

Capstone project report:

IMPROVING NETFLIX MOVIE RECOMMENDATIONS

Mathias Saver.
January 2018.

NETFLIX

The Client ?:

NETFLIX, an online entertainment platform that offers streaming of movies and TV series

The Problem ?:

NETFLIX relies on an accurate recommender system to provide users with a selection of digital content that is most likely to be enjoyed by the users

**Can the accuracy of predicted ratings
be improved ?**

Obtaining the Data: The Netflix Prize Dataset

- Originally released as part of an online competition to improve Netflix recommendation engine
- Downloaded from :
<https://www.kaggle.com/netflix-inc/netflix-prize-data>
- The entire data set contains 4 .txt files, comprising 100480507 ratings, by 480189 users, for a collection of 17770 movies
- The data set also contains a .csv file with a list of movie IDs and titles

Data Wrangling

The format of the data in the .txt files is as follows:

MovieX:

Customer ID_A, rating, date of rating

Customer ID_B, rating, date of rating

Movie Y:

Customer ID_C, rating, date of rating

CustomerID_D, rating, date of rating

The 4 .txt files were compiled and formatted into a single data frame, as shown here:

	User ID	Rating	Date	Movie ID
1	1488844	3	2005-09-06	1
2	822109	5	2005-05-13	1
3	885013	4	2005-10-19	1
4	30878	4	2005-12-26	1
5	823519	3	2004-05-03	1

Data Wrangling:

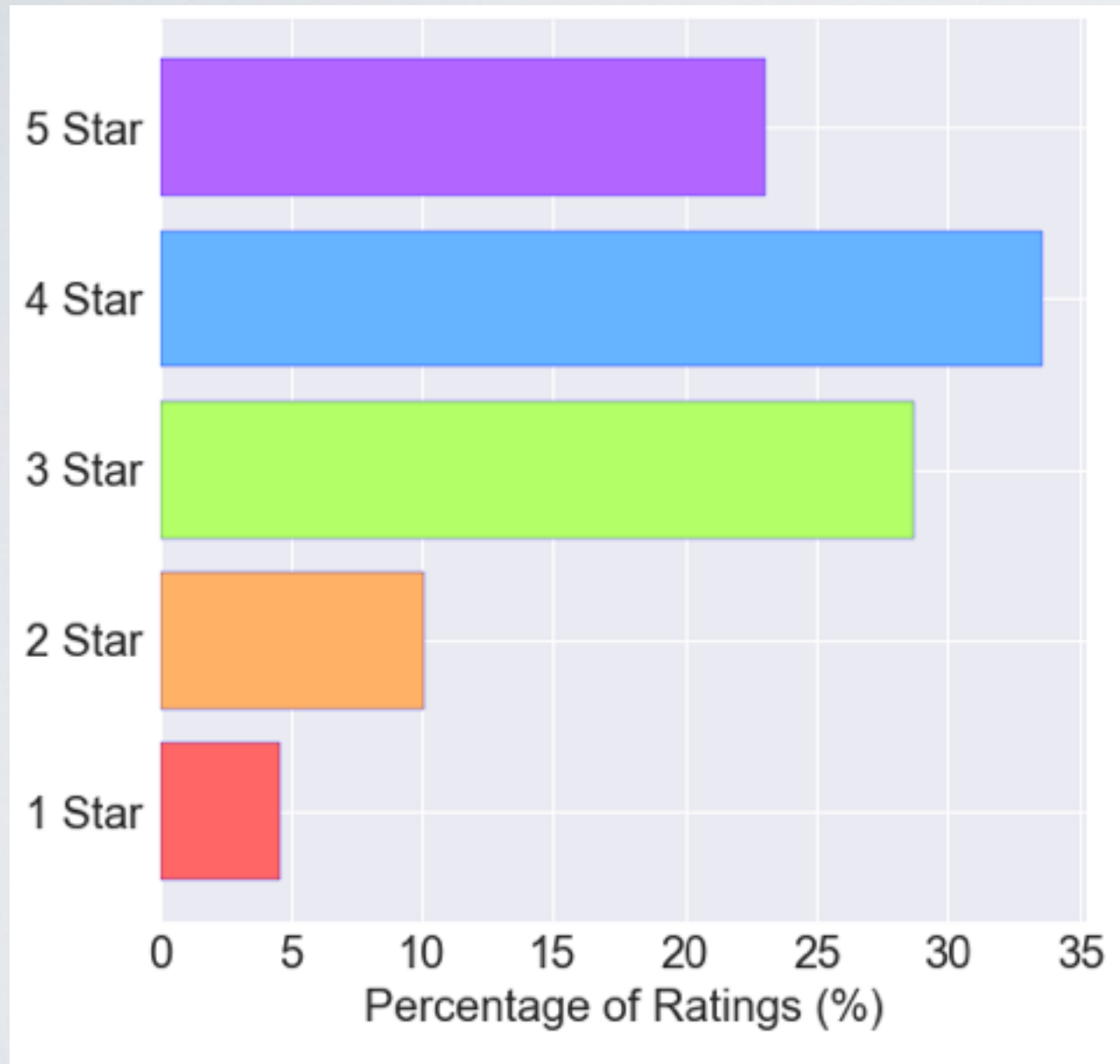
Querying the Open Movie Database (OMDb)

- The OMDb was queried to retrieve the genre of each movie on the dataset. This information was included in the original .csv file containing movie names, as shown here:

	Name	Year	Genres
1	Dinosaur Planet	2003	Documentary, Animation, Family
2	Isle of Man TT 2004 Review	2004	None
3	Character	1997	Crime, Drama, Mystery
4	Paula Abdul's Get Up & Dance	1994	None
5	The Rise and Fall of ECW	2004	None
6	Sick	1997	Short, Drama
7	8 Man	1992	Action, Sci-Fi
8	What the #\$! Do We Know!?	2004	None
9	Class of Nuke 'Em High 2	1991	None
10	Fighter	2001	Action, Drama

Exploratory Data Analysis:

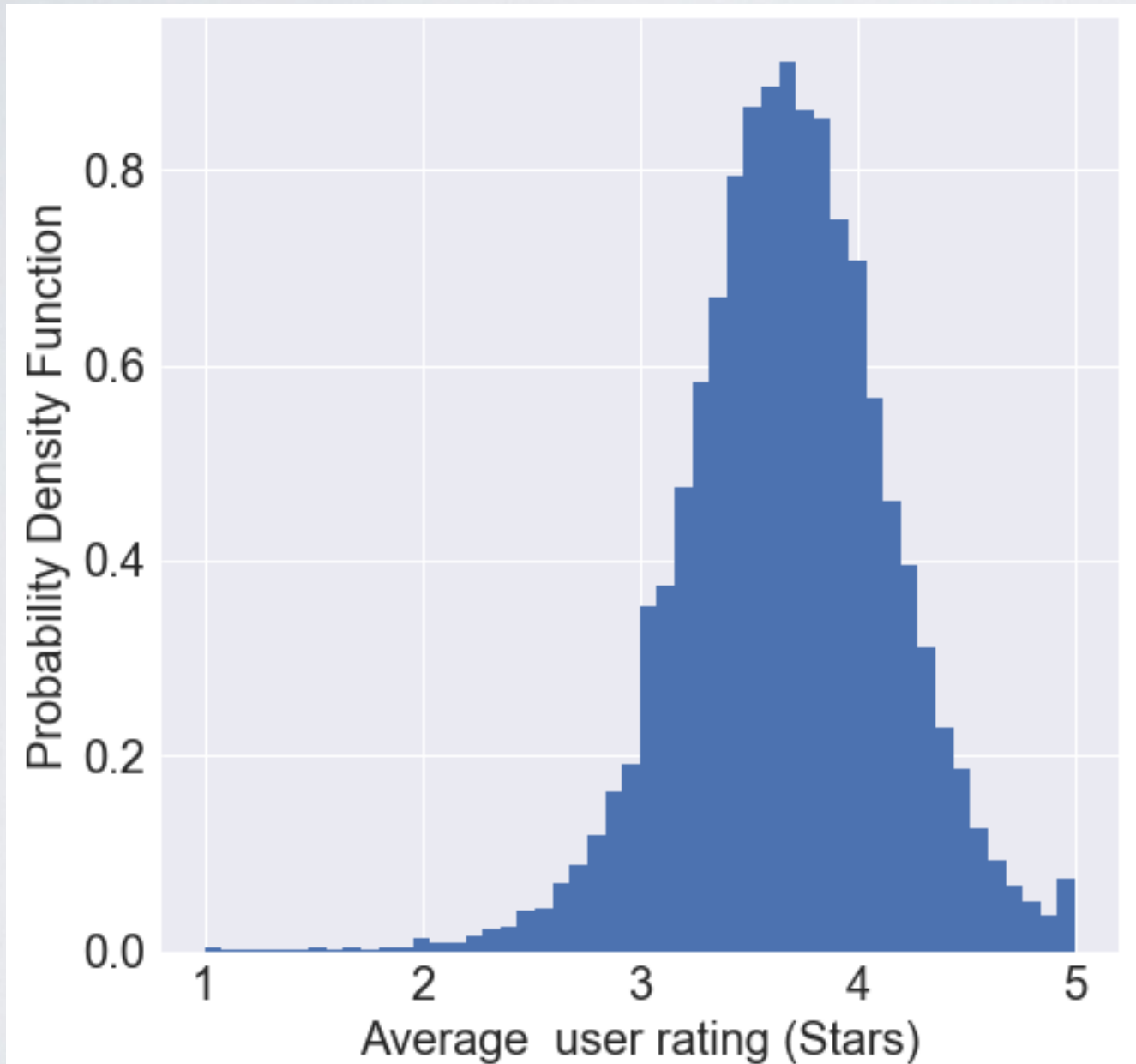
Ratings Frequency Distribution



- Most frequent rating: 4 Stars
- 56% are “good ratings”
(4 or 5 Stars)
- Average rating : ~3.6 Stars
- Standar deviation: 1.09 Stars

Exploratory Data Analysis:

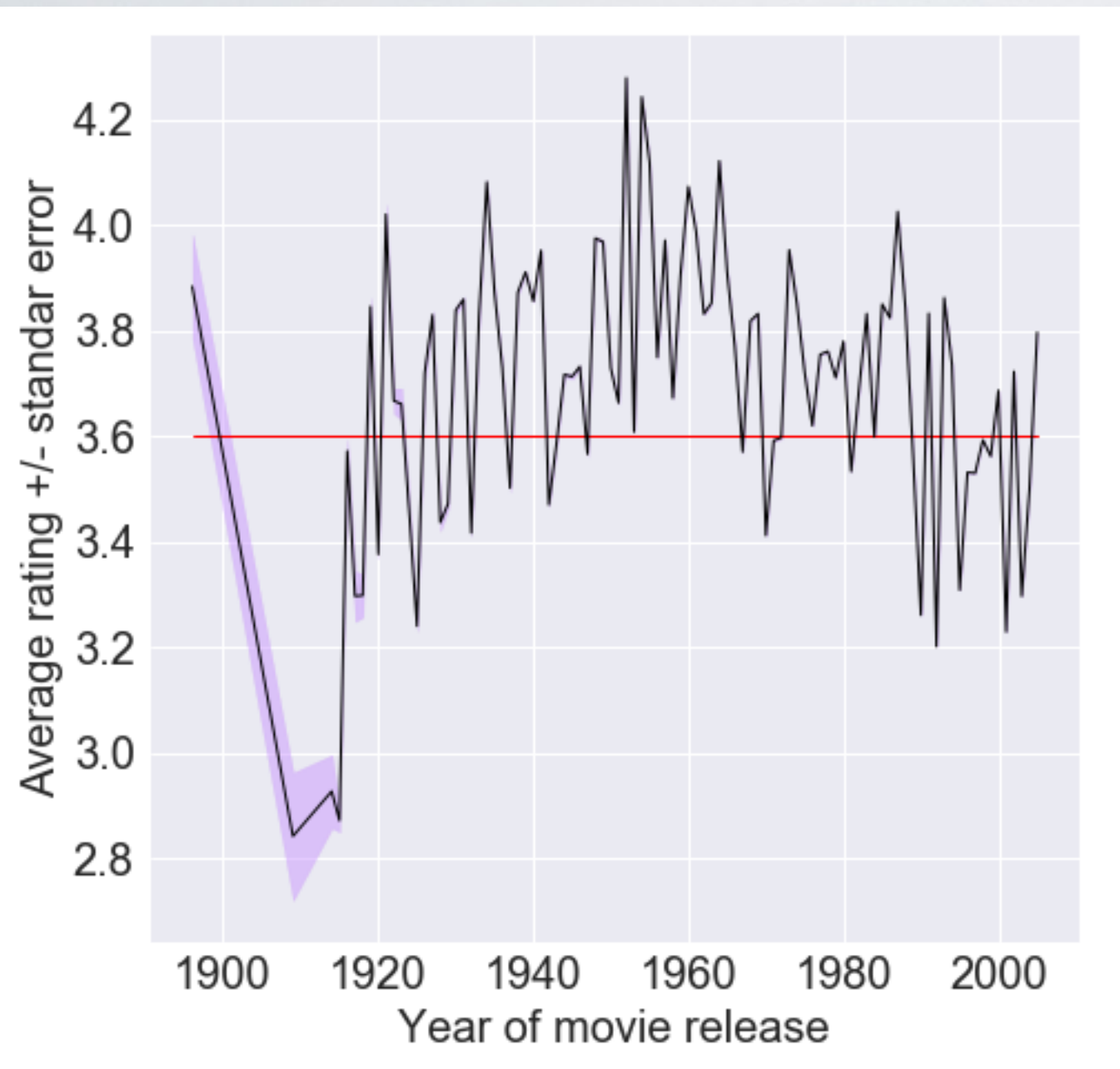
Distribution of users average ratings



- Normally distributed
- ~30% of users have average ratings lower than 3 or higher than 4 Stars.
- Suggests that some users might have a positive or negative rating bias

Exploratory Data Analysis:

