COURSE WORK: PROGRAMMING FOR DATA SCIENCE 2025.1 BATCH

MSc Data Science: Coventry University UK

COMSCDS251P-017
N. M. Pathirage

Contents

Question 1

Project Report on University Management System

University Management System is a large-scale implementation of Object-Oriented Programming (OOP) ideas, implemented in Python. As both a working example and an operational system, the system simulates university operation, student enrollment, instructor workload, departmental organizational hierarchy, and course enrollments. The project expands OOP ideas by the addition of real-world issues like validation, sanity checks on data, and polymorphic behavior that simulates how real-world universities work.

The project is heavily dependent on the four fundamental pillars of OOP: polymorphism, encapsulation, inheritance, and abstraction. All four are demonstrated in a working and rational way, and so not only is the system a learning device but also an expandable structure that can be readily expanded and developed into a full application in the future.

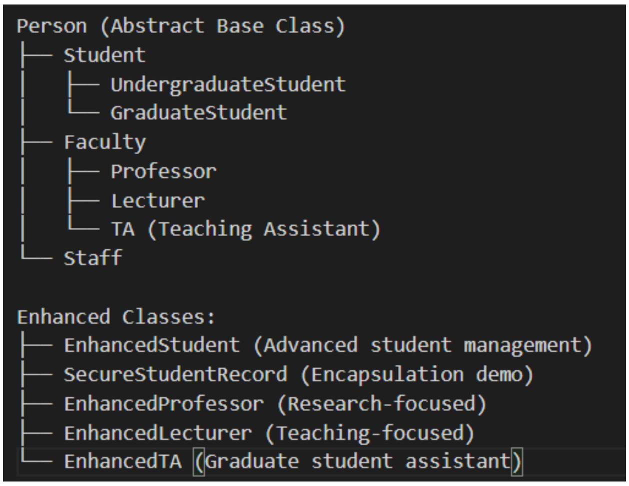
System Architecture and Design

Encapsulation is accomplished with a properly defined hierarchy of classes starting with an abstract top-level class named Person. The rest of the classes in the university system inherit their nature from the abstract class. Students, employees, and teachers are defined as subclasses, each having a list of properties and procedures. The hierarchy is founded on code reuse and indicates how inheritance may be applied more than one level.

Students are categorized in terms of whether they are graduate students or undergraduate students, and faculty are categorized in terms of teaching assistants, lecturers, and professors. The categorization provides polymorphism to the program since various types of faculties distribute workload and responsibility differently. For instance, the professor's workload may be research work and teaching, while the lecturer's workload would be teaching in large quantities and course planning.

Apart from the primitive classes, there are also advanced classes like EnhancedStudent, SecureStudentRecord, and EnhancedProfessor. The classes are more advanced implementations of OOP concepts like encapsulation and specialization. For instance, the SecureStudentRecord class provides secure storage and retrieval of sensitive details like GPA and email addresses, along with tests to verify that they do not allow invalid or malicious input.

The following is the class hierarchy of the system.



```
Person (Abstract Base Class)
├── Student
│   ├── UndergraduateStudent
│   └── GraduateStudent
├── Faculty
│   ├── Professor
│   ├── Lecturer
│   └── TA (Teaching Assistant)
└── Staff

Enhanced Classes:
├── EnhancedStudent (Advanced student management)
├── SecureStudentRecord (Encapsulation demo)
├── EnhancedProfessor (Research-focused)
├── EnhancedLecturer (Teaching-focused)
└── EnhancedTA (Graduate student assistant)
```

Core Features

The most significant building blocks of the system are grouped into several management domains: Person Management – Processes the basic information of all of the individuals within the university, i.e., names and IDs.

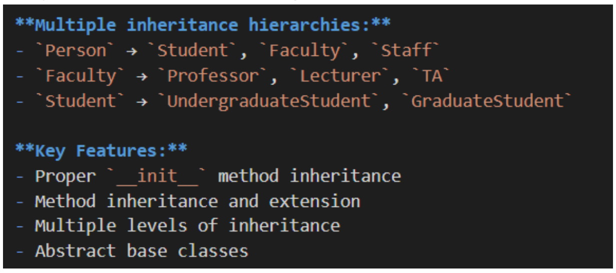
- Student Management – Processes enrollment to courses, calculation of GPA, and academic status.
- Faculty Management – Processes course assignment, workload assignment, and role-dependent functionality.
- Department Management – Processes faculty and course offerings by department.
- Course Management – Imposes prerequisites, capacity for enrollment, and waitlists.
- Registration System – Offers complete course registration, from discovery and prerequisite checks.

Modules are split into separate Python files in a rational way, i.e., student.py, faculty.py, and department.py, with a modular structure that is readable, maintainable, and scalable.

Demonstration of OOP Principles

Inheritance

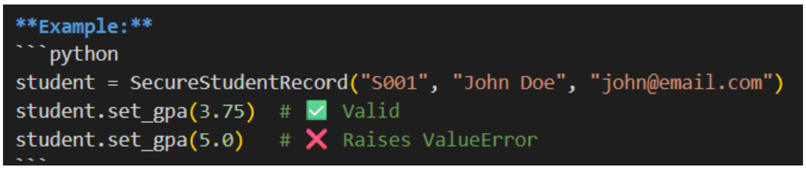
The system applies multiple inheritance hierarchies to specify where methods and properties are inherited and customized in subclassing. Since Person is declared as an abstract base class, the system only instantiates meaningful roles like Student or Faculty, employing a realistic design practice.



```
**Multiple inheritance hierarchies:**
- `Person` → `Student`, `Faculty`, `Staff`
- `Faculty` → `Professor`, `Lecturer`, `TA`
- `Student` → `UndergraduateStudent`, `GraduateStudent`

**Key Features:**
- Proper `__init__` method inheritance
- Method inheritance and extension
- Multiple levels of inheritance
- Abstract base classes
```

Encapsulation

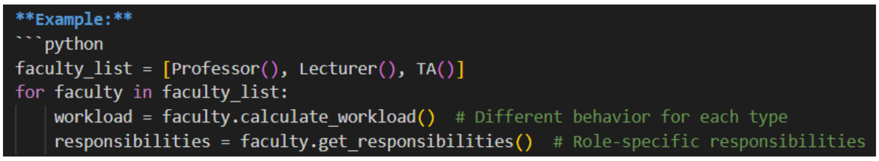
Encapsulation is demonstrated by the SecureStudentRecord class. Private members are accessed but not directly, rather through calling getter and setter methods. Valid input can be supplied only. A GPA, for example, can be only between 0.0 and 4.0, and erroneous input raises exceptions. The limitation mimics how sensitive information is handled in actual systems.



~~~
**Example:**
```python
student = SecureStudentRecord("S001", "John Doe", "john@email.com")
student.set_gpa(3.75)  # ✅ Valid
student.set_gpa(5.0)   # ❌ Raises ValueError
```
~~~

Polymorphism

The program gets polymorphism through method overloading. The calculate_workload method, for example, acts differently depending on being a professor, lecturer, or teaching assistant. Tasks also vary depending on title, but the system can manage all objects of the faculty in a loop without distinction based on their polymorphic nature.



~~~
**Example:**
```python
faculty_list = [Professor(), Lecturer(), TA()]
for faculty in faculty_list:
    workload = faculty.calculate_workload()  # Different behavior for each type
    responsibilities = faculty.get_responsibilities()  # Role-specific responsibilities
~~~

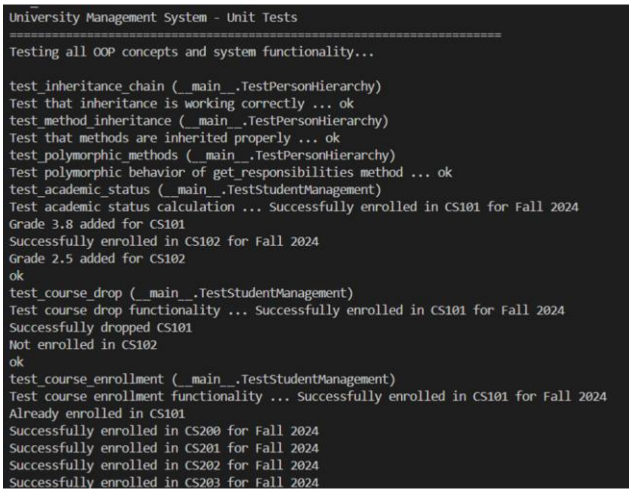
Abstraction

The use of abstract base classes like Person and interfaces as an intermediary of basic functionalities provides abstraction. This allows developers to invoke higher-level functions with no regard for the details behind, where the system is simple to scale and extend.

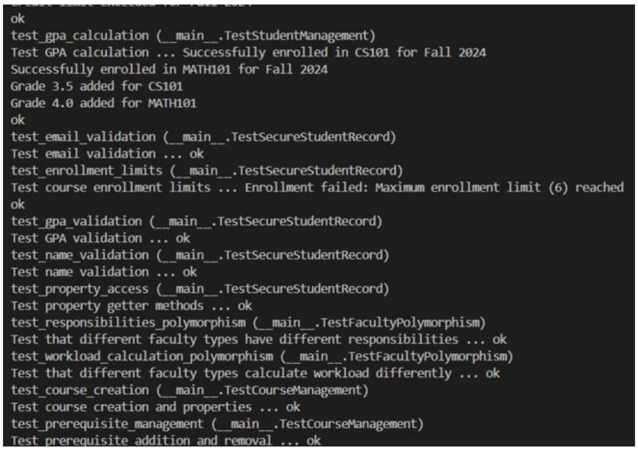
Execution and Testing of the System

There are two modes of the system: interactive mode and auto-demo mode. Interactive mode enables the user to navigate through a menu for a step-by-step demonstration of each OOP concept in isolation. This is useful for test and learn. Automated mode enables sequential demonstration without any intervention, which is an instantaneous demonstration of the capability of the system.
It is also incorporated within the design and provides complete validation of user input, enrollment limits, and course prerequisites. This not only ensures OOP principles, but it also ensures that the system is always consistent at all points for all operations.

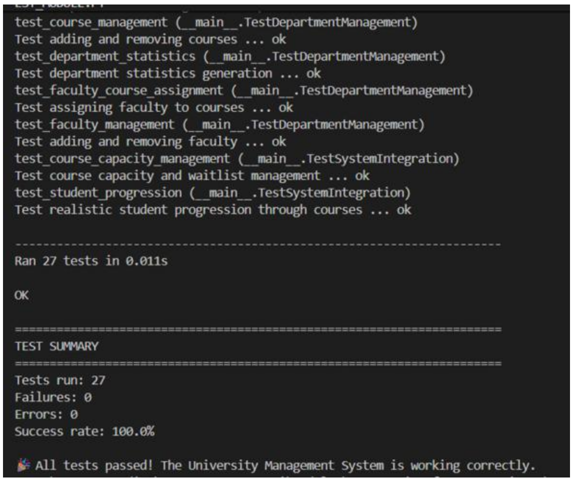
The following are the outputs of testing.



