Netflix Movie Recommendation System

RAGUL VENKATARAMAN RAVISANKAR, University of Colorado Boulder, USA
ARAVINDAKUMAR VIJAYASRI MOHAN KUMAR, University of Colorado Boulder, USA
RAWSHON FERDAWS, University of Colorado Boulder, USA

1 INTRODUCTION

Recommender Systems are a way of predicting what rating or preference a user would give to an item. During the last few decades, with the rise of YouTube, Amazon, Netflix and many other such web services, recommender systems have become more vital. Recommender systems can generate huge revenue when they are efficient. It is also a way to stand out significantly from competitors. In this project, we will take the Netflix prize dataset and build a recommender system to recommend movies to the users based on various techniques.

2 RELATED WORK

There are various techniques to recommend movies to the users.

2.1 Collaborative-Filtering

This technique filters information by using the interactions and data collected by the system from other users. It is based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future. Collaborative-filtering systems focus on the relationship between users and items. There are two classes of Collaborative-Filtering:

(1) **User-User Similarity** This measures the similarity between target users and other users. With the similarity score we can compare each user among the rest of n - 1 users. The higher the similarity between vectors, the higher the similarity between users. The similarity matrix (U) for n users is represented by

$$U = \begin{bmatrix} 1 & sim(1,2)... & sim(1,i)... & sim(1,n) \\ . & 1 & . & sim(2,n) \\ . & . & . & . \\ . & . & 1 & . \\ . & . & . & . \\ sim(n,0) & sim(n,1).. & sim(n,i).. & 1 \end{bmatrix}$$

(2) **Item-Item Similarity** This measures the similarity between the items that target users rate or interact with and other items. The similarity matrix (I) from m items is represented by

Authors' addresses: Ragul Venkataraman Ravisankar, University of Colorado Boulder, Boulder, USA, rave2101@colorado.edu; Aravindakumar Vijayasri Mohan Kumar, University of Colorado Boulder, Boulder, USA, arvi7401@colorado.edu; Rawshon Ferdaws, University of Colorado Boulder, Boulder, USA, rafe7973@colorado.edu.

$$I = \begin{bmatrix} 1 & sim(1,2)... & sim(1,i)... & sim(1,m) \\ . & 1 & . & sim(2,m) \\ . & . & . & . \\ . & . & 1 & . \\ . & . & . & . \\ sim(m,0) & sim(m,1).. & sim(m,i).. & 1 \end{bmatrix}$$

3 PROPOSED WORK

We are planning to take the outputs from the multiple baseline models (built on user-user similarity, SVD etc) and feed them as inputs into our recommender model and analyze how the accuracy can be improved. We are planning to use the Netflix Prize dataset

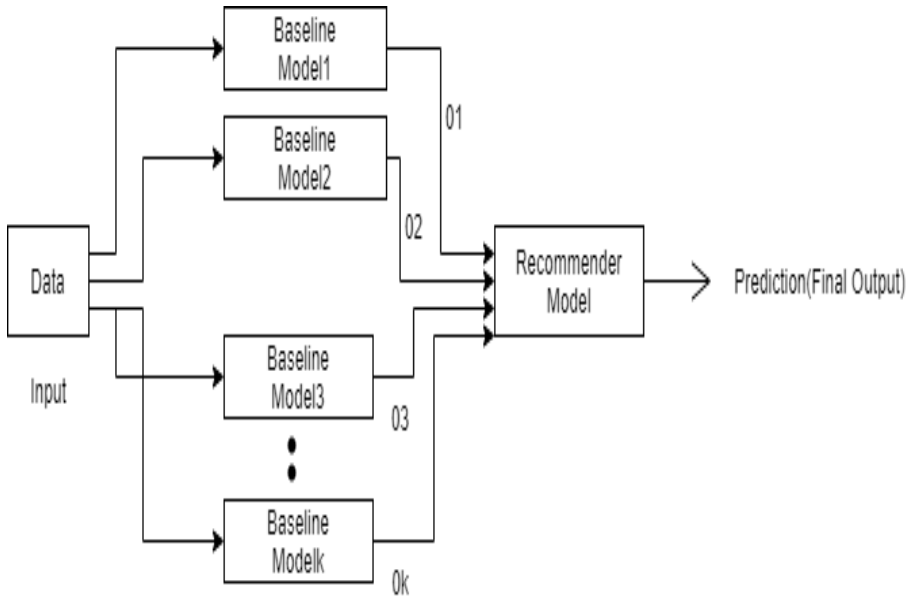

Fig. 1. Proposed System

4 CHALLENGES

4.1 Cold Start Problem

Recommender Systems generally recommends items to the user based on the historical data. When a new user signs up to the product there won't be any historical data about that user. This is called the Cold-Start problem. The cold-start problem is usually handled using Content-based filtering techniques. For example, recommending the most popular items to the new users.