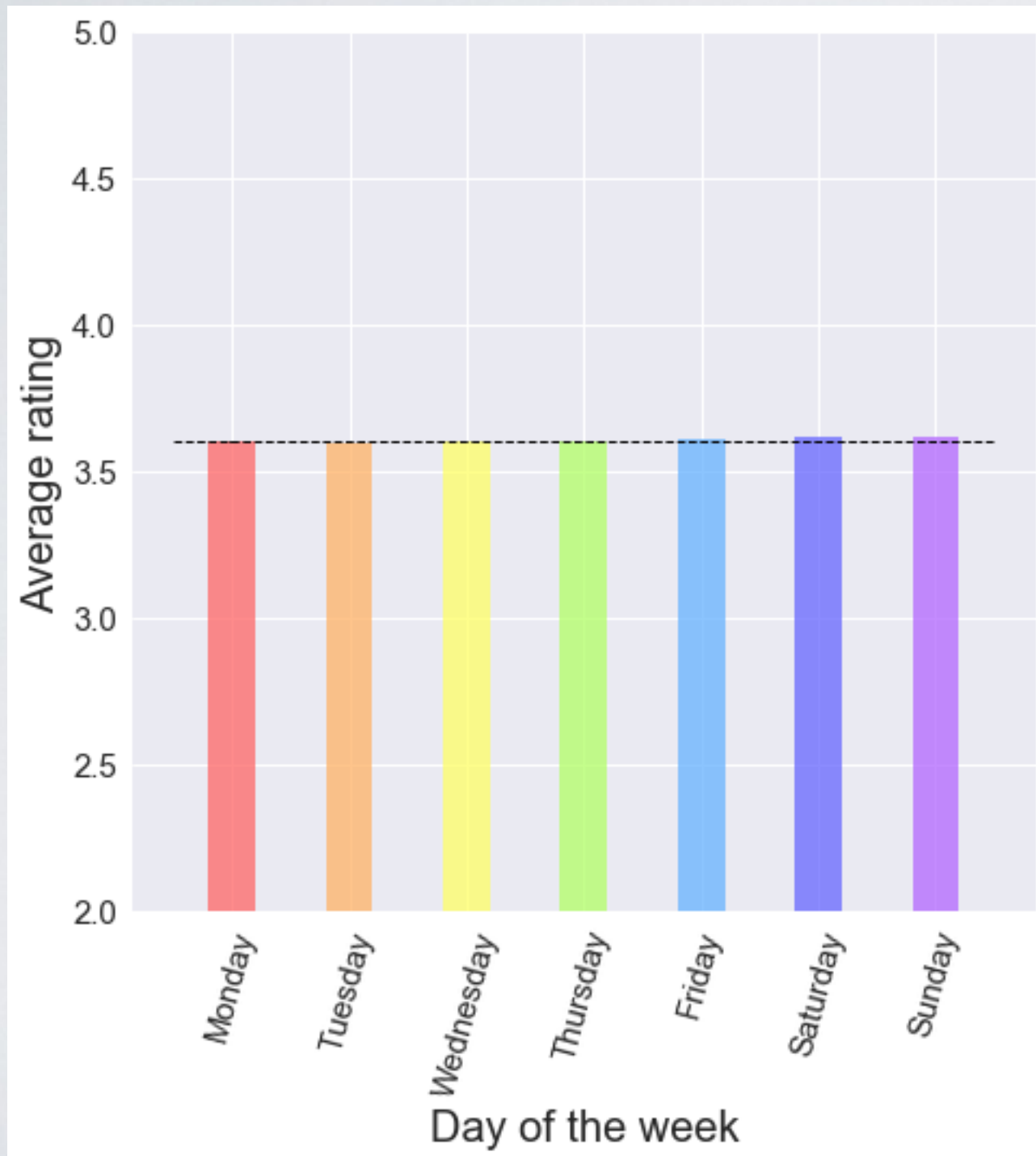
Does the movie release year have an effect on ratings?



- 89 out of 94 years, show significant differences from the overall average (red line)
- Suggest that a movie's release year might significantly affect its rating

Exploratory Data Analysis:

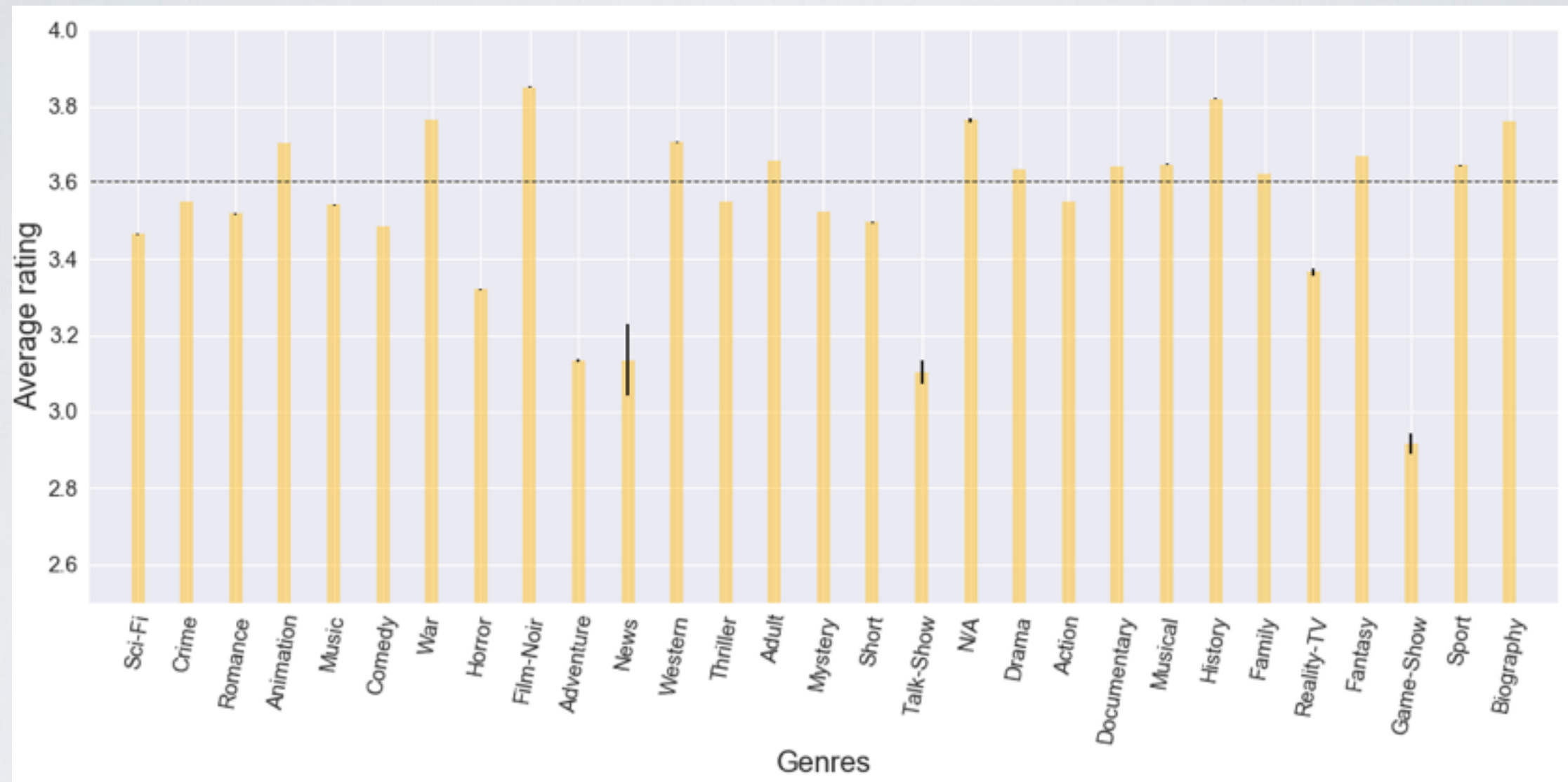
Does the day of the week have an effect of the ratings ?