```
University Management System - Unit Tests
======================================================================
Testing all OOP concepts and system functionality...

test_inheritance_chain (__main__.TestPersonHierarchy)
Test that inheritance is working correctly ... ok
test_method_inheritance (__main__.TestPersonHierarchy)
Test that methods are inherited properly ... ok
test_polymorphic_methods (__main__.TestPersonHierarchy)
Test polymorphic behavior of get_responsibilities method ... ok
test_academic_status (__main__.TestStudentManagement)
Test academic status calculation ... Successfully enrolled in CS101 for Fall 2024
Grade 3.8 added for CS101
Successfully enrolled in CS102 for Fall 2024
Grade 2.5 added for CS102
ok
test_course_drop (__main__.TestStudentManagement)
Test course drop functionality ... Successfully enrolled in CS101 for Fall 2024
Successfully dropped CS101
Not enrolled in CS102
ok
test_course_enrollment (__main__.TestStudentManagement)
Test course enrollment functionality ... Successfully enrolled in CS101 for Fall 2024
Already enrolled in CS101
Successfully enrolled in CS200 for Fall 2024
Successfully enrolled in CS201 for Fall 2024
Successfully enrolled in CS202 for Fall 2024
Successfully enrolled in CS203 for Fall 2024
```



```
ok
test_gpa_calculation (__main__.TestStudentManagement)
Test GPA calculation ... Successfully enrolled in CS101 for Fall 2024
Successfully enrolled in MATH101 for Fall 2024
Grade 3.5 added for CS101
Grade 4.0 added for MATH101
ok
test_email_validation (__main__.TestSecureStudentRecord)
Test email validation ... ok
test_enrollment_limits (__main__.TestSecureStudentRecord)
Test course enrollment limits ... Enrollment failed: Maximum enrollment limit (6) reached
ok
test_gpa_validation (__main__.TestSecureStudentRecord)
Test GPA validation ... ok
test_name_validation (__main__.TestSecureStudentRecord)
Test name validation ... ok
test_property_access (__main__.TestSecureStudentRecord)
Test property getter methods ... ok
test_responsibilities_polymorphism (__main__.TestFacultyPolymorphism)
Test that different faculty types have different responsibilities ... ok
test_workload_calculation_polymorphism (__main__.TestFacultyPolymorphism)
Test that different faculty types calculate workload differently ... ok
test_course_creation (__main__.TestCourseManagement)
Test course creation and properties ... ok
test_prerequisite_management (__main__.TestCourseManagement)
Test prerequisite addition and removal ... ok
```



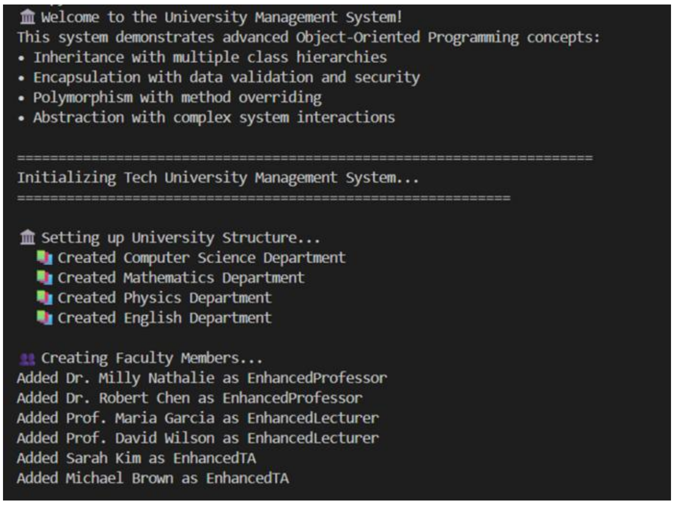
```
test_course_management (__main__.TestDepartmentManagement)
Test adding and removing courses ... ok
test_department_statistics (__main__.TestDepartmentManagement)
Test department statistics generation ... ok
test_faculty_course_assignment (__main__.TestDepartmentManagement)
Test assigning faculty to courses ... ok
test_faculty_management (__main__.TestDepartmentManagement)
Test adding and removing faculty ... ok
test_course_capacity_management (__main__.TestSystemIntegration)
Test course capacity and waitlist management ... ok
test_student_progression (__main__.TestSystemIntegration)
Test realistic student progression through courses ... ok

----------------------------------------------------------------------
Ran 27 tests in 0.011s

OK

======================================================================
TEST SUMMARY
======================================================================
Tests run: 27
Failures: 0
Errors: 0
Success rate: 100.0%

🎉 All tests passed! The University Management System is working correctly.
```

Code Quality and Extensibility

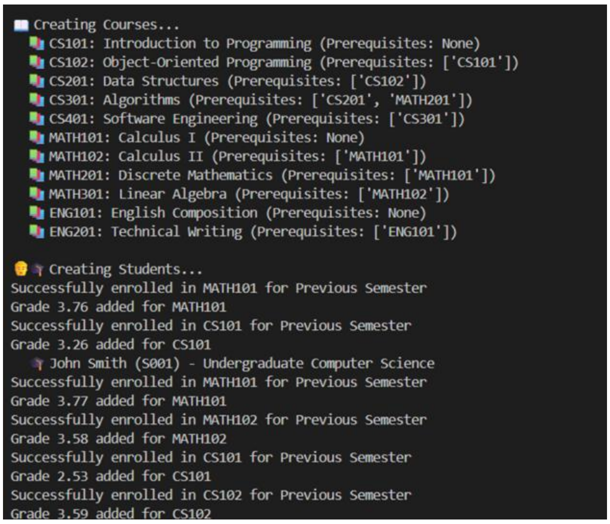
The code is written with Python's best practices like type hints, good docstrings, and modular code. The code is readily readable, understandable, and maintainable by others, and others can easily contribute to the project. The system is also extensible. New positions (new types of faculty, for instance) can be included by future developers, improved reporting, or even a web or database interface.

Other features like logging, testability by unit, and integration with Jupyter notebooks contribute even more to the value of the project. They contribute to the impression that the system is a proof-of-concept system, but a sufficient foundation for a bigger program.

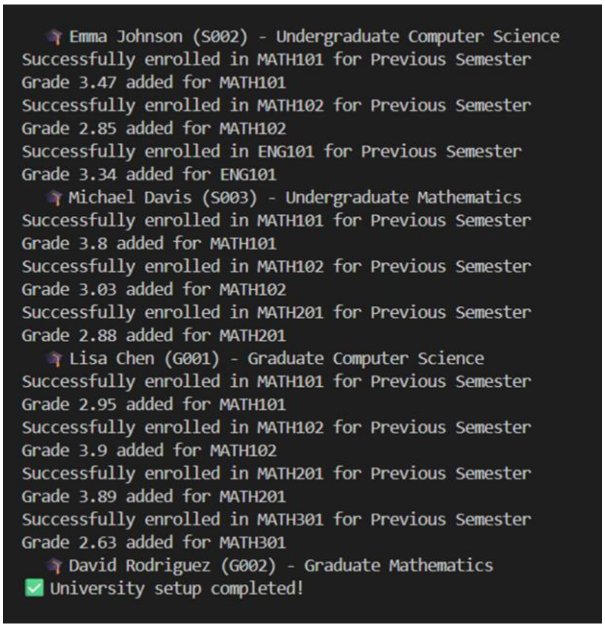
Final Outputs



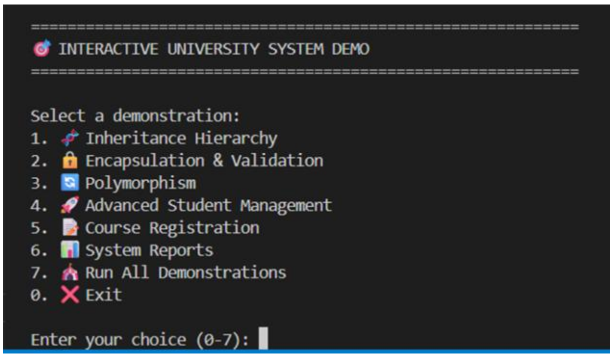
```
🏛 Welcome to the University Management System!
This system demonstrates advanced Object-Oriented Programming concepts:
• Inheritance with multiple class hierarchies
• Encapsulation with data validation and security
• Polymorphism with method overriding
• Abstraction with complex system interactions

