Machine Learning - CS 6140 (Spring 2023)

Restaurant Recommendation System

Submitted by – Pulkit Saharan and Shivani Srikant Naik

Abstract

With the increasing availability of online restaurant reviews, customers rely heavily on them to make dining decisions. The restaurant industry is highly competitive, and people are always looking for new dining experiences. With the ever-growing number of restaurants, it is challenging to find the perfect restaurant that matches the customer's preferences.

Yelp is a popular platform that hosts a large amount of restaurant-related data, including ratings, reviews, and other information. This information can be utilized to develop a restaurant recommendation system that can provide personalized restaurant recommendations to the customers. This system will save customers time and effort and provide relevant restaurant suggestions to them. Additionally, it can benefit restaurant owners and help them to understand customer preferences and make informed decisions about menu, ambiance, and marketing strategies.

We utilized six different algorithms, including SVD, SGD, ALS, Random Forest Regressor, Cosine Similarity, and Pearson Similarity, to predict ratings and make recommendations for users. The Random Forest Regressor performed the best, with the lowest RMSE score of 0.32 for the train set and 1.1 for the test set. It was followed by SGD, with a train RMSE of 1.2 and test RMSE of 1.5. However, the performance of SVD was the worst, likely due to the problem of sparsity. The user-item matrix in our analysis was approximately 95% sparse, which adversely affected the performance of SVD.

Introduction

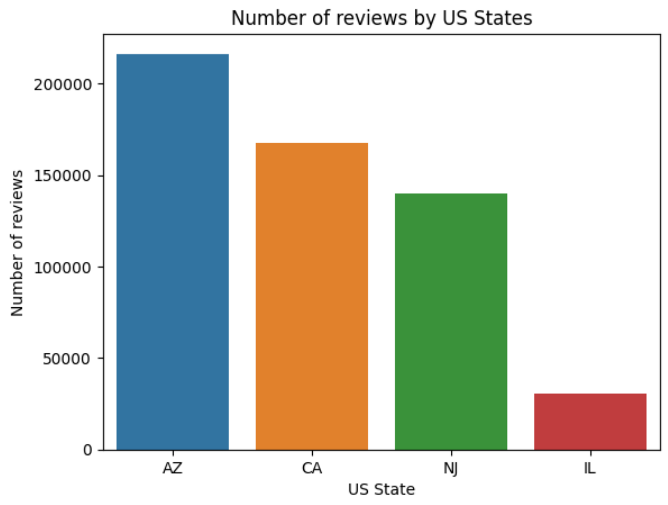
The goal of this report is to introduce our project for a restaurant recommendation system, which intends to offer users a customized dining experience by using cutting-edge algorithms to assess their dining preferences and history. People's dining preferences have changed as the digital age has progressed, with a greater emphasis now being placed on internet food discovery. Our technology provides individualized recommendations for interesting new eateries to consider as a solution to this issue.
We think that the recommendation engine, which has grown in popularity as a tool for online buyers to make educated choices, may also be used in the restaurant business. Our technology generates recommendations by examining user data and activity and comparing them to specific preferences. Our goal is to create a system that not only helps users discover new dining experiences but also enhances their overall dining journey.
In this report, we will discuss the background and motivation for the project, as well as the methodology and techniques used in developing the recommendation system. We will also present the results of our system's performance evaluation and discuss potential areas for improvement. Finally, we will conclude by summarizing the project's achievements and discussing its potential impact on the dining industry.