- Date of rating was converted to day of the week, and the data was grouped accordingly
- No day of the week shows significant differences from the average (dashed line)
- Suggests that the day of the week, does not have an effect on the ratings

Exploratory Data Analysis:

Is the rating of a movie affected by its genre ?



- One-sample t-test showed that all genres differ significantly from the average (dashed line).

Exploratory Data Analysis:

Conclusions

'User specific bias', 'Year of movie release' and 'Movie's Genre' are factors that significantly affect the rating of movies, thus, they can potentially be incorporated into a predictive model for movie preferences and ratings

Predicting movie ratings:

Raw Average

- Rating is predicted as the average of the entire dataset
- In addition, user-, movie-, year-of-release- and genre-specific biases were used to adjusted the predicted ratings
- The root mean squared error (RMSE) was used as a measure of prediction error

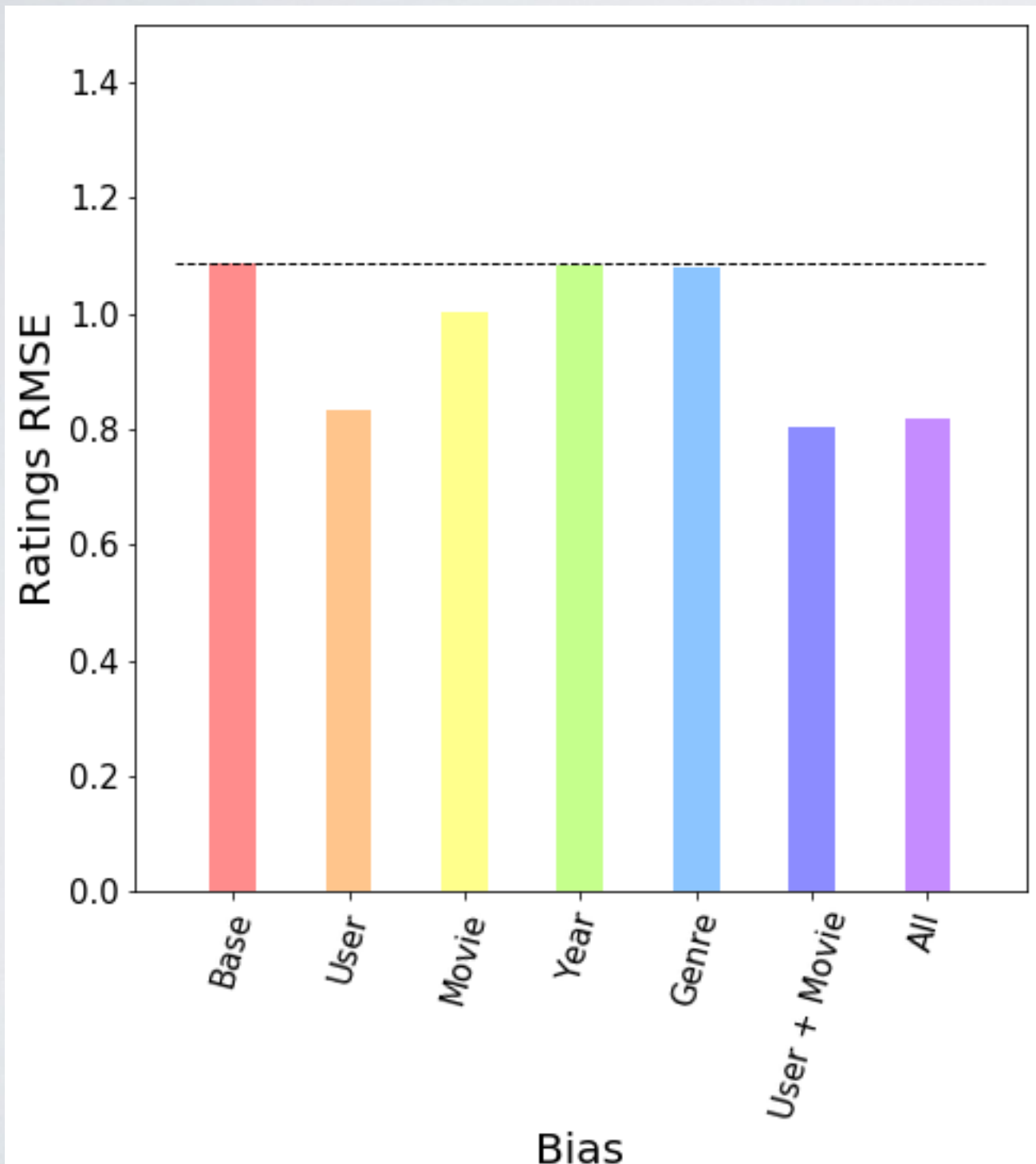
Predicting movie ratings:

Raw Average

******Trying to work with the entire Netflix prize data set on a single machine proved to be too computationally intensive. Instead, I opted to take a random sample of 1 million rating to work with. The following analysis were performed using this sample******


























Predicting movie ratings:

Raw Average Prediction errors



- Incorporating individual biases (user & movie) gives a modest reduction to errors of the predicted ratings
- Combining multiple biases does not provide further improvement

Predicting movie ratings: Colaborative Filtering (CF).

姜春云

- CF methods predict ratings based on the similarity of the items rated by multiple users
- Alternate Least Squares (ALS), is one of such methods, and can be implemented from the pyspark's MLlib

Predicting movie ratings:

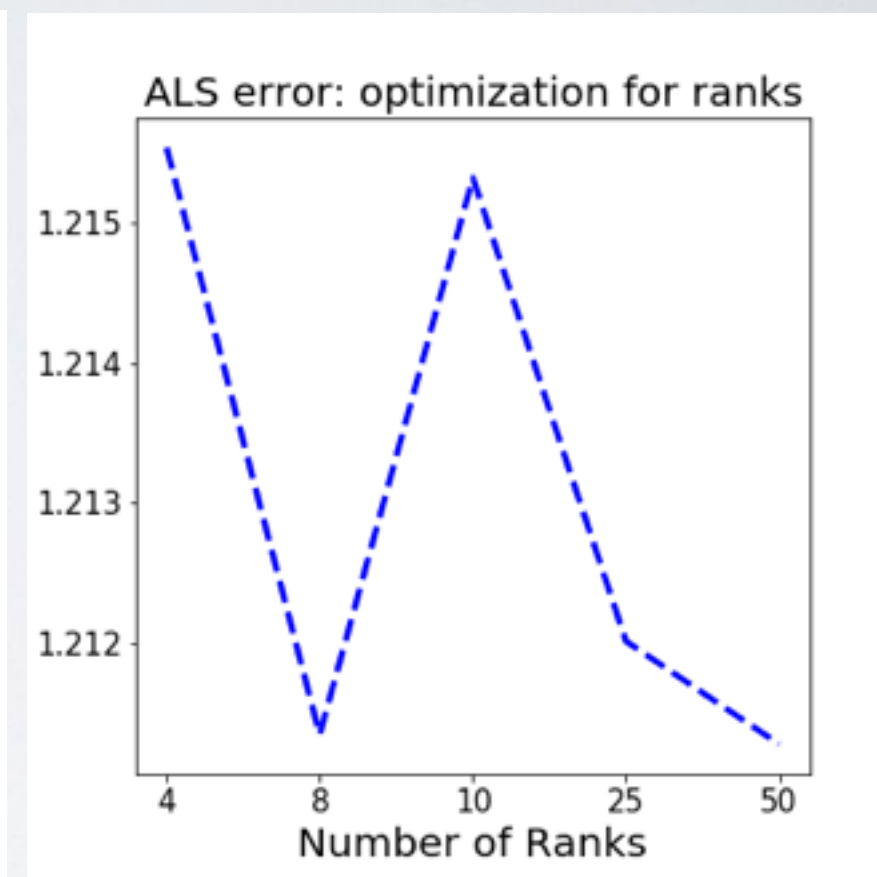
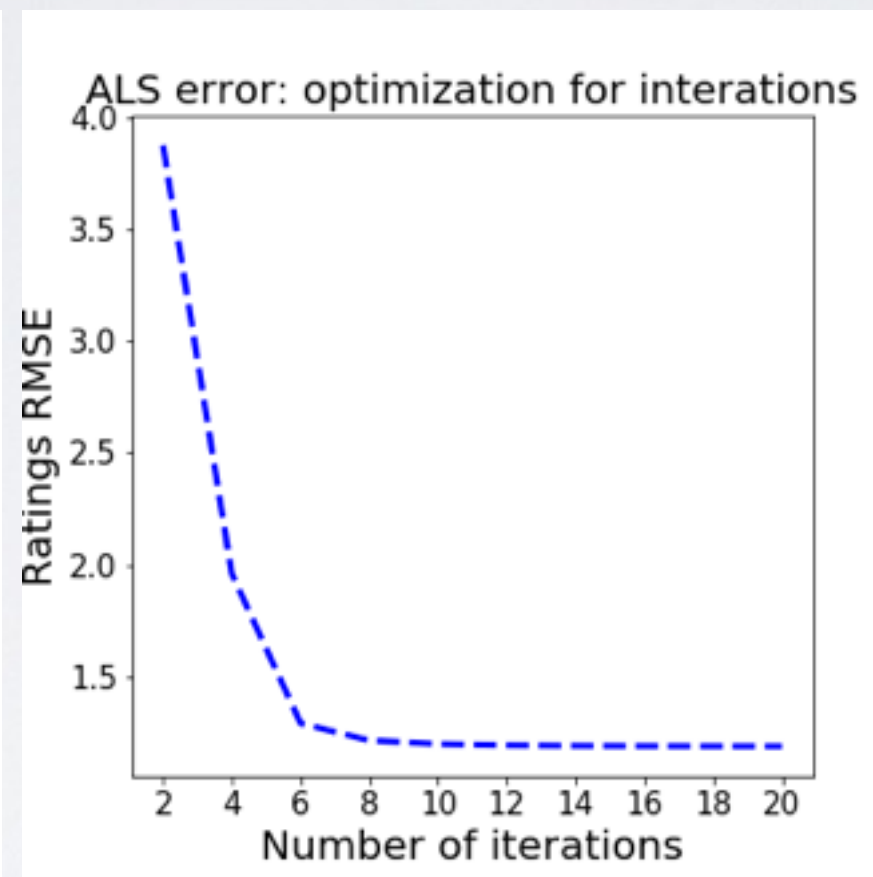
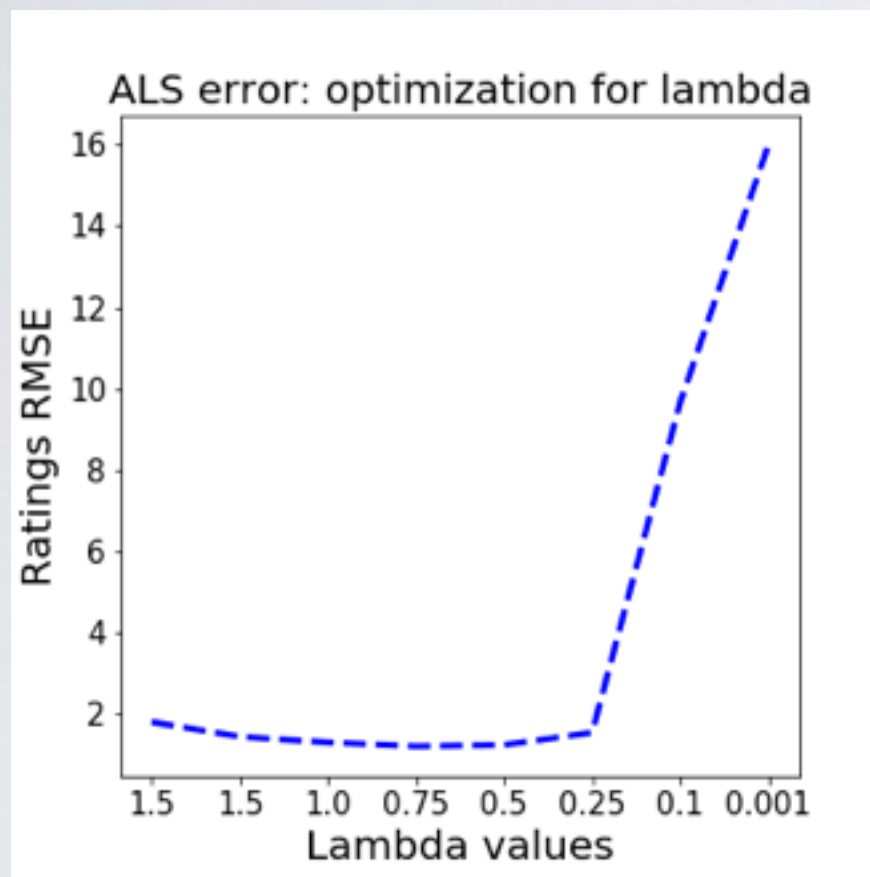
ALS hyperparameters

-) **numBlocks**: is the number of blocks the users and items will be partitioned into in order to parallelize computation (defaults to 10).
-) **rank**: is the number of latent factors in the model (defaults to 10).
-) **maxIter**: is the maximum number of iterations to run (defaults to 10).
-) **regParam**: specifies the regularization parameter in ALS (defaults to 1.0).
-) **implicitPrefs**: specifies whether to use the explicit feedback ALS variant or one adapted for implicit feedback data (defaults to false which means using explicit feedback).
-) **alpha**: is a parameter applicable to the implicit feedback variant of ALS that governs the baseline confidence in preference observations (defaults to 1.0).
-) **nonnegative**: specifies whether or not to use nonnegative constraints for least squares (defaults to false).

