**1. Introduction**

In the realm of recommendation systems, collaborative filtering stands out as a foundational technique, facilitating personalised suggestions by analysing user-item interactions and harnessing collective preferences. Its broad application across various domains, from e-commerce platforms like Amazon to entertainment giants like Netflix and YouTube, highlights its significance in enhancing user experience and driving engagement. By predicting user interests based on similarities with other users, collaborative filtering empowers platforms to anticipate customer preferences before they are even aware of them, thus delivering tailored content or products.

In today's information-driven era, it's crucial to grasp the principles of collaborative filtering from scratch. While recommendation systems are everywhere, understanding the basics gives us deeper insights into how they work and helps develop stronger algorithms. By understanding collaborative filtering's underlying mechanisms, we can untangle the complexities of recommendation systems and fully understand their potential to provide personalised experiences to users.

Our main goals here are to implement collaborative filtering algorithms from scratch and compare their performance with other variations. We'll focus on different methods to implement collaborative filtering using matrix factorization, to see what they're good at and where they fall short. In this project we will focus on two model-fitting methods, Alternating Least Squares (ALS) and Gradient Descent (GD). Through careful testing and analysis, we aim to understand how well these algorithms predict user preferences and recommend items, especially using a song dataset.

**2. Background**

The development of collaborative filtering technologies has kept pace with the rapid growth of the internet and the surge of online data. Originally designed to help manage the vast amount of content on the web, these technologies have advanced from simple heuristic methods to sophisticated algorithms that can detect complex patterns in user behaviour.

One key advancement in this field is matrix factorization, which includes techniques like Singular Value Decomposition (SVD). These methods simplify large user-item interaction matrices into smaller ones, uncovering hidden factors that influence user preferences and item characteristics. As data volumes have grown, the need for algorithms that can efficiently handle large datasets and provide real-time personalised recommendations has become crucial. This background sets the stage for our project, where we explore building these techniques from scratch to better understand their functionality, effectiveness on a song dataset, and their broader impact on machine learning.

**3. Methodology**

**3.1 Dataset**

In this project, we utilised a song dataset sourced from Kaggle, specifically tailored for collaborative filtering tasks. This dataset contains records of past interactions between users and music, including their ratings over a certain period. With 2,000,000 observations from 200,000 users. The dataset contains 3 attributes, which are user ID, song ID and rating, which represents the user, the song, and the rating given for the song respectively.

The dataset comprises numerical data, which are both well-structured and free of missing values, reducing the need for further preprocessing. The main focus of our preprocessing efforts was on remaking the song ID to make it more compatible for the matrix factorization algorithms used in collaborative filtering.

**3.2 Model Building**

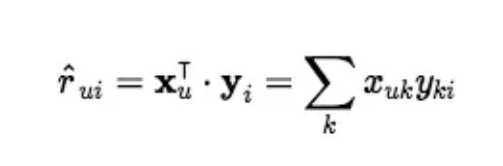
**3.2.1 User-item Matrix**

The user-item matrix is a fundamental representation in collaborative filtering, where rows correspond to users, columns correspond to items, and the entries indicate the interactions or ratings of users for items. In the context of the song dataset used in this project, the user-item matrix encapsulates the interactions between users and songs, with each cell representing a user's rating or interaction with a particular song.

**3.2.2 Matrix Factorization**

In this project, we utilised the matrix factorization technique to decompose a user-item interaction matrix into lower-dimensional matrices, effectively capturing the latent features of both users and items. We specifically chose this approach for its effectiveness in handling sparse matrices within our recommendation systems.

Matrix factorization involves factorising the user-item matrix into two separate matrices: one representing user latent vectors and the other representing item latent vectors. These matrices were initialised randomly with dimensions corresponding to the number of users and items, respectively, and the desired number of latent factors.



**Image source:** Zhou, E. (2018). Matrix Factorization

By decomposing the matrix into user and item latent matrices, our objective is to capture underlying patterns or preferences in the data. These latent vectors serve as feature representations of users and items. For instance, the user latent matrix could encapsulate demographic information, personal interests, and behavioural patterns, while the item latent matrix might represent genres, content, tone, or other relevant characteristics.

**3.2.3 Training Sampling**

In our collaborative model, we explore two variations for training. The first approach involves updating the entire dataset in each iteration, akin to the base model, while the second involves using only a random sample. This is to mitigate the computational burden associated with updating all user-song pairs in each iteration, potentially reaching 2 million entries. Therefore, we opted to only randomly select 100,000 user-song pairs for updating. This strategy not only accelerates our training speed but also facilitates early stopping to prevent overfitting. By focusing on a subset of the data, we efficiently train the model while still capturing crucial patterns and trends in user-song interactions. This approach optimises computational resources, especially when dealing with large datasets, ensuring effective model training and performance.