4.2 Business Metric

In Recommender Systems, there are no standard techniques to measure how good the system performs. Defining a proper business metric in such scenarios to evaluate the accuracy of the system is challenging.

5 DATA PREPROCESSING

- The given user-movie rating dataset was in a text file format we converted it into the csv file to read it as a dataframe (User, Movie, Rating, Rating Date).
- Data Cleaning was not required as there were no empty values.
- We split them into train test data based on an 80:20 split ratio.
- We tried to represent the given data in the form of a sparse matrix, because there are a lot of zero values as users would have watched only a handful of movies. Therefore, for most of the movies the ratings won't be present. To utilize the space efficiently, we went with the sparse matrix representation.
- We then compute user-user similarity (how the given user is similar to all the other users) and movie-movie similarity (how the given movie is similar to all the other movies) using cosine similarity.

6 EXPLORATORY DATA ANALYSIS

For the given training dataset we took basic statistics of the number of unique users, movies and ratings. There were around 405k users, 17k Movies and 100M ratings. The histogram of ratings for the training data set is shown below:

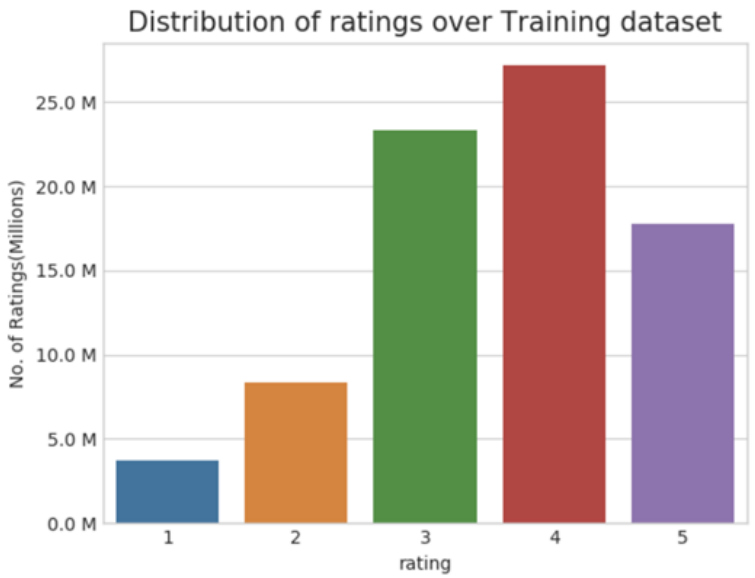

Fig. 2. Distribution of ratings over Training dataset

We also tried to analyze if the day of the week has any significance on the ratings. But on plotting the graph, we found that the above feature had no significance.

