# MDM-2025 Homework 4

Mert Dagli (100830277)
Tuomas Mäkinen (103382573)
Maja Sellmer (103396682)

November 24, 2025

# 1 Methods

The task was to explore associations between molecule substructures and anticancer activities using frequent subgraph mining. To this end, we randomly selected one molecule(number 14) as a test sample and then mined frequent subgraphs of the rest molecules using the software *MoSS*. We searched for significant rules and used them to predict anticancer activities of the test molecule. We completed the assignment in Python, using the following libraries:

- `matlplotlib` for plotting
- `pandas` to handle the given data as a dataframe
- `math` for calculations
- `sklearn.metrics` for mutual information score
- `rdkit.Chem` for drawing molecules

## 1.1 Frequent subgraph mining

Firstly, we identified substructures in the molecules with the *MoSS* software. In order to obtain interesting frequent substructures, we set the minimum size to 6 and minimum support to 12. Then we identified how common each substructure is. We computed the frequency of a structure as the number of molecules containing this substructure divided by the total number of molecules, and visualised the top 5 most frequent ones.

## 1.2 Significant rule search

Next, for each of the three types of cancer, we calculated the mutual information for rules of the form $X \to C$, where $X$ represents the co-occurrence of some molecular substructures and $C$ refers to the positive anticancer activity against that cancer. We only considered sets $X$ with at most two elements, i.e. single substructures or pairs of two substructures. We only keep rules which are positively associated with anticancer activity, i.e. $P(C = c \mid X) > P(C = c)$ and kept those with mutual information higher than the threshold of 0.01.

## 1.3 Selection of representative rules

After identifying the significant rules, we tried to remove overfitting rules. We selected a representative rule set using confidence (rule $X \to C$ is removed if there exists $Y \subset X$ such that $P(C = c \mid Y) \geq P(C = c \mid X)$) and conditional mutual

information (a rule is removed if the conditional mutual information falls under the threshold of 0.01).

### 1.4 Prediction

Lastly, we checked which substructures our test molecule contains and thus which rules apply. We computed the average confidence, average MI, average MI*confidence, average lift, and average leverage of these rules. Based on these goodness measures, we predict whether there is an anticancer activity against each cancer type.

## 2 Results

### 2.1 Frequent substructures

Figure 1 shows the give most frequent molecular substructures.

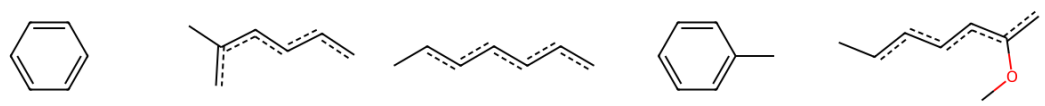


Figure 1: Top 5 substructures

### 2.2 Significant and representative rules

Table 1 shows a summary of the rules we obtained for each cancer type. We can see that the pool of possible rules was significantly narrowed down for types 2 and 3, but it seems the threshold was too low for type 1.

| **Type, Rules** | **# rules** | **Avg freq** | **Avg conf** | **Avg MI** |
|---|---|---|---|---|
| Type 1, Significant | 1680 | 0.242 | 0.896 | 0.328 |
| Type 1, Representative | 482 | 0.275 | 0.957 | 0.489 |
| Type 2, Significant | 1475 | 0.122 | 0.947 | 0.138 |
| Type 2, Representative | 91 | 0.198 | 0.910 | 0.185 |
| Type 3, Significant | 1687 | 0.115 | 0.992 | 0.073 |
| Type 3, Representative | 60 | 0.214 | 0.976 | 0.130 |

Table 1: Summary statistics of significant and representative rules

### 2.3 Predictions

Table 2 shows the average of each goodness measure for the rules applicable to the test molecule. We can see that type 1 seems very likely, although it is not true. If we base our predictions on the lift and leverage, we would incorrectly predict 1 to be true, correctly predict 2 to be true, and incorrectly predict 3 to be false. We could classify correctly by using a confidence threshold of 0.915, but this would have had to be a very lucky guess. It seems the average confidence of the rules we found is overall too high.

| Type | Confidence | MI | MI × Conf | Lift | Leverage | True? |
|---|---|---|---|---|---|---|
| Type 1 | 0.913 | 0.337 | 0.314 | 2.624 | 0.565 | 0 |
| Type 2 | 0.918 | 0.051 | 0.045 | 2.483 | 0.548 | 1 |
| Type 3 | 0.993 | 0.079 | 0.078 | 1.522 | 0.341 | 1 |

Table 2: Predictions of anticancer activities for the test molecule

## 3 Conclusions

To summarize, we identified frequent substructures in the molecules, and searched for rules telling us that the presence of some substructure $X$, or a pair of substructures $(X_1, X_2)$ implies anticancer activity against cancer type $n$. We narrowed down possible rules using confidence, mutual information and conditional mutual information. This allowed us to make predictions about anticancer activites of a test molecule not included in the training data, however we were not really able to predict all three types correctly based on the rules we obtained.