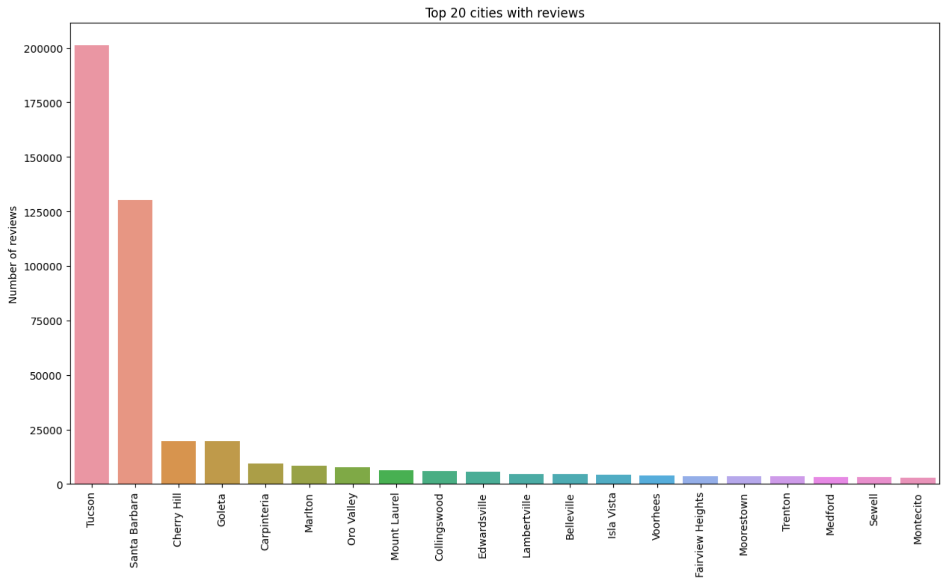
Data

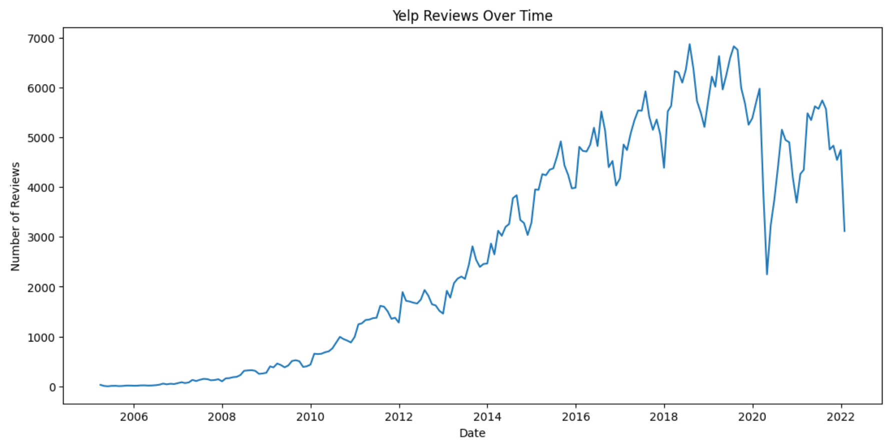
For our restaurant recommendation system project, we utilized data from Yelp, a popular online platform for reviews and recommendations on businesses. The Yelp dataset, which is available on an open-source website called Kaggle - Yelp Dataset | Kaggle, consists of five different JSON files containing information about businesses, check-ins, reviews, users, and tips. This huge dataset is approximately 8GB in size, and to make it feasible for us to use it for our use case, we utilized Kaggle notebooks to preprocess initial data and reduce it. This data subset was then downloaded and utilized for this work.
To focus specifically on the restaurant business, we filtered out data from the businesses file, which contained information on over 11,000 businesses and more than 1,200 business categories. We only included businesses that fell under the category of restaurants, and selected data from four states, including Arizona, New Jersey, Illinois, and California, to manage the size of the dataset.
To enhance the performance of our recommendation system, we initially screened users based on their review count, and excluded those with less than five reviews. We then selected users with more than ten reviews to provide a substantial amount of data for analysis.
From the original data, we focused on important columns that contained relevant information about the business, such as its ID, name, and location, as well as information about the user, such as their name and ratings given to businesses. By analyzing this data, our recommendation system can provide personalized and informed restaurant suggestions to users based on their individual preferences and past dining experiences.