======================================================================
Initializing Tech University Management System...
============================================================

🏛 Setting up University Structure...
  📚 Created Computer Science Department
  📚 Created Mathematics Department
  📚 Created Physics Department
  📚 Created English Department

👥 Creating Faculty Members...
Added Dr. Milly Nathalie as EnhancedProfessor
Added Dr. Robert Chen as EnhancedProfessor
Added Prof. Maria Garcia as EnhancedLecturer
Added Prof. David Wilson as EnhancedLecturer
Added Sarah Kim as EnhancedTA
Added Michael Brown as EnhancedTA
```



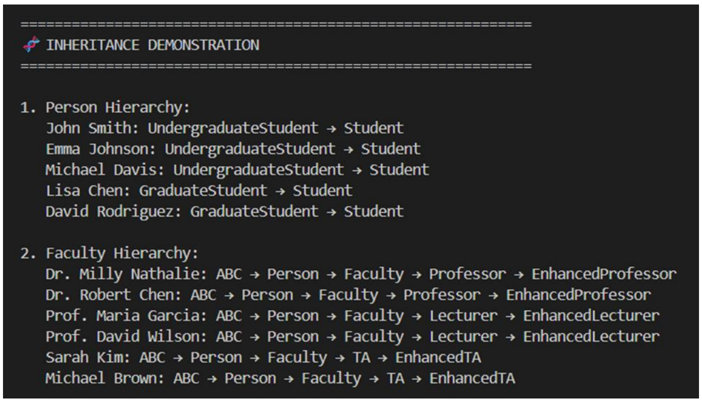
```
📖 Creating Courses...
  📚 CS101: Introduction to Programming (Prerequisites: None)
  📚 CS102: Object-Oriented Programming (Prerequisites: ['CS101'])
  📚 CS201: Data Structures (Prerequisites: ['CS102'])
  📚 CS301: Algorithms (Prerequisites: ['CS201', 'MATH201'])
  📚 CS401: Software Engineering (Prerequisites: ['CS301'])
  📚 MATH101: Calculus I (Prerequisites: None)
  📚 MATH102: Calculus II (Prerequisites: ['MATH101'])
  📚 MATH201: Discrete Mathematics (Prerequisites: ['MATH101'])
  📚 MATH301: Linear Algebra (Prerequisites: ['MATH102'])
  📚 ENG101: English Composition (Prerequisites: None)
  📚 ENG201: Technical Writing (Prerequisites: ['ENG101'])

👨‍🎓 Creating Students...
Successfully enrolled in MATH101 for Previous Semester
Grade 3.76 added for MATH101
Successfully enrolled in CS101 for Previous Semester
Grade 3.26 added for CS101
  🎓 John Smith (S001) - Undergraduate Computer Science
Successfully enrolled in MATH101 for Previous Semester
Grade 3.77 added for MATH101
Successfully enrolled in MATH102 for Previous Semester
Grade 3.58 added for MATH102
Successfully enrolled in CS101 for Previous Semester
Grade 2.53 added for CS101
Successfully enrolled in CS102 for Previous Semester
Grade 3.59 added for CS102
```



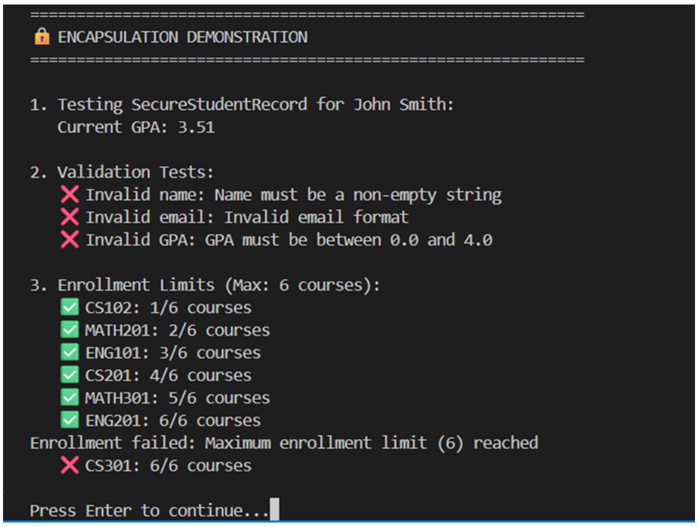
```
  🎓 Emma Johnson (S002) - Undergraduate Computer Science
Successfully enrolled in MATH101 for Previous Semester
Grade 3.47 added for MATH101
Successfully enrolled in MATH102 for Previous Semester
Grade 2.85 added for MATH102
Successfully enrolled in ENG101 for Previous Semester
Grade 3.34 added for ENG101
  🎓 Michael Davis (S003) - Undergraduate Mathematics
Successfully enrolled in MATH101 for Previous Semester
Grade 3.8 added for MATH101
Successfully enrolled in MATH102 for Previous Semester
Grade 3.03 added for MATH102
Successfully enrolled in MATH201 for Previous Semester
Grade 2.88 added for MATH201
  🎓 Lisa Chen (G001) - Graduate Computer Science
Successfully enrolled in MATH101 for Previous Semester
Grade 2.95 added for MATH101
Successfully enrolled in MATH102 for Previous Semester
Grade 3.9 added for MATH102
Successfully enrolled in MATH201 for Previous Semester
Grade 3.89 added for MATH201
Successfully enrolled in MATH301 for Previous Semester
Grade 2.63 added for MATH301
  🎓 David Rodriguez (G002) - Graduate Mathematics
✅ University setup completed!
```



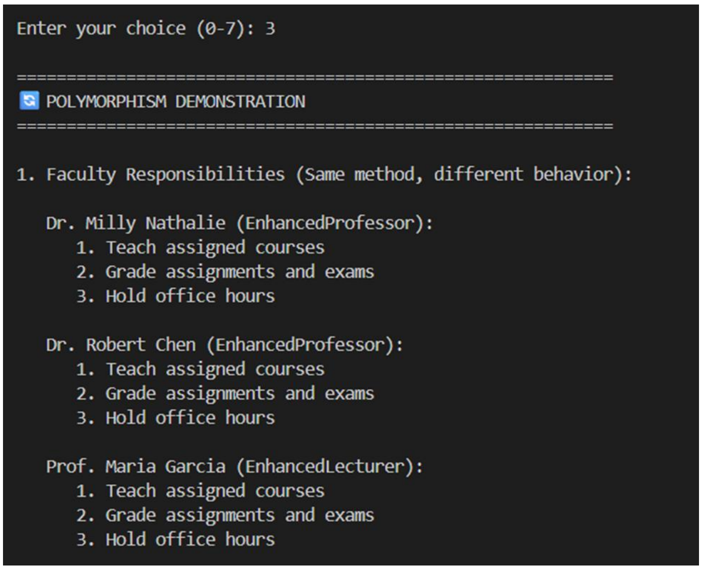
```
============================================================
🎯 INTERACTIVE UNIVERSITY SYSTEM DEMO
============================================================

Select a demonstration:
1. 🧬 Inheritance Hierarchy
2. 🔒 Encapsulation & Validation
3. 🔄 Polymorphism
4. 🚀 Advanced Student Management
5. 📝 Course Registration
6. 📊 System Reports
7. 🎪 Run All Demonstrations
0. ❌ Exit

Enter your choice (0-7): 
```



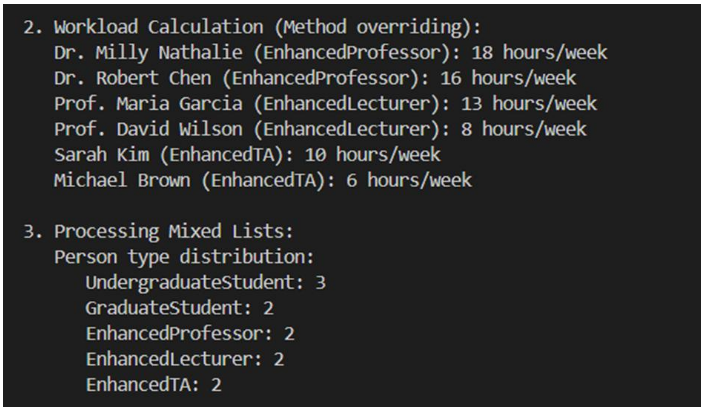
```
============================================================
🧬 INHERITANCE DEMONSTRATION
============================================================

