COURSE WORK: PROGRAMMING FOR DATA SCIENCE 2025.2 BATCH

MSc Data Science: Coventry University UK

Programming for Data Science

Coursework 01 Technical Report

Student Name: K. J. U. Perera

Student Id: COMSCDS252P-008

Contents

1. Executive Summary

The content of the document is organized under three sections. In the first section, it focuses on Object-Oriented Programming (OOP) using Python programming language through developing a simplified University Management System (UMS). When developing the system, it follows OOP concepts for demonstrating proper class hierarchy, inheritance, encapsulation, and polymorphism within a structured program. Furthermore, it considers sound coding practice, including modular design, clear naming convention, and Git branching with repository maintenance.

The second section conducts the data analysis workflow based on over three hundred records. The workflow follows scraping the data from a public website, cleaning and preprocessing data, statistical analysis including descriptive statistics & inferential statistics, visualizing data, and a basic predictive model. The system uses Python and its analytical libraries to demonstrate these processes and transform data into structured information suitable for analysis and interpretation.

In the third section, the report explores Artificial Intelligence (AI) ethics in healthcare field. It examines following aspects of data privacy and algorithmic bias in medical AI, structured approaches to ethical decision-making, and the impact of AI systems on different stakeholders.

1.1. Git Repository Access

Public Git repository: https://github.com/JanakaUPerera/prds_cw01_comscds252p008.git

2. Object-Oriented Programming (OOP) concept implementation

2.1. Architecture and Class Hierarchy

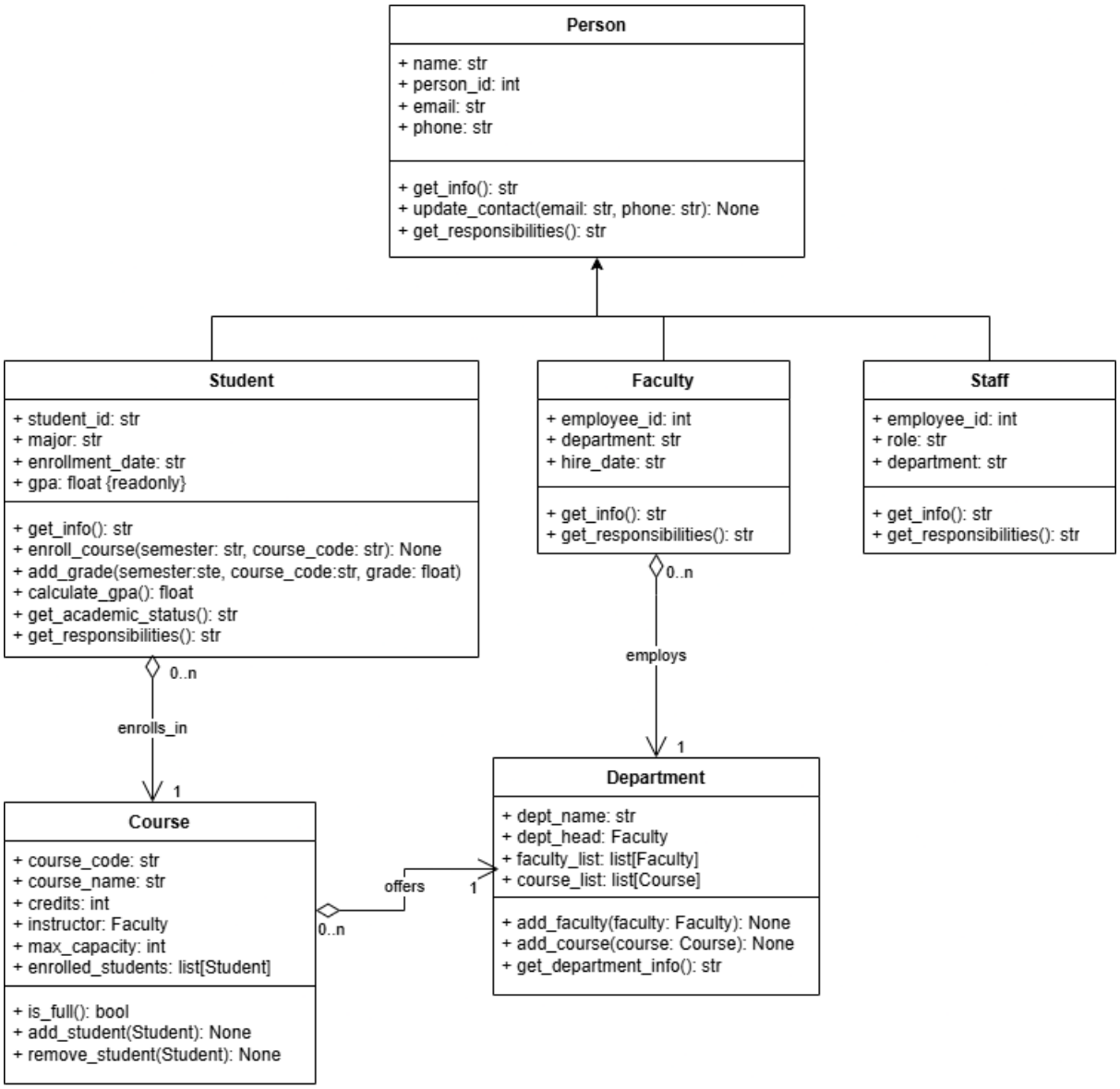


Figure 2.1.1: Architecture and class hierarchy

Figure 2.1.1 presents the system architecture and class hierarchy, illustrating the OOP core concepts adopted in the UMS.

The **`Person`** class appears at the top of the hierarchy. It defines shared `name`, `person_id`, `email`, and `phone` attributes. The attributes represent such as state, characteristics, properties or data of the person. It declares common operations for getting

information, updating contact details and getting responsibilities. The methods, `get_info()`, `update_contact()` and `get_responsibilities()` represent the behaviors of the person.

Three subclasses **`Student`**, **`Faculty`** and **`Staff`** inherit from **`Person`**. This inheritance correctly models an "is-a" relationship. A student is a person. A faculty member is a person. A staff member is a person. Therefore, these three classes has their parent class behaviors, state, characteristics, properties or data. This is like, a real-world person having abilities and appearance from their parents.

The **`Student`** class contains attributes of `student_id`, `major`, `enrollment_date`, and read-only `gpa`. It defines operations for enrolling in courses, adding grades, calculating Grade Point Average (GPA), and checking academic status. The presence of `calculate_gpa()`, `enroll_course()`, `add_grade()`, `calculate_gpa()` methods and `gpa` attribute derive GPA by storing data and calculating them. This approach aligns with good design practice and shows encapsulation behavior.

The **`Faculty`** class and the **`Staff`** class has its own attributes and inherited attributes from its parent class, which maintain structural consistency. They override `get_info()` and `get_responsibilities()`methods. This demonstrates polymorphism behavior.

The **`Course`** class contains its own attributes and a list of enrolled students. It controls course capacity and manipulating registration logic inside the course object using `is_full()`, `add_student()` and `remove_student()`methods.

The **`Department`** class stores `dept_name`, `dept_head` and aggregates faculty and courses using `faculty_list` and `course_list`. Its methods, `add_faculty()` and `add_course()` methods manipulate the department.

