МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

Інформатики та програмної інженерії

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з дисципліни «Аналіз даних в інформаційних системах»

(назва дисципліни)

на тему: «Прогнозування успішності фентезі літератури за кількістю різноманітних угрупувань. Порівняння логістичної регресії і дерева прийняття рішень для прогнозування типу фентезі»

Студента 2 курсу, групи ІП-01

Спеціальності 121 «Інженерія програмного забезпечення»

Храмченко Анатолія Сергійовича

«ПРИЙНЯВ» з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис Дата

Київ- 2022 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційних системах

Спеціальність 121 «Інженерія програмного забезпечення»

Курс 2 Група ІП-01 Семестр 4

**ЗАВДАННЯ**

на курсову роботу студента

Храмченко А.С.

(прізвище, ім’я, по батькові)

1. Тема роботи Прогнозування успішності фентезі літератури за кількістю різноманітних угрупувань. Порівняння логістичної регресії і дерева прийняття рішень для прогнозування типу фентезі

2. Строк здачі студентом закінченої роботи 19.06.2022

3. Вхідні дані до роботи методичні вказівки до курсової роботи, дані з сайтів

<https://en.wikipedia.org>

https://www.litres.ru

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

Постановка задачі

Аналіз предметної області

Розробка сховища даних

Інтелектуальний аналіз даних

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

6. Дата видачі завдання 16.04.2022

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 16.04.22 |  |
| 2. | Визначення зовнішніх джерел даних | 25.04.22 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 02.05.22 |  |
| 4. | Розробка моделі сховища даних | 19.05.22 |  |
| 6. | Обґрунтування методів аналізу даних | 30.05.22 |  |
| 5. | Застосування та порівняння ефективності методів інтелектуального аналізу | 12.06.22 |  |
| 6. | Підготовка пояснювальної записки | 19.06.22 |  |
| 7. | Здача курсової роботи на перевірку | 19.06.22 |  |
| 8. | Захист курсової роботи | 19.06.22 |  |

Студент Храмченко А.С.

(підпис) (прізвище, ім’я, по батькові)

Керівник доц. Ліхоузова Т.А.

(підпис) (прізвище, ім’я, по батькові)

Керівник доц. Олійник Ю.О.

(підпис) (прізвище, ім’я, по батькові)

"16" березня 2022 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 121 сторінка, 10 рисунків, 28 таблиць, 4 посилання.

Об’єкт дослідження: інтелектуальний аналіз даних

Мета дослідження: створення програмного забезпечення для для аналізу даних подальшим прогнозуванням

Дана курсова робота включає в себе: опис підготовки даних для заповнення бази даних, опис створення програмного забезпечення для інтелектуального аналізу даних та прогнозування за допомогою різних моделей.

МОДЕЛЬ ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ЛОГІСТИЧНА РЕГРЕСІЯ, ДЕРЕВО ПРИЙНЯТТЯ РІШЕНЬ, ГРАДІЄНТНИЙ СПУСК, АНАЛІЗ ТЕКСТУ

Зміст

[Вступ 5](#_Toc106540974)

[1 Постановка задачі 6](#_Toc106540975)

[2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7](#_Toc106540976)

[3 ЗАПОВНЕННЯ СХОВИЩА ДАНИХ 8](#_Toc106540977)

[3.1. Збір даних 8](#_Toc106540978)

[3.2. Аналіз опису книги 8](#_Toc106540979)

[3.3. Занесення даних до сховища 9](#_Toc106540980)

[3.4. Отримання даних зі сховища 9](#_Toc106540981)

[4 Тестування програмного забезпечення 11](#_Toc106540982)

[4.1. План тестування 11](#_Toc106540983)

[4.2. Приклади тестування 12](#_Toc106540984)

[5 Інструкція користувача 28](#_Toc106540985)

[5.1. Робота з програмою 28](#_Toc106540986)

[5.2. Формат вхідних даних 33](#_Toc106540987)

[5.3. Системні вимоги 33](#_Toc106540988)

[Висновки 35](#_Toc106540989)

[Перелік посилань 36](#_Toc106540990)

[Додаток А Технічне завдання 60](#_Toc106540991)

[Додаток Б Тексти програмного коду 63](#_Toc106540992)

Вступ

Фентезі – один з найпопулярніших жанрів книг. Успішність книги залежить від багатьох факторів: кількості рас, кількості сюжетних поворотів, кількості персонажів та видів в’язків між ними. При написанні книги автору дуже важливо враховувати всі ці аспекти, оскільки від цього залежить кількість людей, що прочитають книгу, а як наслідок – прибутковість. Також, автору потрібно знати, чи відповідає книга обраному піджанру. Наприклад, якщо автор хоче, щоб книга була орієнтована на дітей, то він повинен прагнути жанру світлого фентезі. Для цього йому потрібно перевірити жанр, якому відповідає його книга «стороннім оком».

Отже, нами було вирішено створити програмне забезпечення, що допоможе вирішити ці проблеми. А саме, на основі даних про кількість сторінок, рік виходу та короткому опису сюжету книги прогнозувати кількість читачів та визначити жанр.

В рамках даної курсової роботи було розроблено сховище даних на основі фізичної моделі бази даних, функціонал якої було розроблено за допомогою SQL скриптів та програмне забезпечення, що заповнює базу даних, отримує дані з неї, проводить інтелектуальний аналіз даних та прогнозує результат.

В ролі системи керування сховищем даних для даної роботи буде виступати MySQL, а мова програмування для реалізації застосунку – Python3.

# Постановка задачі

Під час виконання курсової роботи потрібно виконати наступні завдання:

Створення застосунку, що на основі даних, що містяться у файлах, аналізує їх текст та заповнює базу даних.

Створення застосунку, що отримує дані з бази даних та проводить інтелектуальний аналіз способом логістичної регресії за допомогою градієнтного спуску та способом дерева рішень.

Створення застосунку, що тестує моделі прогнозування.

Проаналізувати параметри на значемість та вирівняти дані за допомогою SMOTE.

()

Використати мову програмування Python 3 для реалізації застосунку.

Курсовий проект здати до дедлайну (початок сесії) та виконати у єдиному стилі написання коду (coding style).

# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Сьогодні фентезі є актуальним жанром книжок. Жанр фентезі почав активно розвиватися лише у XX ст., а отже багато письменників бажають його доповнити своїми творами. Написання такої книги потребує багато зусиль, а отже автору потрібно знати, на яку кількість аудиторії його книга може бути розрахована. А також важливо на який вік орієнтована книга. Якщо автор хоче орієнтуватися на більш дорослу аудиторію, то йому потрібно прагнути до жанру темного фентезі. Отже, автору потрібно знати, якому жанру його книга більше відповідає для подальшого виправлення.

У програмному забезпеченні буде реалізовано наступну функціональність:

* Аналіз опису сюжету книг на різноманітні характеристики
* Заповнення бази даних
* Інтелектуальний аналіз даних
* Використання декількох моделей прогнозування даних
* Прогнозування жанру книги
* Прогнозування успішності книги

# ЗАПОВНЕННЯ СХОВИЩА ДАНИХ

## Збір даних

Для виконання курсової роботи ми вирішили зібрати про книгу наступні параметри: Назва, автор, кількість сторінок, рік виходу, оцінка на ресурсі LiveLib, оцінка на ресурсі LitRes, реальність світу (1 – напівреальний, 0 – повністю вигаданий), піджанр (темне чи світле фентезі) та кількість оцінок. А також короткий опис сюжету книги.

Ці дані було отримано з ресурсів:

* <https://en.wikipedia.org> (назва, автор, кількість сторінок, рік виходу)
* <https://www.litres.ru> (оцінки та їх кількість)

Ці дані було розміщено у файлах такої структури: перший рядок – назва, другий рядок – автор, третій рядок – кількість сторінок, четвертий рядок – рік виходу, п’ятий та шостий рядок – оцінки, сьомий рядок – реальність світу, восьмий рядок – кількість оцінок, дев’ятий рядок – піджанр, інші рядки – опис сюжету. Приклад заповненого файлу представлено на рисунку 3.1.

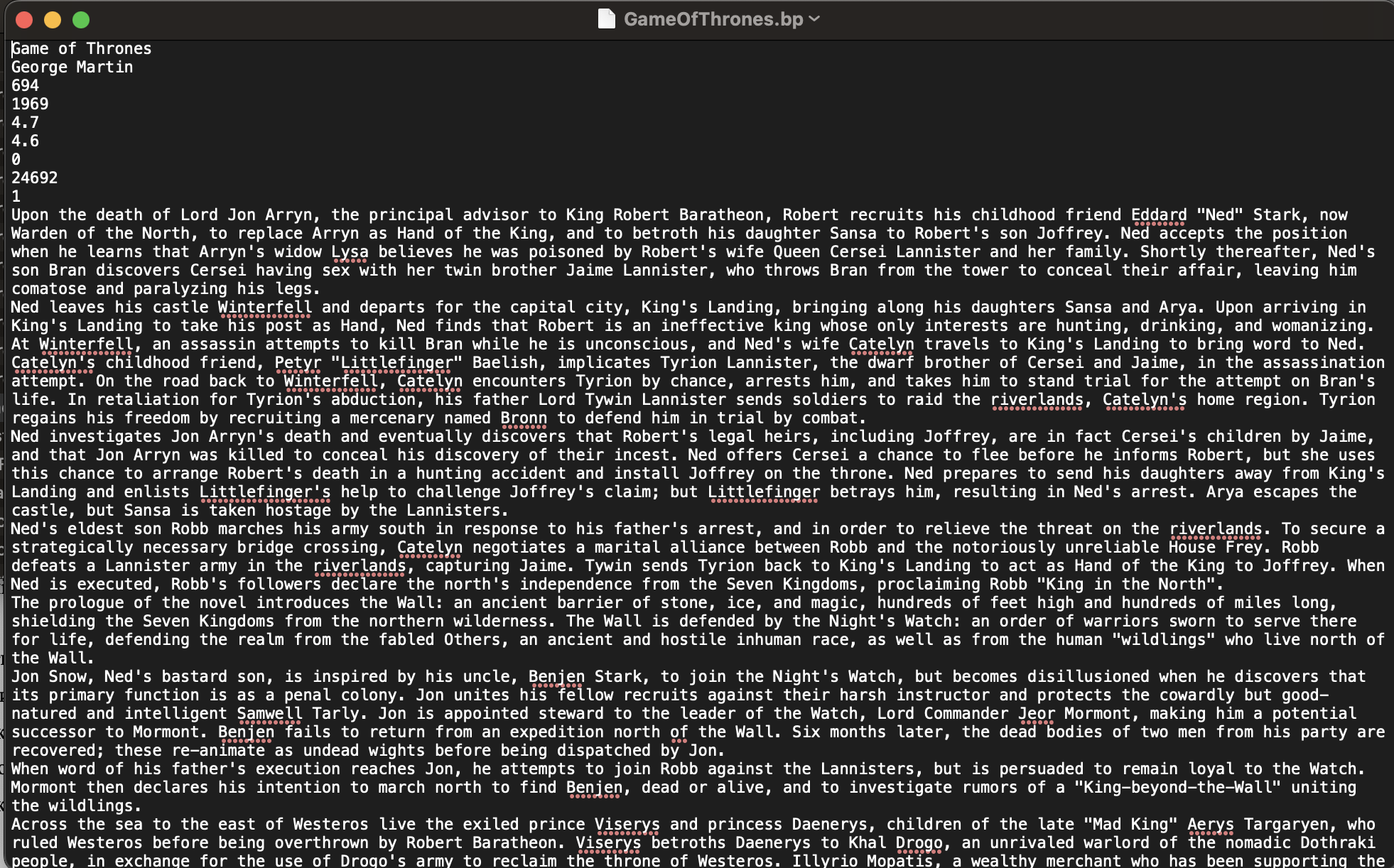


Рисунок 3.1 – Приклад заповненого файлу

Таким чином було створено 125 файлів. 20% даних було відведено для подальшого тестування. Їх назви записані у файл “test.bp”. Інші файли, призначенні для навчання, були записані до файлу “books.bp”.

## Аналіз опису книги

З короткого опису сюжету нами було вирішено виокремити такі характеристики:

* Кількість персонажів
* Відносна кількість чоловічих та жіночих персонажів
* Кількість сюжетних поворотів
* Кількість локацій
* Кількість рас
* Відносна кількість любовних, сімейних, дружніх та ворожих стосунків

Кількість персонажів рахується як кількість імен, що були у тексті. Список всіх імен було отримано з відкритого джерела даних: <https://parse-dashboard.back4app.com/apps/42b8ca5e-4c7a-4fae-bec7-b937694499d6/browser/NamesList>

При зустрічі чоловічого імені збільшувалася кількість чоловіків, при зустрічі жіночого – кількість жінок.

Для визначення інших параметрів було створено файли з переліком усіх ключових слів, що стосуються цих тем.

Для того, щоб усі книги були у рівних умовах усі абсолютні показники було поділено на кількість слів у описі. Таким чином довжина опису ніяк не буде впливати на параметри.

Код аналізу текстів представлено у додатку А, файли Book.py та TextWorker.py.

## Занесення даних до сховища

Код створення сховища та занесення даних представлено у додатку А, файл DBWorkre.py та детально описано у роботі Берлінського Я.В.

Результат заповнення сховища даних зображено на рисунку 3.2

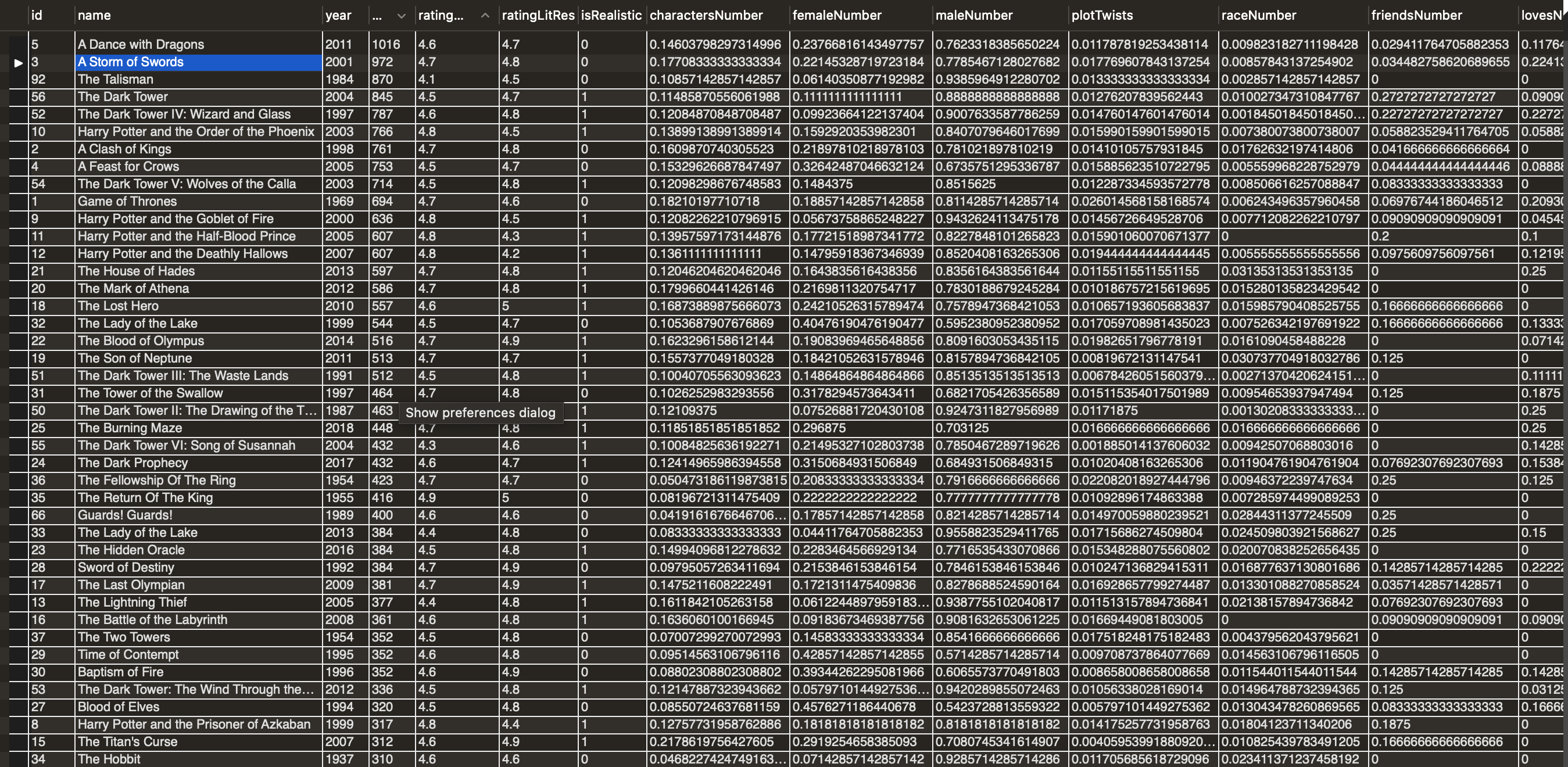


Рисунок 3.2-Скріншот заповненого сховища даних

## Отримання даних зі сховища

При отримані даних зі сховища було вирішено звести кожен параметр до проміжку [0;1]. Це потрібно для пришвидшення роботи алгоритмів навчання.