Exploratory Data Analysis

At the outset, we explored the data through plots to identify patterns and trends.
The following graphs display the US states with the highest number of reviews, the cities with the highest number of reviews, and the trend of reviews over the years. We observed that the number of reviews has increased rapidly in recent years, which is likely due to the proliferation of restaurants and platforms for reviewing them.







Data Cleaning and Preprocessing

1. JSON conversion and merging

As mentioned in the previous section, the dataset comprised of 5 JSON files with a total size of 8GB, which was computationally infeasible for our project to handle. To overcome this challenge, we performed data preprocessing on Kaggle, where we filtered the business category for restaurants in the top states and cities. We then created a CSV file with the required data and used it for further analysis.

2. Filtering states and users

To manage the size of the dataset, we selected data only from four states, namely Arizona, New Jersey, Illinois, and California. Additionally, we only included businesses that were categorized as restaurants in the dataset.

3. Create Utility Matrix

The utility matrix is a matrix that represents the preferences of users with restaurants in the recommendation system. The utility matrix is a two-dimensional matrix, where each row represents a unique user, and each column represents a restaurant. The values in the matrix give the rating that the user has given to a particular restaurant.

Utility matrices can be sparse, so most of the cells in the matrix are empty because users have not interacted with or rated most of the items. This sparsity makes it challenging to compute similarities between users or items and to generate accurate recommendations. It is common for utility matrices to have 95-99% sparsity, which makes the problem of predicting ratings and generating recommendations non-trivial.

For our dataset, when we use city filtered data and take users with minimum 10 ratings, the sparsity ranges from 91-96%. Dealing with sparsity is one of the biggest challenges that we faced in this project.

Methodology

Recommendation System Strategies

A recommendation system is used to recommend products, services or items to users based on their preferences. Having personalized recommendations benefits the users and businesses with increased interaction and profit. There are different approaches for building a recommender system:

1. Collaborative filtering: Collaborative filtering considers users likes and dislikes from user history. User-based collaborative filtering recommends items liked by users like the target user.
2. Content-based filtering: Content- based filtering recommends items that are similar to items that target user has liked in the past based on item features
3. Hybrid: This type of system combines both collaborative and content-based filtering

In this work, we have explored different collaborative filtering strategies.

Collaborative Filtering

Collaborative filtering is a technique used in recommendation systems to provide personalized recommendations to users based on their past behavior and the behavior of other similar users. Collaborative filtering works by analyzing the similarities between users' preferences and making recommendations based on those similarities.

Types of collaborative filtering:

1. User-based filtering: In this method, we find similar users to target user based on a distance metric. We then use the average ratings of these users as the predicted rating for target user and recommend restaurants that are new to target user.
2. Item-based filtering: In this method, we find new restaurants similar to the restaurants that the target user has interacted with before. It involves calculating the similarity between restaurants based on the users who have interacted with them, and using this similarity measure to recommend similar restaurants that the target user may also like.

