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: 63 сторінки, 22 рисунка, 3 таблиці, 7 посиланнь.

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](#_Toc106593714)

[1 Постановка задачі 6](#_Toc106593715)

[2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7](#_Toc106593716)

[3 ЗАПОВНЕННЯ СХОВИЩА ДАНИХ 8](#_Toc106593717)

[3.1. Збір даних 8](#_Toc106593718)

[3.2. Аналіз опису книги 9](#_Toc106593719)

[3.3. Занесення даних до сховища 10](#_Toc106593720)

[3.4. Отримання даних зі сховища 11](#_Toc106593721)

[4 ІНТЕЛЕКтуальний аналіз даних 12](#_Toc106593722)

[4.1. Обґрунтування вибору методів інтелектуального аналізу даних 12](#_Toc106593723)

[4.2. Теоретичне обґрунтування логістичної регресії та методу градієнтного спуску 12](#_Toc106593724)

[4.3. Теоретичне обґрунтування дерева прийняття рішень 14](#_Toc106593725)

[4.4. Підготовка даних до інтелектуального аналізу 14](#_Toc106593726)

[4.5. Логістична регресія 24](#_Toc106593727)

[4.6. Дерево рішень 24](#_Toc106593728)

[4.7. Тестування методів 24](#_Toc106593729)

[4.8. Порівняння методів 27](#_Toc106593730)

[Висновки 28](#_Toc106593731)

[Перелік посилань 29](#_Toc106593732)

[Додаток А Тексти програмного коду 63](#_Toc106593733)

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, мультиколінеарність та вирівняти дані за допомогою SMOTE.

Реалізувати навчання методів логістичної регресії за допомогою градієнтного спуску та дерева прийняття рішень. Та протестувати;

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. [1]

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.

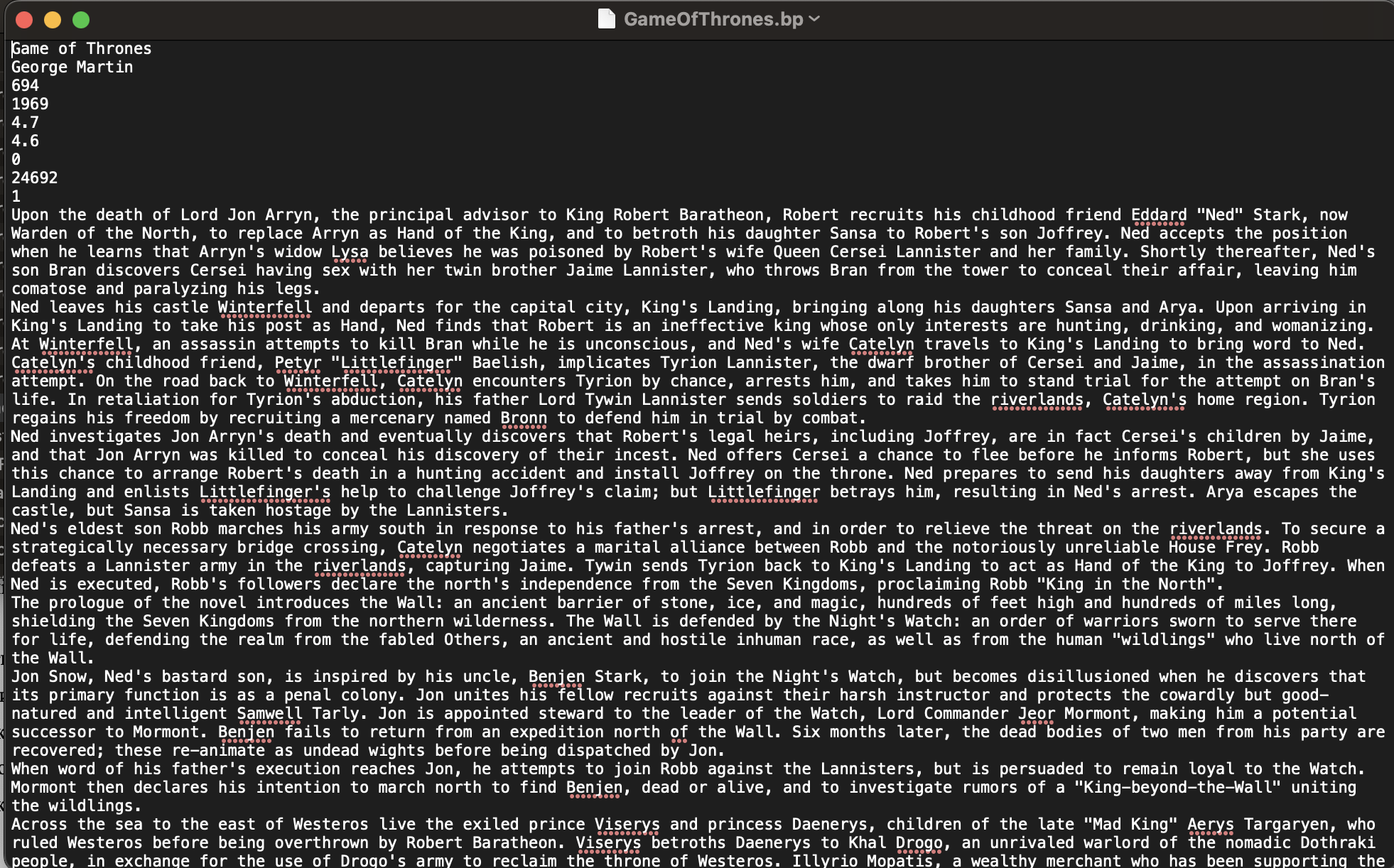


Рисунок 3.1 – Приклад заповненого файлу

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

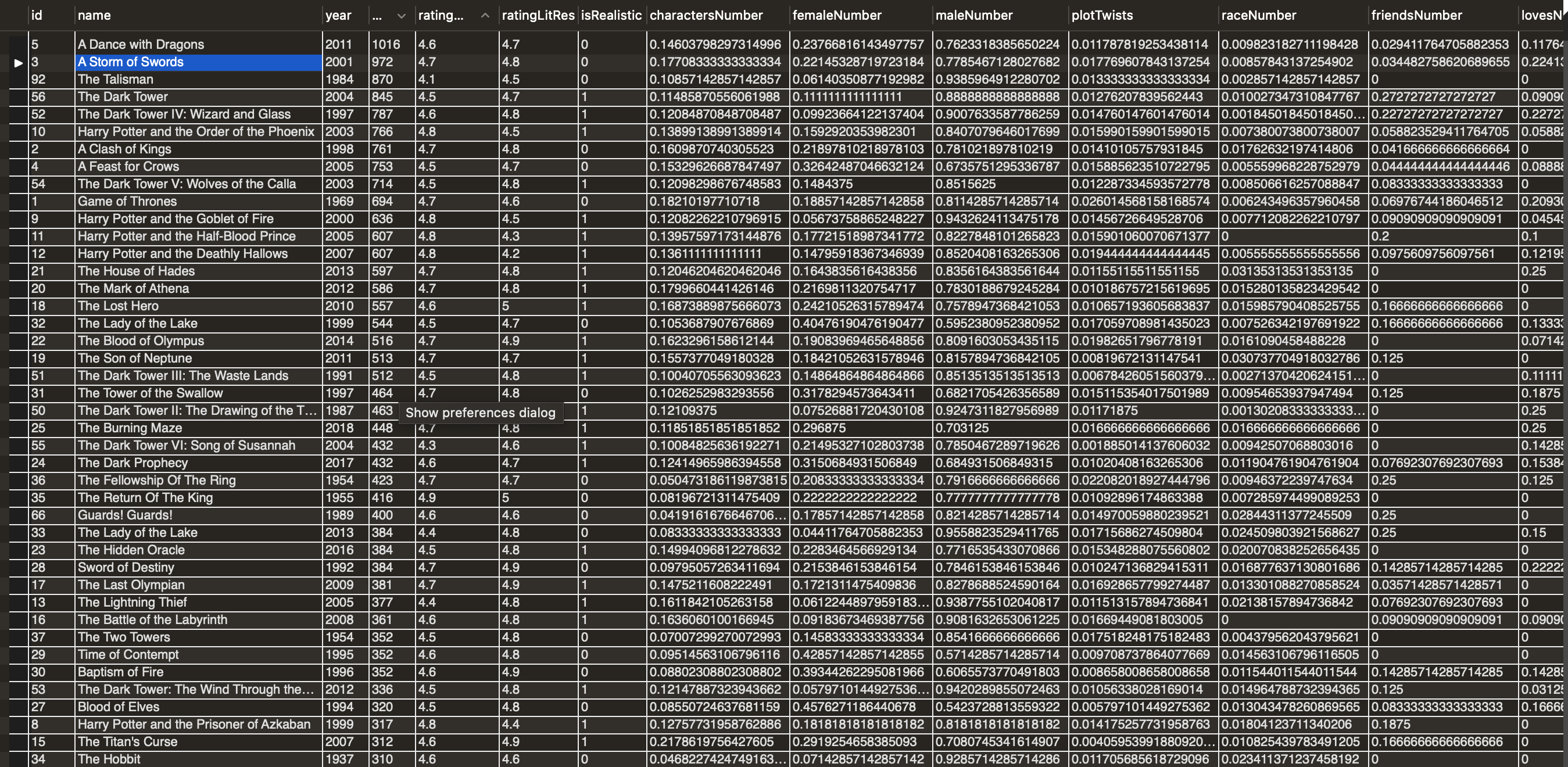


Рисунок 3.2-Скріншот заповненого сховища даних

## 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

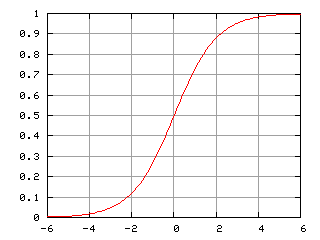


Рисунок 4.1 – Графік сигмоїди

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. [2]

* + - * 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 [3][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. [5]