**3.3. Model Optimization Techniques**

In this project, we employed two model-fitting methods in collaborative filtering for recommendation systems: Gradient Descent and Alternating Least Squares (ALS).

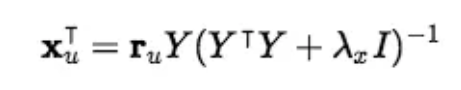
**3.3.1 Gradient Descent**

In employing Gradient Descent (GD), a widely-utilised optimization algorithm, we iteratively update the weights of the latent matrices in the collaborative filtering model. These matrices represent the underlying patterns or features of users and items in the recommendation system. At each iteration, a random entry from the user-item interaction matrix is selected, and the gradient of the loss function with respect to the weights of the user and item latent matrices is computed. This gradient guides the adjustment of the weights to minimise the loss, with updates scaled by a predefined learning rate. Through this iterative process, the latent representations of users and items are refined, ultimately improving the accuracy of recommendations.

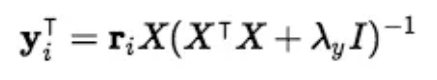
**3.3.2 Alternating Least Squares (ALS)**

To implement Alternating Least Squares (ALS) in collaborative filtering, we initialise the user and item latent matrices randomly. Then, we iteratively optimise these matrices by alternately updating the user matrix while keeping the item matrix fixed, and vice versa. This iterative process continues until convergence within a predefined maximum number of iterations. At each iteration, we solve a least squares optimization problem to update the latent vectors for users and items

Users latent vector:

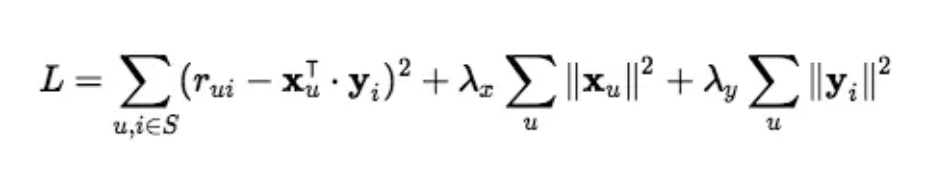


Songs latent vector :



Once convergence is achieved, we utilise the updated latent matrices to predict missing values in the user-item interaction matrix.

**3.3.3 Loss Function**



**Image source:** Zhou, E. (2018). MSE Loss with Regularization

In training our collaborative filtering model, we employed Mean Squared Error (MSE) as our loss function. MSE serves as a measure of the average squared differences between the predicted ratings and the actual ratings in the user-item interaction matrix. It quantifies the overall magnitude of the errors, providing insight into the model's performance in approximating user preferences. MSE is computed by averaging the squared differences between predicted and actual ratings.

**3.3.4 Regularisation**

In addition to optimising our collaborative filtering models using Gradient Descent and Alternating Least Squares, we incorporated regularisation techniques to prevent overfitting and improve generalisation performance. Regularisation is a crucial component in model training, especially when dealing with high-dimensional parameter spaces and sparse data.

We applied L2 regularisation, also known as ridge regression, to our collaborative filtering models. L2 regularisation adds a penalty term to the loss function, which penalises large weights in the model, effectively shrinking them towards zero. This regularisation term is used to prevent our model from becoming overly complex and overly reliant on specific features or interactions present in the training data, thereby improving its ability to generalise to unseen data.

Mathematically, the L2 regularisation term is added to the original loss function as follows:



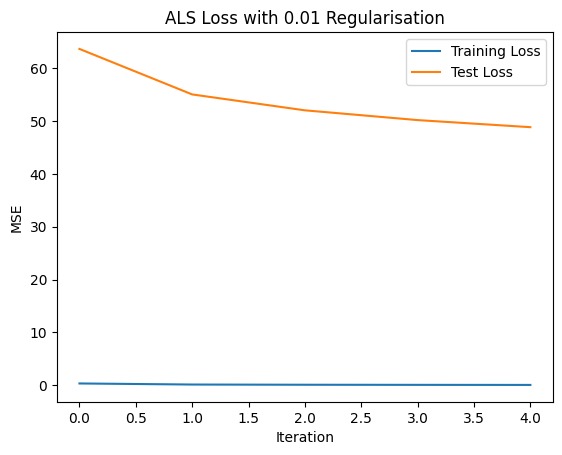
,where λ is the regularisation parameter, controlling the strength of regularisation, and yi represents the weights in the user-item latent matrices. By tuning the regularisation parameter, we aimed to strike a balance between fitting the training data well and avoiding overfitting, ultimately improving the robustness and reliability of our recommendation models.