Furthermore the Figure 2.1.1 shows the relationships among each class. It describe one department can contain many faculty members and many courses. Each faculty member belongs to one department and each course belongs to one department. Regarding the Student class and Course class, a student can enroll in many courses and a course contains many students. It follows the many-to-many relationship.

2.2. Design Decisions

2.2.1. Single Responsibility Principle

Each class bears identity responsibilities and its methods perform the specified responsibilities. The **`Student`** class manipulates the works related to GPA and the enrollment process. The `enroll_course()` function performs enrolling of students in a given semester and course. The `add_grade()` function performs adding of grades in a course in the semester. The `calculate_gpa()` performs calculating of GPA.

The code base shows separation of modules based on purpose. Folder structure represents university system, data analysis, and test modules.

2.2.2. Validation inside Methods

The enroll_course() method validates the maximum number of courses that are possible to enroll in.

```
69    def enroll_course(self, semester: str, course_code: str) -> None:      You, 6 days ago
70        if semester not in self.enrolled_courses:
71            self.enrolled_courses[semester] = []
72
73        if len(self.enrolled_courses[semester]) >= self.MAX_COURSES_PER_SEMESTER:
74            raise ValueError(f"Cannot enroll: The maximum number of courses for a semester
75
76        if course_code not in self.enrolled_courses[semester]:
77            self.enrolled_courses[semester].append(course_code)
```

Figure 2.2.2.1: Validate maximum number of courses that are possible enroll in

Figure 2.2.1 shows validation and will raise a `ValueError` if validation fails.

2.2.3. Error Handling in Application Layer

Exceptions are raised in classes' methods but handled in `main.py`, following best practice separation of concerns.

```
main.py
question1_university_system > main.py > Main > run
 7    class Main:
11        def run(self):
76            try:
77                student.enroll_course(enroll_semester, 'CS106')
78                student.enroll_course(enroll_semester, 'CS107') # Exceeding course limit
79            except ValueError as e:
80                print(f"Error: {e}")
```

Figure 2.2.3.1: Exception handle in main.py

2.3. Key OOP Features Demonstration

2.3.1. Inheritance

`Student`, **`Faculty`** and **`Staff`** classes inherit shared `name`, `person_id`, `email`, `phone` attributes from **`Person`** class. It enables code reusability, improves maintainability and reduce redundancy.

```
14  class Person:       You, 5 days ago • feat: Add person base class
15      def __init__(self,
16                   name: str,
17                   person_id: int,
18                   email: str,
19                   phone: str
20                   ) -> None:
21          # Validate initial data
22          # Validate email
23          email_pattern = r"^[\w\.-]+@[\w\.-]+\.\w+$"
24  >       if not re.match(email_pattern, email):···
26          # Phone number should be digit and length should be greater than or equal 9
27  >       if not phone.isdigit() or len(phone) < 9:···
29
30          self.name = name
31          self.person_id = person_id
32          self.email = email
33          self.phone = phone
```

Figure 2.3.1.1: Person class shares attributes

```python
class Student(Person):
    MAX_COURSES_PER_SEMESTER = 6

    def __init__(self,
                 name: str,
                 person_id: int,
                 email: str,
                 phone: str,
                 student_id: str,
                 major: str,
                 enrollment_date: str):

        # Validate initial data
        # Validate enrollment date format
        try: ...
        except ValueError: ...

        super().__init__(name, person_id, email, phone)
        self.student_id = student_id
        self.major = major
        self.enrollment_date = enrollment_date
```

Figure 2.3.1.2: Student class inherits shared attributes from Person class

2.3.2. Encapsulation

Using the encapsulation OOP concept, restrict the direct access to some of object's components. The UMS uses `@property` for making GPA read-only. It hides data from external classes, controls access to data, and to interact with data, uses getter and setter methods. `@property` represents the getter method of GPA.

```python
    @property
    def gpa(self) -> float:
        return self.calculate_gpa()
```

Figure 2.3.2.1: GPA as a read-only property

2.3.3. Polymorphism

`Student`, **`Faculty`**, and **`Staff`** classes override the `get_responsibilities()` method. It allows dynamic behaviors, and enables flexibility and reusability. Derived classes can be handled without significant modification of the method through this concept.

```
94      # Demonstrate Polymorphism
95      print("\n--- Polymorphism Demonstration ---")
96      persons = [students[0], faculty_members[2], staff_members[4]]
97      for person in persons:
98          print(f"\n{person.__class__.__name__}:\n{'--'*30}\n")
99          print(f"{person.name}: {person.get_responsibilities()}")
```

Figure 2.3.3.1: Demonstrate polymorphism

3. Data Analysis

3.1. Methodology and Tools

The analysis follows a structured data science pipeline. It is a series of works of collecting data from sources, preprocessing/cleaning, feature engineering, Exploratory Data Analysis (EDA), Statistical Analysis, model training, evaluation and visualizing.

3.1.1. Collecting Data from Sources

More than three hundred data records are collected from the Books to Scrape website using an automated web scraping mechanism implemented with the `requests` library and `BeautifulSoup4` parsing framework [1]-[3]. It gathers data of the title, price, rating, availability, and category, then saves them to a CSV file. The mechanism introduces a delay of one-two seconds between HTTP requests to regulate request frequency and avoids server overload. It implements a retry option with a maximum of three attempts to handle transient network failures and temporary server unavailability. For handling runtime crashes and ensuring uninterrupted executions, it uses structured `try-except` mechanism.

3.1.2. Preprocessing/Cleaning

The data processing pipeline uses the `pandas` library to ensure data quality, consistency, and analytical reliability [4]. The cleaning process removes the "£" currency symbol from price values and converts them into floating-point format, transforms textual rating labels One-Five to the corresponding numeral 1-5 values, handles missing values using "Unknown" as default value for textual data and Median for numeric data to maintain the statistical stability, and removes duplicate records to maintain datasets integrity.

3.1.3. Feature Engineering

It derives the `price_category` variable based on prices and divides them into budget, mid-range, and premium categories, and the `in_stock` variable represents a Boolean indicator based on availability status.

3.1.4. Exploratory Data Analysis

The mechanism calculates mean, median, standard deviation, and correlation matrices using `Scipy` and `pandas` libraries [5].

3.1.5. Statistical Analysis

Under the Descriptive Statistics, it calculates range, average top 5 price by category and rating distribution for summarizing, organizing, and characterizing the data using metrics like mean, median, mode, and standard deviation.

Inferential Statistics makes predictions or conclusions about a larger population based on a smaller sample by involving hypothesis testing and estimation.

`Scipy` and `pandas` libraries helps to do statistical analysis in here.

3.1.6. Model Training & Evaluation

Model training involves teaching an algorithm to identify patterns in data for making accurate decisions efficiently by performing on unseen data.

In here, the `scikit-learn` library helps to identify which feature from Rating and Category, has stronger influence on price by using linear-regression model [6]. Evaluation metrics are R^2 score and Mean Absolute Error (MAE)

3.1.7. Visualization

The `Plotly` library helps to create of graphical representations, such as graphs, charts, and maps for communicating, exploring, and analyzing data [7].

3.2. Key Findings from Exploratory Analysis

3.2.1. Price Distribution and Central Tendency

```
-- Central tendency: Mean, Median, Mode for Price --
Mean:           35.04
Median:         36.65
Mode:           13.34
```

Figure 3.2.1.1: Analysis result mean, median, mode

The average price is 35.04, while the median is 36.65. The closeness of the mean and median describes the distribution as relatively symmetrical. The median is greater than the mean represents a distribution that is mildly left-skewed.