## 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.

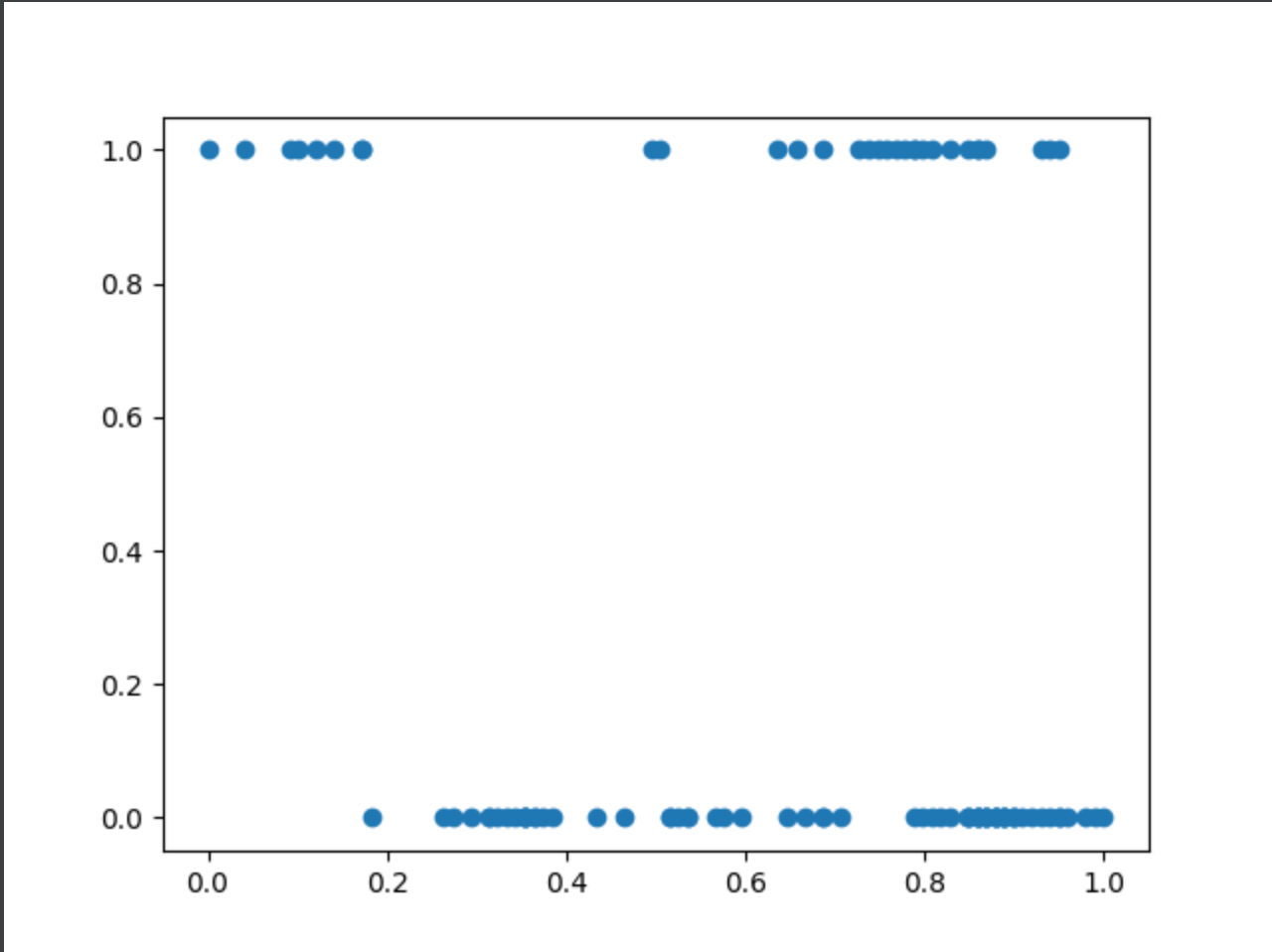


Рисунок 4.2 - Графік залежності жанру фентезі від року виходу

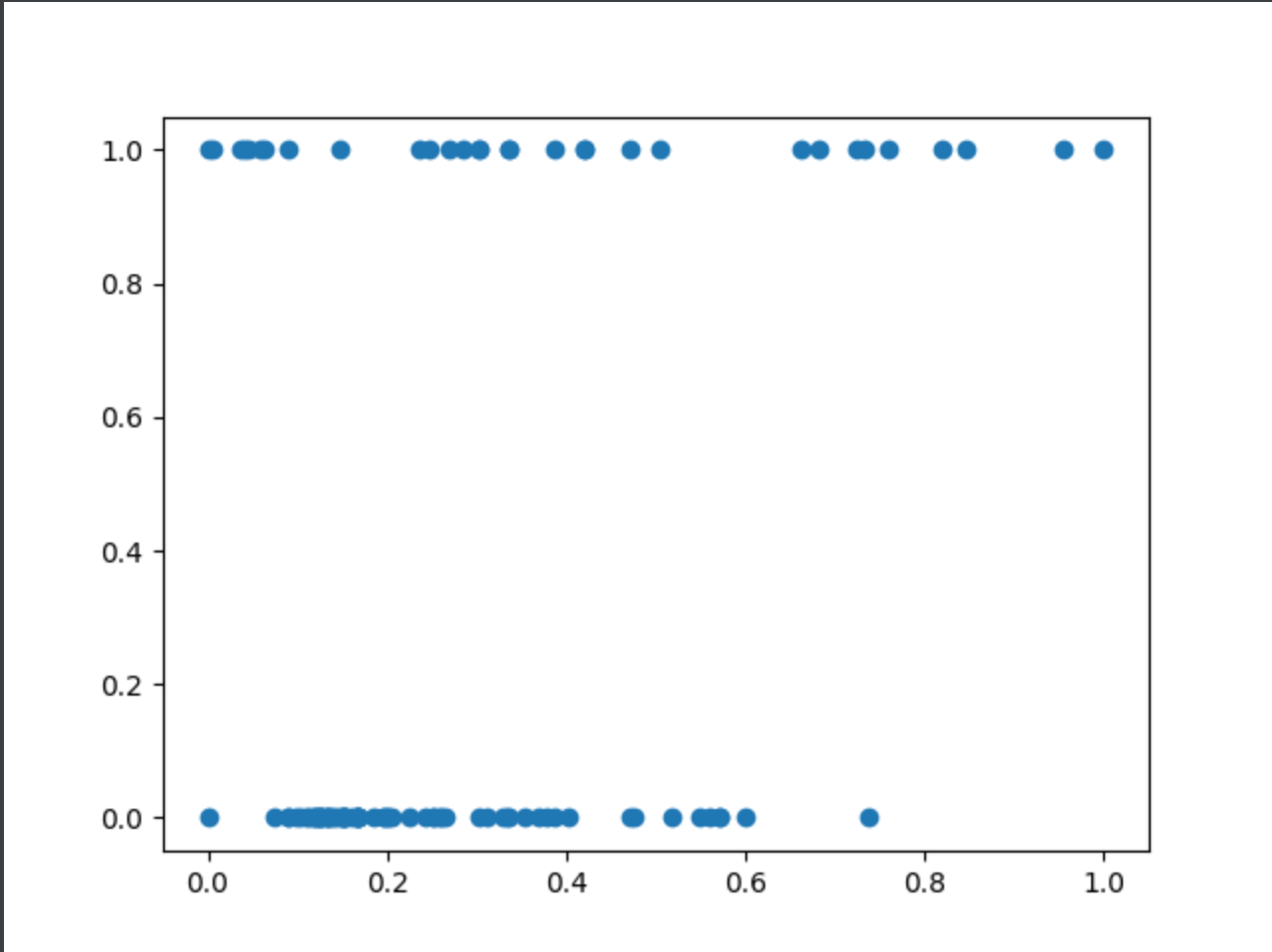


Рисунок 4.3 - Графік залежності жанру фентезі від кількості сторінок

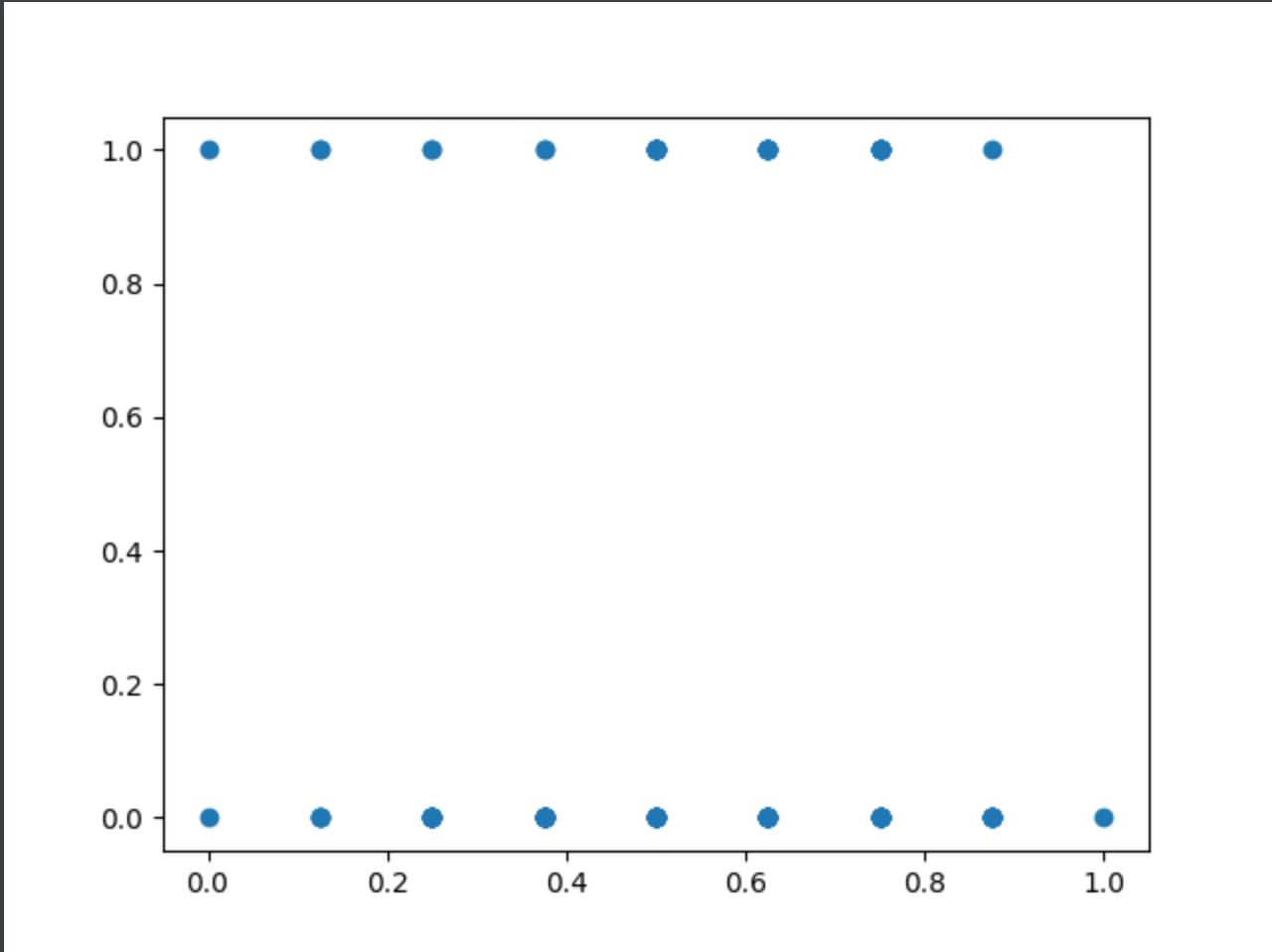


Рисунок 4.4 - Графік залежності жанру фентезі від рейтингу на LiveLib

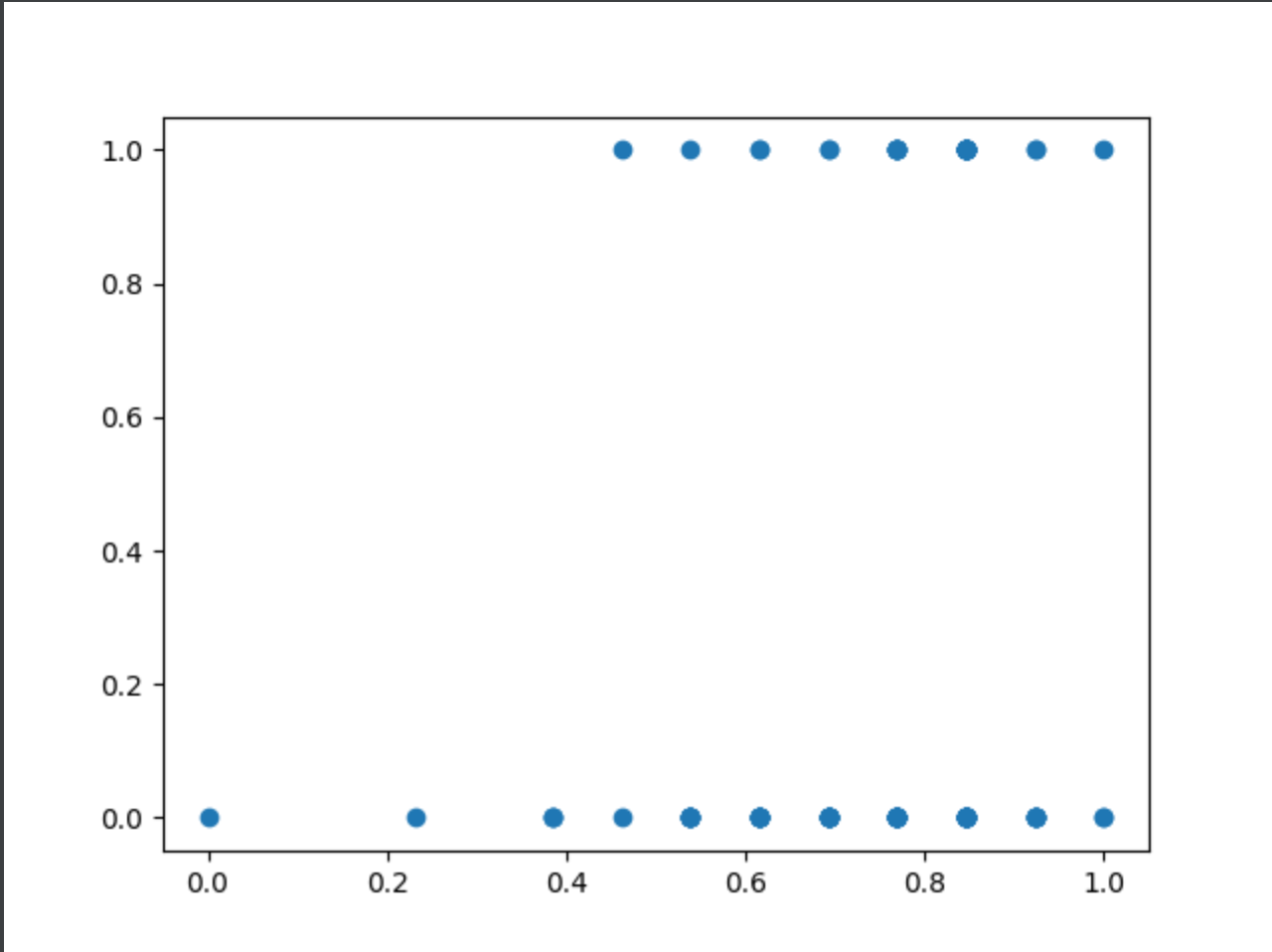


Рисунок 4.5 - Графік залежності жанру фентезі від рейтингу на LitRes

