Probability Programming Project

Book Recommendation System using Hierarchical Poisson Factorization with MCMC

Dolganiuc Ana

For this project I chose a Book Recommendation dataset and based on the user's ratings, age and location I created a recommendation system. The system is a probabilistic collaborative filtering model that uses latent factorization. It uses MCMC sampling to estimate the parameters of the model, which captures the latent structure of user-item interactions. The project starts with data cleaning and preprocessing, and then continues with the model training using the hierarchical poisson factorization with MCMC sampling. After that we tune the hyperparameters and observe the results of the model using the RMSE, MAE, tables and plots of the posterior distribution.

1) Exploratory Data Analysis

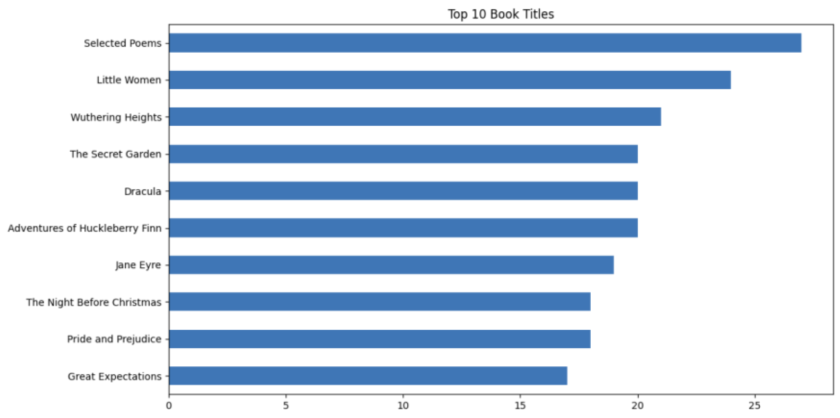
The dataset has :

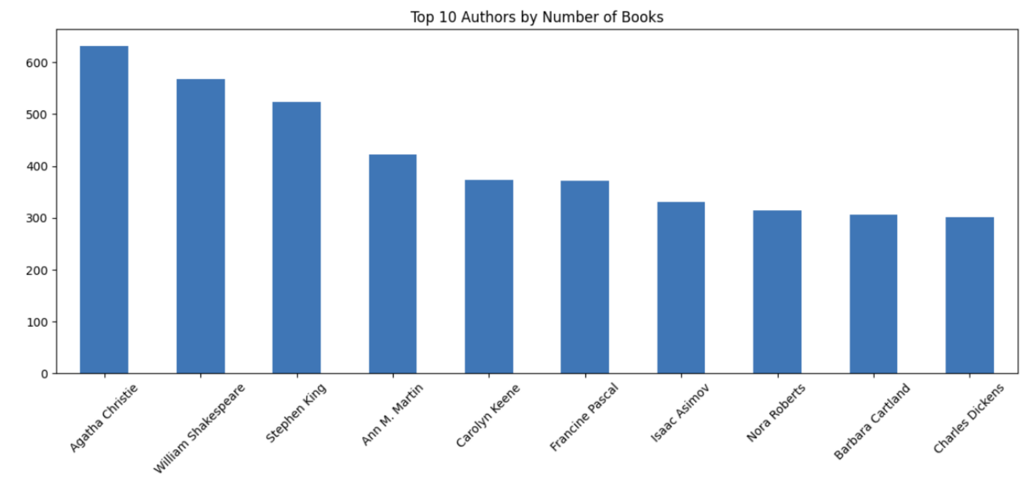
- a number of 105283 Users
- a number of 340556 Books

With the following columns:

	User-ID	ISBN	Book-Rating	Book-Title	Book-Author	Age	Location
2	276727	0446520802	0	The Notebook	Nicholas Sparks	16.0	h, new south wales, australia
3	276729	052165615X	3	Help!: Level 1	Philip Prowse	16.0	rijeka, n/a, croatia
4	276729	0521795028	6	The Amsterdam Connection : Level 4 (Cambridge ...	Sue Leather	16.0	rijeka, n/a, croatia
5	276733	2080674722	0	Les Particules Elementaires	Michel Houellebecq	37.0	paris, n/a, france
16	276747	0060517794	9	Little Altars Everywhere	Rebecca Wells	25.0	iowa city, iowa, usa

Let's look at the 10 most popular book-titles and 10 most popular authors:





- Data pre-processing

The average book rating is 7 and the average age of the user is 45 yo.

In order to make the dataset usable we will delete all the missing values (the majority were located in the 'Age' column) and we'll check for the duplicate rows.

