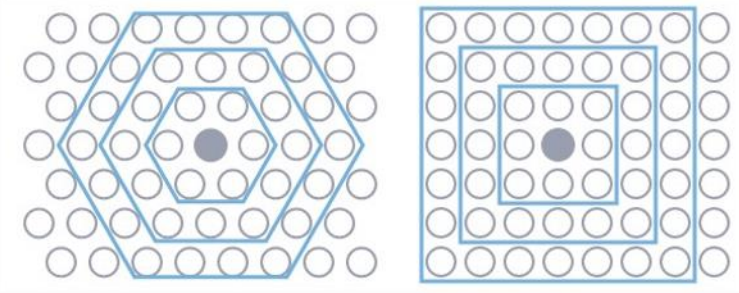
5

- Відбір вибірки об'єктів для кластеризації.
- Визначення безлічі змінних, за якими будуть оцінюватися об'єкти у вибірці. При необхідності - нормалізація значень змінних.
- Обчислення значень міри схожості між об'єктами. Застосування методу кластерного аналізу для створення груп схожих об'єктів (кластерів).
- Представлення результатів аналізу.

Класичний алгоритм навчання карти Кохонена

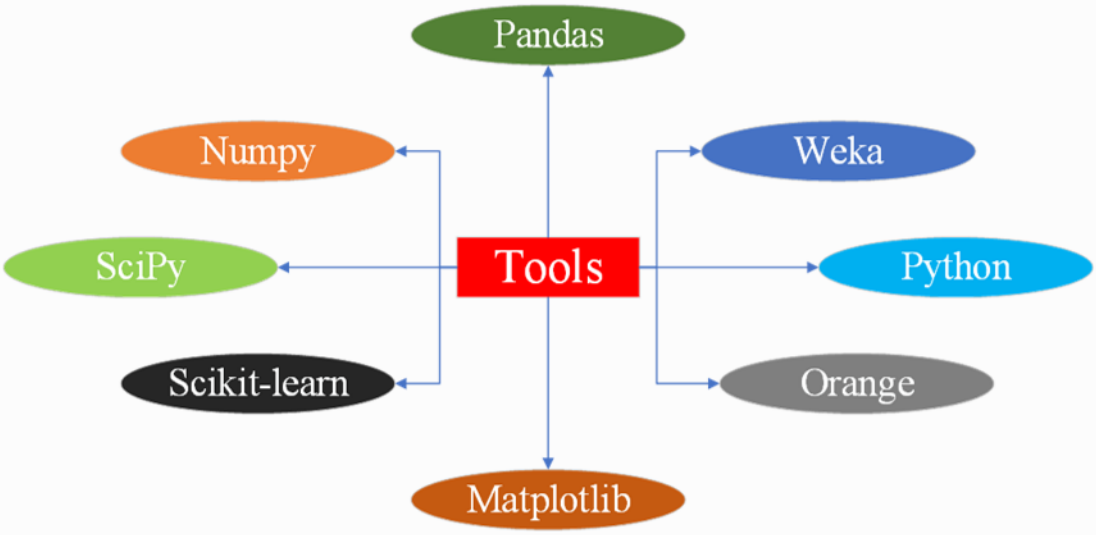
6





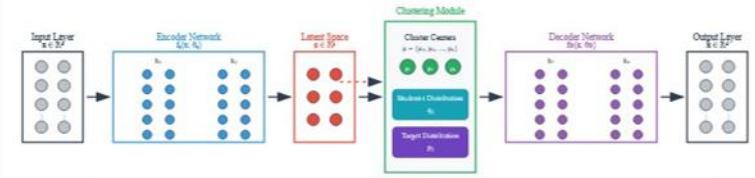
7

Вибір програмних засобів



8

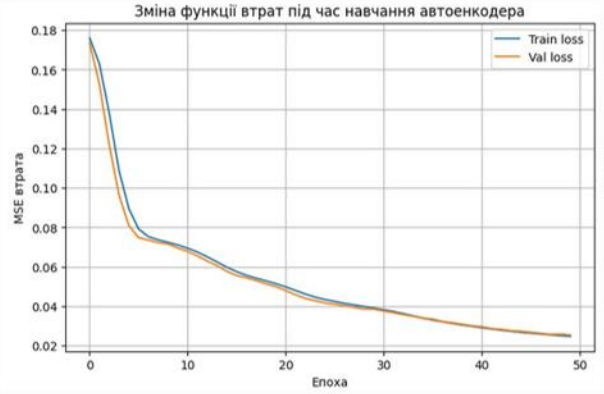
Реалізація в Colab

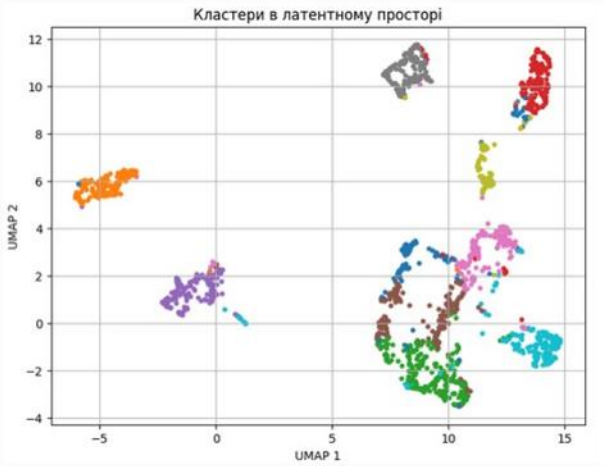



```
# Кластеризація з використанням нейронної мережі та візуалізацій
!pip install umap-learn --quiet

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.metrics import normalized_mutual_info_score, adjusted_rand_score
import tensorflow as tf
from tensorflow.keras import layers, models
import umap

# Завантаження та нормалізація даних
data = load_digits()
X = data.data
y = data.target
X = X / 16.0

# Побудова автоенкодера
input_dim = X.shape[1]
encoding_dim = 10

input_layer = layers.Input(shape=(input_dim,))
encoded = layers.Dense(64, activation='relu')(input_layer)
encoded = layers.Dense(32, activation='relu')(encoded)
latent = layers.Dense(encoding_dim, activation='relu')(encoded)
```

