MSc Data Science: Coventry University, UK

PROGRAMMING FOR DATA SCIENCE

2025.1 BATCH

COURSE WORK

Name – Perera.P.P.G.S

User Name - COMScDS251P-007

9-26-2025

Table of Contents

1. University management system technical document

This is the technical evaluation of a university management system. Implemented by using the Python programming language. In the program, I used object oriented programming mechanisms to develop various sections in university, like, departments, students, faculty, and relationships, which is suitable for them. The main reason that I used object oriented programming is code reusability, code maintainability, and code scalability.

This module is entirely built on object oriented framework, using classes and inheritance to develop a logical hierarchy.

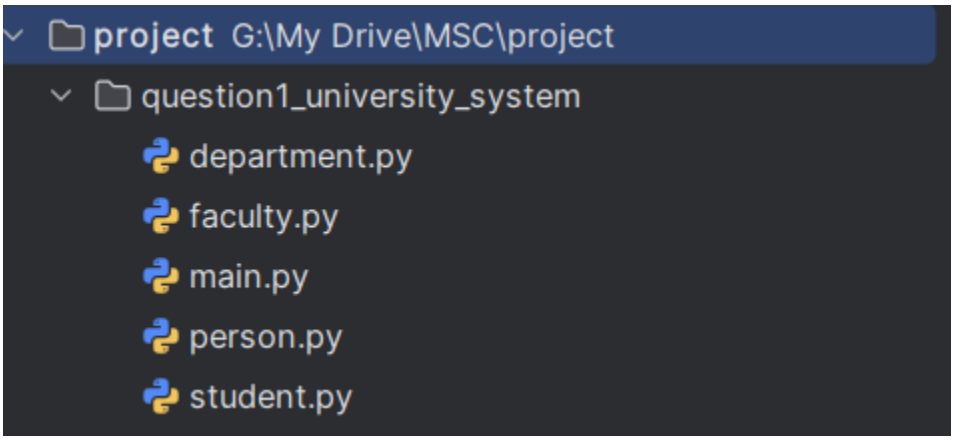
1.1.System Architecture

Person.py script is used to define base classes for all people in the system. This establishes a foundation for inheritance.

Student.py script contains classes that is specific to students, and also contains inheritance that are from the person class. This script manages student-related functions likes course enrollment and GPA calculations.

Faculty.py script contains classes for members in the faculty. and also contains inheritance that are from the person class. This script manages faculty specific functions like checking qualifications and assigning workload.

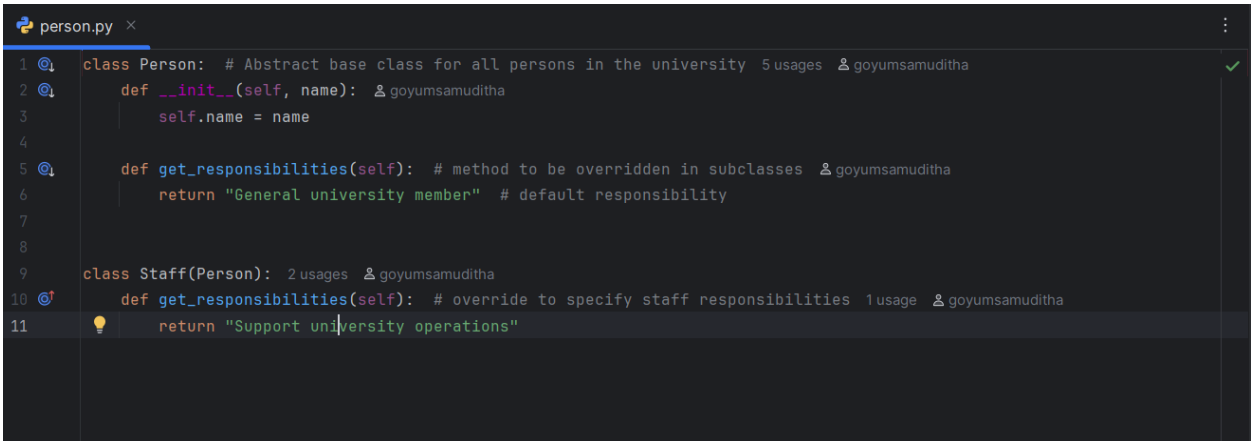
Department.py script contains classes and attributes related to department functions and functions that are related to courses. They are course enrollment, checking prerequisites that are related to the course class. And add courses and faculty that are related to the department class.



1.2. Class Descriptions

1.2.1. person.py

This script is the root module of the class hierarchy for all people in the university management system.



- Base class – Person
    - Inherits from - Person
    - Attributes:
        - name: person's name
    - Methods
        - __init__(self, name) : use the constructor initializes the name attribute.
        - get_responsibilities(self): This method is used to be overridden by subclasses.
- Concrete class – Staff
    - Inherits from - Person
    - Methods
        - get_responsibilities(self): Overrides the parent method to return the given specific string.

1.2.2. student.py

This script is used to model the student entities within the program.

```python
from person import Person

class Student(Person):  # Abstract base class for students  2 usages  goyumsamuditha
    def __init__(self, name, student_id, department):  goyumsamuditha
        super().__init__(name)
        self.student_id = student_id
        self.department = department
        self.courses = []
        self.__gpa = 0.0  # private attribute

    def enroll_course(self, course):  # enroll in a course if prerequisites are met and capacity allows  3 usages  goyumsa
        if course.check_prerequisites(self):    # check if prerequisites are met
            if len(course.enrolled_students) < course.limit:  # check capacity
                self.courses.append(course.code)
                course.enrolled_students.append(self)
                print(f"{self.name} enrolled in {course.code}")  # successful enrollment
            else:
                print(f"Cannot enroll {self.name}: course {course.code} is full")  # course full
        else:
            print(f"Cannot enroll {self.name}: prerequisites not met for {course.code}")  # prerequisites not met

    def drop_course(self, course):  # drop a course  goyumsamuditha
        if course.code in self.courses:  # check if enrolled
            self.courses.remove(course.code)
            course.enrolled_students.remove(self)
            print(f"{self.name} dropped {course.code}")

    def set_gpa(self, gpa):  # set GPA with validation  2 usages  goyumsamuditha
        if 0.0 <= gpa <= 4.0:
            self.__gpa = gpa
        else:
            raise ValueError("GPA must be between 0.0 and 4.0")  # validate GPA
```

```python
    def calculate_gpa(self):  # return current GPA  2 usages  goyumsamuditha
        return self.__gpa

    def get_academic_status(self):  # determine academic status based on GPA (Dean's List, Probation, Good Standing)  2 us
        if self.__gpa >= 3.7:
            return "Dean's List"
        elif self.__gpa < 2.0:
            return "Probation"
        return "Good Standing"

    def get_responsibilities(self):  # override to specify student responsibilities  goyumsamuditha
        return "Attend classes, complete assignments"

class Undergraduate(Student):  # Concrete class for undergraduate students  2 usages  goyumsamuditha
    def get_responsibilities(self):  # override to specify undergraduate responsibilities  1 usage  goyumsamuditha
        return "Attend lectures, participate in labs, complete assignments"

class Graduate(Student):  # Concrete class for graduate students  2 usages  goyumsamuditha
    def get_responsibilities(self):  # override to specify graduate responsibilities  1 usage  goyumsamuditha
        return "Conduct research, attend seminars, complete assignments"
```

- Base class – Student
    - Inherits from - Person
    - Attributes:

        - name: student's name
        - student_id : Unique identifier of the student.
        - department: Department of the student.
        - courses: list of courses enrolled by the student.
        - __gpa: calculate and store student's GPA
    - Methods
        - __init__(self, name, student_id, department): initialize student-specific attributes.
        - enroll_course(self, course): enroll student into the course after checking capacity and completion of prerequisites.
        - drop_course(self, course): drop a student from a course.
        - set_gpa(self, gpa): check if the GPA is between 0.0 and 4.0, if not raise ValueError.
        - calculate_gpa(self): calculate current GPA.
        - get_academic_status(self): Determine academic status based on calculated GPA.
        - get_responsibilities(self): override the parent
- Concrete class – Undergraduate
    - Inherits from - Student
    - Methods
        - get_responsibilities(self): override the parent method to specify undergraduate responsibilities.
- Concrete class – Graduate
    - Inherits from - Student
    - Methods
        - get_responsibilities(self): override the parent method to specify graduate responsibilities.

1.2.3. faculty.py

This script is used to maintain the faculty members of the university.

faculty.py

```python
from person import Person


class Faculty(Person):  # Abstract base class for faculty
    def __init__(self, name, qualification, workload):
        super().__init__(name)
        self.qualification = qualification
        self.workload = workload

    def calculate_workload(self):  # return workload
        return f"Workload: {self.workload} courses"

    def get_responsibilities(self):  # override to specify faculty responsibilities
        return "Teach courses, mentor students"


class Professor(Faculty):  # Concrete class for professors
    def get_responsibilities(self):  # override to specify professor responsibilities
        return "Conduct research, teach advanced courses, mentor graduate students"


class Lecturer(Faculty):  # Concrete class for lecturers
    def get_responsibilities(self):  # override to specify lecturer responsibilities
        return "Teach undergraduate courses, develop curriculum"


class TA(Faculty):  # Concrete class for teaching assistants
    def get_responsibilities(self):  # override to specify TA responsibilities
        return "Assist in teaching, grade assignments, hold office hours"
```

- Base class – Faculty
    - Inherits from - Person
    - Attributes:
        - name: employer's name
        - qualification: academic qualification of the member.
        - workload: Number of courses that are assigned to the member.
    - Methods
        - __init__(self, name, qualification, workload): Initializes faculty specific attributes and calls the parent constructor.
        - calculate_workload(self): return formatted link with the workload of the user.
        - get_responsibilities(self): override by specific faculty responsibilities.

- Concrete class – Professor
    - Inherits from - Faculty
    - Methods:
        - get_responsibilities(self): override by specific responsibilities for professors.
- Concrete class – Lecturer
    - Inherits from - Faculty
    - Methods:
        - get_responsibilities(self): override by specific responsibilities for lecturers.
- Concrete class – TA
    - Inherits from - Faculty
    - Methods:
        - get_responsibilities(self): override by specific responsibilities for teaching assistant.

1.2.4. department.py

This script is used to maintain department structure and courses.

department.py

```python
class Course: # Represents a course in the university  5 usages  goyumsamuditha
    def __init__(self, code, title, credits, prerequisites=None,limit=30): # prerequisites is a list of course codes
        self.code = code
        self.title = title
        self.credits = credits
        self.enrolled_students = [] # List of student IDs
        self.prerequisites = prerequisites if prerequisites else [] # List of prerequisite course codes
        self.limit = limit

    def check_prerequisites(self, student): # Check if a student meets the prerequisites  1 usage (1 dynamic)  goyumsamuditha
        return all(prereq in student.courses for prereq in self.prerequisites)


class Department: # Represents a department in the university  2 usages  goyumsamuditha
    def __init__(self, name: str):  goyumsamuditha
        self.name = name
        self.courses: List[Course] = []
        self.faculty: List[str] = []  # Specify type of faculty, e.g., List[str] for faculty names

    def add_course(self, course: Course): # Add a course to the department  2 usages  goyumsamuditha
        self.courses.append(course)

    def add_faculty(self, faculty: str): # Add a faculty member to the department  2 usages  goyumsamuditha
        self.faculty.append(faculty)
```

- Concrete class – Course
    - Attributes:
        - code: unique identifier for the course.
        - title: name of the course.
        - credits: Number of credits required for the course.
        - enrolled_students: list of students enrolled in the course.
        - prerequisites: list of prerequisites required for the course.
        - limit: maximum number of students that can enroll in the course.
    - Methods
        - __init__(self, code, title, credits, prerequisites=None,limit=30): this constructor initialize the course attributes.
        - check_prerequisites(self, student): check if the student meets prerequisites.
- Concrete class – Department
    - Attributes:
        - name: name of the department.
        - course: list of courses that are associated with each department.
        - faculty: the faculty that the department belongs to.

1.2.5. main.py

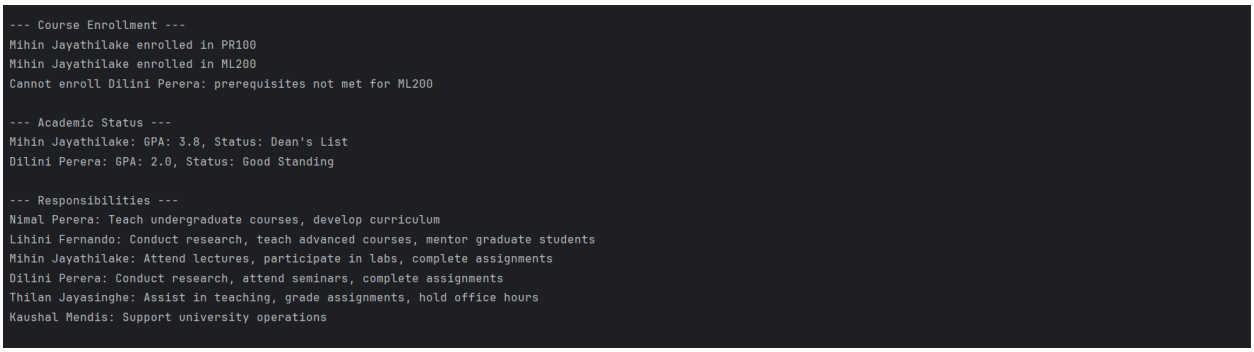
This is the script that serves as a demonstration of the real system's functionality. This script follows a logical sequence of all the operations.

```python
# main.py
from person import Staff
from student import Undergraduate, Graduate
from faculty import Professor, Lecturer, TA
from department import Department, Course

def main():  1 usage  goyumsamuditha
    # Create a department
    department_d1 = Department("Computer Science") # Department for CS courses

    # Create courses
    course_c01 = Course( code: "PR100",  title: "Introduction to Programming",  credits: 2) # Programming course
    course_c02 = Course( code: "ML200",  title: "Machine Learning",  credits: 2, prerequisites=["PR100"]) # ML course with PR100 a

    # Assign courses to department
    department_d1.add_course(course_c01) # Add PR100 course to CS department
    department_d1.add_course(course_c02) # Add ML200 course to CS department

    # Create faculty
    lecturer_l001 = Lecturer( name: "Nimal Perera",  qualification: "MSc in Programming",  workload: 3) # Lecturer for PR100 course
    lecturer_l002 = Professor( name: "Lihini Fernando",  qualification: "PhD in Data Science",  workload: 2) # Professor for ML200
    teaching_assistant_ta001 = TA( name: "Thilan Jayasinghe",  qualification: "BSc in Data Science",  workload: 1) # TA for PR100 c
    staff_st001 = Staff("Kaushal Mendis") # Staff member

    # Assign faculty to department
    department_d1.add_faculty(lecturer_l001) # Add Lecturer to CS department
    department_d1.add_faculty(lecturer_l002) # Add Professor to CS department


    # Create students
    student_s001 = Undergraduate( name: "Mihin Jayathilake",  student_id: "CS001",  department: "CS") # Undergraduate student in
    student_s002 = Graduate( name: "Dilini Perera",  student_id: "CS002",  department: "CS") # Graduate student in CS
```

```python
    print("\n--- Course Enrollment ---")
    # Enroll students in courses
    student_s001.enroll_course(course_c01) # should succeed
    student_s001.enroll_course(course_c02)  # prerequisite check

    student_s002.enroll_course(course_c02)  # should fail due to missing prerequisite


    # Set GPA
    student_s001.set_gpa(3.8)
    student_s002.set_gpa(2.0)


    # Show academic status
    print("\n--- Academic Status ---")
    print(f"{student_s001.name}: GPA: {student_s001.calculate_gpa()}, Status: {student_s001.get_academic_status()}") # S
    print(f"{student_s002.name}: GPA: {student_s002.calculate_gpa()}, Status: {student_s002.get_academic_status()}") # S


    print("\n--- Responsibilities ---")
    # Show responsibilities
    people = [lecturer_l001,lecturer_l002, student_s001, student_s002, teaching_assistant_ta001, staff_st001] # List of
    for person in people:
        print(f"{person.name}: {person.get_responsibilities()}") # Print responsibilities of each person

if __name__ == "__main__":
    main()
```

1. Create a department.
2. Create a new course with prerequisites.
3. Faculty creation with staff members, including lecturers, professors, and teaching assistants.
4. Added courses to the department.
5. Added faculties and faculty members to the department.
6. Undergraduate student enrollment.
7. Graduate students' enrollment.

This is the sample outcome of the program.



This system was entirely developed based on object-oriented principles, using classes and inheritance to build a logical hierarchy. And this system uses the inheritance mechanism widely. With the student, faculty, and staff classes, which all of are inheritance to the "person" base class. Using this mechanism, I was able to avoid code duplication and provide clear structure and understandability.

