Recommender System

Haiyue Yang, Yun Shen, Xin Peng

Most platforms we used nowadays, for example, Youtube, Amazon, Netflix, are supported by recommender systems. And it helps personalize the recommendations based on users' interest and browsing history, purchasing history. Netflix even has provided a Netflix Prize to honour teams that can make improvements on predicting users' ratings for movies.

And in this report, we used MovieLens data sets colleted by the GroupLens Research Project. And it contains 100,000 ratings from 943 users on 1682 movies.

And we have explored the data, brainstormed what factors could help improve the recommender system, proposed several models, and implemented three variants of PMF below.

Add-on Value of the Data

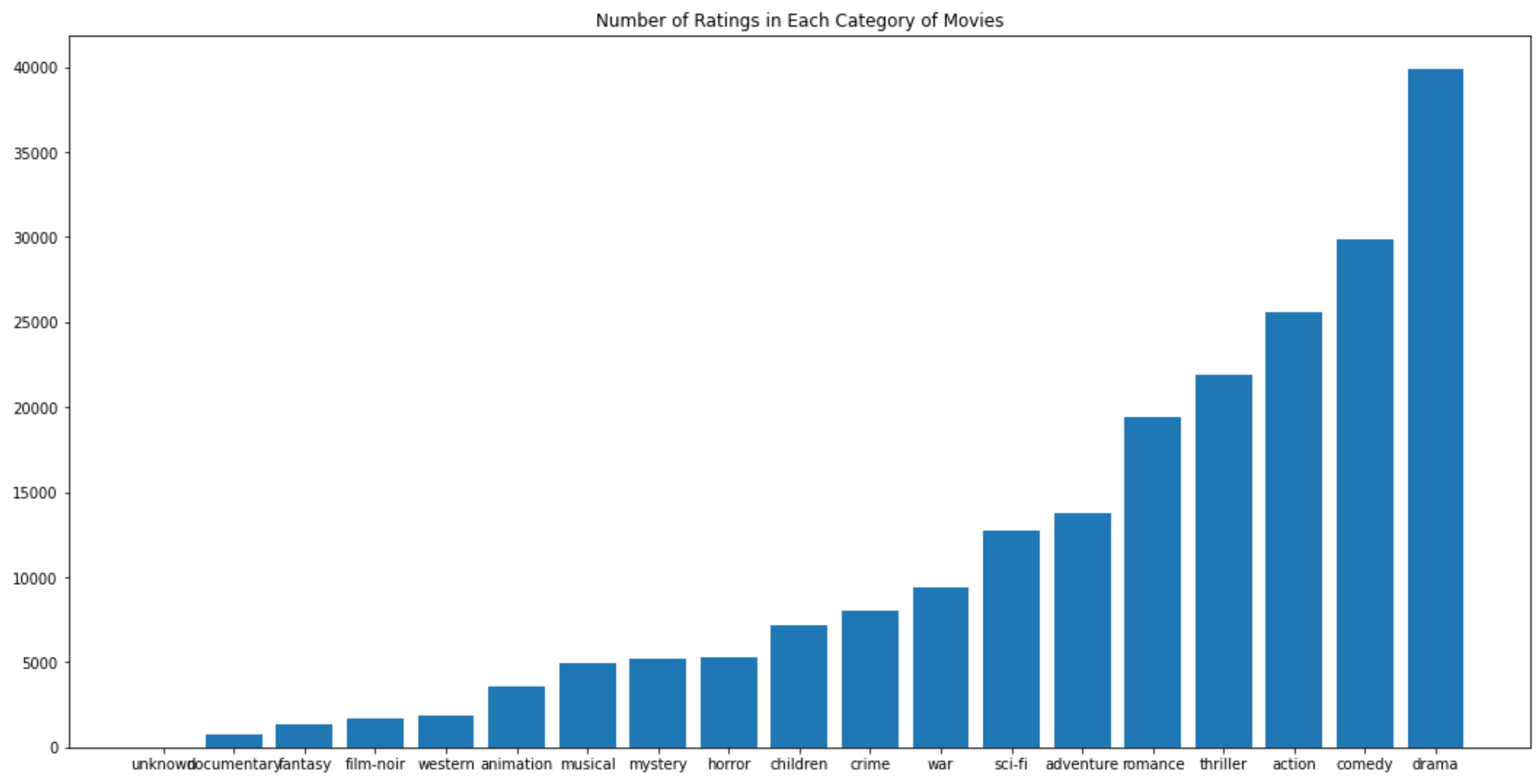
- Make recommendations for groups with similar interest:
  - Imagine your view history and interest are similar to other people, you can be identified as a group and some group actions can be made to the whole group.
- Understand what's trending over time for investment:
  - Companies like Netflix will utilize this to decide what to produce for the next season.
- Suggest friends based on mutual interests: This can be implemented in social networks platforms.
- Show ads to target customers based on their view history