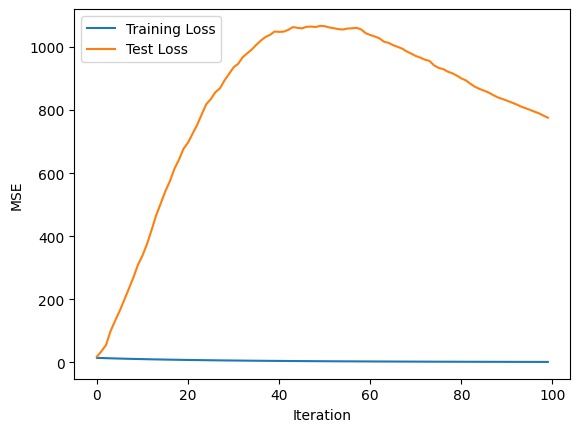
**4. Evaluation & Results**

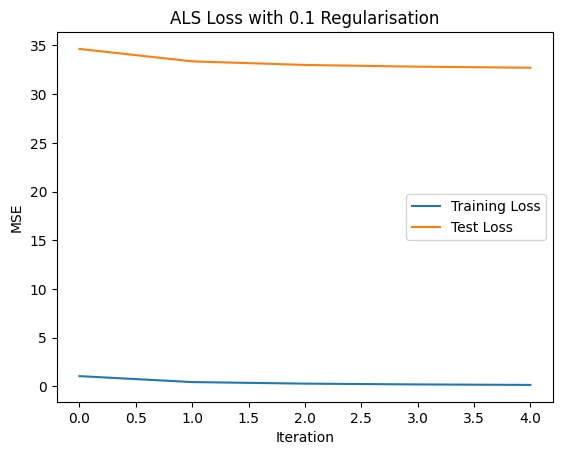
**4.1 Model Performance Metrics**

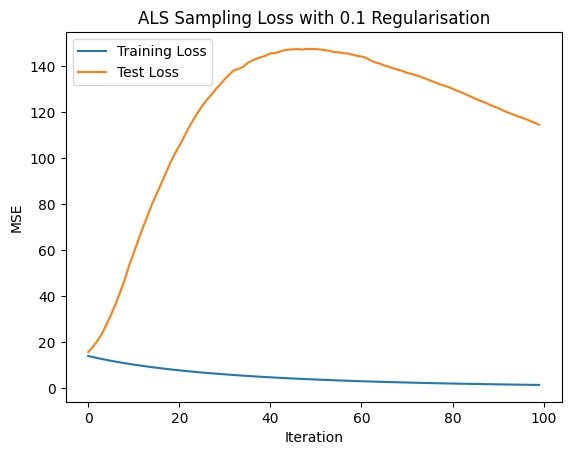
For this Recommendation system, we assess the performance model using the MSE loss metric. This enables us to evaluate how well the model captures the underlying patterns and trends in the data, with lower MSE values indicating better model performance. To gain insights into the effectiveness of the model, we also visualised the results below.

