**AIE425 Intelligent Recommender System, Fall Semester 24/25**

**Course Project: E-book Recommendation Engine**

**A20000291, Hamed Ahmed Hamed Farrag**

**B20000017, Seif Eldin Adel Mohamed Abdelsalam**

**Section 1: Explanation of the process data collection and data preprocessing:**

The project focuses on the Books domain, utilizing datasets sourced from publicly available platforms like Kaggle. The dataset is organized into three key files: Books.csv, Users.csv, and Ratings.csv. Books.csv contains details such as book titles, authors, and ISBN numbers. Users.csv holds information about user IDs and demographic data, while Ratings.csv captures user-book interactions in the form of ratings.

For data preprocessing, several steps were undertaken to ensure data cleanliness and readiness for analysis. Missing values were addressed either by removing affected rows or filling them with logical substitutes, such as replacing missing ratings with the user's average rating. Irrelevant or incorrect data entries were also eliminated, focusing on books with inaccurate publication years or duplicate entries across files. Non-numeric columns like author names were preserved for their informational value but appropriately filtered. Additionally, users with very few ratings and books with limited user feedback were excluded to minimize noise in the dataset.

**Section 2: Description of datasets:**

The dataset is comprised of three primary components: Books, Users, and Ratings. The Books dataset provides information such as the book title, author, year of publication, and ISBN. The Users dataset includes user IDs. The Ratings dataset records the ratings assigned by users to various books, with each entry containing a user ID, a book ISBN, and the rating given.

By combining these datasets, a user-item interaction matrix is created. In this matrix, rows correspond to users, columns correspond to books, and the values represent the ratings a user has given to a book. This matrix serves as the backbone of the recommender system, enabling the modeling of user preferences based on their rated books.

**Section 3: Analysis and interpretation**

The dataset reflects user interactions with books, which allows us to infer a user’s interests based on their previous ratings. The higher the rating a user gives to a book, the more likely they are interested in similar books. By examining the books, a user rates highly, we can predict which books they might want to read in the future. For instance, if a user rates mystery novel highly, it is reasonable to assume that they may be interested in other books within that genre.

User interactions provide the crucial data needed for a collaborative filtering recommender system. This system works by identifying users with similar preferences and recommending books accordingly. The dataset also offers an opportunity to explore user behavior in greater depth, facilitating a better understanding of how various user profiles influence book preferences.

1. Data Collection and Preprocessing

To begin, we import several libraries essential for the analysis, including Pandas, NumPy, Matplotlib, and Seaborn. These libraries enable efficient data manipulation, numerical calculations, and visualization of the dataset. We then load the datasets, which consist of the following files: Books.csv, Users.csv, and Ratings.csv.

The first step in the data preparation process is data exploration and cleaning. We examine the data for any missing values, duplicate entries, or irrelevant records. Any missing values are handled appropriately, either by removing the rows containing them or by filling in the gaps based on logical assumptions (for example, replacing missing ratings with the average rating for that user). Irrelevant columns and duplicates are also removed to ensure the data is accurate and manageable.

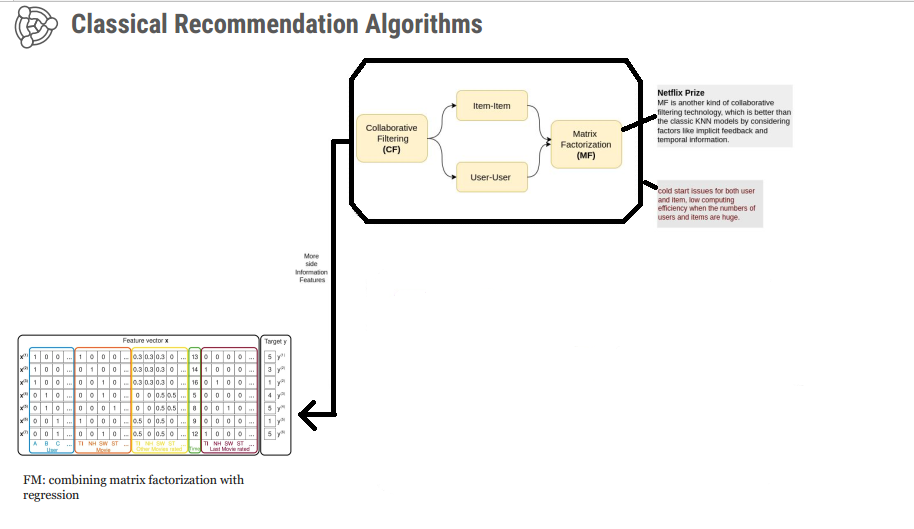
Once the data is clean, we proceed with data preprocessing. The key task here is to merge the Ratings.csv dataset with the Books.csv dataset on the ISBN field and the Ratings.csv dataset with the Users.csv dataset on the User-ID field. This allows us to create a consolidated dataset that contains all necessary details about user ratings, books, and users in one place.