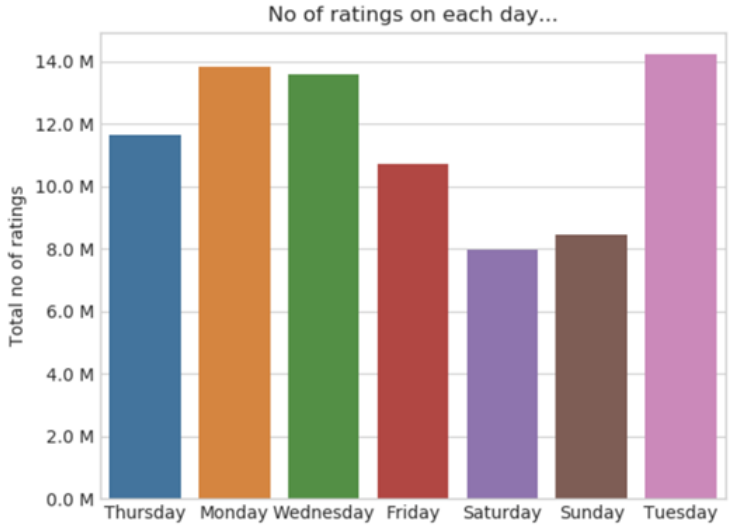

Fig. 3. Ratings given on each day of the week

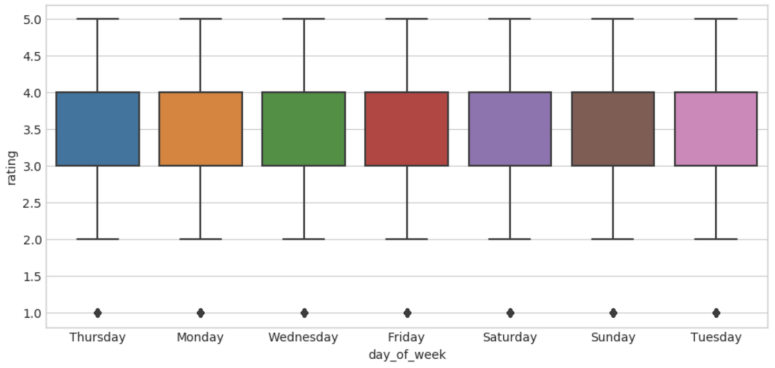

Fig. 4. Box Plot of Days of Week

The probability density function (PDF) was obtained for user-average and movie-average ratings. From the graph we can see that, they are not following a normal distribution but a skewed distribution.

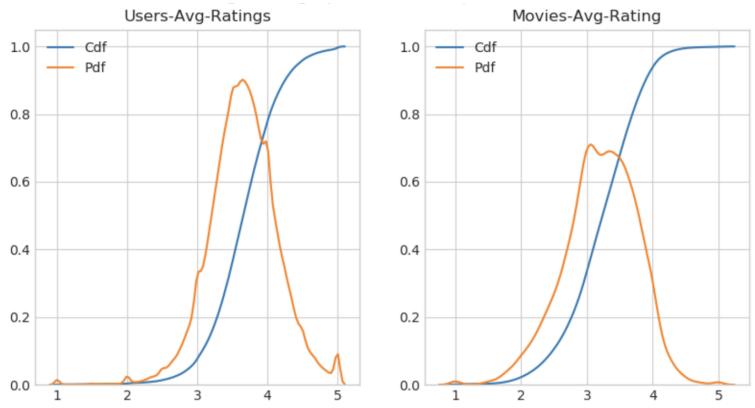

Fig. 5. User Average and Movie Average Ratings

7 FEATURE ENGINEERING

From the given user-movie matrix we are trying to derive features and pose it as a regression problem. There are thirteen handcrafted features.

- Global Average Rating
- User Average Rating
- Movie Average Rating
- Top 5 similar user rating
- Top 5 similar movie rating

We will feed the above features as input to the XgBoost baseline model.

8 ARCHITECTURE

We are using the 13 handcrafted features for the XgBoost baseline model and other Surprise Baseline models, and then take the outputs of each model and give as features to the final XgBoost model along with the 13 handcrafted features.

8.1 XgBoost Baseline

XgBoost is an open-source library, which implements high-performance Gradient-Boosted Decision Trees(GBDT). The underlying implementation is using C++ and on top of that there is a Python wrapper making it very fast for regression problems.

We feed the 13 handcrafted features to the XgBoost baseline model.

8.2 Surprise Baseline

Surprise is a Python scikit library for building and analyzing recommender systems that deal with explicit rating data.

In the Surprise Baseline model we will try to predict the movie rating based on global average, user's average rating and movie's average rating received.

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

Equation Source: https://surprise.readthedocs.io/en/stable/basic_algorithms.html

μ : Global Average Rating
b_u : User bias
b_i : Item bias
$\hat{r}_{ui}$: Predicted rating for a movie i by a user u

8.3 Surprise KNN model

KNNBaseline is a collaborative filtering algorithm which takes the baseline rating into consideration. We built two models, one for users and one for movies.

- KNNBaseLineUser: We predict the rating based on the top-k similar users' rating to that movie (k is the hyper parameter here)
- KNNBaseLineMovie: We predict the ratings based on top-k similar movies to the given movie. The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v).(r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

Equation Source: https://surprise.readthedocs.io/en/stable/knn_inspired.html

$\hat{r}_{ui}$: Predicted rating for a movie i by a user u
sim(u,v): Pearson similarity for u and v (u and v are either users or items)

8.4 Matrix Factorization

Matrix factorization is a collaborative filtering algorithm which works by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices.