Для досягнення цієї мети була використана така формула (Формула 3.1):

* + - * 1. Формула для зведення параметрів у проміжок [0;1], де x – значення параметру, xmin – мінімальне значення параметру, xmax – максимальне значення параметру

Таким чином, мінімальне значення будь-якого параметру набуде 0, а максимальне – 1. Код зміни даних представлено у додатку А, файл DataWorker.py

# ІНТЕЛЕКтуальний аналіз даних

## Обґрунтування вибору методів інтелектуального аналізу даних

Для вирішення задачі прогнозування жанру книги (задача класифікації) потрібно використовувати інтелектуальний аналіз даних. Спираючись на дані про відомі книги, ми зможемо створити математичну модель, що допоможе прогнозувати належність книги до темного чи світлого фентезі. Для алгоритму інтелектуального аналізу важливі 2 характеристики: швидкість та якість. Тому було обрано 2 методи інтелектуального аналізу для порівняння:

* Логістична регресія з методом градієнтного спуску. (якість)
* Дерево прийняття рішень. (швидкість)

## Теоретичне обґрунтування логістичної регресії та методу градієнтного спуску

Логістична регресія використовується для того, щоб прогнозувати дані, що можуть набувати значень 0 чи 1. Для цього використовується сигмоїда. (Формула 4.1)

* + - * 1. Формула сигмоїди

Ця функція не може набувати значень більших за 1 та менших за 0. Можемо побачити це на груфіку. Рисунок 4.1

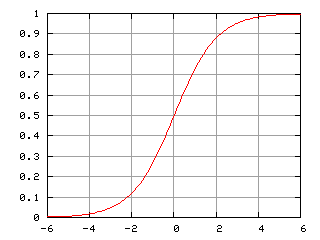


Рисунок 4.1 – Графік сигмоїди

Ця функція залежить від z. Оскільки ми маємо багато параметрів, то виразимо z певною лінійною функцією. (Формула 4.2)

* + - * 1. Лінійна функція z, де xi – jй параметр, – jй коефіцієнт, n – кількість параметрів

Нам потрібно знайти найбільш підходящі . Таким чином, нам потрібно мінімізувати різницю між y та f(z). В результаті математичних перетворень отримаємо, що нам потрібно мінімізувати функцію у формулі 4.3.

* + - * 1. Формула cost-функції для логістичної регресії

Для того, щоб підібрати параметри для F при яких F набуває найменшого значення потрібно часткову похідну від кожної змінної прирівняти до 0 та розв’язати систему рівнянь. Оскільки рівняння не є лінійними, то зробити це програмно важко. Тому використовують метод градієнтного спуску, що дозволяє приблизитись до оптимальних значень параметрів. Метод градієнтного спуску полягає у тому, що параметри функції змінюються короткими кроками, в результаті чого значення функції поступово зміщується до мінімуму. Для градієнтного спуску у логістичній регресії використовується формула 4.4

* + - * 1. Формула для градієнтного спуску

## Теоретичне обґрунтування дерева прийняття рішень

Метод дерева прийняття рішень полягає у тому, що на основі відомих даних формується бінарне дерево. Кожен вузол дерева оцінює певний параметр та в результаті оцінки відправляє його до нащадка. Результатом такого проходу буде потрапляння до листа дерева, що містить значення 1 або 0 – результат прогнозу.

## Підготовка даних до інтелектуального аналізу

Розглянемо кожний з зібраних факторів для того, щоб визначити їх вплив на результат. Знайдемо середнє значення кожного із параметрів при світлому фентезі, темному фентезі та взагалі. Якщо різниця між середнім та середнім при умові буде більшою за 0.05, то фактор можна вважати значимим. Графіки залежностей представлено на рисунках 4.2-4.16. Фактори розглянуті у таблиці 4.1.