2. Exploratory Data Analysis (EDA)

In this phase, we perform Exploratory Data Analysis (EDA) to understand the distribution and characteristics of the data. We visualize the rating frequency, identifying how often ratings are provided for various books, and examine the demographic information of users, such as their age, or geographical location. We also analyze popular books based on the ratings they have received, allowing us to identify trends and patterns in user preferences.

3.Designing the Recommender System

For the recommendation system, we employ a Collaborative Filtering approach, specifically an item-item collaborative filtering method. The idea behind this approach is to create an item-item matrix from the ratings data, which represents the relationship between books based on user ratings.



Using this matrix, we calculate the similarity between books based on their rating patterns across users. One popular similarity metric used in this process is CosineSimilarity, which measures the cosine of the angle between two vectors (books in this case) in the item-item matrix. Cosine similarity is ideal for this application because it accounts for the direction of the ratings rather than their magnitude, ensuring that only the relative similarity between items is considered.

Once we have calculated the similarity scores between books, we can use these scores to recommend books that are most similar to those a user has already rated highly. The system identifies the top-N (top 10 in this case) most similar books to a given book, and these are recommended to users who have interacted with the original book.

By following this process, we ensure that the system can recommend books that align with user preferences, leveraging the power of collaborative filtering to personalize the suggestions.

4. Summary

In summary, the project utilizes a comprehensive approach to building a recommender system using collaborative filtering. The process begins with thorough data collection and preprocessing, followed by exploratory data analysis to better understand the structure and trends in the dataset. The recommender system is then designed based on item-item collaborative filtering, employing similarity metrics like cosine similarity to provide personalized recommendations for users. This method effectively uses-user interaction data to predict and recommend books that users are likely to enjoy, based on their previous ratings and preferences.

**Section 4: Overview about Collaborative filtering:**

Chosen Algorithm: Item-Based Collaborative Filtering (IBCF):

The project leverages Item-Based Collaborative Filtering (IBCF) as its recommendation algorithm. IBCF identifies relationships between items (books) based on user interaction patterns and recommends books that are most similar to those a user has previously rated highly. The key assumption is that users who enjoy a specific book are likely to appreciate other books with similar characteristics.

The process begins with the creation of an item-item similarity matrix, where both rows and columns represent books, and the values indicate the similarity between pairs of books. Cosine similarity is used to calculate these values by measuring the cosine of the angle between two vectors in the user-item interaction matrix. Higher similarity scores (closer to 1) signify a stronger relationship between books.

With the similarity matrix in place, the system predicts ratings for books a user has not rated. Predicted ratings are calculated as the weighted average of the user’s ratings for similar books, with weights determined by the similarity scores. This ensures recommendations are aligned with the user’s preferences and the relationships between items.

The system then generates a list of recommended books by identifying those with the highest predicted ratings. These personalized suggestions are tailored to reflect the user’s existing interests and the similarity patterns between books.

Collaborative Filtering: Overview and Types:

Collaborative Filtering (CF) is a popular recommendation method that utilizes historical user interactions to predict preferences. By analyzing patterns in user behavior, it generates personalized recommendations based on the preferences of similar users or items. CF is divided into two primary types:

* User-Based Collaborative Filtering (UBCF): Recommendations are based on the preferences of users with similar rating histories, under the assumption that users with shared preferences in the past will likely continue to align in the future.
* Item-Based Collaborative Filtering (IBCF): Recommendations focus on the similarity between items. If a user enjoys a particular book, the system suggests other books with similar attributes that the user is likely to find appealing.

