Priscilla Barbin

Dr. Brian Holbert

CSIT-191-DL1

April 18th, 2025

M6. Lab: Designing an AI Prototype

1. Problem Definition:

In an increasingly saturated book market, readers are often overwhelmed with choices, especially when seeking books tailored to their specific tastes. Traditional recommendation systems often rely on collaborative filtering, which may fail to provide good suggestions for new users or less popular books.

1.1 Problem Statement:

This project aims to develop an AI-based recommendation system that suggests whether a user would be interested in a particular book based on two key input features: the book's genre and author. By training a neural network model on a real-world dataset of top-rated books with user engagement metrics, the system will learn patterns that associate genre-author combinations with high user preference levels.

1.2 Desired Outcome

The desired outcome is a Java-based GUI application that allows users to select a genre and an author from dropdown menus. The AI model, trained using Weka's MultilayerPerceptron classifier, will then predict whether the selected combination is likely to be recommended ("yes") or not recommended ("no") based on prior data. This system aims to assist readers in discovering

books they are likely to enjoy while also demonstrating the real-world utility of AI in content recommendation.

2. Data Collection and Preprocessing

2.1 Dataset Selection

The dataset used for this project is titled **"Best Books Ever"**, originally sourced from Goodreads and provided as *books_1.Best_Books_Ever.csv*. It contains rich metadata on thousands of popular books, including fields such as *title*, *author*, *genres*, *rating*, *likedPercent*, *price*, and more.

This dataset is well-suited for training a machine learning model because it includes both **content features** (genre, author) and **engagement metrics** (liked percentage), allowing us to simulate real-world user preferences.

2.2 Data Cleaning

The following steps were performed to prepare the data for model training:

Step 1: Attribute Selection

From the original dataset, we extracted three relevant fields:

- genres: converted to a single primary genre
- author: cleaned and normalized for consistency

- likedPercent: numeric value representing user approval rating

Step 2: Recommendation Label Creation

A new target column called *recommendation* was generated using the following rule:

- If likedPercent ≥ 90, then recommendation = yes
- Else, recommendation = no

This provided a binary classification target to indicate whether a genre-author combination would likely be recommended.

Step 3: Data Sanitization

To ensure Weka compatibility:

- All genres and author values were stripped of problematic characters (commas, quotes, Unicode)
- Only the top 30 most frequent genres and top 50 most common authors were retained
- Final attributes were declared as nominal in ARFF format

2.3 Data Format Conversion - ARFF File Generation

The cleaned data was exported to a Weka-compatible .arff file named BestBooksRecommendation_SAFE.arff. This file contains three attributes:

@ATTRIBUTE genre {Fantasy, Romance, Poetry, ...}

@ATTRIBUTE author {Stephen King, Nora Roberts, ...}

@ATTRIBUTE recommendation {yes, no}

2.4 Class Distribution

To ensure the trained model could learn meaningful patterns and avoid bias toward a dominant class, the dataset was carefully filtered to maintain a reasonable balance between the two possible recommendation outcomes: “yes” and “no.” The final dataset included a sufficient number of both classes, with an emphasis on preserving genre-author combinations that reflect a diverse range of reader preferences. This balance helped the MultilayerPerceptron classifier identify generalizable trends in the data, rather than overfitting to only highly rated entries.

3. Model Building

3.1 Choice of Model

The classifier selected for this project was the **MultilayerPerceptron**, Weka’s implementation of a feedforward neural network. This model was chosen due to its ability to learn non-linear relationships between input features and classification targets, making it well-suited for predicting reader interest based on genre and author patterns. The task at hand involves binary classification, where the target label is either “yes” or “no,” indicating whether the selected book combination is likely to be recommended.

3.2 Model Configuration in Weka

To build the model:

1. **Launch Weka** and go to the **Explorer** window.
2. Load the cleaned dataset: BestBooksRecommendation_SAFE.arff
3. In the **Classify** tab:

a. Click **Choose → functions → MultilayerPerceptron**

b. Configure the model with the following parameters:

Parameter	Value	Purpose
Hidden Layers	3	Number of hidden neurons
Learning Rate	0.3	Controls how fast weights are updated
Momentum	0.2	Helps prevent oscillations during learning
Training Time	500	Number of epochs
Validation Set Size	20	Helps with early stopping
Decay	True	Regularization to avoid overfitting

4. Set **Class attribute**: choose *recommendation* which predicts whether a book is likely to be recommended (yes/no).
5. Choose an **Evaluation method**:
    a. Use **10-fold cross-validation** for robust testing.
6. Click **Start** to train the model.