1. Person Hierarchy:
   John Smith: UndergraduateStudent → Student
   Emma Johnson: UndergraduateStudent → Student
   Michael Davis: UndergraduateStudent → Student
   Lisa Chen: GraduateStudent → Student
   David Rodriguez: GraduateStudent → Student

2. Faculty Hierarchy:
   Dr. Milly Nathalie: ABC → Person → Faculty → Professor → EnhancedProfessor
   Dr. Robert Chen: ABC → Person → Faculty → Professor → EnhancedProfessor
   Prof. Maria Garcia: ABC → Person → Faculty → Lecturer → EnhancedLecturer
   Prof. David Wilson: ABC → Person → Faculty → Lecturer → EnhancedLecturer
   Sarah Kim: ABC → Person → Faculty → TA → EnhancedTA
   Michael Brown: ABC → Person → Faculty → TA → EnhancedTA
```



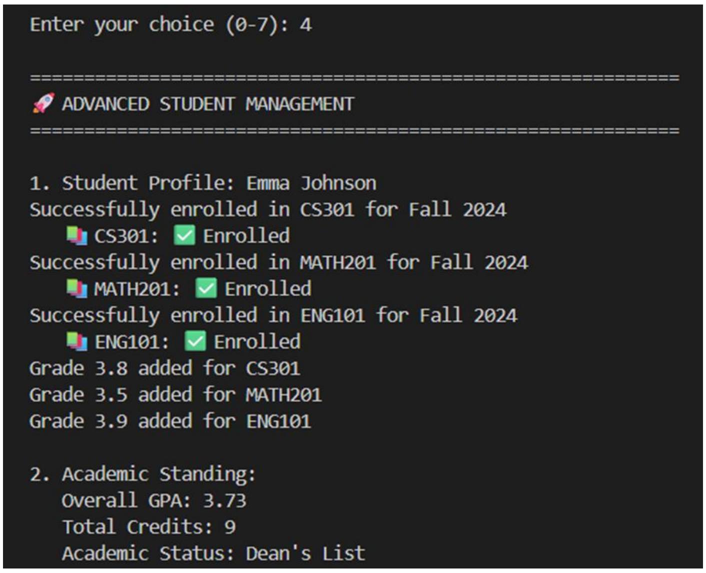
```
============================================================
🔒 ENCAPSULATION DEMONSTRATION
============================================================

1. Testing SecureStudentRecord for John Smith:
   Current GPA: 3.51

2. Validation Tests:
   ❌ Invalid name: Name must be a non-empty string
   ❌ Invalid email: Invalid email format
   ❌ Invalid GPA: GPA must be between 0.0 and 4.0

3. Enrollment Limits (Max: 6 courses):
   ✅ CS102: 1/6 courses
   ✅ MATH201: 2/6 courses
   ✅ ENG101: 3/6 courses
   ✅ CS201: 4/6 courses
   ✅ MATH301: 5/6 courses
   ✅ ENG201: 6/6 courses
Enrollment failed: Maximum enrollment limit (6) reached
   ❌ CS301: 6/6 courses

Press Enter to continue...
```



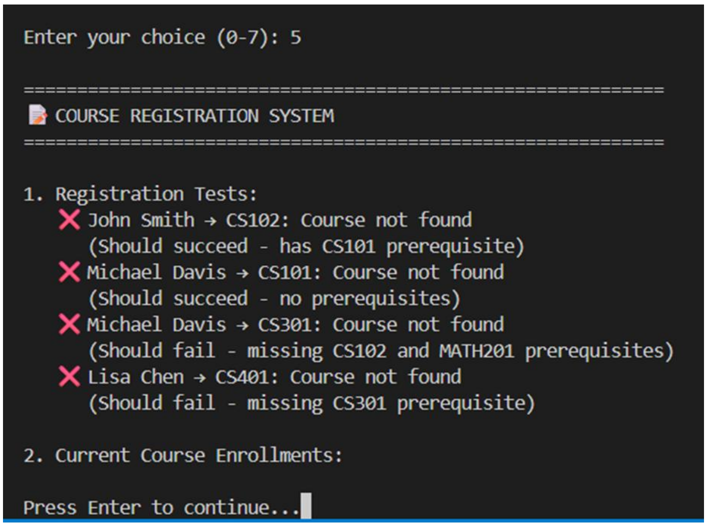
```
Enter your choice (0-7): 3

============================================================
🔄 POLYMORPHISM DEMONSTRATION
============================================================

1. Faculty Responsibilities (Same method, different behavior):

   Dr. Milly Nathalie (EnhancedProfessor):
      1. Teach assigned courses
      2. Grade assignments and exams
      3. Hold office hours

   Dr. Robert Chen (EnhancedProfessor):
      1. Teach assigned courses
      2. Grade assignments and exams
      3. Hold office hours

   Prof. Maria Garcia (EnhancedLecturer):
      1. Teach assigned courses
      2. Grade assignments and exams
      3. Hold office hours
```



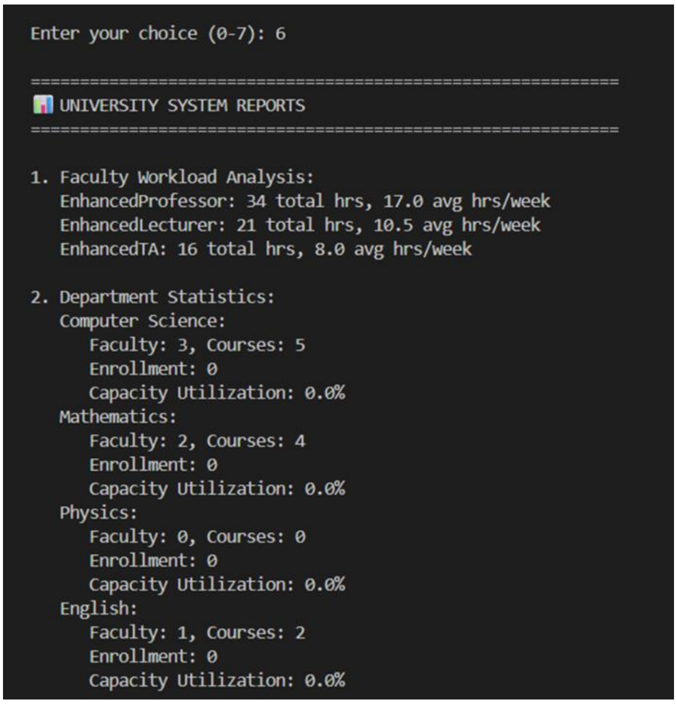
```
2. Workload Calculation (Method overriding):
   Dr. Milly Nathalie (EnhancedProfessor): 18 hours/week
   Dr. Robert Chen (EnhancedProfessor): 16 hours/week
   Prof. Maria Garcia (EnhancedLecturer): 13 hours/week
   Prof. David Wilson (EnhancedLecturer): 8 hours/week
   Sarah Kim (EnhancedTA): 10 hours/week
   Michael Brown (EnhancedTA): 6 hours/week

3. Processing Mixed Lists:
   Person type distribution:
      UndergraduateStudent: 3
      GraduateStudent: 2
      EnhancedProfessor: 2
      EnhancedLecturer: 2
      EnhancedTA: 2
```



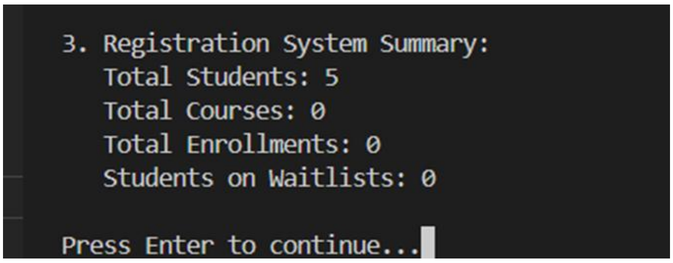
```
Enter your choice (0-7): 4

============================================================
🚀 ADVANCED STUDENT MANAGEMENT
============================================================

1. Student Profile: Emma Johnson
Successfully enrolled in CS301 for Fall 2024
   📚 CS301: ✅ Enrolled
Successfully enrolled in MATH201 for Fall 2024
   📚 MATH201: ✅ Enrolled
Successfully enrolled in ENG101 for Fall 2024
   📚 ENG101: ✅ Enrolled
Grade 3.8 added for CS301
Grade 3.5 added for MATH201
Grade 3.9 added for ENG101

2. Academic Standing:
   Overall GPA: 3.73
   Total Credits: 9
   Academic Status: Dean's List
```



```
Enter your choice (0-7): 5

============================================================
📝 COURSE REGISTRATION SYSTEM
============================================================