Attributes like __gpa are made as private attributes of the "Student" class. This helped me to maintain data integrity.

get_responsibilities(self) is a good example for usage in polymorphism in this scripts.

When I developed this system, I used multiple script files with each script having a unique single task to do. This will increase the ease of maintenance and management. And facilitate testing each component separately.

1.3.Learning Outcomes

This project is an effective learning opportunity for understanding the fundamentals of software development concepts. How to master object oriented concepts, how to define architecture based on classes and objects, code organization, and relationship mapping are the other key outcomes.

2. E-commerce platform data analysis

This is a technical evaluation of an E-commerce platform data analysis developed using Python programming language. In here I developed end to end data process workflow, from raw data collection, data leaning and processing, analyzing, and predictive modeling.

This system follows some critical stages when structuring,

- Data scraping and collecting.
- Data cleaning and processing
- Descriptive analysis
- Predictive analysis and modeling
- Data visualization.

For the data collection phase, we used the http://books.toscrape.com site. It is well known public practice site for web scraping.

2.1.System Architecture

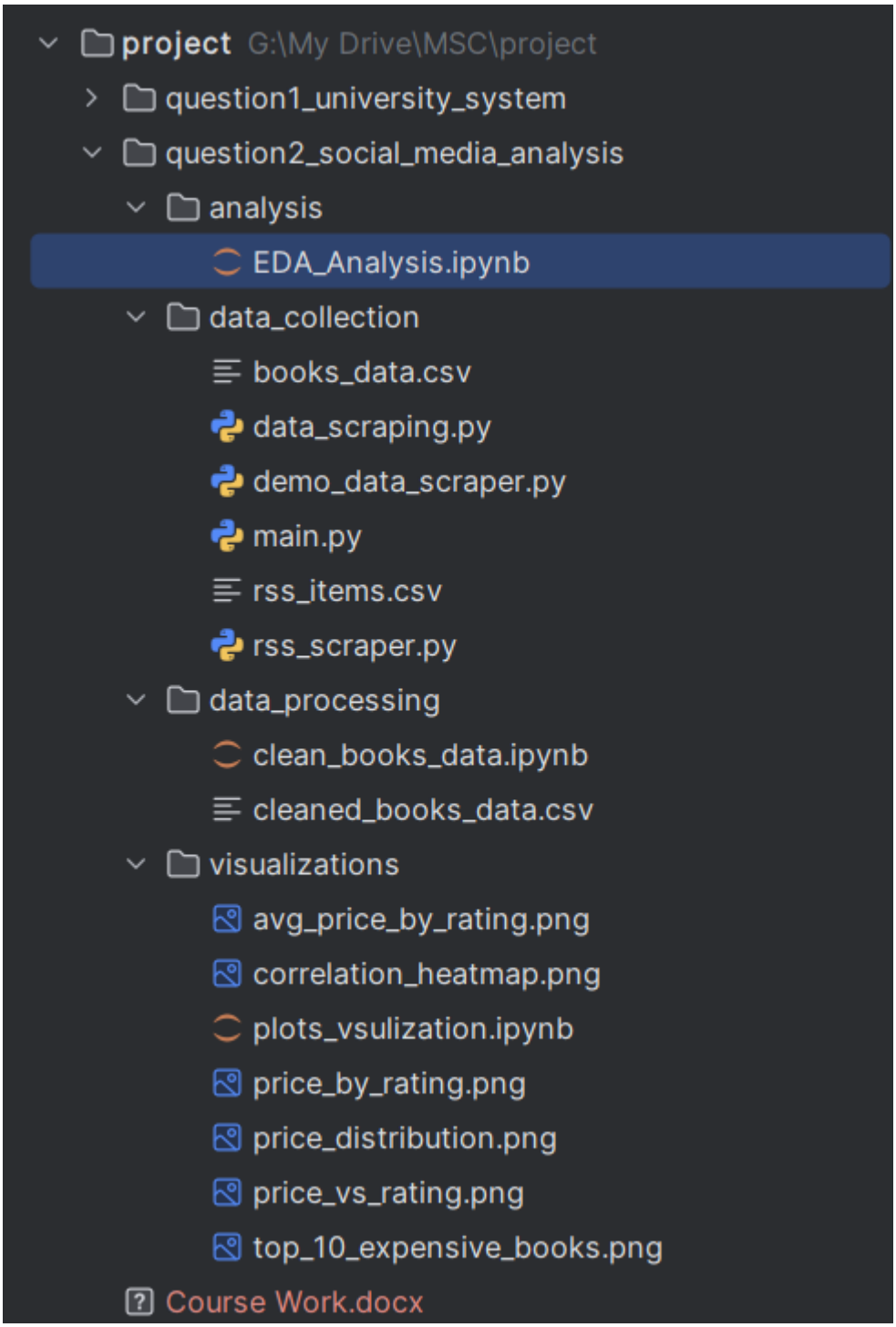
This project follows top-level directory architecture, “question2_social_media_analysis” is the main directory, followed with the subdirectories.

The data_collection directory maintains the scrape_books.py script, which is used to scrape data from books.toscrape.com website and handle basic errors. And also main.py file, which is the script's main function to run scraping and save csv file. And also save books_data.csv which is raw data file contain data that scraped from the website.

The data_processing directory maintains clean_books_data.ipynb, which is a Jupiter notebook used to clean and preprocess books_data.csv. Output of the cleaned data set clean_books.csv is saved back same directory.

The analysis directory maintains EDA_Analysis.ipynb Jupiter notebook, which performs statistical analysis as well as predictive analysis for the cleaned data set.

The visualization directory maintains plots_visualization.ipynb Jupiter notebook, which performs visual outputs generated from the data set. All the plots are saved in image type in this directory.



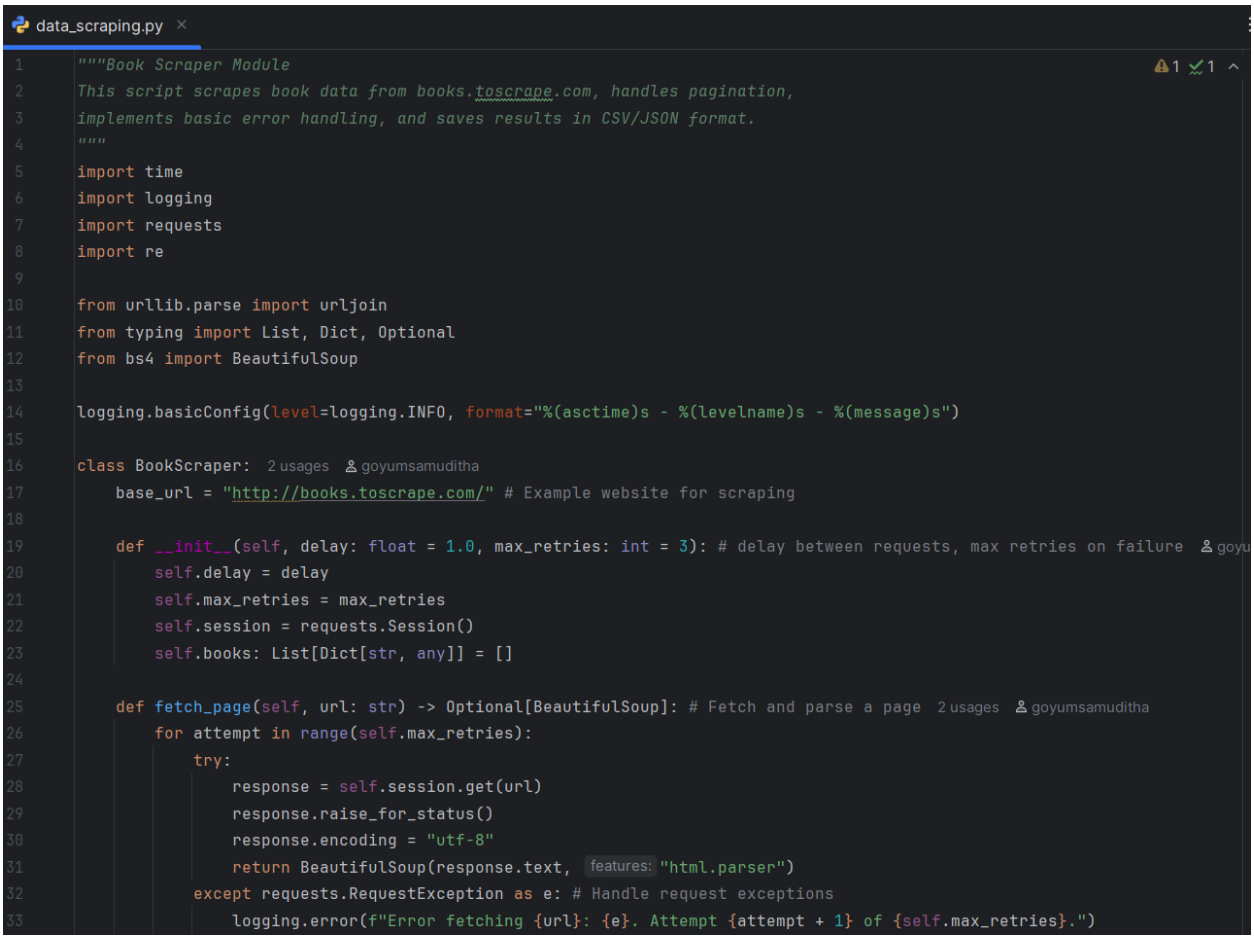
2.2.Data scraping and collecting

There are two main scripts related to this stage. data_scraping.py and main.py scripts are run in here.

The data_scraping.py is a script that scrapes data from the http://books.toscrape.com website. BookScraper is the main class with in this module. Using pagination iterates through multiple pages from the website. This script parses title, price, rating, category, availability and description from each book.

The main.py is the main control hub in data collection. This script calls the BookScraper class to start the data collection process. Using "|save_to_csv()" function, this script saved scraped data into a CSV file.

And also, this script configures some warning and information messages on the scraping process.



```
                return BeautifulSoup(response.text, features: "html.parser")
            except requests.RequestException as e: # Handle request exceptions
                logging.error(f"Error fetching {url}: {e}. Attempt {attempt + 1} of {self.max_retries}.")
                time.sleep(self.delay)
        return None

    def scrape(self, pages: int = 3) -> None: # Scrape a given number of pages  1 usage   goyumsamuditha
        for page in range(1, pages + 1):
            url = f"{self.base_url}catalogue/page-{page}.html"
            html = self.fetch_page(url)
            if not html:
                continue

            books = html.find_all( name: "article", class_="product_pod")
            for book in books:
                title = book.h3.a["title"]

                # Clean price text
                price_text = book.find("p", class_="price_color").text
                price_clean = re.sub( pattern: r"[^\d.]", repl: "", price_text)
                price = float(price_clean)

                rating = book.p["class"][1]

                # Build absolute book URL safely
                relative_url = book.h3.a["href"]
                book_url = urljoin(self.base_url + "catalogue/", relative_url)

                # Fetch category from detail page
                category, availability = "Unknown", "Unknown"
```

```
                # Fetch category from detail page
                category, availability = "Unknown", "Unknown"
                book_html = self.fetch_page(book_url)
                if book_html:
                    breadcrumb = book_html.select("ul.breadcrumb li a")
                    if len(breadcrumb) >= 3:
                        category = breadcrumb[2].text.strip()

                    availability = book_html.select_one("p.availability")
                    if availability:
                        availability = availability.text.strip()

                    description = book_html.select_one("#product_description ~ p")
                    if description:
                        description = description.text.strip()

                        # Append book data to the list
                        self.books.append({
                            "title": title,
                            "price": price,
                            "rating": rating,
                            "category": category,
                            "availability": availability,
                            "description": description
                        })
                time.sleep(self.delay)  # Respectful delay between requests
```

2.3.Data cleaning and processing

In the clean_books_data.ipynb is the Jupiter notebook, which performs data cleaning and preprocessing steps related to the books_data.csv. This script transforms the scraped raw data into cleaned and structured data. This is very helpful for performing statistical analysis and predictive model building.

This script used books_data.csv as input file, which is raw data scraped from the http://books.toscrape.com website.

This script mainly used pandas and numpy libraries for perform cleaning.

First, this script loads books_data.csv dataset in to the data frame by using pandas library. And the check the quality of the data, which means checking for missing values and duplicate records.

Then this script transforms the price column into the float type and removes currency symbol. This will be helpful when we are performing numerical analysis.

And after that this script standardizes the rating column, which is of the string type. Then it is mapped to an integer value. This makes sure we can perform some statistical analysis using rating column.

A final step of this script is to save after cleaning and preprocessing the data frame, is saved to a new CSV file called cleaned_books_data.csv which is available to perform analysis.

Q2 - Data Cleaning and Preprocessing

This notebook performs:

- **Data Cleaning** on `books_data.csv` (Part B)

```python
# Import necessary libraries
import pandas as pd
import numpy as np
```
[25]

```python
# Load the dataset
df = pd.read_csv('G:\\My Drive\\MSC\\project\\question2_social_media_analysis\\data_collection\\books_data.csv')
# display the first few rows of the dataframe
df.head()
```
[26]

5 rows × 6 cols — Static Output

	title	price	rating	category	availability	description
0	A Light in the Attic	51.77	Three	Poetry	In stock (22 available)	It's hard t
1	Tipping the Velvet	53.74	One	Historical Fiction	In stock (20 available)	"Erotic and
2	Soumission	50.10	One	Fiction	In stock (20 available)	Dans une Fr
3	Sharp Objects	47.82	Four	Mystery	In stock (20 available)	WICKED abov
4	Sapiens: A Brief History of Humankind	54.23	Five	History	In stock (20 available)	From a renc

```python
# Remove $ and convert to float
df['price'] = df['price'].replace(r'[\$,]', '', regex=True).astype(float)
df
```
[27]

11 rows × 998 rows × 6 cols — Static Output

	title	price	rating	category	availability	des
0	A Light in the Attic	51.77	Three	Poetry	In stock (22 available)	It's
1	Tipping the Velvet	53.74	One	Historical Fiction	In stock (20 available)	"Er
2	Soumission	50.10	One	Fiction	In stock (20 available)	Dan
3	Sharp Objects	47.82	Four	Mystery	In stock (20 available)	WIC
4	Sapiens: A Brief History of Humankind	54.23	Five	History	In stock (20 available)	Fro
...	...	...	...	...	...	...
993	Beyond Good and Evil	43.38	One	Philosophy	In stock (1 available)	Fri
994	Ajin: Demi-Human, Volume 1 (Ajin: Demi-Human…	57.06	Four	Sequential Art	In stock (1 available)	Hig
995	A Spy's Devotion (The Regency Spies of Londo…	16.97	Five	Historical Fiction	In stock (1 available)	In
996	1st to Die (Women's Murder Club #1)	53.98	One	Mystery	In stock (1 available)	Jam
997	1,000 Places to See Before You Die	26.08	Five	Travel	In stock (1 available)	Aro

```python
rating_mapping = {
    'One': 1,
    'Two': 2,
    'Three': 3,
    'Four': 4,
    'Five': 5
}
df['rating'] = df['rating'].map(rating_mapping)
df
```

[28]

11 rows · 998 rows × 6 cols — Static Output

	title	price	rating	category	availability	desc
0	A Light in the Attic	51.77	3	Poetry	In stock (22 available)	It's
1	Tipping the Velvet	53.74	1	Historical Fiction	In stock (20 available)	"Ero
2	Soumission	50.10	1	Fiction	In stock (20 available)	Dans
3	Sharp Objects	47.82	4	Mystery	In stock (20 available)	WICK
4	Sapiens: A Brief History of Humankind	54.23	5	History	In stock (20 available)	From
...	...	...	...	...	...	...
993	Beyond Good and Evil	43.38	1	Philosophy	In stock (1 available)	Frie
994	Ajin: Demi-Human, Volume 1 (Ajin: Demi-Huma…	57.06	4	Sequential Art	In stock (1 available)	High
995	A Spy's Devotion (The Regency Spies of Lond…	16.97	5	Historical Fiction	In stock (1 available)	In E
996	1st to Die (Women's Murder Club #1)	53.98	1	Mystery	In stock (1 available)	Jame
997	1,000 Places to See Before You Die	26.08	5	Travel	In stock (1 available)	Arou

```python
# Drop rows with missing values in title, price, or rating
df = df.dropna(subset=['title', 'price', 'rating'])
df.info()
```

[29]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 998 entries, 0 to 997
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   title         998 non-null    object
 1   price         998 non-null    float64
 2   rating        998 non-null    int64
 3   category      998 non-null    object
 4   availability  998 non-null    object
 5   description   998 non-null    object
dtypes: float64(1), int64(1), object(4)
```

```
# Remove duplicates
df = df.drop_duplicates()
df
[30]
```

11 rows · 998 rows × 6 cols — Static Output

	title	price	rating	category	availability	description
0	A Light in the Attic	51.77	3	Poetry	In stock (22 available)	It's hard to imagine a world wit
1	Tipping the Velvet	53.74	1	Historical Fiction	In stock (20 available)	"Erotic and absorbing...Written
2	Soumission	50.10	1	Fiction	In stock (20 available)	Dans une France assez proche de
3	Sharp Objects	47.82	4	Mystery	In stock (20 available)	WICKED above her hipbone, GIRL a
4	Sapiens: A Brief History of Humanki…	54.23	5	History	In stock (20 available)	From a renowned historian comes
...	...	...	...	...	...	...
993	Beyond Good and Evil	43.38	1	Philosophy	In stock (1 available)	Friedrich Nietzsche's Beyond Goo
994	Ajin: Demi-Human, Volume 1 (Ajin: D…	57.06	4	Sequential Art	In stock (1 available)	High school student Kei Nagai is
995	A Spy's Devotion (The Regency Spies…	16.97	5	Historical Fiction	In stock (1 available)	In England's Regency era, manner
996	1st to Die (Women's Murder Club #1)	53.98	1	Mystery	In stock (1 available)	James Patterson, bestselling aut
997	1,000 Places to See Before You Die	26.08	5	Travel	In stock (1 available)	Around the World, continent by c

```
# Reset the index
df = df.reset_index(drop=True)
df
[31]
```

11 rows · 998 rows × 6 cols — Static Output

	title	price	rating	category	availability	description
0	A Light in the Attic	51.77	3	Poetry	In stock (22 available)	It's hard to imagine a world wit
1	Tipping the Velvet	53.74	1	Historical Fiction	In stock (20 available)	"Erotic and absorbing...Written
2	Soumission	50.10	1	Fiction	In stock (20 available)	Dans une France assez proche de
3	Sharp Objects	47.82	4	Mystery	In stock (20 available)	WICKED above her hipbone, GIRL a
4	Sapiens: A Brief History of Humanki…	54.23	5	History	In stock (20 available)	From a renowned historian comes
...	...	...	...	...	...	...
993	Beyond Good and Evil	43.38	1	Philosophy	In stock (1 available)	Friedrich Nietzsche's Beyond Goo
994	Ajin: Demi-Human, Volume 1 (Ajin: D…	57.06	4	Sequential Art	In stock (1 available)	High school student Kei Nagai is
995	A Spy's Devotion (The Regency Spies…	16.97	5	Historical Fiction	In stock (1 available)	In England's Regency era, manner
996	1st to Die (Women's Murder Club #1)	53.98	1	Mystery	In stock (1 available)	James Patterson, bestselling aut
997	1,000 Places to See Before You Die	26.08	5	Travel	In stock (1 available)	Around the World, continent by c

```
# save the cleaned dataset
df.to_csv('cleaned_books_data.csv', index=False)
print(f"Cleaned data saved to 'cleaned_books_data.csv' with {len(df)} records.")
[32]
```

```
Cleaned data saved to 'cleaned_books_data.csv' with 998 records.
```

2.4.Exploratory Data Analysis

In the EDA_analysis.ipynb notebook is performed exploratory data analysis and statistical analysis are performed for the cleaned_books_data.csv dataset.

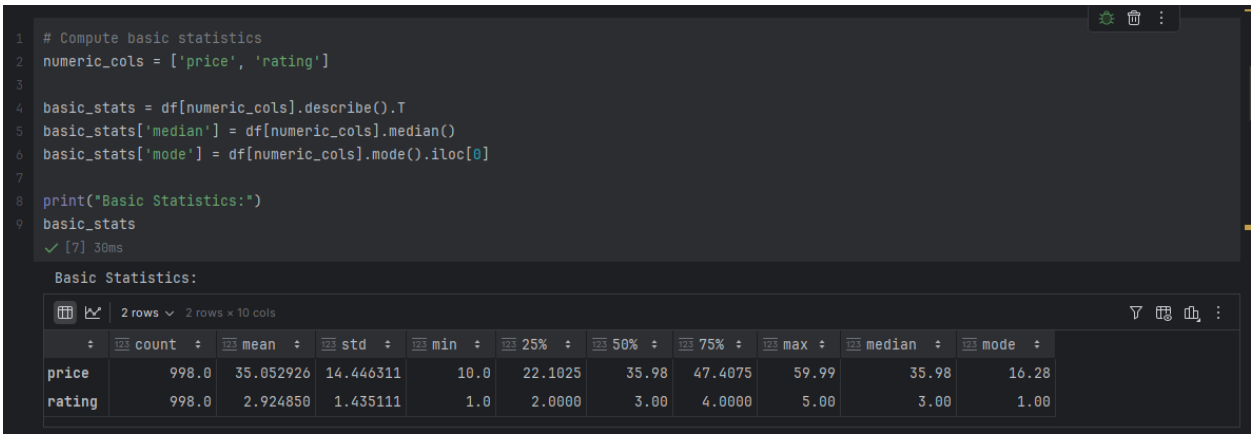
These notebooks use pandas, numpy, scipy, and sklearn libraries.

First, this script performs descriptive analysis for the price and rating columns. This will provide a clear understanding of the central tendency and other statistical details.

And then I performed mean, median, mode, and standard deviation for the price column. This will help to identify the distribution of book prices. And a similar analysis was performed on ratings columns as well to identify the rating distribution.

This script performed IQR analysis to identify outliers in the price column. In this code it defines any data point that falls below the first quartile minus 1.5 times or above the quartile plus 1.5 times.

And also performed a Pearson correlation for the price and rating columns.



```python
# correlation matrix for numeric columns
corr, p_value = stats.pearsonr(df['price'], df['rating'])
print(f"Pearson correlation between 'price' and 'rating': {corr}, p-value: {p_value}")
```

```
[6]
Pearson correlation between 'price' and 'rating': 0.030137094444841823, p-value: 0.3415597130301688
```

```python
# compare prices between top and bottom 10% rated books
top_10_percent = df['rating'].quantile(0.9)
bottom_10_percent = df['rating'].quantile(0.1)
top_rated = df[df['rating'] >= top_10_percent]['price']
bottom_rated = df[df['rating'] <= bottom_10_percent]['price']
t_stat, p_val = stats.ttest_ind(top_rated, bottom_rated, equal_var=False)
print(f"T-test between top and bottom 10% rated books' prices: t-statistic={t_stat}, p-value={p_val}")
```

```
[7]
T-test between top and bottom 10% rated books' prices: t-statistic=0.6384339581667415, p-value=0.5235563334739943
```

```python
# simple linear regression: predicting price based on rating
x = df[['rating']]
y = df['price']
model = LinearRegression()
model.fit(x, y)
print(f"Linear Regression: price = {model.intercept_}")
print(f"Coefficient for rating: {model.coef_[0]}")
print(f"R^2: {model.score(x, y)}")
```

```
[5]
Linear Regression: price = 34.16561350772499
Coefficient for rating: 0.3033702361392453
R^2: 0.0009082444615772234
```

Q2 - Data Analyzing

This notebook performs:

- **Predictive analysis** on `cleaned_books_data.csv` (Part C)

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neighbors import NearestNeighbors
[6]
```

```
df = pd.read_csv('G:\\My Drive\\MSC\\project\\question2_social_media_analysis\\data_processing\\cleaned_books_data.csv')
df
[7]
```

```
# Calculate RMSE
y_predict = model.predict(x)
print(f"RMSE : {np.sqrt(mean_squared_error(y,y_predict)):.3f}")
[8]
```

```
RMSE : 14.433
```

```
# Identify patterns in category and pricing
category_price = df.groupby('category')['price'].agg(['count','mean','median']).sort_values(by='mean', ascending=False)
print("Average price by category:")
display(category_price)
[9]
```

```
Average price by category:
```

50 rows × 3 cols — Static Output

category	count	mean	median
Suspense	1	58.330000	58.330
Novels	1	54.810000	54.810
Politics	3	53.613333	52.650
Health	4	51.452500	51.345
New Adult	6	46.383333	44.180
Christian	3	42.496667	47.720
Sports and Games	5	41.166000	44.840
Self Help	5	40.620000	46.350
Travel	11	39.796545	38.950

```
# recommend similar books based on price and rating using nearest nehibors model
features_df = df[['price', 'rating']].dropna()
from sklearn.neighbors import NearestNeighbors
nbrs = NearestNeighbors(n_neighbors=6).fit(features_df)
distances, indices = nbrs.kneighbors(features_df)
df['similar_books'] = indices.tolist()
print("Sample similar books based on price and rating:")
display(df[['title', 'similar_books']].head())
[14]
```

```
Sample similar books based on price and rating:
```

5 rows × 2 cols — Static Output

	title	similar_books
0	A Light in the Attic	[0, 904, 338, 274, 983, 841]
1	Tipping the Velvet	[1, 221, 708, 996, 898, 238]
2	Soumission	[2, 746, 774, 869, 886, 646]
3	Sharp Objects	[3, 665, 580, 407, 440, 290]
4	Sapiens: A Brief History of Humankind	[4, 884, 127, 768, 431, 463]

```
# Trends in availability vs price and rating
availability_trends = df.groupby('availability')[['price', 'rating']].mean()
print("Trends in availability vs price and rating:")
display(availability_trends)
[13]
```

Trends in availability vs price and rating:

21 rows × 2 cols — Static Output

availability	price	rating
In stock (1 available)	36.090928	3.051546
In stock (10 available)	37.140000	3.500000
In stock (11 available)	31.203571	2.928571
In stock (12 available)	38.500294	3.058824
In stock (13 available)	25.702000	2.200000
In stock (14 available)	35.342868	3.007353
In stock (15 available)	32.974070	2.872093
In stock (16 available)	35.311667	2.785714
In stock (17 available)	37.614000	3.400000
In stock (18 available)	30.272727	3.454545
In stock (19 available)	33.654783	2.956522
In stock (2 available)	30.350000	2.928571
In stock (20 available)	51.472500	2.750000

2.5.Data visualization.

The plots_visulization.ipynb performed and generated some visualizations for cleaned_books_data.csv. I used pandas, numpy, matplotlib, and seaborn libraries for perform this visualization.

In this notebook, I performed four key visualizations targeting some key business questions. All the plot generation follows the same pattern by following the pattern of defining figure size and finally saving each output to the directory.

Using seaborn.histplot function in this script performed the visualization of the distribution of the price column. And the seaborn.barplot was performed a horizontal bar diagram to visualize the top 10 b

ooks by categories.

Q2 - Data Visulization

This notebook performs:

- **Data Visulization** on `cleaned_books_data.csv` (Part C)

```
# Import necessary libraries
import ...
```
[2]

Code | Markdown | SQL

```
# load the dataset
df=pd.read_csv('G:\\My Drive\\MSC\\project\\question2_social_media_analysis\\data_processing\\cleaned_books_data.csv')
df
```
[4]

11 rows · 998 rows × 6 cols — Static Output

	title	price	rating	category	availability	description
0	A Light in the Attic	51.77	3	Poetry	In stock (22 available)	It's hard to imagine a wo
1	Tipping the Velvet	53.74	1	Historical Fiction	In stock (20 available)	"Erotic and absorbing...W
2	Soumission	50.10	1	Fiction	In stock (20 available)	Dans une France assez pro
3	Sharp Objects	47.82	4	Mystery	In stock (20 available)	WICKED above her hipbone,
4	Sapiens: A Brief History of Humankind	54.23	5	History	In stock (20 available)	From a renowned historian
...	...	...	...	...	...	...

```
# Price distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], bins=30, kde=True)
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.savefig('price_distribution.png')
plt.show()
```
[5]

```
# Price vs Rating scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='rating', y='price', data=df)
plt.title('Price vs Rating')
plt.xlabel('Rating')
plt.ylabel('Price')
plt.savefig('price_vs_rating.png')
plt.show()
```
[6]

```
# Price by rating box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='rating', y='price', data=df)
plt.title('Price by Rating')
plt.xlabel('Rating')
plt.ylabel('Price')
plt.savefig('price_by_rating.png')
plt.show()
```
[7]

Price by Rating

```python
# Average price by rating
avg_price_by_rating: pd.DataFrame = df.groupby('rating')['price'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='rating', y='price', data=avg_price_by_rating)
plt.title('Average Price by Rating')
plt.xlabel('Rating')
plt.ylabel('Average Price')
plt.savefig('avg_price_by_rating.png')
plt.show()
[8]
```

```python
# correlation heatmap
corr = df.select_dtypes(include=['number']).corr()

plt.figure(figsize=(10, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", cbar=True, square=True)
plt.title('Correlation Heatmap')
plt.savefig('correlation_heatmap.png')
plt.show()
[9]
```

```python
# Top 10 most expensive books
top_10_expensive = df.nlargest(10, 'price')
plt.figure(figsize=(12, 8))
sns.barplot(x='price', y='title', data=top_10_expensive, hue = None, palette='viridis', legend=False)
plt.title('Top 10 Most Expensive Books')
plt.xlabel('Price')
plt.ylabel('Title')
plt.savefig('top_10_expensive_books.png')
plt.show()
[17]
```

2.6. Outcome evaluation

Basic Statistics:

2 rows · 2 rows × 10 cols

	count	mean	std	min	25%	50%	75%	max	median	mode
price	998.0	35.052926	14.446311	10.0	22.1025	35.98	47.4075	59.99	35.98	16.28
rating	998.0	2.924850	1.435111	1.0	2.0000	3.00	4.0000	5.00	3.00	1.00

The above image shows a descriptive summary statistic table for price and rating, two numerical variables.

Both variables have 998 records. The mean of the price is 35.05 median of the price is 35.98 and the most frequent price is 16.28. We can assume the price distribution is not symmetric and there is some skewness.

Mean rating is 2.92, the median is 3.00, and the most frequent rating is 1.0, which is the minimum rating. This will be show some critical quality issues or negative customer satisfaction.

```
Outliers in 'price':0 records found.
Empty DataFrame
Columns: [title, price, rating, category, availability, description]
Index: []
```

The above image shows the output of the IQR analysis of the price. It shows there are no any data points identified as outliers. We can conclude that the price data is either very clean or heavily clustered.

```
Pearson correlation between 'price' and 'rating': 0.030137094444841823, p-value: 0.3415597130301688
```

The above image shows the correlation analysis for the price and the rating columns. The Pearson correlation between price and the rating is 0.0301. which is very close to the 0. That means an extreme weak positive linear relationship between pricing and rating. The book price increases, the rating will slightly increase.

```
T-test between top and bottom 10% rated books' prices: t-statistic=0.6384339581667415, p-value=0.5235563334739943
```

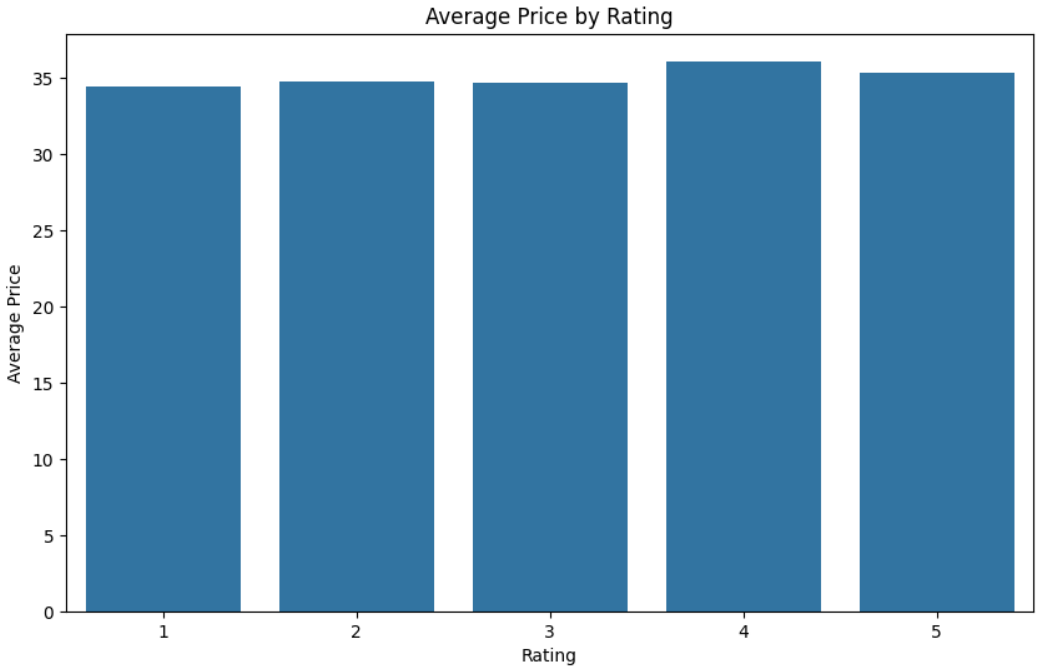
The above output shows the result of a T-test comparing the mean price of books in the top 10 rated and the bottom 10% rated group.

```
Linear Regression: price = 34.16561350772499
Coefficient for rating: 0.3033702361392453
R^2: 0.0009082444615772234
```

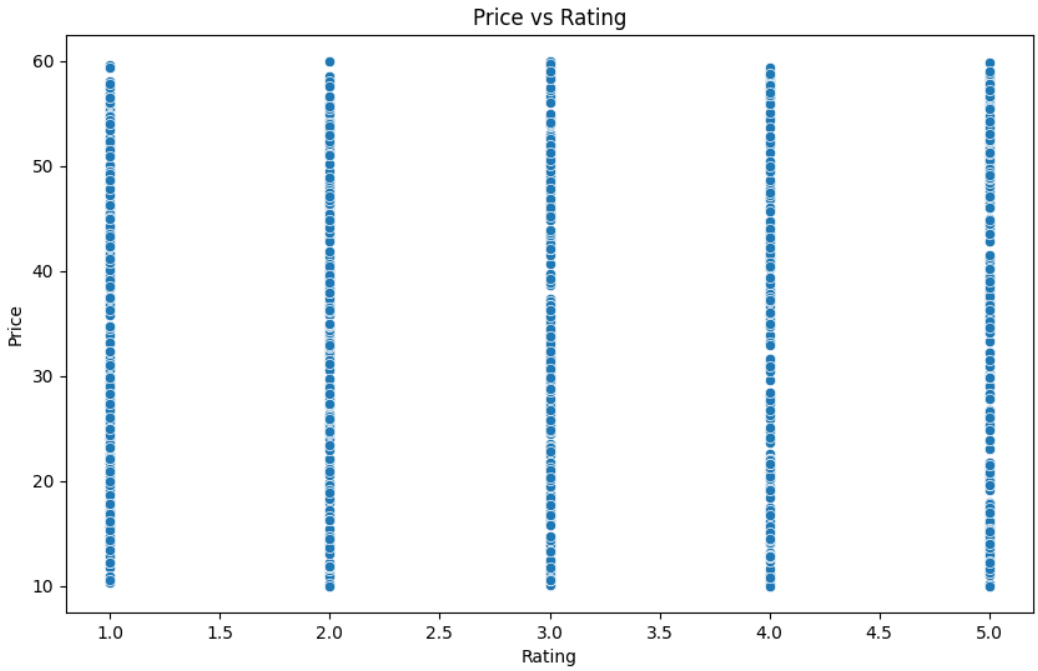
The above image shows, result of the simple linear regression model. We used rating as the independent variable and price as the dependent variable.

$$Price = 34.165 + (0.3033 * Rating)$$

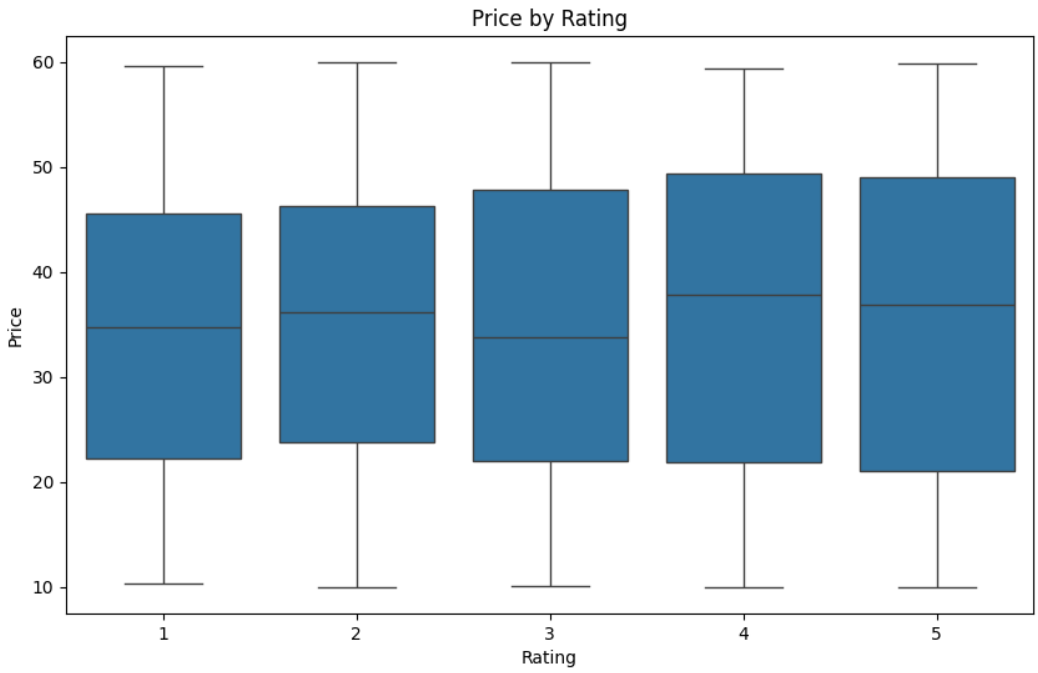
According to the above linear regression equation, when the rating is equal to 0 price of the book is 34.1656. And if the rating increased by one unit price increased by 0.3033.



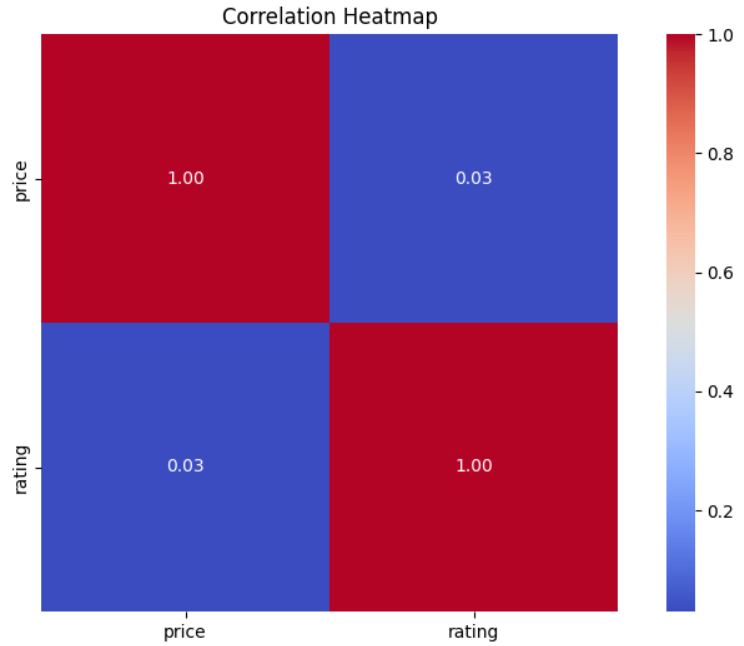
The above bar plot visualizes the average price by rating. The X-axis represents the rating, and y-axis represents the average price.



This is the scatterplot of the price vs rating. It visualizes the relationship between price and rating. According to the above scatterplot, we cannot see any correlation. No any linear relationship between price and rating.



This image shows the bar plot of the price by rating. Each bar shows the rating. The height of the bar represents the average price for books. There is no significant trend increasing or decreasing as the rating moves from 1.0 to 5.0.



This image shows a correlation heatmap for the pricing and rating. We can conclude that the book price does not linear relationship with the rating.

3. Data Ethics: AI Ethics in Healthcare Data

3.1. Introduction

Artificial intelligence and data science are affecting every sector in the modern world. Even the health sector could not survive. Disease diagnosis, efficient treatment plans, forecasting, and predictive processes to identify outbreaks. In addition to that, these have the potential to improve living standards and save lives worldwide.

This means that AI would be developed to serve the society and the ethical principles of equity, transparency and confidentiality in the patient's treatment will be observed. This means that AI would be developed to serve the society and the ethical principles of equity, transparency and confidentiality in the patient's treatment will be observed.

3.2. Healthcare Data Privacy Challenges

As much similar to the complexities of the GDPR, the ethical use of AI along data arcs within the healthcare domain still faces more complicated obstacles like answering the question of how to appropriately anonymize personal health data at scale, where such anonymization technologies offer a reasonable assurance of privacy, the balancing act of how to protect patient information while still being able to use patient data for research, or the compliance with particular regulations that govern health information in the country such as the Health Insurance Portability and Accountability Act (HIPAA) within the United State of America.

The United States of America has established the Health Insurance Portability and Accountability Act, also called as HIPAA, which operates with far greater precision than other U.S. laws because it establishes and implements state-specific regulations on patient data protection, followed by the general data regulation doctrine of the HIPAA systems. It establishes the more complex rules for protected health information, or PHI, and goes far beyond other US laws. A broader and ambiguous description of privacy and information sharing is given by the HIPAA law, which applies to business associates, covered entities, clinics, hospitals, secondary data users, billing experts, and IT consulting firms. There is an even greater chance that it will be revealed given the proliferation of restricted data in the Blinkers and Big Data.

Data protection laws vary around the world. As an example, Canada's Personal Information Protection and Electronic Documents Act - PIPEDA and its provincial equivalents both address

unit data protection and electronic documents, while Canada's specific health laws for Australia include the "Privacy Act," which lays out the "Australian Privacy Principles" (APPs) on health information. International cooperation and the advancement of global AI for the healthcare sector were hampered by these laws.

World must come up with strategies for efficiently optimised data use for research without sacrificing data security. Making medical data anonymous is extremely challenging. Although methods like differential privacy and k-anonymity are widely used in the field, they can deprive the data of much of its accuracy and value for use in research and AI training.

3.3. Algorithmic Bias in Medical AI

In the medical field, biases in algorithms and AI models create a serious ethical problem and have the potential to maintain or worsen already-existing inequalities in healthcare.

The data that is used to train these models is the main source of these biases. Bias in medical datasets may occur many different and hidden causes, including social, economic, geographic, and other social and cultural factors. Those are called data driven biases. As an example, most of the research are happened in Western region countries. Most of the time, social, economic, and cultural status are totally different when compared to other regions. Mostly available datasets are not suitable for other regional contexts due to this reason.

Algorithms are can also be biased. Those are mainly happened in the development phase by the machine learning engineer. When the ML engineer trains an algorithm or a model on biased data, this is likely to identify and reinforce patterns and trends from the dominant category in the data that we used to train. As an example, the AI model might not identify a similar condition on darker skin tones if a dataset contains more photos of skin conditions that appear on lighter skin tones.

Human bias is can also happen in the development phase. And data gaps may also lead to some bias in algorithms as well.

Those kinds of bias can lead to biased insights and decision making. Misdiagnose some diseases, allocate medicines or other resources incorrectly. And also, these kinds of biased algorithms tend to lead to misinterpretation of disease identification and wrong treatment suggestions.

We can decrease those kinds of issues that are happened due to biased decisions making promoted by biased algorithms and models by introducing some strategies.

1. Data Centric Strategies.
    a. Make sure that the training datasets include a wide range of demographic groups and scenarios to prevent favoritism to one group.
    b. When data processing, make sure to address data imbalances before training the model by using reweighting or relabeling techniques.
    c. Implement standard processes to make sure that the high accuracy and suitability of the input data.
2. Development phase strategies
    a. There should be human interaction to provide review and direction to the process to mitigate delicate misunderstandings and contexts that AI models may miss.
    b. Include various backgrounds and expertise in to the development team to get different views.
    c. Include a representative from the not represented or underrepresented population.
3. Governance strategies.
    a. Make velar and understandable decision making algorithms of the AI models. This will increase the transparency and help to identify and correct any bias that occurs.
    b. Setting some specific benchmarks to detect discrepancies in outcomes after deployment and evaluate continuously.
    c. Establish a mechanism to users to provide feedback, evaluate those feedbacks, and make the improvements accordingly

3.4.Ethical Decision-Making Framework

When dealing with ethical issues in data science and artificial intelligence applications is very crucial. This cruciality is high when it comes to the healthcare sector. It is important to having real world, practical ethical decision making framework to address this. It’s also helpful to machine learning engineers and data scientists to proceed and evaluate their projects in an ethical way.

There are four core principles that will merge AI decision making in the healthcare sector.

1. **Autonomy**: AI should only support to patients to make their own healthcare decisions. AI only provides suitable information to patients to support their decision making. Not to override.
2. **Beneficence:** AI models should support improving the diagnostic procedure of patient disease and increase the overall efficiency of healthcare.
3. **Non-maleficence:** AI models should avoid causing harm through errors and biases.
4. **Justice:** AI models need to promote equality and justice while making sure they don't discriminate against people on the basis of economic status, gender, or race.

To ensure the above core principles are met, we need some actionable steps.

1. Use diversification and well-representative and accurate datasets.
2. Audit datasets as well as algorithms continuously to avoid biases.
3. Ensure transparency and understandability
4. Remain in human interaction and review sessions with healthcare professionals.
5. Maintain data security and privacy.
6. Design AI architecture promoting equity.

The “right to explanation” is one of the crucial concepts in this framework. Not only this healthcare framework, but also for any AI system. Mainly, high risk AI models like health care models, to ensure accountability and transparency, and understandability. This concept represents the right to get a meaningful explanation for a decision made by models. When we talk about the healthcare sector, patients should be empowered and should have a clear idea of how the AI models generate specific recommendations. It also helps to increase patients’ confidence regarding AI tools. As an example, some predictive analytics models can predict an outbreak of the disease. This could lead to social anxiety and can cause unwanted favoritism for businesses in the healthcare sector.

3.5.Stakeholder Impact Analysis

There are two main stakeholders in the healthcare sector.

The main stakeholder is the patient. AI may offer a number of added advantages to patients. AI can recommend personalized treatments that are suitable for their diseases and overall efficient in personal health management. But in the meantime patient’s privacy may be at risk. Biases that

lead to misdiagnosis. And also, sometimes a patient may not be satisfied with the quality of the service.

Empowering the patient through out informed and transparent communication process is the solution for this.

Doctors and nurses are the other stakeholders. Doctors and nurses are also getting major benefits from AI tools in the healthcare sector. They can diagnose diseases efficiently and plan accurate treatments for critical diseases, and improve overall efficiency. But sometimes professionals may see AI as a threat because of the job uncertainty.

We should not replace professionals with AI applications, instead of replacing we must use AI as an assistance technology.

Data Scientists are one of the important roles when we talking about the developing ethical healthcare AI models. They are responsible for delivering transparent, fair, and secure AI models to the society. Data scientists should be skillful not only in data, mathematical, and technical skills, but also in data ethics, data privacy law, and avoiding bias in algorithms. And also make sure developments are in line with the ethical framework, and promote transparency and accountability. Those actions will ensure the responsible and ethical deployment of AI in the healthcare sector by any data scientist.

Both social and economic effects are significant in the healthcare sector. Using AI in the healthcare sector sometimes reduces the costs of health benefits and increases accessibility in some rural areas in world. Sometimes this may increase the gap between people who have access to advanced technology and those who don’t. This will worsen global health equity.

When developing AI tools, developers should consider following the global health equity guidelines. This will ensure that those technologies are accessible to all.

As a conclusion, while not simple, the use of AI in healthcare is possible. Making sure it functions properly will now be crucial, which is why it is necessary to go beyond following the law to protect patient data, fix algorithms that are inappropriate and result in unfair health outcomes, and develop sound ethical standards for AI decision-making. It will eventually take a concentrated effort across

multiple dimensions aimed at improving staff support, patient support, and universally equitable healthcare to make AI a useful technology in the healthcare sector.

4. GitHub Repository

https://github.com/goyumsamuditha/project.git

5. References

5.1. **Article 29 Working Party** (2017) *Guidelines on Automated individual decision-making and Profiling for the purposes of Regulation 2016/679 (GDPR)*. Available at: [Insert URL/Document Source for Right to Explanation] (Accessed: Day Month Year).

5.2. **Council of Europe** (1981) *Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data*. Strasbourg: Council of Europe.

5.3. **European Commission** (2016) *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation)*. Available at: [Insert URL/Document Source for GDPR].

5.4. **Fuster, G.G.** (2014) *The right to the explanation of automated decisions under the GDPR. International Data Privacy Law*, 4(1), pp. 75–88.

5.5. **Ghassemi, M., Naumann, T., D'Amour, A., et al.** (2020) 'A comparison of clinical and algorithmic bias in medical decision making', *The Lancet Digital Health*, 2(11), pp. e557-e565.

5.6. **HIPAA** (1996) *Health Insurance Portability and Accountability Act*. Public Law 104-191, 42 U.S.C. § 1320d et seq. [Specific CFR citations for Privacy and Security Rules would also be relevant].

5.7. **Organisation for Economic Co-operation and Development (OECD)** (2017) *Recommendation of the Council on Health Data Governance*. Paris: OECD Publishing.

5.8. **Privacy Commissioner of Canada** (2000) *Personal Information Protection and Electronic Documents Act (PIPEDA)*. [Insert relevant statute details].