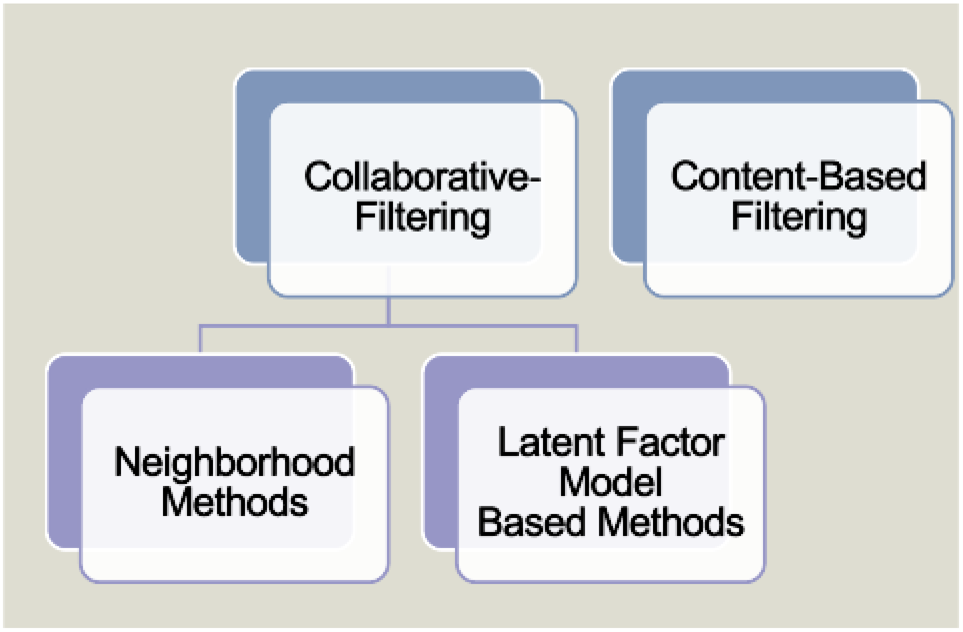
Train Test Split

For creating the train test sets for modeling, the utility matrix is split into two matrices. Both train and test matrices are of same dimensions. For each user, we transfer 30% of their ratings randomly to the test set. This ensures that all users are present in both the training and testing set.

Model Building

According to the chart, there are two types of recommendation systems: Collaborative Filtering and Content-based Filtering. Our project focuses on Collaborative Filtering, which has two approaches for predicting ratings and making recommendations: Neighborhood Method and

Latent Factor Model Based Methods. We have implemented both of these methods in our project, while Content-based Filtering is not within the scope of our project.



Neighborhood Methods

The Neighborhood Method, also known as Memory-based Collaborative Filtering, predicts a user's rating for an item by finding similar users or items and using their ratings as a basis for prediction. It involves dividing users or items into groups based on similarity and using the ratings of those similar users or items to make recommendations.

Latent Factors Model Based Methods

Latent Factor Model Based Methods, also known as Model-based Collaborative Filtering, use statistical and machine learning algorithms to learn the latent factors that influence the user-item interactions. These methods assume that there are underlying factors, such as preferences or tastes, that affect how users interact with items. By learning these latent factors, the model can predict how users would rate unseen items and make recommendations accordingly.

1. Singular Value Decomposition (SVD)

Collaborative Filtering for recommendation systems can use Singular Value Decomposition (SVD), a matrix factorization technique.
In SVD, a user-item rating matrix is divided into three matrices, U, and V, where U represents the users, represents the strengths of the latent components, and V represents the items. The singular values, represented by the diagonal values of, are used to rank the weight of the latent factors.
By locating the key latent components that most significantly contribute to the variability in the data, SVD can be used to minimize the dimensionality of the original

data. Recommendation calculation is accelerated and made more effective by the reduced dimensionality.
It has been demonstrated that SVD-based techniques perform less well in recommendation systems for sparse datasets, where many ratings are absent. Additionally, SVD's scalability may be constrained by the computational cost of handling huge datasets.

2. Stochastic Gradient Descent (SGD)

In the SGD model for collaborative filtering, we learn latent feature vectors for each user and restaurant. In addition to the latent feature vectors, a bias term is also learned for each user and restaurant. This bias term is used to account for the tendency of some users or restaurants to give or receive higher ratings on average compared to others. By learning these bias terms, we can ensure that the latent feature vectors are free from these biases.
The final rating given by a user to a restaurant is broken down into four components: a global bias (which is the average of all available ratings), a user bias, a restaurant bias, and a component that depends on the interaction between the user and restaurant. Stochastic gradient descent is used to learn all of the model parameters, including the bias terms and latent feature vectors.
This model is called the SGD model because it uses stochastic gradient descent to learn the parameters. By learning these parameters, the model can accurately predict missing ratings and make personalized recommendations for each user.