Predicting movie ratings:

ALS hyperparameters optimization

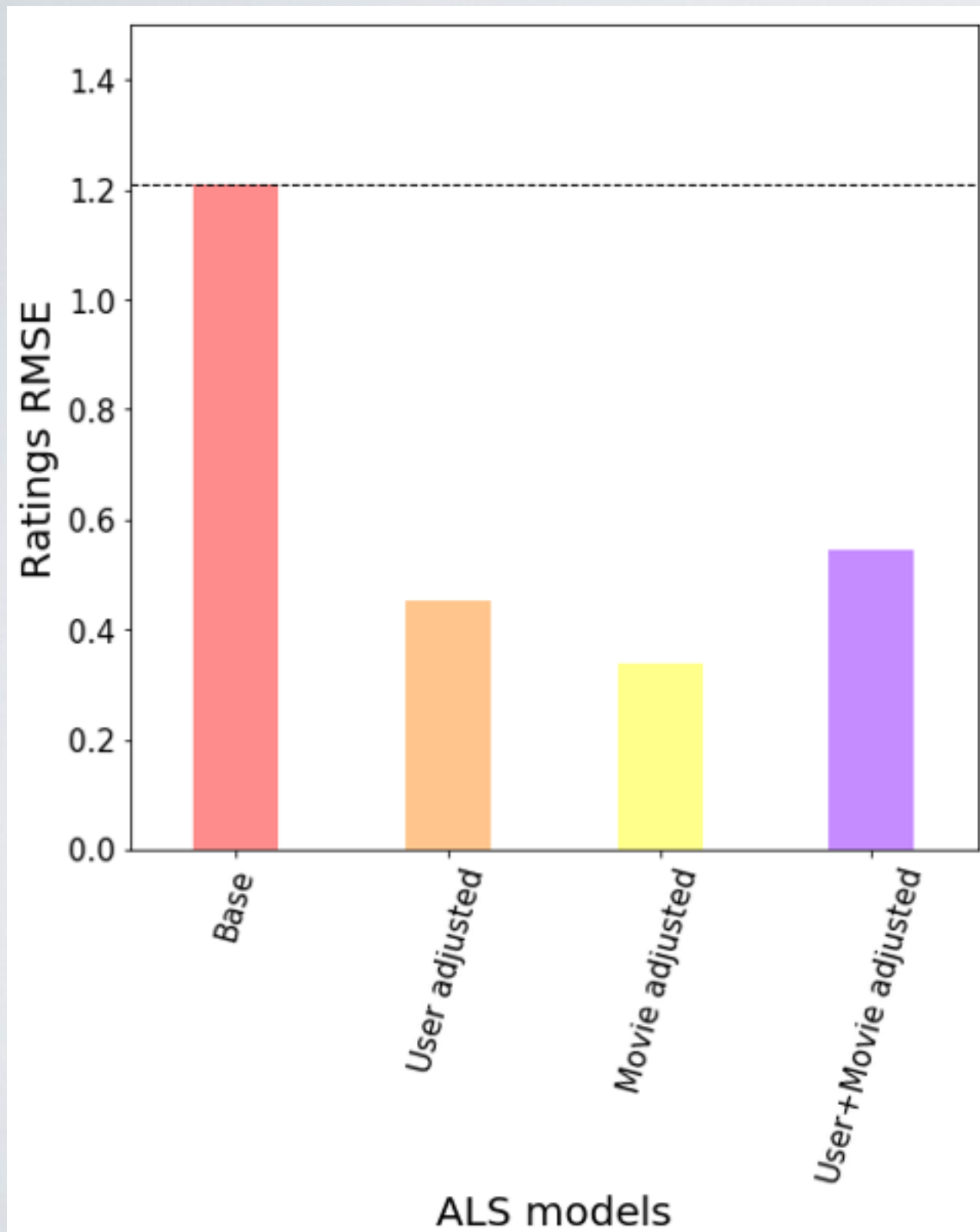
-) Two of the parameters (rank, maxIter, regParam or lambda) were kept constant, while changing the third one. As before, to measure the errors I used RMSE.



- **ALS optimal hyperparameters:**
 - regParam: 0.5
 - iterations: 8
 - ranks: 8

Predicting movie ratings:

optimal ALS model + user & movie biases



- Optimal hyperparameters were used to train an ALS model.
- Predicted ratings are adjusted with user and movie rating biases
- Predicted ratings were also adjusted to the 1 to 5 Stars Netflix Scale
- Incorporating individual biases (user & movie) reduces the errors of the predicted ratings
- Combining multiple biases does not provide further improvement

Using the optimal ALS model to make movie recommendations

We use the optimal ALS model to make movie recommendations for a new user (not in the original dataset):

- We start with a list of movies (shown below) the the user has rated

```
new_ratings = [  
    (0,14941,4),# The Matrix  
    (0,14928,4),# Dead Poets Society  
    (0,5344,5),# Fullmetal Alchemist  
    (0,10463,4),# Pokemon: The First Movie  
    (0,10453,4),# Pokemon Advanced  
    (0,5732,4),# Good Will Hunting  
    (0,15096,5),# The Notebook  
    (0,17132,5),# Waking Life  
    (0,11763,3),# Serendipity  
    (0,178,5) #A Beautiful Mind  
]
```

- We add this ratings to the dataset, and retrain the model
- We recommend movie only if they have a high recommended rating

Using the optimal ALS model to make movie recommendations

For a movie to be recommended, I considered it should have:

- A predicted rating of 4.5 Stars or more
- At least 20 reviews in the dataset used to train the ALS model

A sample of the recommended movies is shown below:

Sample list of recommended movies:

```
Angel: Season 5. Predicted rating: 5 Stars
Spirited Away. Predicted rating: 4.67015305531 Stars
Batman Begins. Predicted rating: 4.80449606347 Stars
I. Predicted rating: 4.69739734497 Stars
Buffy the Vampire Slayer: Season 4. Predicted rating: 5 Stars
The Shield: Season 3. Predicted rating: 5 Stars
Home Improvement: Season 1. Predicted rating: 4.69458404747 Stars
Dark Angel: Season 1. Predicted rating: 4.79938823529 Stars
Stargate SG-1: Season 4. Predicted rating: 4.76801740771 Stars
Million Dollar Baby. Predicted rating: 4.64580944925 Stars
Freaks & Geeks: The Complete Series. Predicted rating: 5 Stars
Sex and the City: Season 1. Predicted rating: 4.74706793216 Stars
Absolutely Fabulous: Series 1. Predicted rating: 4.64477690002 Stars
Star Trek: The Next Generation: Season 6. Predicted rating: 4.94167663198 Stars
Curb Your Enthusiasm: Season 4. Predicted rating: 5 Stars
CSI: Season 3. Predicted rating: 5 Stars
Braveheart. Predicted rating: 4.77114929443 Stars
Finding Nemo (Widescreen). Predicted rating: 5 Stars
Six Feet Under: Season 1. Predicted rating: 4.91393002626 Stars
Life Is Beautiful. Predicted rating: 4.65389538027 Stars
```