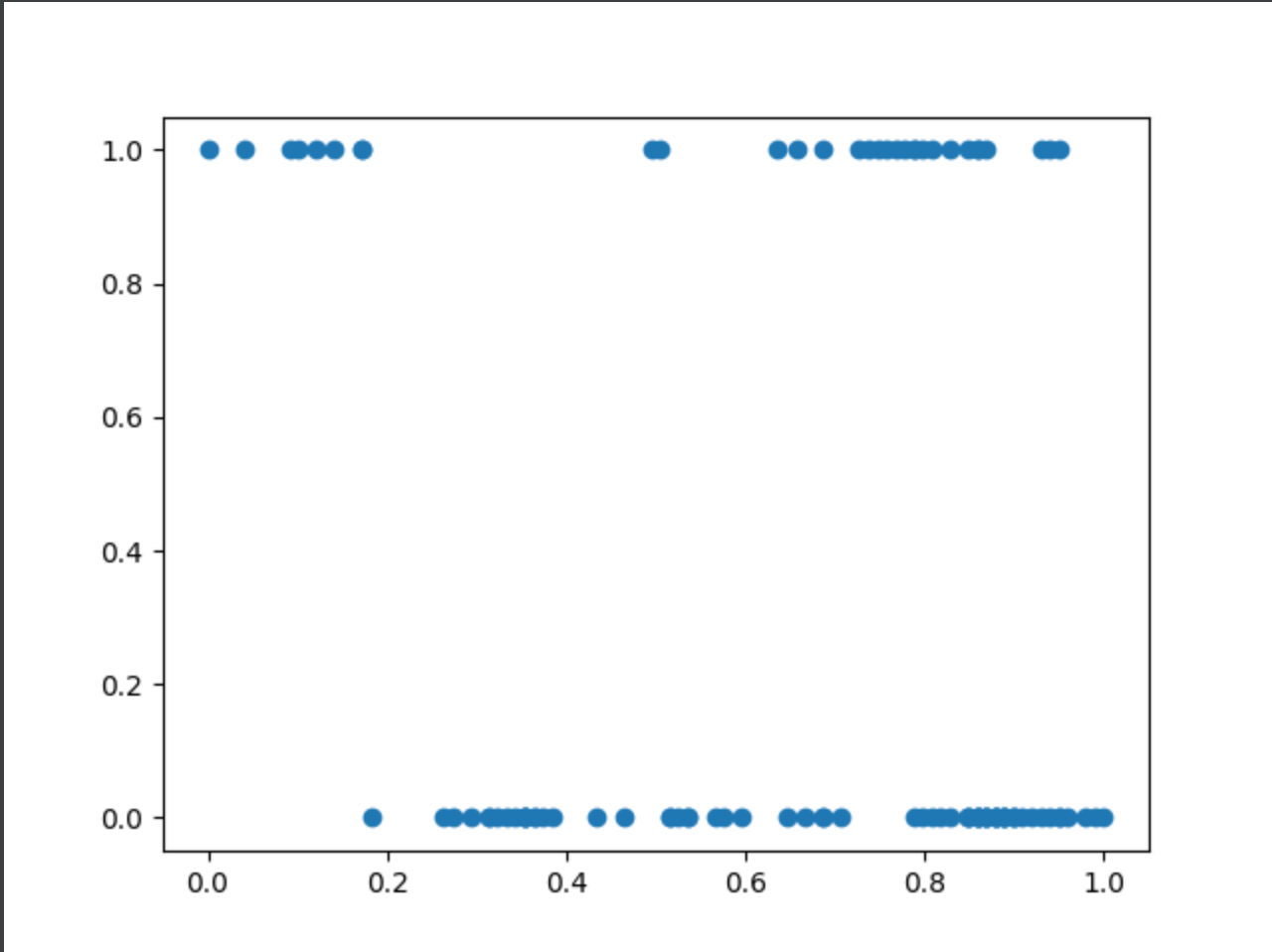


Рисунок 4.2 - Графік залежності жанру фентезі від року виходу

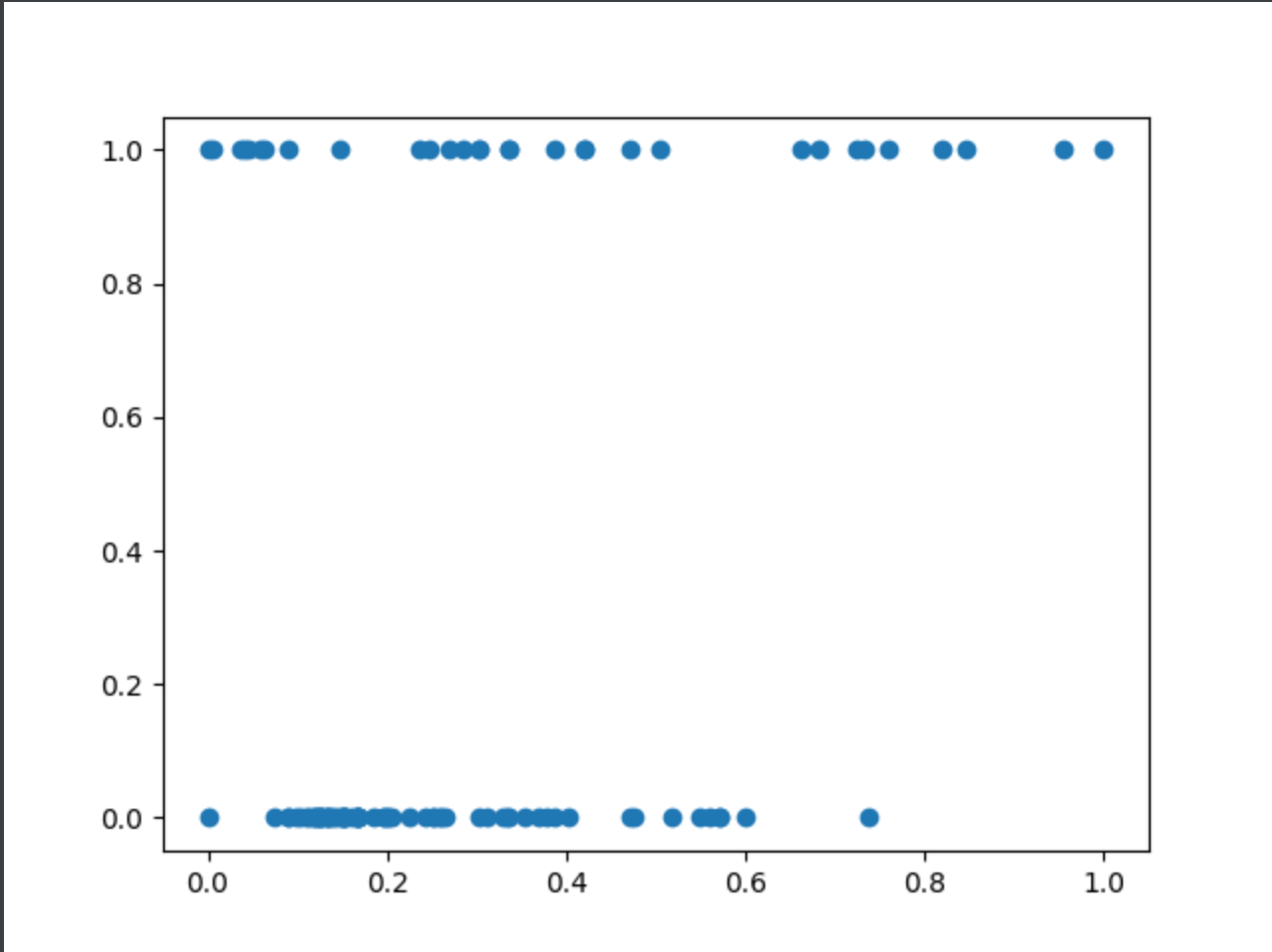


Рисунок 4.3 - Графік залежності жанру фентезі від кількості сторінок

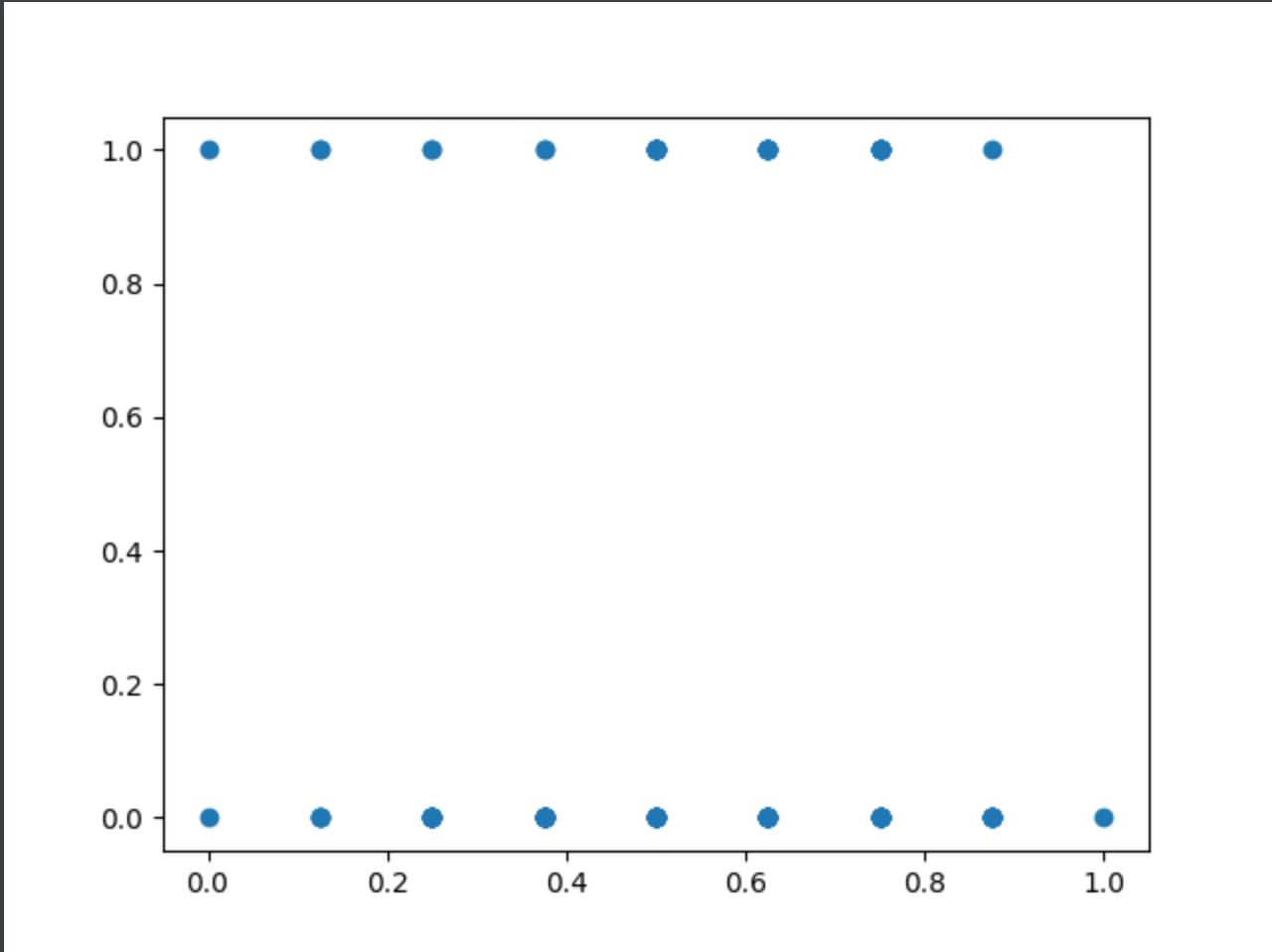


Рисунок 4.4 - Графік залежності жанру фентезі від рейтингу на LiveLib

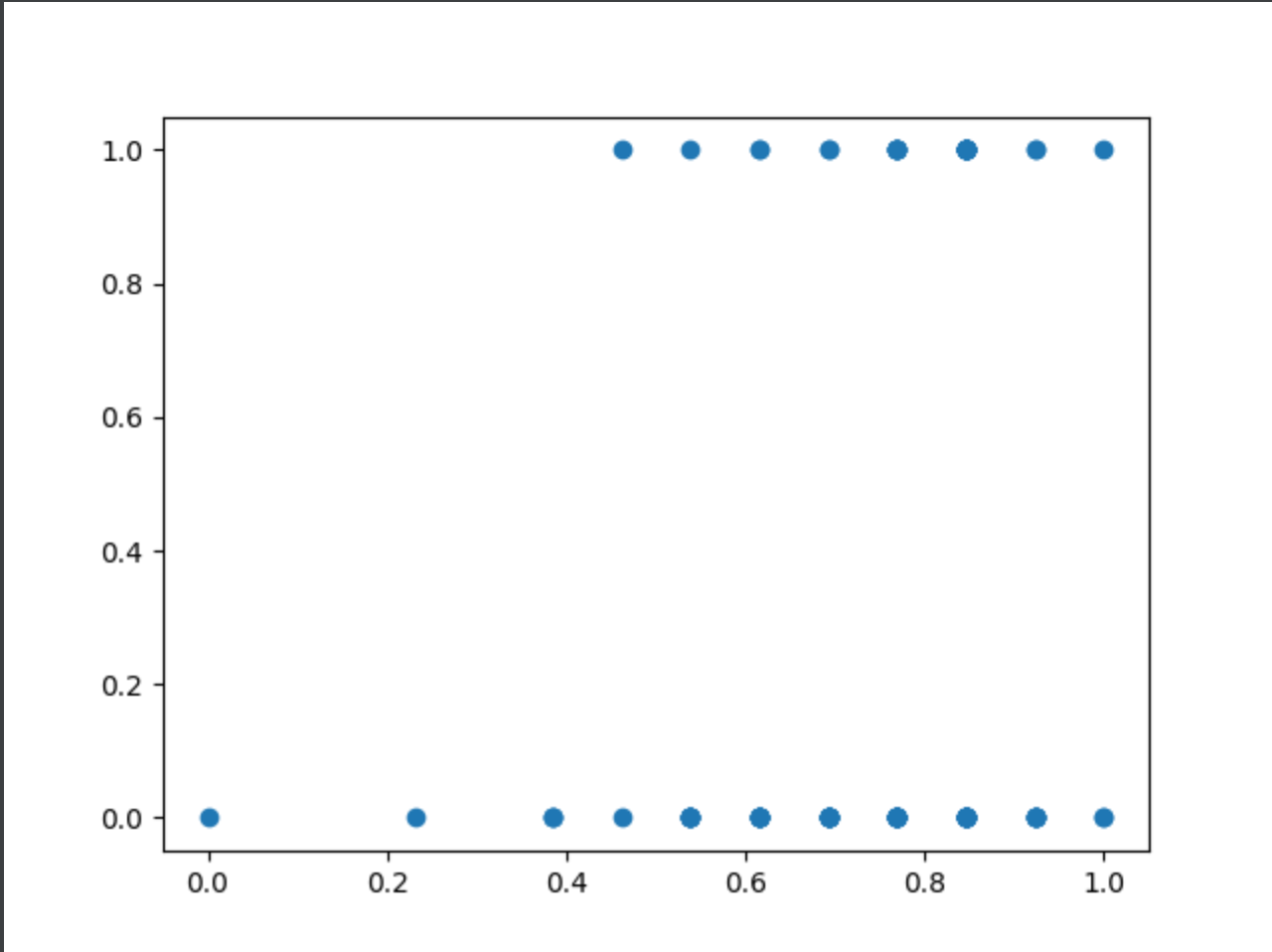


Рисунок 4.5 - Графік залежності жанру фентезі від рейтингу на LitRes

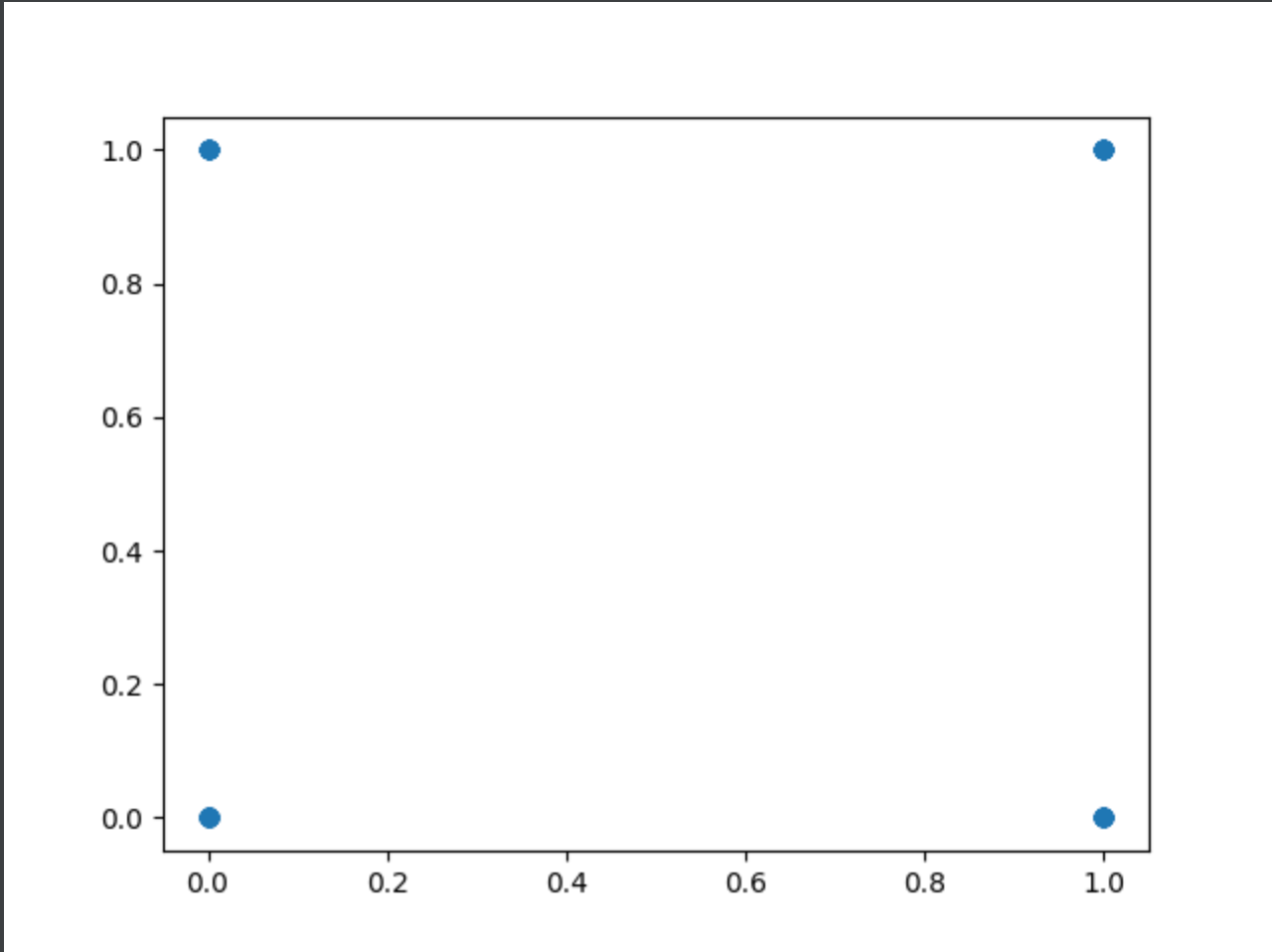


Рисунок 4.6 – Графік залежності жанру фентезі від реалістичності світу

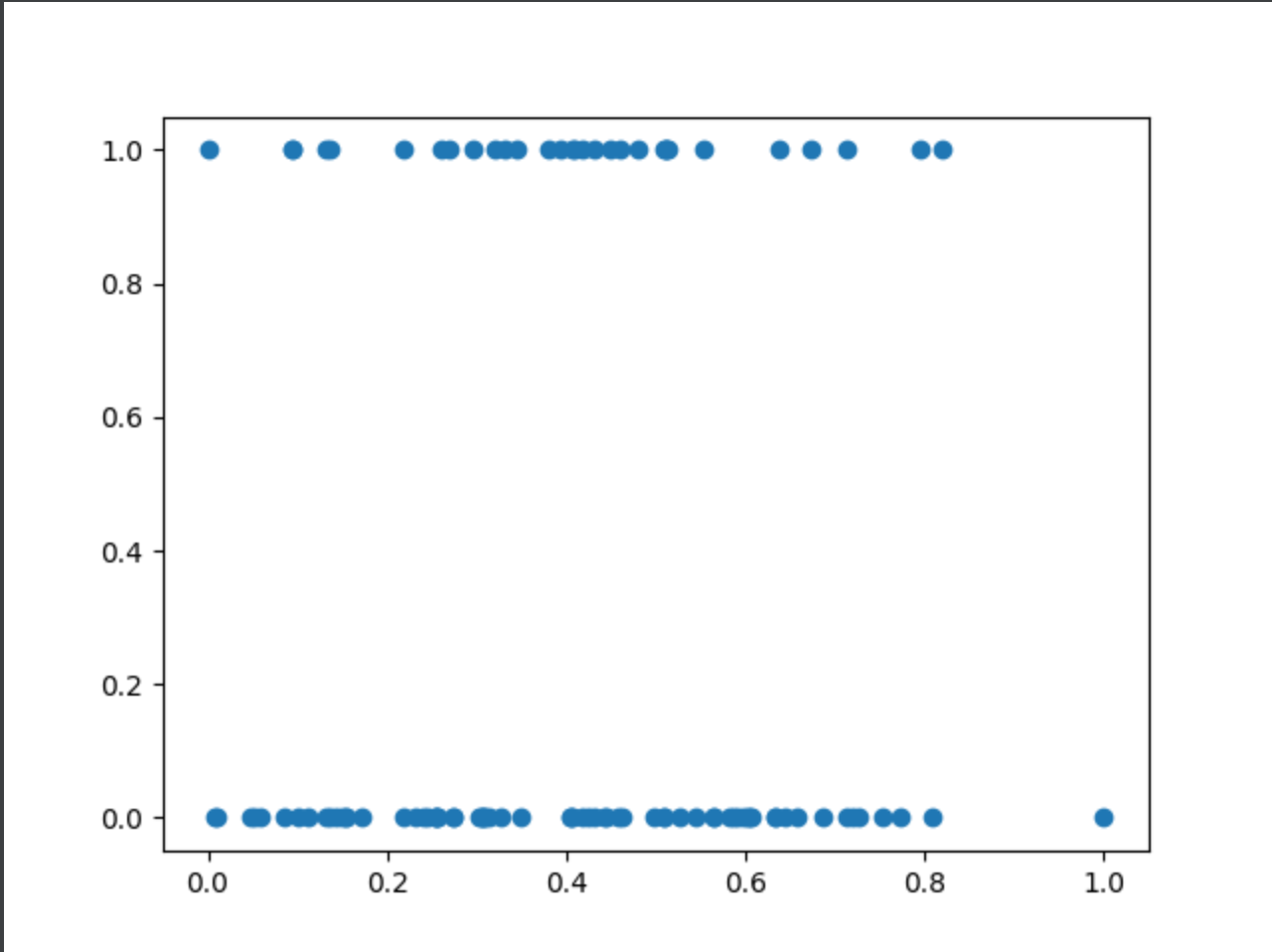


Рисунок 4.7 – Графік залежності жанру фентезі від кількості персонажів

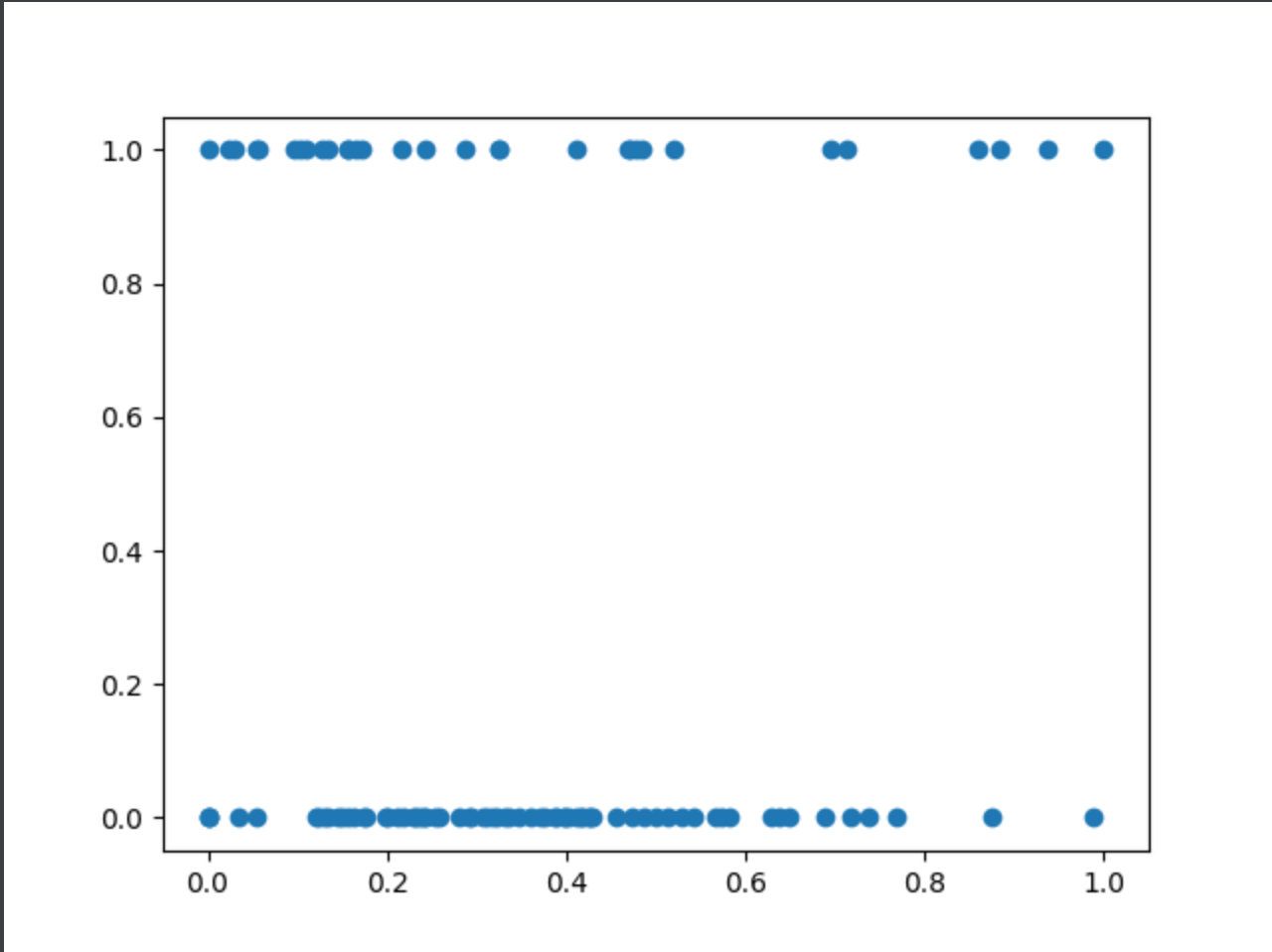


Рисунок 4.8 – Графік залежності жанру фентезі від відсотку жінок

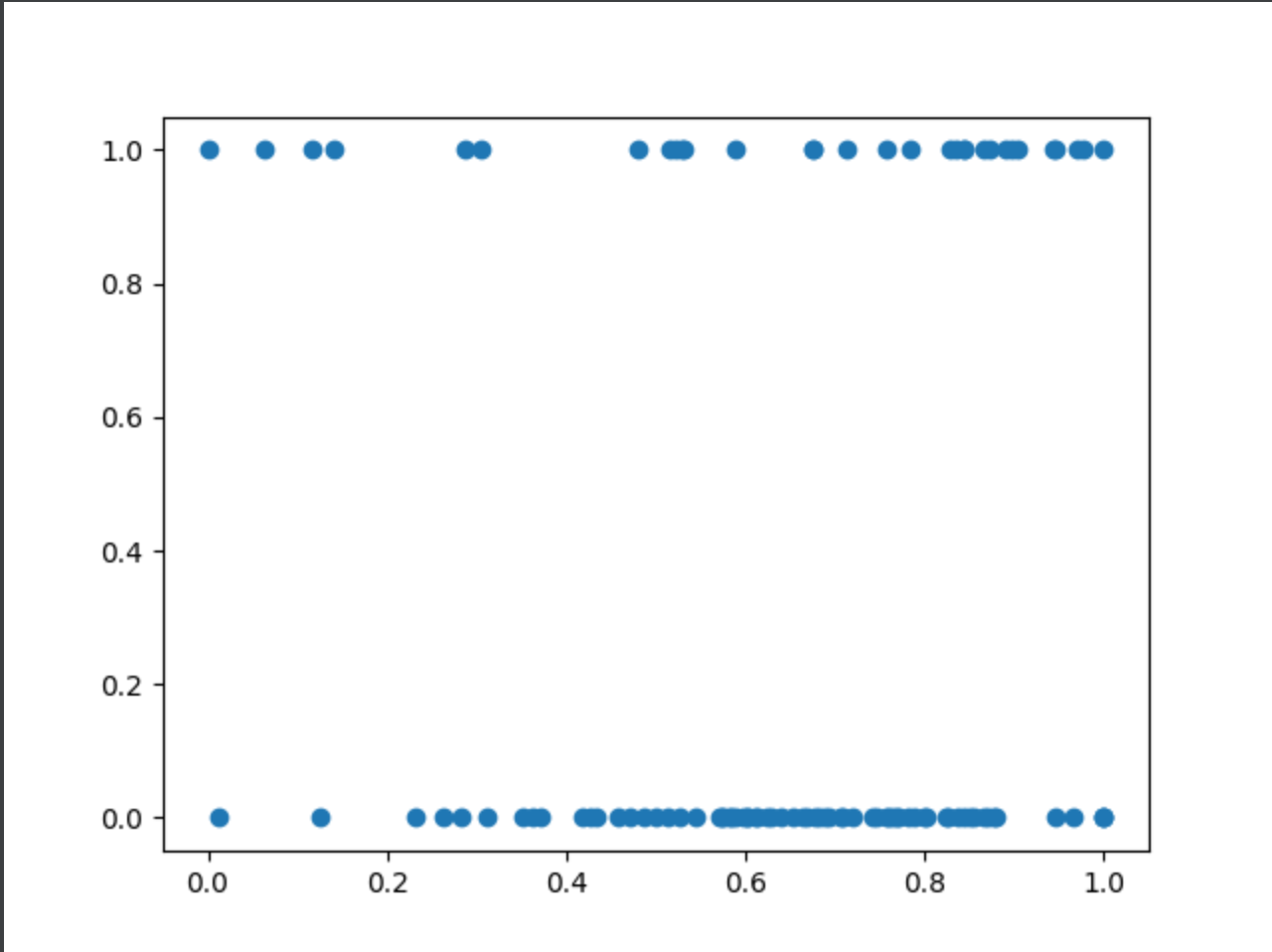


Рисунок 4.9 – Графік залежності жанру фентезі від відсотку чоловіків

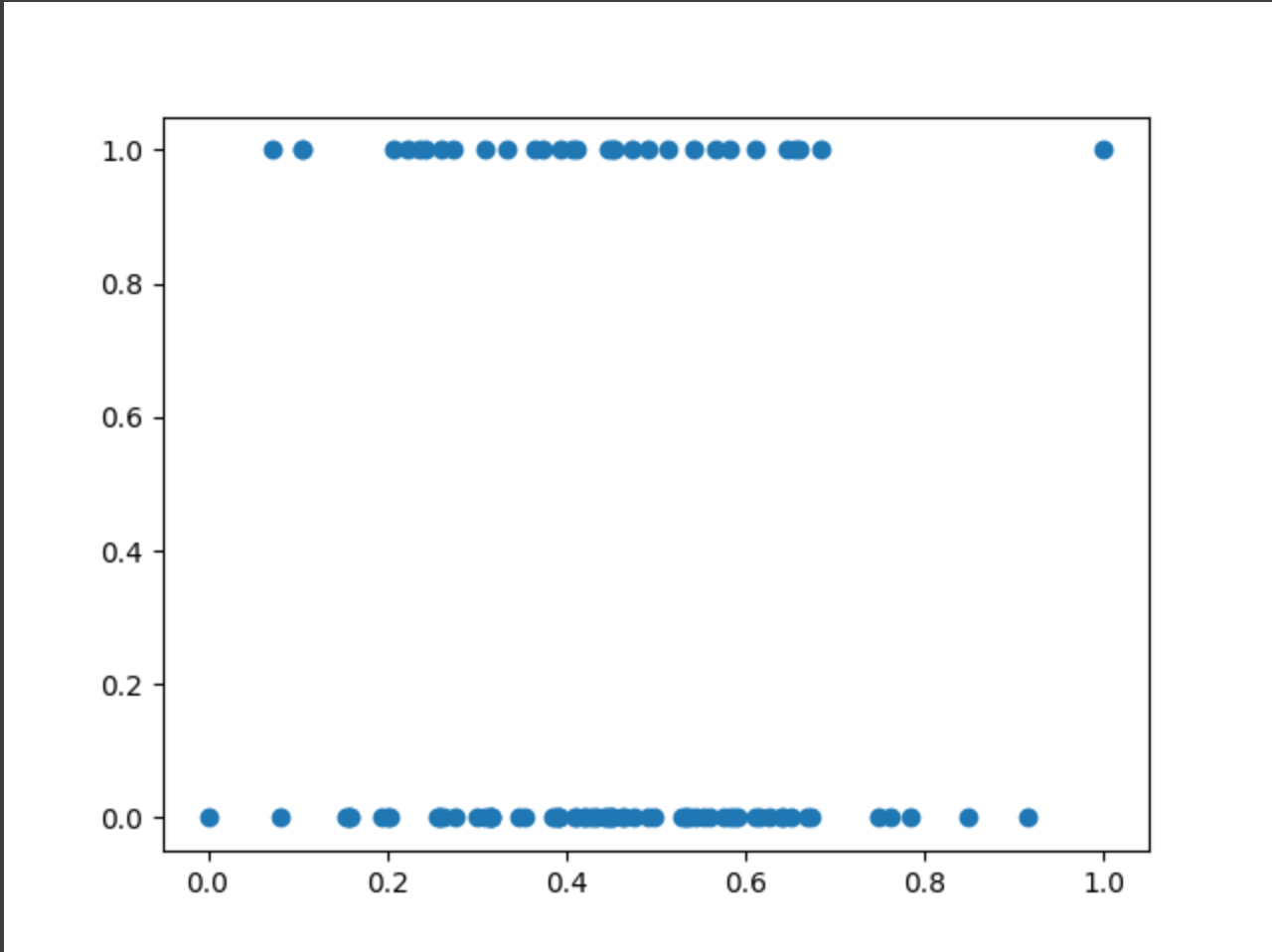


Рисунок 4.10 – Графік залежності жанру фентезі від кількості сюжетних поворотів

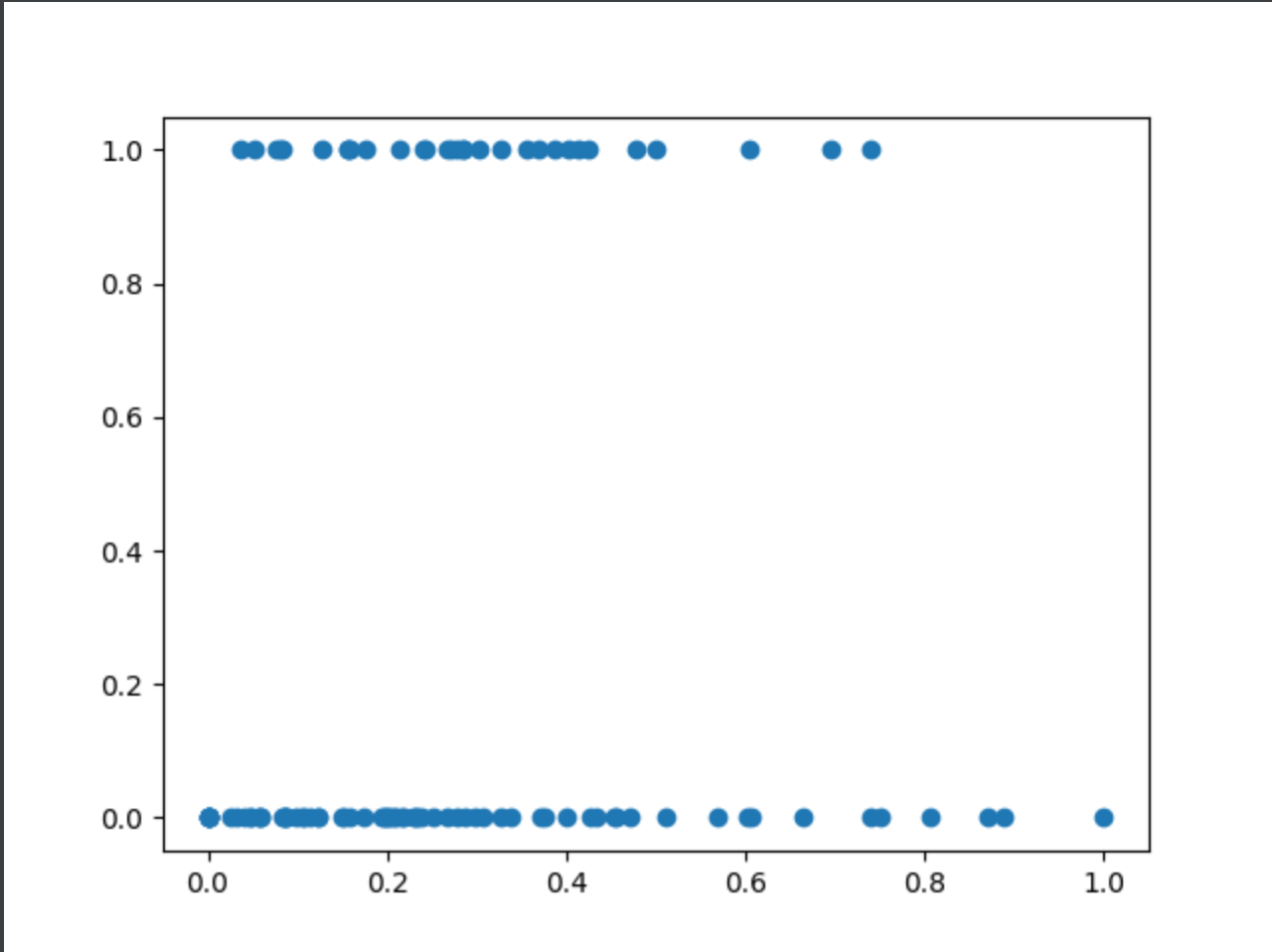


Рисунок 4.11 – Графік залежності жанру фентезі від кількості рас

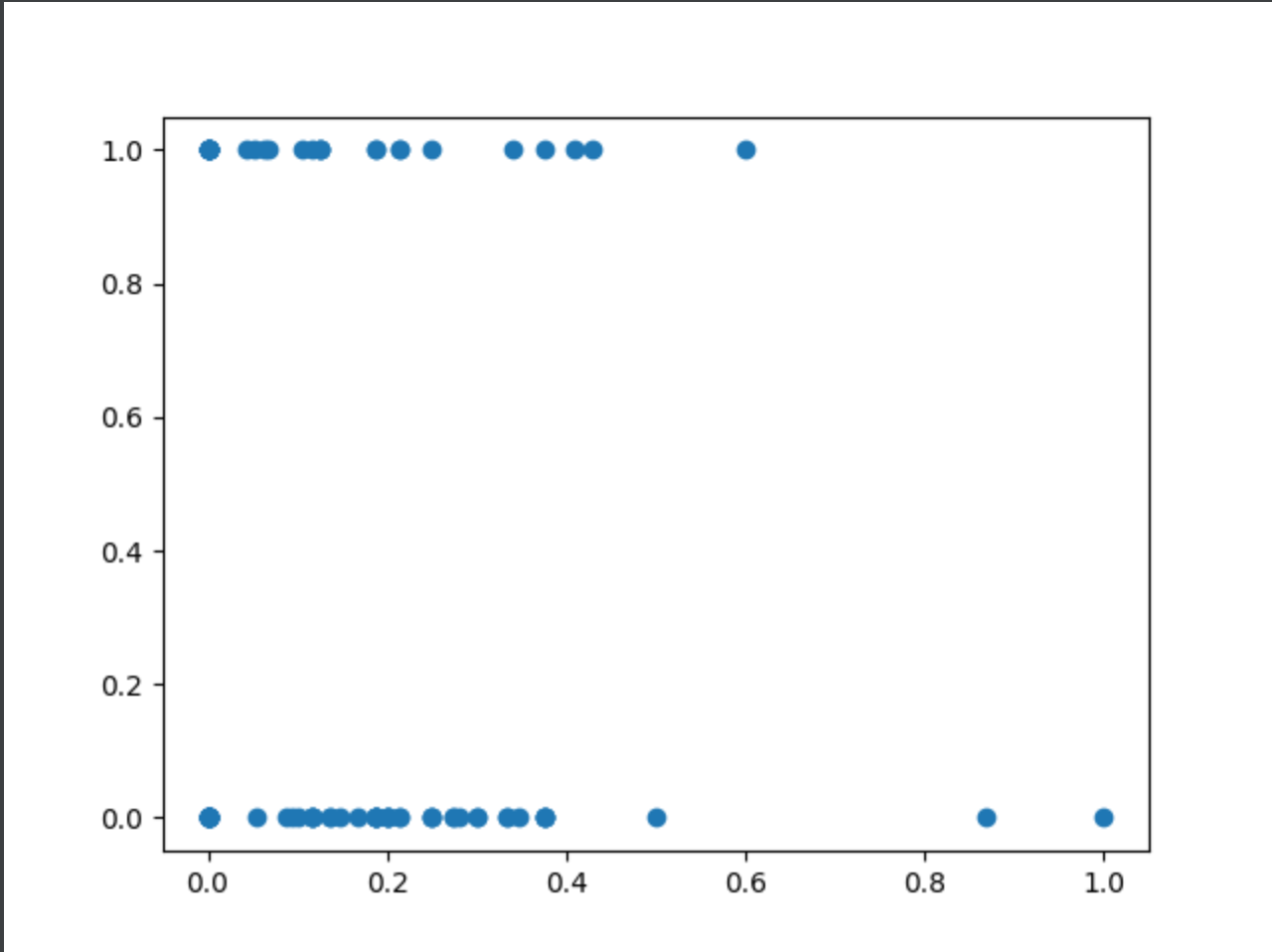


Рисунок 4.12 – Графік залежності жанру фентезі від проценту дружніх стосунків

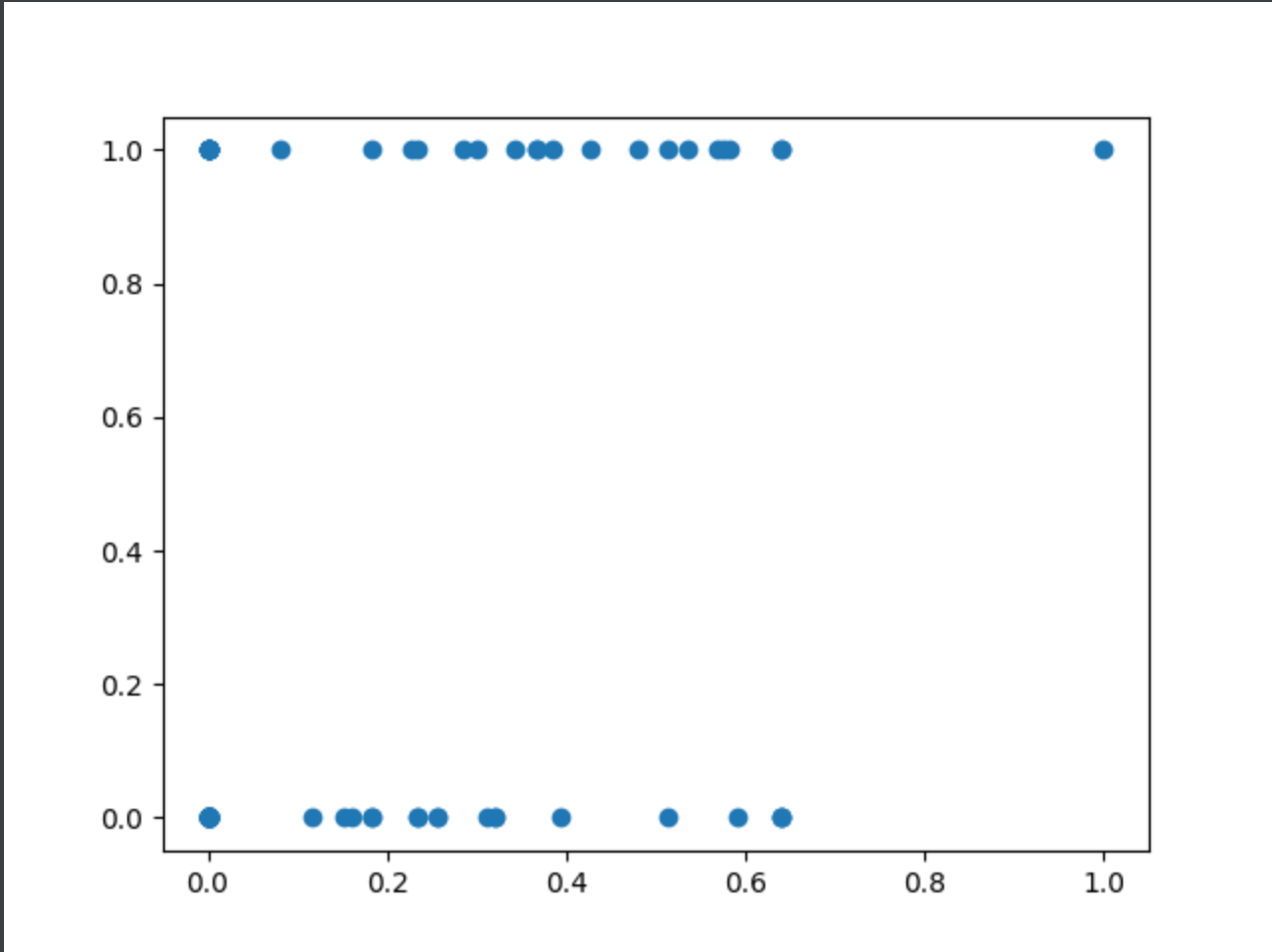


Рисунок 4.13 – Графік залежності жанру фентезі від проценту любовних стосунків

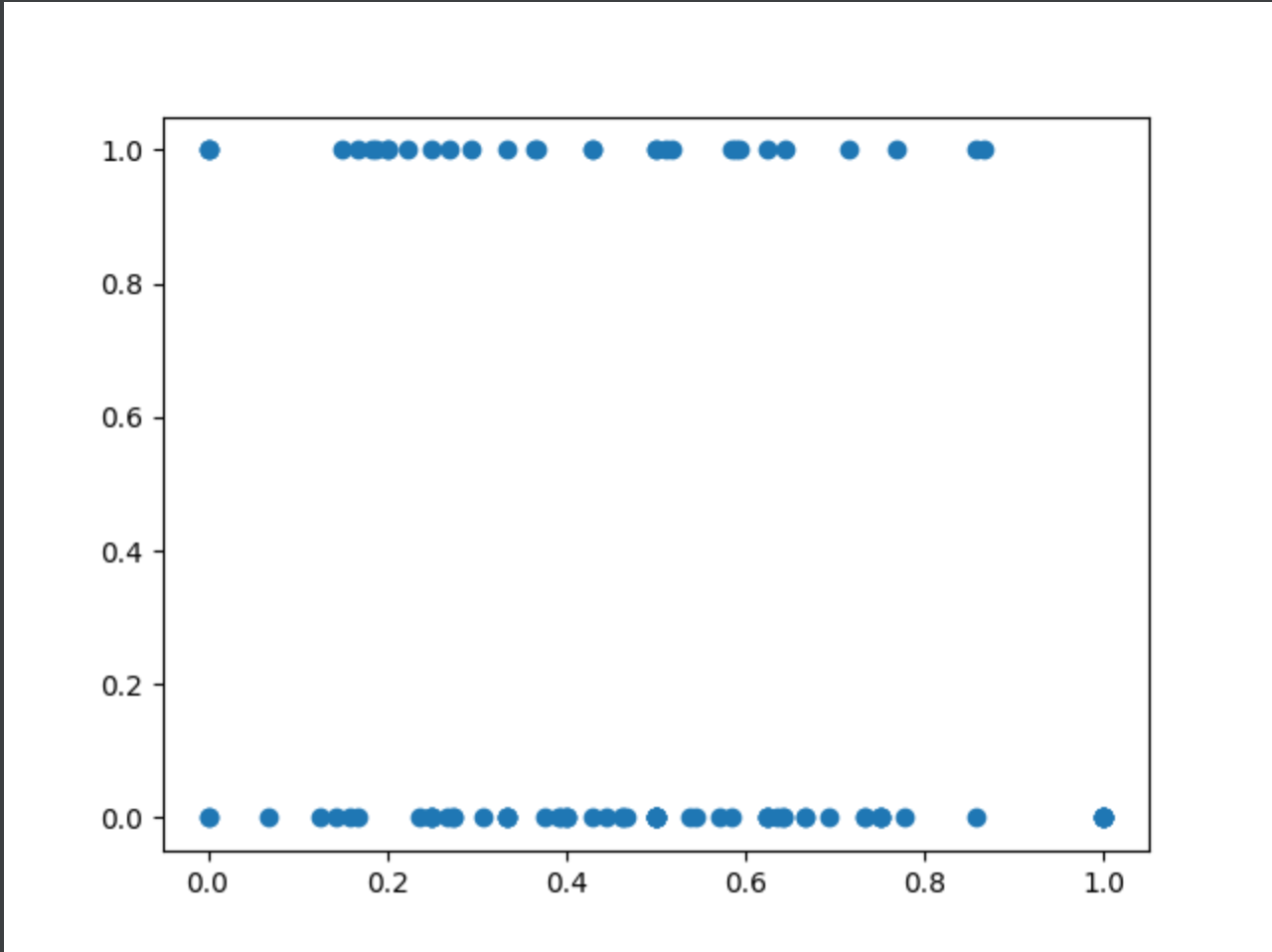


Рисунок 4.14 – Графік залежності жанру фентезі від проценту родинних стосунків

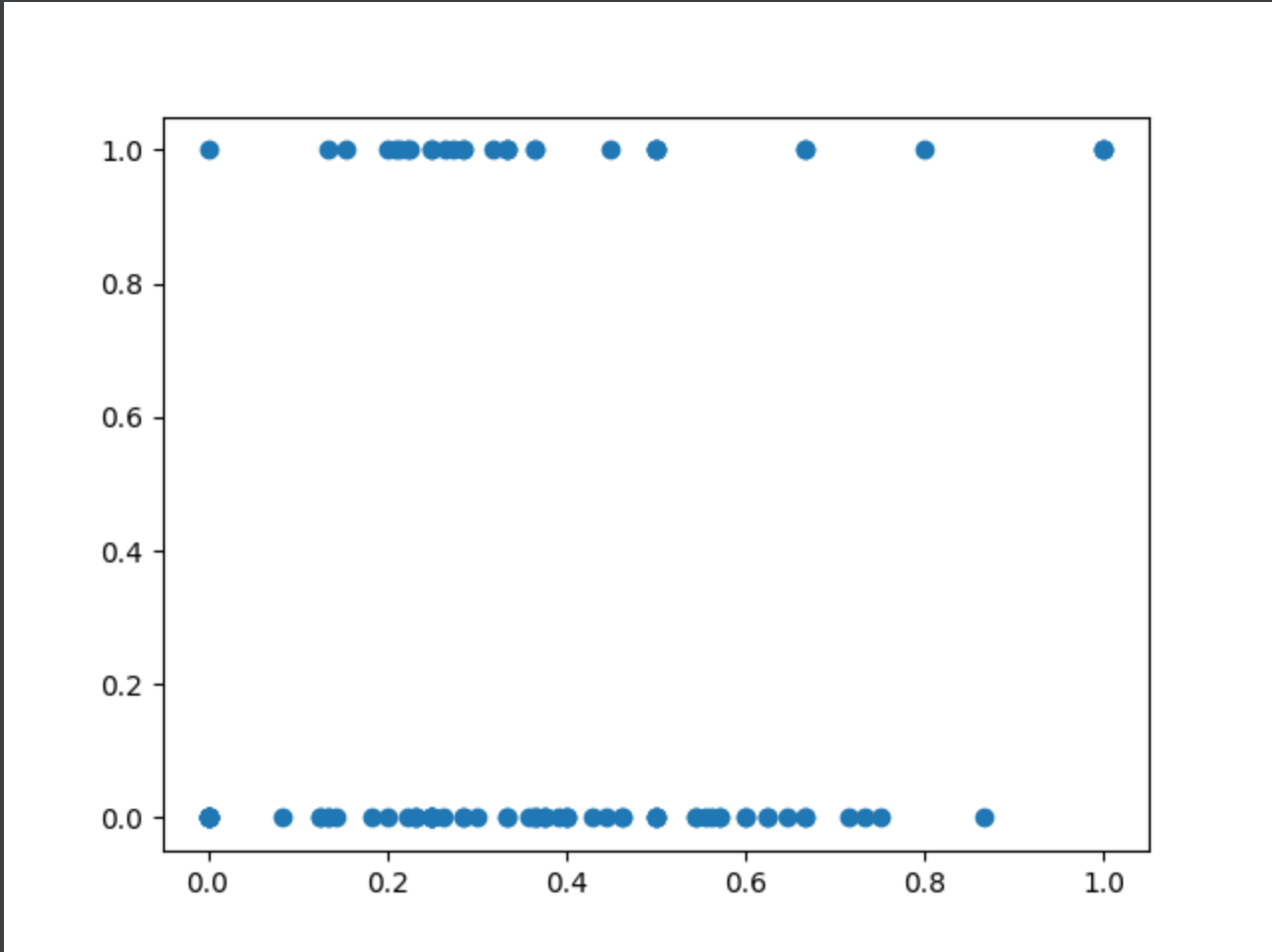


Рисунок 4.15 – Графік залежності жанру фентезі від проценту ворожих стосунків

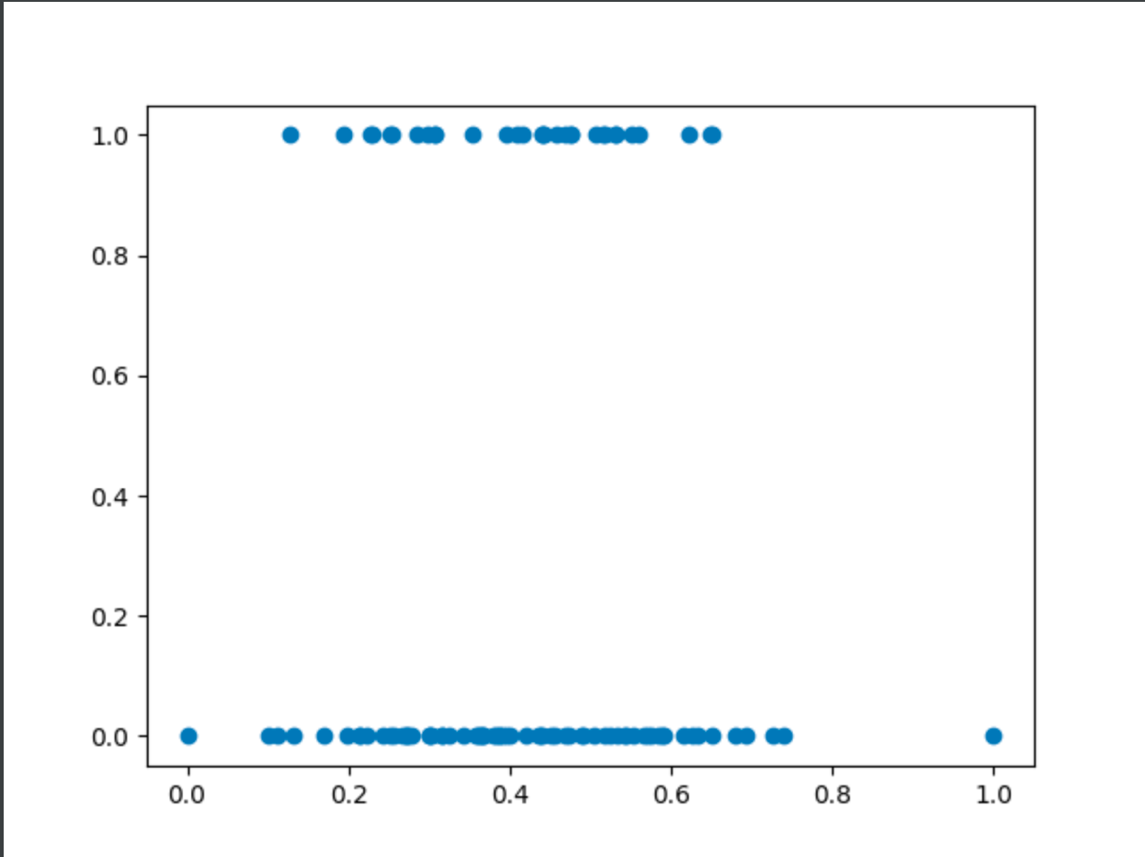


Рисунок 4.16 – Графік залежності жанру фентезі від кількості локацій