3. Alternating Least Squares (ALS)

In the Alternating Least Squares (ALS) method for collaborative filtering, each user and restaurant is represented by a k-dimensional feature vector containing k latent features that are learned by the model. Each feature dimension represents a different attribute of the user or restaurant. The data instance for the model includes a randomly initialized k-dimensional vector representing the user, a randomly initialized k-dimensional vector representing the restaurant , and the rating given by the user to the restaurant.
The goal of the model is to predict the rating given by the user to the restaurant from the user and restaurant feature vectors . The prediction function is a simple dot product between the feature vectors. The model learns the optimal values of user features and item features by minimizing the root mean squared error.
Once the feature vectors have been learned, the model can be used to predict missing ratings in the user-item rating matrix. By filling in these missing values, the matrix can be completed, allowing for personalized recommendations to be made for each user. This problem can be treated as a typical supervised learning problem, where the feature vectors for users and items are the input variables and the ratings are the target variable. By training the model on a large set of ratings, it can make accurate predictions for new users and items.

4. Regression Problem with Random Forest

To predict user-item ratings, we transformed the problem to a regression problem and used a Random Forest Regressor. We used latent feature vectors learned through SGD as input vectors, and ratings as the target variable. The latent feature vectors for a specific user-item pair were concatenated to create the independent variable for the model. The model was trained to predict the rating a user would give to an item they have not yet rated based on the behavior of other similar users. This approach allows us to leverage information from similar users to make accurate predictions, and has been successfully used in various applications such as movie and product recommendations.

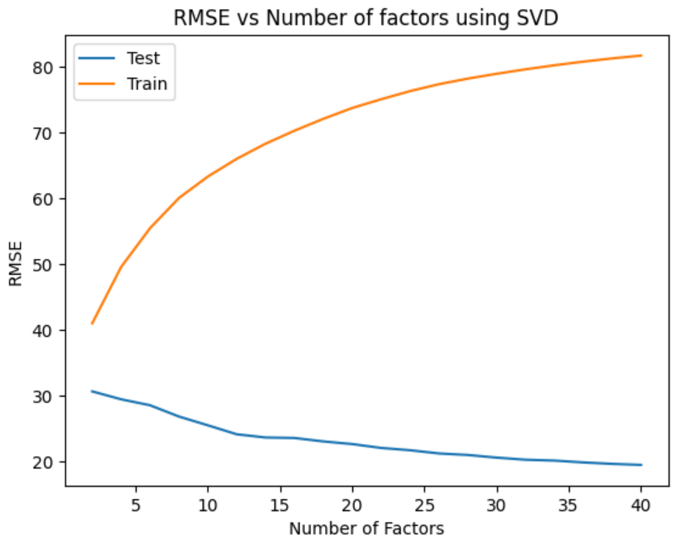
Performance Evaluation

In order to assess the effectiveness of our algorithms, we utilized the Root Mean Squared Error (RMSE) metric. This metric is appropriate for continuous target variables, and it quantifies the square root of the average of the squared differences between the predicted ratings and the actual ratings.

Results

1. Singular Value Decomposition (SVD)

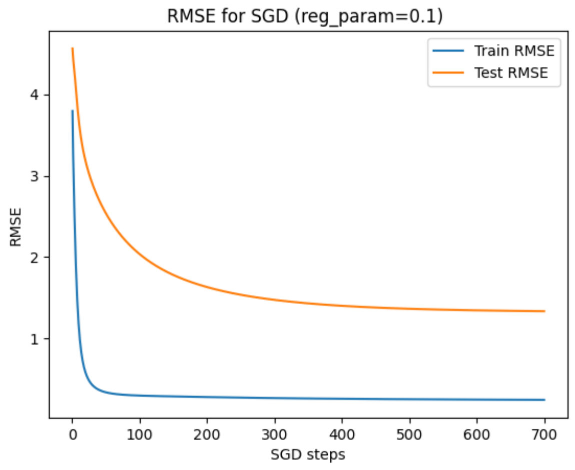
The graph below depicts the errors generated by the SVD algorithm, as measured by the Root Mean Squared Error (RMSE). The RMSE values are shown for both the train and test datasets, as the number of factors increases. The number of factors corresponds to the latent features learned by the algorithm. The RMSE value for the train dataset is 81.6, while the RMSE value for the test dataset is 19.4