1. Registration Tests:
   ❌ John Smith → CS102: Course not found
      (Should succeed - has CS101 prerequisite)
   ❌ Michael Davis → CS101: Course not found
      (Should succeed - no prerequisites)
   ❌ Michael Davis → CS301: Course not found
      (Should fail - missing CS102 and MATH201 prerequisites)
   ❌ Lisa Chen → CS401: Course not found
      (Should fail - missing CS301 prerequisite)

2. Current Course Enrollments:

Press Enter to continue...
```



```
Enter your choice (0-7): 6

============================================================
📊 UNIVERSITY SYSTEM REPORTS
============================================================

1. Faculty Workload Analysis:
   EnhancedProfessor: 34 total hrs, 17.0 avg hrs/week
   EnhancedLecturer: 21 total hrs, 10.5 avg hrs/week
   EnhancedTA: 16 total hrs, 8.0 avg hrs/week

2. Department Statistics:
   Computer Science:
      Faculty: 3, Courses: 5
      Enrollment: 0
      Capacity Utilization: 0.0%
   Mathematics:
      Faculty: 2, Courses: 4
      Enrollment: 0
      Capacity Utilization: 0.0%
   Physics:
      Faculty: 0, Courses: 0
      Enrollment: 0
      Capacity Utilization: 0.0%
   English:
      Faculty: 1, Courses: 2
      Enrollment: 0
      Capacity Utilization: 0.0%
```



```
3. Registration System Summary:
   Total Students: 5
   Total Courses: 0
   Total Enrollments: 0
   Students on Waitlists: 0

Press Enter to continue...
```

GitHub URL: https://github.com/NawanjanaMP/Programming-For-Data-Science_Project.git

Learning Outcomes

Educationally, the project is providing important lessons in object-oriented programming, system design, and software engineering. It demonstrates that theoretical models like inheritance and polymorphism can be applied to a real-world kind of application. It illustrates how validation, security, and modular design may be included in the process of constructing systems dealing with sensitive data. The students who do this project will be able to understand OOP at a higher level and will also have a clear concept of coding standards. The project fills the gap between practice and theory, thus making students ready for real development problems.

Conclusion

The University Management System project is the ultimate application of OOP to model processes in the real world. The merging of the four fundamental principles of OOP with real-world characteristics like validation, error checking, and modularity not only builds an operational learning system but also an extendable development system. Its emphasis on quality of code and extensibility makes it extendable into a bigger system if necessary, beyond that of a learning project.

Overall, the project is technically sound in terms of OOP concepts and sound software engineering practice. Done either as a class, as a learning project by a student, or as a base for a larger project, the system is a solid base to learn and implement object-oriented programming in Python.

Question 2

Report for the Retail Pricing Analysis Pipeline Project

Retail Pricing Analysis Pipeline is a complete data science pipeline project covering web scraping, statistical modeling, data visualization, and predictive modeling in a modular pipeline. The pipeline is basically intended for online retail price trend analysis, price correlation with product rating and product availability, and finally deriving actionable insights that can be implemented by e-commerce businesses to improve their price strategy.

Most impressive about this project is the ambition. Rather than performing one piece of data science, it attempts to do the entire pipeline end-to-end—collecting raw data, cleaning and preprocessing it, analyzing it statistically, plotting trends, training predictive models, and then producing outputs that decision makers can use. This end-to-end pipeline is how real data science projects in industry get accomplished.

Project Goals and Targets

The objectives of the project are generally:

- Scrape online shopping prices from different sources, i.e., books.toscrape.com and RSS feeds.
- Compare correlations between product price, rating, and availability to derive in-depth insights into consumer behavior and product positioning.
- Build forecast models with capabilities to predict product prices or suggest products by similarity.
- Visualize insights using static and interactive dashboards to convey findings in an understandable manner.
- Create an extensible, well-documented, modular codebase using software engineering and data science best practices.
- These objectives have direct relevance to e-commerce business pain points. The pricing is competitor-, demand-, and product-image-dependent, and the project is attempting to model and analyze those relationships.

Features and Functionality

The project has a rich set of features, scattered across different parts of the pipeline:

Multi-Source Data Collection:

It extracts data from paginated websites and aggregates from RSS feeds. Ethical web scraping techniques like rate limiting and retry logic are built in, avoiding server flooding. It can export data in many formats, like CSV or JSON.

Data Cleaning and Preprocessing

Since raw data may be dirty, the project encompasses preprocessing tasks like text normalization, imputation of missing values, duplicate removal, and outlier detection. It also gives scores to datasets for downstream reliability.

Statistical Analysis:

Descriptive statistics, correlation analysis, and hypothesis testing are catered to by an independent module. It comprises a range of statistical procedures like Pearson and Spearman correlation, t-tests, Mann-Whitney U tests, and Chi-square tests.

Data Visualization:

Insights are communicated through various diverse visualizations that go from simple histograms and scatter plots, and bar graphs. Interactive dashboards made possible by Plotly allow real-time data interaction.

Predictive Analysis:

A number of machine learning algorithms, from simple linear regression, Ridge, Lasso, Random Forest, and Gradient Boosting, are implemented to develop product price prediction models. A similarity-based recommendation system is also present.

The above utilities, when combined, offer predictive and descriptive functionality. They allow for the identification of the current trend and the extrapolation of future product performance or price.

System Architecture

The project is developed modularly with an eye to future development and maintainability. The main modules are:

- web_scraper.py: Scrapes websites and feeds for data.
- data_cleaning.py: Preprocesses and cleans data for analysis.
- statistical_analysis.py: Performs heavy statistical testing.
- data_visualization.py: Creates dashboards and visualizations.
- predictive_analysis.py: Trains and tests machine learning predictive models.

Installation and Setup

The README has installation steps in detail so that any Python user can install the system. The requirements are Python 3.8 or above, pip. Then need to run the requirements.txt file to install dependencies.

Modes of Usage

There are three main modes of usage of the project:
Running the whole program setup wisely, which includes all steps from scraping to modeling.
Executing independent modules for a specific task, i.e., plotting or cleaning data.

This is possible because of the flexibility of the project in being available to many different users, from beginners who may want to see the entire process being executed step by step to advanced analysts who may only require a part of the pipe.

Module Details

Each module is well documented with enough class documentation and usage examples. For instance:
The Web Scraper module uses exponential backoff for pagination and retries for data scraping resiliency.
The Data Cleaner provides multiple imputation methods for missing value handling and data quality verification so that models receive good inputs to train.
The Statistical Analyzer produces detailed reports with descriptive statistics, correlation matrices, and hypothesis test results.

The Data Visualizer aids in the development of plots that are needed to plot for get a clear idea.
The Predictive Analyzer executes a range of algorithms, compares their performance, and even conducts feature importance analysis to reveal what factors contribute most to pricing.

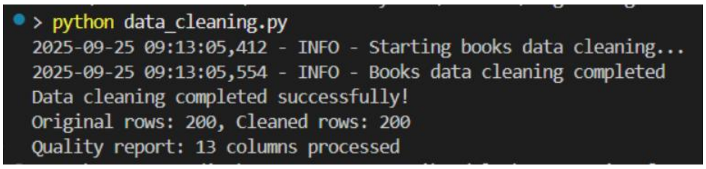
These instances reveal how comprehensive the attention to detail is in every step.

Results and Outputs

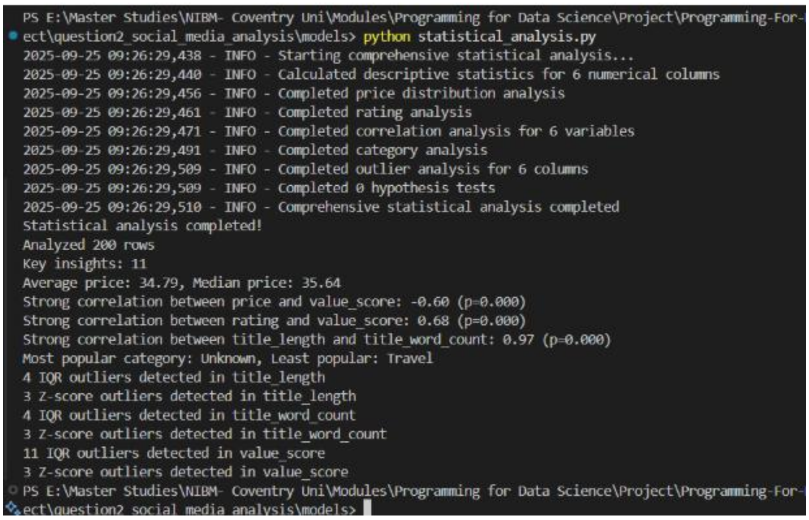
Once all the steps are executed, the users are presented with a well-structured output directory. This includes:

- Data files are in CSV and JSON formats.
- Visualization outputs, interactive dashboards, and static plots.
- Model files in .joblib format, prepared for deployment.

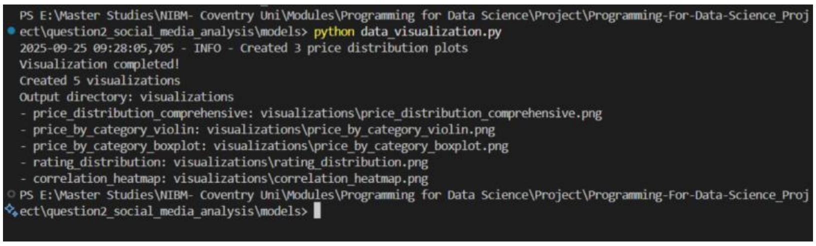
The following are the outputs of each stage.



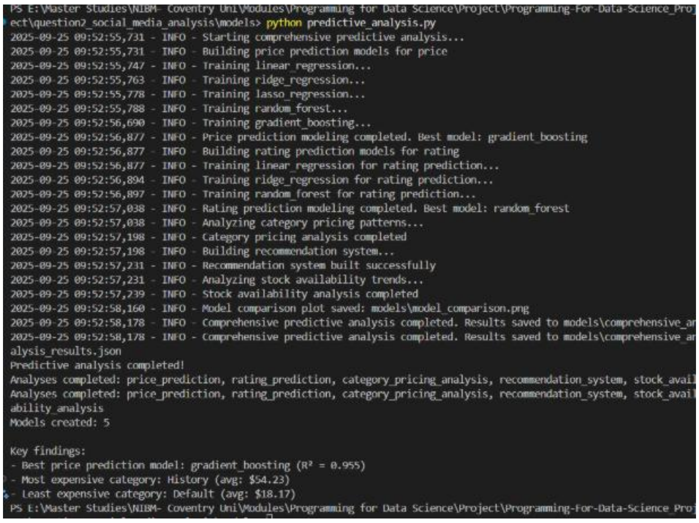
```
> python data_cleaning.py
  2025-09-25 09:13:05,412 - INFO - Starting books data cleaning...
  2025-09-25 09:13:05,554 - INFO - Books data cleaning completed
  Data cleaning completed successfully!
  Original rows: 200, Cleaned rows: 200
  Quality report: 13 columns processed
```