9

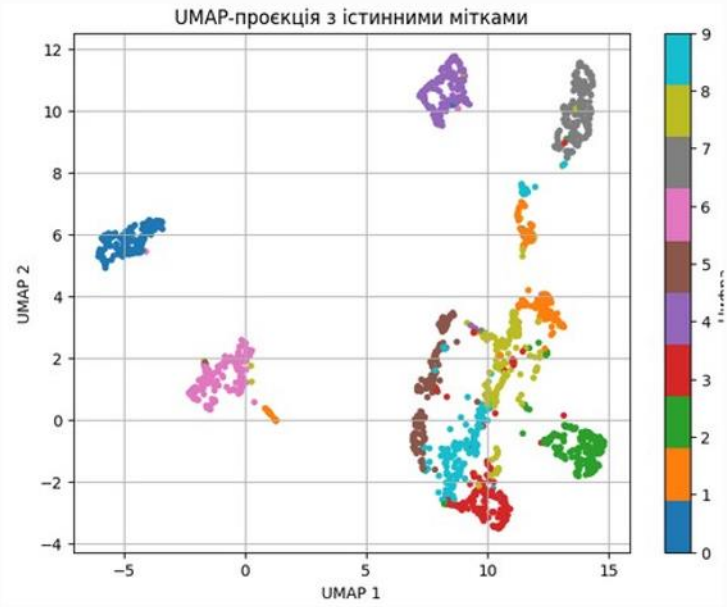
Результати роботи





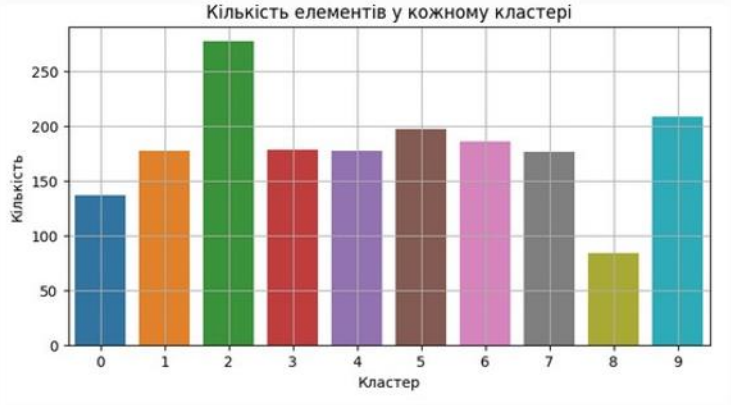
10

Результати роботи



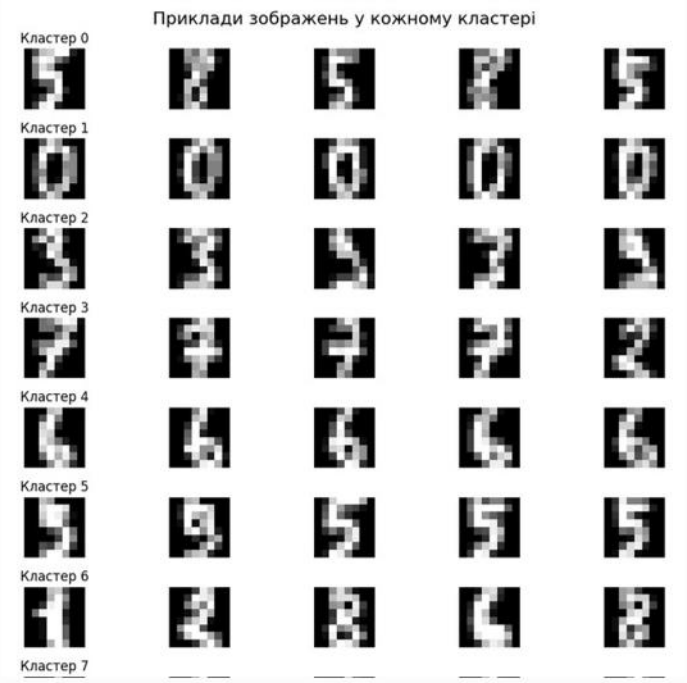
11

Результати роботи



12

Результати роботи



Висновки

13

У результаті виконаної роботи було здійснено глибокий аналіз кластеризації рукописних цифр із використанням автоенкодера, алгоритмів зниження розмірності та методів кластерного аналізу. Автоенкодер успішно навчився відтворювати вхідні зображення, що підтверджується візуальною подібністю між оригіналами та їх реконструкціями, а також зменшенням функції втрат протягом епох навчання. Це свідчить про здатність моделі ефективно кодувати й відновлювати інформацію.

Метод UMAP дозволив візуалізувати дані у двовимірному латентному просторі, де спостерігається чітке формування кластерів. Зіставлення кластерів із істинними мітками показало високий рівень відповідності, що свідчить про ефективність використаних підходів для виявлення прихованих структур у даних. Додатковий аналіз розподілу елементів по кластерах та прикладів зображень у кожному кластері підтвердив, що кластеризація зберігає логічну подібність між елементами всередині кожного кластеру.

Використані програмні інструменти Python, Pandas, NumPy, Matplotlib, Scikit-learn та Orange забезпечили повний цикл обробки, аналізу та візуалізації даних. Отримані результати підтверджують доцільність застосування автоенкодерів у поєднанні з методами зниження розмірності та кластеризації для аналізу складних багатовимірних даних.