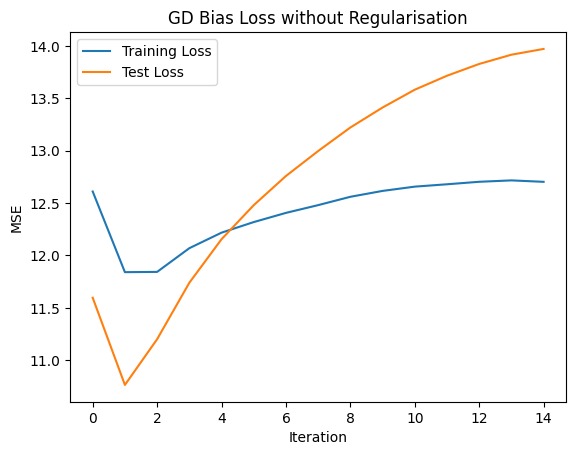
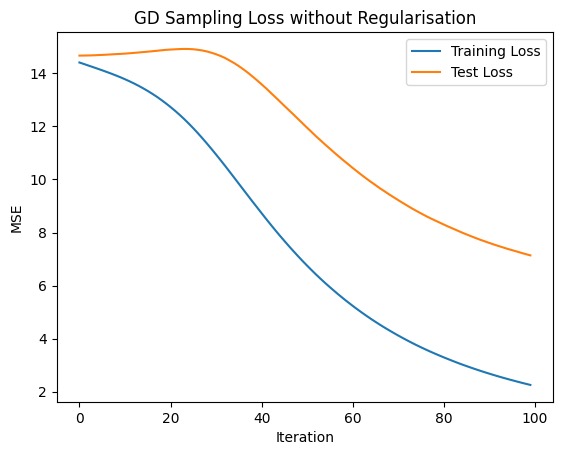
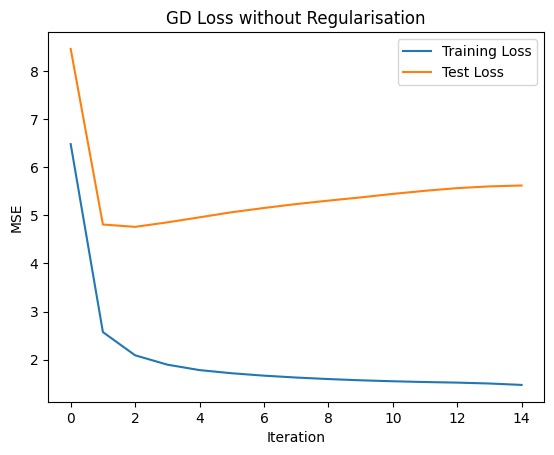
**4.2 Visualisation of Results**

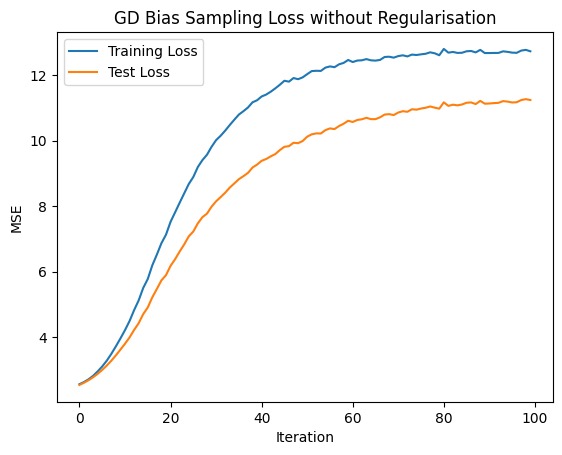
****

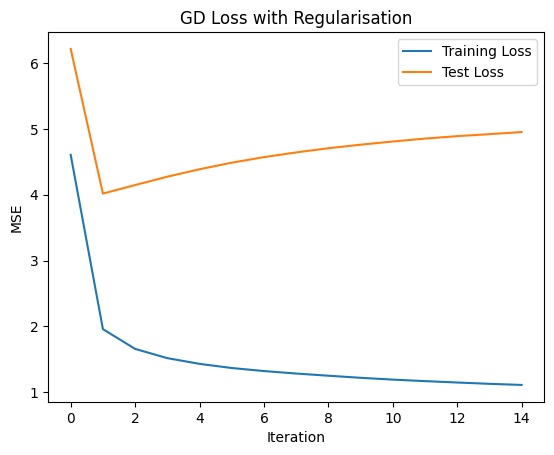
ALS Sampling Loss with 0.01 Regularisation ****