The mode is 13.34 that lower than both of mean and median, shows low-cost books are more popular according to dataset.

3.2.2. Price Variability

```
-- Dispersion: standard deviation, range --
STD:            14.55
Range:          49.74
```

Figure 3.2.2.1: Analysis result STD and Range

The standard deviation of 14.55 indicates a medium spread-out of data from the mean value. The price range shows that the book prices vary significantly through the entire dataset.

```
-- Outlier detection: Use IQR method for price outliers --
Q1:                             21.86
Q3:                             47.74
IQR:                            25.89
Lower Bound:                    -16.97
Upper Bound:                    86.58
Number of Outliers:             0
```

Figure 3.2.2.1: Analysis result Interquartile Range (IQR)

The interquartile range detects no price outlier and all prices keeps within -16.97 and 86.58. It indicates a stable pricing structure across the dataset.

3.2.3. Category-Based Price Differences

```
-- Group statistics: average price by category (top 5) --
         category  price
0         Fiction  36.29
1   Add a comment  36.01
2         Default  35.90
3  Sequential Art  33.69
4      Nonfiction  31.72
```

Figure 3.2.3.1: Analysis result category based average price

This result indicates fiction is the most expensive category. However, the top 5 categories do not show a significant difference of the mean.

3.2.4. Rating Distribution

```
-- Rating distribution: frequency count --
   rating  count
0       1    110
1       2    102
2       3     99
3       4     92
4       5     97
```

Figure 3.2.4.1: Analysis result rating distribution

The distribution of rating indicates fair diversity among all rates. It represents customers have evaluated the books without special bias

3.2.5. Relationship between Price and Rating

```
-- Correlation analysis: Pearson correlation between price and rating --
Pearson Correlation Coefficient (r):    0.004
P-value:                                0.9251
Conclusion: No statistically significant correlation.
```

Figure 3.2.5.1: Analysis result Pearson correlation Coefficient

The Pearson Correlation Coefficient (r) of 0.004 with a P-value of 0.9251. This has no significant relationship. This means higher-priced books do not get higher rates and vice versa.

3.2.6. Hypothesis Testing: Fiction vs Nonfiction

```
-- Hypothesis testing - Compare average prices between Fiction vs Non-Fiction --
Fiction Mean Price:     36.29
Non-Fiction Mean Price: 31.72
T-statistic:            1.457
P-value:                0.1500
Conclusion: No significant difference in average prices.
```

Figure 3.2.6.1: Analysis result hypothesis testing

Comparing average prices between Fiction and Non-Fiction produces a t-value of 1.457. Since the P-value of 0.15 is greater than the significant value of 0.05, it indicates no statistically significant difference between the two categories.

3.3. Predictive Analysis

The linear regression model outputs show weak predictive capability. The R^2 value of -0.145 indicates that the model performs worst on predicting the average price for all books. Therefore, it suggests only rating and category are not sufficient predictors of price fluctuation.

The MAE of 11.867 confirms predictions deviate from actual prices. Regression coefficients reveal that categories influence prices more strongly than rating. Categories such as Crime (value of -26.47), Erotica (value of -18.15), and Biography (value of -16.27) reduce price, while such as Suspense (value of 20.94), and Women's Fiction (value of 19.99) increase price. The model does not include the factors beyond rating and categories that affect the price.

3.4. Statistical Analysis with Visualization

3.4.1. Price Distribution (Histogram with Mean Line)

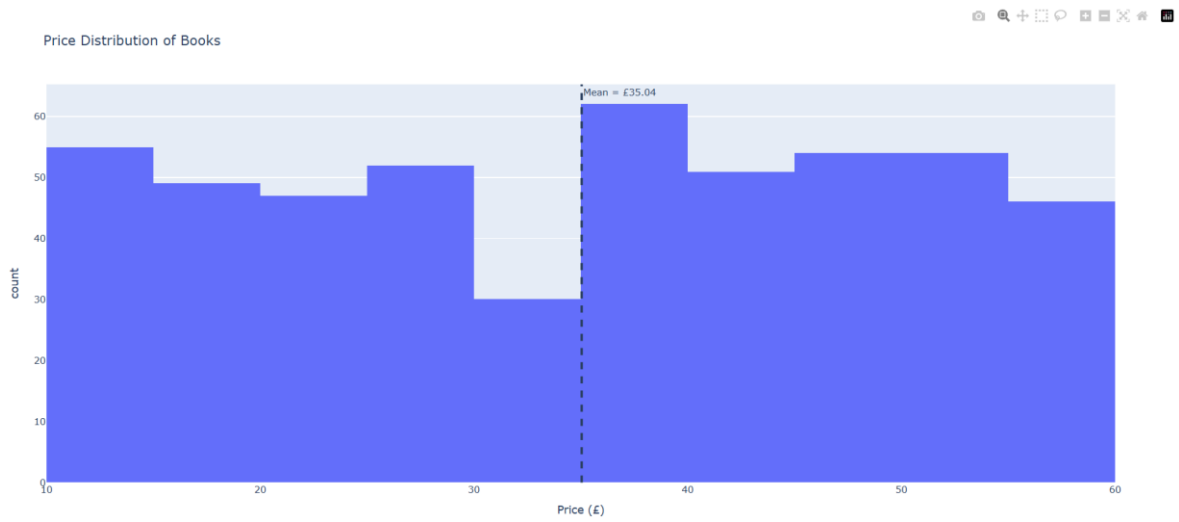


Figure 3.3.1.1: Histogram - Price distribution

This histogram shows prices are spread out between 10 and 60 fairly and evenly with a mean of 35.04. The distribution is not skewed to any side. It indicates the pricing stability of the dataset.

3.4.2. Price Comparison across Categories (Box Plot)

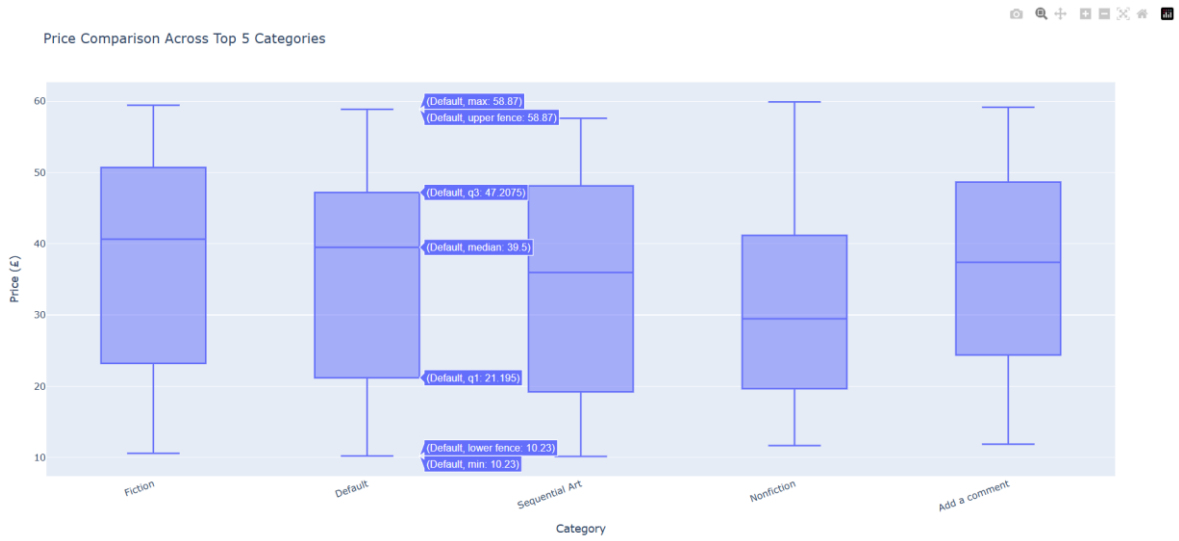


Figure 3.3.2.1: Box Plot - Price Comparison across categories

The box plot shows price distribution of the top five categories. It confirms the p-value of 0.150 is greater than significant value of 0.0f, and indicates that although minor differences exist between categories. Therefore, there is no statistically significant difference in mean prices of categories. So it shows there is no significant price variance across categories.