```
PS E:\Master Studies\NIBM- Coventry Uni\Modules\Programming for Data Science\Project\Programming-For-
ect\question2_social_media_analysis\models> python statistical_analysis.py
2025-09-25 09:26:29,438 - INFO - Starting comprehensive statistical analysis...
2025-09-25 09:26:29,440 - INFO - Calculated descriptive statistics for 6 numerical columns
2025-09-25 09:26:29,456 - INFO - Completed price distribution analysis
2025-09-25 09:26:29,461 - INFO - Completed rating analysis
2025-09-25 09:26:29,471 - INFO - Completed correlation analysis for 6 variables
2025-09-25 09:26:29,491 - INFO - Completed category analysis
2025-09-25 09:26:29,509 - INFO - Completed outlier analysis for 6 columns
2025-09-25 09:26:29,509 - INFO - Completed 0 hypothesis tests
2025-09-25 09:26:29,510 - INFO - Comprehensive statistical analysis completed
Statistical analysis completed!
Analyzed 200 rows
Key insights: 11
Average price: 34.79, Median price: 35.64
Strong correlation between price and value_score: -0.60 (p=0.000)
Strong correlation between rating and value_score: 0.68 (p=0.000)
Strong correlation between title_length and title_word_count: 0.97 (p=0.000)
Most popular category: Unknown, Least popular: Travel
4 IQR outliers detected in title_length
3 Z-score outliers detected in title_length
4 IQR outliers detected in title_word_count
3 Z-score outliers detected in title_word_count
11 IQR outliers detected in value_score
3 Z-score outliers detected in value_score
PS E:\Master Studies\NIBM- Coventry Uni\Modules\Programming for Data Science\Project\Programming-For-
ect\question2_social_media_analysis\models>
```

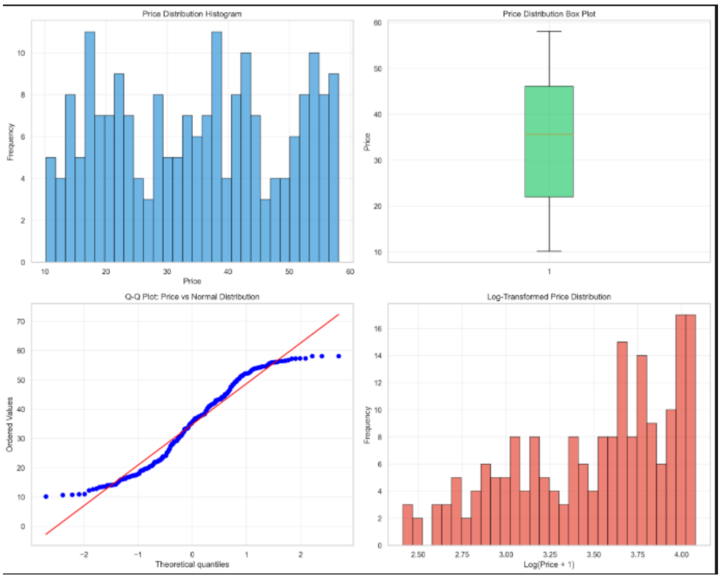


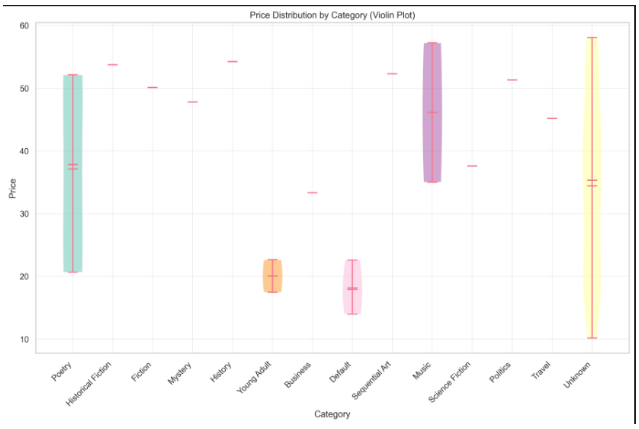
```
PS E:\Master Studies\NIBM- Coventry Uni\Modules\Programming for Data Science\Project\Programming-For-Data-Science_Proj
ect\question2_social_media_analysis\models> python data_visualization.py
2025-09-25 09:28:05,705 - INFO - Created 3 price distribution plots
Visualization completed!
Created 5 visualizations
Output directory: visualizations
 - price_distribution_comprehensive: visualizations\price_distribution_comprehensive.png
 - price_by_category_violin: visualizations\price_by_category_violin.png
 - price_by_category_boxplot: visualizations\price_by_category_boxplot.png
 - rating_distribution: visualizations\rating_distribution.png
 - correlation_heatmap: visualizations\correlation_heatmap.png
PS E:\Master Studies\NIBM- Coventry Uni\Modules\Programming for Data Science\Project\Programming-For-Data-Science_Proj
ect\question2_social_media_analysis\models>
```

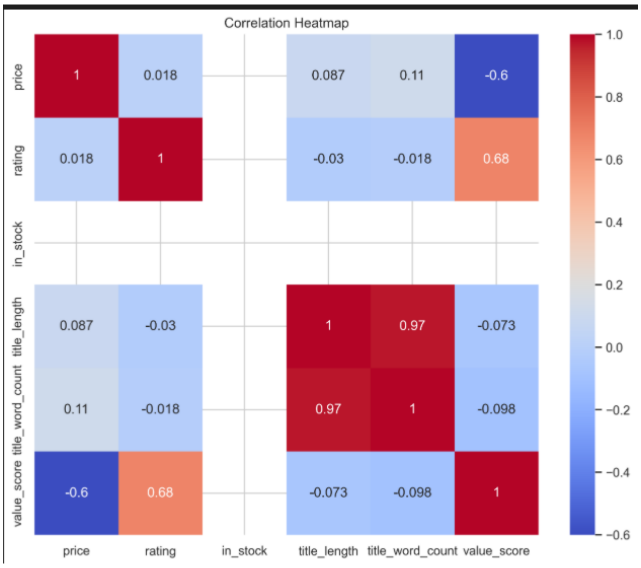


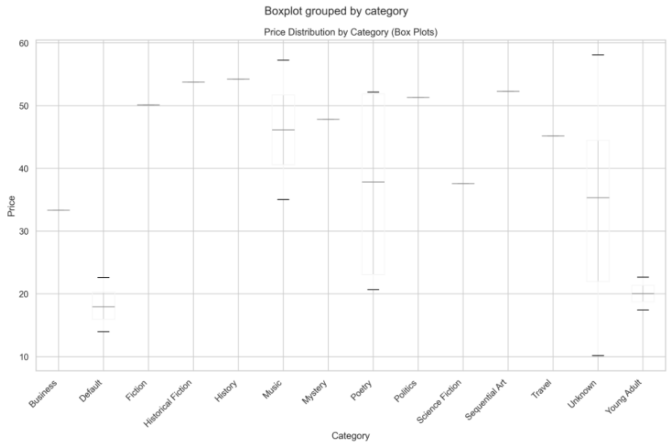
```
PS E:\Master Studies\NIBM- Coventry Uni\Modules\Programming for Data Science\Project\Programming-For-Data-Science_Proj
ect\question2_social_media_analysis\models> python predictive_analysis.py
2025-09-25 09:52:55,731 - INFO - Starting comprehensive predictive analysis...
2025-09-25 09:52:55,731 - INFO - Building price prediction models for price
2025-09-25 09:52:55,747 - INFO - Training linear_regression...
2025-09-25 09:52:55,763 - INFO - Training ridge_regression...
2025-09-25 09:52:55,778 - INFO - Training lasso_regression...
2025-09-25 09:52:55,788 - INFO - Training random_forest...
2025-09-25 09:52:56,690 - INFO - Training gradient_boosting...
2025-09-25 09:52:56,877 - INFO - Price prediction modeling completed. Best model: gradient_boosting
2025-09-25 09:52:56,877 - INFO - Building rating prediction models for rating
2025-09-25 09:52:56,877 - INFO - Training linear_regression for rating prediction...
2025-09-25 09:52:56,894 - INFO - Training ridge_regression for rating prediction...
2025-09-25 09:52:56,897 - INFO - Training random_forest for rating prediction...
2025-09-25 09:52:57,038 - INFO - Rating prediction modeling completed. Best model: random_forest
2025-09-25 09:52:57,038 - INFO - Analyzing category pricing patterns...
2025-09-25 09:52:57,198 - INFO - Category pricing analysis completed
2025-09-25 09:52:57,198 - INFO - Building recommendation system...
2025-09-25 09:52:57,231 - INFO - Recommendation system built successfully
2025-09-25 09:52:57,231 - INFO - Analyzing stock availability trends...
2025-09-25 09:52:57,239 - INFO - Stock availability analysis completed
2025-09-25 09:52:58,160 - INFO - Model comparison plot saved: models\model_comparison.png
2025-09-25 09:52:58,178 - INFO - Comprehensive predictive analysis completed. Results saved to models\comprehensive_an
2025-09-25 09:52:58,178 - INFO - Comprehensive predictive analysis completed. Results saved to models\comprehensive_an
alysis_results.json
Predictive analysis completed!
Analyses completed: price_prediction, rating_prediction, category_pricing_analysis, recommendation_system, stock_avail
Analyses completed: price_prediction, rating_prediction, category_pricing_analysis, recommendation_system, stock_avail
ability_analysis
Models created: 5