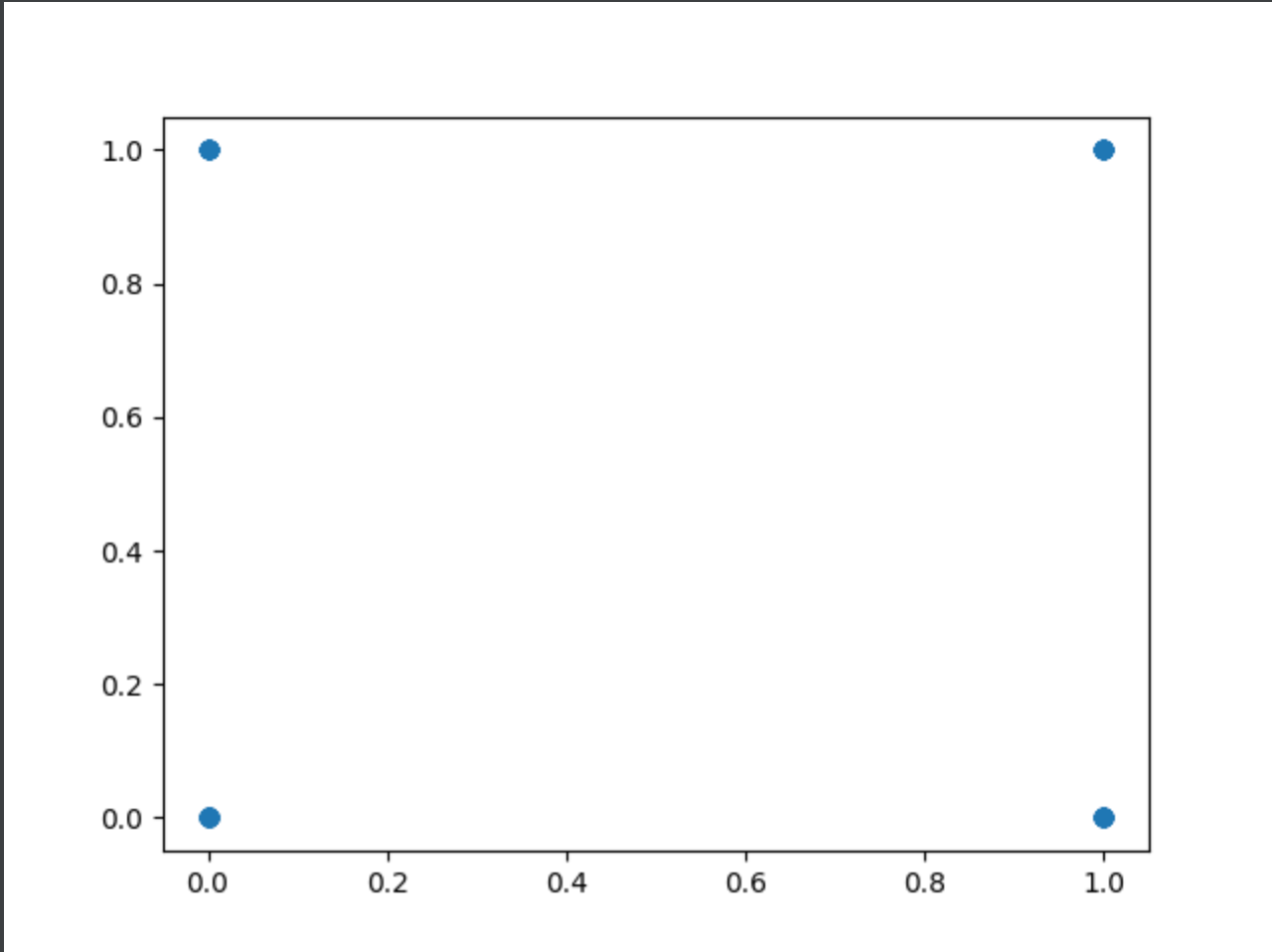


Рисунок 4.6 – Графік залежності жанру фентезі від реалістичності світу

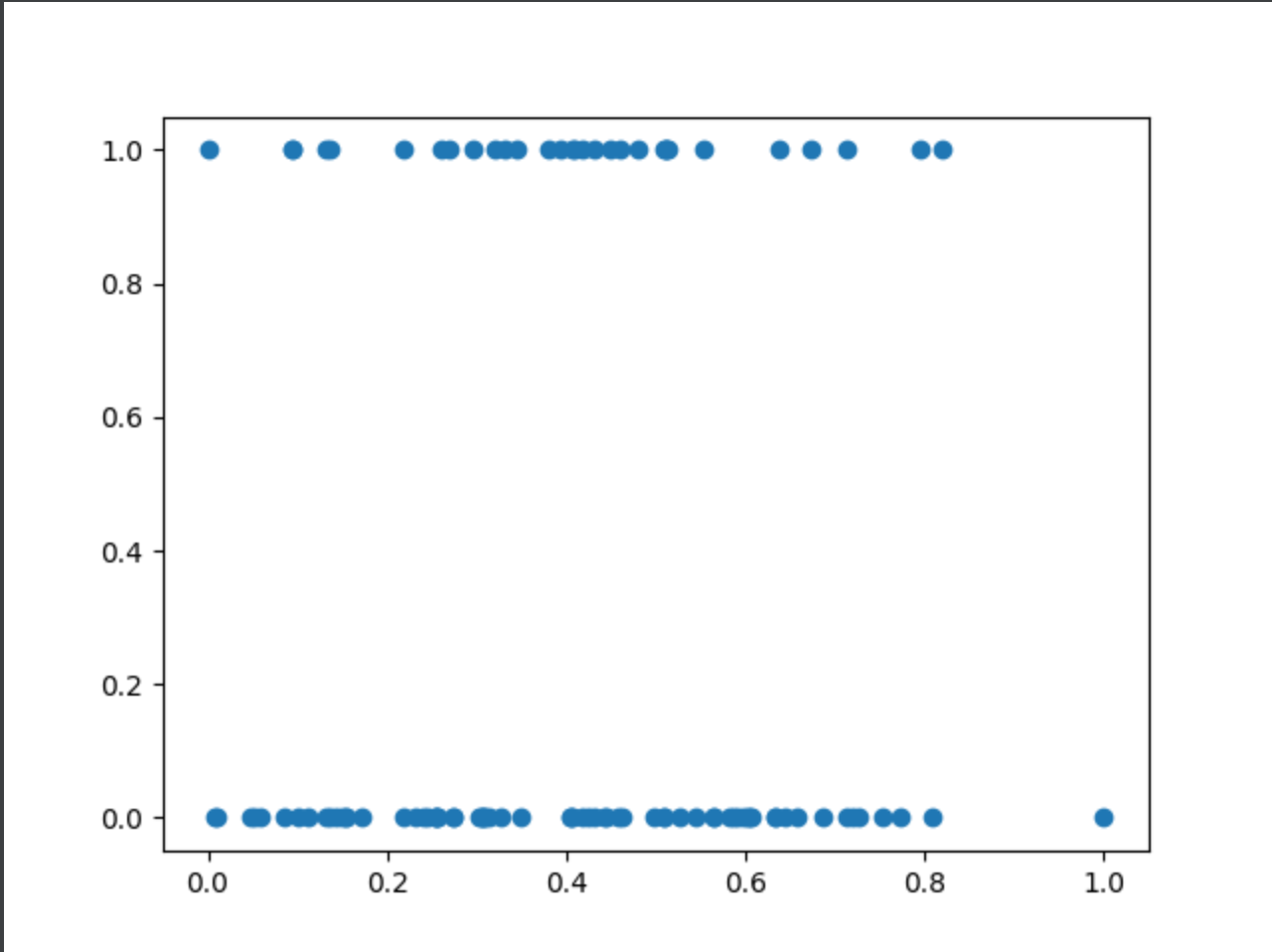


Рисунок 4.7 – Графік залежності жанру фентезі від кількості персонажів

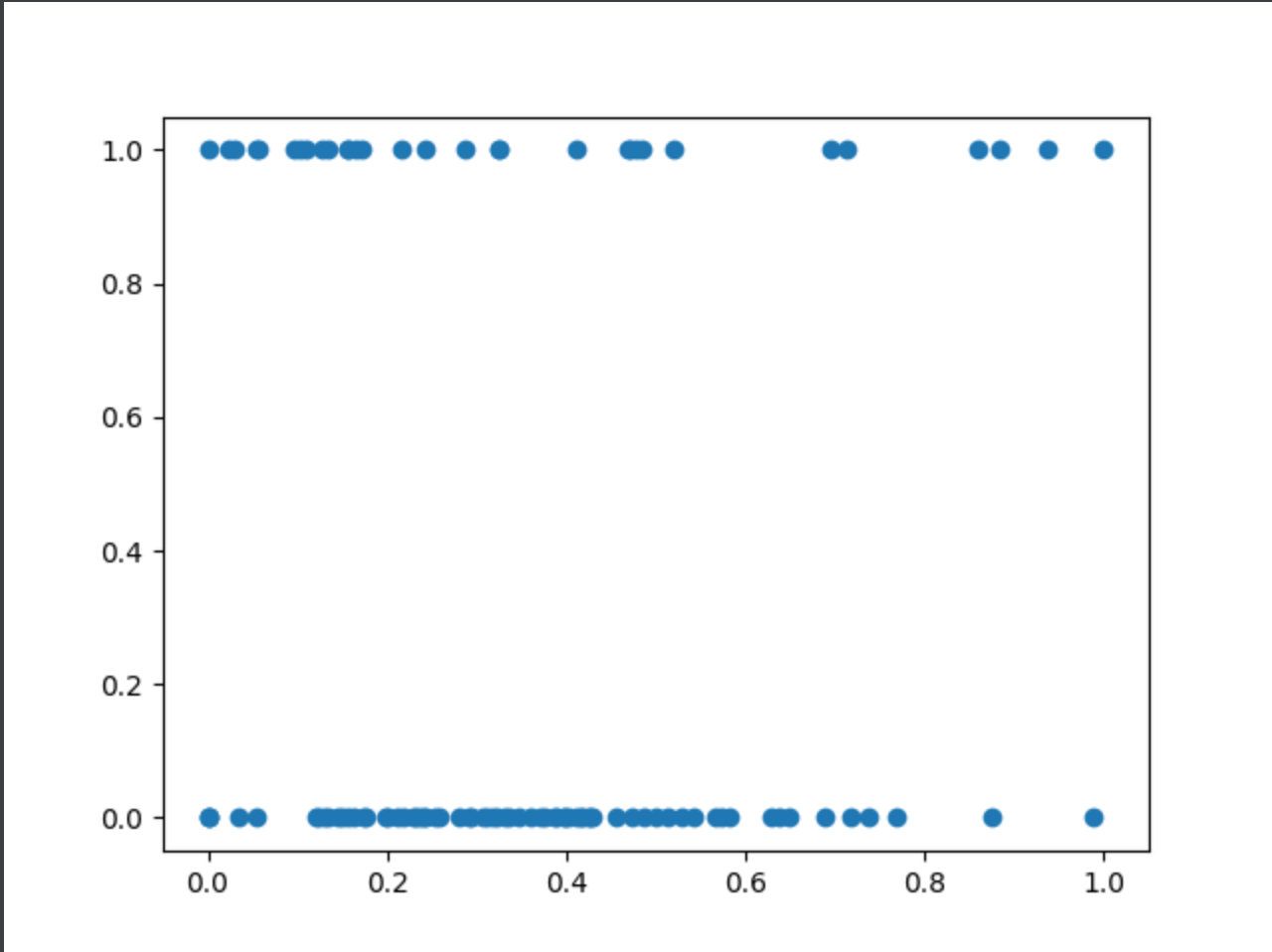


Рисунок 4.8 – Графік залежності жанру фентезі від відсотку жінок

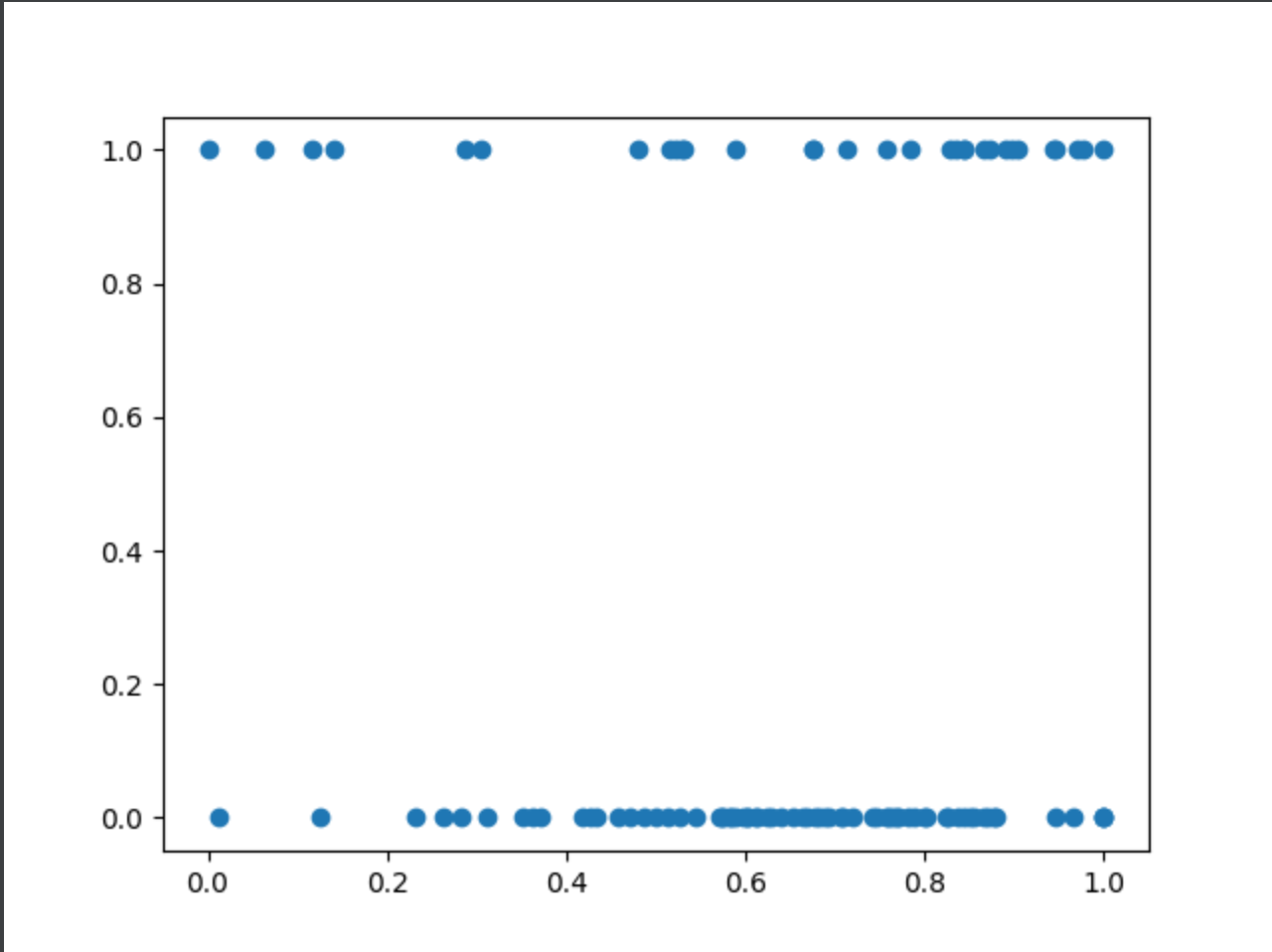


Рисунок 4.9 – Графік залежності жанру фентезі від відсотку чоловіків

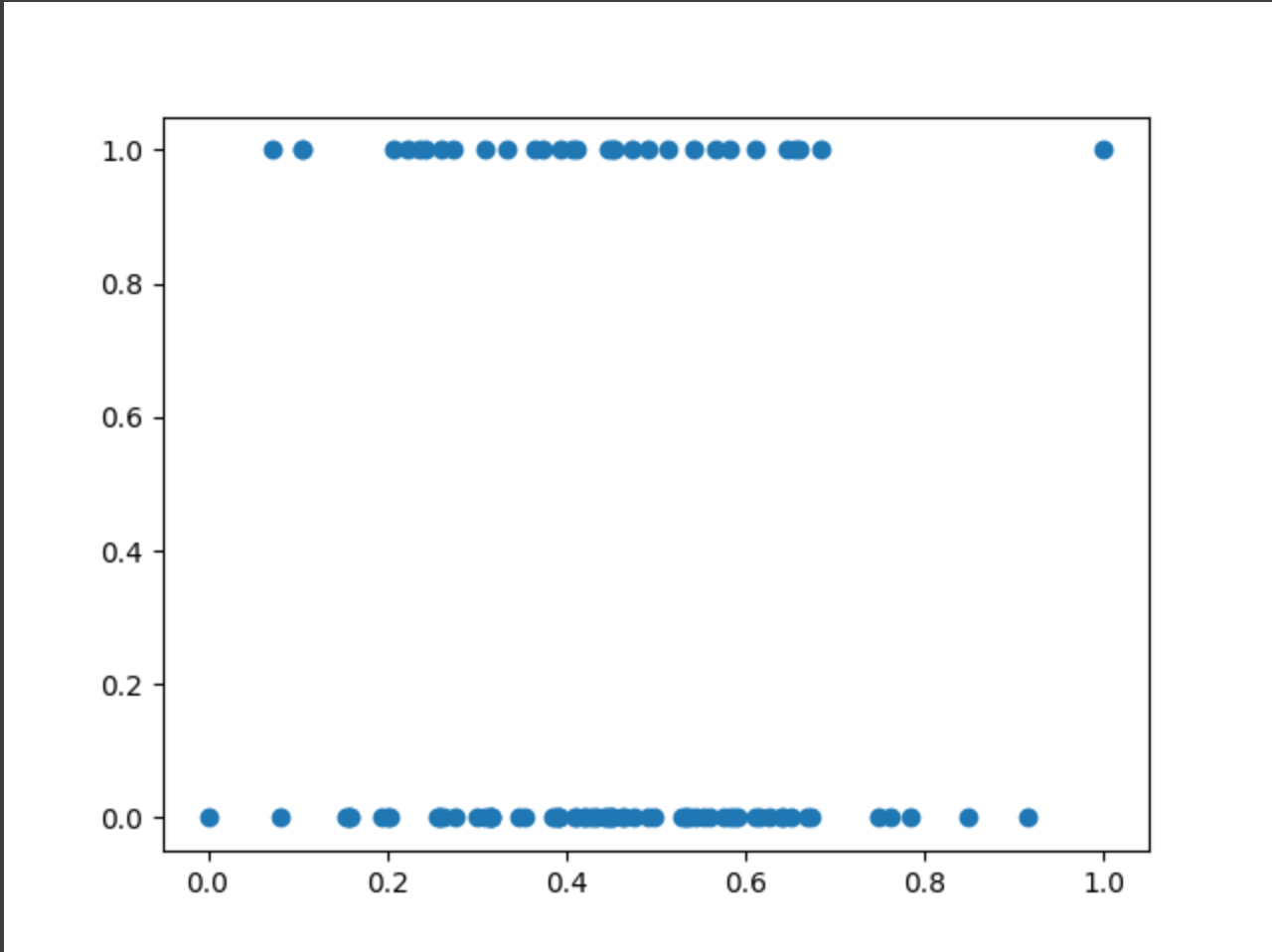


Рисунок 4.10 – Графік залежності жанру фентезі від кількості сюжетних поворотів

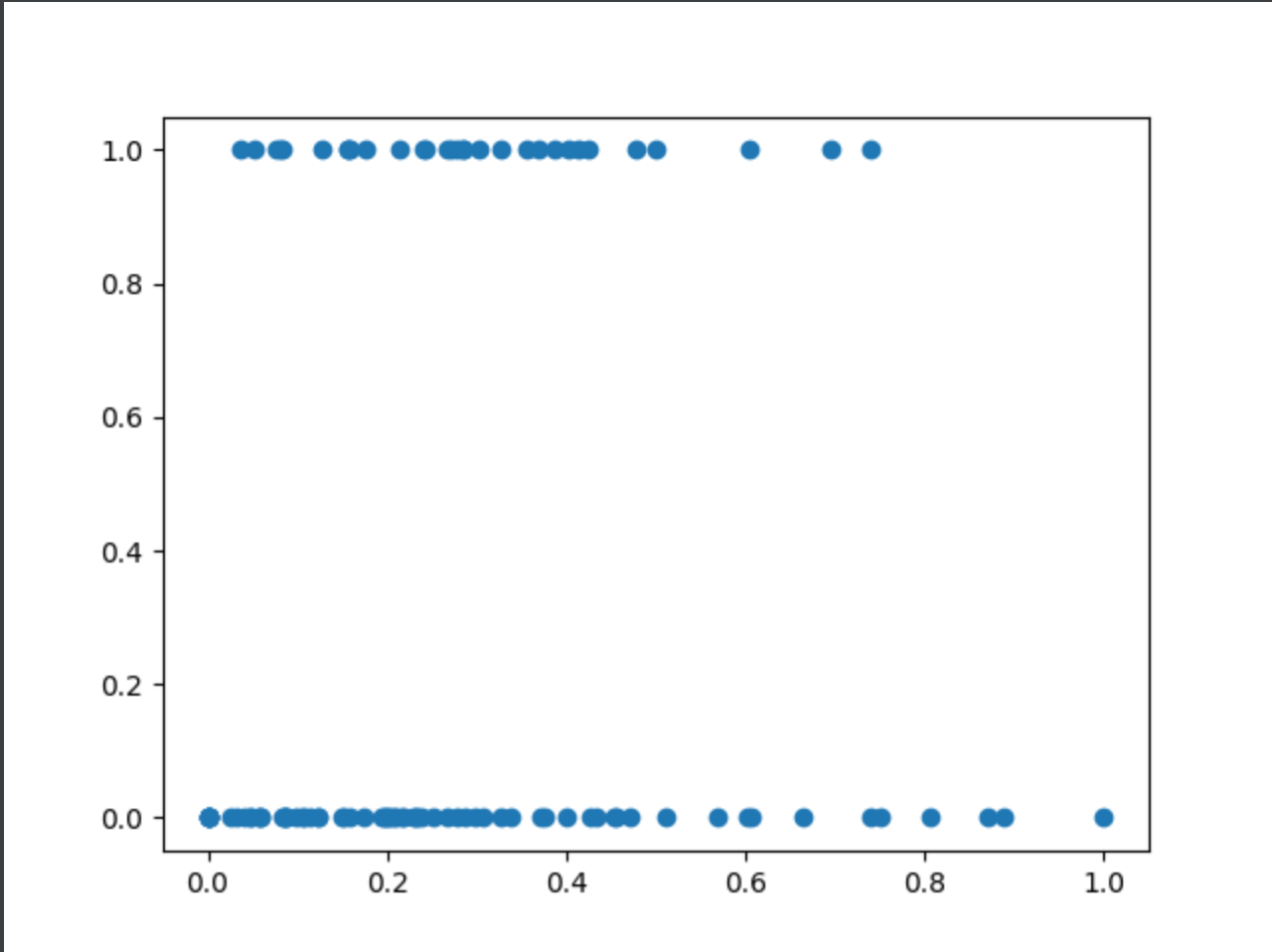


Рисунок 4.11 – Графік залежності жанру фентезі від кількості рас

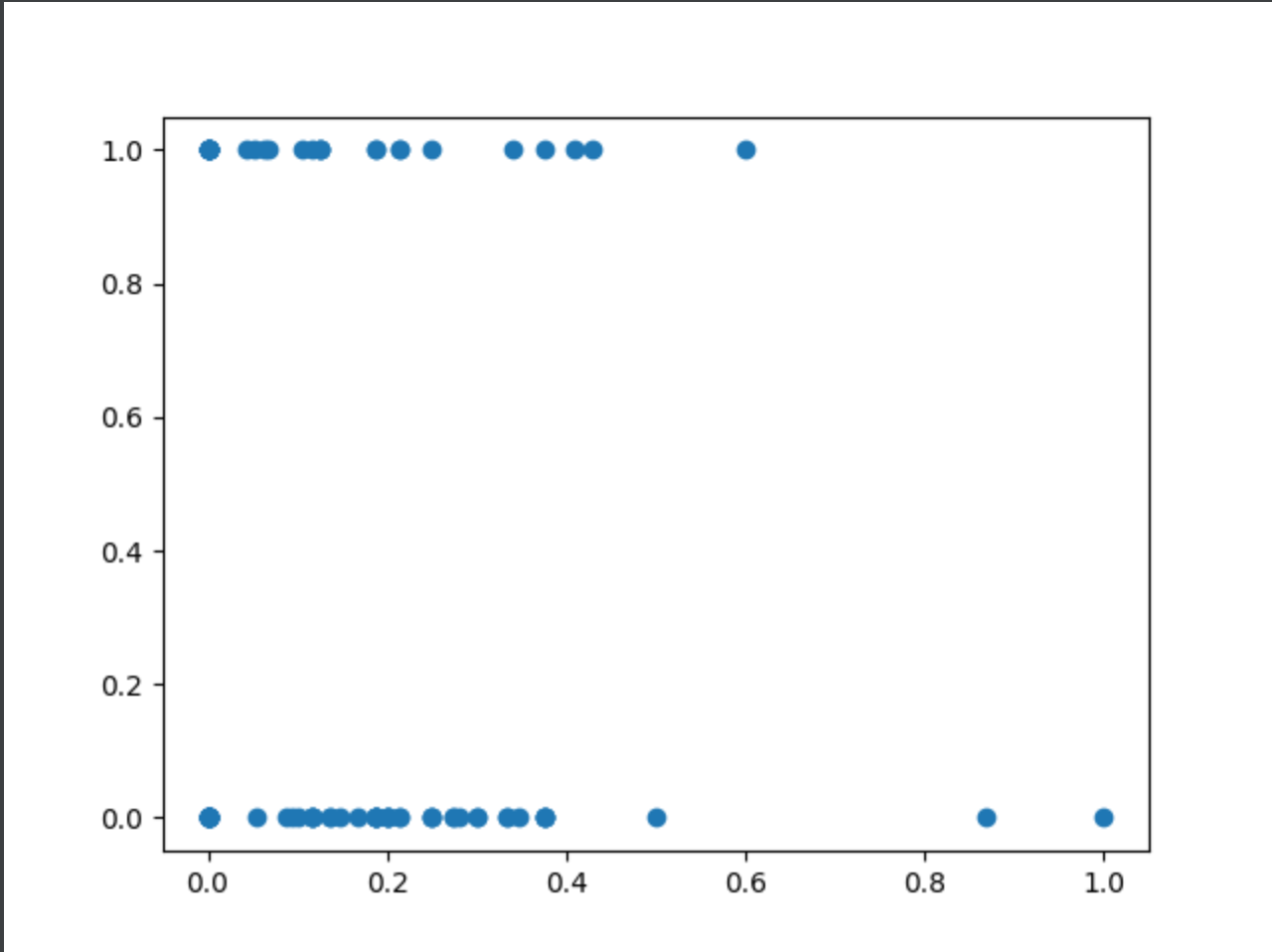


Рисунок 4.12 – Графік залежності жанру фентезі від проценту дружніх стосунків

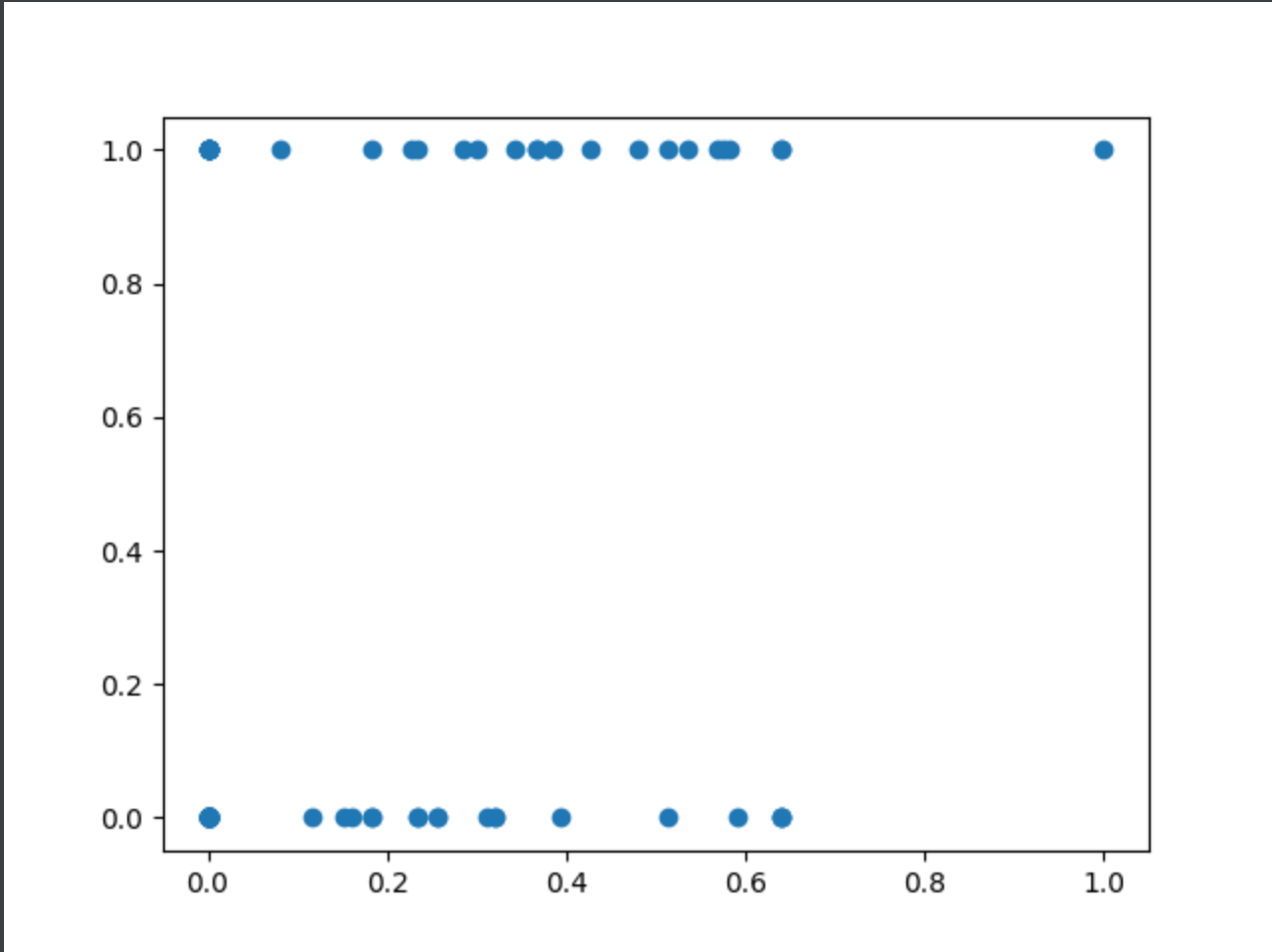


Рисунок 4.13 – Графік залежності жанру фентезі від проценту любовних стосунків

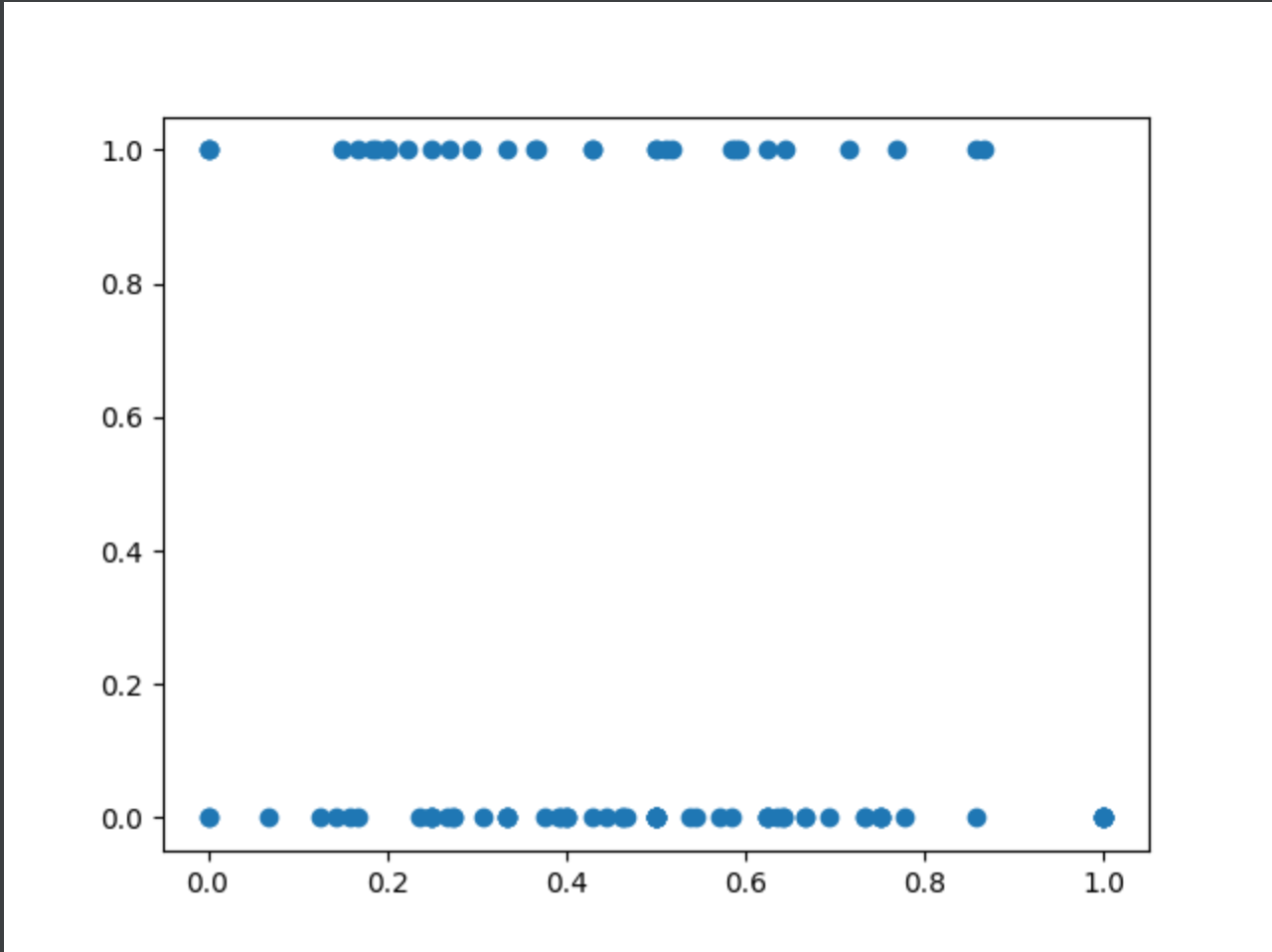


Рисунок 4.14 – Графік залежності жанру фентезі від проценту родинних стосунків

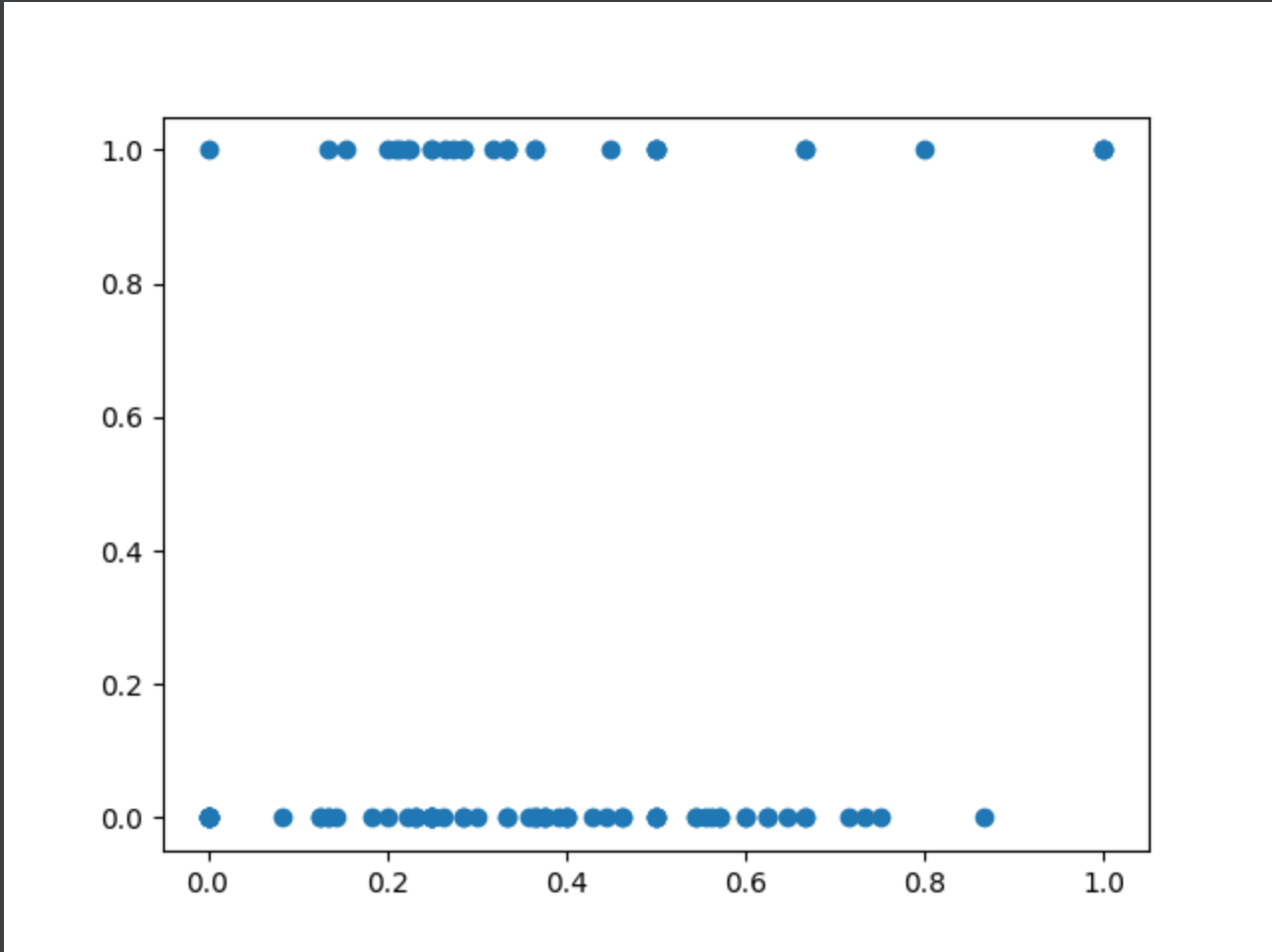


Рисунок 4.15 – Графік залежності жанру фентезі від проценту ворожих стосунків

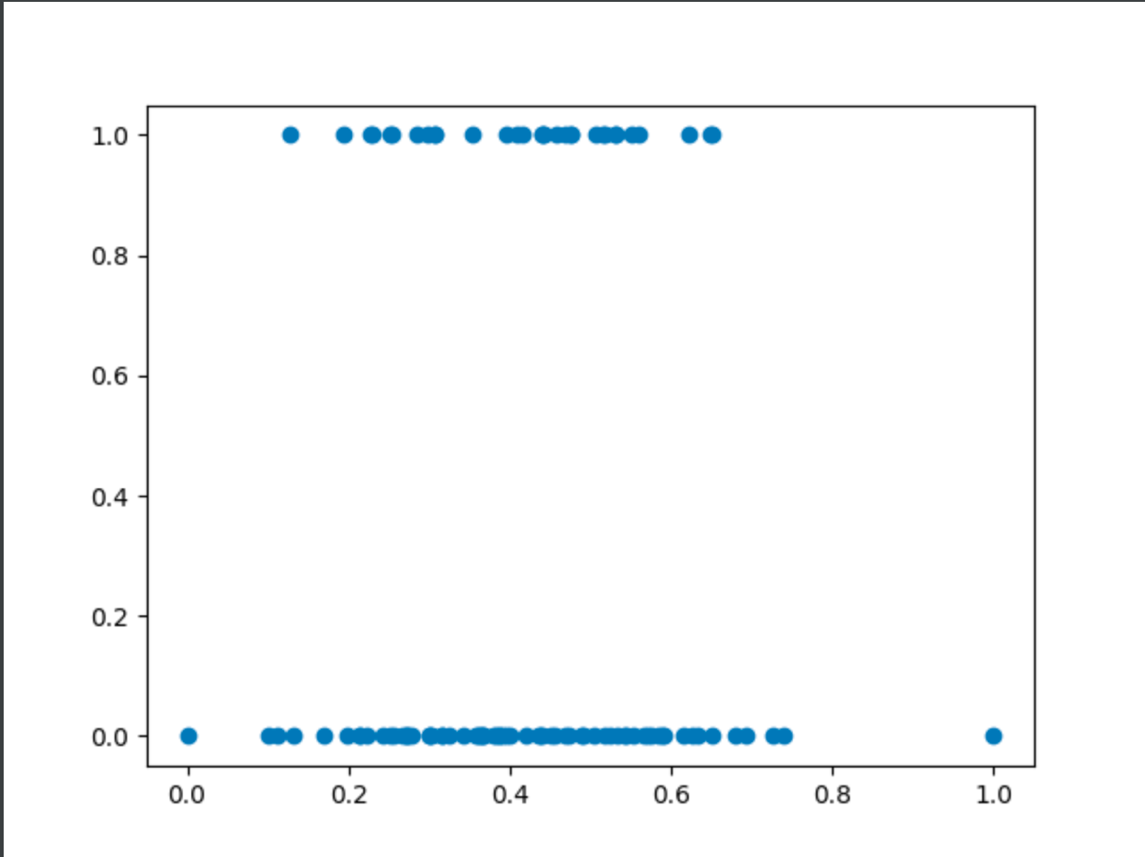


Рисунок 4.16 – Графік залежності жанру фентезі від кількості локацій

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) [6]

* + - * 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.

## Логістична регресія

Приступимо до регресійного аналізу. Параметри логістичної регресії знаходяться за формулою, представленою у 2му пункті. Код представлено у додатку А, файл LogisticRegresion.py. В результаті для функції z отримуємо наступні параметри. (Формула 4.6)

* + - * 1. Отримана функція

## Дерево рішень

Код для побудови дерева рішень представлено у додатку А, файли DecisionTree.py, DecisionCell.py. Отримане дерево рішень зображене на рисунку 4.17. У хорошій якості доступно за посиланням: https://drive.google.com/file/d/1JXbIuxBhItHYHYBsJJY\_tlFypeAh-JBU/view?usp=sharing

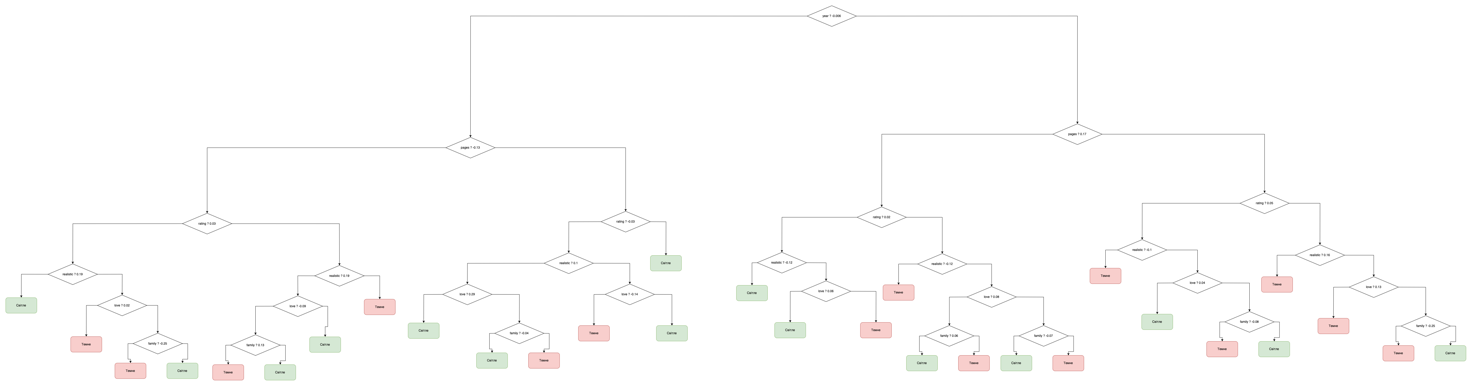


Рисунок 4.17 - Загальний план дерева рішень

## Тестування методів

Для тестування алгоритмів було відібрано 25 книг, що не брали участь у навчанні. На рисунку 4.18 та 4.19 зображено графік результатів рівняння логістичної регресії.

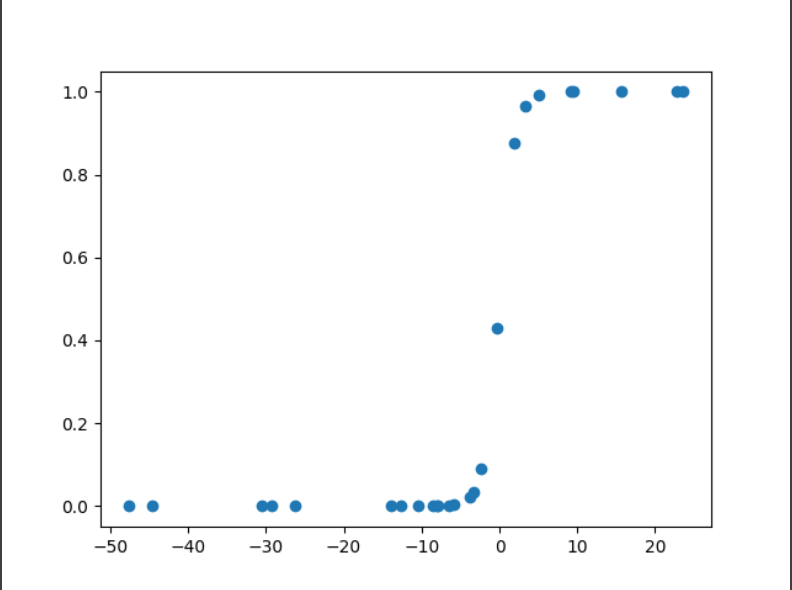


Рисунок 4.18 – Графік залежності прогнозованого y від z

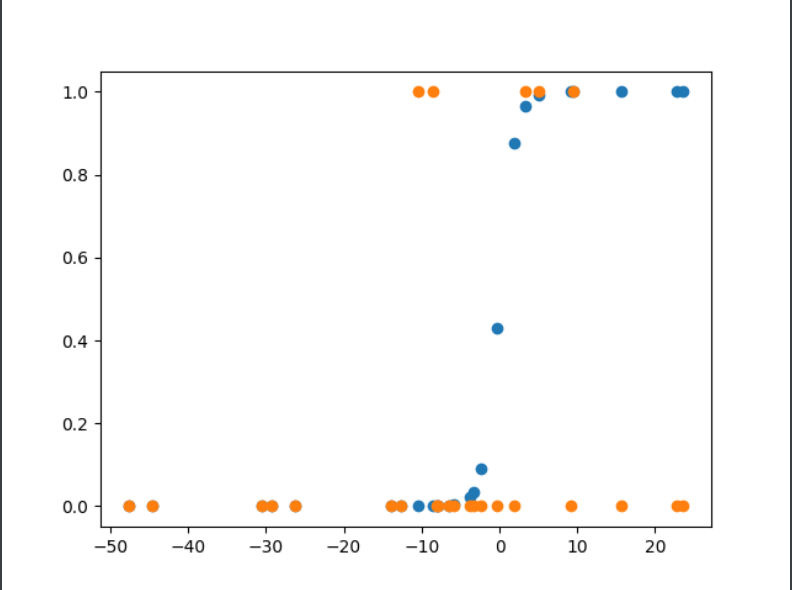


Рисунок 4.19 - Графік залежності прогнозованого y від z, з накладанням реального y

Проведемо аналіз залишків для логістичної регресії. Гістограма залишків зображена на рисунку 4.20

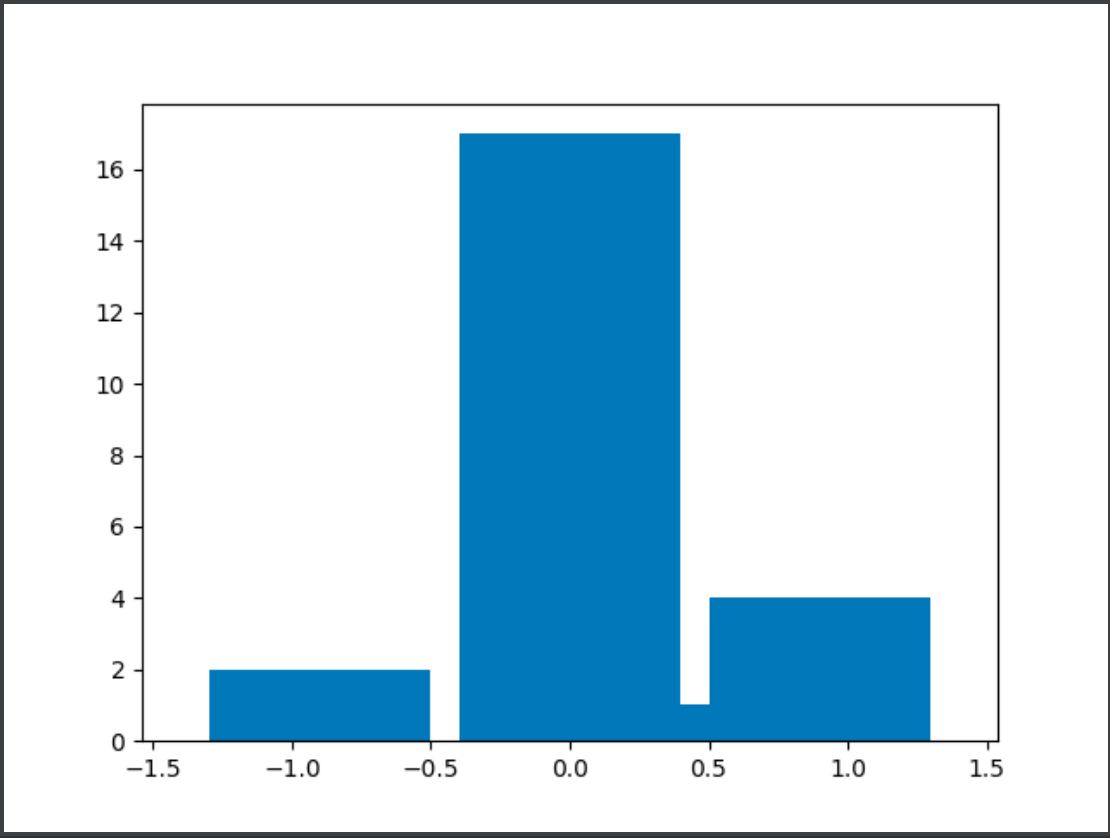


Рисунок 4.20 – Діаграма залишків

Як видно з діаграми – розподіл залишків – нормальний.

Знайдемо коефіцієнт детермінації та точність передбачення. Застосуємо формули 4.7 – 4.9 [7]

* + - * 1. Формула знаходження коефіцієнту детермінації
        2. Формула знаходження RSS
        3. Формула знаходження TSS

Результати запишемо до таблиці 4.2

Результати аналізу результатів

|  |  |  |
| --- | --- | --- |
|  | Логістична регресія | Дерево рішень |
| RSS | 4 | 4 |
| TSS | 12 | 6.96 |
| R2 | 0.66 | 0.42 |
| Точність  (к-сть правильних/  к-сть усіх) | 0.72 | 0.52 |

## Порівняння методів

Таким чином, протестувавши обидва методи, можемо зазначити, що метод логістичної регресії є точнішим, має більший коефіцієнт детермінації, а отже є кращим. Але, варто зазначити, що метод дерева рішень навчається набагато швидше, отже також має право на існування.

Висновки

В результаті виконання курсової роботи було оброблено та проаналізовано короткі описи книг, завантажено дані до сховища даних. Для реалізації даних задач було використано MySQL та мову Python3.

На основі проведеного аналізу предметної області інтелектуального аналізу даних для прогнозування піджанру фентезі книг було реалізовано 2 методи інтелектуального аналізу: логістична регресія за допомогою градієнтного спуску та дерево прийняття рішень. Перед реалізацію методів з вибірки було відсортовано незначущі та мультиколінеарні параметри.

Було протестовано обидва методи. В результаті тестування виявилося, що метод логістичного аналізу працює краще ніж метод дерева прийняття рішень.

Отже, поставлені задачі були виконані, а також планується розширення функціоналу, за рахунок збільшення кількості даних та кількості порівнюваних алгоритмів.

Перелік посилань

1. Темне фентезі [Електронний ресурс] Доступ за посиланням:

https://uk.wikipedia.org/wiki/Темне\_фентезі

1. Логістична регресія [Електронний ресурс] Доступ за посиланням:

https://uk.wikipedia.org/wiki/Логістична\_регресія

1. Градієнтний спуск [Електронний ресурс] Доступ за посиланням:

https://uk.wikipedia.org/wiki/Градієнтний\_спуск

1. Градієнтний спуск [Електронний ресурс] Доступ за посиланням:

https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

1. Дерево ухвалення рішень [Електронний ресурс] Доступ за посиланням:

https://uk.wikipedia.org/wiki/Дерево\_ухвалення\_рішень

1. Коефіцієнт кореляції Пірсона [Електронний ресурс] Доступ за посиланням:

https://uk.wikipedia.org/wiki/Коефіцієнт\_кореляції\_Пірсона

1. Коефіцієнт детермінації [Електронний ресурс] Доступ за посиланням:

https://uk.wikipedia.org/wiki/Коефіцієнт\_детермінації

Додаток А Тексти програмного коду

*Тексти програмного коду програмного забезпечення Прогнозування успішності фентезі-літератури*

(Найменування програми)

*CD-RW*

(Вид носія даних)

*63 арк., 41 мб*

(Обсяг програми)

Студента групи ІП-01 2 курсу

Храмченко А.С.