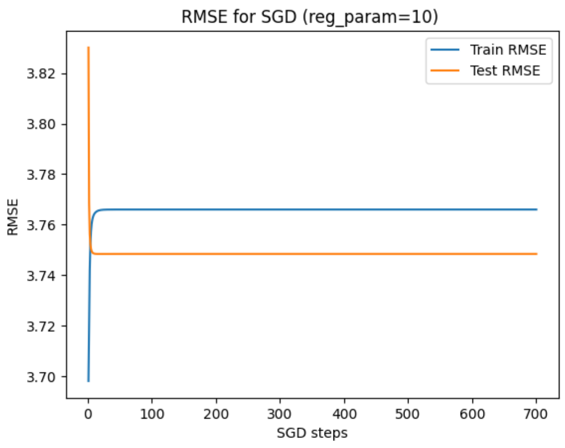


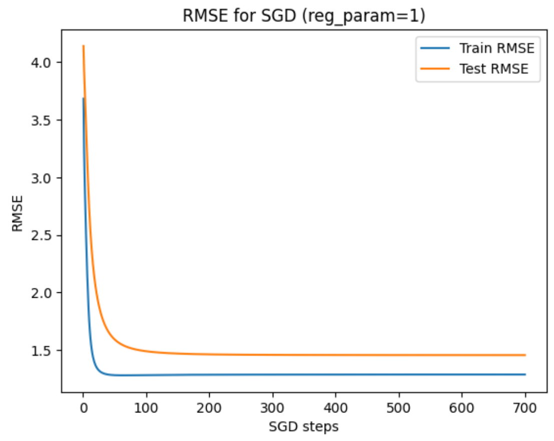
2. Stochastic Gradient Descent (SGD)

The performance of the SGD model is depicted in the plots below, which illustrate the Root Mean Squared Error (RMSE) for different values of hyperparameters. Specifically, the plots show the RMSE values for different numbers of latent factors for both the train and test datasets. Three separate plots are shown for different values of regularization parameters.

The plot reveals that the RMSE decreases as the number of latent factors increase in both the train and test datasets and converges to 0.4 in the train set and 1.5 in the test set when the regularization parameter is 0.1. However, the RMSE increases as the value of the regularization parameter increases. Notably, when the regularization parameter is 10, the train and test set RMSE values increase significantly.





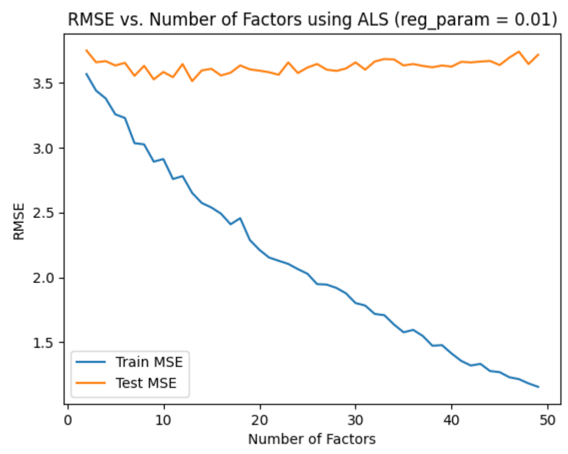


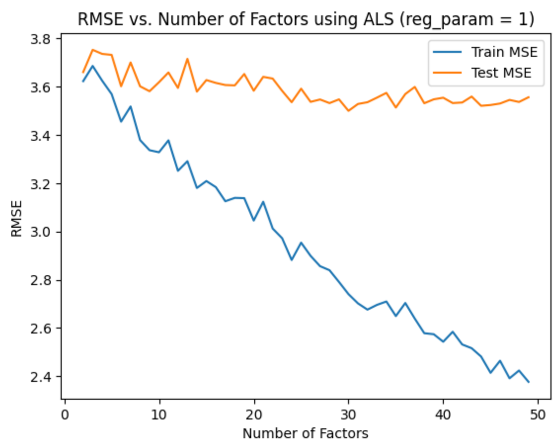
3. Alternating Least Squares (ALS)