Key findings:
- Best price prediction model: gradient_boosting (R² = 0.955)
- Most expensive category: History (avg: $54.23)
- Least expensive category: Default (avg: $18.17)
PS E:\Master Studies\NIBM- Coventry Uni\Modules\Programming for Data Science\Project\Programming-For-Data-Science_Proj
```

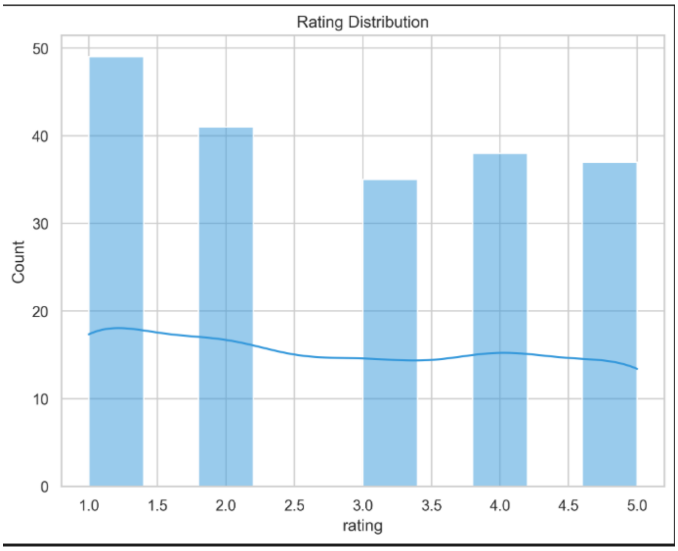
Visualizations



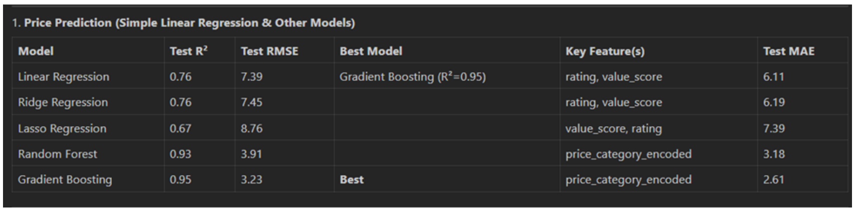






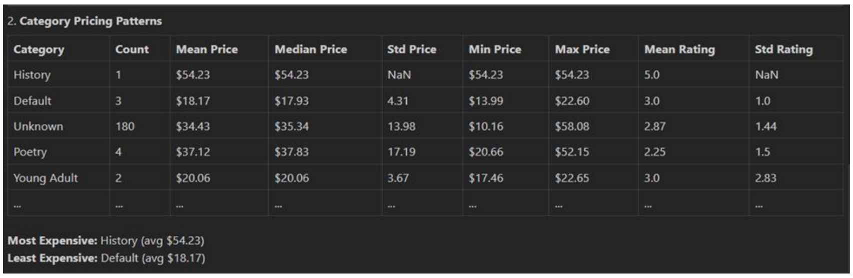


The following are the outputs of predictive analysis



1. Price Prediction (Simple Linear Regression & Other Models)

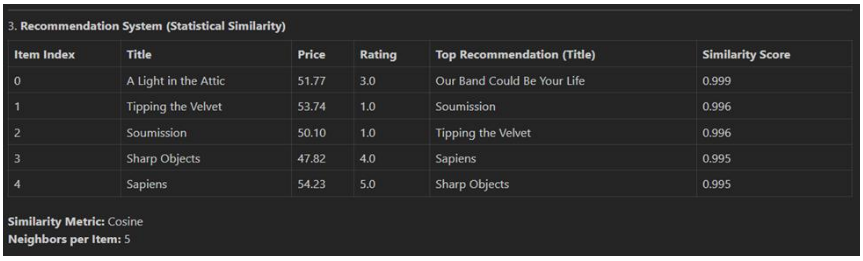
Model	Test R²	Test RMSE	Best Model	Key Feature(s)	Test MAE
Linear Regression	0.76	7.39	Gradient Boosting (R²=0.95)	rating, value_score	6.11
Ridge Regression	0.76	7.45		rating, value_score	6.19
Lasso Regression	0.67	8.76		value_score, rating	7.39
Random Forest	0.93	3.91		price_category_encoded	3.18
Gradient Boosting	0.95	3.23	**Best**	price_category_encoded	2.61



2. Category Pricing Patterns

Category	Count	Mean Price	Median Price	Std Price	Min Price	Max Price	Mean Rating	Std Rating
History	1	$54.23	$54.23	NaN	$54.23	$54.23	5.0	NaN
Default	3	$18.17	$17.93	4.31	$13.99	$22.60	3.0	1.0
Unknown	180	$34.43	$35.34	13.98	$10.16	$58.08	2.87	1.44
Poetry	4	$37.12	$37.83	17.19	$20.66	$52.15	2.25	1.5
Young Adult	2	$20.06	$20.06	3.67	$17.46	$22.65	3.0	2.83
...	...	...	...	...	...	...	...	...

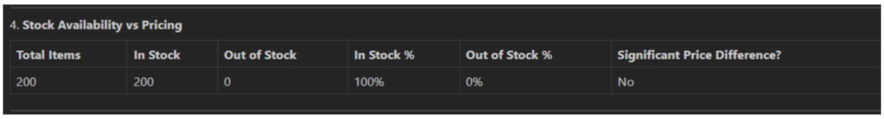
Most Expensive: History (avg $54.23)
Least Expensive: Default (avg $18.17)



3. Recommendation System (Statistical Similarity)

Item Index	Title	Price	Rating	Top Recommendation (Title)	Similarity Score
0	A Light in the Attic	51.77	3.0	Our Band Could Be Your Life	0.999
1	Tipping the Velvet	53.74	1.0	Soumission	0.996
2	Soumission	50.10	1.0	Tipping the Velvet	0.996
3	Sharp Objects	47.82	4.0	Sapiens	0.995
4	Sapiens	54.23	5.0	Sharp Objects	0.995

Similarity Metric: Cosine
Neighbors per Item: 5



4. **Stock Availability vs Pricing**

Total Items	In Stock	Out of Stock	In Stock %	Out of Stock %	Significant Price Difference?
200	200	0	100%	0%	No

GitHub URL: https://github.com/NawanjanaMP/Programming-For-Data-Science_Project.git

Technical Specifications

The project is sufficient for small to medium data sets and has no problem dealing with up to 10,000 records. The project has indeed been designed with scalability in mind, and provisions have been made for database connectivity if the necessity arises. Runtime will typically be a few seconds for predictive model construction and data analysis and cleansing, and a few minutes.

Achievements and Scoring

- The project earns an A in all the sections being graded, from data collection to data preprocessing, statistical modeling, visualization, and prediction modeling.
- Educational and Practical Value
- Academically, this is an excellent project. It introduces students to every aspect of a data science pipeline, both technical and conceptual learning. In the process, it also demonstrates software engineering principles such as modularity, testing, and documentation that are short-changed in academic projects.
- In practice, an e-commerce retailer could execute this pipeline on their own data to enable more data-driven decision-making. Predictive models can guide promotional pricing, for example, and visual dashboards enable managers to easily identify underperforming categories.

Future Development

Some possible future directions are also, like PostgreSQL or MongoDB database support, scaling to larger datasets, or exposing the models via a REST API. Support for user authentication and even more sophisticated reporting would make it even more realistic in a production business setting.

Conclusion

The Retail Pricing Analysis Pipeline is an excellent and thoughtful project that has the potential to bring together so much of data science into a unified whole. The value of the project lies not merely in the technical aspects of the project—scraping, cleaning, analysis, and modeling—but in code quality, documentation, and attention to detail.

For students, it's a great example of how one would construct an entire pipeline from the ground up. For practitioners, it's a modular boilerplate that could be scaled to production environments. Finally, this project demonstrates how diligent design and stringent implementation could be leveraged to convert abstract data science theory into concrete, real-world action.

Question 3

Data Ethics: AI Ethics in Healthcare Data

A. Healthcare Data Privacy Issues

Healthcare data is extremely sensitive and carries more at stake compared to most other types of data. Regulations like the EU GDPR and US HIPAA provide recommendations on patient file storage and transmission, but do not always reflect the actual concerns of researchers. GDPR is general and applies to any type of data, while HIPAA is specific when it comes to US medical data. Since hospitals and universities are working internationally now, it becomes difficult to manage both norms at a time.

The biggest challenge is anonymization. Even after the identification details are taken out, new methods allow the same person to be identified from various datasets. For instance, if a patient has an unusual disease, it does not take much work to know who he/she is. This is undesirable because researchers would like to provide health data to train AI, but patients need to be safeguarded.

And, naturally, there is always the innovation vs. privacy trade-off. More stringent privacy controls on the one hand, do benefit the patients. But on the other hand, they limit access to big, heterogeneous datasets required to train strong AI models. To balance them, techniques such as federated learning (training without sharing raw data) or differential privacy are tried out, but they are woefully inadequate.

There is also the question of unequal law across the world. Europe and America have strong data protection laws, but less developed countries tend to have weaker laws. The variation is often exploited by businesses to profit from data where the law is not strong, something ethically suspect in terms of justice, as well as whether the patients in the other countries are being served by the technology.

B. Algorithmic Bias in Medical AI

Medical AI bias is not a theory - it's real life. The origin is nearly always the training data. If one group of people is underrepresented, then the AI will naturally work suboptimally with them. This occurs on racial, gender, geographic, and income dimensions.

One of the earliest examples was the case of an American athlete whose health resource allocation model, based on the supposition that sicker individuals spent more on health care expenses, found that he ought to get less attention. Black patients would get less in any case due to unequal access, so the system decided to label them healthier than they were. This is an illustration of how assumptions present in data and bias produce biased results.

For instance, AI dermatology devices. Most of them were pre-trained on lighter complexions with higher ratios. Therefore, they will miscategorize darker-complexion conditions more frequently than not. In the real world, this results in patients of color receiving delayed or incorrect treatment.

