**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ІВАНА ФРАНКА**

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра інформаційних систем

(повна назва кафедри)

**КУРСОВА РОБОТА**

на тему:

Розробка системи кластеризації антитіл на основі коефіцієнту перехресного зв’язування

Студента 1 курсу, групи ПМІМ-52,

напряму підготовки Комп’ютерні науки

Зелінського Олександра

(прізвище та ініціали)

Керівник доцент, кандидат фізико-математичних наук, Горлач В.М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Львів – 2022

**Зміст**

Вступ 3

Розділ 1. Теоретичні відомості 4

Розділ 2. Опис даних та алгоритму 6

Розділ 3. Практична реалізація 9

Розділ 4. Результати 10

Висновки 14

Список використаних джерел 15

Додатки 16

Додаток А. Код який відповідає за кластеризацію 16

Додаток Б. Код який відповідає за збереження результатів в Excel 18

# Вступ

В сучасному світі поширення комп’ютери та інша потужна обчислювальна техніка набули величезної популярності в житті кожної людини. Саме тому зручним та важливим є використання комп’ютерних алгоритмів та моделей для виконання завдань пов’язаних з дослідженнями у різних сферах науки та техніки. Використання математичних моделей суттєво зменшує кількість рутинної тривалої та дорогої роботи в лабораторіях для хіміків, фізиків, біологів та інших науковців. Тому з часом математичні та комп’ютерні моделі почали зменшувати кількість експеримент.

В умовах нещодавньої пандемії вірусу Covid-19 надзвичайно важливими були швидкі дослідження які несли безпосередню користь для виявлення, запобігання та лікування вірусних захворювань, в тому числі вірусу Covid-19 або SARS-CoV-2. Саме тому в цій роботі буде розглянуто задачу, яка допоможе в знаходженні двох оптимальних антитіл до будь-якого вірусу (для прикладу взято SARS-CoV-2) та можливий спосіб її вирішення за допомогою алгоритму кластеризації.

# Розділ 1. Теоретичні відомості

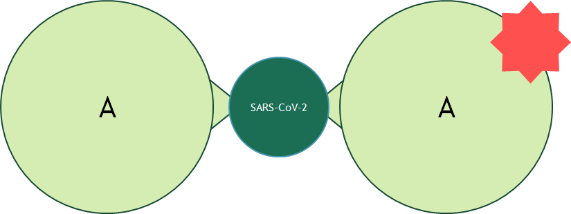
Вірус(*virus* – отрута) – неклітинний інфекційний агент, який може відтворюватися лише всередині живих клітин.

Антитіла, або імуноглобуліни (Ig) – білкові сполуки, які організм хребетних тварин (в тому числі людей) виробляє у відповідь на антигени, чужорідні речовини, що потрапляють до крові, лімфи або тканин організму, з метою знищити або нейтралізувати потенційно небезпечні з них – бактерії, віруси, отрути та деякі інші речовини.

У поставленій задачі дано молекулу вірусу SARS-CoV-2, до якої приєднуються два антитіла (вони можуть бути як різними так і однаковими), для того щоб можна було їх відрізнити одне з них помічається \*.

Для простоти вважатимемо, що все відбувається на площині, антитіла це два круги однакового розміру, що мають невеликий „дзьоб” для взаємодії з вірусом. Антитіла в свою чергу приєднуються до меншого круга який представляє молекулу вірусу. Схематичне зображення цього процесу можна побачити на рисунку 1.

У нашому випадку конкуренція йде між антитілами з молярною вагою180 кД за зв’язування з вірусом з молярною вагою 45 кД. З цієї причини відбувається запекла конкуренція.

  
Рис. 1. Модель приєднання антитіл до вірусної молекули

Основне завдання полягає у знаходженні двох антитіл (можуть бути однаковими), таких що знаходяться на оптимальній відстані одне від одного тобто не перетинаються та не знаходяться занадто близько один до одного, щоб почати конкурувати між собою.

Зручним теоретичним методом вирішення цієї проблеми є розбиття списку антитіл на групи за ознакою того наскільки вони заважають один одному або іншими словами чи приєднуються вони до вірусу в одній і тій же області. Те що антитіла належать до різних груп означає, що вони прив’язуються в різних областях та взаємодіють краще ніж якби вони були з однієї.

В сучасних комп’ютерних науках популярним способом для розбиття даних на групи є група алгоритмів, яка називається кластерним аналізом.