Book.py

from TextWorker import TextWorker

class Book:

name = ""

author = ""

pages = 0

year = 0

ratingLiveLib = 0.0

ratingLitRes = 0.0

isReal = 0

count = 0

isDark = 0

text = ""

analyzeResult = None

def \_\_init\_\_(self, data):

self.name = data[0]

print(f'Book {self.name} is on work')

self.author = data[1]

self.pages = int(data[2])

self.year = int(data[3])

self.ratingLiveLib = float(data[4])

self.ratingLitRes = float(data[5])

self.isReal = int(data[6])

self.count = int(data[7])

self.isDark = int(data[8])

otherDataCount = len(data)-9

for i in range(otherDataCount):

self.text += data[i+9]

self.\_\_analyzeData()

print(f'Book {self.name} is finished')

def \_\_analyzeData(self):

text = TextWorker(self.text)

self.analyzeResult = text.getData()

def getName(self):

newName = ""

for char in self.name:

if char == "'":

newName += "\\"

newName += char

return newname

DataWorker.py

import sys

from FileWorker import MaxMinFile

class DataWorker:

maxValues = [0]\*17

minValues = [sys.maxsize]\*17

def updateData(self, book):

data = [

book.year,

book.pages,

book.ratingLiveLib,

book.ratingLitRes,

book.isReal,

book.analyzeResult['count'],

book.analyzeResult['woman\_count'],

book.analyzeResult['man\_count'],

book.analyzeResult['twist\_count'],

book.analyzeResult['race\_count'],

book.analyzeResult['friends\_count'],

book.analyzeResult['lovers\_count'],

book.analyzeResult['families\_count'],

book.analyzeResult['enemies\_count'],

book.analyzeResult['location\_count'],

book.isDark,

book.count

]

for i in range(len(data)):

if data[i] > self.maxValues[i]:

self.maxValues[i] = data[i]

if data[i] < self.minValues[i]:

self.minValues[i] = data[i]

def loadData(self, dbData):

result = []

file = MaxMinFile("solution/MaxMin.bp")

self.maxValues = file.getMaxArray()

self.minValues = file.getMinArray()

for data in dbData:

array = [

data['year'],

data['pages'],

data['ratingLiveLib'],

data['ratingLitRes'],

data['isRealistic'],

data['charactersNumber'],

data['femaleNumber'],

data['maleNumber'],

data['plotTwists'],

data['raceNumber'],

data['friendsNumber'],

data['lovesNumber'],

data['relativesNumber'],

data['enemiesNumber'],

data['locationsNumber'],

data['isDark'],

data['countOfRate']

]

subResult = []

for i in range(len(array)):

subResult.append((array[i] - self.minValues[i])/(self.maxValues[i] - self.minValues[i]))

result.append(subResult)

return result

def saveMaxMinData(self):

file = MaxMinFile("solution/MaxMin.bp","w")

file.writeToFile(self.maxValues)

file.writeToFile(self.minValues)

DBWorker.py

import pymysql

class DBWorker:

\_\_host = ""

\_\_user = ""

\_\_port = 3306

\_\_password = ""

\_\_db\_name = ""

\_\_connection = None

def \_\_init\_\_(self, host="localhost", user="root", port="3306", password="password", db\_name="BooksMeasurement"):

self.\_\_host = host

self.\_\_user = user

self.\_\_port = int(port)

self.\_\_password = password

self.\_\_db\_name = db\_name

def connect(self):

try:

self.\_\_connection = pymysql.connect(

host=self.\_\_host,

port=self.\_\_port,

user=self.\_\_user,

password=self.\_\_password,

database=self.\_\_db\_name,

cursorclass=pymysql.cursors.DictCursor

)

print("Successfully connected!")

except Exception as ex:

print("Connection refused...")

print(ex)

def create\_table(self):

self.connect()

try:

with self.\_\_connection.cursor() as cursor:

drop\_script = "DROP TABLE IF EXISTS Parameters;"

create\_script = "CREATE TABLE Parameters ( `id` DOUBLE NOT NULL AUTO\_INCREMENT, " \

"`name` VARCHAR(150) NOT NULL, `year` DOUBLE NOT NULL, `pages` DOUBLE NOT NULL, " \

"`ratingLiveLib` DOUBLE NOT NULL, `ratingLitRes` DOUBLE NOT NULL, `isRealistic` TINYINT NOT NULL, `charactersNumber` " \

"DOUBLE NOT NULL, `femaleNumber` DOUBLE NOT NULL, `maleNumber` DOUBLE NOT NULL, " \

"`plotTwists` DOUBLE NOT NULL, `raceNumber` DOUBLE NOT NULL, `friendsNumber` " \

"DOUBLE " \

"NOT NULL, `lovesNumber` DOUBLE NOT NULL, `relativesNumber` DOUBLE NOT NULL, " \

"`enemiesNumber` DOUBLE NOT NULL, `locationsNumber` DOUBLE NOT NULL, `countOfRate` INT NOT NULL, `isDark` TINYINT, PRIMARY KEY (" \

"`id`)) "

cursor.execute(drop\_script)

cursor.execute(create\_script)

finally:

self.\_\_connection.close()

def insert\_params(self, name, year, pages, ratingLiveLib, ratingLitRes, is\_realistic, characters\_number, female\_number, male\_number,

plot\_twists, race\_number, friends\_number, loves\_number, relatives\_number, enemies\_number,

locations\_number, rate\_number, isDark):

self.connect()

try:

with self.\_\_connection.cursor() as cursor:

insert\_script = "INSERT INTO `Parameters`(`name`, `year`, `pages`, `ratingLiveLib`, `ratingLitRes`, `isRealistic`, `charactersNumber`, " \

"`femaleNumber`, `maleNumber`, `plotTwists`, `raceNumber`, `friendsNumber`, `lovesNumber`, " \

"`relativesNumber`, `enemiesNumber`, `locationsNumber`, `countOfRate`, `isDark`) VALUES(" + "'{}', {}, {}, {}, {}, {}, " \

"{}, {}, {}, {}, {}, {}," \

" {}, {}, {}, {}, {}, {});".format(

name, year, pages, ratingLiveLib, ratingLitRes, is\_realistic, characters\_number, female\_number, male\_number,

plot\_twists, race\_number, friends\_number, loves\_number, relatives\_number, enemies\_number,

locations\_number, rate\_number, isDark)

cursor.execute(insert\_script)

self.\_\_connection.commit()

finally:

self.\_\_connection.close()

def fetch\_all\_data(self):

self.connect()

try:

with self.\_\_connection.cursor() as cursor:

select\_script = "SELECT \* FROM Parameters;"

cursor.execute(select\_script)

rows = cursor.fetchall()

return rows

finally:

self.\_\_connection.close()

DecisionCell.py

from FileWorker import CellFile

class DecisionCell:

meaningfulParameters = [0, 1, 3, 4, 11, 12]

yColumnIndex = 15

parametrIndex = 0

avarageParam = 0

values = []

leftCell = None

rightCell = None

isLast = False

prediction = None

def \_\_init\_\_(self,data,index):

if index == None or len(data) == 1:

self.isLast = True

avarage = 0

for example in data:

avarage += example[self.yColumnIndex]

if avarage>0.5:

self.prediction = 1

else:

self.prediction = 0

else:

self.parametrIndex = index

avarage = 0

for example in data:

avarage += example[index]

avarage /= len(data)

self.avarageParam = avarage

rightData = []

leftData = []

for example in data:

if example[index] > avarage:

rightData.append(example)

else:

leftData.append(example)

if len(leftData) == 0:

leftData = rightData

if len(rightData) == 0:

rightData = leftData

indexInParams = self.meaningfulParameters.index(index)

newIndex = None

if indexInParams < len(self.meaningfulParameters)-1:

newIndex = self.meaningfulParameters[indexInParams+1]

self.leftCell = DecisionCell(leftData,newIndex)

self.rightCell = DecisionCell(rightData,newIndex)

def saveData(self, name):

file = CellFile("solution/tree/"+name+".bp","w")

if self.isLast:

file.writeLastCell(self.prediction)

return

else:

file.writeCell(self.parametrIndex,self.avarageParam)

newName = name

if newName == "root":

newName = ""

self.leftCell.saveData(newName+"l")

self.rightCell.saveData(newName+"r")

DecisionTree.py

from DecisionCell import DecisionCell

import random

class DecisionTree:

root = None

data = []

yColumnIndex = 15

meaningfulParameters = [0, 1, 3, 4, 11, 12]

percentOfDark = 0.31

def \_\_init\_\_(self, data):

print("--------------------------------------------")

print("Decision tree start")

self.data = data

self.overSamplingData()

self.root = DecisionCell(self.data,0)

def overSamplingData(self):

#вирівнюємо вибірку

countOfNewData = 0

neededNewCount = int(len(self.data)\*(1-2\*self.percentOfDark))

for example in self.data:

if example[self.yColumnIndex] == 1:

newExample = example

for paramIndex in self.meaningfulParameters:

newExample[paramIndex] \*= random.uniform(0.85,1.15)

self.data.append(newExample)

countOfNewData += 1

if countOfNewData >= neededNewCount:

break

print("Data is ready to analyze")

def saveData(self):

print("Saving...")

self.root.saveData("root")

LogisticRegresion.py

import numpy as np

import matplotlib.pyplot as plt

import random

import math

from FileWorker import ParamFile

class LogisticRegresion:

data = None

names = [

"year","pages",

"ratingLiveLib","ratingLitRes",

"isRealistic",

"characters","female","male",

"twists","race",

"friends","loves","family","enemy",

"locations",

"isDark",

"countOfRate"

]

yColumnIndex = 15

meaningfulParameters = []

alpha = 0.005

parametrs = []

oldParametrs = []

percentOfDark = 0

isFinish = False

def \_\_init\_\_(self, data):

self.data = data

print("--------------------------------------------")

print("Logistic regresion start")

def go(self):

self.checkData()

self.overSamplingData()

self.parametrs = [1]\*(len(self.meaningfulParameters)+1)

self.trainData()

print(self.parametrs)

print("Training is finish")

def checkMulticol(self):

avaragesValues = [0]\*len(self.meaningfulParameters)

for example in self.data:

for i in range(len(self.meaningfulParameters)):

avaragesValues[i] += example[self.meaningfulParameters[i]]

newData = self.data

for i in range(len(avaragesValues)):

avaragesValues[i] /= len(self.data)

for i in range(len(newData)):

for j in range(len(self.meaningfulParameters)):

newData[i][self.meaningfulParameters[j]] -= avaragesValues[j]

corTable = [None]\*len(avaragesValues)

for i in range(len(corTable)):

corTable[i] = [0]\*len(avaragesValues)

for j in range(len(corTable[0])):

xy = 0

xx = 0

yy = 0

for example in newData:

xy += example[self.meaningfulParameters[i]]\*example[self.meaningfulParameters[j]]

xx += example[self.meaningfulParameters[i]]\*\*2

yy += example[self.meaningfulParameters[j]]\*\*2

corTable[i][j] = xy / ((xx\*yy)\*\*0.5)

print(f'CorTable is {corTable}')

self.meaningfulParameters = [0, 1, 3, 4, 11, 12]

def checkData(self):

print("Data checking start")

# y - це колонка isDark (15), дослідимо параметри на значемість та мультиколінеарність

for i in range(len(self.names)-1):

avarege = 0

avarege0 = 0

n0 = 0

avarege1 = 0

n1 = 0

x = []

y = []

for example in self.data:

x.append(example[i])

y.append(example[self.yColumnIndex])

avarege += example[i]

if example[self.yColumnIndex] == 0:

avarege0 += example[i]

n0 += 1

else:

avarege1 += example[i]

n1 += 1

plt.scatter(x,y)

plt.show()

avarege /= len(self.data)

if i == self.yColumnIndex:

self.percentOfDark = avarege

avarege0 /= n0

avarege1 /= n1

print(f'Avarage in column {self.names[i]} is {avarege}; \nLight fantazy - {avarege0}\nDark fantazy - {avarege1}')

self.meaningfulParameters = [0,1,3,4,11,12,13]

self.checkMulticol()

def overSamplingData(self):