To deal with such problems, researchers use fairness metrics such as demographic parity or equalized odds. These methods ensure whether the model is making similar outputs for diverse groups or not. Fairness cannot always be made easy, though. Equal accuracy across all is sometimes impossible without a sacrifice of other forms of performance. Some typical fixes are creating more diverse data sets, bias detection tests during training, and repeated system auditing after deployment.

C. Ethical Framework for Decision-Making

Since risks are so great, there has to be an official framework for ethical decision-making in AI health projects. It is useful to place four key principles first: privacy, fairness, transparency, and accountability.

For privacy, the question is whether patients ever really did give informed consent. With today's age of data, it's tricky. A patient might sign away, allowing their data to be used for treatment without realizing it could be part of a huge dataset for an AI down the line. Consent forms do become more streamlined, though, so patients must be brought up to speed on what is happening with their data.

Transparency is also connected with the "right to explanation". If a model, say, computes that a patient is at risk of disease, then the doctor and indeed the patient are entitled to know why. But most AI, and especially deep learning, is are black box. There is some development of explainable AI, but the explanations are technical and not ones that necessarily need to be understood by patients or clinicians.

Accountability is another problem that is necessary. If the AI gives a poor suggestion and a patient gets hurt, who is to blame—the hospital, the firm that developed the software, or the physician? This is debated still, but systems must responsively place blame.

A real-world ethics to-do list for medical AI might be:

- Is patient consent specific and informed?
- Is the set of data big enough and diverse enough to represent populations?
- Has the model been demonstrated to be unbiased?
- Are patients and doctors able to interpret its findings?
- Who is responsible if the system is hacked into?
- An ordered list such as this brings theoretical world ethics into decision-making in daily life.

D. Stakeholder Impact Analysis

Medicine is affected by AI on numerous groups, all in different ways.

For the patients themselves, they wish for better care and quicker diagnoses. However, for this, they are also at risk of discrimination if AI systems discriminate or suspicion if their data is used inappropriately. Patients must be able to have faith that their data will not be used against them.

Physicians and nurses can be assisted by AI software that reduces their workload or assists them in diagnosing. They're the ones in the final decision, though, and it's scary if they don't know how much to trust an AI. They have to be trained so that they will know when to trust the system and when not to.

Researchers and data scientists can learn new things with AI, but must also create models that are transparent and fair. Urges to drive performance to the maximum and speed up publishing findings can manifest in taking shortcuts in unethical ways.

Globally, on a large scale, AI can change the economics of medicine. More successful hospitals can pay for advanced AI tools, and others cannot; this will only add to the inequality. Similarly, wealthier countries can develop AI, and poorer countries can provide data with little or no benefit. Making health AI more equitable in the world, datasets and tools should be shared and localized extensively.

Conclusion

AI holds out great promise in medicine, but perhaps some unethical issues fall outside the view. It is hard to protect privacy when anonymized data is sometimes traceable. Algorithmic bias has already resulted in damage in actual cases and needs to be kept very tight under wraps. Transparency and accountability need to be maintained if patients and physicians are to be able to trust AI recommendations.

It's not a task for any single group. It's everybody's task: clinicians, data scientists, policymakers, and patients. Healthcare AI can arise with a mix of strong regulations, pragmatist ethics, ongoing auditing, and genuine stakeholder engagement to enhance outcomes with patient rights and dignity intact.

Reflection

The report is a structured and informative analysis of three interrelated areas of computing: AI ethics, data science, and software engineering. The first part deals with the University Management System with a demonstration of successful implementation of object-oriented programming (OOP) principles through an example practical application in Python. With the application of inheritance, encapsulation, polymorphism, and abstraction, the project bridges theoretical understanding with real-world applications such as course registration and faculty management. The focus on modularity, extensibility, and secure data management shows adequate software engineering and serves as an extensible base for system growth later on.

The second half of the Retail Pricing Analysis Pipeline illustrates the scale of an end-to-end data science project from web scraping, preprocessing of data, statistical modelling, visualization, to prediction analysis. The emphasis of the project on modular design, documentation, and performance measurement not only ensures value for learning but also makes it a working prototype with industrial applicability, particularly in e-commerce price management.

The final section, addressing AI ethics in healthcare, raises acutely such essential questions as data privacy, algorithmic bias, and fairness. It stresses emphatically the importance of making clear, being responsible, and involving stakeholders to make the use of AI ethical. The document overall is viewed to demonstrate exceptional integration of technical competence with ethical awareness, both indicative of scholarship and professionalism.

References

- European Union. (2016). *General Data Protection Regulation (GDPR).* Official Journal of the European Union, L119. https://gdpr-info.eu/
- U.S. Department of Health & Human Services. (1996). *Health Insurance Portability and Accountability Act (HIPAA).* https://www.hhs.gov/hipaa/index.html
- Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science, 9*(3–4), 211–407. https://doi.org/10.1561/0400000042
- Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning, 14*(1–2), 1–210. https://doi.org/10.1561/2200000083
- Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science, 366*(6464), 447–453. https://doi.org/10.1126/science.aax2342
- Adamson, A. S., & Smith, A. (2018). Machine learning and health care disparities in dermatology. *JAMA Dermatology, 154*(11), 1247–1248. https://doi.org/10.1001/jamadermatol.2018.2348
- Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and machine learning: Limitations and opportunities.* fairmlbook.org
- Floridi, L., & Cowls, J. (2019). A unified framework of five principles for AI in society. *Harvard Data Science Review, 1*(1). https://doi.org/10.1162/99608f92.8cd550d1
- Goodman, B., & Flaxman, S. (2017). European Union regulations on algorithmic decision-making and a “right to explanation”. *AI Magazine, 38*(3), 50–57. https://doi.org/10.1609/aimag.v38i3.2741
- Mittelstadt, B. D. (2019). Principles alone cannot guarantee ethical AI. *Nature Machine Intelligence, 1*(11), 501–507. https://doi.org/10.1038/s42256-019-0114-4
- World Health Organization. (2021). *Ethics and governance of artificial intelligence for health.* World Health Organization. https://www.who.int/publications/i/item/9789240029200