(1) **Singular Value Decomposition (SVD)**

$$A = USV^T$$

where

U, V: Orthogonal matrices with orthonormal eigenvectors
S :Diagonal matrix with square root of positive eigenvalues

The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Equation Source: https://surprise.readthedocs.io/en/stable/matrix_factorization.html

(2) **SVD++ (or) SVDpp**
It is an extension of SVD technique which also takes implicit ratings into consideration. The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j\right)$$

Equation Source: https://surprise.readthedocs.io/en/stable/matrix_factorization.html

Using techniques like SVD and SVD++ we will try to factor the input sparse matrix and predict the ratings for a movie for a given user.

8.5 XgBoost Final Model

We take output from all the above models and in addition the 13 handcrafted features we provide in total of 19 features to the XgBoost final model and predict the rating.

9 EVALUATION

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - O_i)^2}$$

Si : Predicted values
Oi : Observed values
n : Number of samples

The performance of our model will be evaluated based on how close the predicted score is to the actual score. The RMSE value should be as minimal as possible.

10 HYPERPARAMETER TUNING

Hyperparameter is a parameter on which the model performs it's best for the given data. The process of determining this ideal hyperparameter is called hyperparameter-tuning.

There are two types of hyperparameter tuning.

- Grid Search : The search operation is performed in a manually specified subset of hyperparameter space
- Random Search : Random combinations of hyperparameters are used to find the best argument in the given range

We performed the hyperparameter tuning using Grid Search technique on four models namely KNN User, KNN Movie, SVD and SVDpp

11 RESULTS

11.1 XgBoost Baseline model Feature importance

From the plot below, we can observe that **movieAvg** has the highest feature importance for the XgBoost baseline model followed by the **userAvg**. Similarity rating of the fifth user (sur5) has the least feature importance.

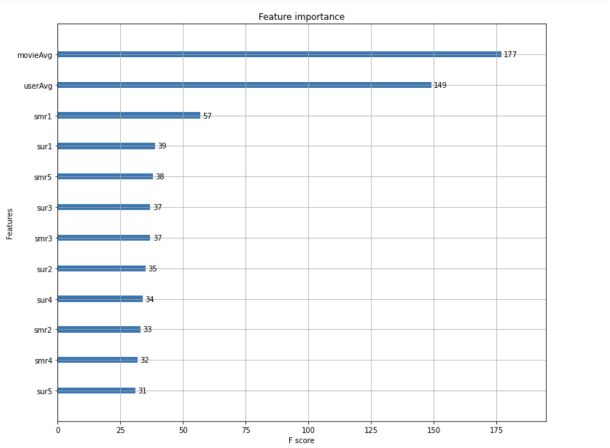

Fig. 6. XgBoost Baseline model Feature importance

11.2 XgBoost Final model Feature importance

From the plot below, we can observe that **baseline_xgb** has the highest feature importance for the XgBoost final model followed by the movieAvg. Similarity rating of the second user (sur2) has the least feature importance.

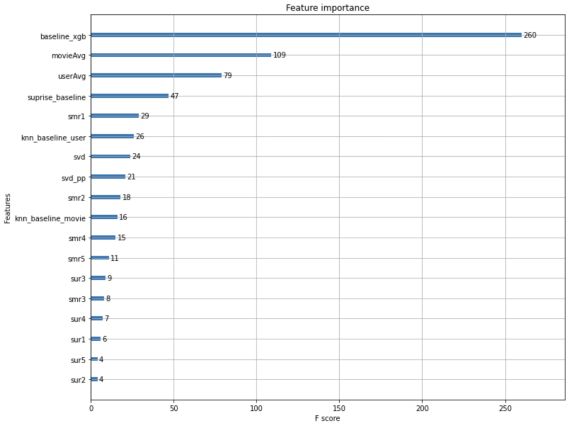

Fig. 7. XgBoost Final model Feature importance

11.3 RMSE Plot on Test Data

From the RMSE plot for the test data for all the different models we observe that **baseline_xgb** has the least RMSE value followed by **xgb_final**. For the other models the RMSE values are slightly higher than these two models.