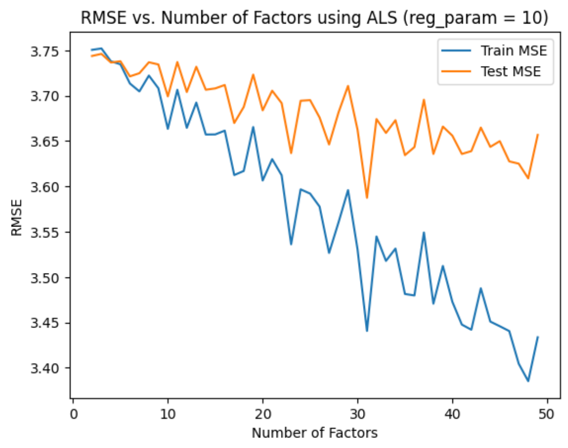
The results of the ALS algorithm are presented in the form of line graphs, which illustrate the Root Mean Squared Error (RMSE) for the number of latent factors for users and items, as well as different values of the regularization parameter.

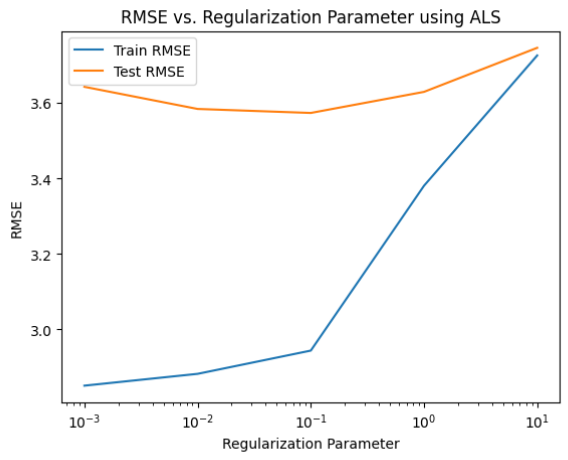
The model's performance in terms of RMSE was best when the regularization parameter was set to the lowest value, 0.01. The RMSE value for the train dataset decreased to below 1.5, while the test dataset remained relatively constant around 3.6. However, as we increased the regularization parameter, the learning process slowed down, and the RMSE values remained somewhat higher than when the parameter was low.

This same pattern is observed in the plot of RMSE versus the regularization parameter. Specifically, as the parameter value increases, the RMSE of the train set increases, while the RMSE of the test set remains in the same range.