Таблиця аналізу факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Середнє | Сер. при св. | Сер. при тем. | Чи значущий |
| Рік | 0.65 | 0.68 | 0.59 | 1 |
| К-сть сторінок | 0.28 | 0.23 | 0.39 | 1 |
| LiveLib | 0.51 | 0.5 | 0.53 | 0 |
| LitRes | 0.74 | 0.71 | 0.79 | 1 |
| Реалістичність | 0.43 | 0.39 | 0.51 | 1 |
| К-сть персонажів | 0.4 | 0.39 | 0.4 | 0 |
| Жінок | 0.35 | 0.35 | 0.34 | 0 |
| Чоловіків | 0.64 | 0.64 | 0.65 | 0 |
| Поворотів | 0.43 | 0.44 | 0.42 | 0 |
| Рас | 0.27 | 0.26 | 0.29 | 0 |
| Дружніх зв. | 0.14 | 0.15 | 0.12 | 0 |
| Любовних зв. | 0.14 | 0.08 | 0.28 | 1 |
| Сімейних зв. | 0.47 | 0.51 | 0.39 | 1 |
| Ворожих зв. | 0.36 | 0.34 | 0.41 | 1 |
| Локацій | 0.41 | 0.41 | 0.41 | 0 |

Таким чином для визначення жанру фентезі нам потрібно оперувати лише такими параметрами: рік, кількість сторінок, рейтинг LitRes, реалістичність, кількість любовних зв’язків, кількість сімейних зв’язків та кількість ворожих зв’язків.

Далі дослідимо вибірку на мультиколінеарні фактори. Розрахуємо таблицю кореляцій за формулою Пірсона. (Формула 4.5)

* + - * 1. Формула Пірсона

Розраховані значення наявні в таблиці 4.2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | рік | сторінки | рейтинг | реаліст. | любовних | сімейних | ворожих |
| рік | 1 | 0.5 | -0.09 | -0.05 | 0.08 | -0.05 | 0.003 |
| сторінки | 0.5 | 1 | 0.1 | 0.12 | 0.3 | -0.07 | 0.006 |
| рейтинг | -0.09 | 0.1 | 1 | 0.0008 | 0.18 | 0.02 | -0.02 |
| реаліст. | -0.05 | 0.12 | 0.0008 | 1 | 0.09 | -0.09 | 0.11 |
| любовних | 0.08 | 0.3 | 0.18 | 0.09 | 1 | -0.33 | -0.02 |
| сімейних | -0.05 | -0.07 | 0.02 | -0.09 | -0.33 | 1 | -0.81 |
| ворожих | 0.003 | 0.006 | -0.02 | 0.11 | -0.02 | -0.81 | 1 |