Кластерний аналіз (англ. *Data clustering*) – задача розбиття заданої вибірки об'єктів (ситуацій) на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об’єктів, а об’єкти різних кластерів істотно відрізнялися. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя.

# Розділ 2. Опис даних та алгоритму

Дані з експерименту подані у вигляді таблиці, де кожна комірка це коефіцієнт перехресного зв’язування міченого антитіла (зі стовпця) та не міченого (з рядка). В рядку позначеному як “blank” надані максимальні значення коефіцієнтів перехресного зв’язування для відповідного міченого антитіла.

Зображення, що містить стіл

Автоматично згенерований опис  
Рис. 2. Фрагмент початкових даних

Для подальшої роботи з даними їх позначають за наступним алгоритмом:

* Цифрою 3 (темно-зеленим кольором) антитіла з поганим зв’язуванням, якщо
* Цифрою 2 (світло-зеленим кольором) антитіла з середнім зв’язуванням, якщо
* Цифрою 1 (білим кольором) антитіла з хорошим зв’язування, у всіх інших випадках.

Зображення, що містить текст, шафа

Автоматично згенерований опис  
Рис. 3. Фрагмент позначених даних

Тепер задача полягає у розбитті матриці перехресного зв’язування антитіл на групи за ознакою подібності раніше створеного показника зв’язування для полегшення виявлення оптимальних пар та приблизної локалізації місця зв’язування. Для цього використовують методи кластеризації, а саме модифікація методу k-means для якісних змінних що називається k-modes.

k-means (k-середніх) – це популярний алгоритм кластеризації на основі центроїдів, який розділяє дані представлені у вигляді точок на k кластерів, кожен із яких має майже рівну кількість цих точок. Ідея цього алгоритму кластеризації полягає в тому, щоб знайти k центроїд, де кожна точка з набору даних буде належати будь-якій з k-множин з найближчим (найчастіше мінімальна евклідова відстань) до нього середнім значенням.

k-modes метод, розроблений Хуагном в 1998 році, що визначає кластери на основі відповідності категорій між точками даних. В цьому алгоритмі:

1. відстань між двома точками (проста міра не схожості) даних X та Y описується як сума не схожих елементів:

де

1. Середнє змінюється на моду
2. Мода знаходиться на основі частоти

Нижче наведено кроки для кластеризації на основі k-modes:

1. Виберіть k початкових мод
2. Розділіть елементи на кластери на основі простої міри несхожості відносно початкових мод. Оновлюйте моду кожного з кластерів після додавання нового елементу.
3. Після того, як усі елементи були віднесені до кластера, перевірте значення несхожості кожного спостереження з модою. Якщо виявляється, що найближча мода знаходиться в іншому кластері, перемістіть елемент у відповідний кластер і оновіть моду обох кластерів.
4. Повторюйте крок 3, доки жоден із елементів не змінить кластер на інший

Метод k-modes працює лише при початкові відомій кількості кластерів, оскільки вона нам не відома, то для визначення оптимальної кількості кластерів використовується ліктьовий метод (elbow method), який для різних значень k буде вибирати значення k у тій точці, де значення істотно не зменшується зі збільшенням значення k. Метод модифікований, для щоб використовувати різницю всередині кластера (within-cluster difference).

де WCD – різниця всередині кластера, k – кількість кластерів, m – кількість спостережень у кожному кластері, c – центроїд кластера, а – проста міра несхожості.

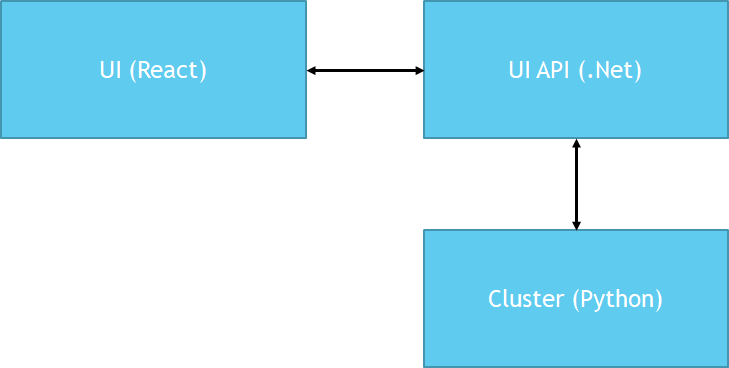
У якості початкового розбиття на кластери використовується алгоритм розроблений Fuyuan Cao в 2009 році. Цей метод ініціалізації для категоріальних даних, у якому враховують відстань між об’єктами та щільність об’єкта, що визначається на основі частоти значень атрибутів.