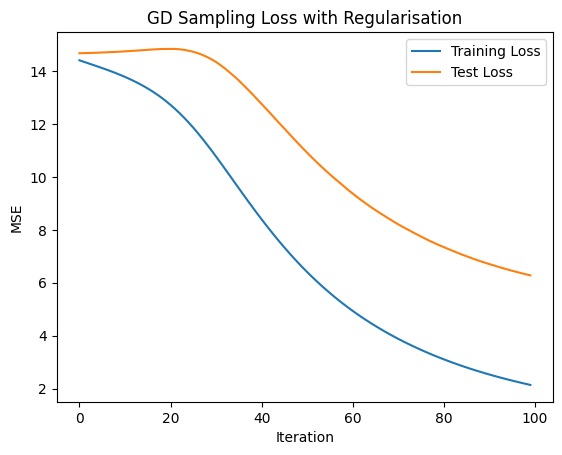
****

****

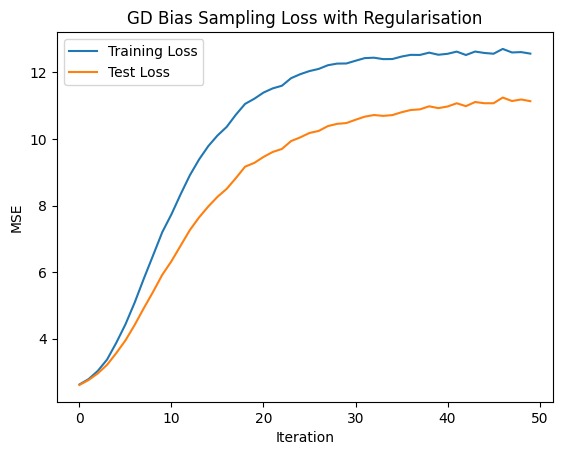
****

****

****

****

****

****

**5. Discussion**

**5.1 Performance analysis**

In the results, we observed that models without regularisation exhibited strong performance during training, as indicated by consistently low loss values. However, during testing, there was a significant increase in loss, suggesting a notable decline in performance and indicating potential overfitting. To address this issue, we implemented regularisation techniques, such as L2 regularisation, to penalise overly complex models and encourage simpler, more generalised solutions. Integrating regularisation improved the model's performance in testing, although further fine-tuning of the regularisation parameter could enhance results.

We only presented results for the regularised ALS method because attempting ALS without regularisation resulted in NaN values in the loss, indicating an exploding gradient problem during training. This further proves the necessity of regularisation for a proper model performance.

Another noteworthy observation is the loss graph for GD with bias. Unlike other models, both training and testing losses increased with each iteration rather than decrease. This anomaly could be attributed to the addition of bias, which may have introduced unnecessary complexity leading to overfitting. Alternatively, improper initialization of the bias weights might have contributed to this issue, although attempts to mitigate it through different initializations yielded minimal change in results.

**5.2 Future improvements**

For further work, we suggest the following:

1. **Cross-Validation and Model Selection:** Employ rigorous cross-validation to improve the model performance. Techniques such as k-fold cross-validation or holdout validation can aid in evaluating the generalisation capability of your models and mitigating issues related to data variability and overfitting.
2. **Implement SVD or SVD++ for Matrix Factorization:** Explore Singular Value Decomposition (SVD) and its variations, including SVD++, as alternative techniques for matrix factorization within collaborative filtering.
3. **Implement LBFGS as an Optimization Method for Matrix Factorization:** Utilise LBFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) as an alternative optimization algorithm for training matrix factorization models in recommendation systems.
4. **Adjust latent factor :** This adjustment allows for fine-tuning the model's flexibility, as increasing the latent factors can capture more intricate user-item interactions, potentially enhancing recommendation accuracy.
5. **Finding the optimal regularisation parameter:** Further experiment with various regularisation parameters to determine the most suitable regularisation parameter for our recommendation system, particularly focusing on mitigating overfitting.

**6. Conclusion**

In conclusion, this project explored the implementation and evaluation of collaborative filtering algorithms for recommendation systems, focusing on matrix factorization techniques and regularisation methods. We implemented and compared Alternating Least Squares (ALS) and Gradient Descent (GD) algorithms, revealing the importance of regularisation in improving model generalisation.

Through experimenting with fine-tuning ALS and GD algorithms, we observed the impact of regularisation on model performance, highlighting its significance in mitigating overfitting and improving generalisation capabilities. Additionally, we encountered challenges such as exploding gradient problems and anomalous behaviour in certain models, underscoring the importance of regularisation and careful model tuning.

More specifically, ALS highlighted the necessity of regularisation, as attempts without it led to training challenges and an anomaly in GD with bias suggested potential issues with model complexity or improper initialization.

Looking ahead, there are several avenues for further improvement, including the exploration of alternative matrix factorization techniques such as SVD and SVD++, optimization methods like LBFGS, and adjustment of latent factors to enhance recommendation accuracy. Moreover, finding the optimal regularisation parameter remains a crucial task for future research, aiming to strike a balance between model complexity and generalisation performance. By addressing these areas, we can continue to advance recommendation systems and deliver more personalised and relevant experiences to users in various domains.

**7. References**

Casalegno, F. (2022, December 12). Recommender Systems - A Complete Guide to Machine Learning Models. Medium. https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748

Dataset for collaborative filters. (2016b, December 22). Kaggle. <https://www.kaggle.com/datasets/rymnikski/dataset-for-collaborative-filters>.

Insight. (2018, June 10). Explicit matrix factorization: ALS, SGD, and all that jazz. Medium. <https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea>