СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет *Інформаційних технологій та електроніки*

Кафедра *Інформаційних технологій та програмування*

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дипломної роботи

Магістр
(освітньо-кваліфікаційний рівень)

на тему *Дослідження методів класифікації даних, на прикладі алгоритму розподілу абітурієнтів за рейтинговою системою*

Виконав: *студент 2 курсу, групи ІСТ-22ДМ 126 «Інформаційні системи та технології»*
(шифр і назва спеціальності)

Кручківський І. А.
(ініціали і прізвище)

Керівник *Захожай О. І.*
(ініціали і прізвище)

Рецензент *Ратов Д. В.*
(ініціали і прізвище)

Київ - 2023

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет *Інформаційних технологій та електроніки*
Кафедра *Інформаційних технологій та програмування*
Освітньо-кваліфікаційний рівень *Магістр*
Спеціальність *126 Інформаційні системи та технології*
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., проф., Захожай О. І.
(підпис)
"___" _____ 2023 року

ЗАВДАННЯ

На магістерську дипломну роботу студента
Кручківський Ігор Анатолійович

1. Тема роботи *Дослідження методів класифікації даних, на прикладі алгоритму розподілу абітурієнтів за рейтинговою системою*

Керівник роботи *д.т.н., професор, Захожай Олег Ігорович*,
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
Затверджений наказом університету від "*21*"*жовтня* 2023року №*614/15.15-С*
2. Строк подання студентом роботи *08 грудня 2023 р.*
3. Вихідні дані роботи *Матеріали науково-дослідницької практики, науково-методична література, дані інтернет мережі*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ

4.2 Аналітичний огляд питання.

4.3 Основна частина, де досліджено та проаналізовано методи для реалізації проекту, а також сформовано вирішення для розв'язку поставленої задачі.

4.4 Практична частина, в якій зроблено огляд технологій та інструментарію, які використовуються для реалізації проекту.

4.5 Висновки

4.6 Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *20 жовтня 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1.	Отримання завдання на виконання роботи	20.10.2023	
2.	Укладення та погодження з керівником плану і етапів виконання роботи	24.10.2023	
3.	Погодження з керівником оптимального шляху виконання завдання.	01.11.2023	
4.	Аналіз технічних засобів існуючих систем	10.11.2023	
5.	Реалізація практичної частини	24.11.2023	
6.	Написання, оформлення та узгодження пояснювальної записки з керівником	05.12.2023	
7.	Здача пояснювальної записки на кафедру	08.12.2023	
8.	Підготовка доповіді та презентації	10.12.2023	

Студент _____ *Кручківський І. А.*
(підпис) (ініціали і прізвище)

Керівник роботи _____ *Захожай О. І.*
(підпис) (ініціали і прізвище)

РЕФЕРАТ

Магістерська дипломна робота: 46 стор., 2 табл., 12 рис., 7 джерел.

Об’єкт дослідження – порівняння алгоритмів класифікації даних.

Мета роботи – аналіз існуючих алгоритмів класифікації даних, та пошук найоптимальнішого з них, для розробки власного алгоритму розподілу абітурієнтів за рейтинговою системою.

Проаналізовано алгоритми класифікації даних, порівняно між собою за допомогою певних критеріїв. Розроблений власний алгоритм на основі цих даних, та інтегрований в програмний комплекс.

Для досягнення поставленої мети, дипломна робота поділена на такі завдання:

1) Дослідити предметну область;
2) Порівняти та виокремити алгоритми класифікації даних;
3) Розробити власний алгоритм, з урахуванням критеріїв;
4) Створити програмне забезпечення, яке включає даний алгоритм.

АЛГОРИТМ, КЛАСИФІКАЦІЯ ДАНИХ, СПИСОК, ДЕРЕВО РІШЕНЬ, LINQ, MICROSOFT INTEROP, MVC, C#, VISUAL STUDIO, MICRSOFT EXCEL, ГРАФІК ФУНКЦІЇ, ОПОРНИЙ ВЕКТОР, БАЙЄСІВСЬКИЙ КЛАСИФІКАТОР, ПАРАДОКС

ABSTRACT

Master's thesis: 46 pages, 2 tables, 12 pictures, 7 sources.

Object of study - comparison of data classification algorithms.

The aim of the research is to analyze existing data classification algorithms and find the most optimal to develop own algorithm for distributing applicants according to the rating system.

Data classification algorithms are analyzed and compared with each other using certain criteria. The algorithm was developed on the basis of these data and integrated into the software system.

To achieve this goal, the thesis is divided into the following tasks:

1) Research the subject area;
2) Compare and differentiate data classification algorithms;
3) Develop an own algorithm, taking into account the criteria;
4) Create software that includes this algorithm.

ALGORITHM, DATA CLASSIFICATION, LIST, DECISION TREE, LINQ, MICROSOFT INTEROP, MVC, C#, VISUAL STUDIO, MICROSOFT EXCEL, FUNCTION GRAPH, SUPPORT VECTOR, BAYES CLASSIFIER, PARADOX

ЗМІСТ

ВСТУП

На сьогоднішній день нема жодної сфери діяльності людини, де би не використовувалися алгоритми. Наприклад людина, яка працює поваром, чи програміст який дописує тисячну строку свого коду. Цих людей об’єднує алгоритмізація їх дій. Тому, це поняття не обійшло стороною сферу інформаційних технологій. Воно є основою програмування, і допомагає програмістам ефективніше виконувати поставлені задачі.

Беручи невеликий історичний екскурс, 1843 року, Ада Лавлейс описала алгоритм обчислення чисел Бернуллі на аналітичній машині Беббіджа. [1] Цей факт можна вважати дуже важливим в історії інформаційних технологій, оскільки цей алгоритм були признано першим програмним забезпеченням для виконання на електронно-обчислювальній машині, а також за цим фактом Аду вважають першим програмістом у світі.

Також потрібно зазначити, що алгоритми в свої основі використовують вхідні дані, які представлені у певній структурі (масив, список, граф та інші.). То ж алгоритми тісно пов’язані зі структурами даних.

В дипломній роботі, представлена конкретна задача – розподіл абітурієнтів за рейтинговою системою, яка вирішується за допомогою алгоритму. Актуальність теми полягає в тому, що кінцевий результат (програмний комплекс), допоможе ефективніше проводити вступну кампанію університету. Також потрібно зазначити, що даний алгоритм вже має свою реалізацію, у Єдиній Державній Електронній Базі з питань Освіти (ЄДЕБО), але наявний там алгоритм немає відкритого вихідного коду [2], тож було вирішено розробити власний алгоритм на його основі, з урахуванням критеріїв заявлених керівництвом університету.

Головню метою дипломної роботи, як було зазначено вище, є дослідження наявних алгоритмів класифікації даних, виокремлення за певними критеріями найдоцільнішого для вирішення завдання, і у кінцевому результаті розробка програмного забезпечення з його використанням.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Алгоритми та їх характеристики

Як було зазначено у вступі, **алгоритм** – це точна послідовність дій для вирішення конкретної проблеми або класу проблем. Алгоритми мають чіткі властивості: дискретність, визначеність, результативність, масовість.

У книзі «Мистецтво програмування. Основні алгоритми» Дональд Кнут виділяє такі властивості [3], як наявність вхідних (можливо й нульових; ці властивості беруться з певного набору і визначаються до початку роботи алгоритму чи визначаються динамічно під час роботи) і вихідних (результуючих, яких може бути декілька) даних, а також ефективність (алгоритм вважають ефективним, якщо його можна точно виконати протягом скінченного часу за допомогою олівця паперу).

Алгоритми можуть бути детермінованими й недетермінованими. Детермінований алгоритм для тих самих вхідних даних видає ті ж самі результати і передбачає єдиний шлях їхньої обробки (Цей алгоритм не передбачає жодної випадковості). В свою чергу, недетермінований алгоритм передбачає кілька шляхів обробки тих самих вихідних вхідних даних і може призвести до різних результатів, в залежності від випадкових чинників.

Ще однією не менш важливою ознакою алгоритмів є, складність. Складність алгоритмів – це концепція, що визначає кількість ресурсів, таких як час і пам'ять, які потрібні для виконання алгоритму. Ця концепція є важливою при розробці програмного забезпечення, оскільки вибір оптимального алгоритму може суттєво впливати на продуктивність програмного забезпечення.

У більшості випадків складність залежить від обсягу вхідних даних. Наприклад, масив з 10 елементів буде оброблений швидше, аніж із 1000 елементів. При цьому точний час залежить від процесора, типу даних, мови програмування та інших параметрів.

Для оцінювання складності алгоритмів використовують поняття, **асимптотична складність**, і його пов’язують з обробкою дуже великих наборів даних.

Нехай n – кількість елементів вхідних даних, функція f(n) описує верхню межу максимальної кількості основних операцій (додавання, множення, порівняння, присвоювання, перестановок тощо), які виконуються в алгоритмі. Якщо функція f(n) зростає не швидше, ніж деякий багаточлен (поліном) P(n), то кажуть що алгоритм має поліноміальну складність, або алгоритм є поліноміальним, в іншому випадку алгоритм є неполіноміальним. Поліноміальні алгоритми для великих обсягів даних, характерних для практичних задач, можуть виконуватися на сучасних комп'ютерах, а неполіноміальні (як правило, алгоритми повного перебору) через великі потреби часу не можуть виконуватися повністю.

Оскільки, в більшості випадків, точний вигляд функції f(n) записати складно (навіть, коли алгоритм деталізовано і добре досліджено, отримати точний вигляд функції досить важко), то використовують її асимптотичну оцінку O(g(n)).

Функцію O(g(n)) називають асимптотичною складністю алгоритму, якщо:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = const$$

Наприклад, якщо певний алгоритм виконує $7n^3 + 9n$ операцій, то асимптотична складність цього алгоритму дорівнює $O(n^3)$ (відповідно алгоритм називають кубічним), оскільки границя функції є константою, 7.

Формально O(g(n)) означає, що час роботи алгоритму або обсяг занятої пам'яті зростає в залежності від обсягу вхідних даних не швидше, ніж певна константа помножена на g(n).

Найчастіше, використовують такі класи складності алгоритмів:

O(1) – константа складність алгоритму (сталий час виконання алгоритму), який не залежить від обсягів вхідних даних. Наприклад, отримання елементу за індексом в масиві.

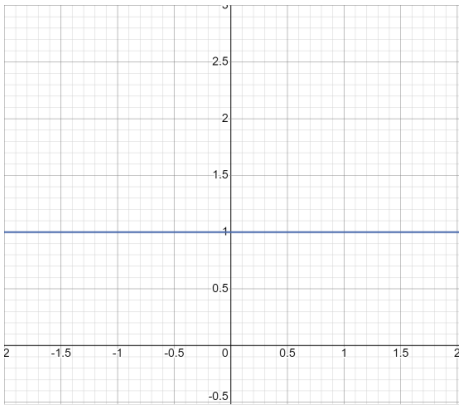


Рисунок 1.1.1 Графік функції O(1)

O(n) – лінійна складність, де час або простір лінійно зростає з розміром вхідних даних. Наприклад, перегляд усіх елементів у масиві. Графік функції цього типу складності зображений на рисунку 2.1.2 .

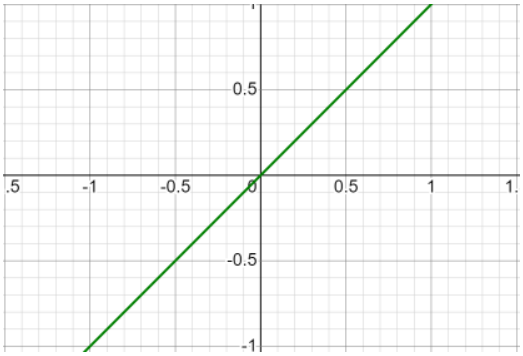


Рисунок 1.1.2 Графік функції O(n)

$O(n^2)$ – квадратична складність, де час або простір зростає квадратично з розміром вхідних даних. Наприклад, вкладений цикл для обробки кожної пари елементів у масиві. Або, ще один гарний приклад це, алгоритм Дейкстри знаходження найкоротшого шляху між вершинами зваженого графа. Графік функції зображений на рисунку 1.1.3 .

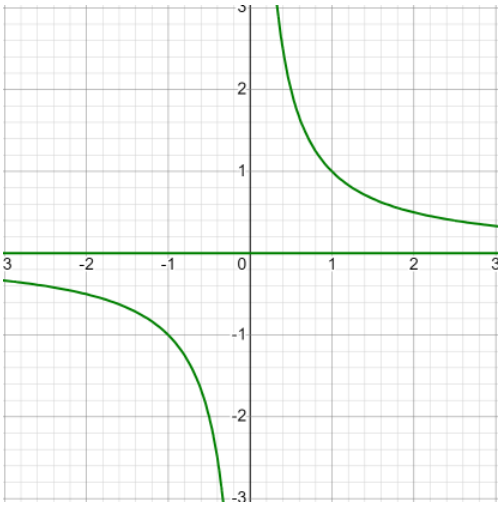


Рисунок 1.1.3 Графік функції $O(n^2)$

O(log n) – логарифмічна складність, де час або простір логарифмічно зростає з розміром вхідних даних. Наприклад, бінарний пошук у відсортованому масиві. Алгоритм працює, скорочуючи обсяг оброблюваних даних на певний постійний коефіцієнт при кожній ітерації. Графік функції, зображений на рисунку 1.1.4 .

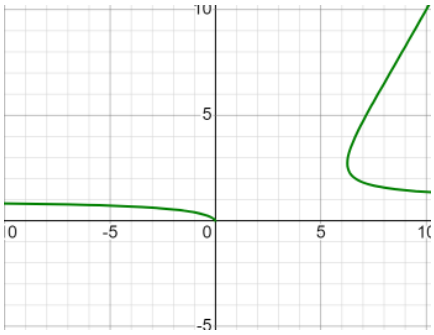


Рисунок 1.1.4 Графік функції O(log n)

$O(k^n)$ – експоненціальна складність, де час або простір експоненційно зростає з розміром вхідних даних. Наприклад вирішення задачі комівояжера за допомогою рекурсії. Графік функції цього типу складності, зображений на рисунку 1.1.5 .

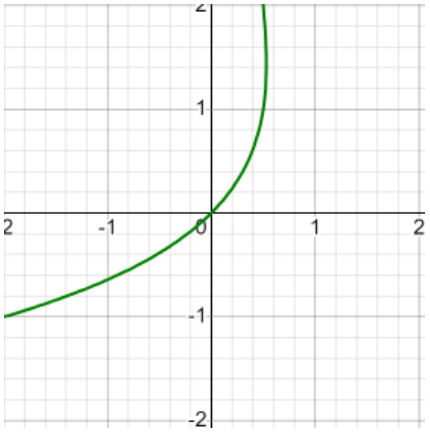


Рисунок 1.1.5 Графік функції $O(2^n)$

В цілому, задачі для яких розроблено алгоритми з константною чи логарифмічною часовою складністю, розв’язуються дуже швидко. Більшість використовуваних алгоритмів розв’язання задач мають поліноміальну складність $O(n^a)$, де $a \geq 1$ (a може бути і дробовим числом). Задачі, які можна розв’язувати алгоритмами з поліноміальною часовою складністю належать до класу « P ». Знайти їхній розв’язок для великих обсягів вхідних даних можна за прийнятний час, тобто задачі є практично розв’язними.

В свою чергу, алгоритми з експоненційною і факторіальною складністю можуть розв’язувати задачі при дуже незначних обсягах вхідних даних. Для загального розуміння, у Таблиці 1.1.1 показана залежність часу від обсягу даних і складності алгоритму, при тому що одна операція виконується одну наносекунду (тобто виконується один мільярд операцій за секунду).

Таблиця 1.1.1 Час виконання алгоритмів

Обсяг даних / Складність алгоритму	10	20	30	40	50
n	0.00001 с	0.00002 с	0.00003 с	0.00004 с	0.00005 с
n^2	0.0001 с	0.0004 с	0.0009 с	0.001 с	0.0025 с
n^3	0.001 с	0.008 с	0.027 с	0.064 с	0.125 с
$\log_2 n$	$3{,}32*10^{-6}$ с	$4{,}32*10^{-6}$ с	$4{,}91*10^{-6}$ с	$5{,}32*10^{-6}$ с	$5{,}64*10^{-6}$ с
$n * \log_2 n$	$3{,}32*10^{-5}$ с	$8{,}64*10^{-5}$ с	$1{,}47*10^{-4}$ с	$2{,}13*10^{-4}$ с	$2{,}82*10^{-4}$ с
$\log_2 \log_2 n$	$1{,}73*10^{-6}$ с	$2{,}11*10^{-6}$ с	$2{,}29*10^{-6}$ с	$2{,}41*10^{-6}$ с	$2{,}5*10^{-6}$ с
3^n	0.059 с	58.11 хв	6.53 років	386.59 століть	$2.28*10^8$ століть

Отже, при виборі конкретного алгоритму треба враховувати вимоги й умови щодо його використання, наприклад: алгоритм який виконується в 10 разів швидше, але використовує в 10 разів більше пам’яті може працювати з великою кількістю даних на сервері чи робочій станції, з великим обсягом пам’яті, але на звичайних комп’ютерах з невеликим обсягом пам’яті він буде виконуватися дуже довго, що показує в його недоцільності, і заміні більш ефективнішим алгоритмом.

1.2 Класифікація даних, та її види

Класифікація даних – це процес упорядковування даних, за певними класифікаційними ознаками, обраних для схожості або відмінності цих даних. Цей процес, є важливим аспектом машинного навчання. Основними поняттями класифікації даних є:

1. **Об’єкт** – це дані для яких ми хочемо здійснити класифікацію. Це може бути текст, зображення, числа, тощо.
2. **Ознаки** – це властивості цих об’єктів, які використовуються для призначення їх до певних класів.
3. **Класи** – це унікальні категорії або мітки, до яких ми можемо призначити об’єкти за певними ознаками.

Також, класифікація даних вимагає дотримання таких правил:

- При кожному етапі сортування, необхідно застосовувати тільки одну основу;
- Сортування повинно бути «пропорційним», тобто загальний обсяг видових понять повинен дорівнювати обсягу ділення родового поняття;
- Члени сортування повинні взаємно виключати один одного – їх обсяги не повинні перехрещуватися;
- Сортування даних повинно бути послідовним.

Розрізняють два види класифікації даних, штучну та природню. **Штучна класифікація** , це та яка здійснюється за зовнішньою ознакою і служить для надання множині предметів (процесів, явищ) потрібного порядку. **Природня класифікація** , в свою чергу, це та яка здійснюється за істотними ознаками, що характеризують внутрішню спільність предметів і явищ. Вона є результатом і важливим засобом наукового дослідження, тому що передбачає і закріплює результати вивчення закономірностей об’єктів, що класифікуються.

Прикладом штучної класифікації, є розподіл студентів курсу в список, за алфавітним порядку, або такий же розподіл машин в каталозі виробників за алфавітним порядком. Знаючи порядок букв в алфавіті, можна легко і швидко знайти потрібне нам прізвище, або інші об’єкти за певними ознаками. Але знання того, яке місце в допоміжній класифікаційній системі займає той чи інший об’єкт, не дає можливості щось стверджувати про його властивості. Наприклад, якщо студент Антонов внесений в список першим, а Янович – останнім, абсолютно нічого не говорить про їх здібності до навчання. Тому можна вважати, що штучна класифікація не є до кінця науковою.

В свою чергу, природня класифікація являє собою розподіл об’єктів за класами на підставі їх найбільших суттєвих ознак. Наприклад найбільш суттєвою ознакою людини є її здатність до праці. Ця ознака обумовлює наявність у людини таких ознак, як прямоходіння, здатність до спілкування, здатність до мислення та ін.

Також потрібно зазначити, що між природньою і штучною класифікацію, все ж існує певний зв'язок. Він полягає в тому, що штучна класифікація може намагатись відтворити або використовувати природні принципи класифікації для ефективного аналізу даних чи об’єктів. (Аспекти цього зв’язку показані на рисунку 1.2.1)

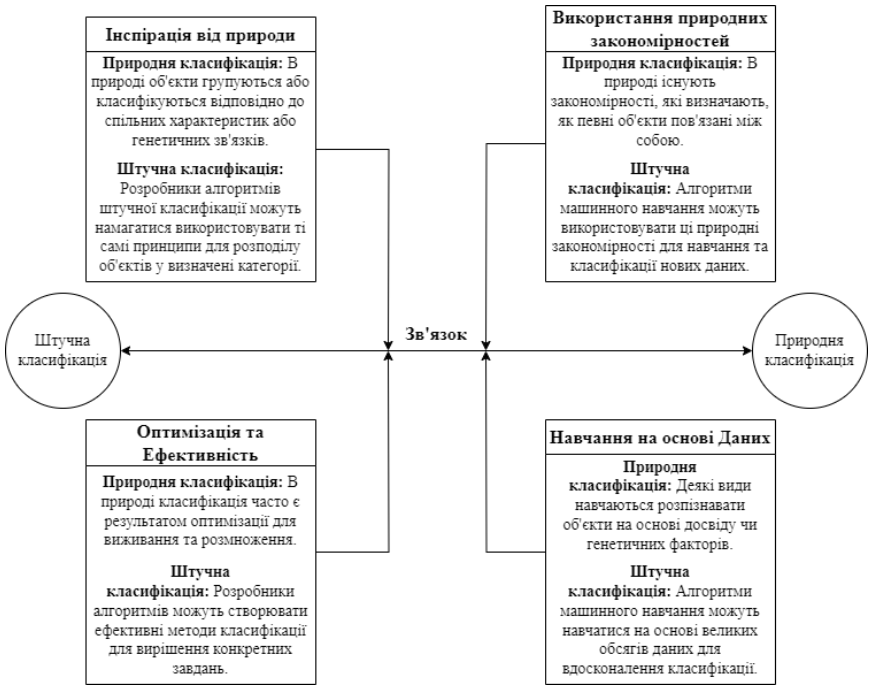


Рисунок 1.2.1 Зв’язок між штучною та природньою класифікацією

1.3 Метрика класифікації даних

Метрика класифікації даних – це числова величина, яка використовується для вимірювання якості роботи моделі класифікації (об’єкту) на основі порівняння її прогнозів з фактичними класами даних. Ці метрики надають інформацію про те наскільки ефективно модель розпізнає класи та здатність уникати помилок. Нижче наведений приклад таких метрик:

1. **Правильність(Accuracy)** – це відношення кількості правильно класифікованих екземплярів, до загальної кількості екземплярів.

   Формула: $\frac{\text{Кількість правильних прогнозів}}{\text{Загальна кількість прогнозів}}$

2. **Точність(Precision)** – це відношення кількості правильно класифікованих позитивних екземплярів до загальної кількості позитивних прогнозів.

   Формула: $\frac{\text{Кількість правильних позитивних прогнозів}}{\text{Загальна кількість позитивних прогнозів}}$

3. **Повнота(Recall or Sensitivity)** – це відношення кількості правильно класифікованих позитивних екземплярів до загальної кількості фактично позитивних екземплярів.

   Формула: $\frac{\text{Кількість правильних позитивних прогнозів}}{\text{Загальна кількість фактично позитивних екземплярів}}$

4. **F1-Оцінка(F1-Score)** – це середнє гармонічне точності та повноти. Використовується для об’єднання обох метрик у випадку, коли важливо отримати баланс між ними.

   Формула: $\mathrm{F}_1 = \frac{2 * \text{Точність} * \text{Повнота}}{\text{Точність} + \text{Повнота}}$

5. **Специфічність(Specificity)** – це відношення кількості правильно класифікованих негативних екземплярів до загальної кількості фактично негативних екземплярів.

   Формула: $\frac{\text{Кількість правильних негативних прогнозів}}{\text{Загальна кількість фактично негативних екземплярів}}$

Ці метрики допомагають отримати об’єктивну оцінку продуктивності класифікаційної моделі, враховуючи різні аспекти її роботи. Вибір конкретної метрики залежить від характеру даних та специфікацій задачі.

Метрики, мають свої парадокси. [4] Ці парадокси можуть виникати через специфіку розподілу класів у наборі даних та взаємодією між різними метриками. Нижче, наведено декілька парадоксів.

Парадокс точності(переваги класу) , з’являється коли класи нерівномірно розподілені в наборі даних, і один клас переважає за кількістю. Наприклад, якщо 95% зразків належать до класу А, а 5% - до класу Б, модель може досягти високої точності, класифікуючи все як клас А, але при цьому не бути ефективною для класу Б.

Парадокс чутливості(специфічності). Його суть полягає в тому, що метрики чутливості (повноти) можуть конфліктувати, і покращення однієї може призвести до погіршення іншої. Наприклад, збільшення чутливості може призвести до більше помилок першого роду (фальшивих позитивів), що знижує специфічність.

Парадокс F-міри. Виникає коли міра об’єднує чутливість та точність у одну метрику, але є компромісом між ними. Наприклад, висока F-міра може бути результатом або високої точності і низької чутливості, або низької точності і високої чутливості.

Розуміння цих парадоксів допомагає визначити, які аспекти класифікації важливі для конкретної задачі та як підібрати метрики відповідно до цих аспектів.

1.4 Висновки розділу

Отже, в цьому розділі були розглянуті основні поняття алгоритмізації, та викладені їх детальні характеристики. Також були досліджені основні характеристики класифікації даних.

В наступному розділі, дана інформація допоможе описати і класифікувати, основні види алгоритмів класифікації даних, та вибрати серед них найоптимальніший для вирішення задачі.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ АЛГОРИТМІВ КЛАСИФІКАЦІЇ ДАНИХ

2.1 Аналіз алгоритмів класифікації даних

Задача класифікації – це певна задача, яка містить множину об’єктів, поділених певним чином на класи. Початково, задана скінчена множина об’єктів, для яких відомо, до яких класів вони належать. Ця множина називається вибіркою. Невідомо, до якого класу належать інші об’єкти. Не обхідно побудувати такий алгоритм, який буде здатний класифікувати довільний об’єкт з вхідної множини. Для вирішення цієї задачі, існують алгоритми класифікації даних. Найпопулярніші серед них, це метод опорних векторів, наївний Баєсів класифікатор, та алгоритм побудови дерева рішень. Розглянемо кожен з них, та проведемо аналіз.

Метод опорних векторів (Support Vector Machine – SVM) Це алгоритм машинного навчання для вирішення задач класифікації, і регресії. [5] Основна ідея полягає в тому, щоб знайти гіперплощину в просторі великої розмірності, яка найкращим чином розділяє екземпляри одного класу від інших. До цього методу, входить такий набір даних:

1) **Роздільна гіперплощина** – це оптимальна площина, що розділяє дані на класи так, щоб максимізувати відстань (зазначену як маржа) між двома класами.
2) **Опорні вектори** – це точки навчальної вибірки, яка знаходяться найближче до гіперплощини. Ці вектори визначають маржу і є ключовими для побудови гіперплощини.
3) **Ядро** – це ядро яке дозволяє алгоритму працювати в просторах вищої розмірності, що робить його дуже потужним для розв’язування складних задач.
4) **Регуляризація** – це параметр, який допомагає уникнути перенавчання, забезпечуючи оптимальну гіперплощину з адекватною маржею.
5) **Функція вибору** – це функція, яка визначає як алгоритм має взаємодіяти з різними класами. Вона може бути лінійною або нелінійною, залежно від природи даних.

Також потрібно зазначити графік, який описує процес роботи алгоритму (Див. Рисунок 2.1.1). На графіку, зображено набір даних за осями x_1 та x_2 та гіперплощини які вони утворюють, з відстанню $\|\vec{\omega}\|$, класифікатори $\vec{\omega} * \vec{x} - b = 1$, $\vec{\omega} * \vec{x} - b = -1$, $\vec{\omega} * \vec{x} - b = 0$. З цього всього, можна отримати задачу оптимізації «Мінімізувати $\|\vec{\omega}\|$ за умови $y_i(\vec{\omega} * \vec{x_i} - b) \geq 1$ для $i = 1, \ldots, n$, де $\vec{\omega}$ та b, які розв'язують цю задачу, визначають класифікатор $\vec{x} \to sgn(\vec{\omega} * \vec{x} - b)$. В наслідку, цього геометричного опису є те що максимально розділова гіперплощина повністю визначена $\vec{x_i}$, які лежать найближче до неї. Ці $\vec{x_i}$, і називають опорними векторами.

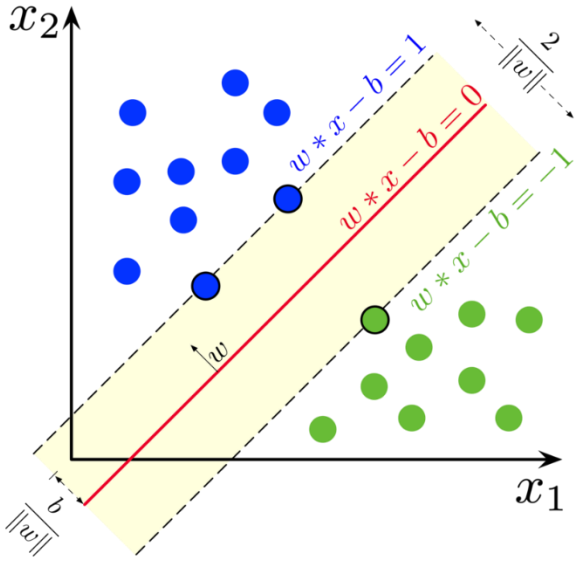


Рисунок 2.1.1 Графік гіперплощини зразків з двома класами

Отже, алгоритм опорних векторів, добре працює для багатьох типів даних і використовується в різних областях, таких як розпізнавання обличчя, розпізнавання рукописного тексту та інші.

Баєсовські алгоритми. [6] Одними, з відомих алгоритмів побудови класифікаційних правил є Баєсевський класифікатор. Наївний Баєсевський Класифікатор (точна назва з англійської Naive Bayes Classifier, NBC). Даний алгоритм, використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні (наївному, що означає вхідні змінні незалежні один від одного) незалежності змінних. Для кращого розуміння, алгоритму потрібно розглянути Теорему Баєса.

Теорема Баєса описує ймовірність події, спираючись на обставини, що могли би бути пов’язані з цією подією. Наприклад припустимо, що хтось цікавиться, чи має рак певна особа, і причому відомий вік цієї особи. Якщо рак пов'язаний з віком, то застосовуючи теорему Баєса, інформацію про вік осіб можливо використати для точнішої оцінки ймовірності того, що вони мають рак.

Теорему Баєса можна представити такою формулою:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Де:

- P(A|B) – ймовірність події A при умові, що сталося подія B.
- P(B|A) – ймовірність події B при умові, що сталося подія A.
- P(A) та P(B) – ймовірність події A та B відповідно.

При використанні, теореми Баєса, ймовірності можуть мати різні інтерпретації. В одній з них, теорема використовується безпосередньо у певному підході до статистичного узагальнення. При Баєсовій інтерпретації ймовірності, ця теорема виражає, як повинна раціонально змінюватися суб’єктивна міра впевненості при врахуванні свідчення, а саме це є Баєсовим узагальненням, що є фундаментальним для Баєсовської статистики.

В свою чергу, алгоритм Баєса можна поділити на таку послідовність дій:

- **Підготовка даних.** Під час підготовки даних, потрібно надати навчальну вибірку – сукупність усіх даних для навчання, яка складається з об’єктів та їхніх класів.
- **Навчання.** Під час цього процесу, потрібно провести обчислення ймовірностей. (Ймовірність кожного класу $P(C_k)$, та ймовірність кожного атрибуту для кожного класу $P(X_i|C_k)$.
- **Класифікація нового об’єкта.** В решті решт, найголовніший етап під час якого потрібно використовуючи теорему Баєса, обчислити ймовірність для кожного класу з урахуванням атрибутів нового об’єкту. І вибрати клас з найвищою ймовірністю.

Перераховані дії алгоритму, дають можливість описати це все за допомогою формули, яка має в своїй основі теорему Баєса. Дана формула представлена нижче:

$$P(C_k|x) = \frac{P(x|C_k) * P(C_k)}{P(x)}$$

Де:

- $P(C_k|x)$ – ймовірність класу C_k при умові, що спостерігається об’єкт x.
- $P(x|C_k)$ – ймовірність класу x при умові, що спостерігається об’єкт C_k.
- $P(C_k)$ – апріорна ймовірність класу C_k.
- P(x) – ймовірність об’єкта x.

Отже, алгоритм Баєса, так само як і алгоритм опорних векторів знаходить практичне застосування у багатьох сферах діяльності. Далі, на черзі, останній алгоритм побудови дерева рішень.

Алгоритм побудови дерев рішень. [7] Дерева рішень у машинному навчанні використовуються як передбачувані моделі, що відображають знання про об’єкт, які представлені гілками, у множині рішень. Такими моделями оперує даний алгоритм.

Метою алгоритму є розбивка початкової вибірки таким чином, щоб одержати підмножини, що відповідають класам. Даний підхід полягає в побудові дерев рішень для кожного класу окремо. На кожному кроці алгоритму обирається значення змінної, яке розділяє всю множину на дві підмножини. Дана розбивка, буде виконуватися доти, поки не буде побудована підмножина, що містить тільки об’єкти одного класу.

Точна послідовність дій алгоритму описана нижче:

- **Вибір атрибуту для розділення.** Потрібно обрати атрибут, який найкраще розділить набір даних. Це, наприклад, може бути визначено за допомогою індексу Джині (Показник нерівності розподілу деякої величини чисел, що приймає значення між 0 і 1, де 0 означає абсолютну рівність, а 1 позначає повну нерівність), вираховується за такою формулою: $1 - \sum_{i=1}^{c} p_i^2$.

- **Розділення даних навчального набору.** Розділити дані на підмножини відповідно до значень обраного атрибуту
- **Рекурсивна побудова.** Для кожної нової підмножини використовуємо алгоритм рекурсії, повторюючи кроки 1-2.
- **Зупинка.** Алгоритм зупиняється, коли виконано певний критерій зупинки, наприклад, досягнута максимальна глибина дерева або інший параметр.
- **Призначення класів.** Коли дерево побудовано, потрібно призначити класи листям або вузлам дерева.

Гарним прикладом, є навчальний набір даних про фрукти з атрибутами, такими як колір, форма, розмір. Дерево рішень може визначати правила, які дозволяють класифікувати фрукти (яблука, апельсини та інші) на основі цих характеристик. На рисунку 2.1.2 зображений приблизний вигляд дерева ухвалення рішень.

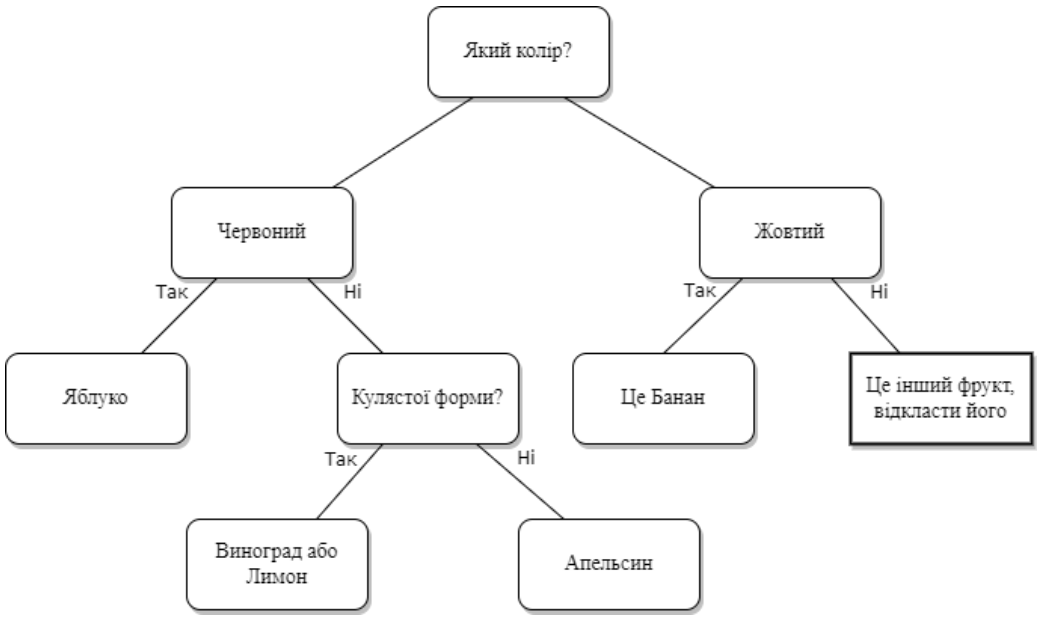


Рисунок 2.1.2 Дерево ухвалення рішень, на прикладі вибірки даних про фрукти

2.2 Порівняння та пошук оптимального алгоритму для вирішення задачі

Проаналізувавши наявні алгоритми класифікації даних, за їх характеристиками, які були описані у попередніх підрозділах, можна порівняти та обрати найкращий серед них для вирішення задачі розподілу абітурієнтів за рейтинговою системою.

Серед характеристик були обрані такі: точність, повнота, час виконання, складність. В таблиці 2.2.1 відображене порівняння цих алгоритмів.

Таблиця 2.2.1 Порівняльна таблиця алгоритмів класифікації

Критерії	Алгоритм побудови дерева рішень	Баєсівський класифікатор	Алгоритм побудови опорних векторів
Точність	85%	80%	90%
Повнота	95%	70%	85%
Час виконання	7 хв.	5 хв.	10 хв.
Складність	O(N*M*log(N))	O(N*M)	O(N^2*M)

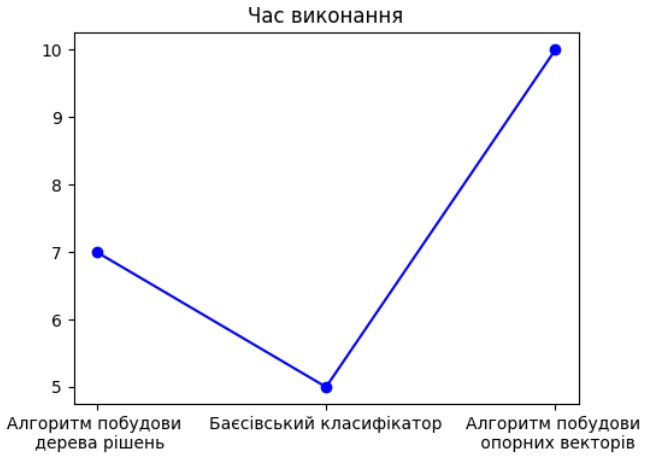


Рисунок 2.2.1 Графік часу виконання алгоритмів

За заданими критеріями керівництва університету(швидкість і точність), можна зробити висновок що найкращим кандидатом є алгоритм побудови дерева рішень. Його точність і час виконання, найкраще підходять під ці стандарти, хоча можна було б віддати перевагу алгоритму побудови опорних векторів, але його складність більше аніж алгоритму побудови дерева рішень.

2.3 Алгоритм розподілу абітурієнтів за рейтинговою системою

Для даного алгоритму, будемо використовувати загальну структуру алгоритму побудови дерева рішень. Його було обрано, за певними характеристиками: 7 хв. час виконання, точність 85%, та складність O(N*M*log(N)). Алгоритм можна поділити, на такий ряд етапів:

- Розділити початкову вибірку на дві, де знаходяться магістри, і бакалаври.
- В кожній з цих вибірок розділити її на умовні класи – код групи та форма навчання(Денна або заочна).
- Групам вказати кількість бюджетних місць, контрактних та загальну.
- Класи наповнити найменшими неділимим об’єктами – абітурієнтами. І відсортовані за рейтинговим балом.
- Пошук абітурієнтів які подали документи на інші спеціальності, та фільтрація у групах, за найвищим пріоритетом.

Алгоритм дуже простий у послідовності дій. На основі цього можна побудувати саме дерево ухвалення рішень, яке представлено на рисунку 2.3.1.

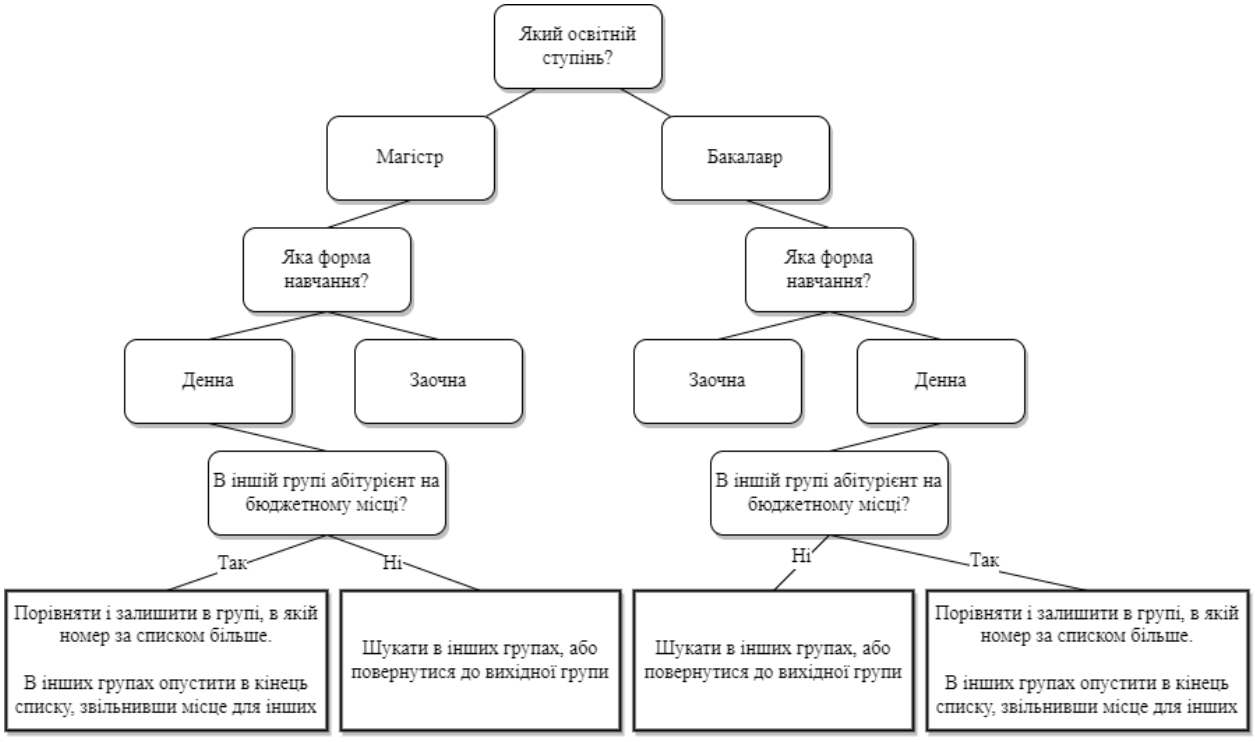


Рисунок 2.3.1 Дерево рішень алгоритму розподілу абітурієнтів

Як видно, із дерева рішень, проходить процес ділення початкової вибірки до тих пір поки не буде побудована підмножина, яка включає в себе об’єкти тільки одного класу – в цьому випадку, академічна група. В самих же групах, виконується розподіл абітурієнтів за рейтинговою системою, шляхом пошуку заявок конкретного абітурієнта на інших спеціальностях з урахуванням пріоритетів. Якщо студент має кращі шанси потрапити на бюджет в іншій групі (Його порядковий номер більше, оскільки більше його рейтинг порівняно з іншими абітурієнтами в цій групі), то в поточній і в усіх інших групах, він стає не активним та опускається в кінець списку, тим самим звільняючи місце для інших вступників. Ці дії повторюються до останньої академічної групи, денної форми навчання.

Таким чином, зберігається основний принцип алгоритму побудови дерева рішень.

2.4 Висновки розділу

В даному розділі, було розглянуто основні характеристики алгоритмів класифікації даних. Проаналізовано, та обрано найоптимальніший з алгоритмів за характеристиками, який задовільнив вимоги та дав змогу вирішити задачу розподілу абітурієнтів за рейтинговою системою. Також алгоритм було детально розписано, та проілюстровано за допомогою дерева рішень. На цьому етапі можна проектувати програмний комплекс, та імплементувати в нього алгоритм, що і буде описано в наступному розділі.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ АЛГОРИТМУ «РОЗПОДІЛУ АБІТУРІЄНТІВ» В ПРОГРАМНОМУ КОМПЛЕКСІ

3.1 Інструментарій для роботи

Для втілення роботи алгоритму використовується об’єктно-орієнтована мова C# та інтегрована середа розробки Visual Studio. Причиною вибору такої мови програмування, став набір бібліотек Microsoft Interop для взаємодії з програмами Microsoft Office, такими як Word чи Excel. Головною ідеєю цієї бібліотеки є забезпечення доступу до існуючих компонентів COM без необхідності модифікації оригінальних компонентів. Дана бібліотека дає змогу зробити типи .Net еквівалентними типам COM.

В рамках, принципів об’єктно орієнтованого програмування (інкапсуляцію, наслідування, поліморфізм), використовується шаблон програмування Модель-Вид-Контролер (Model View Controller). MVC – це архітектурний патерн в програмуванні, який дозволяє відокремити представлення, логіку та управління один від одного. Цей патерн використовується для побудови гнучких та легко змінюваних систем. Основна ідея полягає в тому, щоб розділити програму на три взаємодіючі складові:

- **Модель:** Відповідає за обробку даних та логіку програми. Модель представляє собою структуру або класи даних, які використовуються для зберігання та обробки інформації.
- **Подання:** Представляє дані користувачеві та обробляє введення від користувача. Подання може бути графічним інтерфейсом користувача, текстовим виводом, або будь-яким іншим способом відображення інформації.
- **Контролер:** Обробляє введення від користувача та забезпечує взаємодію між Моделлю та Поданням.

Ці компоненти взаємодіють таким чином:

- Користувацький інтерфейс генерує введення.
- Контролер обробляє введення та взаємодіє з моделлю.
- Модель обробляє логіку додатку та оновлює свій стан.
- Модель також оголошує події або оновлення для контролера та подання.
- Подання відображає дані користувачеві та може відправляти запити до контролера.
- Контролер знову реагує на введення та оновлює модель або подання за необхідності.

На рисунку 3.1.1, зображена схема взаємодії елементів патерну MVC.

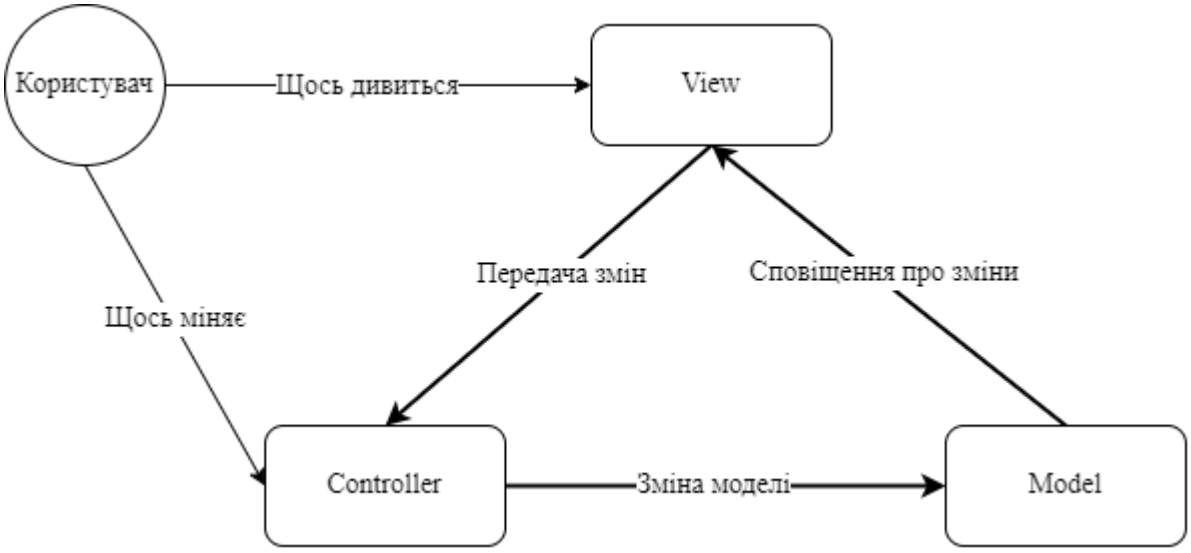


Рисунок 3.1.1 Схема роботи MVC-патерну

Також можна виділити такі переваги використання даного патерну:

- **Відокремлення обов'язків:** Кожен компонент (Модель, Подання, Контролер) відповідає лише за свою область відповідальностей, що сприяє модульності та легкості розширення.
- **Повторне використання коду:** Через відокремлення компонентів, можливо використовувати їх незалежно один від одного.
- **Підтримка різних інтерфейсів:** Ви можете легко змінити візуальне подання чи взаємодіяти з даним без зміни логіки додатку.

Таким чином, Visual Studio, як продукт Microsoft, дає змогу повноцінно використовувати функціонал мови програмування C#, написаний тими ж працівниками Microsoft, що в свою чергу полегшує процес роботи з бібліотекою Microsoft Interop. І все це, в цілому вирішує проблему обширності поставленої задачі за допомогою моделі MVC, яка дає змогу швидко вносити зміни в процес побудови програмного забезпечення.

Ще одним, не менш важливим інструментарієм для реалізації алгоритму, стали вбудовані методи, типи даних та концепції мови програмування C#, а саме списки, LINQ, паралельне програмування(асинхронність). Це і стало ще одним аргументом на користь вибору C#, в якості мови написання програмного комплексу.

Як було сказано, у вступі, алгоритми тісно пов’язані зі структурами даних. Однією з таких структур, є список, що і стало основою для зберігання великої кількості даних. Список, або List, це тип колекції, який представляє собою динамічний масив, що може змінюватись за розміром, під час виконання програми. У списках зберігаються всі дані про абітурієнтів, що дає змогу просто та ефективно взаємодіяти з ними за допомогою певних функцій. Нижче наведений список основних характеристик списків:

- **Динамічний розмір:** Списки можуть змінюватись за розміром динамічно, додаючи чи видаляючи елементи.
- **Висока продуктивність:** Доступ до елементів у списку відбувається за константний час (O(1)), що робить його ефективним для багатьох ситуацій використання.
- **Гнучкість:** Списки можуть зберігати елементи різних типів, і вони можуть бути змінені, без великих зусиль.
- **Методи для операцій:** Списки мають ряд вбудованих методів для виконання різних операцій, таких як сортування, фільтрація, копіювання тощо.

Остання характеристика дуже важлива, оскільки окрім стандартних методів і функцій для взаємодії зі списками, було використано і набір функцій LINQ.

LINQ (Language Integrated Query) – це компонент C#, який дозволяє виконувати структурований запит до різноманітних джерел даних таких як масиви, колекції, бази даних і звісно списків. LINQ включає в себе набір операторів та методів розширення для написання запитів в стилі SQL прямо в коді C#. Основні компоненти LINQ, включають в себе: LINQ to Objects, LINQ to SQL, LINQ to XML, LINQ to Entities, LINQ to DataSet. Для вирішення ж поточно ї задачі, було використано LINQ to Objects, який в собі має набір функцій та операторів для роботи зі списками. Наприклад, пошук абітурієнтів за ім’ям, чи інші функції фільтрації.

І останнім, з інструментарію, якому варто приділити увагу – це асинхронність, як частина паралельного програмування. Асинхронність – це механізм, який дозволяє виконувати асинхронні(тобто, не чекати завершення) операції. Це особливо в сучасних додатках, де велика частина роботи пов’язана з операціями вводу-виводу, мережевими викликами, анімацією та іншими завданнями, які можуть викликати блокування, але не повинні зупиняти весь потік виконання програми. Переваги асинхронності, наведені нижче:

- **Покращення продуктивності:** Асинхронні операції, дозволяють виконувати інші роботи під час очікування завершення операцій вводу-виводу. Це покращує реактивність програми та загальний час відгуку.
- **Ефективне використання ресурсів:** Оскільки потік не блокується під час очікування завершення операцій, це дозволяє оптимально використовувати системні ресурси.
- **Покращення користувацького інтерфейсу:** В асинхронних додатках користувачі можуть продовжувати працювати з інтерфейсом користувача, навіть якщо деякі завдання ще не завершено.

При вирішенні ж поточної задачі побудови алгоритму, цей механізм, допоміг реалізувати асинхронний пошук та редагування даних абітурієнтів перед тим як будуть сформовані остаточні списки на вступ.

3.2 Розробка програмного комплексу для демонстрації роботи алгоритму розподілу абітурієнтів

Для розуміння побудови програмного комплексу, алгоритм потрібно представити у вигляді блок-схеми. (Див. Рисунок А.1). Блок вхідних і вихідних даних позначено паралелограмом, блок обчислень даних – прямокутником, блок прийняття рішень – ромбом, а еліпсом – початок та кінець алгоритму.

З початку, список абітурієнтів ділиться на два списки бакалаврів та магістрів, за допомогою структурованих запитів LINQ. Так само проходить другий раз розбиття списку на денну і заочну форму навчання.

Лістинг коду Б.1

Групування абітурієнтів за освітньо-кваліфікаційним рівнем

```
//Вибірка тільки Магістрів
var magistr_list = list_student.Where(m => m.okr == "Магістр").ToList();

if (magistr_list.Count == 0)
   return;

//Групування за спеціальностями
var magistr_spec_list = magistr_list.GroupBy(s => s.spec).ToList().OrderBy(g => g.Key).ToList();
```

Під час цього також, формується кількість місць в групі за розділенням, на контрактні та бюджетні місця, не враховуючи заочну форму навчання, оскільки там є тільки загальна кількість місць в групі. Дані отримуються за допомогою діалогового вікна, на початкових етапах роботи зі списками, в обробнику кнопки відправити – BtnAcc_Places_Click( ), даної форми, передаються дані до основного методу GetMagistrList ( ).

Лістинг коду Б.2

Отримання вхідних даних про кількість місць у групі

```
private void BtnAcc_Places_Click(object sender, EventArgs e)
        {
            var magistr_list = list_student.Where(m => m.okr == "Магістр").ToList();
            var magistr_spec_list = magistr_list.GroupBy(s => s.spec).ToList().OrderBy(g => g.Key).ToList();
            string spec_name = "";
            foreach (var curSpec in magistr_spec_list)
            {
                if (curSpec.Key != "")
                {
                    if (curSpec.Key.Length <= 30)
                    {
                        spec_name = curSpec.Key;
                    }
                    else
```

```
                    spec_name = curSpec.Key.Substring(0, 30);
                    spec_name = GetSpecCode(spec_name);
                    list_groups_places.Add(Tuple.Create(spec_name,
                            Int32.Parse(Form_Count_Places.Controls["TextBox_B_" +
spec_name].Text),
                            Int32.Parse(Form_Count_Places.Controls["TextBox_C_" +
spec_name].Text)));
                }
            }
            Form_Count_Places.Close();
        }
```

Наступним кроком, створюємо академічні групи, та вносимо в них дані про кількість бюджетних та контрактних місць. Далі, абітурієнтів за формою навчання та спеціальністю додаємо до певної групи, і при наповненні групи до кінця, фільтруємо її за рейтинговим балом. Останньою дією, на цьому етапі є фільтрування людей, виділенням кольору в кінцевому звіті, хто за рейтинговим балом потрапив на бюджет, а хто на контракт.

Лістинг коду Б.1

Створення нової академічної групи

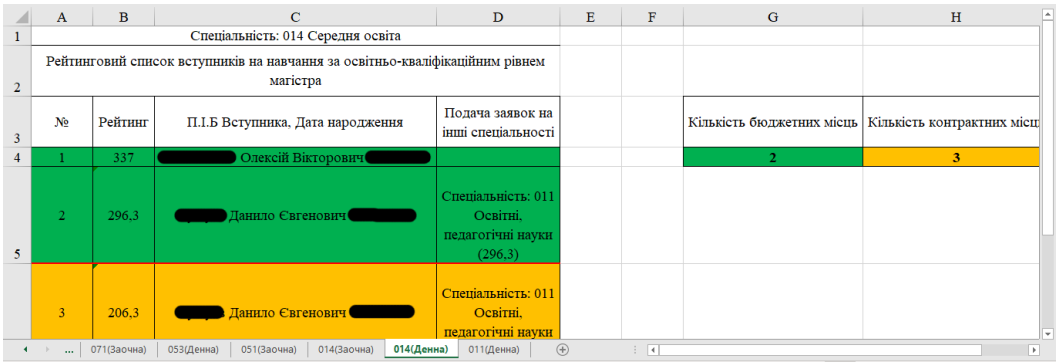
```
//Створення нової вкладки
Excel.Worksheet ExWS = ExWB.Worksheets.Add();
string title = "";
       if (curSpec.Key != "")
       {
          if (curSpec.Key.Length <= 30)
          {
             title = curSpec.Key;
          }
          else
             title = curSpec.Key.Substring(0, 30);
             title = GetSpecCode(title);
          }
ExWS.Name = title + "(" + "Денна" + ")";//Назва вкладки, код спеціальності
```

Лістинг коду Б.1

Пошук абітурієнтів в інших академічних групах

```
//Пошук абітурієнтів на інших спеціальностях
string groups = "";
var group_stud_list = magistr_list.Where(m => m.name == curStud.name).OrderByDescending(o
=> o.rating).ToList();
foreach (var cur_stud in group_stud_list)
{
        if (cur_stud.spec != curSpec.Key)
        {
           groups = groups + Environment.NewLine + "Спеціальність: " + cur_stud.spec + "
(" + cur_stud.rating.ToString() + ")";
        }
}
```

Останнім кроком, як зображено на блок-схемі, виконується формування особистого списку студента де зазначені всі його подачі на інші спеціальності, в яких він на бюджетному місці. Далі цей список фільтрується за найбільшим рейтинговим балом, та місцем в цих групах. І абітурієнта лишають в найвигіднішій для нього групі, на бюджеті. В усі інших групах абітурієнт опускається в кінець списку, теоретично звільняючи місце для інших абітурієнтів. На рисунку 3.2.1 зображений результат кінцевого звіту



	A	B	C	D	E	F	G	H
1	Спеціальність: 014 Середня освіта							
2	Рейтинговий список вступників на навчання за освітньо-кваліфікаційним рівнем магістра							
3	№	Рейтинг	П.І.Б Вступника, Дата народження	Подача заявок на інші спеціальності			Кількість бюджетних місць	Кількість контрактних місц
4	1	337	[redacted] Олексій Вікторович [redacted]				2	3
5	2	296,3	[redacted] Данило Євгенович [redacted]	Спеціальність: 011 Освітні, педагогічні науки (296,3)				
	3	206,3	[redacted] Данило Євгенович [redacted]	Спеціальність: 011 Освітні, педагогічні науки				

... | 071(Заочна) | 053(Денна) | 051(Заочна) | 014(Заочна) | 014(Денна) | 011(Денна)

Рисунок 3.2.1 Звітність за однією з груп магістратури

3.3 Висновки розділу

Підсумовуючи, в розділі було викладено детально про весь інструментарій, який використовувався для демонстрації роботи алгоритму, проаналізовано його переваги порівняно з іншими. Надано

Було розглянуто детально схему побудови блок-схеми алгоритму, для подальшого полегшення його програмування. Також в останньому підрозділі, були наведені приклади програмного коду реалізації найважливіших частин програмного комплексу, які відповідають за роботу алгоритму.

ВИСНОВКИ

Під час дослідницької роботи, було розглянуто основні поняття алгоритмізації, класифікації даних та їх метрики. Проведено аналіз та пошук оптимального алгоритму для вирішення поставленої задачі.

Результатом дослідження, є побудова алгоритму розподілу абітурієнтів за рейтинговою системою, який в свою чергу має на меті полегшити роботу працівників вступної кампанії, шляхом автоматизації та оптимізації процесів розподілу абітурієнтів за рейтинговою системою.

Використовуючи результати дослідження, було практично продемонстровано роботу алгоритму, засобами програмного комплексу написаного мовою програмування C#.

Підсумовуючи загалом, даний алгоритм який розроблений на основі алгоритму класифікації побудови дерева рішень показав себе ефективно для обробки даних великої кількості, що дає змогу використовувати його не тільки в цій конкретній задачі, а й в загалом інших задача де потрібно ефективно оброблювати велику кількість даних, та проводити над ними певні дії.

ПЕРЕЛІК ПОСИЛАНЬ

1. Fuegi, J. and Francis, J. «Lovelace & Babbage and the creation of the 1843 'notes'.» Annals of the History of Computing 25 #4 (October-December 2003)
2. Додаток до Умов прийому на навчання до закладів вищої освіти України, 2018. – 10 с. – (Міністерство освіти України). – [Режим доступу до дод.: https://mon.gov.ua/storage/app/media/vyshcha/vstup-2018/veliky-umovi/8-dodatok-6.pdf]
3. Donald E. Knuth. "Volume 1: Fundamental Algorithms. Third Edition". Reading, Massachusetts: Addison-Wesley, 1997. 650 pages. ISBN 0-201-89683-4
4. Nadeau, C., & Bengio, Y. "Inference for the generalization error". Machine learning, 52(3), 2003. – 239-281 pages [Режим доступу до статті: https://link.springer.com/article/10.1023/A:1024068626366]
5. Cortes, C.; Vapnik, V. (1995). Support-vector networks. Machine Learning 20 (3): 273–297 [Режим доступу до статті: https://link.springer.com/article/10.1007/BF00994018]
6. Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001, 7 pages Workshop on Empirical Methods in Artificial Intelligence. [Режим доступу до статті: https://web.archive.org/web/20120302175421/http://www.research.ibm.com/people/r/rish/papers/RC22230.pdf]
7. Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. "Classification and Regression Trees". Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. ISBN 978-0-412-04841-8, 1984. – 368 с.

ДОДАТКИ

Додаток А

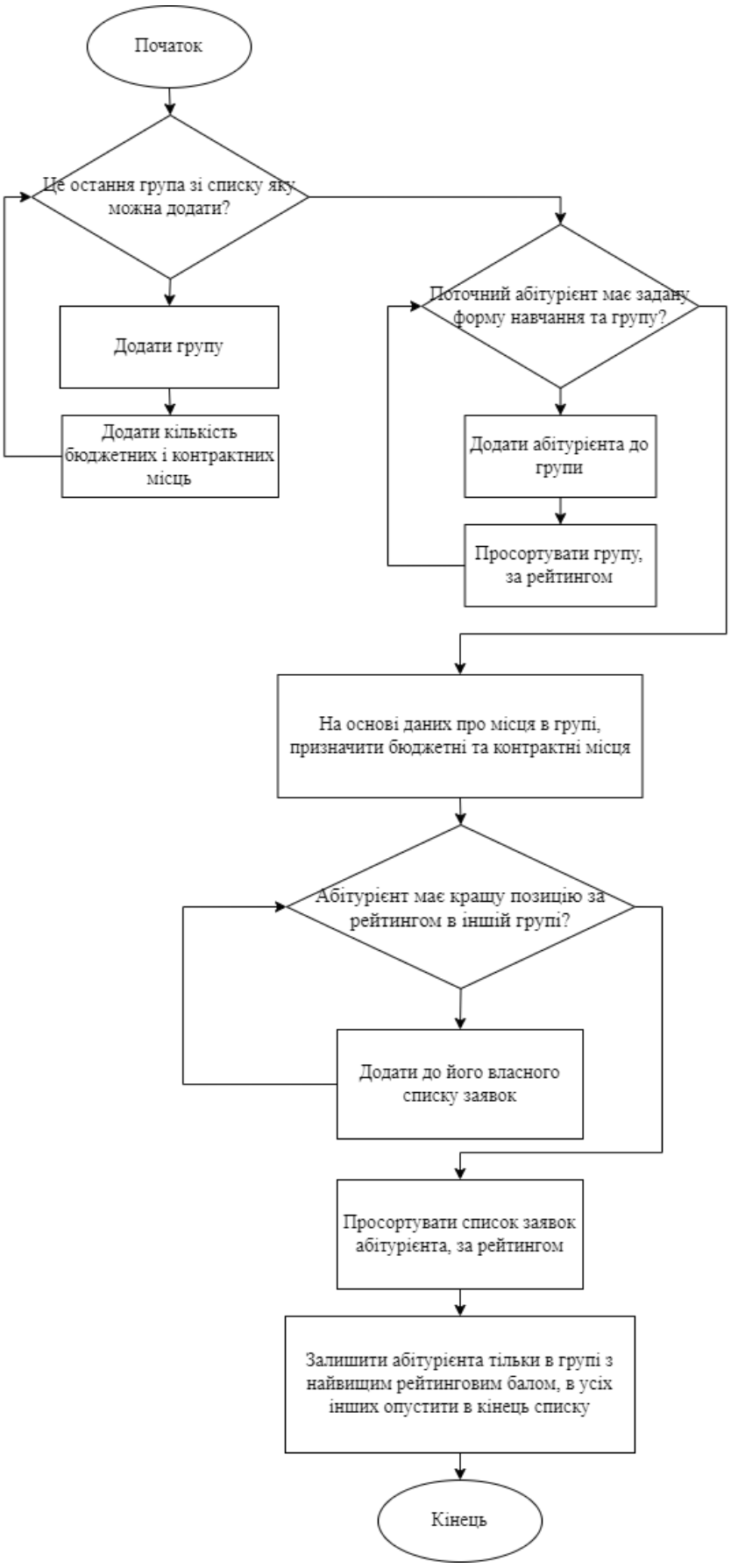


Рисунок А.1 Блок схема алгоритму розподілу абітурієнтів

Лістинг коду Б.1 Метод обробки списку магістрів

```
private void GetMagistrList()
      {
         if (ConnectFile)
         {
            if (File.Exists(Str_FileName))
            {
               try
               {
                  Excel.Application ExApp = new Excel.Application();

                  //Вибірка тільки Магістрів
                  var magistr_list = list_student.Where(m => m.okr == "Магістр").ToList();
                  if (magistr_list.Count == 0)
                     return;
                  //Групування за спеціальностями
                  var magistr_spec_list = magistr_list.GroupBy(s => s.spec).ToList().OrderBy(g => g.Key).ToList();
                  Form_Count_Places.Text = "My Test Automaticly Create Form";
                  Form_Count_Places.AutoScroll = true;
                  Form_Count_Places.Width = 470;

                  Label spec_label_b = new Label();
                  Label spec_label_c = new Label();

                  Form_Count_Places.Controls.Add(spec_label_b);
                  Form_Count_Places.Controls.Add(spec_label_c);

                  spec_label_b.Location = new System.Drawing.Point(55, 10);
```

```
spec_label_b.Text = "Кількість бюджетних місць";
spec_label_b.Name = "spec_label_b";
spec_label_b.Size = new System.Drawing.Size(180, 17);
spec_label_b.TextAlign = ContentAlignment.MiddleCenter;

spec_label_c.Location = new System.Drawing.Point(240, 10);
spec_label_c.Text = "Кількість контрактних місць";
spec_label_c.Name = "spec_label_c";
spec_label_c.Size = new System.Drawing.Size(180, 17);
spec_label_c.TextAlign = ContentAlignment.MiddleCenter;

int left_label = 10;
int top_label = 30;

int left_textb = 55;
int top_textb = 30;

int left_textc = 240;
int top_textc = 30;
string spec_title = "";
int textb_value = 0;
foreach (var curSpec in magistr_spec_list)
{
   if (curSpec.Key != "")
   {
      if (curSpec.Key.Length <= 30)
      {
         spec_title = curSpec.Key;
      }
      else
         spec_title = curSpec.Key.Substring(0, 30);
      spec_title = GetSpecCode(spec_title);}
```

```
                Label spec_label = new Label();

                TextBox spec_textb_budget = new TextBox();
                TextBox spec_textb_contract = new TextBox();

                Form_Count_Places.Controls.Add(spec_label);
                Form_Count_Places.Controls.Add(spec_textb_budget);
                Form_Count_Places.Controls.Add(spec_textb_contract);

                spec_label.Location = new System.Drawing.Point(left_label,
top_label);
                spec_label.Text = spec_title;
                spec_label.Name = "Label_" + spec_title;
                spec_label.Size = new System.Drawing.Size(32 , 17);
                top_label += 25;

                spec_textb_budget.Location = new
System.Drawing.Point(left_textb, top_textb);
                spec_textb_budget.Name = "TextBox_B_" + spec_title;
                spec_textb_budget.Size = new System.Drawing.Size(180, 20);
                spec_textb_budget.KeyPress += new
KeyPressEventHandler(TextBox_BC_KeyPress);
                spec_textb_budget.Text = textb_value.ToString();
                textb_value++;
                top_textb += 25;

                spec_textb_contract.Location = new
System.Drawing.Point(left_textc, top_textc);
                spec_textb_contract.Name = "TextBox_C_" + spec_title;
                spec_textb_contract.Size = new System.Drawing.Size(180, 20);
                spec_textb_contract.KeyPress += new
KeyPressEventHandler(TextBox_BC_KeyPress);
```

```
                spec_textb_contract.Text = textb_value.ToString();
                textb_value++;
                top_textc += 25;
            }
            var textb_last = Form_Count_Places.Controls["TextBox_C_" + spec_title];
            var textb_last_top_pos = textb_last.Location.Y + 35;

            Button button_acc_places = new Button();
            Button button_can_places = new Button();

            button_acc_places.Location = new System.Drawing.Point(240, textb_last_top_pos);
            button_acc_places.Text = "Підтвердити";
            button_acc_places.Name = "BtnAcc_Places";
            button_acc_places.Size = new System.Drawing.Size(95, 25);
            button_acc_places.Click += new EventHandler(BtnAcc_Places_Click);

            button_can_places.Location = new System.Drawing.Point(345, textb_last_top_pos);
            button_can_places.Text = "Відміна";
            button_can_places.Name = "BtnCan_Places";
            button_can_places.Size = new System.Drawing.Size(75, 25);

            Form_Count_Places.Controls.Add(button_acc_places);
            Form_Count_Places.Controls.Add(button_can_places);

            Form_Count_Places.ShowDialog();

            ExApp = new Excel.Application();
            Excel.Workbook ExWB = ExApp.Workbooks.Add();
```

```
                //Вкладки абітурієнтів
                if (magistr_spec_list.Count > 0)
                {
                    Form_Elements_Invoke(label1, null, null, () => label1.Visible =
true);
                    Form_Elements_Invoke(null, progressBar1, null, () =>
progressBar1.Visible = true);
                    Form_Elements_Invoke(label2, null, null, () => label2.Visible =
true);
                    Form_Elements_Invoke(label1, null, null, () => label1.Text =
"Сортування");
                    Form_Elements_Invoke(null, progressBar1, null, () =>
progressBar1.Maximum = magistr_spec_list.Count);
                    Form_Elements_Invoke(null, progressBar1, null, () =>
progressBar1.Value = 0);
                    foreach (var curSpec in magistr_spec_list)
                    {
                        var curspec_list = curSpec.Select(s => s).ToList();
                        var intramural_list = curspec_list.Where(m => m.form_educ ==
"Денна").ToList();//Денна форма навчання
                        var inabsentia_list = curspec_list.Where(m => m.form_educ ==
"Заочна").ToList();//Заочна форма навчання

                        if (intramural_list.Count > 0)//Денна форма
                        {
                            //Створення нової вкладки
                            Excel.Worksheet ExWS = ExWB.Worksheets.Add();
                            string title = "";
                            if (curSpec.Key != "")
                            {
                                if (curSpec.Key.Length <= 30)
```

```
                    {
                        title = curSpec.Key;
                    }
                    else
                        title = curSpec.Key.Substring(0, 30);
                    title = GetSpecCode(title);
                }
                ExWS.Name = title + "(" + "Денна" + ")";//Назва вкладки, код спеціальності

                //Заповнення аркушу абітурієнтами
                var studList = curSpec.Select(s => s).OrderByDescending(r => r.rating).ToList();
                if (studList.Count > 0)
                {
                    int num = 0;
                    int contracts = 1;
                    ColorConverter cc = new ColorConverter();
                    foreach (var curStud in studList)
                    {
                        num++;
                        ExWS.Cells[num + 1, 1].Value = num.ToString();
                        ExWS.Cells[num + 1, 2].Value = curStud.rating.ToString();
                        ExWS.Cells[num + 1, 3].Value = curStud.name;
                        ExWS.Cells[num + 1, 1].Borders.Weight = 2d;
                        ExWS.Cells[num + 1, 2].Borders.Weight = 2d;
                        ExWS.Cells[num + 1, 3].Borders.Weight = 2d;

                        //Пошук абітурієнтів на інших спеціальностях
                        string groups = "";
```

```
                        var group_stud_list = magistr_list.Where(m => m.name
== curStud.name).OrderByDescending(o => o.rating).ToList();
                        foreach (var cur_stud in group_stud_list)
                        {
                            if (cur_stud.spec != curSpec.Key)
                            {
                                groups = groups + Environment.NewLine +
"Спеціальність: " + cur_stud.spec + " (" + cur_stud.rating.ToString() + ")";
                            }
                        }
                        ExWS.Cells[num + 1, 4].Value = groups;
                        ExWS.Cells[num + 1, 4].Borders.Weight = 2d;
                        foreach (var currGroup in list_groups_places)
                        {
                            if (currGroup.Item1 == title)
                            {
                                ExWS.Cells[2, 7].Value = currGroup.Item2;
                                ExWS.Cells[2,          7].Interior.Color          =
ColorTranslator.ToOle((Color)cc.ConvertFromString("#00B050"));
                                ExWS.Cells[2, 8].Value = currGroup.Item3;
                                ExWS.Cells[2,          8].Interior.Color          =
ColorTranslator.ToOle((Color)cc.ConvertFromString("#FFC000"));
                            }
                        }
                        if (num <= ExWS.Cells[2, 7].Value)
                        {
                            ExWS.Range["A" + (num + 1) + ":D" + (num +
1)].Interior.Color                                                    =
ColorTranslator.ToOle((Color)cc.ConvertFromString("#00B050"));
                        }
```

```
                    if (num > ExWS.Cells[2, 7].Value && contracts <=
ExWS.Cells[2, 8].Value)
                    {
                        if (contracts == 1 && ExWS.Cells[2, 7].Value > 0)
                        {
                            ExWS.Range["A" + (num + 1) + ":D" + (num +
1)].Borders[Excel.XlBordersIndex.xlEdgeTop].Weight = 3d;
                            ExWS.Range["A" + (num + 1) + ":D" + (num +
1)].Borders[Excel.XlBordersIndex.xlEdgeTop].Color =
System.Drawing.ColorTranslator.ToOle(System.Drawing.Color.Red);
                        }
                        contracts++;
                        ExWS.Range["A" + (num + 1) + ":D" + (num +
1)].Interior.Color =
ColorTranslator.ToOle((Color)cc.ConvertFromString("#FFC000"));
                    }

                    Form_Elements_Invoke(label2, null, null, () =>
label2.Text = "Абітурієнт " + num + " / " + studList.Count());
                }
            }
//Зовнішній вигляд сторінки
            ExApp.StandardFont = "Times New Roman";
            ExApp.StandardFontSize = 14;

            ExWS.Range["A1:D1"].Merge(true);
            ExWS.Cells[1, 1].Value = "Спеціальність: " +
curSpec.Key.ToString();
            ExWS.Range["A1:D1"].VerticalAlignment =
Excel.XlVAlign.xlVAlignCenter;
            ExWS.Range["A1:D1"].HorizontalAlignment =
Excel.XlHAlign.xlHAlignCenter;
```

```
                ExWS.Rows[2].Insert();
                ExWS.Range["A2:D2"].Merge(true);
                ExWS.Cells[2, 1].Value = "Рейтинговий список вступників на навчання за освітньо-кваліфікаційним рівнем магістра";
                ExWS.Range["A2:D2"].VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;
                ExWS.Range["A2:D2"].HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
                ExWS.Range["A2:D2"].WrapText = true;
                ExWS.Range["A2:D2"].RowHeight = 50;
                ExWS.Rows[3].Insert();

                ExWS.Cells[3, 1].Value = "№";
                ExWS.Cells[3, 2].Value = "Рейтинг";
                ExWS.Cells[3, 3].Value = "П.І.Б Вступника, Дата народження";
                ExWS.Cells[3, 4].Value = "Подача заявок на інші спеціальності";
                ExWS.Cells[3, 7].Value = "Кількість бюджетних місць";
                ExWS.Cells[3, 8].Value = "Кількість контрактних місць";
                ExWS.Range["G:H"].VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;
                ExWS.Range["G:H"].HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
                ExWS.Range["G:H"].Rows[3].Borders.Weight = 2d;
                ExWS.Range["G:H"].Rows[4].Borders.Weight = 2d;
                ExWS.Range["G:H"].Rows[4].Font.Bold = true;
                ExWS.Columns["G"].AutoFit();
                ExWS.Columns["H"].AutoFit();
                ExWS.Columns["C"].AutoFit();
                ExWS.Columns["D"].ColumnWidth = 17;
```

```
                    ExWS.Range["A:D"].Rows[3].Borders.Weight = 2d;
                }
Form_Elements_Invoke(null, progressBar1, null, () => progressBar1.Value = progressBar1.Maximum);
                }
                Form_Elements_Invoke(null, progressBar1, null, () => progressBar1.Value = progressBar1.Maximum);
                if (ExWB.Sheets.Count > 1)
                    ExWB.Sheets[ExWB.Sheets.Count].Delete();
                }

                //Додаткова інформація для користувача
                MessageBox.Show("Сортування завершено", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

                Form_Elements_Invoke(label1, null, null, () => label1.Visible = false);
                Form_Elements_Invoke(null, progressBar1, null, () => progressBar1.Visible = false);
                Form_Elements_Invoke(label2, null, null, () => label2.Visible = false);

                //Збереження у файл
                string newFileName = Path.GetDirectoryName(Str_FileName) + "\\" + Path.GetFileNameWithoutExtension(Str_FileName) + "_магистр.xlsx";
                ExWB.SaveAs(newFileName, Microsoft.Office.Interop.Excel.XlFileFormat.xlWorkbookDefault,
                    Type.Missing, Type.Missing, false, false, Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlNoChange,
                    Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing);
```

```
                //Закриття процесу
                GC.Collect();
                GC.WaitForPendingFinalizers();

                ExWB.Close();
                Marshal.ReleaseComObject(ExWB);
                ExApp.Quit();
                Marshal.ReleaseComObject(ExApp);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            Cursor.Current = Cursors.Default;
        }
        else
        {
            Form_Elements_Invoke(label3, null, null, () => label3.Text = "Файл не
знайден");
            Form_Elements_Invoke(label3, null, null, () => label3.Visible = true);
            Form_Elements_Invoke(label3, null, null, () => label3.ForeColor =
Color.Red);
        }
    }
    else
    {
Form_Elements_Invoke(label3, null, null, () => label3.Text = "Файл не обрано");
Form_Elements_Invoke(label3, null, null, () => label3.Visible = true);
Form_Elements_Invoke(label3, null, null, () => label3.ForeColor = Color.Red);
    }};
```

Лістинг коду Б.2 Метод передачі інформації, про кількість місць в групі

```
private void BtnAcc_Places_Click(object sender, EventArgs e)
      {
         var magistr_list = list_student.Where(m => m.okr == "Магістр").ToList();
         var magistr_spec_list = magistr_list.GroupBy(s => s.spec).ToList().OrderBy(g => g.Key).ToList();
         string spec_name = "";
         foreach (var curSpec in magistr_spec_list)
         {
            if (curSpec.Key != "")
            {
               if (curSpec.Key.Length <= 30)
               {
                  spec_name = curSpec.Key;
               }
               else
                  spec_name = curSpec.Key.Substring(0, 30);
               spec_name = GetSpecCode(spec_name);
               list_groups_places.Add(Tuple.Create(spec_name,
                     Int32.Parse(Form_Count_Places.Controls["TextBox_B_" + spec_name].Text),
                     Int32.Parse(Form_Count_Places.Controls["TextBox_C_" + spec_name].Text)));
            }
         }
         Form_Count_Places.Close();
         }
```