# Розділ 3. Практична реалізація

Веб застосунок розроблений в результаті виконання курсової роботи складається з трьох частин:

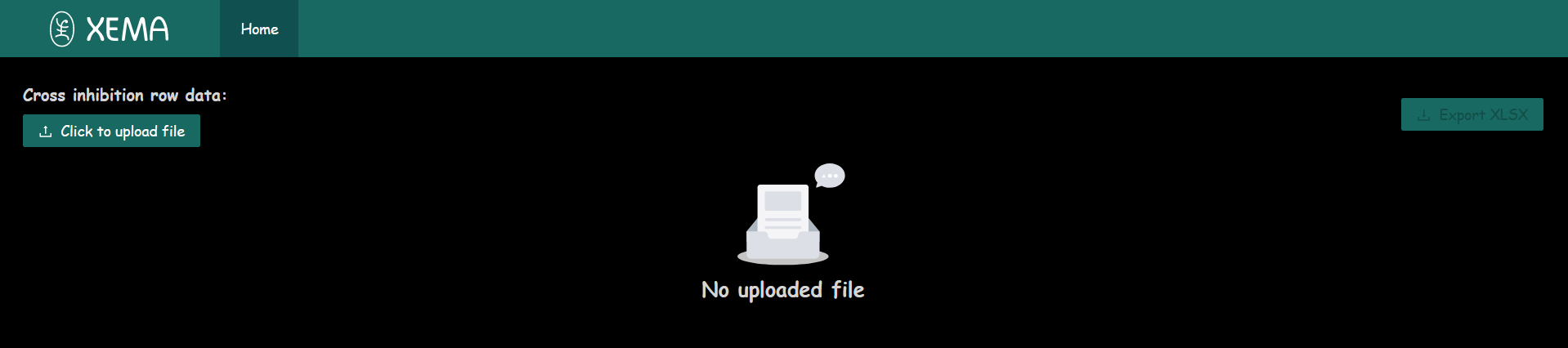
1. Інтерфейс користувача, написаний на мові програмування TypeScript з використанням бібліотеки React, який реалізує функції:
   1. Вивантаження файлу з початковими даними
   2. Перегляд результатів
   3. Завантаження .xlsx файлу з результатами
2. Прикладний програмний інтерфейс (API), який відповідає за кластеризацію написаний на мові програмування Python 3. Для завантаження та обробки даних використовувалась бібліотека pandas, для візуалізації бібліотека matplotlib, для кластеризації kmodes та kneed для знаходження оптимальної кількості кластерів.
3. Прикладний програмний інтерфейс, що поєднує між собою два попередні пункти та реалізує перетворення даних для відображення та збереження результатів. Цей інтерфейс написаний на мові NET 5, та ASP.NET Core 5.0.

Архітектура веб застосунку зображена на рисунку 4.

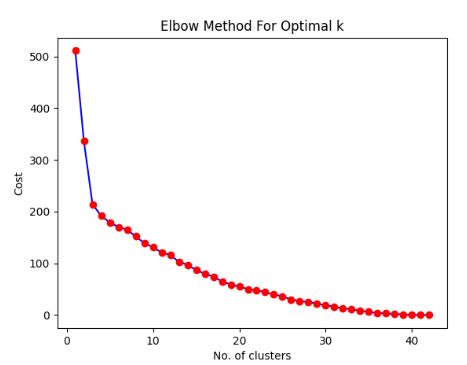
  
Рис. 4. Архітектура веб застосунку

# Розділ 4. Результати

Розроблений веб застосунок має декілька сторінок на рисунку 5 зображено початкову сторінку на якій можна завантажити файл з даними про перехресне зв’язування антитіл.

  
Рис. 5. Початкова сторінка веб застосунку

Після завантаження файлу відбувається його відправка на API яке відповідає за кластеризацію. Спочатку дані маркуються кольорами з гідно з алгоритмом наведеним в розділі 2. Після цього знаходиться відстань в середині кластера для k – кількість кластерів від 0 до розмір вибірки та на основі цих даних будується графік. Для наших даних видно, що оптимальна кількість кластерів буде 9.

  
Рис. 6. Графік кількості кластерів до відстані в середині кластера

Коли оптимальна кількість кластерів знайдена, відбувається безпосередня кластеризація методом k-modes та відправка результатів на інтерфейс користувача. На інтерфейсі користувача дані відображаються у два способи:

1. У вигляді карток де заголовок вказує на групу, а в тілі міститься посортований список антитіл (рис. 7).
2. У вигляді початкових даних, позначеними кольорами (рис. 8). Співпадіння кольорів в тій чи іншій колонці в середині певного кластеру підтверджує, що вони справді належать до однієї групи.

Зображення, що містить текст, екран, знімок екрана, шафа

Автоматично згенерований опис

Рис. 7. Результат кластеризації у вигляді карток

Зображення, що містить текст, електроніка, інший, кілька

Автоматично згенерований опис

Рис. 8. Фрагмент початкових даних розбитих на групи

|  |  |  |
| --- | --- | --- |
| Групи | Під групи | Елементи |
| 1 | A | NP1501, NP1514, NP1516, NP1517, NP1507 |
| B | X190, NP1526, X200, X201 |
| 1B/2 | | NP1512, NP1521 |
| 2 | | NP1502, NP1503, NP1508, NP1510, NP1520, NP1522, NP1525, X221, X271, NP3701, NP3708 |
| 2B/3 | | NP1528 |
| 3 | A | X202, X218, NP1518, NP1527 |
| B | X32, X155, X41, X212, X213, X217, X223, X224, X233, NP1524, NP3715 |
| 4 | A | NP3706 |
| B | X211 |
| C | X215 |
| 5 | | X220 |
| 6 | | X275 |

Табл. 1. Очікуваний результат

|  |  |
| --- | --- |
| Кластери | Елементи |
| 1 | NP1501, NP1512, NP1514, NP1516, NP1517, NP1521, X211, X215, X218, X220, X275 |
| 2 | NP1528 |
| 3 | NP1502, NP1503, NP1508, NP1510, NP1520, NP1522, NP1525, NP3708, X190, X221, X271 |
| 4 | NP1524, NP3715, X212, X217, X223, X224 |
| 5 | NP1526, NP3701, X200, X201 |
| 6 | NP1507 |
| 7 | X155, X202, X213, X233, X32, X41 |
| 8 | NP1518 |
| 9 | NP1527, NP3706 |

Табл. 2. Отриманий в результаті кластеризації результати

В результаті експериментальних досліджень доведено, що для даного вірусу найкращий результат дає пара де одне антитіло з групи 3B (X155, X41, X213 та X32), а інше з групи 4A (NP3706). Можемо побачити що розглянутий в цій роботі алгоритм кластеризації добре, розділив саме ці антитіла на групи.

Також реалізовано збереження результатів у форматі .xlsx. Ці результати подані у вигляді таблиці схожі на результати подані на веб сторінці та складаються з:

1. Таблиці де заголовок вказує на групу, а в тілі міститься посортований список антитіл (рис. 9).
2. Таблиця з початковими даними розбита на групи та позначена кольорами (рис. 10).

Зображення, що містить стіл

Автоматично згенерований опис

Рис. 9. Результат кластеризації записаний у файл

Зображення, що містить стіл

Автоматично згенерований опис

Рис. 10. Фрагмент початкових даних розбитих на групи та записаних у файл

# Висновки

В результаті виконання цієї курсової було розроблено веб застосунок, який полегшує задачу розбиття списку антитіл на групи на основі індексу перехресного зв’язування. Цей застосунок представляє з себе три мікросервіси:

1. Інтерфейс користувача для вивантаження файлів, перегляду та завантаження результатів.
2. Прикладний програмний інтерфейс, який відповідає за кластеризацію
3. API, що поєднує інтерфейс для кластеризацію та інтерфейс користувача, а також реалізує завантаження результатів.

Для розробки системи було використано NET 5 з ASP.NET, Python 3 з Flask та Type Script з React.

Також було розділено антитіла для конкретного вірусу. Зважаючи на те, які антитіла опинились в однакових групах порівнюючи очікуваний та отриманий результат можна сказати, що правильність розбиття на кластери склала 70%.

# Список використаних джерел

1. Satyam Kumar Clustering Algorithm for data with mixed Categorical and Numerical features [Electronic resource]. – 2021. – URL: https://towardsdatascience.com/clustering-algorithm-for-data-with-mixed-categorical-and-numerical-features-d4e3a48066a0
2. Z. Huang. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values (1998). Data Mining and Knowledge Discovery. 2(3): 283–304.
3. Python: K-modes explanation [Electronic resource]. – 2017. – URL:

https://stackoverflow.com/questions/42639824/python-k-modes-explanation

1. Audhi Aprilliant The k-modes as Clustering Algorithm for Categorical Data Type [Electronic resource]. – 2021. – URL: https://medium.com/geekculture/the-k-modes-as-clustering-algorithm-for-categorical-data-type-bcde8f95efd7.
2. Fuyuan Cao A new initialization method for categorical data clustering, Fuyuan Cao, Jiye Liang, Liang Bai [Electronic resource]. – 2009. – URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.8181&rep=rep1&type=pdf

# Додатки

Додаток А. Код який відповідає за кластеризацію

**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**import** numpy.typing **as** npt  
**import** pandas **as** pd  
**from** kmodes.kmodes **import** KModes *# pip install kmodes***from** kneed **import** KneeLocator *# pip install kneed***from** constants **import** cluster\_column\_name, labelled\_column\_name  
  
*# 1 to show and 0 to hide*verbose = 1  
  
  
**def** setup\_kmodes(num\_clusters: int, data: pd.DataFrame) -> KModes:  
 *"""  
 Setup KModes* **:param** *num\_clusters: amount of clusters* **:return***: instance of KModes  
 """* **return** KModes(n\_clusters=num\_clusters, init=**"Cao"**, n\_init=15, verbose=verbose)  
  
  
**def** build\_elbow\_curve(data: pd.DataFrame) -> tuple[range, list[int]]:  
 *"""  
 Build Elbow curve to find optimal cluster amount* **:param** *data: data to be clustered* **:return***: range from 1 to max amount of clusters, cost of each clustered model  
 """* cost = []  
 cluster\_amount\_range = range(1, data.shape[0])  
 **for** num\_clusters **in** list(cluster\_amount\_range):  
 kmodes = setup\_kmodes(num\_clusters, data)  
 kmodes.fit\_predict(data)  
 cost.append(kmodes.cost\_)  
  
 **if** verbose == 1:  
 build\_elbow\_plot(cluster\_amount\_range, cost)  
 **return** cluster\_amount\_range, cost  
  
  
**def** build\_elbow\_plot(cluster\_amount\_range: range, cost: list[int]):  
 *"""  
 Build and show Elbow curve plot* **:param** *cluster\_amount\_range: range from 1 to max amount of clusters* **:param** *cost: cost of clustering  
 """* plt.plot(cluster\_amount\_range, cost, **'o-'**, color=**"blue"**, markerfacecolor=**'red'**, markeredgecolor=**'red'**)  
 plt.xlabel(**'No. of clusters'**)  
 plt.ylabel(**'Cost'**)  
 plt.title(**'Elbow Method For Optimal k'**)  
 plt.show()  
  
  
**def** get\_optimal\_cluster\_amount(data: pd.DataFrame) -> int:  
 *"""  
 Get exact cluster amount* **:param** *data: data to be clustered* **:return***: optimal cluster amount base on elbow method  
 """* cluster\_amount\_range, cost = build\_elbow\_curve(data)  
 kl = KneeLocator(cluster\_amount\_range, cost, curve=**"convex"**, direction=**"decreasing"**)  
 exact\_cluster\_amount = kl.elbow  
 **return** exact\_cluster\_amount  
  
  
**def** get\_clusters\_for\_optimal\_model(data: pd.DataFrame, cluster\_amount: int) -> npt.NDArray[np.uint16]:  
 *"""  
 Build optimal model with 'cluster\_amount' clusters* **:param** *cluster\_amount: cluster amount* **:param** *data: data to be clustered* **:return***: list of cluster number for each data row  
 """* kmodes = setup\_kmodes(cluster\_amount, data)  
 clusters = kmodes.fit\_predict(data)  
 **return** clusters  
  
  
**def** format\_response(data: pd.DataFrame, clusters: npt.NDArray[np.uint16]) -> pd.DataFrame:  
 *"""  
 Format response data* **:param** *data: data to be clustered* **:param** *clusters: list of cluster number for each data row* **:return***: json with clustered data  
 """* data.insert(0, cluster\_column\_name, clusters, **True**)  
 data = data.reset\_index()  
 data = data.sort\_values(by=[cluster\_column\_name, labelled\_column\_name])  
 data = data.set\_index(labelled\_column\_name)  
 **return** data  
  
  
**def** clustering(data: pd.DataFrame) -> pd.DataFrame:  
 *"""  
 Clustering data* **:param** *data: data to be clustered* **:return***: json with clustered data  
 """* cluster\_amount = get\_optimal\_cluster\_amount(data)  
 print(**f"Optimal cluster amount {**cluster\_amount**}"**)  
 clusters = get\_clusters\_for\_optimal\_model(data, cluster\_amount)  
 response = format\_response(data, clusters)  
 **return** response

Додаток Б. Код який відповідає за збереження результатів в Excel

public class ExcelService : IExcelService

{

private readonly IStyleProvider \_styleProvider;

public ExcelService(IStyleProvider styleProvider)

{

\_styleProvider = styleProvider;

}

public MemoryStream GetFileStream(CrossInhibitorRawDataModel dataModel)

{

var sl = new SLDocument();

var rowIndex = 2;

var columnIndex = 2;

var groupHeaderStyle = \_styleProvider.GetGroupHeaderStyle();

var groupCellStyle = \_styleProvider.GetGroupCellStyle();

var lastGroupCellStyle = \_styleProvider.GetGroupLastCellStyle();

var darkGreenStyle = \_styleProvider.GetFilledCellStyle(System.Drawing.Color.FromArgb(0, 176, 79));

var lightGreenStyle = \_styleProvider.GetFilledCellStyle(System.Drawing.Color.FromArgb(146, 208, 80));

// Add clusters to file

foreach (var keyValue in dataModel.Clusters)

{

sl.SetCellValue(rowIndex, columnIndex, $"Group {keyValue.Key + 1}");

sl.SetCellStyle(rowIndex, columnIndex, groupHeaderStyle);

rowIndex++;

for (var i = 0; i < keyValue.Value.Count; i++)

{

var item = keyValue.Value[i];

sl.SetCellValue(rowIndex, columnIndex, item);

if (i == keyValue.Value.Count - 1)

{

// Add border in the bottom to close table

sl.SetCellStyle(rowIndex, columnIndex, lastGroupCellStyle);

}

else

{

sl.SetCellStyle(rowIndex, columnIndex, groupCellStyle);

}

rowIndex++;

}

columnIndex++;

rowIndex = 2;

}

var intialRawDataRow = rowIndex + dataModel.Clusters.Values.ToList().Max(x => x.Count) + 2;

// Add marked antigen labels to file

rowIndex = intialRawDataRow;

columnIndex = 3;

foreach (var label in dataModel.MarkedAntigenLabels)

{

sl.SetCellValue(rowIndex, columnIndex, label);

columnIndex++;

}

// Add antigen labels to file

rowIndex++;

columnIndex = 1;

for (var i = 0; i < dataModel.AntigenLabels.Count; i++)

{

sl.SetCellValue(rowIndex, columnIndex, $"Group {i + 1}");

columnIndex++;

var labelGroup = dataModel.AntigenLabels[i];

foreach (var label in labelGroup)

{

sl.SetCellValue(rowIndex, columnIndex, label);

rowIndex++;

}

rowIndex++;

columnIndex = 1;

}

// Add cross inhibition indexes to file

rowIndex = intialRawDataRow + 1;

columnIndex = 3;

foreach (var clusterGroup in dataModel.CrossInhibitionIndexes)

{

foreach (var row in clusterGroup)

{

foreach (var cell in row)

{

sl.SetCellValue(rowIndex, columnIndex, cell.Value);

switch (cell.MarkerColor)

{

case InhibitionColors.DarkGreen:

sl.SetCellStyle(rowIndex, columnIndex, darkGreenStyle);

break;

case InhibitionColors.LightGreen:

sl.SetCellStyle(rowIndex, columnIndex, lightGreenStyle);

break;

default:

break;

}

columnIndex++;

}

rowIndex++;

columnIndex = 3;

}

rowIndex++;

}

using var stream = new MemoryStream();

sl.SaveAs(stream);

return stream;

}

}

public class StyleProvider : IStyleProvider

{

public SLStyle GetGroupHeaderStyle()

{

var style = new SLStyle();

style.SetTopBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetLeftBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetRightBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetBottomBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetHorizontalAlignment(HorizontalAlignmentValues.Center);

return style;

}

public SLStyle GetGroupCellStyle()

{

var style = new SLStyle();

style.SetLeftBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetRightBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

return style;

}

public SLStyle GetGroupLastCellStyle()

{

var style = new SLStyle();

style.SetLeftBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetRightBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

style.SetBottomBorder(BorderStyleValues.Thin, SLThemeColorIndexValues.Dark1Color);

return style;

}

public SLStyle GetFilledCellStyle(System.Drawing.Color color)

{

var style = new SLStyle();

style.Fill.SetPattern(PatternValues.Solid, color, color);

return style;

}

}