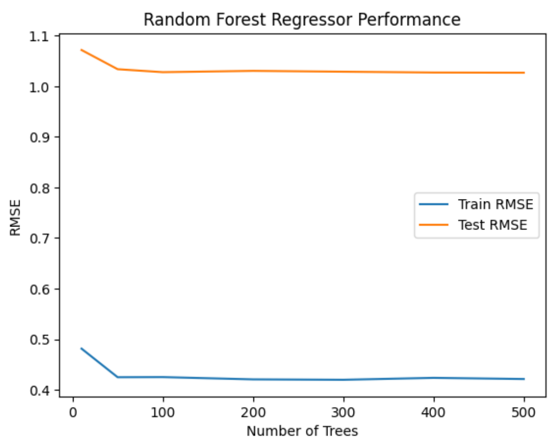


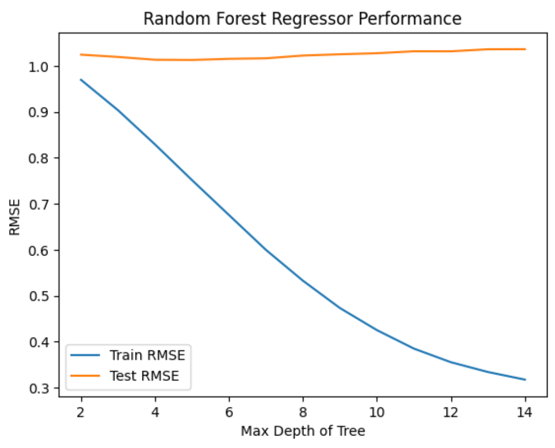
4. Regression Problem with Random Forest

The results obtained from the implementation of the Random Forest Regressor for predicting ratings are presented in the form of line graphs, which plot the Root Mean Squared Error (RMSE) against the number of trees and the maximum depth of the tree.

The graph that plots RMSE versus the number of trees indicates that the RMSE decreases as the number of trees increases, until it reaches 50, after which it remains relatively constant. The RMSE values converge to 0.3 for the train set and 1.1 for the test set.

The plot of RMSE versus the maximum depth of the tree also shows similar RMSE values for the train and test sets. However, as the maximum depth increases, the train RMSE keeps decreasing, which indicates overfitting of the model. Meanwhile, the test RMSE increases slightly.





Summary of all the models

Type	Model	Train RMSE	Test RMSE
Latent Factor Model Based Methods	SVD (Singular Value Decomposition)	81.6	19.4
Latent Factor Model Based Methods	ALS (Alternating Least Square)	2.3	3.6
Latent Factor Model Based Methods	SGD (Stochastic Gradient Descent)	1.2	1.5
Latent Factor Model Based Methods	Random Forest Regressor	0.32	1.1
Neighborhood Method	Cosine Similarity	2.5	3.41
Neighborhood Method	Pearson Similarity	2.5	3.45

After analyzing the results of all the algorithms, we can conclude that the Random Forest Regressor performs the best in terms of the Root Mean Squared Error (RMSE) for both the train and test sets. It has the lowest RMSE values among all the models. The SGD algorithm also performed well, with an RMSE value slightly higher than the Random Forest Regressor but lower than the other models.

Future Scope

- One area for future development is to explore different ensemble techniques that can improve the accuracy of our model even further. This could include using techniques such as stacking or boosting to combine multiple models.
- Another area for future work is to develop more sophisticated content-based recommendation systems that take into account more information about each item, such as its cuisine, location, or prices.
- Addressing the cold start problem remains an important area for future research, and one possible approach is to use multi-armed bandit algorithms that balance exploration and exploitation to learn more about new users and items.
- Finally, to make our system more scalable, we could explore distributed computing frameworks such as Apache Spark or Apache Flink, or use cloud-based services such as

Amazon Web Services or Google Cloud Platform to handle larger amounts of data and higher traffic.

Reflection

This project has been a great learning experience for us because recommendation systems is a new area that we had not explored previously. From this project, we gained a key understanding of the concepts and techniques involved in building recommendation systems, including collaborative filtering, utility matrices, SVD, ALS, SGD and Random Forest Regressor. In addition, we also got to work with a large real-world dataset like the Yelp dataset. This project also introduced us to the challenges of dealing with sparse matrices.
Overall, this work has provided us with a solid foundation in recommendation systems. As this is an exciting and rapidly evolving area with enormous potential for innovation, we are extremely grateful for having had an opportunity to work on this problem statement.

Acknowledgement

We would like to thank Professor Bruce for giving us the opportunity to work on this project, even though it was outside of the course curriculum. We got to explore new areas, and this was a great learning experience for us as students.

1. Yelp, I. (2022, March 17). *Yelp dataset*. Kaggle. Retrieved April 6, 2023, from https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset
2. Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal. Retrieved April 6, 2023, from https://www.sciencedirect.com/science/article/pii/S1110866515000341