3.4.3. Price vs Rating (Scatter Plot with Regression Line)

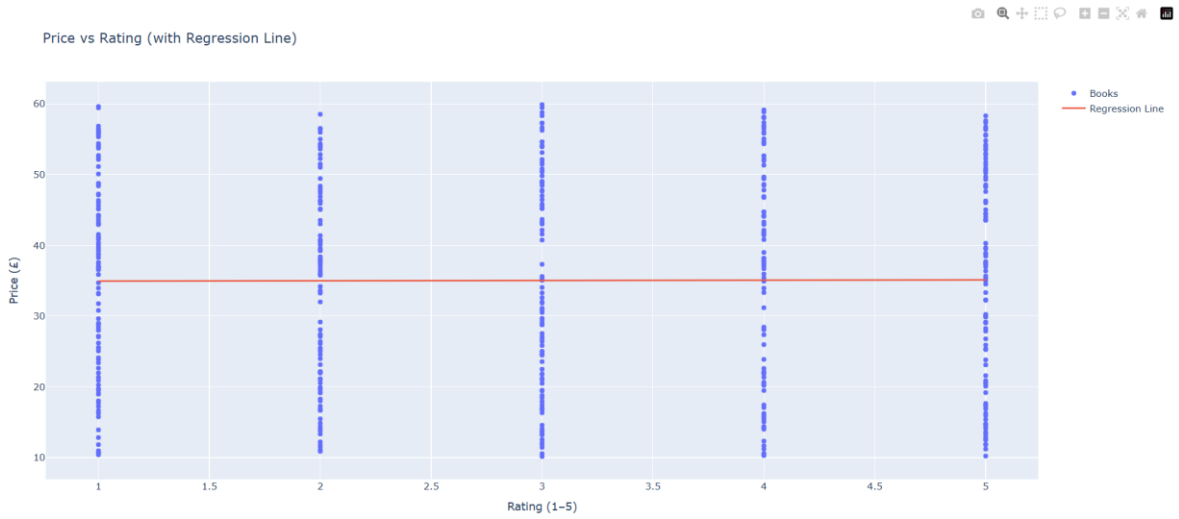
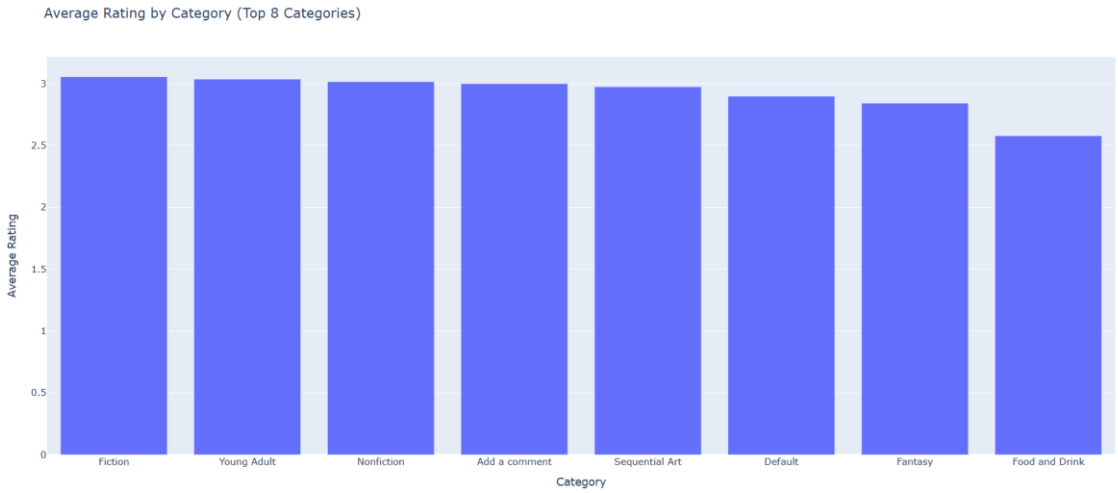


Figure 3.3.3.1: Scatter Plot - Price vs Rate

This scatter plot chart shows points are widely spread vertically for each rating level and the regression line is very closes to horizontal. Therefore, it supports the Pearson

Correlation Coefficient (r) of 0.004 with a P-value of 0.9251 and demonstrates no significant relationship between price and rate.

3.4.4. Average Rating by Category (Bar Chart)



The bar chart shows average prices across the top 8 categories that closely clustered around 3.0 with a minor difference of 0.5, indicating no dominant category.

3.5. Business Insight

Considering key findings of analysis and statistical results,

- ✓ The correlation coefficient of rate and price suggests that pricing can be optimize according to demand and market position rather than the rate.
- ✓ Identification of no price outliers using IQR method, reflects a standardize price structure rather than aggressive price or extreme discount.
- ✓ Evenly distributed ratings across all levels, suggesting gaining an advantage through improved quality of product and service.
- ✓ Average ratings cluster around 3.0, reflecting that marketing efforts should focus on specific high-performance items rather than categories.
- ✓ Prices vary moderately with a standard deviation and range, suggests to allow businesses to target different customer tiers, such as budget, mid-range, and premium, and use strategy bundling or subscriptions to boost sales and revenue.

4. Data Ethics: AI Ethics in Healthcare

4.1. Healthcare Data Privacy

Healthcare data privacy requires special and strict protection because medical records contain such things as biometric identifiers, genetic information, diagnostic history, treatment plans, mental status, and extremely personal diseases. Once someone's medical data is revealed, it can't be undone or changed.

For examples

Example 01: If cyber attackers stole records of medical services and prescription drugs, they can make fake insurance claims in the victim's name. This is a financial loss for insurance company and bad effect on victim's dignity.

Example 02: If expose highly sensitive information such as mental health records, treatments for extreme personal diseases like social diseases can be used to blackmail the patient.

Therefore, the keeping and maintenance of healthcare should be regulated under proper and strict regulations. This responsibilities apply uniformly across both of private and public sectors including hospitals, clinics, any kind of healthcare service providers, pharmacies, diagnostic laboratories, and pharmaceutical e-commerce platforms.

4.1.1. Health Insurance Portability and Accountability Act (HIPAA)

HIPAA governs the Protected Health Information (PHI) in the United States of America (USA). PHI is any information of a patient's any identity and medical history keep or transmitted in any form verbal, paper, or electronic [8],[9]. PHI specifically contains;

- ✓ **Personal Identifiers**: Such things as Name, Address, Social Security Number, and date of birth
- ✓ **Health Status**: Information about physical and mental health conditions past, present, or future.
- ✓ **Care and Payment**: Details about provided care to a patient and the past, present, and future payment for the providing care.