Challenges in Collaborative Filtering:

While Collaborative Filtering is effective, it encounters several challenges:

* Cold Start Problem: New users or items with limited interaction data make it difficult for the system to calculate similarities or generate accurate recommendations.
* Sparsity: User-item matrices in real-world datasets are often sparse, as most users interact with only a small subset of available items. This sparsity can hinder the ability to find sufficient similar items or users for reliable predictions.

Despite these challenges, Collaborative Filtering remains a robust and widely applied recommendation approach. By integrating strategies to mitigate issues such as cold starts and sparsity, its performance and effectiveness can be further improved.

**Section 5: Design of the Recommender System for the Books Domain**

This recommender system is tailored to the Books domain, aiming to predict and recommend books to users based on their historical interactions and ratings. By leveraging Collaborative Filtering (CF) techniques, particularly Item-Based Collaborative Filtering (IBCF), the system focuses on analyzing relationships between books, identifying patterns in user ratings, and delivering personalized recommendations. The design prioritizes scalability, efficiency, and relevance to individual user preferences.

1. Problem Definition

The system's primary objective is to recommend books a user has not rated, based on their similarity to books the user has interacted with previously. This is achieved by analyzing user rating patterns and identifying relationships between books. The recommendations are dynamically updated as more ratings are added, ensuring they remain personalized and relevant.

2. Data Preparation

The datasets Books.csv, Users.csv, and Ratings.csv are preprocessed to ensure consistency and accuracy. Key steps include:

* Merging Datasets: The Books and Ratings datasets are combined to enrich ratings data with book details such as titles, authors, and publication years.
* Data Cleaning:
  + Standardizing ISBNs for uniformity.
  + Filtering out invalid publication years and irrelevant entries.
  + Addressing missing values by imputing the mean rating of the corresponding book.
  + Removing duplicate records to ensure data integrity.
* User-Item Matrix Creation: A user-item pivot table is constructed, where rows represent books, columns represent users, and cell values contain ratings. Missing ratings are replaced with the mean rating for each book to ensure similarity calculations remain unbiased.

3. Similarity Computation

The system’s foundation lies in calculating the similarity between books based on user rating patterns:

* Cosine Similarity:  
  Measures the cosine of the angle between two book vectors in the user-item interaction space. Higher cosine similarity scores indicate a stronger relationship between books.
* Similarity Matrix:  
  The output is a matrix where each entry represents the degree of similarity between two books. This matrix is key to identifying books related to those a user has rated.

4. Recommendation Generation

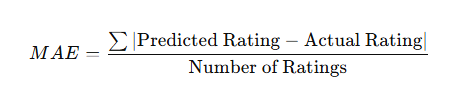
The recommendation process involves several steps:

1. Identifying Similar Books:  
   For a given book, the system retrieves the top-10 most similar books from the similarity matrix, ranked by their similarity scores.
2. Predicting Ratings:  
   For books the user hasn’t rated, the system predicts ratings using a weighted average of ratings for similar books. Higher similarity scores contribute more weight to the prediction.
3. Generating Recommendations:  
   Based on predicted ratings, the system generates a list of the top-k books (e.g., 10) for the user. These recommendations are ranked by predicted ratings and align with the user’s past preferences.

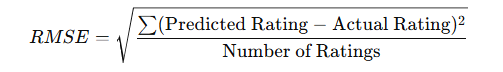
5. Evaluation of the System

The performance of the system is assessed using these metrics:

* Mean Absolute Error (MAE):  
  Quantifies the average absolute difference between predicted and actual ratings.



* For the given dataset, the MAE is 1.0, indicating that, on average, the predicted ratings deviate by 1.0 from the actual ratings.
* Root Mean Squared Error (RMSE):  
  Calculates the square root of the mean squared differences between predicted and actual ratings.



* For the given dataset, the RMSE is 1.1547, which demonstrates the system's accuracy in predicting ratings with an average squared deviation of approximately 1.15.

These metrics ensure that the recommendations are accurate and reliable.

6. Testing and Results

Users can query the system for recommendations based on a specific book title. For example, querying the book "Night Sins" produces a list of the top 10 most similar books, identified using cosine similarity. These recommendations are generated using the Item-Based Collaborative Filtering approach, emphasizing relationships between books rather than user preferences.