Таким чином легко побачити, що кількість сімейних та кількість ворожих зв’язків є мультиколінеарними, а отже не будемо враховувати один з них – кількість ворожих.

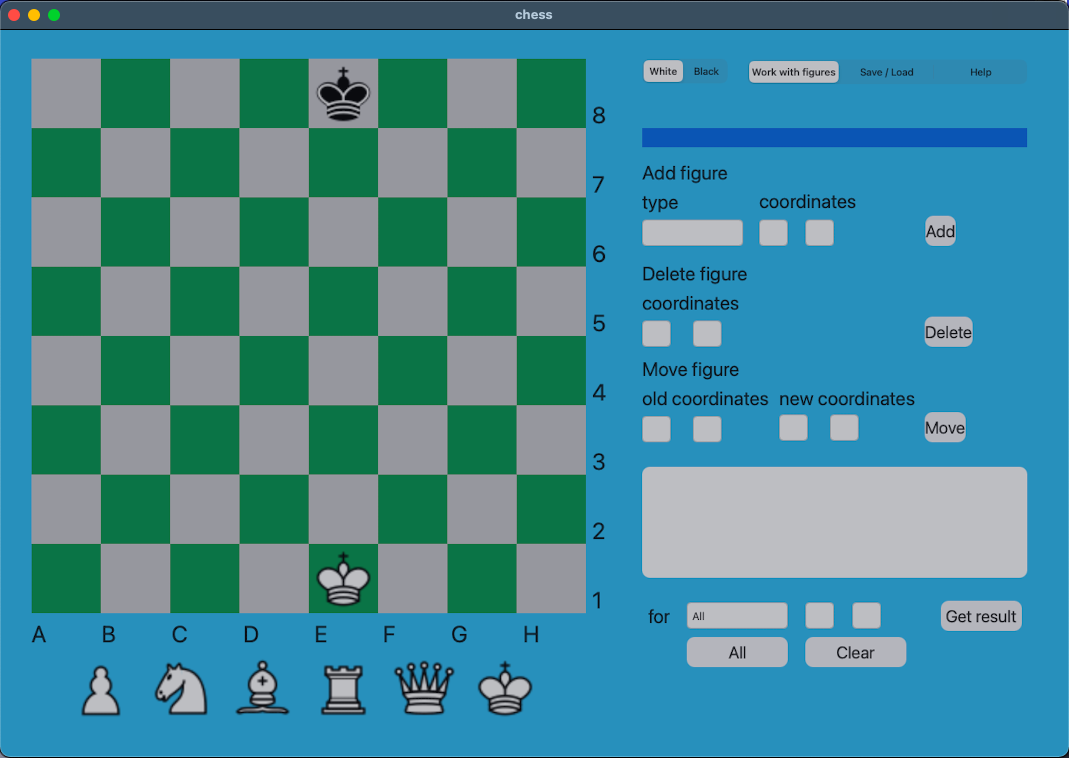
Оскільки кількість світлих і темних фентезі у вибірці нерівна, то нам потрібно її доповнити. Скористаємося алгоритмом SMOTE для доповнення вибірки. Цей алгоритм на основі вже відомих значень, з урахуванням випадкового чинника генерує нові значення для вибірки. В результаті отримаємо 138 значень, 69 з яких – світле фентезі та 69 – темне фентезі.

Код аналізу даних представлено у додатку А, файл LogisticRegresion.py.

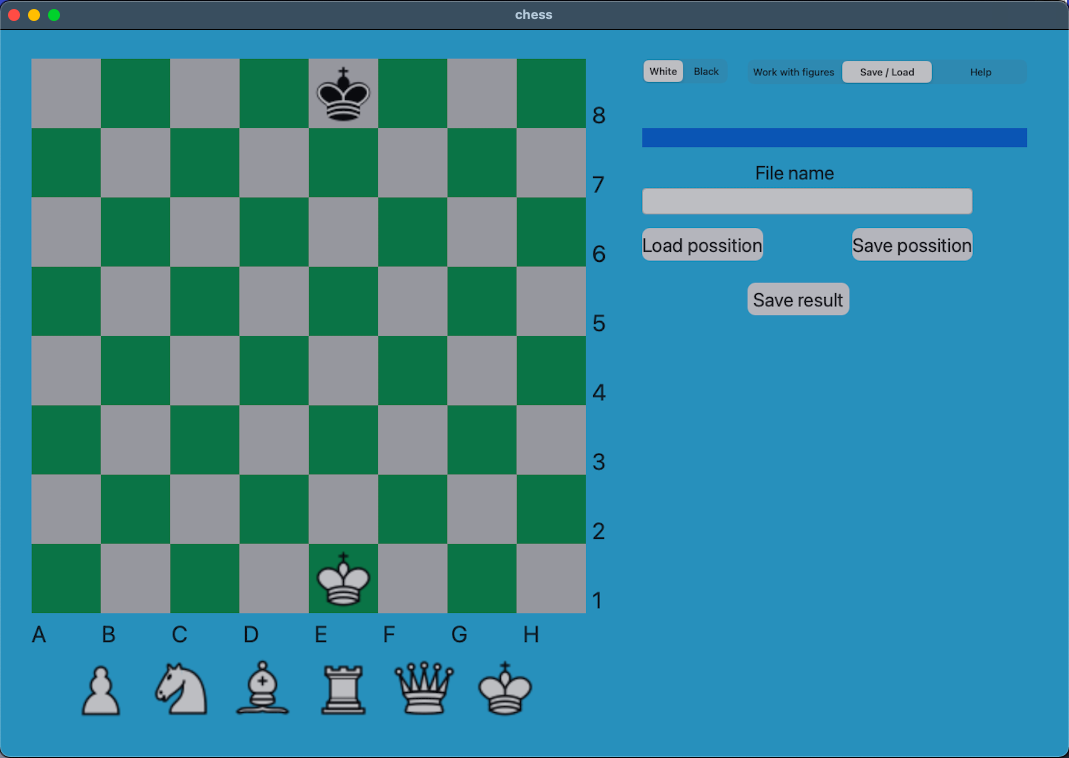
# Інструкція користувача

## Робота з програмою

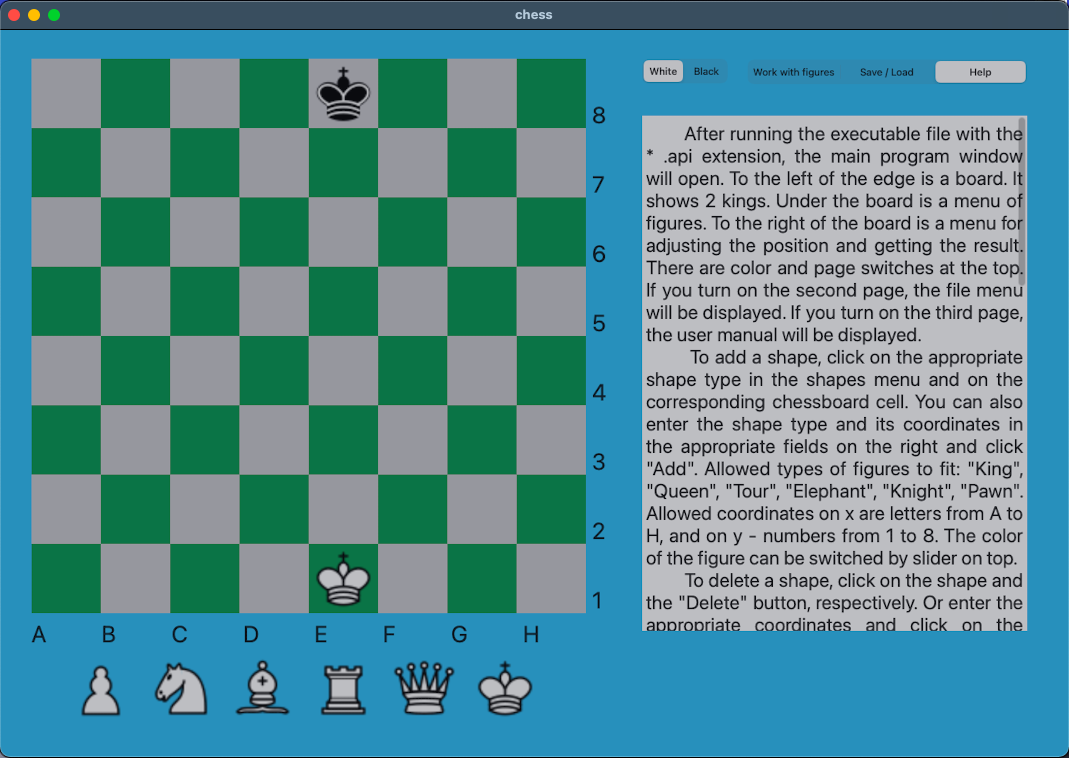
Після запуску виконавчого файлу за розширенням \*.api, відкриється головне вікно програми (рис. 6.1). Лівіше до краю розташована дошка. На ній відображено 2 королі. Під дошкою розташоване меню фігур. Справа від дошки розташоване меню для налаштування позиції і отримання результату. Зверху містяться перемикачі кольору та сторінок. Якщо увімкнути другу сторінку, то буде відображено меню роботи з файлами(рис. 6.2). Якщо увімкнути третю сторінку, то буде відображено інструкцію користувача.(рис. 6.3)



* + - * 1. Початкове вікно програми



* + - * 1. Сторінка роботи з файлами



* + - * 1. Інструкція користувача

Для того щоб додати фігуру потрібно натиснути на відповідний тип фігури у меню фігур та на відповідну клітинку шахівниці. Також можна вписати тип фігури та її координати у відповідні поля справа та натиснути "Add". Дозволені типи фігур для вписування: "King", "Queen", "Tour", "Elephant", "Knight", "Pawn". Дозволені координати по х це літери від A до H, а по у - цифри від 1 до 8. Колір фігури можна переключити слайдером зверху.