We will remove the noise in the dataset by reducing the books where the rating is not in the 75th percentile.

```
Rating threshold (75th percentile): 3.0
Original dataset size: (753300, 7)
Filtered dataset size: (542460, 7)
```

- Sampling methods:

In order to ensure that the resulting sample maintains the same proportion of certain characteristics as the original dataset we'll use the Stratified Sampling, and to prioritize or data points based on their popularity we'll use Popularity-Based Sampling.

Stratified Sampling:
For this approach we'll select the users based on their Age and their Location. This is because Location and Age can significantly influence reading preferences and behaviors. People from different locations may have varying interests due to cultural, regional, or availability differences. Similarly, age can reflect different reading interests and comprehension levels. Stratifying by these variables helps ensure the sample reflects this diversity. This will help with:

- Personalization: by including diverse age groups and locations, the system can learn patterns that are more specific to certain demographics, enhancing its ability to provide relevant recommendations.
- Relationships Between Selected Users and Items Demographic Patterns: By selecting users and items based on age and location, we might uncover demographic patterns in book preferences.

- Age Appropriateness: The stratification by age ensures that the selected books cover a range of interests suitable for different age groups. This might help in understanding how preferences evolve with age or identifying books that cross generational boundaries.

Popularity-Based Sampling:

In order to do this sampling we'll select the most popular users and books in the data set. This will help us with:

- Bias Reduction: Sampling only the most active users or the most rated books might introduce bias towards certain types of books or user groups.
- Improving Generalization
- Balancing Exploration and Exploitation

Model Training

The implementation strategy for this HPF model was driven by a need to build a robust, scalable, and interpretable model that can handle the typical challenges of recommendation systems, including sparsity, scalability, and the need to infer latent structures that explain the observed data.

These where the steps for the implementation:

1. We use sparse matrices to efficiently manage the inherently sparse nature of user-item interaction data. The sparse matrices store only non-zero entries, therefore significantly reducing memory usage and computational cost.
2. We will define several Gamma-distributed variables to model the activity levels of users, the popularity of items, user preferences, and item attribute. The number of latent factors is a hyperparameter that allows us to for tune the model's complexity and its ability to capture nuanced relationships. We will decompose user-item interactions into latent factors for users (theta) and items (beta). By doing this we can capture underlying patterns in the data, such as user preferences and item characteristics, that are not directly observable.
3. We will then calculate the expected rate of user-item interactions by taking the dot product user preferences and item attributes, essentially modeling the interaction between user preferences and item attributes. By modeling the data with a Poisson distribution, the model can effectively capture the intensity of interaction (e.g., number of times a book is rated) as a proxy for user preference.
4. Finally, the model will be fitted using MCMC sampling, specifically with the No-U-Turn Sampler, targeting an acceptance rate of 0.95 for the sampler. The pm.sample function is used to draw 3000 samples after tuning for 800 iterations. By the model with MCMC sampling, it will provide point estimates for the parameters and quantify uncertainty

around these estimates. This is crucial for us to make informed decisions and understand the reliability of the recommendations.

Hyperparameter tunning and sanity check on 10 users 10 items:

Scenario	Nr samples	Nr iterations	Results:

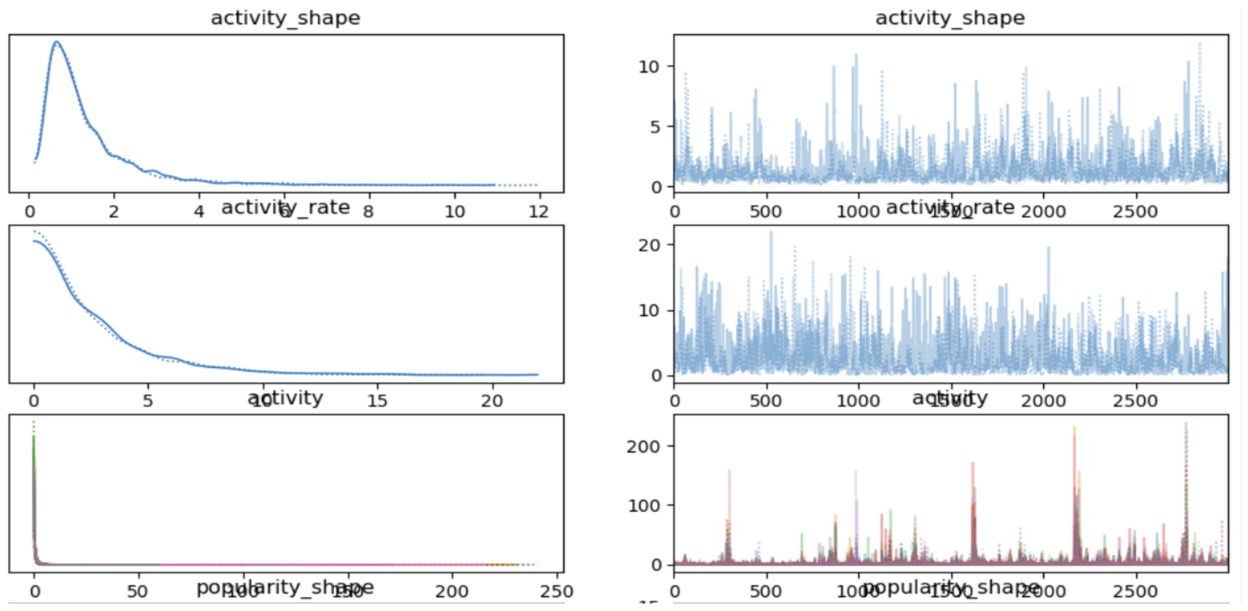
1	1000	500	
			```
			                    ess_tail  r_hat
			activity_shape     647.0   1.02
			activity_rate      777.0   1.01
			activity[0]        176.0   1.04
			activity[1]        581.0   1.02
			activity[2]        400.0   1.02
			...                  ...    ...
			beta[22, 0]        850.0   1.01
			beta[22, 1]       1240.0   1.01
			beta[22, 2]       1128.0   1.01
			beta[22, 3]        860.0   1.01
			beta[22, 4]        908.0   1.01
			```