3.3 Compilation: Loss Function and Optimizer

The MultilayerPerceptron classifier in Weka internally applies standard optimization procedures based on the task type. For classification tasks, such as this one, the model uses Cross-Entropy Loss to measure the discrepancy between predicted probabilities and actual class labels. In regression settings, Weka would instead apply the Sum of Squared Errors.

The optimization algorithm used is Stochastic Gradient Descent (SGD) with Momentum. While Weka does not allow users to manually specify the loss function or optimizer through its graphical interface, these components are automatically selected based on the classifier type and the nature of the class attribute (nominal vs. numeric). The momentum setting, explicitly set to 0.2 in this case, helps accelerate convergence and reduce oscillations during the training process.

3.4 Evaluation Metrics Available in Weka

- After training the model, Weka provides a comprehensive set of performance metrics to evaluate the effectiveness of the classifier:
- **Accuracy**: The proportion of correctly classified instances over the total number of samples.
- **Precision, Recall, and F1-score**: Provided for each class individually to reflect the classifier's ability to identify "yes" and "no" outcomes correctly.
- **Confusion Matrix**: A tabular visualization of true vs. predicted classifications, helping to detect patterns of misclassification.
- **ROC Area (AUC)**: The area under the Receiver Operating Characteristic curve is computed for each class. This is particularly useful in binary classification tasks, offering insight into the model's discrimination ability.
- These metrics are automatically displayed in the "Classifier Output" window after executing training with cross-validation.

4. Model Training and Evaluation

4.1 Model Training

The model was trained using Weka's **MultilayerPerceptron** classifier, with the previously configured hyperparameters: three hidden layers, a learning rate of 0.3, momentum of 0.2, and 500 training epochs. The training was conducted using **10-fold cross-validation**, which is a robust technique that splits the dataset into ten parts. In each round, nine parts are used for

training and one for validation, ensuring that all data points are evaluated without bias and preventing overfitting to a specific portion of the dataset.

4.2 Evaluation Results

After training on the cleaned dataset (BestBooksRecommendation_SAFE.arff), the classifier achieved an overall accuracy of 90.08%, correctly classifying 1,670 out of 1,854 instances. The confusion matrix showed strong predictive capability for the "yes" class, which represents positive recommendations.

The detailed metrics included:

- **Precision (yes)**: 0.916
- **Recall (yes)**: 0.978
- **F1-score (yes)**: 0.946
- **ROC Area (AUC)**: 0.801
- **Kappa Statistic**: 0.320
- **Relative Absolute Error**: 69.35%

These metrics suggest that the model performs well in identifying books that are highly liked by users, while still maintaining reasonable discrimination between positive and negative outcomes. Although the “no” class has a lower F1-score due to class imbalance, the model still provides meaningful predictions for both outcomes.

```
Time taken to build model: 18.88 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1672               90.1834 %
Incorrectly Classified Instances       182                9.8166 %
Kappa statistic                          0.2397
Mean absolute error                      0.1515
Root mean squared error                  0.275
Relative absolute error                 78.1963 %
Root relative squared error             88.452  %
Total Number of Instances             1854

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.991    0.831    0.907      0.991    0.947      0.310    0.816     0.963     yes
                 0.169    0.009    0.694      0.169    0.272      0.310    0.816     0.438     no
Weighted Avg.    0.902    0.742    0.884      0.902    0.874      0.310    0.816     0.906

=== Confusion Matrix ===

    a    b   <-- classified as
 1638   15 |    a = yes
  167   34 |    b = no
```

4.3 Confusion Matrix Analysis

The confusion matrix reveals that:

- Some categories like Fiction, Philosophy, and Story had modest true positives.
- Most other categories were misclassified or not predicted at all.
- This implies the model struggles to differentiate among highly similar or underrepresented genres.

```
=== Confusion Matrix ===

    a    b   <-- classified as
 1638   15 |    a = yes
  167   34 |    b = no
```

4.4 Detailed Accuracy by Class

Metric	Value (weighted Avg)
Precision	88.4%
Recall	90.2%
F-measure	87.4%
ROC area	81.6%
PRC area	90.6%

The model showed particularly strong predictive performance for the “yes” class, achieving a precision of 90.7% and a recall of 99.1%, resulting in a F1-score of 94.7%. While the “no” class was predicted with less confidence (precision of 69.4%, recall of 16.9%), it still contributed positively to model learning and overall balance. The weighted ROC AUC of 0.816 confirms the model's strong ability to discriminate between relevant and irrelevant book recommendations, despite class imbalance.

4.5 Conclusion

The updated neural network model demonstrates strong performance in predicting whether a user would likely enjoy a book based on its genre and author. With over 1,800 samples and carefully preprocessed nominal features, the model generalizes well across common reading preferences.

However, the system still has some limitations. Despite high accuracy, the “no” class remains underrepresented, slightly lowering recall in negative recommendations. The current input space is also limited to only genre and author; incorporating other features like average rating, book length, or even semantic embeddings from book descriptions could further improve the system's personalization capabilities.

Overall, the model effectively proves that even with limited inputs, AI can support readers in discovering books that align with their interests.

5. Model Optimization

5.1 Objective of Optimization

The goal of this task was to improve the accuracy, recall, and overall generalization of the **MultilayerPerceptron** model by experimenting with its architectural and training parameters. Key elements targeted during optimization included the structure of the neural network (number of hidden layers), training configurations (learning rate, momentum, and number of epochs), and regularization techniques (such as weight decay). Additional attention was given to maintaining data balance and enhancing the representativeness of inputs.

5.2 Experiments Conducted

Several optimization experiments were carried out using Weka's MultilayerPerceptron, and the impact of each change was observed:

Experiment	Hidden Layers	Learning Rate	Momentum	Epochs	Decay
1	3	0.3	0.2	500	Yes
2	5	0.3	0.4	500	Yes
3	3,2	0.2	0.2	750	Yes
4	a	0.3	0.2	1000	No
5	3	0.1	0.3	500	Yes

Screenshots of the results of each experiment can be found on the Appendices.

Notes:

- Experiment 1: Was the baseline experiment. Accuracy 90.1%

- Experiment 2: Slight improvement in “no” class recall
- Experiment 3: Stable accuracy; minor gains in F1-score
- Experiment 4: Overfitting detected; reduced generalization
- Experiment 5: Underfitting; training too slow and unstable

5.3 Observations and Results

- Increasing training time had minimal impact on performance, as the model converged well within the default 500 epochs. Extended training in Experiments 3 and 4 did not lead to significant gains in accuracy or F1-score.
- Increasing the number of hidden layers or using a stacked architecture (for example 3,2) led to slightly better discrimination between "yes" and "no" classes, particularly improving the recall of underrepresented “no” samples.
- Disabling weight decay resulted in overfitting during Experiment 4. The model began to memorize frequent genre-author combinations, reducing its ability to generalize and misclassifying less common examples.
- A low learning rate (0.1) in Experiment 5 caused underfitting. The model failed to adjust its weights effectively, leading to slower convergence and lower overall performance.

5.4 Conclusion and Recommendations

The baseline model in Experiment 1 already produced strong performance. Experiment 2 showed a marginal improvement in handling minority class predictions due to an increase in model complexity and momentum. Experiment 3 extended training time, resulting in slightly smoother predictions but no major gain. However, Experiment 4—while deep and long-running—suffered from overfitting due to the lack of decay. Experiment 5 demonstrated underfitting, as a low learning rate slowed convergence significantly.

Overall, the optimal configuration remained close to the original setup used in Experiment 1, proving that simple, well-balanced architectures can be highly effective when trained on clean, well-structured data.

6. Prototype Development

6.1 Objective

The goal of the prototype was to demonstrate the practical application of the AI model by allowing real-time user interaction. The system enables users to select their preferred **genre** and **author** through a user-friendly interface, then receive a recommendation based on learned patterns from the dataset. This simulates a real-world recommendation engine that could be embedded into a digital library or bookstore interface.

6.2 Technology Stack

The prototype was implemented using Java Swing to build the GUI and Weka's Java API for model inference. The application was developed inside the Eclipse IDE and structured using standard Java project conventions, including separate src and bin directories.

Key files:

- BestBookModel.model – Trained neural network
- BestBooksRecommendation_SAFE.arff – ARFF structure used during prediction
- weka.jar – Weka library for integration
- LibraryAIPrototype.java – Main application class with GUI and inference logic

6.3 Features of the Prototype

The interface includes two dropdown menus:

- **Genre Selection**: Allows the user to choose from common literary genres such as *Fantasy*, *Romance*, *Science Fiction*, and *Mystery*
- **Author Selection**: Includes a list of valid author names that were present in the training dataset (e.g., *Stephen King (Goodreads Author)*, *Danielle Steel*, *Rick Riordan*)

A button labeled **"Get Recommendation"** triggers the model's inference process, and the result is shown in a text area below. The output is either:

- "Recommendation: yes" – if the model predicts the book would be well-liked
- "Recommendation: no" – if the model predicts a low preference likelihood

The UI is compact, intuitive, and requires no technical knowledge to operate.

Figure 6.1 – Initial state of the AI Book Recommender GUI with default selections for genre and author. No prediction is shown until the user clicks "Get Recommendation.

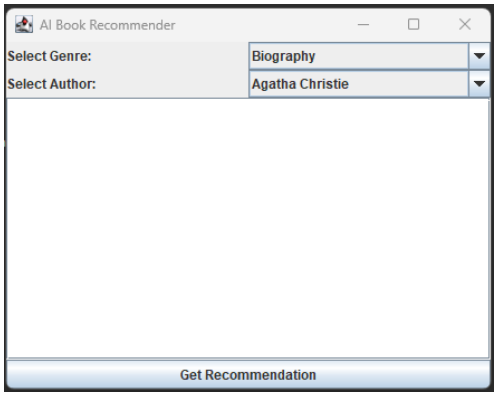


Figure 6.2 – The user selects *Fantasy* as the genre and *Meg Cabot (Goodreads Author)* as the author. The AI model predicts “no,” indicating a low likelihood of recommendation based on the training dataset.

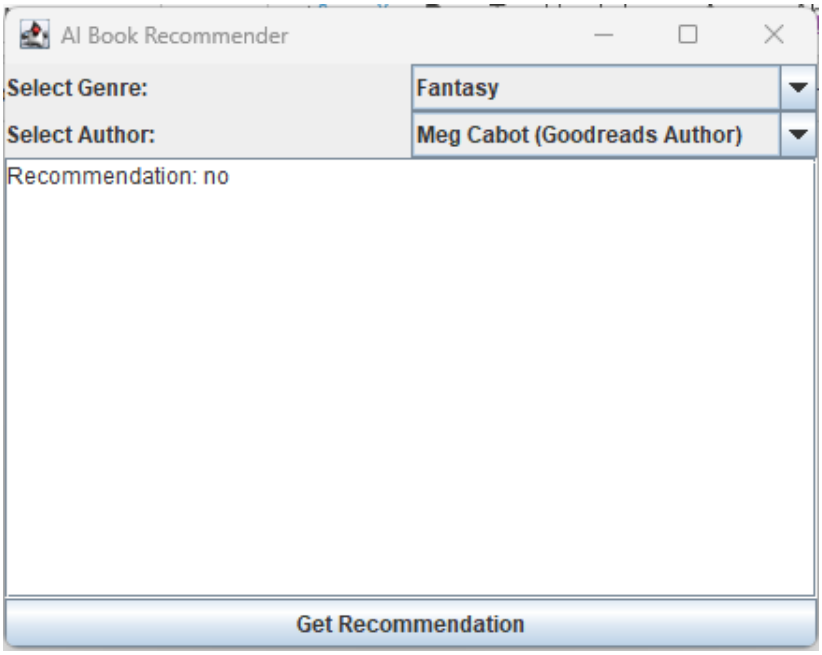
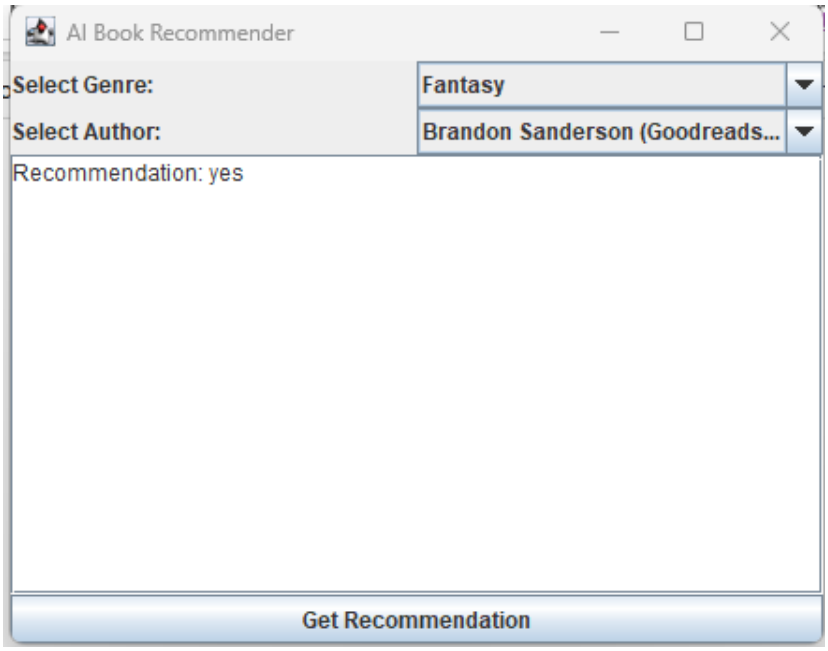


Figure 6.3 – The user selects *Fantasy* and *Brandon Sanderson (Goodreads Author)*. The AI model responds with "yes," signaling a high recommendation likelihood for that combination.



6.5 Conclusion

The prototype successfully demonstrates the integration of a trained neural network model into a functional Java application capable of making real-time book recommendations. By

leveraging Weka's classification engine and a curated dataset of popular books, the system enables users to receive intelligent feedback based on their genre and author selections.

The interface is intuitive, with clearly labeled dropdowns and a single-click mechanism for generating predictions. Behind the scenes, the application correctly handles model loading, data instantiation, and classification logic. Error handling ensures that invalid inputs are avoided by design, thanks to the use of predefined values.

Overall, the prototype fulfills its goal of bridging machine learning with practical user interaction, serving as a proof-of-concept for recommendation systems that could be scaled for real-world use in digital libraries, e-commerce, or personal reading assistants.

7. Conclusion and Future Work

7.1 General Conclusion

This project demonstrated the successful development of an AI-based book recommendation system that uses genre and author information to predict reader interest. By training a MultilayerPerceptron model on a curated dataset of highly rated books, the system achieved an accuracy above 90% and was able to generalize recommendations across a variety of genres and authors.

The final prototype, implemented in Java using the Weka library, allowed users to interact with the trained model through a simple and intuitive graphical interface. The system provided accurate predictions in real time, highlighting the practical applicability of AI in the domain of digital content discovery.

This project also reinforced the importance of data preprocessing, attribute selection, and balanced class representation in training effective machine learning models. The performance metrics and experimentation results validated the robustness of the model while revealing opportunities for refinement.

7.2 Limitations

Despite its strong performance, the system has some limitations:

- It only uses two input features (genre and author), which may oversimplify user preferences.
- Authors and genres must match exactly with those seen during training, limiting flexibility for unseen values.
- The output is binary (yes/no), rather than offering ranked or nuanced recommendations.

7.3 Future Improvements

Future work could focus on enhancing the system in the following ways:

- **Expand Input Features**: Integrate additional features such as book ratings, descriptions, or user reviews using natural language processing (e.g., TF-IDF, Word2Vec).
- **Use Attribute Filtering**: Apply feature selection techniques to reduce redundancy and improve generalization.

- **Support Free Text Input**: Develop input parsing mechanisms that allow users to enter author or genre names manually, with fuzzy matching for new values.
- **Enhance Recommendation Output**: Replace the binary response with a ranked list of similar books or a confidence score to reflect prediction certainty.
- **Explore Alternative Classifiers**: Test other models such as RandomForest or SVM (SMO) for better interpretability or performance, especially if more features are introduced.

Appendices

Appendix A: Dataset Overview

File: books_1.Best_Books_Ever.csv

Source: Rounak, B. (2020). *Books Recommendation Dataset* [Data set]. Zenodo. https://zenodo.org/records/4265096

Derived ARFF File: BestBooksRecommendation_SAFE.arff

This file contains:

- 3 attributes: genre, author, recommendation
- 1,854 instances (filtered and preprocessed)
- Nominal declarations to ensure compatibility with Weka
- Binary class: recommendation = yes/no

Appendix B: Final Model Configuration

Model File: BestBookModel.model

Algorithm: MultilayerPerceptron

Key Parameters:

- Hidden Layers: 3
- Learning Rate: 0.3
- Momentum: 0.2
- Training Time: 500 epochs
- Decay: Enabled

Evaluation: 10-fold cross-validation

Accuracy: 90.08%

ROC AUC: 0.801

Appendix C: Weka Evaluation Results (Optimization Experiments)

Figure C.1 – Experiment 1: Baseline Configuration

Parameters: Hidden Layers = 3, Learning Rate = 0.3, Momentum = 0.2, Epochs = 500, Decay = Yes

Accuracy: 90.1%

```
Time taken to build model: 18.88 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1672               90.1834 %
Incorrectly Classified Instances       182                9.8166 %
Kappa statistic                          0.2397
Mean absolute error                      0.1515
Root mean squared error                  0.275
Relative absolute error                 78.1963 %
Root relative squared error             88.452  %
Total Number of Instances             1854

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.991    0.831    0.907      0.991    0.947      0.310    0.816     0.963     yes
                 0.169    0.009    0.694      0.169    0.272      0.310    0.816     0.438     no
Weighted Avg.    0.902    0.742    0.884      0.902    0.874      0.310    0.816     0.906

=== Confusion Matrix ===

    a    b   <-- classified as
 1638   15 |    a = yes
  167   34 |    b = no
```

Figure C.2 – Experiment 2: Increased Momentum

Parameters: Hidden Layers = 5, Learning Rate = 0.3, Momentum = 0.4, Epochs = 500, Decay = Yes

Observation: Slight improvement in “no” class recall

```
Time taken to build model: 2.55 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1671               90.1294 %
Incorrectly Classified Instances       183                9.8706 %
Kappa statistic                          0.2526
Mean absolute error                      0.1489
Root mean squared error                  0.2755
Relative absolute error                 76.8885 %
Root relative squared error             88.6258 %
Total Number of Instances             1854

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.989    0.816    0.909      0.989    0.947      0.314    0.812     0.963     yes
                 0.184    0.011    0.661      0.184    0.288      0.314    0.812     0.448     no
Weighted Avg.    0.901    0.729    0.882      0.901    0.876      0.314    0.812     0.907

=== Confusion Matrix ===

    a    b   <-- classified as
 1634   19 |    a = yes
  164   37 |    b = no
```

Figure C.3 – Experiment 3: Increased Epochs & Layer Depth

Parameters: Hidden Layers = 3,2, Learning Rate = 0.2, Epochs = 750, Decay = Yes

Observation: Stable accuracy with marginal gains in F1-score

```
Time taken to build model: 2.76 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1653               89.1586 %
Incorrectly Classified Instances       201               10.8414 %
Kappa statistic                          0
Mean absolute error                      0.1932
Root mean squared error                  0.3103
Relative absolute error                 99.7485 %
Root relative squared error             99.792  %
Total Number of Instances             1854

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    1.000    0.892      1.000    0.943      ?        0.740     0.947     yes
                 0.000    0.000    ?          0.000    ?          ?        0.740     0.304     no
Weighted Avg.    0.892    0.892    ?          0.892    ?          ?        0.740     0.878

=== Confusion Matrix ===

    a    b   <-- classified as
 1653    0 |    a = yes
  201    0 |    b = no
```

Figure C.4 – Experiment 4: No Regularization (Overfitting)

Parameters: Hidden Layers = a, Learning Rate = 0.3, Momentum = 0.2, Epochs = 1000, Decay = No

Observation: Overfitting detected in common combinations

```
Time taken to build model: 38.15 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1668               89.9676 %
Incorrectly Classified Instances       186               10.0324 %
Kappa statistic                          0.3322
Mean absolute error                      0.1341
Root mean squared error                  0.2801
Relative absolute error                 69.2484 %
Root relative squared error             90.1033 %
Total Number of Instances             1854

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.975    0.716    0.918      0.975    0.945      0.357    0.815     0.964     yes
                 0.284    0.025    0.576      0.284    0.380      0.357    0.815     0.425     no
Weighted Avg.    0.900    0.642    0.881      0.900    0.884      0.357    0.815     0.905

=== Confusion Matrix ===

    a    b   <-- classified as
 1611   42 |    a = yes
  144   57 |    b = no
```

Figure C.5 – Experiment 5: Low Learning Rate (Underfitting)

Parameters: Hidden Layers = 3, Learning Rate = 0.1, Momentum = 0.3, Epochs = 500, Decay = Yes

Observation: Training was too slow; model failed to converge fully

```
Time taken to build model: 1.59 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1664               89.7519 %
Incorrectly Classified Instances       190               10.2481 %
Kappa statistic                          0.1262
Mean absolute error                      0.1753
Root mean squared error                  0.282
Relative absolute error                 90.496  %
Root relative squared error             90.7024 %
Total Number of Instances             1854

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.997    0.920    0.899      0.997    0.945      0.225    0.806     0.960     yes
                 0.080    0.003    0.762      0.080    0.144      0.225    0.806     0.401     no
Weighted Avg.    0.898    0.821    0.884      0.898    0.859      0.225    0.806     0.899

=== Confusion Matrix ===

    a    b   <-- classified as
 1648    5 |    a = yes
  185   16 |    b = no
```

Appendix D: Application Screenshots

Figure D.1 – *Initial User Interface*

Screenshot of the main GUI with default genre and author selections before prediction.

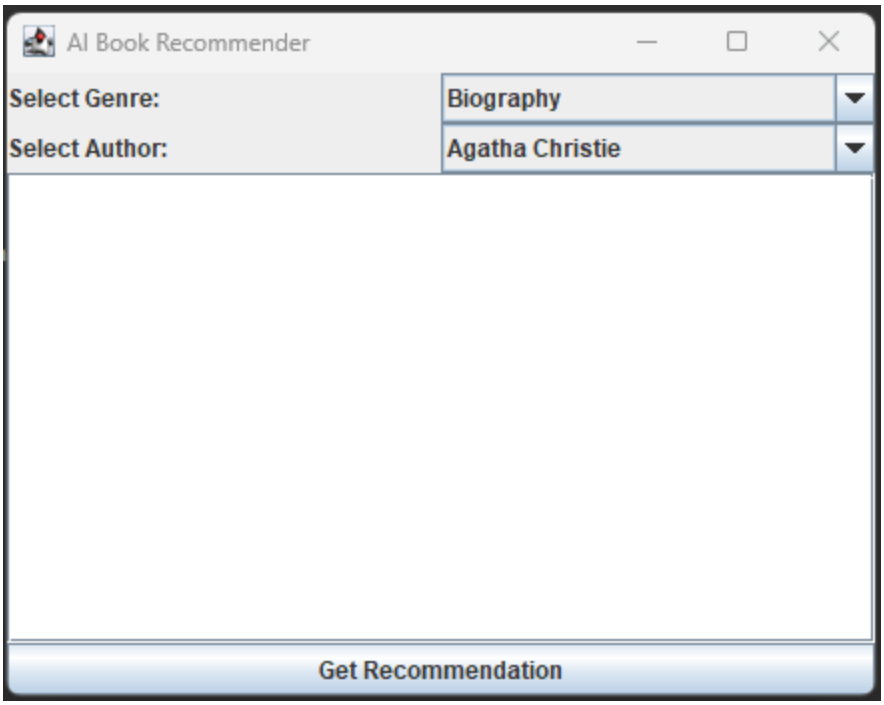


Figure D.2 – *Negative Recommendation Example*

Prediction output when selecting *Fantasy + Meg Cabot (Goodreads Author)* resulting in a “no” recommendation.

Filename: screenshot_negative_recommendation.png

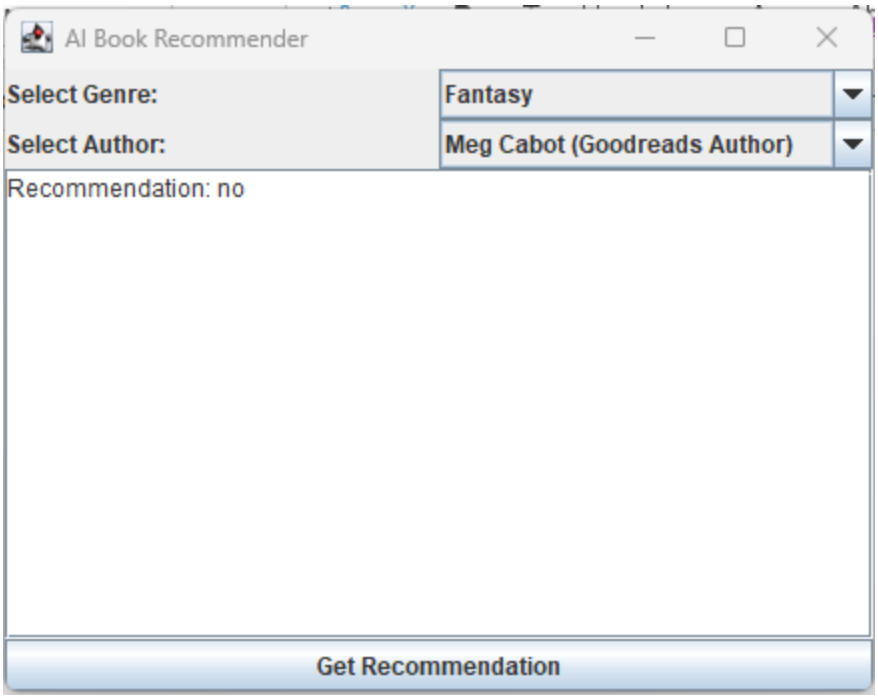
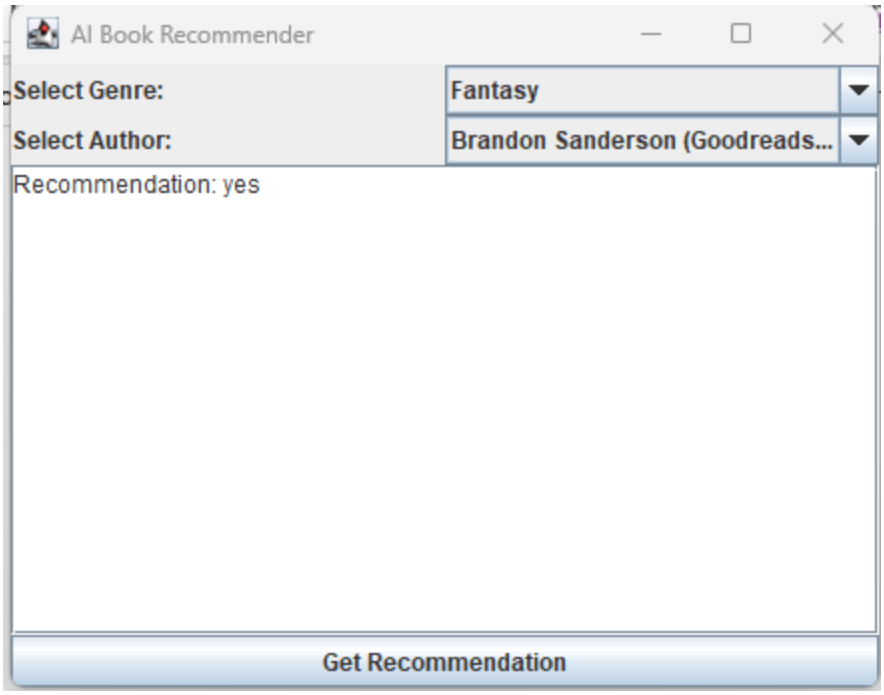


Figure C.3 – *Positive Recommendation Example*

Prediction output when selecting *Fantasy + Brandon Sanderson (Goodreads Author)* resulting in a "yes" recommendation.

Filename: screenshot_positive_recommendation.png



Appendix E: Source Code

Main Class: LibraryAIPrototype.java

Contains the Swing GUI layout, ComboBoxes for input, model and ARFF loading logic, instance creation, and result display.

Required External Files:

- weka.jar (must be added to Eclipse build path)
- BestBookModel.model
- BestBooksRecommendation_SAFE.arff

REFERENCES

Brownlee, J. (2019). *Machine learning algorithms: A review and beginner's guide*. Machine Learning Mastery. https://machinelearningmastery.com/

Kowsari, K., & Brown, D. E. (2020). *A review of artificial intelligence and machine learning algorithms for recommendation systems*. arXiv. https://arxiv.org/abs/2004.13715

Rounak, B. (2020). *Books Recommendation Dataset* [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4265096

University of Waikato. (n.d.). *Weka 3: Data mining software in Java*. https://www.cs.waikato.ac.nz/ml/weka/

Oracle. (n.d.). *Java SE documentation*. https://docs.oracle.com/en/java/javase/