PHI defines requirements for providers under major three categories

Administrative

This includes building policies, procedure, and protocols, analyzing security risk about data and provide appropriate solutions, assigning a specific employee to oversee the adaption and adhere privacy procedures, protocols and policies, arranging training sessions for acknowledging procedures, protocols, and policies and ensure the employee compliance.

Physical

Providers should maintain proper infrastructure to keep records securely by upgrading software and hardware technologies, maintaining proper server room, considering devices protection and following cybersecurity approaches.

Technical

Providers should have a proper access control method to access records. Access controls define who has availability to access data and monitor who accesses records. When keeping sensitive data, should follows encryption method. Constantly need maintain authentication system and keep audit logs and secure transmission protocols.

4.1.2. General Data Protection Regulation (GDPR)

GDPR defines rules, privacy, and protection of healthcare data under the special section “Personal Data Protection" [10], [11]. These regulations are regulated by the European Union. Healthcare data is categorize as sanative personal data. This category includes;

- ✓ **Data concerning health:** This contains personal data related to physical and mental health and including provided healthcare services to a person and records of medical status.
- ✓ **Genetic Data:** Data derived from biological samples are consider as genetic data
- ✓ **Biometric Data:** The data that want to do a technical process to get results are considered biometric data, such things as facial recognition and fingerprints. These data help to do unique identification.

Due to these classifications, they prohibits the processing healthcare data by default but only permitted under these conditions.

- If the patient clearly know and allows to collect, store, and share the health information.
- To do medical necessity things, such as diagnosis treatment, preventive treatment, and occupational health and healthcare system management.

- Insure protection against from cross-border health threat and medical products and healthcare services for improving the believability and interest of public.

Data Subject Rights in Healthcare Sector

GDPR grants rights for data aspect which are impact in healthcare system.

- ✓ **Right of Access:** Patient have the right to access their personal health data and acknowledge how it is processed
- ✓ **Right to Data Portability:** Enables patients to transfer their data between health service providers for improving consistency of care and patient independence.
- ✓ **Right to Be Forgotten:** Allows patients to request termination of keeping and processing their personal health data.

This is the most complex principle in GDPR that executing in healthcare environment. Due to requesting termination, providers face legal and ethical limitations including;

- Mandatory medical record-keeping laws
- Responsibility and bond with public health
- Clinical accountability and fulfilling and maintaining medical-legal requirements.
- Necessary for preserving research and epidemiological data

This creates a conflict between individual privacy rights and social health responsibilities, making impractical status in many healthcare sectors. As a result, healthcare providers have to maintain a controlled deletion system and restricted processing for achieving legal mechanisms.

Governance, Accountability and Risk Management

GDPR forces healthcare providers to mandate a Data Protection Officer (DPO) for ensuring regulatory compliance of processing sensitive data. Furthermore providers should conducts Data Protection Impact Assessments (DPIAs) to evaluate risks associated with the data processing activities. Healthcare institutions should legally follows;

- ✓ Keep detailed records of processing activities
- ✓ Set up Data Processing Agreements (DPAs)

- ✓ Implement data breach detection mechanisms and response procedure
- ✓ Report data breaches within 72 hours to authorized parties and effected individuals

Due to outdated Information Technology (IT) infrastructures, and weak cyber security approaches, the healthcare sector become a primary target of cyber-attacks. Therefore GDPR requires sophisticated solutions including;

- ✓ Well organized access control system
- ✓ Encryption mechanism for keeping sensitive data
- ✓ Secure cloud and server infrastructures
- ✓ Continues staff training and arranges incident response simulations.

4.1.3. Anonymization Challenges

For anonymization to prevent breaching data, healthcare institutions have to ignore identifiers such as names, addresses, and ID numbers. However healthcare data can be re-identified through;

- ✓ Linking datasets among gathered datasets from several data sources
- ✓ Quasi-Identifiers such as Age, Gender, Zip Codes, Diagnosis Patterns
- ✓ Machine learning assumptions techniques
- ✓ Biometric and behavioral patterns

Therefore simple de-identification is not sufficient for true privacy protection. Then healthcare institutions follows;

- ✓ Differential privacy
- ✓ Data masking
- ✓ Synthetic data generation
- ✓ Secured multi-party computation

4.2. Algorithmic Bias in Medical AI

Implementing AI for healthcare can show revolutionary development but creates more critical points. One of them, the algorithmic bias in medical AI systems on predictive models tends to generate unequal results for different population groups. In healthcare, this can directly affect diagnosis, treatment prescribing, and outcomes of patients.

4.2.1. Sources of Bias

- ✓ **Underrepresentation in training data:** Algorithms often fail when training on non-diverse datasets, such as a skin cancer detection model trained on light skin using dermoscopic and macroscopic photographs, which can't diagnose accurately on dark skin [12].
- ✓ **Geographic/Socioeconomic collection bias:** Medical data is often gathered from well-funded urban academic institutes, causing the training of models that do not present accurate outcomes on rural or low-income populations. For example, a model trained using the Colombo area dataset based on urban and too-busy people does not represent the whole country. So this model does not give accurate outcomes for Anuradhapura area dataset.
- ✓ **Historical treatment biases:** Using unfair historical treatments datasets, such as treatment records age between 20 and 30, to train a model, is not made equal outcomes for other age ranges.

4.2.2. Real-World Impact

In 2019 in the United States of America, researchers found the algorithm that they used has a significant racial bias [13]. Black patients were considerably sicker than white patients, even when Black patients and white patients had the same "risk score" from the AI. For an example, at the highest risk levels, Black patients had 26.3% chronic illnesses more than white patients.

The real-world impact is the Black patients are restricted from getting the extra medical treatments that they need. If remove this bias, the ability to get extra medical treatment for black patients is more than doubled from 17.7% to 46.5%.

The algorithm used past datasets, including the cost for healthcare. In this dataset the Black patient spent less money than the White patients for their healthcare because of structural inequities and barriers to access to healthcare. As a result, the system interprets the Black patients as healthier. Therefore, the Black patients were classified as low risk even they needed extra care and treatments.

To fix this bias, researchers replace the algorithm's variable "total cost" with direct health indicators such as combination of total cost and number of chronic conditions, which reduced the racial bias by over 80%.

4.2.3. Mitigation Strategies

- ✓ **Technical Approaches:**
  - Deploying real-time auditing mechanism that track the model perform across demographic groups such as race, gender, age, and socioeconomic status to identify unequal outcomes and trigger model retraining or do correction when bias pattern occurred.
  - Apply bias-control techniques such as reweighting data, balanced sampling, and fairness constrained using varied and representative data during training the model to detect, reduce, and prevent unfair bias.
- ✓ **Process Approaches:**
  - Establish an expert review team from multi-professionals such as clinicians, data scientists, ethicists, and patient advocates to assess AI system for fairness, transparency, and safety before and after AI systems are used.
  - Before real-world implementation, it is necessary to ensure medical AI systems are tested independently, clinically validated, and officially approved by regulators as systems that are fair, reliable, safe, and accountable.

4.3. Ethical Decision Framework

Using AI in healthcare directly impacts patient safety. Therefore, data scientists must apply a well-structured ethical review before deploying the AI. These six checkpoints offer a clear and practical ethical decision framework for applying AI in healthcare.

- ✓ **Data Minimization:**

  Question: Are only necessary patient data elements collected and processed?

  Explanation: Excessively collecting data increases privacy risk and data exposure when breached. Constraining data to strictly necessary reduces unnecessary sensitivity and complexity. So this aligns with regulatory expectations of GDPR [10].
- ✓ **Informed Consent:**

  Question: Do patients aware how their data will be manipulated?

  Explanation: Patients must be aware of process and potential of secondary uses of their data. Transparency supports autonomy and trust in digital health systems.

- ✓ **Bias and Fairness Assessment:**

  Question: Has the model been evaluated across different demographic groups to identify performance gaps or unfair bias in outcomes?

  Explanation: The model should evaluate separately across race, age, gender, and socioeconomic groups to detect bias and prevent AI system from reinforcing existing social and healthcare inequalities [13].

- ✓ **Transparency and Documentation:**

  Question: Do scientists clearly document the model's assumptions, training data sources, and validation results to ensure transparency and accountability?

  Explanation: Healthcare AI systems require traceability mechanism, and documentation allows independent review and supports regulatory accountability.

- ✓ **Security and Access Controls**

  Question: Do system implements encryption, authentication, and audit mechanism to protect patient data?

  Explanation: Technical safeguard prevents unauthorized access, helping maintain patient's confidentiality.

- ✓ **Accountability and Oversight**

  Question: Who is the authorized person bears the responsibilities if the AI system provides incorrect or harmful outcomes?

  Explanation: Such as scientists, developers, clinicians, and healthcare institutions must follow clear governance structure. The governance structure requires defined job description, duties, responsibilities, procedures, and protocols.

Beyond the evaluating of checklists, the "right to explanation" is an essential thing when assisting AI in healthcare to make diagnoses. Patients can request clear information about automated decisions that affect them. In healthcare, both the patient and clinician or healthcare provider should aware and trust AI-suggested interpretable, transparent, and clinical defensive diagnosis and treatment plans.

The model interpretation ability plays a critical role when applying AI for healthcare. The predictions without any explanation make ethical and legal risks. The clinician or

healthcare provider remains responsible for the final decision and justifies treatment choices. Interpretable models build trust, support clinical validation, and help detect hidden bias or logical fallacies. In healthcare, accuracy alone is not enough and ethical use depends on transparency, accountability, and explainability.

4.4. Stakeholder Impact Analysis

AI deployment in healthcare affects multiple stakeholder groups. Each one experience distinct benefits and risks, which must be evaluated through responsible governance and effective policy designs.

- ✓ **Patients:** Patients arrive more benefits through earlier diagnosis, improved risk prediction, personalized treatment, and faster, consistent clinical analysis even in under-resources regions. However, misdiagnosis, algorithmic bias, unequal outcomes, and data breaches makes long term harm. Therefore, deploying AI requires strong data protection, fairness control, encryption, access management, auditing, and transparent acceptance in accordance with HIPAA and GDPR requirements [9], [10].
- ✓ **Healthcare Providers:** Healthcare providers use AI as a decision-supportive tool to improve diagnosis, reduce brain-tiring workload, prioritize high-risk patients, and support treatment plans. They should be aware of the AI limitations and avoid overreliance on automated outcomes and should have good interpretation ability with AI outcomes. Clear guideline, protocols, and holding time-to-time awareness programs are necessary to improves critical evaluation on AI outputs.
- ✓ **Researchers and Developers:** Researchers and developers have a responsibility for designing ethical AI, not just caring about technical accuracy. They must ensure data quality, transparent documentation, diverse validation, bias assessment, and continuous monitoring. Ethical practice also requires open reporting, after-deployment evaluation, and system revision when harmful or unintended effects occur.

5. Technical Implementation

5.1. Code Quality Approach

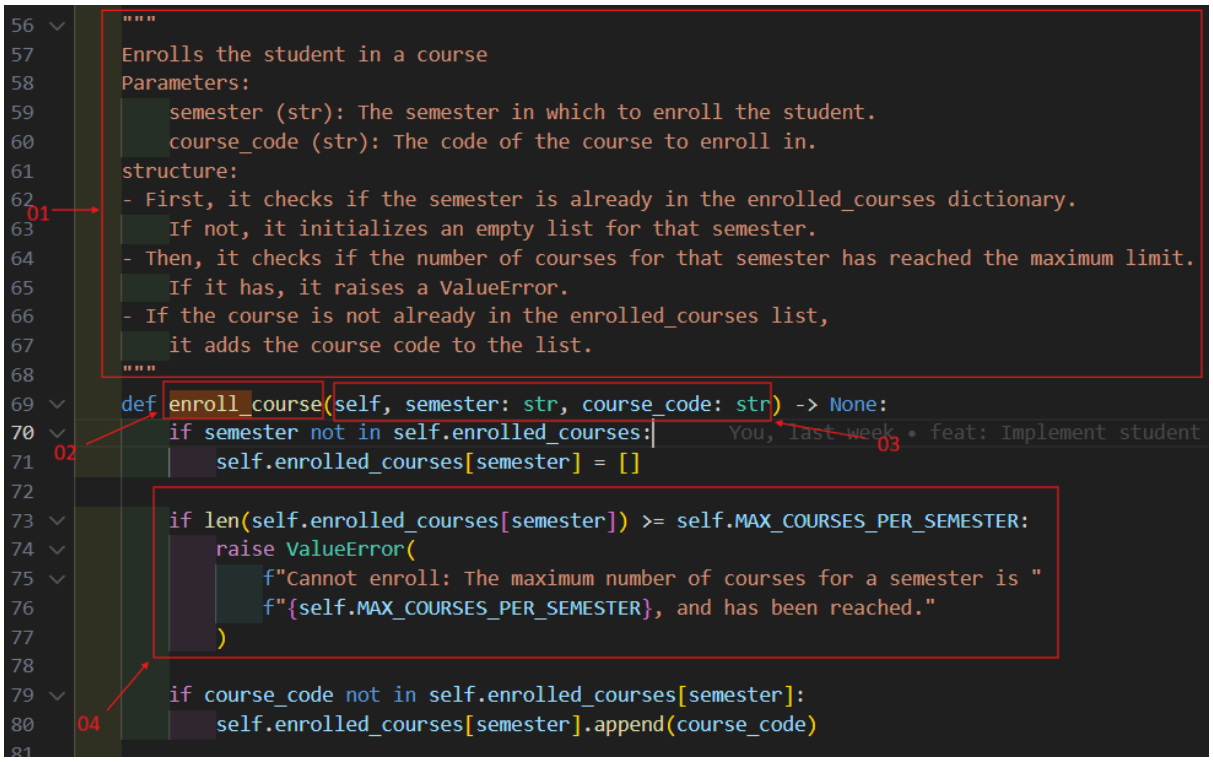


Figure 5.1.1: Demonstrate the code quality

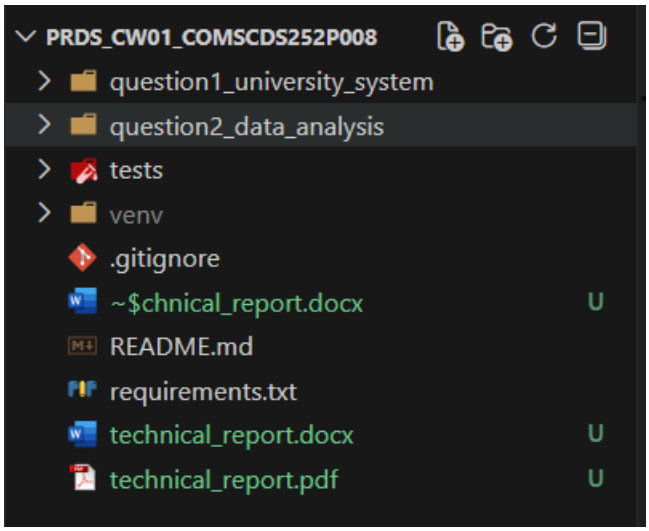


Figure 5.1.2: File Structure

For maintaining the code quality, follows these approaches;

- ✓ Module-wise file structure: The system maintains separate folders for each module (Figure 5.1.2).

- ✓ Meaningful variable, function, and class naming: Naming the function by defining the functionality of it. For an example enrolling courses function named as `enroll_course()` (Figure 5.1.1, label 02).
- ✓ Giving type hints by mentioning parameter type and return type for improving code readability and type safety (Figure 5.1.1, label 03).
- ✓ Use of Docstrings: The system uses structured Docstrings to document purpose of functions, parameters, logic flow in the function, and error conditions for improving developer understandability (Figure 5.1.1, label 01).
- ✓ Separation of concerns: A detailed explanation is at 2.2.1 (Single Responsibility Principle)

5.2. Validation and Error Handling

Validation in class methods (Figure 5.1.1, label 04) and error handling is implemented at application layer. A detailed explanation is provided at section 2.2.2 (Validation inside Methods) and 2.2.3 (Error Handling in Application Layer).

5.3. Unit Testing

Unit testing uses Python's `unittest` framework to ensure the accuracy of class functionalities including `Person`, `Student`, `Faculty`, `Staff`, `Course`, and `Department`. Each class were tested independently to validate key behaviors and restrictions. For example, tests confirm grade validation by checking value is between 0.0 and 4.0, and course enrollment restrictions by checking course capacity. This testing structure helps to identify logical errors early and improve overall code reliability.

```
19      def test_course_limit(self):
20          for i in range(6):
21              self.s.enroll_course("2025S1", f"DS10{i}")
22          with self.assertRaises(ValueError):
23              # 7th course should fail      You, last week • feat: I
24              self.s.enroll_course("2025S1", "DS999")
25
26      def test_grade_validation(self):
27          self.s.enroll_course("2025S1", "DS101")
28          with self.assertRaises(ValueError):
29              self.s.add_grade("2025S1", "DS101", 4.5)  # invalid
30          self.s.add_grade("2025S1", "DS101", 3.5)  # valid
31          self.assertAlmostEqual(self.s.gpa, 3.5)
32
33      def test_gpa_read_only(self):
34          self.s.enroll_course("2025S1", "DS101")
35          self.s.add_grade("2025S1", "DS101", 3.0)
36          with self.assertRaises(AttributeError):
37              # should fail because property has no setter
38              self.s.gpa = 4.0
```

Figure 5.3.1: Unit testing on grade, course limit, GPA read only

5.4. Challenges and Solutions

- ✓ **Keep code versions:** Need to track codes such as when, what, who, and for what change the codes. To solve, using Git helps to manage repository by committing code changes activity by activity, managing branches for module by module or task by task. These features make records about who did the code change at which time for what.
- ✓ **Reducing code redundancy:** Redundancy increases code lines, volume, and complexity of the code base. To reduce that use OOP concepts such as Inheritance, Encapsulation, Polymorphism
- ✓ **Run the application continuously without corruption:** Handling exceptions and errors using the try-except technique helps to work on applications continuously and track errors. Therefore later, it has an opportunity to fix tracked errors.
- ✓ **Enroll courses semester-wise:** It uses `dictionary` with semester as key and courses list as value in the enrolled course `list.`
- ✓ **Converting text values to numeric value:** Mapping strategy is used to solve this with using constant mapping help data. For an example for mapping text rating value to numeric, it uses Figure 5.4.1

```
# Define the map for converting text rating to int rating
RATING_MAP = {
    "One": 1,
    "Two": 2,
    "Three": 3,
    "Four": 4,
    "Five": 5,
}
```

Figure 5.4.1: Mapping constant data

- ✓ **Version conflict with Python and Libraries:** When installing libraries, it shows conflicts with Python v3.13.7. Installing libraries by mentioning compatible version solves the issue.
- ✓ **Need to run continuously web scrapping function:** To prevent the error of too-many-request, it uses a random delay between requests, and to manage request failures, it uses a retry mechanism.
- ✓ **Data file encode-decode mismatch:** When read the written CSV data file, it shows encode-decode mismatch. Using standard encode and decode method "`utf-8-sig`" solves the issue.

6. Reflection

6.1. Learning outcomes

- ✓ This coursework develops the skill of Python programming such as follows OOP concepts, error handling, and code documentation.
- ✓ It improves the analytical skills through applying data science techniques with handling network errors and data file writing and reading concepts.
- ✓ It enhances comprehensive knowledge of ethical standards, regulatory frameworks, and responsibilities when using AI in the modern world.

6.2. Future Improvements

- ✓ Implementing a GUI based data manipulating UMS with permanent database to make more user friendly access.
- ✓ Automating the basic statistical analysis and interpretation implementing machine learning and user-given parametric values to reduce analysis and interpretation time and increase reusability of developments.
- ✓ Testing the model multiple times on different subsets of the data to make sure its prediction is accurate.

6.3. Real-World Applications

The skills of programming that improved from this coursework, can be used to build well organized Learning Management System (LMS) for education institute.

Statistical analytical skills can be applied to develop the E-Commerce business by knowing the demand and market positions. For an example analysis provide an idea about fast-moving and slow-moving items. Therefore, item producers can make a priority level for each item and make marketing campaigns for low-moving items to grow sales.

7. References

[1] “All products | Books to Scrape - Sandbox,” *books.toscrape.com*. https://books.toscrape.com/, accessed Feb. 2026.

[2] K. Reitz, “Requests: HTTP for Humans™ — Requests 2.27.1 Documentation,” *requests.readthedocs.io*, 2024. https://requests.readthedocs.io/en/latest/, accessed Feb. 2026.

[3] L. Richardson, “Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation,” *Crummy.com*, 2019. https://www.crummy.com/software/BeautifulSoup/bs4/doc/, accessed Feb. 2026.

[4] Pandas, “Python Data Analysis Library,” *Pydata.org*, 2018. https://pandas.pydata.org/, accessed Feb. 2026.

[5] SciPy, “SciPy.org,” *Scipy.org*, 2020. https://scipy.org/, accessed Feb. 2026.

[6] scikit-learn, “scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation,” *Scikit-learn.org*, 2019. https://scikit-learn.org/, accessed Feb. 2026.

[7] Plotly, “Plotly Python Graphing Library,” *plotly.com*, 2023. https://plotly.com/python/, accessed Feb. 2026.

[8] U.S. Department of Health & Human Services, “Health Information Privacy,” *HHS.gov*, 2020. https://www.hhs.gov/hipaa/index.html, accessed Feb. 2026.

[9] Centers for Medicare & Medicaid Services, "HIPAA Basics for Providers: Privacy, Security, & Breach Notification Rules," MLN Fact Sheet MLN909001, May 2025. https://www.cms.gov/files/document/mln909001-hipaa-basics-providers-privacy-security-breach-notification-rules.pdf, accessed Feb. 2026.