#вирівнюємо вибірку

countOfNewData = 0

neededNewCount = int(len(self.data)\*(1-2\*self.percentOfDark))

for example in self.data:

if example[self.yColumnIndex] == 1:

newExample = example

for paramIndex in self.meaningfulParameters:

newExample[paramIndex] \*= random.uniform(0.85,1.15)

self.data.append(newExample)

countOfNewData += 1

if countOfNewData >= neededNewCount:

break

print("Data is ready to analyze")

def costFunction(self):

result = 0

for example in self.data:

p = self.expon(example)

if p == 0 or p == 1:

self.isFinish = True

else:

result += example[self.yColumnIndex]\*math.log(p)

result += (1-example[self.yColumnIndex])\*math.log(1-p)

return -result/len(self.data)

def expon(self, example):

z = 0

for i in range(len(self.oldParametrs)):

if i == len(self.oldParametrs) - 1:

z += self.oldParametrs[i]

else:

z += self.oldParametrs[i] \* example[self.meaningfulParameters[i]]

return(1 / (1 + (math.e \*\* (-z))))

def trainData(self):

print("Training is start")

self.isFinish = False

count = 0

self.oldParametrs = self.parametrs

prevValue = self.costFunction()

while not self.isFinish:

self.oldParametrs = self.parametrs

if count % 100 == 0:

print(count)

print(self.parametrs)

count += 1

for j in range(len(self.parametrs)):

sum = 0

for i in range(len(self.data)):

subsum = self.expon(self.data[j])-self.data[i][self.yColumnIndex]

mn = 1

if j != len(self.parametrs)-1:

mn = self.data[i][self.meaningfulParameters[j]]

sum += subsum\*mn

sum /= len(self.data)

self.parametrs[j] -= sum\*self.alpha

nowValue = self.costFunction()

if abs(nowValue-prevValue) < 0.00001:

self.isFinish = True

self.oldParametrs = self.parametrs

def saveResult(self):

file = ParamFile("solution/LogResParams.bp","w")

j = 0

resParams = []

for i in range(len(self.names)-2):

if i in self.meaningfulParameters:

resParams.append(self.parametrs[j])

j += 1

else:

resParams.append(0)

resParams.append(self.parametrs[j])

file.writeToFile(resParams)

main.py

from Interface import Interface

interface = Interface()

interface.go()

TestWorkerLogistic.py

from FileWorker import MaxMinFile, ParamFile, CellFile

import math

import matplotlib.pyplot as plt

class TestWorkerLogistic:

data = []

names = []

results = []

x = []

y = []

paramNames = [

"year", "pages",

"ratingLiveLib", "ratingLitRes",

"isRealistic",

"characters", "female", "male",

"twists", "race",

"friends", "loves", "family", "enemy",

"locations",

"isDark",

"countOfRate"

]

def \_\_init\_\_(self, data, names, results):

self.data = data

self.names = names

self.results = results

maxMinFile = MaxMinFile("solution/MaxMin.bp")

maxValues = maxMinFile.getMaxArray()

minValues = maxMinFile.getMinArray()

print(maxValues)

print(minValues)

for i in range(len(self.data)):

for j in range(len(self.data[i])):

data[i][j] = (data[i][j] - minValues[j])/(maxValues[j] - minValues[j])

def printCell(self, fileName):

path = "solution/tree/"

file = CellFile(path + fileName + ".bp")

cellData = file.getCellValues()

if cellData[0] == "1":

print(f'res - {cellData[1]}')

return

if fileName == "root":

self.printCell("l")

else:

self.printCell(fileName+"l")

parametr = int(cellData[1])

value = float(cellData[2])

print(f'{self.paramNames[parametr]} - {value}')

if fileName == "root":

self.printCell("r")

else:

self.printCell(fileName+"r")

def printTree(self):

self.printCell("root")

def goTesting(self):

self.testLogisticReg()

plt.scatter(self.x, self.y)

#plt.scatter(self.x, self.results)

plt.show()

self.testTree()

def testTree(self):

resultNames = ["Light", "Dark"]

print("----------------------------------")

print("Start testing tree")

rss = 0

tss = 0

avarageY = 0

for i in range(len(self.data)):

avarageY += self.results[i]

avarageY /= len(self.data)

rightAnswers = 0

path = "solution/tree/"

for i in range(len(self.data)):

fileName = "root"

predict = 0

while True:

file = CellFile(path+fileName+".bp")

if fileName == "root":

fileName = ""

cellData = file.getCellValues()

if cellData[0] == "1":

predict = int(cellData[1])

break

parametr = int(cellData[1])

value = float(cellData[2])

if self.data[i][parametr] > value:

fileName += "r"

else:

fileName += "l"

print(f'{self.names[i]} is {resultNames[self.results[i]]}; Prediction is {resultNames[predict]}')

rss += (self.results[i] - avarageY) \*\* 2

tss += (self.results[i] - predict) \*\* 2

if self.results[i] == predict:

rightAnswers += 1

print(f'Correctness of predictions - {rightAnswers / len(self.data)}')

print(f'RSS = {rss}, TSS = {tss}, R2 = {1 - rss / tss}')

def testLogisticReg(self):

rozp = [0]\*21

xInGisto = []

xValInGisto = -1

for \_ in range(21):

xInGisto.append(xValInGisto)

xValInGisto += 0.1

rss = 0

tss = 0

avarageY = 0

for i in range(len(self.data)):

avarageY += self.results[i]

avarageY /= len(self.data)

resultNames = ["Light", "Dark"]

print("----------------------------------")

print("Start testing logistic regresion")

paramFile = ParamFile("solution/LogResParams.bp")

parametrs = paramFile.getParams()

rightAnswers = 0

for i in range(len(self.data)):

predictionVal = self.expon(parametrs,self.data[i])

rozp[int((predictionVal-self.results[i])\*10) + 10] += 1

if predictionVal>0.5:

prediction = 1

else:

prediction = 0

print(f'{self.names[i]} is {resultNames[self.results[i]]}; Prediction is {resultNames[prediction]}')

rss += (self.results[i] - avarageY) \*\* 2

tss += (self.results[i] - predictionVal) \*\* 2

if self.results[i] == prediction:

rightAnswers += 1

plt.bar(xInGisto,rozp,0.4)

plt.show()

print(f'Correctness of predictions - {rightAnswers/len(self.data)}')

print (f'RSS = {rss}, TSS = {tss}, R2 = {1 - rss/tss}')

def expon(self, parametrs, example):

z = 0

for i in range(len(parametrs)):

if i == len(parametrs) - 1:

z += parametrs[i]

else:

z += parametrs[i] \* example[i]

self.x.append(z)

self.y.append((1 / (1 + (math.e \*\* (-z)))))

return (1 / (1 + (math.e \*\* (-z))))

TextWorker.py

from FileWorker import ControlFile

class TextWorker:

\_\_count = 0

\_\_womanCount = 0

\_\_manCount = 0

\_\_twistCount = 0

\_\_raceCount = 0

\_\_locationCount = 0

\_\_frindlyCount = 0

\_\_loveCount = 0

\_\_familyCount = 0

\_\_enemyCount = 0

def \_\_init\_\_(self, text):

#init control words

friendly = ControlFile("controlWords/friendly.bp").getControlWords()

enemy = ControlFile("controlWords/enemy.bp").getControlWords()

race = ControlFile("controlWords/race.bp").getControlWords()

love = ControlFile("controlWords/love.bp").getControlWords()

locations = ControlFile("controlWords/locations.bp").getControlWords()

femaleNames = ControlFile("controlWords/femaleNames.bp").getControlWords()

family = ControlFile("controlWords/family.bp").getControlWords()

twist = ControlFile("controlWords/twist.bp").getControlWords()

maleNames = ControlFile("controlWords/maleNames.bp").getControlWords()

#analyze

textArray = text.split(" ")

self.\_\_count = len(textArray)

for word in textArray:

l = len(word)

if word in maleNames or (l>2 and word[:l-1] in maleNames) or (l>3 and word[:l-2] in maleNames):

self.\_\_manCount += 1

continue

if word in femaleNames or (l>2 and word[:l-1] in femaleNames) or (l>3 and word[:l-2] in femaleNames):

self.\_\_womanCount += 1

continue

word.lower

if word in twist or (l>2 and word[:l-1] in twist) or (l and word[:l-2] in twist):

self.\_\_twistCount += 1

if word in race or (l>2 and word[:l-1] in race) or (l>3 and word[:l-2] in race):

self.\_\_raceCount += 1

if word in locations or (l>2 and word[:l-1] in locations) or (l>3 and word[:l-2] in locations):

self.\_\_locationCount += 1

if word in friendly or (l>2 and word[:l-1] in friendly) or (l>3 and word[:l-2] in friendly):

self.\_\_frindlyCount += 1

continue

if word in love or (l > 2 and word[:l - 1] in love) or (l > 3 and word[:l - 2] in love):

self.\_\_loveCount += 1

continue

if word in family or (l>2 and word[:l-1] in family) or (l>3 and word[:l-2] in family):

self.\_\_familyCount += 1

continue

if word in enemy or (l>2 and word[:l-1] in enemy) or (l>3 and word[:l-2] in enemy):

self.\_\_enemyCount += 1

def getData(self):

return {

'count': (self.\_\_womanCount + self.\_\_manCount) / self.\_\_count,

'woman\_count': self.\_\_womanCount / (self.\_\_womanCount+self.\_\_manCount),

'man\_count': self.\_\_manCount / (self.\_\_womanCount+self.\_\_manCount),

'twist\_count': self.\_\_twistCount / self.\_\_count,

'race\_count': self.\_\_raceCount / self.\_\_count,

'location\_count': self.\_\_locationCount / self.\_\_count,

'friends\_count': self.\_\_frindlyCount / (self.\_\_frindlyCount+self.\_\_loveCount+self.\_\_familyCount+self.\_\_enemyCount),

'lovers\_count': self.\_\_loveCount / (self.\_\_frindlyCount+self.\_\_loveCount+self.\_\_familyCount+self.\_\_enemyCount),

'families\_count': self.\_\_familyCount / (self.\_\_frindlyCount+self.\_\_loveCount+self.\_\_familyCount+self.\_\_enemyCount),

'enemies\_count': self.\_\_enemyCount / (self.\_\_frindlyCount+self.\_\_loveCount+self.\_\_familyCount+self.\_\_enemyCount),

}

FileWorker.py

class FileWorker:

\_\_rootPath = "data/"

fileData = ""

file = None

def \_\_init\_\_ (self, fileName, format = 'r'):

self.file = open(self.\_\_rootPath+fileName, format)

if format == 'r':

self.fileData = self.file.read()

class MainFile(FileWorker):

def getFilesNames(self):

return self.fileData.splitlines()

class BookFile(FileWorker):

def getBookData(self):

return self.fileData.splitlines()

class ControlFile(FileWorker):

def getControlWords(self):

return self.fileData.splitlines()

class MaxMinFile(FileWorker):

def writeToFile(self, arr):

for i in arr:

self.file.write(f'{i} ')

self.file.write("\n")

def getMaxArray(self):

result = []

for i in self.fileData.splitlines()[0].split(" "):

if i != '':

result.append(float(i))

return result

def getMinArray(self):

result = []

for i in self.fileData.splitlines()[1].split(" "):

if i != '':

result.append(float(i))

return result

class ParamFile(FileWorker):

def writeToFile(self, param):

for p in param:

self.file.write(f'{p} ')

def getParams(self):

result = []

for param in self.fileData.split(" "):

if param != '':

result.append(float(param))

return result

class CellFile(FileWorker):

def writeCell(self, parametr, avarage):

self.file.write(f'0\n{parametr}\n{avarage}')

def writeLastCell(self, predict):

self.file.write(f'1\n{predict}')

def getCellValues(self):

return self.fileData.splitlines()

Interface.py

from DBWorker import DBWorker

from FileWorker import MainFile, BookFile

from Book import Book

from DataWorker import DataWorker

from LogisticRegresion import LogisticRegresion

from TestWorkerLogistic import TestWorkerLogistic

from DecisionTree import DecisionTree

class Interface:

def go(self):

print("Program start")

isValueInput = False

db = DBWorker()

data = DataWorker()

while not isValueInput:

inputValue = input("What do you want to do?\n 0 - upload data to database\n 1 - train the model \n 2 - predict for my book\n")

if inputValue == "0":

db.create\_table()

print("Uploading data...")

fileWorker = MainFile('books.bp')

for bookNameFile in fileWorker.getFilesNames():

bookFile = BookFile(bookNameFile)

book = Book(bookFile.getBookData())

data.updateData(book)

db.insert\_params(

book.getName(),

book.year,

book.pages,

book.ratingLiveLib,

book.ratingLitRes,

book.isReal,

book.analyzeResult['count'],

book.analyzeResult['woman\_count'],

book.analyzeResult['man\_count'],

book.analyzeResult['twist\_count'],

book.analyzeResult['race\_count'],

book.analyzeResult['friends\_count'],

book.analyzeResult['lovers\_count'],

book.analyzeResult['families\_count'],

book.analyzeResult['enemies\_count'],

book.analyzeResult['location\_count'],

book.count,

book.isDark

)

isValueInput = True

data.saveMaxMinData()

elif inputValue == "1":

loadedData = db.fetch\_all\_data()

normalizedData = data.loadData(loadedData)

logReg = LogisticRegresion(normalizedData)

logReg.go()

logReg.saveResult()

decisionTree = DecisionTree(normalizedData)

decisionTree.saveData()

isValueInput = True

elif inputValue == "2":

print("Uploading test data...")

fileWorker = MainFile('test.bp')

testData = []

testNames = []

testResults = []

for bookNameFile in fileWorker.getFilesNames():

bookFile = BookFile(bookNameFile)

book = Book(bookFile.getBookData())

testNames.append(book.name)

testResults.append(book.isDark)

testData.append([

book.year,

book.pages,

book.ratingLiveLib,

book.ratingLitRes,

book.isReal,

book.analyzeResult['count'],

book.analyzeResult['woman\_count'],

book.analyzeResult['man\_count'],

book.analyzeResult['twist\_count'],

book.analyzeResult['race\_count'],

book.analyzeResult['friends\_count'],

book.analyzeResult['lovers\_count'],

book.analyzeResult['families\_count'],

book.analyzeResult['enemies\_count'],

book.analyzeResult['location\_count'],

])

logTest = TestWorkerLogistic(testData,testNames,testResults)

logTest.goTesting()

#logTest.printTree()

isValueInput = True

else:

print("error")