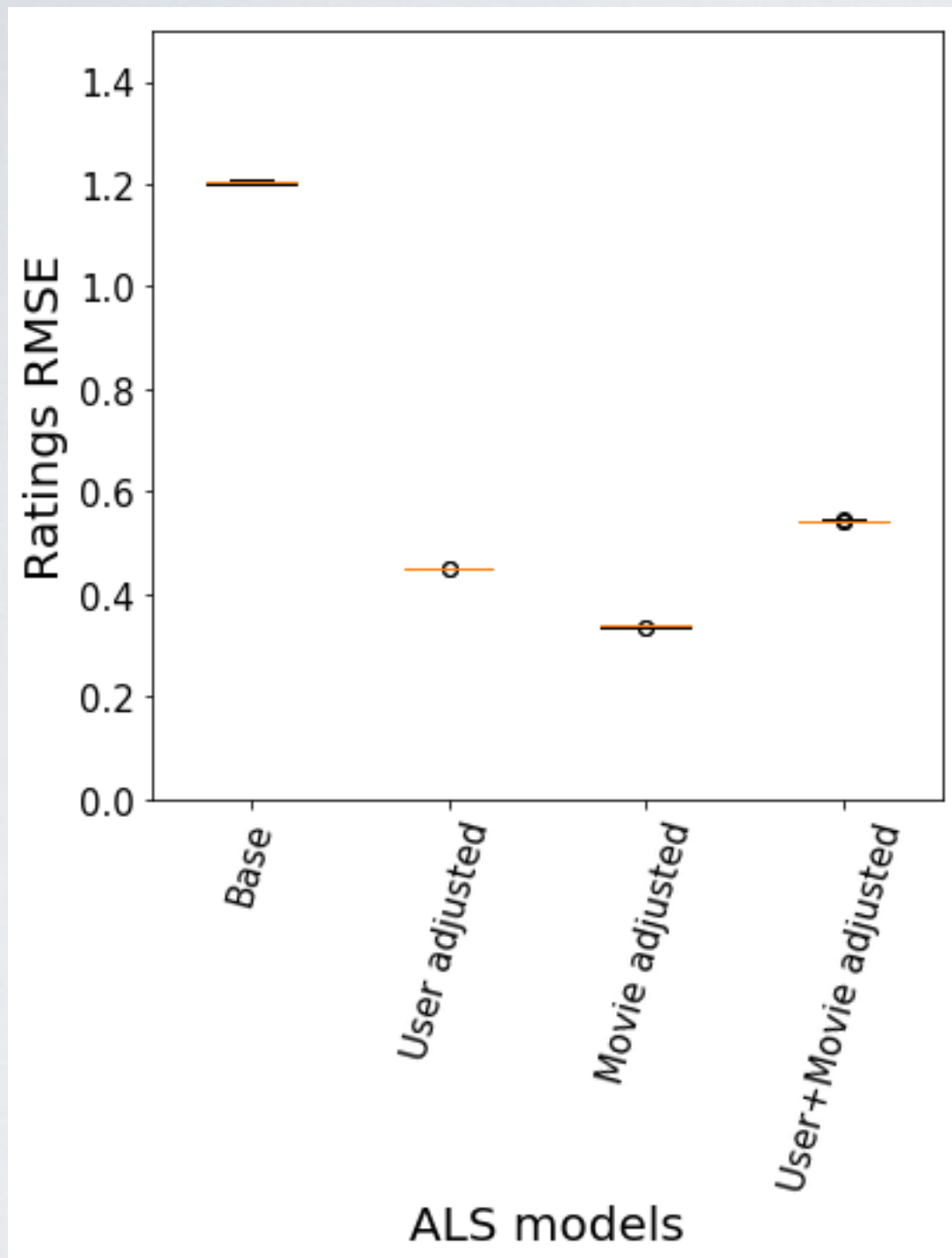
Total number of recommended movies: 419

Using the optimal ALS model to make movie recommendations

The sample of recommended movie has some consistency with the initial set of new ratings:

- If you liked "*The Matrix*" you might also like "Batman Begins" or "Star Gate SG-1"
- If you liked "Full Metal Alchemist" or "Pokemon", you might also like "Spirited Away"
- The only recommended movie/series in the sample list that seems a bit out of place, might be "Sex and the City".

ALS model validation

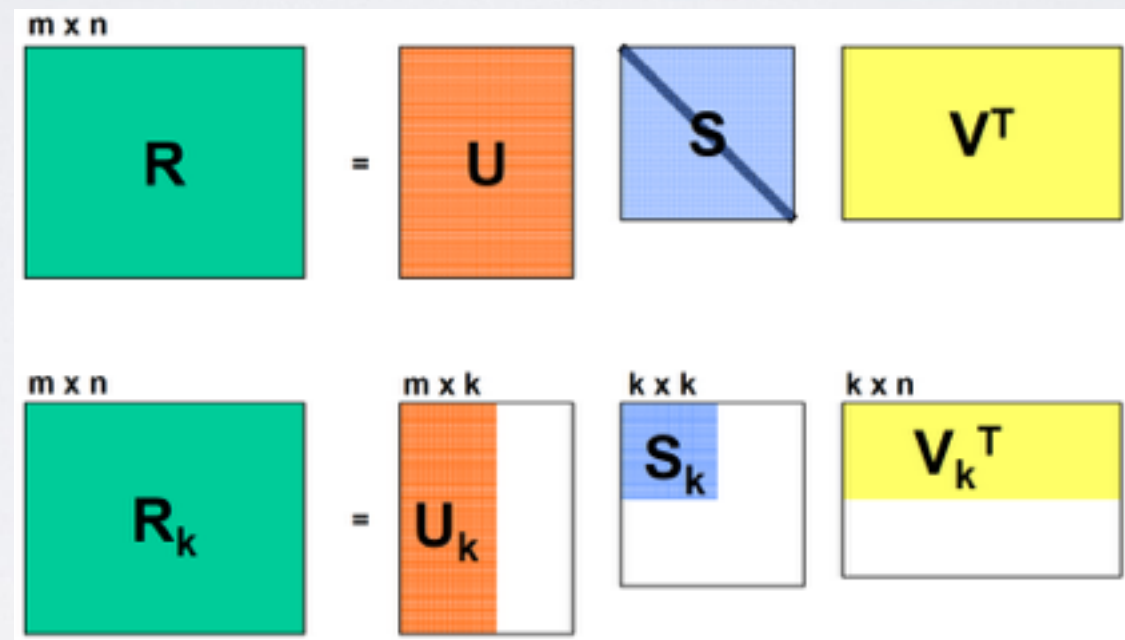


- A 10-fold cross validation, shows consistency of prediction error for different samples of the original dataset.
- Two of the samples produced recommendations with high degree of similarity (~80%)

Singular Value Decomposition (SVD) for recommender systems

A sparse ratings matrix R can be factorized as the product of three matrices U , S and V , such that:

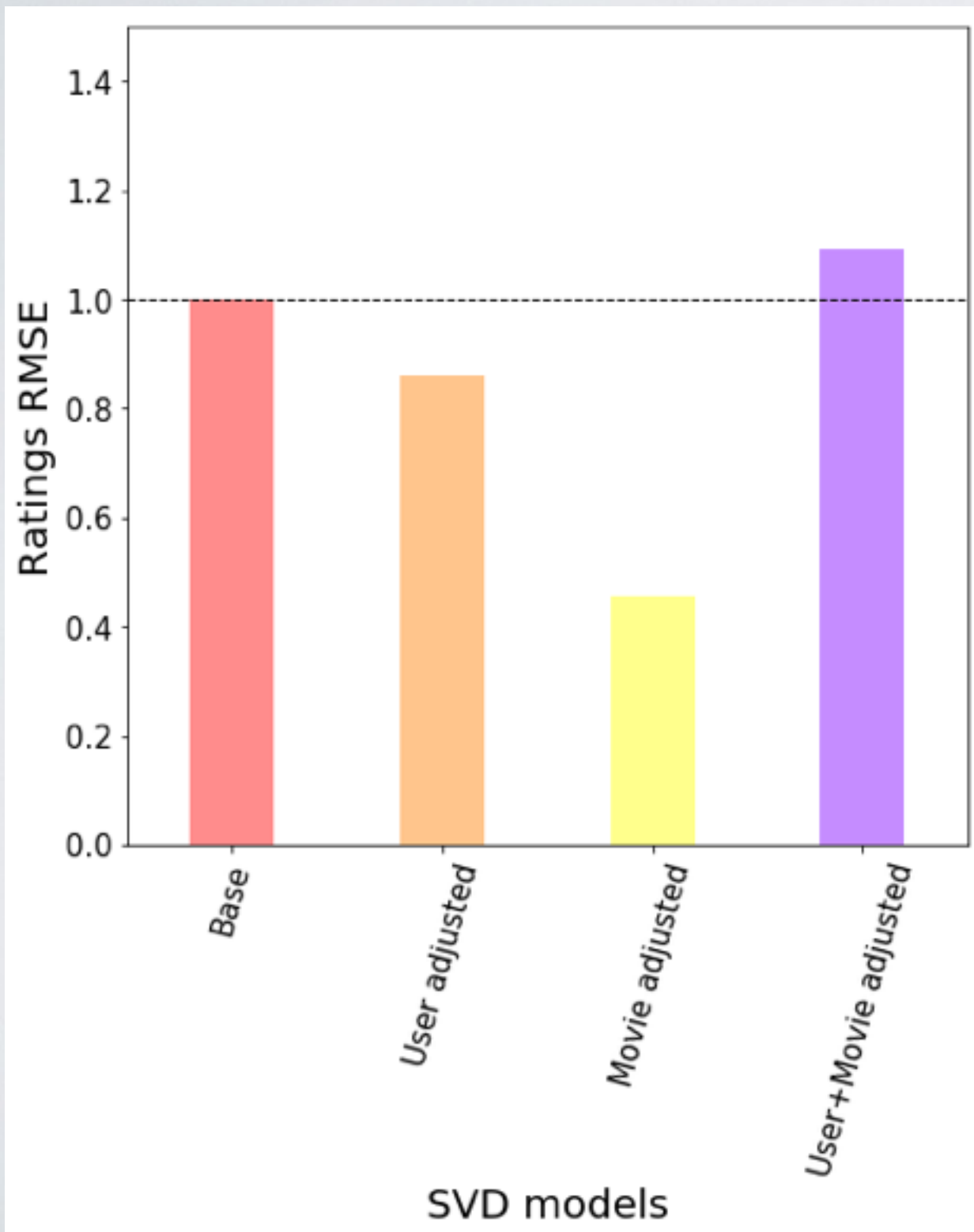
$$R = USV^T$$



A lower rank approximation of R can be obtained by taking the top “ k ” features (rows) of U and V . This approximation of R is then used to make movie recommendations.

A SVD based recommender can easily be implemented using ‘Suprise’, a python sci-kit for recommender systems

SVD recommender prediction error (RMSE)



- Predicted ratings are adjusted with user and movie rating biases
- Predicted ratings were also adjusted to the 1 to 5 Stars Netflix Scale
- Incorporating individual biases (user or movie) improves the prediction error
- Combining multiple biases (user + movie) worsens the prediction error

SVD recommender prediction error (RMSE)

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.9974	0.9962	1.0004	0.9994	0.9993	0.9985	0.0015

A 5-fold cross-validation shows that the SVD based recommender produces stable prediction errors for different folds of the sample data set.

Using the SVD model to make movie recommendations

Similarly to the ALS model, we will use the SVD model to make movie recommendations for a new user (not in the original dataset):

- We start with a list of movies (shown below) the the user has rated

```
new_ratings = [  
    (0,14941,4),# The Matrix  
    (0,14928,4),# Dead Poets Society  
    (0,5344,5),# Fullmetal Alchemist  
    (0,10463,4),# Pokemon: The First Movie  
    (0,10453,4),# Pokemon Advanced  
    (0,5732,4),# Good Will Hunting  
    (0,15096,5),# The Notebook  
    (0,17132,5),# Waking Life  
    (0,11763,3),# Serendipity  
    (0,178,5) #A Beautiful Mind  
]
```

- We add this ratings to the dataset, and retrain the model
- We recommend movie only if they have a high predicted rating

SVD model: List of recommended movies

For a movie to be recommended, I considered it should have:

- A high predicted ratings
- At least 20 reviews in the dataset used to train the SVD model

A sample of the recommended movies is shown below:

```
Recomended movies for new user:

Dead Like Me: Season 2. Predicted rating: 5 Stars
Lost: Season 1. Predicted rating: 5 Stars
Lord of the Rings: The Return of the King: Extended Edition. Predicted rating: 5 Stars
The Lord of the Rings: The Fellowship of the Ring: Extended Edition. Predicted rating: 5 Stars
Seinfeld: Season 3. Predicted rating: 5 Stars
Lord of the Rings: The Two Towers: Extended Edition. Predicted rating: 5 Stars
Angel: Season 5. Predicted rating: 5 Stars
The Simpsons: Season 6. Predicted rating: 5 Stars
The Shawshank Redemption: Special Edition. Predicted rating: 5 Stars
Firefly. Predicted rating: 5 Stars
Cinderella: Special Edition. Predicted rating: 5 Stars
Family Guy: Vol. 1: Seasons 1-2. Predicted rating: 5 Stars
Nip/Tuck: Season 2. Predicted rating: 5 Stars
Raiders of the Lost Ark. Predicted rating: 5 Stars
Anne of Green Gables. Predicted rating: 5 Stars
Gladiator: Extended Edition. Predicted rating: 5 Stars
The Twilight Zone: Vol. 42. Predicted rating: 5 Stars
Family Guy: Vol. 2: Season 3. Predicted rating: 5 Stars
The West Wing: Season 3. Predicted rating: 5 Stars
Dragon Ball Z: Frieza. Predicted rating: 5 Stars
Gilmore Girls: Season 4. Predicted rating: 5 Stars
The Rise and Fall of ECW. Predicted rating: 5 Stars
Kyo Kara Maoh: God Save Our King. Predicted rating: 5 Stars
Star Wars: Episode V: The Empire Strikes Back. Predicted rating: 5 Stars
The Simpsons: Season 4. Predicted rating: 5 Stars
```

We see similar recommendations (e.g “Lord of the Rings”, “Gladiator”, “The Simpsons”, “Family guy”) to the ALS based recommender

Conclusion and Future Directions

- Ratings prediction errors greatly diminished if a user-specific or movie-specific bias is considered. Therefore, it is suggested that future recommender systems would benefit from incorporating this type of biases in ratings.
- Better, more accurate, recommendations might be achieved when training the model with the complete dataset, or when more ratings are available for the new user
- Better predictions can also be achieve by combining multiple predictive algorithms.

References

"Collaborative Filtering - RDD-based API",

Link: <https://spark.apache.org/docs/2.2.0/mllib-collaborative-filtering.html>

"An on-line movie recommending service using Spark",

Link: <https://github.com/jadianes/spark-movie-lens/blob/master/notebooks/building-recommender.ipynb>

"Surprise: A python scikit for recommender systems",

Link: <http://surpriselib.com/>

"NetFlix Prize",

Link: https://en.wikipedia.org/wiki/Netflix_Prize

"Winning the Netflix Prize: A Summary",

Link: <http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/>