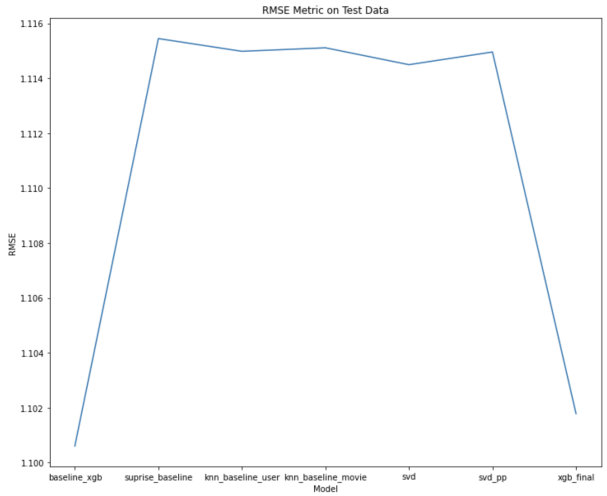

Fig. 8. RMSE Metric on Test Data

11.4 MAPE plot on Test Data

Similar to the RMSE plot on test data the MAPE values for both **baseline_xgb** and **xgb_final** are the lowest. For the other models MAPE values are slightly higher than these two models.

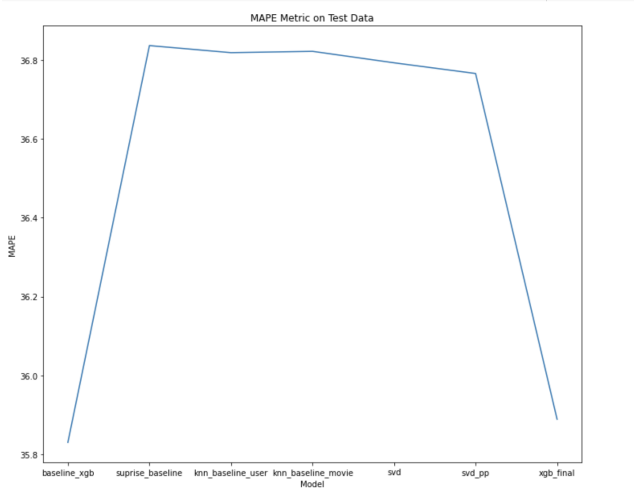

Fig. 9. MAPE Metric on Test Data

11.5 RMSE Values for different models

```
baseline_xgb            1.1006
xgb_final              1.10178
svd                    1.11449
svd_pp                 1.11496
knn_baseline_user      1.11498
knn_baseline_movie     1.11511
suprise_baseline       1.11545
Name: rmse, dtype: object
```

Fig. 10. RMSE Values for different models

ACKNOWLEDGMENTS

We sincerely thank our Professor Qin Lv and teaching assistant Yichen Wang for this wonderful opportunity. We had a great time exploring various techniques of recommender systems and we were able to come up with an ensemble approach to give better recommendations. By submitting this work we adhere to the honor code pledge: **"On our honor, as University of Colorado Boulder students, we have neither given nor received unauthorized assistance"**

12 CODE CONTRIBUTIONS

- **Aravindakumar Vijayasri Mohan Kumar**: EDA, KNN user, KNN Movie, SVD++, Final XgBoost, Hyperparameter Tuning, Documentation
- **Ragul Venkataraman Ravisankar**: Feature Engineering, XgBoost Baseline, Surprise Baseline, Final XgBoost, Hyperparameter Tuning, Documentation
- **Rawshon Ferdaws**: Data preprocessing, EDA, SVD, Documentation

Git Hub Repository: Netflix Movie Recommendation

REFERENCES

[1] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. Commun. ACM 53, 4 (April 2010), 89–97. DOI:https://doi.org/10.1145/1721654.1721677

[2] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

[3] R. M. Bell, J. Bennett, Y. Koren and C. Volinsky, "The million dollar programming prize," in IEEE Spectrum, vol. 46, no. 5, pp. 28-33, May 2009, doi: 10.1109/MSPEC.2009.4907383

[4] P. Bedi, C. Sharma, P. Vashisth, D. Goel and M. Dhanda, "Handling cold start problem in Recommender Systems by using Interaction Based Social Proximity factor," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015, pp. 1987-1993, doi: 10.1109/ICACCI.2015.7275909.

[5] A Beginner's guide to XGBoost

[6] Matrix factorization (recommender systems)

[7] Machine Learning - Singular Value Decomposition (SVD) and Principal Component Analysis (PCA)

[8] Building and Testing Recommender Systems With Surprise, Step-By-Step