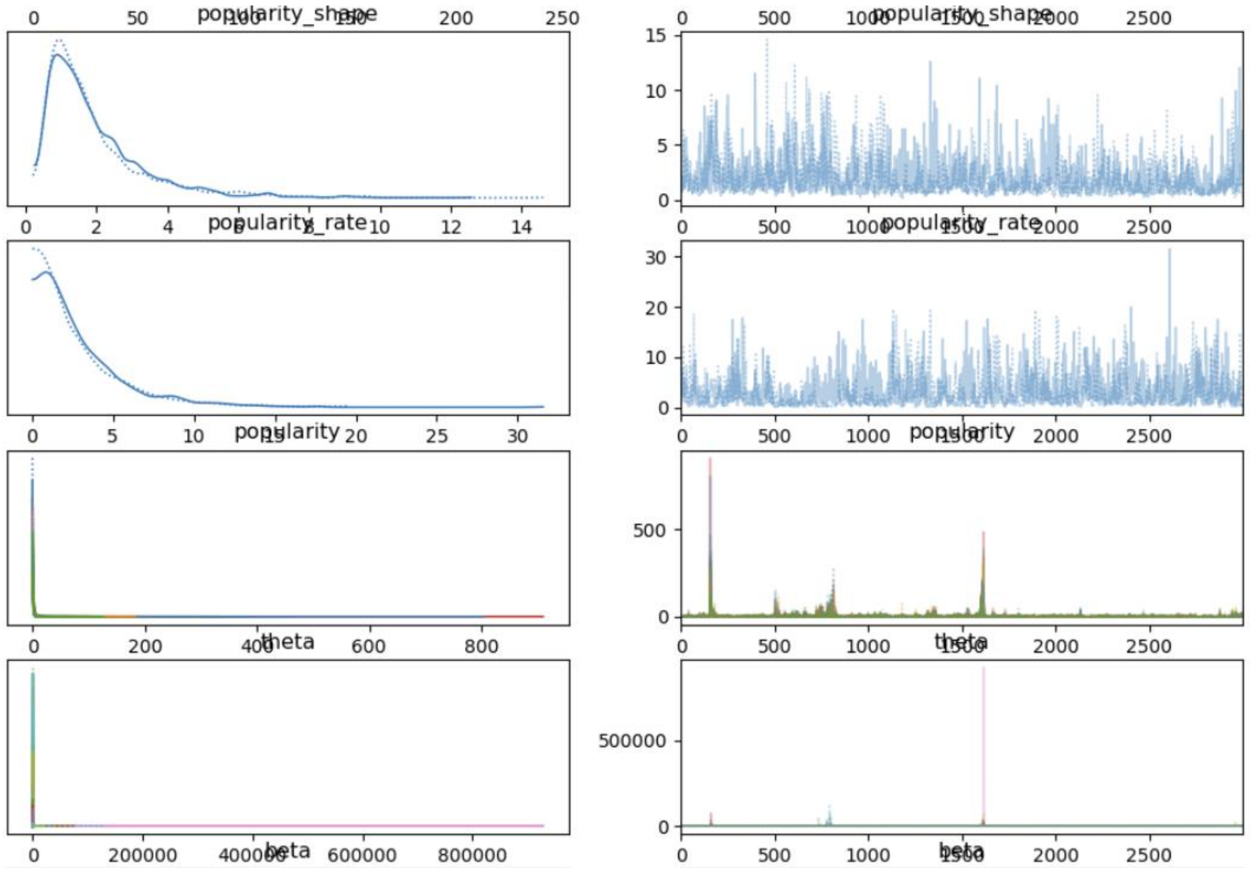
2	3000	800	
			```
			                    ess_tail  r_hat
			activity_shape     668.0    1.02
			activity_rate      835.0    1.01
			activity[0]        253.0    1.03
			activity[1]        833.0    1.01
			activity[2]        610.0    1.01
			...                  ...     ...
			beta[22, 0]        1258.0   1.01
			beta[22, 1]        1149.0   1.01
			beta[22, 2]        1198.0   1.01
			beta[22, 3]        1075.0   1.01
			beta[22, 4]        953.0    1.01
			```