Для видалення фігури потрібно натиснути на відповідно фігуру та на кнопку "Delete". Або ввести відповідні координати та натиснути на кнопку "Delete". Для переміщення фігури потрібно натиснути на фігуру та на її нове місце. Або ж ввести відповідні координати та натиснути "Move".

Кількість фігур на шахівниці має відповідати правилам гри у шахи. для кожного кольору: 1 король, 1 ферзь, 2 тури, 2 коня, 2 слони та 8 пішаків. також, програма враховує, що 1 пішак може буди перетвореним на "зайву" фігуру, окрім короля.

Перед тим як отримати результат потрібно впевнитися, що король суперника не знаходиться під шагом, бо в такому випадку хід неможливий.

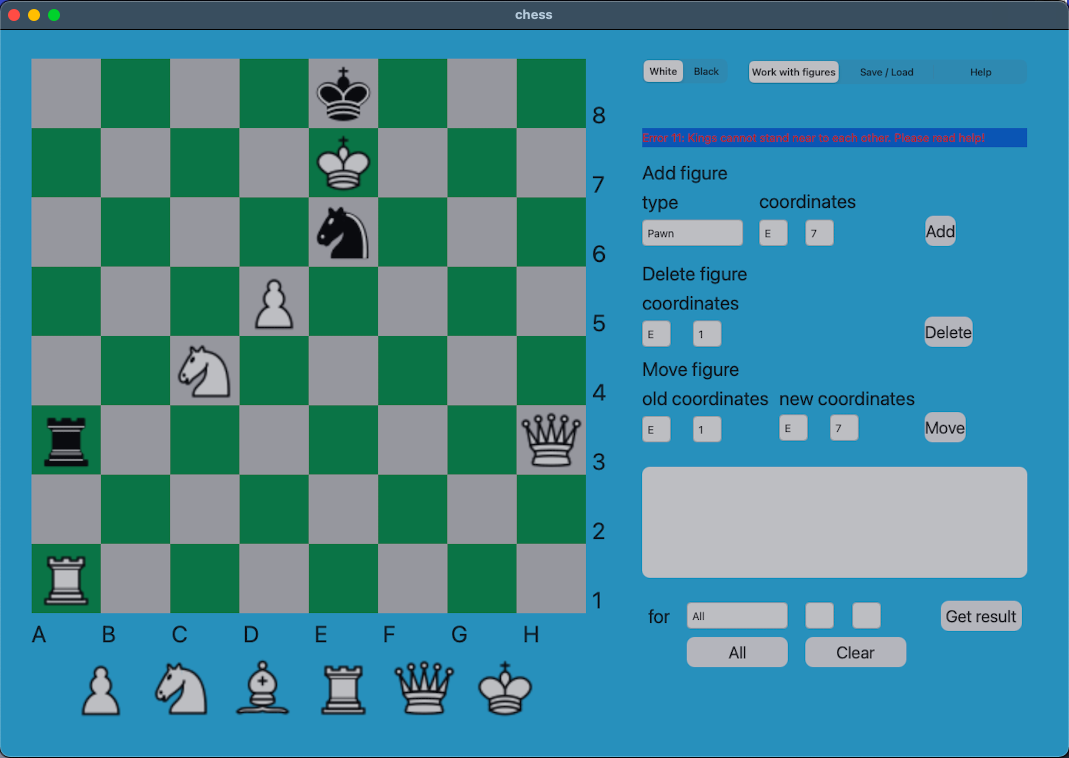
Результат буде відображено у відповідному текстовому полі та кольором на шахівниці. Жовта клітинка - фігура яка б'є, червона клітинка - фігура, яку поб'ють, червоний кружечок - шлях (рис. 6.4). Для отримання результату для конкретної фігури достатньо просто на неї натиснути після отримання результату або вписати її тип та координати до відповідних полів до отримання результату. Для того щоб знову отримати результат для всіх фігур потрібно натиснути на кнопку “All”, щоб очистити дошку від кольорів достатньо натиснути на кнопку “Clear”. Для того, щоб отримати результат для іншого кольору потрібно переключити слайдер угорі.



* + - * 1. Відображення результату виконання

Для збереження результату в файл потрібно обов'язково його отримати. Потім перейти на сторінку роботи з файлами, ввести адресу файлу та натиснути "Save result". Для збереження позиції потрібно зайти на сторінку роботи з файлами, ввести адресу файлу та натиснути "Save possition". Для завантаження позиції потрібно перейти на сторінку роботи з файлами, ввести адресу існуючого файлу та натиснути "Load result".

При виконанні певних дій неправильно будуть виводитися помилки(рис. 6.5).



* + - * 1. Відображення помилки 11

Типи помилок:

Помилка 0: Неправильно введено ти фігури, дозволені типи: King, Queen, Tour, Elephant, Knight, Pawn.

Помилка 1: Неправильно введена координата по х, можна вводити лише літери від A до H

Помилка 2: Неправильно введена координата по у, можна вводити лише цифри від 1 до 8

Помилка 3: Король суперника під шагом, такий хід неможливий. Перевірте правильність позиції, або змініть колір ходу.

Помилка 4: Невірна адреса файлу

Помилка 5: На полі не може бути 2 королі. Можливо вам потрібно перемістити короля

Помилка 6: На полі має бути хоча б 1 король. Можливо вам потрібно перемістити короля

Помилка 7: Неможлива кількість фігур, перевірте позицію ще раз

Помилка 8: Введена позиція неіснуючої фігури. Таку фігуру неможливо перемістити/видалити. Можете скористатись візуальним способом переміщення/видалення

Помилка 9: Неможливо додати фігуру у цю позицію, так як на цій позиції є інша фігура

Помилка 10: Пішак не може мати координату 1 чи 8 по у.

Помилка 11: Королі не можуть розташовуватися один біля одного, змініть позицію.

## Формат вхідних даних

Вхідними даними програми є розташування фігур. Розташування фігур може бути заданим користувачем під час роботи програми або ж завантажено з файлу, створеного в програмі до цього.

Результатом виконання програми є набір рядків, в яких відображається яка фігура яку б’є. Наприклад: “Queen E3-E4”. Також результат графічно відображено на шахівниці. Жовтою позначена фігура, що б’є, червоною – яку б’ють, а червоними кружечками позначено шлях бою.

## Системні вимоги

Системні вимоги до програмного забезпечення наведені в табл. 6.1

Системні вимоги до програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | **Мінімальні** | **Рекомендовані** |
| Операційна система | macOS Sierra, macOS High Sierra, macOS Mojave, macOS Catalina, macOS Big Sur | macOS Big Sur |

Продовження таблиці 6.1

|  |  |  |
| --- | --- | --- |
|  | **Мінімальні** | **Рекомендовані** |
| Процесор | Intel® Pentium® ІІІ  1.0 GHz або  AMD Athlon™ 1.0 GHz або M1 | Intel® Pentium® D або AMD Athlon™ 64 X2 або M1 |
| Оперативна пам’ять | 1 GB | 2 GB |
| Відеоадаптер | Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Дисплей | 1024х768 | 1024х768 і краще |
| Прилади введення | Клавіатура (необов’язково), комп’ютерна миша/тачпад | |
| Додаткове програмне забезпечення | не потрібне | |

Висновки

Отже, у розділі 1 була поставлена задача, сформульовані можливості користувача при взаємодії з системою.

У розділі 2 були описані правила, за якими працює програма, описано можливості ходів різних фігур, зазначено кількість можливих фігур, що можуть розташовуватися на дошці та розглянуто декілька нюансів відносно розміщення королів.

У розділі 3 було описано загальний алгоритм, алгоритм методу додавання фігури, видалення фігури, переміщення фігури, аналізу дошки, перевірки чи під шагом король, а також отримання можливих ходів для кожного типу фігур.

У розділі 4 було представлено та описано діаграму класів. Було написано програмне забезпечення на мові програмування Swift для комп’ютерів на операційній системі macOS. Були описані користувацькі та стандартні функції.

У розділі 5 було проведено тестування програмного забезпечення на всі можливі функції та типи розташування.

У розділі 6 було написано інструкцію користувача та формалізовано системні вимоги.

Перелік посилань

1. Шахи [Електронний ресурс] Доступ за посиланням:

<https://uk.wikipedia.org/wiki/Шахи>

1. Правила гри в шахи ФІДЕ [Електронний ресурс] Доступ за посиланням:

<http://ukrchess.org.ua/arbiter/laws_07_2017.pdf>

Додаток А Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко Максим Миколайович

«09» березня 2021 р.

Виконавець:

Студент Храмченко Анатолій Сергійович

«08» березня 2021р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: Гра “Шахові фігури”

з дисципліни:

«Основи програмування»

Київ 2021

* 1. *Мета*: Метою курсової роботи є розробка програми, що за заданим списком фігур та координат, розташує їх на шахівниці та покаже, яка фігура може побити яку.
  2. *Дата початку роботи*: «08» березня 2021 р.
  3. *Дата закінчення роботи*: «31»травня 2021 р.
  4. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:
   * Можливість відображення поля
   * Можливість за введеними у текстовий файл даними розмістити на шахівниці такі фігури: король, ферзь, тура, слон, кінь
   * Можливість обирати одну с фігур (король, ферзь, тура, слон або кінь)
   * Можливість додати фігуру (король, ферзь, тура, слон або кінь) заданого кольору
   * Можливість контролю кількості фігур на дошці, у відповідності з правилами гри у шахи
   * Можливість видалити певну фігуру
   * Можливість вивести повідомлення про помилку, у разі розташування в одній клітинці декількох фігур
   * Можливість показати яка фігура яку може побити
   * Можливість збереження розташування та результатів роботи у текстовому файлі
2. Нефункціональні вимоги:
   * Інтерфейс має бути зручним для користувача
   * Під час написання програми має бути використано систему контролю версій Git
   * Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до29.03.2021 р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до 04.04.2021р.)
3. Розробка програмного забезпечення (до 15.04.2021р.)
4. Тестування розробленої програми (до 29.04.2021р.)
5. Розробка пояснювальної записки (до 02.05.2021 р.).
6. Захист курсової роботи (до 04.05.2021 р.).
   1. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

Додаток Б Тексти програмного коду

*Тексти програмного коду програмного забезпечення гри “Шахові фігури”*

(Найменування програми)

*CD-RW*

(Вид носія даних)

*57*

(Обсяг програми)

Студента групи ІП-01 1 курсу

Храмченко А.С.

MainVC.swift

import UIKit

class MainVC: UIViewController {

var arrangement = Arrangement()

var color: String = "w" //нинішній колір

var lastAction = 0 // 0 - никакое из, 1 - в пустую клетку, 2 - в заполненую клетку, 3 - в фигуру, 4 - получить результат

let file = FileOperation()

var figuresMenu: [UIImageView] = [] //зображення в меню фігур

@IBOutlet var figuresMenuButton: [UIButton]! //кнопки меню

@IBOutlet var board: [UIButton]! //кнопки дошки

var whiteCell = UIColor(cgColor: CGColor(red: 1, green: 1, blue: 1, alpha: 1)) //колір білої клітинка

var blackCell = UIColor(cgColor: CGColor(red: 1, green: 1, blue: 1, alpha: 1)) //колір чорної клітинки

//switches

@IBOutlet weak var colorSwitch: UISegmentedControl!

@IBOutlet weak var pageSwitch: UISegmentedControl!

@IBOutlet weak var errorLable: UILabel!

//groupe1

@IBOutlet weak var addFigureLable: UILabel!

@IBOutlet weak var typeLable: UILabel!

@IBOutlet weak var addTypeText: UITextField!

@IBOutlet weak var addXText: UITextField!

@IBOutlet weak var addYText: UITextField!

@IBOutlet weak var addCoordinatesLable: UILabel!

@IBOutlet weak var addButtonVar: UIButton!

@IBOutlet weak var delFigureLable: UILabel!

@IBOutlet weak var delCoordinateLable: UILabel!

@IBOutlet weak var delXText: UITextField!

@IBOutlet weak var delYText: UITextField!

@IBOutlet weak var delButtonVar: UIButton!

@IBOutlet weak var moveLable: UILabel!

@IBOutlet weak var moveOldCoordinatesLable: UILabel!

@IBOutlet weak var moveOldCoordinatesXText: UITextField!

@IBOutlet weak var moveOldCoordinatesYText: UITextField!

@IBOutlet weak var moveButtonVar: UIButton!

@IBOutlet weak var moveNewCoordinatesLable: UILabel!

@IBOutlet weak var moveNewCoordinatesXText: UITextField!

@IBOutlet weak var moveNewCoordinatesYText: UITextField!

@IBOutlet weak var resoultView: UITextView!

@IBOutlet weak var getResoultButtonVar: UIButton!

@IBOutlet weak var getResForLable: UILabel!

@IBOutlet weak var getResTypeText: UITextField!

@IBOutlet weak var getResX: UITextField!

@IBOutlet weak var getResY: UITextField!

@IBOutlet weak var setAllButton: UIButton!

@IBOutlet weak var clearButton: UIButton!

//groupe2

@IBOutlet weak var pathText: UITextField!

@IBOutlet weak var pathLable: UILabel!

@IBOutlet weak var loadPossitionButtonVar: UIButton!

@IBOutlet weak var savePossitionButtonVar: UIButton!

@IBOutlet weak var saveResoulButtonVar: UIButton!

//groupe3

@IBOutlet weak var helpView: UITextView!

override func viewDidLoad() {

super.viewDidLoad()

whiteCell = board[0].backgroundColor!

blackCell = board[1].backgroundColor!

self.overrideUserInterfaceStyle = .light //робимо тему завжди світлою

arrangement.addFigure(figure: King(type: "King", color: "w", x: "E", y: 1))

arrangement.addFigure(figure: King(type: "King", color: "b", x: "E", y: 8)) //створюємо 2 королів

self.view.addSubview(arrangement.figures[0].image)

self.view.addSubview(arrangement.figures[1].image)

addConstrain(numbFigure: 0)

addConstrain(numbFigure: 1)

addFiguresMenu()

} //запускаєтся при завантаженні Вью Контроллера

@IBAction func pageSwitchAction(\_ sender: Any) {

if (lastAction != 4) {

lastAction = 0

}

if pageSwitch.selectedSegmentIndex == 0 {

installedGroupe1()

}

if pageSwitch.selectedSegmentIndex == 1 {

installedGroupe2()

}

if pageSwitch.selectedSegmentIndex == 2 {

installedGroupe3()

}

} //перемикач стоорінок

@IBAction func colorSwitchAction(\_ sender: Any) {

if colorSwitch.selectedSegmentIndex == 0 {

color = "w"

} else {

color = "b"

}

changeColorFiguresMenu()

if lastAction == 4 {

lastAction = 0

changeColorWhite()

getResoultAction(0)

}

} //перемикач кольору

@IBAction func addFigureAction(\_ sender: Any) {

lastAction = 0

changeColorWhite()

if !addTypeText.text!.isNormal(indType: 0) {

Errors.isError(type: 0)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else if !addXText.text!.isNormal(indType: 1) {

Errors.isError(type: 1)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else if !addYText.text!.isNormal(indType: 2) {

Errors.isError(type: 2)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

errorLable.text = ""

switch addTypeText.text! {

case "King":

arrangement.addFigure(figure: King(type: addTypeText.text!, color: color, x: addXText.text!.uppercased(), y: Int(addYText.text!)!))

case "Queen":

arrangement.addFigure(figure: Queen(type: addTypeText.text!, color: color, x: addXText.text!.uppercased(), y: Int(addYText.text!)!))

case "Tour":

arrangement.addFigure(figure: Tour(type: addTypeText.text!, color: color, x: addXText.text!.uppercased(), y: Int(addYText.text!)!))

case "Elephant":

arrangement.addFigure(figure: Elephant(type: addTypeText.text!, color: color, x: addXText.text!.uppercased(), y: Int(addYText.text!)!))

case "Knight":

arrangement.addFigure(figure: Knight(type: addTypeText.text!, color: color, x: addXText.text!.uppercased(), y: Int(addYText.text!)!))

case "Pawn":

arrangement.addFigure(figure: Pawn(type: addTypeText.text!, color: color, x: addXText.text!.uppercased(), y: Int(addYText.text!)!))

default:

print("er")

}

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

self.view.addSubview(arrangement.figures[arrangement.figures.count-1].image)

addConstrain(numbFigure: arrangement.figures.count-1)

}

}

} //додавання фігури

@IBAction func delFigureAction(\_ sender: Any) {

lastAction = 0

changeColorWhite()

if !delXText.text!.isNormal(indType: 1) {

Errors.isError(type: 1)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else if !delYText.text!.isNormal(indType: 2) {

Errors.isError(type: 2)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

errorLable.text = ""

arrangement.dellFigure(x: delXText.text!.uppercased(), y: Int(delYText.text!)!)