ДОДАТОК Б

Програмний код

Б.1 Лістинг коду

```
# Кластеризація з використанням нейронної мережі та візуалізацій
!pip install umap-learn --quiet

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.metrics import normalized_mutual_info_score,
adjusted_rand_score
import tensorflow as tf
from tensorflow.keras import layers, models
import umap

# Завантаження та нормалізація даних
data = load_digits()
X = data.data
y = data.target
X = X / 16.0

# Побудова автоенкодера
input_dim = X.shape[1]
encoding_dim = 10

input_layer = layers.Input(shape=(input_dim,))
encoded = layers.Dense(64, activation='relu')(input_layer)
encoded = layers.Dense(32, activation='relu')(encoded)
latent = layers.Dense(encoding_dim, activation='relu')(encoded)
decoded = layers.Dense(32, activation='relu')(latent)
decoded = layers.Dense(64, activation='relu')(decoded)
output_layer = layers.Dense(input_dim,
activation='sigmoid')(decoded)

autoencoder = models.Model(inputs=input_layer,
outputs=output_layer)
encoder = models.Model(inputs=input_layer, outputs=latent)

autoencoder.compile(optimizer='adam', loss='mse')
history = autoencoder.fit(X, X, epochs=50, batch_size=256,
verbose=0, validation_split=0.1)

# Побудова графіка втрат
```

```
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label='Train loss')
plt.plot(history.history['val_loss'], label='Val loss')
plt.title('Зміна функції втрат під час навчання автоенкодера')
plt.xlabel('Епоха')
plt.ylabel('MSE втрата')
plt.legend()
plt.grid(True)
plt.show()

# Отримання latent space
X_latent = encoder.predict(X)

# Кластеризація за допомогою KMeans
n_clusters = 10
kmeans = KMeans(n_clusters=n_clusters, n_init=20)
y_pred = kmeans.fit_predict(X_latent)

# Візуалізація кластерів у просторі UMAP
X_umap = umap.UMAP(n_neighbors=15,
min_dist=0.1).fit_transform(X_latent)

plt.figure(figsize=(8, 6))
plt.scatter(X_umap[:, 0], X_umap[:, 1], c=y_pred, cmap='tab10',
s=10)
plt.title("Кластери в латентному просторі")
plt.xlabel("UMAP 1")
plt.ylabel("UMAP 2")
plt.grid(True)
plt.show()

# Візуалізація UMAP з істинними мітками
plt.figure(figsize=(8, 6))
plt.scatter(X_umap[:, 0], X_umap[:, 1], c=y, cmap='tab10', s=10)
plt.title("UMAP-проєкція з істинними мітками")
plt.xlabel("UMAP 1")
plt.ylabel("UMAP 2")
plt.colorbar(label="Цифра")
plt.grid(True)
plt.show()

# Візуалізація UMAP з кластерними мітками
plt.figure(figsize=(8, 6))
plt.scatter(X_umap[:, 0], X_umap[:, 1], c=y_pred, cmap='tab10',
s=10)
plt.title("UMAP-проєкція з кластерними мітками")
plt.xlabel("UMAP 1")
plt.ylabel("UMAP 2")
plt.colorbar(label="Кластер")
plt.grid(True)
plt.show()

# Візуалізація реконструкції автоенкодером
```

```
decoded_imgs = autoencoder.predict(X[:10])

plt.figure(figsize=(10, 3))
for i in range(10):
    ax = plt.subplot(2, 10, i + 1)
    plt.imshow(X[i].reshape(8, 8), cmap='gray')
    plt.axis("off")

    ax = plt.subplot(2, 10, i + 11)
    plt.imshow(decoded_imgs[i].reshape(8, 8), cmap='gray')
    plt.axis("off")
plt.suptitle("Оригінали (верх) та реконструкції (низ)")
plt.show()

# Гістограма розподілу по кластерах
plt.figure(figsize=(8, 4))
sns.countplot(x=y_pred, palette='tab10')
plt.title("Кількість елементів у кожному кластері")
plt.xlabel("Кластер")
plt.ylabel("Кількість")
plt.grid(True)
plt.show()

# Приклади зображень у кожному кластері
fig, axes = plt.subplots(n_clusters, 5, figsize=(10, 12))
for cluster in range(n_clusters):
    cluster_idx = np.where(y_pred == cluster)[0][:5]
    for i, idx in enumerate(cluster_idx):
        axes[cluster, i].imshow(X[idx].reshape(8, 8),
cmap='gray')
        axes[cluster, i].axis('off')
        if i == 0:
            axes[cluster, i].set_title(f'Кластер {cluster}')
plt.suptitle("Приклади зображень у кожному кластері",
fontsize=16)
plt.tight_layout()
plt.show()

# Оцінка кластеризації
print("NMI:", normalized_mutual_info_score(y, y_pred))
print("ARI:", adjusted_rand_score(y, y_pred))
```