SPRAWOZDANIE

Mateusz Gawin

292409

I. Wprowadzenie

Celem analizy jest przeprowadzenie hierarchicznej analizy skupień (HCA, ang. Hierarchical Cluster Analysis) dla wybranych związków chemicznych na podstawie ich deskryptorów molekularnych. W analizie wykorzystano metrykę euklidesową oraz dwie metody łączenia klastrów: complete linkage oraz single linkage. Porównano otrzymane dendrogramy i zinterpretowano różnice w wynikach.

II. Dane

Analiza objęła 20 związków chemicznych, którym przypisano identyfikatory numeryczne od 1 do 20. Dla każdego związku wygenerowano deskryptory molekularne za pomocą biblioteki RDKit na podstawie reprezentacji SMILES. Deskryptory stanowią cechy wejściowe dla HCA.

Zbiór danych zawierał 217 deskryptorów, z których 68 przyjmowały stałe wartości — zostały one usunięte. Deskryptory zostały pobrane automatycznie, a dane zapisano w postaci tabeli do analizy.Fragment kodu generującego deskryptory:

```
# - - - przygotowanie danych - - -
descriptorNames =  [desc[0] for desc in Descriptors._descList]
calculator = MoleculeDescriptors.MolecularDescriptorCalculator(descriptorNames)

def Descriptors(mols):
    results = []
    for name, smi in mols.items():
        mol = Chem.MolFromSmiles(smi)
        if mol is None:
            print("Error")
            desc = [None] * len(descriptorNames)
        else:
            desc = calculator.CalcDescriptors(mol)
        results.append([name]+list(desc))
    df = pd.DataFrame(results, columns=["Compound"]+descriptorNames)
    return df

desc_df = Descriptors(molecules)
```

Figure 1

III. Czyszczenie i przetwarzanie danych

Dane zostały oczyszczone poprzez:
- usunięcie kolumn stałej wartości,
- usunięcie brakujących wartości (NaN),
- standaryzację (Z-score) każdej cechy.

Standaryzacja była niezbędna, aby uniknąć dominacji cech o większym zakresie w obliczaniu odległości euklidesowej.

Fragment kodu przetwarzającego dane:

```
def clean_and_scale(df):
    df_cleaned = df.copy()
    if "Compound" in df_cleaned.columns:
        compound_names = df_cleaned["Compound"]
        df_cleaned = df_cleaned.drop(columns=["Compound"])
    else:
        compound_names = pd.Series([f"Mol_{i}" for i in range(len(df_cleaned))])
    df_cleaned = df_cleaned.loc[:, df_cleaned.nunique() > 1]
    df_cleaned = df_cleaned.dropna(axis=1)
    mean = df_cleaned.mean()
    std = df_cleaned.std(ddof=0)
    X_scaled = (df_cleaned - mean) / std
    scaled_df = X_scaled.copy()
    scaled_df.insert(0, "Compound", compound_names.values)
      Mati G, 14 hours ago • ADD
    return scaled_df, X_scaled.to_numpy(), compound_names

scaled_df, X_scaled, compound_names = clean_and_scale(desc_df)
```

Figure 2

IV. Obliczanie macierzy odległości i analiza HCA

Do obliczenia macierzy odległości zastosowano funkcję liczącą miarę euklidesową (Figure 3). Następnie wykonano hierarchiczną analizę skupień z dwiema metodami łączenia klastrów (Figure 4):
- Complete linkage – największa odległość między punktami klastrów,
- Single linkage – najmniejsza odległość między punktami klastrów.

Wybrałem metrykę euklidesową jako najbardziej intuicyjną i często stosowaną w analizach wielowymiarowych. Choć możliwe było również użycie metryk Manhattan lub Czebyszewa, zdecydowałem się ograniczyć analizę do jednej metryki dla przejrzystości porównań między typami łączeń.