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

}

}

} //видалення фігури

@IBAction func moveFigureAction(\_ sender: Any) {

lastAction = 0

changeColorWhite()

if !moveOldCoordinatesXText.text!.isNormal(indType: 1) && !moveNewCoordinatesXText.text!.isNormal(indType: 1) {

Errors.isError(type: 1)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else if !moveOldCoordinatesYText.text!.isNormal(indType: 2) && !moveNewCoordinatesYText.text!.isNormal(indType: 2) {

Errors.isError(type: 2)

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

errorLable.text = ""

arrangement.moveFigure(oldX: moveOldCoordinatesXText.text!.uppercased(), oldY: Int(moveOldCoordinatesYText.text!)!, newX: moveNewCoordinatesXText.text!.uppercased(), newY: Int(moveNewCoordinatesYText.text!)!)

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

self.view.addSubview(arrangement.figures[arrangement.figures.count-1].image)

addConstrain(numbFigure: arrangement.figures.count-1)

}

}

} //переміщення фігури

@IBAction func getResoultAction(\_ sender: Any) {

lastAction = 4

changeColorWhite()

errorLable.text = ""

let toPrintArr = arrangement.analysisBoard(color: color)

var toPrintStr = ""

if (getResTypeText.text != "All") && (!((getResTypeText.text?.isNormal(indType: 0))!)) {

Errors.isError(type: 0)

}

if (getResX.text != "") && (!((getResX.text?.isNormal(indType: 1))!)) {

Errors.isError(type: 1)

}

if (getResY.text != "") && (!((getResY.text?.isNormal(indType: 2))!)) {

Errors.isError(type: 2)

}

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

if toPrintArr.isEmpty {

toPrintStr = "Nothing can be defeated"

} else {

for i in 0..<toPrintArr.count {

if (getResTypeText.text == "All") || ((getResTypeText.text == arrangement.figureToGo[i].type) && (getResX.text == arrangement.figureToGo[i].x) && (getResY.text == String(arrangement.figureToGo[i].y))) {

toPrintStr += toPrintArr[i]

toPrintStr += "\n"

}

}

}

resoultView.text = toPrintStr

changeColorRed()

}

} //отримання результату

@IBAction func saveResoultAction(\_ sender: Any) {

errorLable.text = ""

let path = pathText.text!

let res = resoultView.text!

file.recordRes(path: path, res: res)

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

}

} //збереження результату у файл

@IBAction func loadPossitionAction(\_ sender: Any) {

let path = pathText.text!

errorLable.text = ""

let pos = file.readPos(path: path)

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

} else {

arrangement.clearBoard()

var counter = 0

var colorThis = ""

var type = ""

var x = ""

var y = ""

for i in pos {

if i == "\_" {

counter += 1

} else if i == "|" {

addTypeText.text = type

color = colorThis

addXText.text = x

addYText.text = y

counter = 0

colorThis = ""

type = ""

x = ""

y = ""

addFigureAction(0)

} else {

switch counter%4 {

case 0:

colorThis.append(i)

case 1:

type.append(i)

case 2:

x.append(i)

case 3:

y.append(i)

default:

print("er")

}

}

}

addTypeText.text = ""

colorSwitchAction(0)

addXText.text = ""

addYText.text = ""

lastAction = 0

}

} //завантаження позиції з файлу

@IBAction func savePossitionAction(\_ sender: Any) {

errorLable.text = ""

let path = pathText.text!

let pos = arrangement.getPos()

file.recordPos(path: path, pos: pos)

if Errors.flag {

errorLable.text = Errors.errorMessage

Errors.errorSolved()

}

} //збереження позиції у файл

@IBAction func tapBoard(\_ sender: Any) {

changeColorWhite()

(sender as! UIButton).backgroundColor = UIColor.yellow

let positionButton = position(x: Int((sender as! UIButton).frame.midX), y: Int((sender as! UIButton).frame.midY))

if arrangement.isInPossition(x: positionButton[0], y: Int(positionButton[1])!) {

delXText.text = positionButton[0]

delYText.text = positionButton[1]

moveOldCoordinatesXText.text = positionButton[0]

moveOldCoordinatesYText.text = positionButton[1]

if arrangement.colorInPossition == color {

getResX.text = positionButton[0]

getResY.text = positionButton[1]

getResTypeText.text = arrangement.figureInPossition

if lastAction == 4 {

getResoultAction(0)

} else {

lastAction = 2

}

} else if lastAction != 4 {

lastAction = 2

}

} else {

addXText.text = positionButton[0]

addYText.text = positionButton[1]

moveNewCoordinatesXText.text = positionButton[0]

moveNewCoordinatesYText.text = positionButton[1]

if lastAction == 3 {

addFigureAction(0)

} else if lastAction == 2 {

moveFigureAction(0)

} else {

lastAction = 1

}

}

} //натискання на дошку

@IBAction func tapFigure(\_ sender: Any) {

changeColorWhite()

let figuresNames = ["Pawn", "Knight", "Elephant", "Tour", "Queen", "King"]

addTypeText.text = figuresNames[(Int((sender as! UIButton).frame.midX) - 86)/105]

figuresMenuButton[(Int((sender as! UIButton).frame.midX) - 86)/105].backgroundColor = UIColor.yellow

lastAction = 3

} //натискання на фігуру з меню фігур

@IBAction func setAllAction(\_ sender: Any) {

getResTypeText.text = "All"

getResX.text = ""

getResY.text = ""

if lastAction == 4 {

getResoultAction(0)

} else {

lastAction = 0

}

} //натискання на кнопку "Всі"

@IBAction func clearAction(\_ sender: Any) {

lastAction = 0

resoultView.text = ""

changeColorWhite()

} //натискання на кнопку "Очистити"

func changeColorRed() {

var colorCell: UIColor

for i in 0..<arrangement.cellToGo.count {

if (getResTypeText.text == "All") || ((getResTypeText.text == arrangement.figureToGo[i].type) && (getResX.text == arrangement.figureToGo[i].x) && (getResY.text == String(arrangement.figureToGo[i].y))) {

for j in 0..<arrangement.cellToGo[i].count/2 {

colorCell = board[(7-arrangement.cellToGo[i][j\*2+1])\*8 + (arrangement.cellToGo[i][j\*2])].backgroundColor!

if j == 0 {

colorCell = UIColor.yellow

} else if j == arrangement.cellToGo[i].count/2-1 {

colorCell = UIColor.red

} else if board[(7-arrangement.cellToGo[i][j\*2+1])\*8 + (arrangement.cellToGo[i][j\*2])].backgroundColor != UIColor.red && board[(7-arrangement.cellToGo[i][j\*2+1])\*8 + (arrangement.cellToGo[i][j\*2])].backgroundColor != UIColor.yellow {

board[((7-arrangement.cellToGo[i][j\*2+1])\*8 + (arrangement.cellToGo[i][j\*2]))].setImage(UIImage(named: "circle"), for: [])

}

board[((7-arrangement.cellToGo[i][j\*2+1])\*8 + (arrangement.cellToGo[i][j\*2]))].backgroundColor = colorCell

}

}

}

} //встановлення кольорів бою

func changeColorWhite() {

for i in figuresMenuButton {

i.backgroundColor = UIColor.clear

}

if lastAction != 4 {

getResTypeText.text = "All"

getResX.text = ""

getResY.text = ""

}

for i in 0...63 {

if (i%8 + Int(i/8)) % 2 == 0 {

board[i].backgroundColor = whiteCell

} else {

board[i].backgroundColor = blackCell

}

if board[i].imageView?.image == UIImage(named: "circle") {

board[i].setImage(UIImage(named: "AccentColor"), for: [])

}

}

} //встановлення початкових кольорів

func addFiguresMenu() {

let nameOfImages = ["wPawn", "wKnight", "wElephant", "wTour", "wQueen", "wKing"]

for i in 0...5 {

figuresMenu.append(UIImageView(frame: CGRect(x: figuresMenuButton[i].frame.midX, y: 811, width: 90, height: 90)))

figuresMenu[i].image = UIImage(named: nameOfImages[i])

self.view.addSubview(figuresMenu[i])

figuresMenu[i].translatesAutoresizingMaskIntoConstraints = false

let horizontalConstraint = NSLayoutConstraint(item: figuresMenu[i], attribute: NSLayoutConstraint.Attribute.centerX, relatedBy: NSLayoutConstraint.Relation.equal, toItem: figuresMenuButton[i], attribute: NSLayoutConstraint.Attribute.leading, multiplier: 1, constant: 45)

let verticalConstraint = NSLayoutConstraint(item: figuresMenu[i], attribute: NSLayoutConstraint.Attribute.centerY, relatedBy: NSLayoutConstraint.Relation.equal, toItem: figuresMenuButton[i], attribute: NSLayoutConstraint.Attribute.top, multiplier: 1, constant: 45)

let widthConstraint = NSLayoutConstraint(item: figuresMenu[i], attribute: NSLayoutConstraint.Attribute.width, relatedBy: NSLayoutConstraint.Relation.equal, toItem: nil, attribute: NSLayoutConstraint.Attribute.notAnAttribute, multiplier: 1, constant: 90)

let heightConstraint = NSLayoutConstraint(item: figuresMenu[i], attribute: NSLayoutConstraint.Attribute.height, relatedBy: NSLayoutConstraint.Relation.equal, toItem: nil, attribute: NSLayoutConstraint.Attribute.notAnAttribute, multiplier: 1, constant: 90)

view.addConstraints([horizontalConstraint, verticalConstraint, widthConstraint, heightConstraint])

}

} //малює меню фігур

func changeColorFiguresMenu() {

let figuresNames = ["Pawn", "Knight", "Elephant", "Tour", "Queen", "King"]

for i in 0..<figuresMenu.count {

figuresMenu[i].image = UIImage(named: color + figuresNames[i])

}

} //змінює колір для меню фігур

func position(x: Int, y:Int) -> [String] {

var res: [String] = []

let realX = (x - 39) / 90

switch realX {

case 0:

res.append("A")

case 1:

res.append("B")

case 2:

res.append("C")

case 3:

res.append("D")

case 4:

res.append("E")

case 5:

res.append("F")

case 6:

res.append("G")

case 7:

res.append("H")

default:

res.append("")

}

res.append(String(8 - (y-41) / 90))

return res

} //повертає позицію відповідно до координат

func addConstrain(numbFigure: Int) {

arrangement.figures[numbFigure].image.translatesAutoresizingMaskIntoConstraints = false

let horizontalConstraint = NSLayoutConstraint(item: arrangement.figures[numbFigure].image, attribute: NSLayoutConstraint.Attribute.centerX, relatedBy: NSLayoutConstraint.Relation.equal, toItem: board[arrangement.figures[numbFigure].number], attribute: NSLayoutConstraint.Attribute.leading, multiplier: 1, constant: 45)

let verticalConstraint = NSLayoutConstraint(item: arrangement.figures[numbFigure].image, attribute: NSLayoutConstraint.Attribute.centerY, relatedBy: NSLayoutConstraint.Relation.equal, toItem: board[arrangement.figures[numbFigure].number], attribute: NSLayoutConstraint.Attribute.top, multiplier: 1, constant: 45)

let widthConstraint = NSLayoutConstraint(item: arrangement.figures[numbFigure].image, attribute: NSLayoutConstraint.Attribute.width, relatedBy: NSLayoutConstraint.Relation.equal, toItem: nil, attribute: NSLayoutConstraint.Attribute.notAnAttribute, multiplier: 1, constant: 90)

let heightConstraint = NSLayoutConstraint(item: arrangement.figures[numbFigure].image, attribute: NSLayoutConstraint.Attribute.height, relatedBy: NSLayoutConstraint.Relation.equal, toItem: nil, attribute: NSLayoutConstraint.Attribute.notAnAttribute, multiplier: 1, constant: 90)

view.addConstraints([horizontalConstraint, verticalConstraint, widthConstraint, heightConstraint])

} //додає констрейни до зображень

func installedGroupe1() {

pathText.isHidden = true

pathLable.isHidden = true

loadPossitionButtonVar.isHidden = true

savePossitionButtonVar.isHidden = true

saveResoulButtonVar.isHidden = true

addFigureLable.isHidden = false

typeLable.isHidden = false

addTypeText.isHidden = false

addXText.isHidden = false

addYText.isHidden = false

addCoordinatesLable.isHidden = false

addButtonVar.isHidden = false

delFigureLable.isHidden = false

delCoordinateLable.isHidden = false

delXText.isHidden = false

delYText.isHidden = false

delButtonVar.isHidden = false

moveLable.isHidden = false

moveOldCoordinatesLable.isHidden = false

moveOldCoordinatesXText.isHidden = false

moveOldCoordinatesYText.isHidden = false

moveButtonVar.isHidden = false

moveNewCoordinatesLable.isHidden = false

moveNewCoordinatesXText.isHidden = false

moveNewCoordinatesYText.isHidden = false

resoultView.isHidden = false

getResoultButtonVar.isHidden = false

getResForLable.isHidden = false

getResTypeText.isHidden = false

getResX.isHidden = false

getResY.isHidden = false

setAllButton.isHidden = false

clearButton.isHidden = false

helpView.isHidden = true

errorLable.isHidden = false

} //робить видимими елементи 1ї сторінки

func installedGroupe2() {

pathText.isHidden = false

pathLable.isHidden = false

loadPossitionButtonVar.isHidden = false

savePossitionButtonVar.isHidden = false

saveResoulButtonVar.isHidden = false

addFigureLable.isHidden = true

typeLable.isHidden = true

addTypeText.isHidden = true

addXText.isHidden = true

addYText.isHidden = true

addCoordinatesLable.isHidden = true

addButtonVar.isHidden = true

delFigureLable.isHidden = true

delCoordinateLable.isHidden = true

delXText.isHidden = true

delYText.isHidden = true

delButtonVar.isHidden = true

moveLable.isHidden = true

moveOldCoordinatesLable.isHidden = true

moveOldCoordinatesXText.isHidden = true

moveOldCoordinatesYText.isHidden = true

moveButtonVar.isHidden = true

moveNewCoordinatesLable.isHidden = true

moveNewCoordinatesXText.isHidden = true

moveNewCoordinatesYText.isHidden = true

resoultView.isHidden = true

getResoultButtonVar.isHidden = true

getResForLable.isHidden = true

getResTypeText.isHidden = true

getResX.isHidden = true

getResY.isHidden = true

setAllButton.isHidden = true

clearButton.isHidden = true

helpView.isHidden = true

errorLable.isHidden = false

} //робить видимими елементи 2ї сторінки

func installedGroupe3() {

pathText.isHidden = true

pathLable.isHidden = true

loadPossitionButtonVar.isHidden = true

savePossitionButtonVar.isHidden = true

saveResoulButtonVar.isHidden = true

addFigureLable.isHidden = true

typeLable.isHidden = true

addTypeText.isHidden = true

addXText.isHidden = true

addYText.isHidden = true

addCoordinatesLable.isHidden = true

addButtonVar.isHidden = true

delFigureLable.isHidden = true

delCoordinateLable.isHidden = true

delXText.isHidden = true

delYText.isHidden = true

delButtonVar.isHidden = true

moveLable.isHidden = true

moveOldCoordinatesLable.isHidden = true

moveOldCoordinatesXText.isHidden = true

moveOldCoordinatesYText.isHidden = true

moveButtonVar.isHidden = true

moveNewCoordinatesLable.isHidden = true

moveNewCoordinatesXText.isHidden = true

moveNewCoordinatesYText.isHidden = true

resoultView.isHidden = true

getResoultButtonVar.isHidden = true

getResForLable.isHidden = true

getResTypeText.isHidden = true

getResX.isHidden = true

getResY.isHidden = true

setAllButton.isHidden = true

clearButton.isHidden = true

helpView.isHidden = false

errorLable.isHidden = true

} //робить видимими елементи 3ї сторінки

}

Pawn.swift

import Foundation

class Pawn: Figure {

override func getMovies() -> [[Int]] {

var resoult: [[Int]] = [[]]

if self.color == "w" {

if self.xValue > 1 && self.y < 8 {

resoult.append([self.xValue-2, self.y])

}

if self.xValue < 8 && self.y < 8 {

resoult.append([self.xValue, self.y])

}

} else {

if self.xValue > 1 && self.y > 1 {

resoult.append([self.xValue-2, self.y-2])

}

if self.xValue < 8 && self.y > 1 {

resoult.append([self.xValue, self.y-2])

}

}

return resoult

} //ходи для пішака

}

Knight.swift

import Foundation

class Knight: Figure {

override func getMovies() -> [[Int]] {

var resoult: [[Int]] = [[]]

if self.xValue < 8 && self.y < 7 {

resoult.append([self.xValue, self.y+1])

}

if self.xValue < 7 && self.y < 8 {

resoult.append([self.xValue+1, self.y])

}

if self.xValue < 7 && self.y > 1 {

resoult.append([self.xValue+1, self.y-2])

}

if self.xValue < 8 && self.y > 2 {

resoult.append([self.xValue, self.y-3])

}

if self.xValue > 1 && self.y > 2 {

resoult.append([self.xValue-2, self.y-3])

}

if self.xValue > 2 && self.y > 1 {

resoult.append([self.xValue-3, self.y-2])

}

if self.xValue > 2 && self.y < 8 {

resoult.append([self.xValue-3, self.y])

}

if self.xValue > 1 && self.y < 7 {

resoult.append([self.xValue-2, self.y+1])

}

return resoult

} //ходи для коня

override func getCells(resX: Int, resY: Int) -> [Int] {

var result: [Int] = []

result.append(self.xValue-1)

result.append(self.y-1)

let defX = resX - self.xValue + 1

let defY = resY - self.y + 1

if abs(defX) == 2 {

result.append(self.xValue - 1 + (defX/2))

result.append(self.y - 1)

result.append(self.xValue - 1 + defX)

result.append(self.y - 1)

result.append(self.xValue - 1 + defX)

result.append(self.y - 1 + defY)