[10] European Union, “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, and repealing Directive 95/46/EC (General Data Protection Regulation),” *Europa.eu*, Apr. 27, 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj, accessed Feb. 2026.

[11] “Navigating GDPR in Healthcare: Essential Guide,” *GDPR Register*, Oct. 30, 2024. https://www.gdprregister.eu/articles/healthcare-sector-gdpr/, accessed Feb. 2026.

[12] R. J. Chen *et al.*, “Algorithmic fairness in artificial intelligence for medicine and healthcare,” *Nature biomedical engineering*, vol. 7, no. 6, pp. 719–742, Jun. 2023, doi: https://doi.org/10.1038/s41551-023-01056-8, accessed Feb. 2026.

[13] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, “Dissecting racial bias in an algorithm used to manage the health of populations,” Oct. 2019. Available: https://www.ftc.gov/system/files/documents/public_events/1548288/privacycon-2020-ziad_obermeyer.pdf, accessed Feb. 2026

Appendix A - University Management System Demonstration

A.1. Student Enrollment and GPA Calculation

```
--- Enrolling Courses for a Student ---
--------------------------------------------------
Name: A. L. Amal Gunarathne
ID: PS001
Email: amal.ps001@nibm.lk
Phone: 0710000001
Student ID: S001
Major: Artificial Intelligence
Enrollment Date: 2025-10-01
--------------------------------------------------
Adding Courses under semester 2025S1
--------------------------------------------------
CS101
CS102
CS103
CS104
CS105
--------------------------------------------------
Adding Grades
--------------------------------------------------
CS101 :                 3.5
CS102 :                 3.7
CS103 :                 3.2
CS104 :                 4.0
CS105 :                 3.8
--------------------------------------------------

--- Academic Status for A. L. Amal Gunarathne ---
--------------------------------------------------
GPA: 3.64
Academic Status: Dean's List
```

A.2. Error Handling

```
--- Error Handling Demonstration ---
--------------------------------------------------
Attempting to enroll in more than 6 courses in a semester:
Error: Cannot enroll: The maximum number of courses for a semester
is 6, and has been reached.

Attempting to add a grade for a course the student is not enrolled
in:
Error: Cannot add grade: student is not enrolled in semester
2025S1 course CS999

Attempting to add an invalid grade:
Error: Grade must be between 0.0 and 4.0
```

A.3. Polymorphism

```
--- Polymorphism Demonstration ---
--------------------------------------------------
Student:
--------------------------------------------------
A. L. Amal Gunarathne: Attend lectures, complete coursework, and
maintain academic progress.
--------------------------------------------------
Faculty:
--------------------------------------------------
Mr. R. M. Jayasinghe: Teach courses, conduct research, and
supervise/mentor students.
--------------------------------------------------
Staff:
--------------------------------------------------
Ms. R. S. Oshadi Karunaratne: Manage student services and welfare.
As a Student Affairs Executive in the DEP-BUS department.
--------------------------------------------------
```

A.4. Department Summary

```
--- Department Summary (Computing) ---
--------------------------------------------------
Department: Computing
Head: Dr. N. K. Perera
Faculty: Ms. S. D. Wijeratne
Courses: DS101, DS102, DS103
   DS101 - Intro to Data Science (3 credits) | Instructor: Dr. N.
K. Perera | Enrolled: 2/2
   DS102 - Python for Data Science (3 credits) | Instructor: Ms.
S. D. Wijeratne | Enrolled: 2/3
   DS103 - Machine Learning (3 credits) | Instructor: Dr. N. K.
Perera | Enrolled: 0/2

--- Department Summary (Business) ---
--------------------------------------------------
Department: Business
Head: Mr. R. M. Jayasinghe
Faculty:
Courses: BU201, BU202, BU203
   BU201 - Business Analytics (3 credits) | Instructor: Mr. R. M.
Jayasinghe | Enrolled: 2/3
   BU202 - Research Methods (3 credits) | Instructor: Mr. R. M.
Jayasinghe | Enrolled: 1/3
   BU203 - Project Management (3 credits) | Instructor: Mr. R. M.
Jayasinghe | Enrolled: 0/3
```

Appendix B – Data Analysis Console Outputs

B.1. Data Cleaning

```
-- Cleaning Data ---
-- Before Cleaning --
Initial Data Shape: (500, 5)

Data Types:
-------------------------------------------------
title           object
price           object
rating          object
category        object
availability    object
dtype: object

Data Summary:
-------------------------------------------------
                        title   price rating category
availability
count                     500     500    500      500
500
unique                    499     474      5       44
1
top     The Star-Touched Queen  £37.80    One  Default     In
stock
freq                        2       2    110       79
500

NULL Data:
-------------------------------------------------
title           0
price           0
rating          0
category        0
availability    0
dtype: int64

-- Cleaned Data saved to
question2_data_analysis/data/cleaned/cleaned_books_data_500.csv --
-- After Cleaning --
Data Shape: (500, 7)

Data Types:
-------------------------------------------------
title              object
price             float64
rating              int64
category           object
availability       object
price_category     object
in_stock             bool
```

```
dtype: object

Data Summary:
--------------------------------------------------
           price      rating
count  500.000000  500.000000
mean    35.040400    2.928000
std     14.553083    1.429301
min     10.160000    1.000000
25%     21.857500    2.000000
50%     36.650000    3.000000
75%     47.745000    4.000000
max     59.900000    5.000000

NULL Data:
--------------------------------------------------
title             0
price             0
rating            0
category          0
availability      0
price_category    0
in_stock          0
dtype: int64
```

B.2. Exploratory Analysis

```
-- Central tendency: Mean, Median, Mode for Price --
Mean:           35.04
Median:         36.65
Mode:           13.34

-- Dispersion: standard deviation, range --
STD:            14.55
Range:          49.74

-- Group statistics: average price by category (top 5) --
         category  price
0         Fiction  36.29
1   Add a comment  36.01
2         Default  35.90
3  Sequential Art  33.69
4      Nonfiction  31.72

-- Rating distribution: frequency count --
   rating  count
0       1    110
1       2    102
2       3     99
3       4     92
4       5     97

-- Outlier detection: Use IQR method for price outliers --
Q1:                           21.86
```

```
Q3:                              47.74
IQR:                             25.89
Lower Bound:                     -16.97
Upper Bound:                     86.58
Number of Outliers:              0

-- Correlation analysis: Pearson correlation between price and
rating --
Pearson Correlation Coefficient (r):    0.004
P-value:                                0.9251
Conclusion: No statistically significant correlation.

-- Hypothesis testing - Compare average prices between Fiction vs
Non-Fiction --
Fiction Mean Price:     36.29
Non-Fiction Mean Price: 31.72
T-statistic:            1.457
P-value:                0.1500
Conclusion: No significant difference in average prices.
```

B.3. Predictive Analysis

```
R² Score:                        -0.145
Mean Absolute Error (MAE):       11.867

-- Feature Influence (sorted by strength): --
category_Crime             -26.472438
category_Suspense           20.939491
category_Womens Fiction     19.995456
category_Erotica           -18.148579
category_Novels             17.471421
category_Politics           17.351509
category_Biography         -16.266474
category_Health             14.006421
category_Christian          13.482474
category_Autobiography     -11.113783
dtype: float64
```