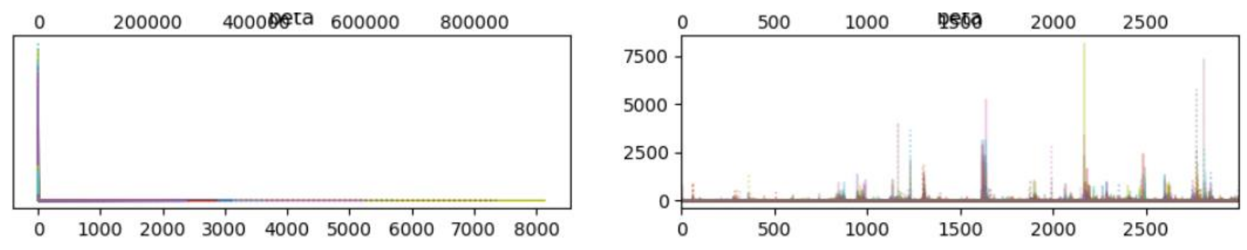
Results:

The r_hat values are close to 1 in both scenarios suggesting that the chains have converged. Meanwhile, for the ESS measure we see that Scenario 2 has higher ESS values across the board, which suggests that the increase in iterations and samples have provided more independent samples that are effectively obtained from the correlated MCMC samples

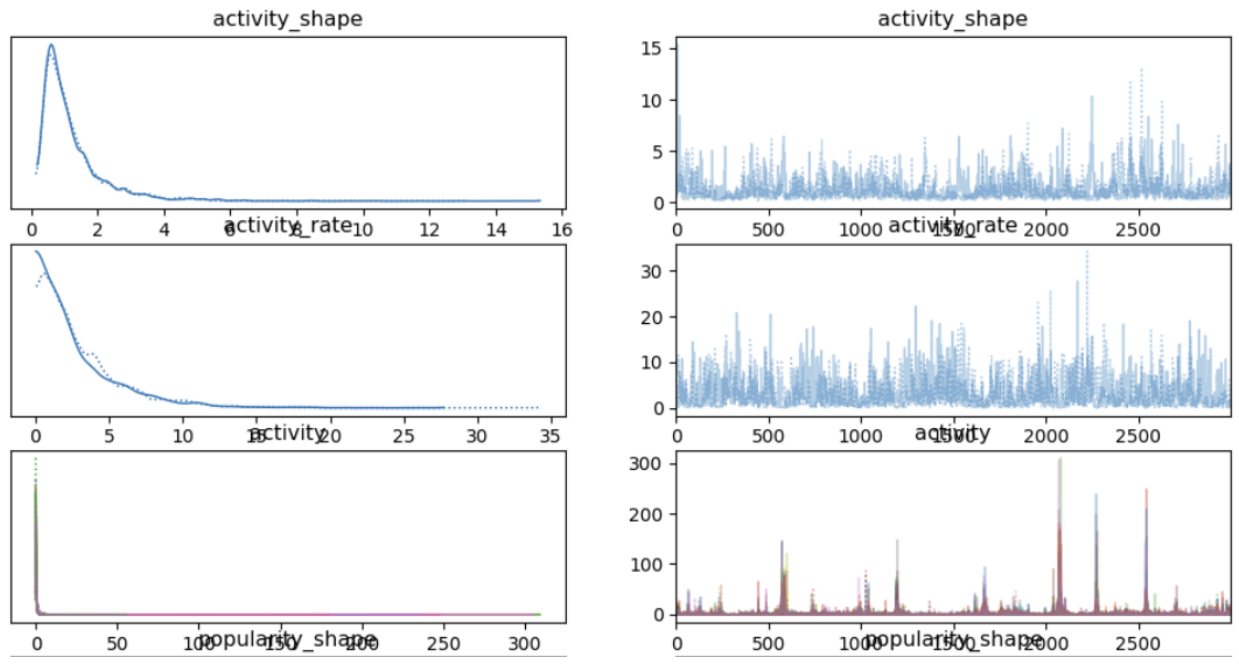
Resulting Plots:
Scenario 1: 1000 Samples, 500 Iterations