}

if abs(defY) == 2 {

result.append(self.xValue - 1)

result.append(self.y - 1 + (defY/2))

result.append(self.xValue - 1)

result.append(self.y - 1 + defY)

result.append(self.xValue - 1 + defX)

result.append(self.y - 1 + defY)

}

return result

} //повертає шлях коня до клітинки

}

Elephant.swift

import Foundation

class Elephant: Figure {

override func getMovies() -> [[Int]] {

var resoult: [[Int]] = [[]]

var movX = 0

var movY = 0

var directionOfMov: [Int] = []

movX = self.xValue

movY = self.y

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX += 1

movY += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-2

movY = self.y

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX -= 1

movY += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-2

movY = self.y-2

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX -= 1

movY -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue

movY = self.y-2

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX += 1

movY -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

return resoult

} //ходи для слона

}

Tour.swift

import Foundation

class Tour: Figure {

override func getMovies() -> [[Int]] {

var resoult: [[Int]] = [[]]

var movX = 0

var movY = 0

var directionOfMov: [Int] = []

movX = self.xValue

movY = self.y - 1

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue - 1

movY = self.y

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movY += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-2

movY = self.y-1

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-1

movY = self.y-2

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movY -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

return resoult

}//ходи для тури

}

Queen.swift

import Foundation

class Queen: Figure {

override func getMovies() -> [[Int]] {

var resoult: [[Int]] = [[]]

var movX = 0

var movY = 0

var directionOfMov: [Int] = []

movX = self.xValue

movY = self.y - 1

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue

movY = self.y

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX += 1

movY += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue - 1

movY = self.y

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movY += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-2

movY = self.y

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX -= 1

movY += 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-2

movY = self.y-1

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-2

movY = self.y-2

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX -= 1

movY -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue-1

movY = self.y-2

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movY -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

directionOfMov = []

movX = self.xValue

movY = self.y-2

while movX>=0 && movX<=7 && movY>=0 && movY<=7 {

directionOfMov.append(movX)

directionOfMov.append(movY)

movX += 1

movY -= 1

}

if directionOfMov != [] {

resoult.append(directionOfMov)

}

return resoult

}//ходи для ферзя

}

King.swift

import Foundation

class King: Figure {

override func getMovies() -> [[Int]] {

var resoult: [[Int]] = [[]]

if self.xValue < 8 {

resoult.append([self.xValue, self.y-1])

}

if self.xValue < 8 && self.y < 8 {

resoult.append([self.xValue, self.y])

}

if self.y < 8 {

resoult.append([self.xValue-1, self.y])

}

if self.xValue > 1 && self.y < 8 {

resoult.append([self.xValue-2, self.y])

}

if self.xValue > 1 {

resoult.append([self.xValue-2, self.y-1])

}

if self.xValue > 1 && self.y > 1 {

resoult.append([self.xValue-2, self.y-2])

}

if self.y > 1 {

resoult.append([self.xValue-1, self.y-2])

}

if self.xValue < 8 && self.y > 1 {

resoult.append([self.xValue, self.y-2])

}

return resoult

} //ходи для короля

}

Figure.swift

import UIKit

class Figure {

let type: String //тип фігури

private let typeInd: Int //індикатор фігури цифрами

let color: String

let x: String

let xValue: Int

let y: Int

var isReal = true

let number: Int //координата на дошці (від 0 до 63)

let image: UIImageView //зображення

init(type: String, color: String, x: String, y: Int) {

self.type = type

switch type {

case "Pawn":

typeInd = 1

case "Knight":

typeInd = 2

case "Elephant":

typeInd = 3

case "Tour":

typeInd = 4

case "Queen":

typeInd = 5

default:

typeInd = 6

}

self.color = color

self.x = x

self.y = y

switch x {

case "A":

xValue = 1

case "B":

xValue = 2

case "C":

xValue = 3

case "D":

xValue = 4

case "E":

xValue = 5

case "F":

xValue = 6

case "G":

xValue = 7

default:

xValue = 8

}

image = UIImageView(frame: CGRect(x: 41+(self.xValue-1)\*90, y: 77+(8-self.y)\*90, width: 90, height: 90))

image.image = UIImage(named: color+type)

number = (8 - y)\*8 + xValue - 1

} //конструктор

func isEqually(x: String, y:Int) -> Bool {

if x == self.x && y == self.y {

return true

}

return false

} //чи рівна позиція

func move(x:String, y:Int) -> Figure {

let figureMove: Figure

switch self.type {

case "King":

figureMove = King(type: type, color: color, x: x, y: y)

case "Queen":

figureMove = Queen(type: type, color: color, x: x, y: y)

case "Tour":

figureMove = Tour(type: type, color: color, x: x, y: y)

case "Elephant":

figureMove = Elephant(type: type, color: color, x: x, y: y)

case "Knight":

figureMove = Knight(type: type, color: color, x: x, y: y)

default:

figureMove = Pawn(type: type, color: color, x: x, y: y)

}

return figureMove

} //змінює координати фігури(повртає таку ж фігуру, з іншими координатами)

func getInd() -> Int {

if color == "w" {

return typeInd

} else {

return -typeInd

}

} //повертає індикатор фігури

func getMovies() -> [[Int]] {

return [[]]

}

func getCells(resX: Int, resY: Int) -> [Int] {

return []

}

func getDescription() -> String {

let res = "\(color)\_\(type)\_\(x)\_\(y)|"

return res

} //опис фігури рядком

}

Errors.swift

import Foundation

class Errors {

static var flag = false //чи є помилка

static var errorMessage = "" //текст помилки

static let errorMessages = [

"Such a figure does not exist",//0

"Enter the correct x coordinate",//1

"Enter the correct y coordinate",//2

"The king is under the step",//3

"Non-existent file path",//4

"Can't add more than one king",//5

"There must be at least 1 king",//6

"The number of figures does not match the rules",//7

"There is no figure in this position",//8

"There is a figure in this position",//9

"The pawn cannot be added to the last row",//10

"Kings cannot stand near to each other"//11

] //усі помилки

static func isError(type: Int) {

flag = true

errorMessage = "Error \(type): " + errorMessages[type] + ". Please read help!"

} //помилку виявлено

static func errorSolved() {

flag = false

errorMessage = ""

} //помилку виведено

}

ExString.swift

import Foundation

extension String {

private func isFigure() -> Bool {

let figures = ["King", "Queen", "Tour", "Elephant", "Knight", "Pawn"]

if figures.contains(self) {

return true

} else {

return false

}

} //чи рядок є типом фігури

private func isXCoordinate() -> Bool {

let coordinates = ["A", "B", "C", "D", "E", "F", "G", "H"]

if coordinates.contains(self.uppercased()) {

return true

} else {

return false

}

} //чи рядок є координатою по х

private func isYCoordinate() -> Bool {

let coordinates = ["1", "2", "3", "4", "5", "6", "7", "8"]

if coordinates.contains(self) {

return true

} else {

return false

}

} //чи рядок є координатою по у

func isNormal (indType: Int) -> Bool {

var res: Bool

switch indType {

case 0:

res = isFigure()

case 1:

res = isXCoordinate()

case 2:

res = isYCoordinate()

default:

res = false

}

return res

} //чи є рядок вірним

}

Arrangement.swift

import UIKit

class Arrangement {

var figures: [Figure] = [] //перелік усіх фігур

private var board = [[Int]] (repeating: [Int] (repeating: 0, count: 8), count: 8) //дошка

//лічільники фігур

private var countWKing = 0

private var countWPawn = 0

private var countWFigures = 0

private var similarWFigures = 0

private var countBKing = 0

private var countBPawn = 0

private var countBFigures = 0

private var similarBFigures = 0

var cellToGo: [[Int]] = []//клітинки, по яким потрібно пройти для бою

var figureToGo: [Figure] = []//фігури, які будуть бити

var figureInPossition = ""//тип фігури на позиції

var colorInPossition = ""//колір на позиції

private func fillingBoard() {

for i in 0..<figures.count {

if figures[i].isReal {

board[figures[i].xValue-1][figures[i].y-1] = figures[i].getInd()

} else {

board[figures[i].xValue-1][figures[i].y-1] = 0

}

}

} //заповнення матриці

func isInPossition(x: String, y: Int) -> Bool {

var res = false

for i in 0..<figures.count {

if figures[i].isEqually(x: x, y: y) && figures[i].isReal {

res = true

figureInPossition = figures[i].type

colorInPossition = figures[i].color

break

}

}

return res

} //чи є фігура на позиції

private func isKingOnStep(colorInt: Int, boardAfter: [[Int]]) -> Bool {

let colors = ["b", "", "w"]

var move: [[Int]]

var pointX: Int

var pointY: Int

for i in 0..<figures.count {

if figures[i].isReal && figures[i].color == colors[colorInt+1] {

move = figures[i].getMovies()

for j in 1..<move.count {

for numbOfPoint in 0..<move[j].count/2 {

pointX = move[j][numbOfPoint\*2]

pointY = move[j][numbOfPoint\*2+1]

if boardAfter[pointX][pointY] != 0 {

if colorInt\*boardAfter[pointX][pointY] < 0 && abs(boardAfter[pointX][pointY]) == 6 {

return true

}

break

}

}

}

}

}

return false

} //чи під шагом король

private func isSimilar(figure: Figure) -> Bool {

let maxCount: Int

var count = 0

figure.type == "Queen" ? (maxCount = 1) : (maxCount = 2)

for i in figures {

if i.color == figure.color && i.type == figure.type && i.isReal {

count += 1

if count == maxCount {

return true

}

}

}

return false

} //чи є фігура такого ж типу

func addFigure(figure: Figure) {

if isInPossition(x: figure.x, y: figure.y) {

Errors.isError(type: 9)

return

}

if figure.color == "w" {

if figure.type == "King" {

countWKing += 1

if countWKing > 1 {

Errors.isError(type: 5)

countWKing -= 1

return

}

} else if figure.type == "Pawn" {

countWPawn += 1

if countWPawn+countWFigures > 15 || (countWPawn + similarWFigures)>8 {

Errors.isError(type: 7)

countWPawn -= 1

return

}

if figure.y == 1 || figure.y == 8 {

Errors.isError(type: 10)

countWPawn -= 1

return

}

} else {

countWFigures += 1

if isSimilar(figure: figure) {

similarWFigures += 1

}

if countWPawn+countWFigures > 15 {

Errors.isError(type: 7)

countWFigures -= 1

return

}

if (countWPawn + similarWFigures)>8 {

Errors.isError(type: 7)

countWFigures -= 1

similarWFigures -= 1

return

}

}

} else {

if figure.type == "King" {

countBKing += 1

if countBKing > 1 {

Errors.isError(type: 5)

countBKing -= 1

return

}

} else if figure.type == "Pawn" {

countBPawn += 1

if countBPawn+countBFigures > 15 || (countWPawn + similarBFigures)>8 {

Errors.isError(type: 7)

countBPawn -= 1

return

}

if figure.y == 1 || figure.y == 8 {

Errors.isError(type: 10)

countBPawn -= 1

return

}

} else {

countBFigures += 1

if isSimilar(figure: figure) {

similarBFigures += 1

}

if countBPawn+countBFigures > 15 {

Errors.isError(type: 7)

countBFigures -= 1

return

}

if (countBPawn + similarBFigures)>8 {

Errors.isError(type: 7)

countBFigures -= 1

similarBFigures -= 1

return

}

}

}

figures.append(figure)

} //додавання фігури

func dellFigure(x: String, y: Int) {

var isInData = false

for i in 0..<figures.count {

if figures[i].isEqually(x: x, y: y) && figures[i].isReal {

if figures[i].color == "w" {

if figures[i].type == "King" {

countWKing -= 1

if countWKing < 1 {

Errors.isError(type: 6)

countWKing += 1

return

}

} else if figures[i].type == "Pawn" {

countWPawn -= 1

} else {

countWFigures -= 1

}

if isSimilar(figure: figures[i]) {

similarWFigures -= 1

}

} else {

if figures[i].type == "King" {

countBKing -= 1

if countBKing < 1 {

Errors.isError(type: 6)

countBKing += 1

return

}

} else if figures[i].type == "Pawn" {

countBPawn -= 1

} else {

countBFigures -= 1

}

if isSimilar(figure: figures[i]) {

similarBFigures -= 1

}

}

isInData = true

figures[i].image.isHidden = true

figures[i].isReal = false

break

}

}

if !isInData {

Errors.isError(type: 8)

}

} //видалення фігури

func moveFigure(oldX: String, oldY: Int, newX: String, newY: Int) {

if isInPossition(x: newX, y: newY) {

Errors.isError(type: 9)

return

}

var isInData = false

for i in 0..<figures.count {

if figures[i].isEqually(x: oldX, y: oldY) && figures[i].isReal {

if figures[i].color == "w" {

if figures[i].type == "Pawn" {

if newY == 1 || newY == 8 {

Errors.isError(type: 10)

return

}

countWPawn -= 1

} else if figures[i].type == "King" {

countWKing -= 1

} else {

countWFigures -= 1

if isSimilar(figure: figures[i]) {

similarWFigures -= 1

}

}

} else {

if figures[i].type == "Pawn" {

if newY == 1 || newY == 8 {

Errors.isError(type: 10)

return

}

countBPawn -= 1

} else if figures[i].type == "King" {

countBKing -= 1

} else {

countBFigures -= 1

if isSimilar(figure: figures[i]) {

similarBFigures -= 1

}

}

}

isInData = true

addFigure(figure: figures[i].move(x: newX, y: newY))

figures[i].image.isHidden = true

figures[i].isReal = false

break

}

}

if !isInData {

Errors.isError(type: 8)

}

} //переміщення фігури

func analysisBoard(color: String) -> [String] {

fillingBoard()

cellToGo = []

figureToGo = []

var tempArray: [Int] = []

let letters = ["A", "B", "C", "D", "E", "F", "G", "H"]

var res: [String] = []

var colorInt: Int

if color == "w" {

colorInt = 1

} else {

colorInt = -1

}

var move: [[Int]]

var newBoard: [[Int]]

var pointX: Int

var pointY: Int

for i in 0..<figures.count {

if figures[i].isReal && figures[i].color == color {

move = figures[i].getMovies()

for j in 1..<move.count {

for numbOfPoint in 0..<move[j].count/2 {

pointX = move[j][numbOfPoint\*2]

pointY = move[j][numbOfPoint\*2+1]

if board[pointX][pointY] != 0 {

if colorInt\*board[pointX][pointY] < 0 && abs(board[pointX][pointY]) != 6 {

newBoard = board

newBoard[pointX][pointY] = figures[i].getInd()

newBoard[figures[i].xValue-1][figures[i].y-1] = 0

if !isKingOnStep(colorInt: -colorInt, boardAfter: newBoard) {

res.append("\(figures[i].type) \(figures[i].x)\(figures[i].y) - \(letters[pointX])\(pointY+1)")

tempArray = []

tempArray.append(figures[i].xValue-1)

tempArray.append(figures[i].y-1)

figureToGo.append(figures[i])

if figures[i].type != "Knight" {

for tempInd in 0...numbOfPoint\*2+1{

tempArray.append(move[j][tempInd])

}

cellToGo.append(tempArray)

} else {

cellToGo.append(figures[i].getCells(resX: pointX, resY: pointY))

}

}

}

if colorInt\*board[pointX][pointY] < 0 && abs(board[pointX][pointY]) == 6 {

if figures[i].type == "King" {

Errors.isError(type: 11)

} else {

Errors.isError(type: 3)

}

return res

}

break

}

}

}

}

}

return res

} //аналіз дошки

func getPos()->String {

var res = ""

for i in figures {

if i.isReal {

res += i.getDescription()

}

}

return res

} //отримання позиції для запису у файл

func clearBoard() {

for i in figures {

i.image.isHidden = true

i.isReal = false

}

countWKing = 0

countWPawn = 0

countWFigures = 0

countBKing = 0

countBPawn = 0

countBFigures = 0

} //видалення всіх фігур

}

FileOperation.c

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "FileOperatios.h"

void recordToFile (const char\* path, const char \*data) {

FILE \*file;

file = fopen(path, "w");

if (file != NULL) {

fprintf(file,"%s", data);

}

fclose(file);

} //запис у файл

char\* readFromFile (const char\* path) {

FILE \*file;

char \*res;

res = (char \*) malloc(2000);

file = fopen(path, "rt");

if (file != NULL) {

fscanf(file,"%s",res);

//res = "y";

} else {

res[0] = '-';

}

return res;

} //читання з файлу

FileOperation.h

#ifndef FileOperatios\_h

#define FileOperatios\_h

#include <stdio.h>

#include <stdlib.h>

#endif /\* FileOperatios\_h \*/

chess-Bridging-Header.h

//оголошення функцій

void recordToFile (const char\* path, const char \*data);

char\* readFromFile (const char\* path);

FileOperation.swift

import Foundation

class FileOperation {

init() {}

func recordRes(path: String, res: String) {

var pathToC = path

if !path.contains(".txt") && path.contains(".") {

Errors.isError(type: 4)

return

} else if !path.contains(".txt") {

pathToC += ".txt"

}

recordToFile(pathToC, res)

} //запис результату

func recordPos (path: String, pos: String) {

var pathToC = path

if !path.contains(".chs") && path.contains(".") {

Errors.isError(type: 4)

return

} else if !path.contains(".chs") {

pathToC += ".chs"

}

recordToFile(pathToC, pos)

} //запис позиції

func readPos (path: String) -> String {

var pathToC = path

if !path.contains(".chs") && path.contains(".") {

Errors.isError(type: 4)

return ""

} else if !path.contains(".chs") {

pathToC += ".chs"

}

let pos: UnsafeMutablePointer<CChar> = readFromFile(pathToC)

let res = String(validatingUTF8: pos)

if res == "-" {

Errors.isError(type: 4)

}

pos.deallocate()

return res!

} //зчитування позиції

}