Kod funkcji HCA (Figure 5) i obliczeń:

```
# - - - obliczanie danych - - -
def euklidesowa(m):
    n = m.shape[0]
    result = np.zeros((n, n))
    for i in range(n):
        for j in range(i+1, n):
            dist = np.sqrt(np.sum((m[i] - m[j]) ** 2))
            result[i, j] = dist
            result[j, i] = dist
    return pd.DataFrame(result)      Mati G, 2 hours a
```

Figure 3 - odległość Euklidesowa

```
def maxDist(c1, c2, mo): #funkcja potrzebna do liczenia complete linkage
    return np.max([mo[i][j] for i in c1 for j in c2])

def minDist(c1, c2, mo): #funkcja potrzebna do liczenia single linkage
    return np.min([mo[i][j] for i in c1 for j in c2])
```

Figure 4

```python
def HCA(mo, max_min):
    n = mo.shape[0]
    klastry = [[i] for i in range(n)]
    ID_Klastry = list(range(n))
    dis = mo.to_numpy().copy()
    np.fill_diagonal(dis, np.inf)
    connections = []
    nextClusterID = n

    while len(klastry) > 1:
        best = np.inf
        pair = None
        for i in range(len(klastry)):
            for j in range(i + 1, len(klastry)):
                d = max_min(klastry[i], klastry[j], dis)
                if d < best:
                    best = d
                    pair = (i, j)
        i, j = pair
        new_cluster = klastry[i] + klastry[j]
        connections.append([ID_Klastry[i], ID_Klastry[j], best, len(new_cluster)])
        for k in sorted([i, j], reverse=True):
            del klastry[k]
            del ID_Klastry[k]
        klastry.append(new_cluster)
        ID_Klastry.append(nextClusterID)
        nextClusterID += 1
      Mati G, 2 hours ago • ADD
    return np.array(connections)
```

Figure 5

V. Wyniki analizy

Dendrogram – Complete Linkage:

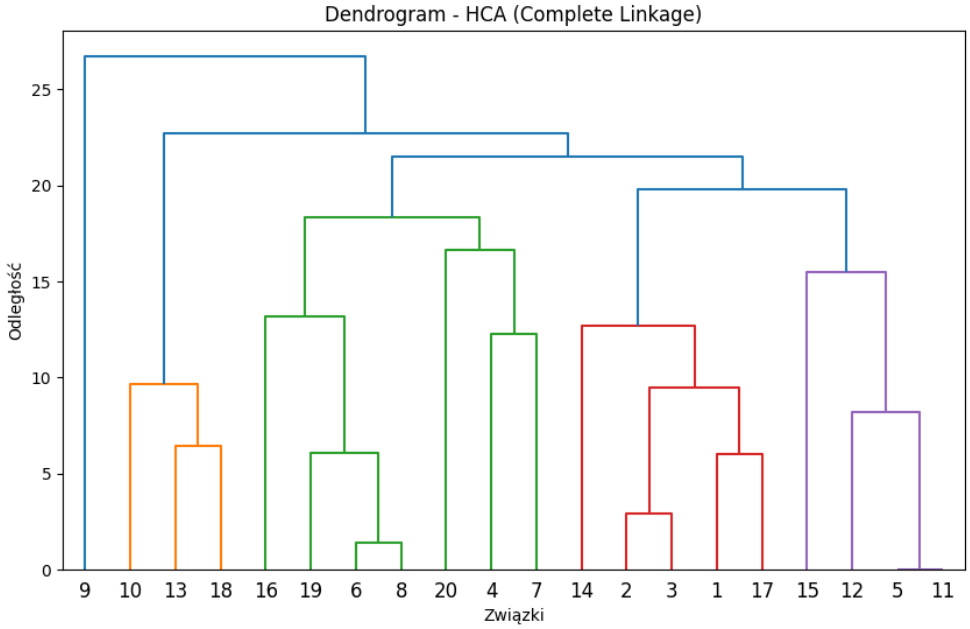

Figure 6

Dendrogram – Single Linkage:

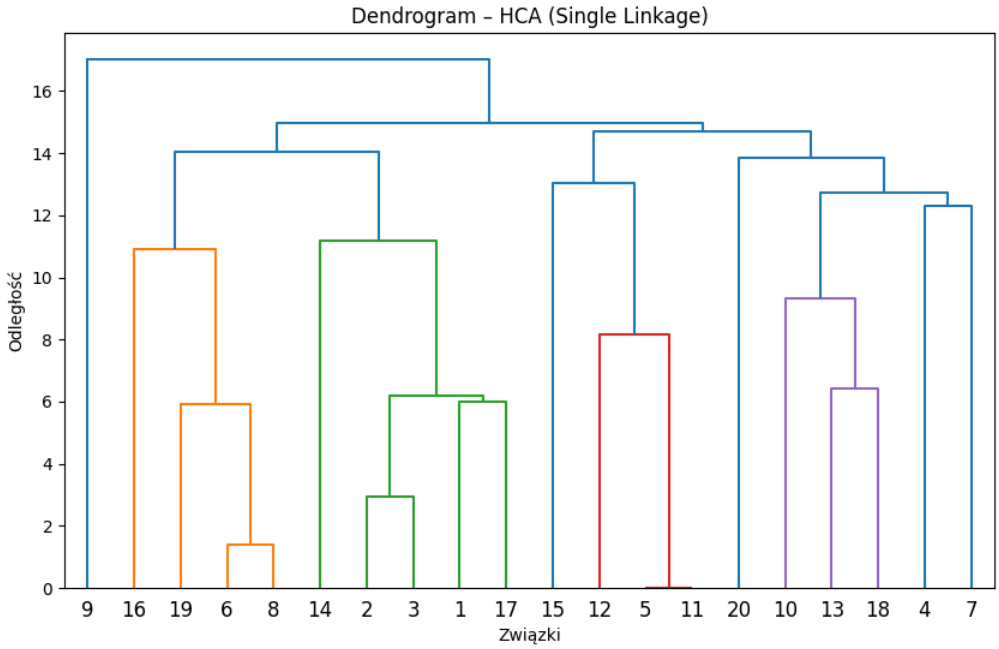

Figure 7

Legenda związków:

1: Mocap
2: Frumin
3: Rampart
4: Thiacloprid
5: DDT
6: Clopyralid
7: Acetamiprid
8: Aminopyralid
9: Cadusafos
10: Clothianidin
11: DDD
12: DDE
13: Imidacloprid
14: Malathion
15: Methoxychlor
16: Picloram
17: Terbufos
18: Thiamethoxam
19: Triclopyr
20: Veltin

VI. Wnioski i interpretacja

Analiza dendrogramów pozwala na ocenę podobieństw chemicznych pomiędzy badanymi związkami. Na podstawie przecięcia dendrogramu na poziomie odległości ~10 można wyróżnić 4–5 głównych klastrów dla complete linkage i 3 dla single linkage.

- W przypadku metody *complete linkage* widać, że grupy tworzą się dopiero przy wyższych wartościach odległości, co sugeruje, że tylko bardzo podobne chemicznie związki są łączone wcześnie. Taka metoda daje zwarte, silnie powiązane klastry. Przykładowo, związki 1 (Mocap), 2 (Frumin) i 3 (Rampart) są łączone relatywnie wcześnie, co wskazuje na ich dużą zgodność w wybranych deskryptorach.

- Dendrogram wygenerowany metodą *single linkage* pokazuje z kolei szybkie łączenie wielu punktów przez najbliższe sąsiedztwo – może prowadzić do wydłużonych, mniej jednorodnych klastrów. Tutaj grupa zawierająca związki 17–20 tworzy się wcześniej, mimo że niektóre z nich mogą być chemicznie mniej zbliżone.

- Związek 9 (Cadusafos) zarówno w complete, jak i w single linkage łączy się z innymi bardzo późno, co może sugerować jego unikalny charakter chemiczny.

- Związek 13 (Imidacloprid) w obu dendrogramach znajduje się w grupie związków 10 (Clothianidin) i 18 (Thiamethoxam), co może świadczyć o podobnym mechanizmie działania lub strukturze molekularnej (np. obecność grup nitrowych czy heterocykli).

- Widoczne są też pary silnie powiązane, np. DDT (5) i DDD (11), a także DDD (11) i DDE (12), które różnią się tylko drobnymi modyfikacjami strukturalnymi i faktycznie tworzą jedną grupę w obu podejściach.

Wnioski potwierdzają, że HCA umożliwia rozpoznanie klastrów chemicznie spójnych i odróżnienie związków odstających, przy czym sposób łączenia klastrów (linkage) ma istotny wpływ na końcowy podział.

Dodatkowo warto zauważyć, że algorytm single linkage może być bardziej podatny na „efekt łańcucha", gdzie dalsze elementy są dołączane przez bliskość do pojedynczego punktu, co może zaniżać interpretacyjną jakość klastrów.