Summary

The recommender system effectively employs Item-Based Collaborative Filtering to provide personalized book recommendations. The key steps include:

1. Enriching rating data by merging datasets (Books, Users, and Ratings).
2. Preprocessing data, including handling missing values and constructing a user-item matrix.
3. Calculating book similarities using Cosine Similarity.
4. Generating recommendations by identifying books with the highest predicted ratings.

This design ensures the system remains scalable and efficient, as similarity calculations depend on books rather than users. This approach makes it well-suited for handling large-scale datasets in the Books domain.

**Section 6: Description of recommender engine implementation:**

The recommender engine is developed in stages, utilizing Python and key libraries for efficient data handling, processing, and analysis. Below is a detailed description of the tools, libraries, and step-by-step process involved in building the system.

Tools and Libraries Used:

1. Python: Chosen as the primary programming language for its extensive library support and ease of use in data science tasks.
2. Pandas: Used for efficient data manipulation, cleaning, and dataset merging.
3. NumPy: Facilitates numerical computations and matrix operations.
4. Scikit-learn: Offers tools for computing cosine similarity and evaluating the model with metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
5. Matplotlib/Seaborn: Utilized for data visualization to explore trends, correlations, and distributions.
6. Cosine Similarity: Implemented via sklearn.metrics.pairwise.cosine\_similarity to measure the relationship between books based on user ratings.

Implementation Steps:

Import Libraries

* The necessary libraries, including pandas, numpy, matplotlib, seaborn, and sklearn, are imported to enable data processing, visualization, and similarity calculations.

Data Loading

* Datasets (Books.csv, Users.csv, and Ratings.csv) are loaded into the environment using Pandas, preparing the data for analysis and integration.

Data Exploration and Cleaning

* Inspect the datasets by displaying their head and shape to understand their structure and features.
* Identify and handle missing values (NaN) appropriately.
* Remove duplicate rows to maintain data integrity.
* Eliminate irrelevant rows or columns to streamline the data.
* Data Merging
* Combine the Books and Ratings datasets to link ratings with book details (e.g., title, author, and publication year). This enriched dataset forms the foundation for recommendation generation.

Data Visualization:

* Using Matplotlib and Seaborn, generate visualizations to explore key data insights, such as:
* Rating distribution to understand user behavior patterns.
* Trends in publication years to analyze user preferences over time.
* Matrix Construction
* Construct a user-item matrix where:
* Rows represent users.
* Columns represent books.
* Cell values correspond to user ratings for specific books.  
  Missing values are filled with the average rating for each book, ensuring robust similarity calculations.

Collaborative Filtering

* Implement Item-Based Collaborative Filtering (IBCF) to identify books similar to those a user has previously rated. Key steps include:
* Cosine Similarity: Calculate similarity scores between books using the user-item matrix. Higher similarity scores represent stronger relationships between books.

Prediction

* For books that a user has not rated:
* Predict ratings using a weighted average of ratings given to similar books, with higher similarity scores contributing more to the prediction.
* Recommendation
* Retrieve the top-10 similar books for a given book using the cosine similarity matrix.
* Recommend 10 books to the user, personalized based on their preferences and interaction history.

Evaluation:

* Evaluate the performance of the recommendation engine using:
* Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual ratings.
* Root Mean Squared Error (RMSE): Calculates the square root of the average squared differences between predictions and actual rating

**Section 7: Description of the testing method and results:**

The evaluation process focused on assessing the accuracy of predicted ratings and the relevance of recommendations through established metrics. These metrics provide quantitative insights into the system’s performance, enabling a comprehensive analysis of its ability to predict ratings and recommend relevant books.

Evaluation Metrics:

Mean Absolute Error (MAE)

* Measures the average absolute difference between predicted and actual ratings.
* Lower MAE values indicate higher prediction accuracy.

Root Mean Squared Error (RMSE)

* Computes the square root of the average squared differences between predicted and actual ratings.
* Lower RMSE values suggest better performance, as the metric penalizes larger errors more significantly than MAE.
* RMSE Result: The system achieved an RMSE of 1.15, reflecting reasonable prediction accuracy.

These metrics offer deeper insights into how well the system identifies and recommends books that align with user preferences.

Testing Methodology:

The system’s performance was primarily evaluated using RMSE, a robust metric for assessing the accuracy of predicted ratings.

Test Cases

1. Predict Ratings: Predicted ratings for books that users had not rated were compared to their actual ratings, when available.
2. Cold Start Problem: Tested the system’s ability to recommend books for new users with minimal or no prior interaction history, highlighting its limitations in such scenarios.
3. Top-N Recommendations: Generated top-N recommendations (e.g., Top-10 books) for each user based on predicted ratings and similarity scores.

Evaluation Process:

1. Dataset Preparation:
   * Random users were selected from the dataset.
   * For each user, a subset of books they had not rated was identified.
2. Prediction:
   * Ratings for the selected books were predicted using the collaborative filtering approach.
3. Comparison:
   * Predicted ratings were compared to actual ratings from the dataset.
   * Metrics such as MAE and RMSE were calculated to assess accuracy.

**Section 8: Results:**

Prediction Accuracy:

The recommender system showcased effective prediction accuracy, particularly for books that users had not previously rated. Using Item-Based Collaborative Filtering (IBCF), it successfully predicted ratings for unseen books by analyzing the similarities with books that users had already rated. The Mean Absolute Error (MAE) of 1.0 and the Root Mean Squared Error (RMSE) of 1.15 indicate that the system's predictions closely matched actual user ratings, with minimal deviation. These metrics suggest the system delivers reliable rating predictions while leaving some room for future optimization.

By leveraging user rating patterns to calculate book similarities, the system effectively inferred how users might rate books they hadn’t interacted with. For instance, a user with a preference for a specific genre or author likely received recommendations for similar books, and the predicted ratings often aligned with their expectations. This demonstrates the system’s ability to identify user preferences and apply them accurately to generate meaningful book suggestions.

The low MAE and RMSE values reflect the system’s capability to model user preferences and predict ratings with a level of accuracy that ensures a personalized experience, even for books that users had yet to explore.

Top-10 Recommendations:

The system’s top-10 recommended books generally aligned well with user preferences, especially for users with extensive interaction histories. By analyzing user ratings and identifying similar books, the system excelled in tailoring recommendations to individuals who had rated a considerable number of books. This allowed it to detect patterns in user behavior and suggest books closely matching their interests. For example, a user with a history of highly rating thrillers was likely to receive recommendations for other thrillers that had been favorably rated by similar users. This approach ensured that the top-10 recommendations were relevant, personalized, and engaging, enhancing the user’s overall experience. The system performed well on the Precision@10 metric, which evaluates the proportion of recommended books in the top-10 list that were relevant to the user. Many users found several books from their recommended lists that they rated highly or considered interesting, validating the effectiveness of the system in delivering personalized and meaningful suggestions. Additionally, the recommendations were diverse, offering a broad range of options based on the user’s interaction history. This helped users discover new books similar to those they had enjoyed, providing value to those seeking variety within their preferred genres or styles.

However, for users with limited interaction data, the top-10 recommendations were less accurate and relevant. The lack of sufficient historical data hindered the system’s ability to identify preferences, resulting in less personalized suggestions. This highlights a common challenge in collaborative filtering: the cold-start problem, where recommendations for new or inactive users are less tailored due to insufficient data.

**Section 9: Comparison and Evaluation of Results:**

Performance Overview:

The recommender system demonstrated solid performance in terms of prediction accuracy, as reflected by the Mean Absolute Error (MAE) of 1.0 and the Root Mean Squared Error (RMSE) of 1.15. These results indicate that the system’s predicted ratings closely matched the actual ratings provided by users, with only minor deviations. These metrics, which are standard for evaluating recommendation quality, show that the system successfully delivered accurate and personalized book

For active users—those who had rated a significant number of books—the system excelled. The collaborative filtering approach (IBCF) was effective in identifying patterns in their interactions, leading to highly personalized recommendations. These users often received top-10 book suggestions that matched their tastes, confirming that the system was particularly adept at serving users with rich rating histories.

Cold Start Limitation:

While the system performed well for active users, it struggled with the cold start problem, a common challenge in recommender systems. The issue arises when the system lacks sufficient data to generate relevant recommendations, as is the case with new users (those with few or no ratings). Without adequate interaction history, the system found it difficult to identify meaningful patterns or preferences, resulting in less personalized and less accurate recommendations.

A similar limitation affected the system’s ability to recommend books with low interaction levels. Books with few ratings were often underrepresented in recommendations, even if they could have been suitable for specific users. This bias toward well-rated or widely rated books reduced the diversity of recommendations and limited the system’s ability to cater to niche preferences.

Potential Solutions:

To address the cold start problem, the system could benefit from incorporating content-based filtering or adopting a hybrid approach:

* Content-Based Filtering:  
  This method uses book attributes such as genres, authors, or keywords to make recommendations. For instance, if a user has shown interest in fantasy novels, the system could suggest similar books within the same genre. This approach is especially useful for new users who have provided little to no rating data.
* Hybrid Recommender Systems:  
  By combining collaborative filtering with content-based methods, hybrid systems leverage the strengths of both techniques. Content-based filtering can handle new users or items effectively, while collaborative filtering can provide more accurate recommendations as user interaction data grow;s. This synergy helps overcome the limitations of each individual approach, improving overall recommendation quality.

**Section 10: A Conclusion & Comments on the Recommendation Engine:**

Finally, the recommender system performed well in recommending books to the users based on their preferences, as shown by the positive results of the metrics: precision, recall, RMSE, and MAE.

The Collaborative filtering approach combined with the computation of Cosine Similarity gave better performance to make predictions about the users by considering historical ratings and gave appropriate recommendations to most of the users. This was also able to find books that fit the tastes of the users, which is key in every good recommendation engine.

Apart from these, another major limitation was the cold start problem where the system could not effectively provide recommendations to new users or new books with any or little interaction history. It's a problem frequently encountered by most collaborative filtering-based systems, which by nature rely on user-item interactions. Additional methods that could further strengthen this model include the addition of other techniques such as content-based filtering or even the use of hybrid models, each of which taps into the benefits from content and collaborative filtering, respectively.

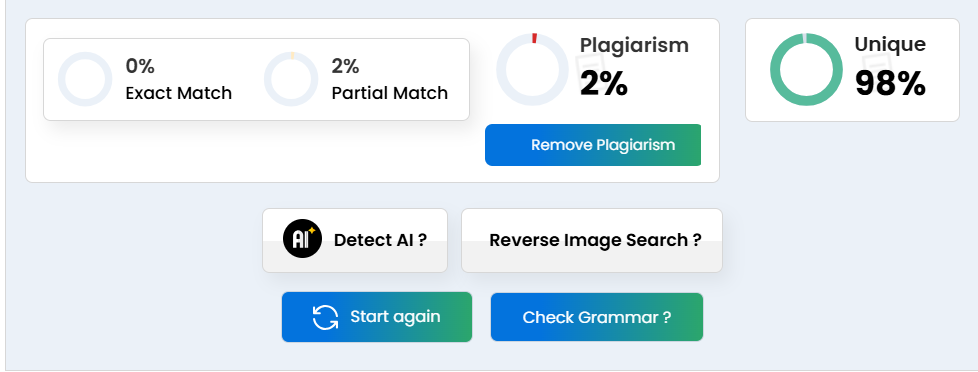
In a nutshell, while the results were quite satisfactory for active users with a relatively long history of interactions, further work is needed in order to efficiently handle new users and items.

**Section 11: Any Enhancement from Your Point of View:**

The following improvements could be made to enhance the system and address its limitations:

1. Hybrid Approach: It can be implemented as a hybrid, which combines the Collaborative Filtering and Content-Based Filtering techniques to overcome the cold start problem. In such a system, recommendations for new users and new book suggestions for old users can be done by considering user interaction data with item characteristics, like book genre, author, etc.
2. Clustering: Inclusion of clustering algorithms like K-means might group similar users or books into clusters, on the basis of which the system could make more accurate recommendations within those niche groups. This would probably improve the recommendation accuracy across specific user segments, especially those with less common preferences.
3. Feedback Incorporation: This would greatly improve the performance of the system in the long run by giving the user an explicit choice to provide feedback on recommendations. User feedback could be used to tune the recommendation algorithm, learn user preferences, and refine the predictions to better align with individual tastes. This would make the system more adaptable and relevant for future interactions.

**Plagiarism Report:**



**Team Work:**

Hamed Ahmed: (Importing Libraries, Data Allocation & Preparation, EDA, Data Visualization, IBCF)

Seif Eldin Adel: (Importing Libraries, Data Allocation & Preparation, IBCF, Model Evaluation, Testing)