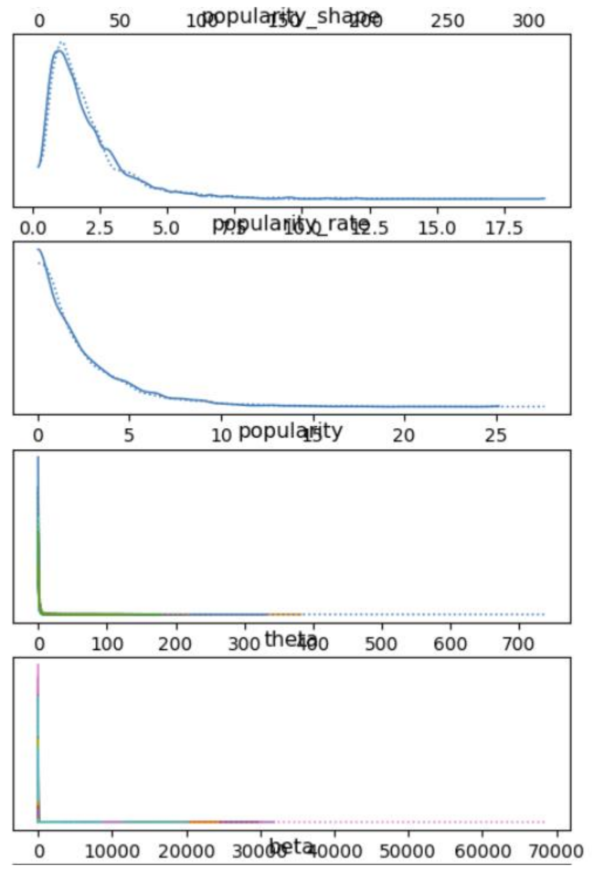

activity_shape
activity_shape
activity_rate
activity_rate
activity
activity
popularity_shape
popularity_shape

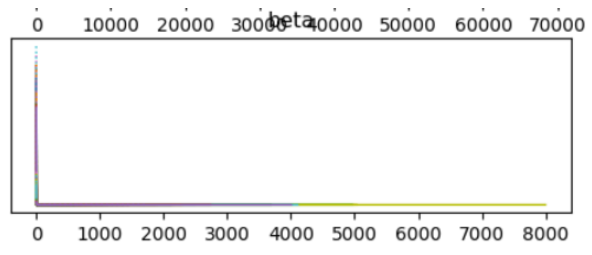

popularity_shape
popularity_shape
popularity_rate
popularity_rate
popularity
popularity
theta
theta
beta
beta



Scenario 2: 3000 Samples, 800 Iterations







Comparison between the 2 scenarios:

- Trace plots:
  In the second scenario the traces are denser than in the first one, which indicates a better mixing of the parameter space.
- Autocorrelation plots:
  In the first scenario the plots show that the autocorrelation didn't drop quickly, meanwhile in the second scenario there is a quick drop to lower values, which is a sign of less autocorrelation and better mixing

Results after training the model on 200 users and 200 items

	Training set	Test set

RMSE	0.619524384503	0.655327507267032
MAE	0.3926839405734	0.445086757990867

As we can see, both RMSE and MAE values are less than 1 on both the training and test sets. This indicates that on average, the model's predictions are within one point of the actual ratings, which can be considered a good level of accuracy in many contexts. The difference between the training and test RMSE and MAE values is not very large, which suggests that the model has not overfit the training data much. This is a good sign of the model's ability to generalize to unseen data.

Conclusions

In this project, we developed a Book Recommendation system using a probabilistic collaborative filtering approach, utilizing latent factorization to predict user ratings based on user demographics and historical interactions with books. The project involved several key steps, including data pre-processing, model training, hyperparameter tuning, and evaluation. In conclusion, the model successfully leverages user ratings, age, and location to provide personalized book recommendations. Through careful data preprocessing, sampling, and probabilistic modeling with MCMC, we built a scalable and interpretable system. The final results, including convergence diagnostics and predictive accuracy, suggest that the model can effectively capture the latent structure of user-item interactions, providing accurate and reliable recommendations.