## 4 Appendix – Code

```
import os, sys, random, itertools, math
import pandas as pd
import numpy as np
from sklearn.metrics import mutual_info_score
from rdkit import Chem
from rdkit.Chem import Draw
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import ast

#____________________Prep____________________#

MOLECULE_CSV = 'molecule.csv'
MOSS_IDENTIFIERS = f"newdata/moss_identifiers.csv"
MOSS_SUBSTRUCTURES = f"newdata/moss_substructures.csv"
molecule_df = pd.read_csv(MOLECULE_CSV, sep=";")
```

```
# Excluded Test Molecule
test_mol = molecule_df.iloc[14]
molecule_df = molecule_df.drop([14])

with open("molecule_moss.smile", "w") as f:
    for i, row in molecule_df.iterrows():
        mol_id = f"{i+1}"
        smiles = row["SMILES"]
        f.write(f"{mol_id},0,{smiles}\n")

#_____________________A_____________________#

molecule_df = pd.read_csv(MOLECULE_CSV, sep=";")
# Excluded Molecule
test_mol = molecule_df.iloc[14]
molecule_df = molecule_df.drop([14])

moss_df = pd.read_csv(MOSS_SUBSTRUCTURES, sep=",")
moss_df["mol_ids"] = pd.read_csv(MOSS_IDENTIFIERS, sep=";")['list'].apply(lambda x
    : [int(i) for i in ast.literal_eval(x)])


cleaned_moss_df = moss_df.drop_duplicates(subset=['description']).copy().
    reset_index()
cleaned_moss_df.rename(columns={'description': 'substructure'}, inplace=True)
cleaned_moss_df["frq"] = (cleaned_moss_df["mol_ids"].apply(len) / molecule_df.
    shape[0])

print(f"Original size: {moss_df.shape[0]}")
print(f"Cleaned size: {cleaned_moss_df.shape[0]}")
print(f"Removed {len(moss_df) - len(cleaned_moss_df)} duplicates")   #NOTE! Idk
    why there are duplicates. Use the cleaned set. (type 3 is the worst, others
    don't lose much)

sorted_df = cleaned_moss_df.sort_values("frq", ascending=False).head(5).drop(
    columns=["nodes","edges","s_abs","s_rel","c_abs","c_rel","id"])
top5 = sorted_df.head(5)
top5_mols = []
test = [0,0,0,0,0]
for sml in top5["substructure"]:
    mol = Chem.MolFromSmiles(sml, sanitize=False)
    top5_mols.append(mol)

for sml in molecule_df["SMILES"]:
    mol = Chem.MolFromSmiles(sml)
    for i,q_mol in enumerate(top5_mols):
        if mol.HasSubstructMatch(q_mol):
            test[i] += 1

img = Draw.MolsToGridImage(top5_mols, molsPerRow=5, subImgSize=(250,250),
    returnPNG=False)
img.save(f"results/top5_substructures.png")
print(test)

#_____________________B_____________________#

molecule_df["id"] = list(molecule_df.index.values +1)
mis = []

def mutual_information_binary(X, C):
    X = np.asarray(X)
```

```
    C = np.asarray(C)

    P11 = np.mean((X == 1) & (C == 1))
    P10 = np.mean((X == 1) & (C == 0))
    P01 = np.mean((X == 0) & (C == 1))
    P00 = np.mean((X == 0) & (C == 0))

    PX1 = P11 + P10
    PX0 = P01 + P00
    PC1 = P11 + P01
    PC0 = P10 + P00

    def calc(Pxy, Px, Pc):
        if Pxy == 0 or Px == 0 or Pc == 0:
            return 0
        return Pxy * np.log2(Pxy / (Px * Pc))

    MI = (calc(P11, PX1, PC1) +calc(P10, PX1, PC0) +calc(P01, PX0, PC1) +calc(P00,
         PX0, PC0))

    return MI

# CHOOSE CANCER TYPE n = 1, 2, 3
n = 3

MI_threshold = 0.01
MIC_threshold = 0.01

molecule_df["id"] = list(molecule_df.index.values + 1)

significant_MI_ids = []
significant_mis = []
significant_supports = []
significant_P_C_given_X = []

rep_MI_ids = []
rep_mis = []
rep_supports = []
rep_P_C_given_X = []

C = molecule_df[f'anti_cancer_{n}'].values
P_C_global = C.mean()

for idx1, row1 in cleaned_moss_df.iterrows():
    for idx2, row2 in cleaned_moss_df.iterrows():
        if idx1 > idx2:
            continue
        if idx1 == idx2:
            pair_id = f"{row1['id']}"
        else:
            pair_id = f"{row1['id']}&{row2['id']}"

        X1 = molecule_df["id"].isin(row1['mol_ids']).values
        X2 = molecule_df["id"].isin(row2['mol_ids']).values
        X = X1 & X2

        if np.sum(X) == 0:
            continue

        P_C_X = C[X].mean()
        if P_C_X <= P_C_global:
            continue
```

```
        MI = mutual_information_binary(X, C)
        if MI < MI_threshold:
            continue

        significant_MI_ids.append(pair_id)
        significant_mis.append(MI)
        significant_supports.append(X.mean())
        significant_P_C_given_X.append(P_C_X)

        if idx1 != idx2:
            P_C_X1 = C[X1].mean()
            P_C_X2 = C[X2].mean()

            if P_C_X <= max(P_C_X1, P_C_X2):
                continue

            MI_Y1 = mutual_information_binary(X1, C)
            MI_Y2 = mutual_information_binary(X2, C)
            MIC = MI - max(MI_Y1, MI_Y2)
            if MIC <= MIC_threshold:
                continue

        rep_MI_ids.append(pair_id)
        rep_mis.append(MI)
        rep_supports.append(X.mean())
        rep_P_C_given_X.append(P_C_X)

sig_df = pd.DataFrame({
    "id": significant_MI_ids,
    f"MI_anti_cancer_{n}": significant_mis,
    "support": significant_supports,
    "P_C_given_X": significant_P_C_given_X
})

rep_df = pd.DataFrame({
    "id": rep_MI_ids,
    f"MI_anti_cancer_{n}": rep_mis,
    "support": rep_supports,
    "P_C_given_X": rep_P_C_given_X
})

print("Significant rules summary")
print(sig_df.head())
stats = {
    'num_rules': len(sig_df),
    'avg_support': sig_df['support'].mean(),
    'avg_confidence': sig_df['P_C_given_X'].mean(),
    'avg_MI': sig_df[f'MI_anti_cancer_{n}'].mean(),
}
print(stats)

print("Representative rules summary")
print(rep_df.head())
stats = {
    'num_rules': len(rep_df),
    'avg_support': rep_df['support'].mean(),
    'avg_confidence': rep_df['P_C_given_X'].mean(),
    'avg_MI': rep_df[f'MI_anti_cancer_{n}'].mean(),
}
print(stats)
```

```
mol = Chem.MolFromSmiles(test_mol["SMILES"])
test_substructures = []
for i,q_mol_smile in enumerate(cleaned_moss_df["substructure"]):
    q_mol =  Chem.MolFromSmiles(q_mol_smile, sanitize=False)
    if mol.HasSubstructMatch(q_mol):
            test_substructures.append(i +1)
print(test_substructures)

selected_rules = []

for idx, row in rep_df.iterrows():
    rule_id = row['id']
    sub_ids = list(map(int, rule_id.split('&')))
    if all(s in test_substructures for s in sub_ids):
        selected_rules.append(row)

selected_rules_df = pd.DataFrame(selected_rules)
print(selected_rules_df)

C = molecule_df[f'anti_cancer_{n}'].values
P_C_global = np.mean(C)

selected_rules_df['MI_conf'] = selected_rules_df[f'MI_anti_cancer_{n}'] *
    selected_rules_df['P_C_given_X']
selected_rules_df['lift'] = selected_rules_df['P_C_given_X'] / P_C_global
selected_rules_df['leverage'] = selected_rules_df['P_C_given_X'] - P_C_global

stats = {
    'avg_confidence': selected_rules_df['P_C_given_X'].mean(),
    'avg_MI': selected_rules_df[f'MI_anti_cancer_{n}'].mean(),
    'avg_MI_conf': selected_rules_df['MI_conf'].mean(),
    'avg_lift': selected_rules_df['lift'].mean(),
    'avg_leverage': selected_rules_df['leverage'].mean()
}

print(stats)

predicted_activity = 1 if stats['avg_confidence'] > 0.85 else 0
true_activity = test_mol[f'anti_cancer_{n}']
print(f"Predicted: {predicted_activity}, True: {true_activity}")
```

# 5 Bibliography

# References

[1] Charu C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015.