Key Factors to Measure Success

- the number of daily active users:
  - A streaming service with a sophisticated recommender system will attract users to stay active because they will be watching the recommended movies.
- the increase in the number of views for recommended movies:
  - If the movies are recommended to the correct groups of people, there should be more people watching it.
- the rating of recommended movies by target users:
  - If the user likes the recommended movie, he tends to give a high rating.
- “not interested” reaction
  - One example is the like/dislike button on youtube.
  - If the user doesn’t like the recommended movie, it indicates that the recommender system goes wrong sometimes.

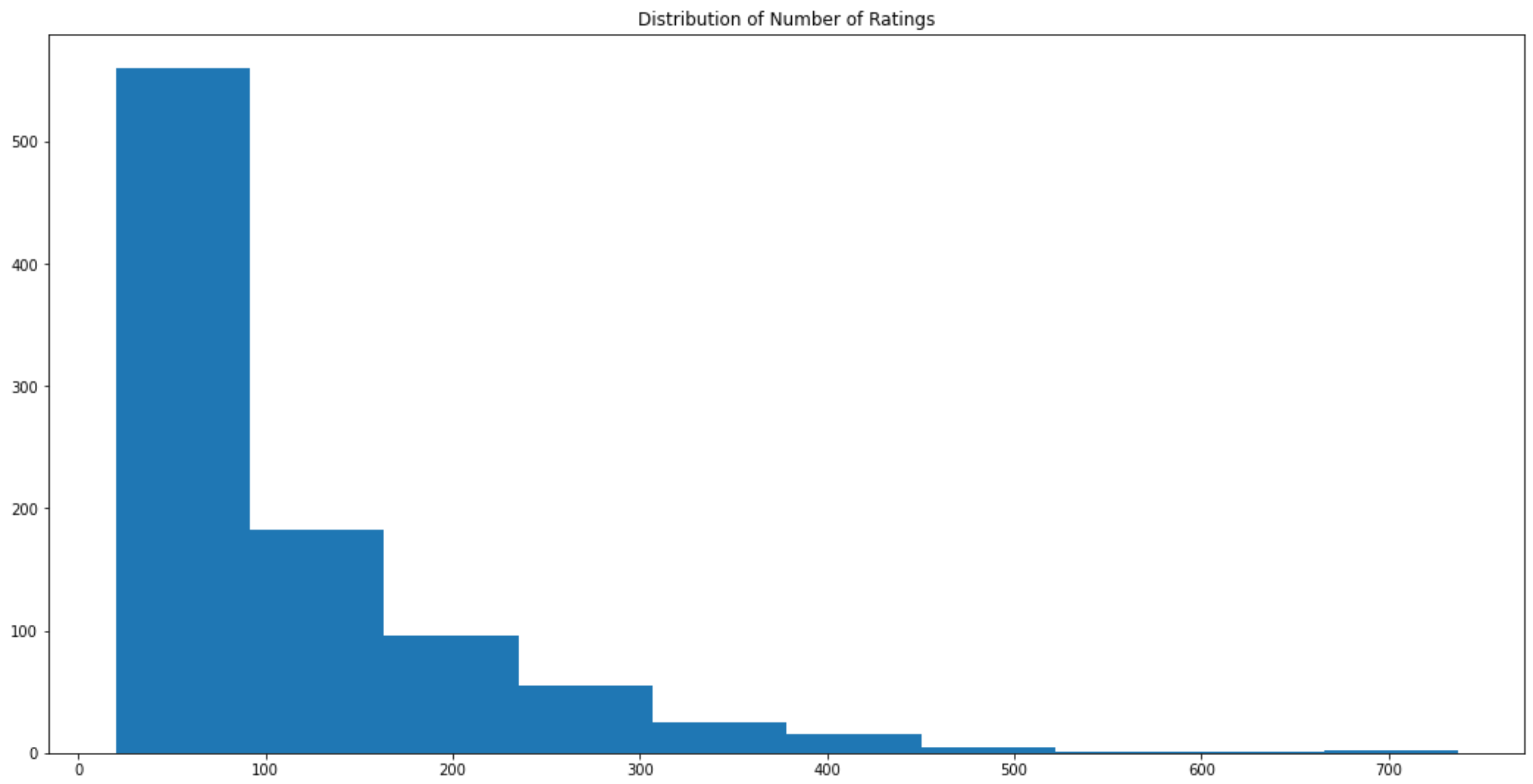
Data Exploration

Problems in the Data



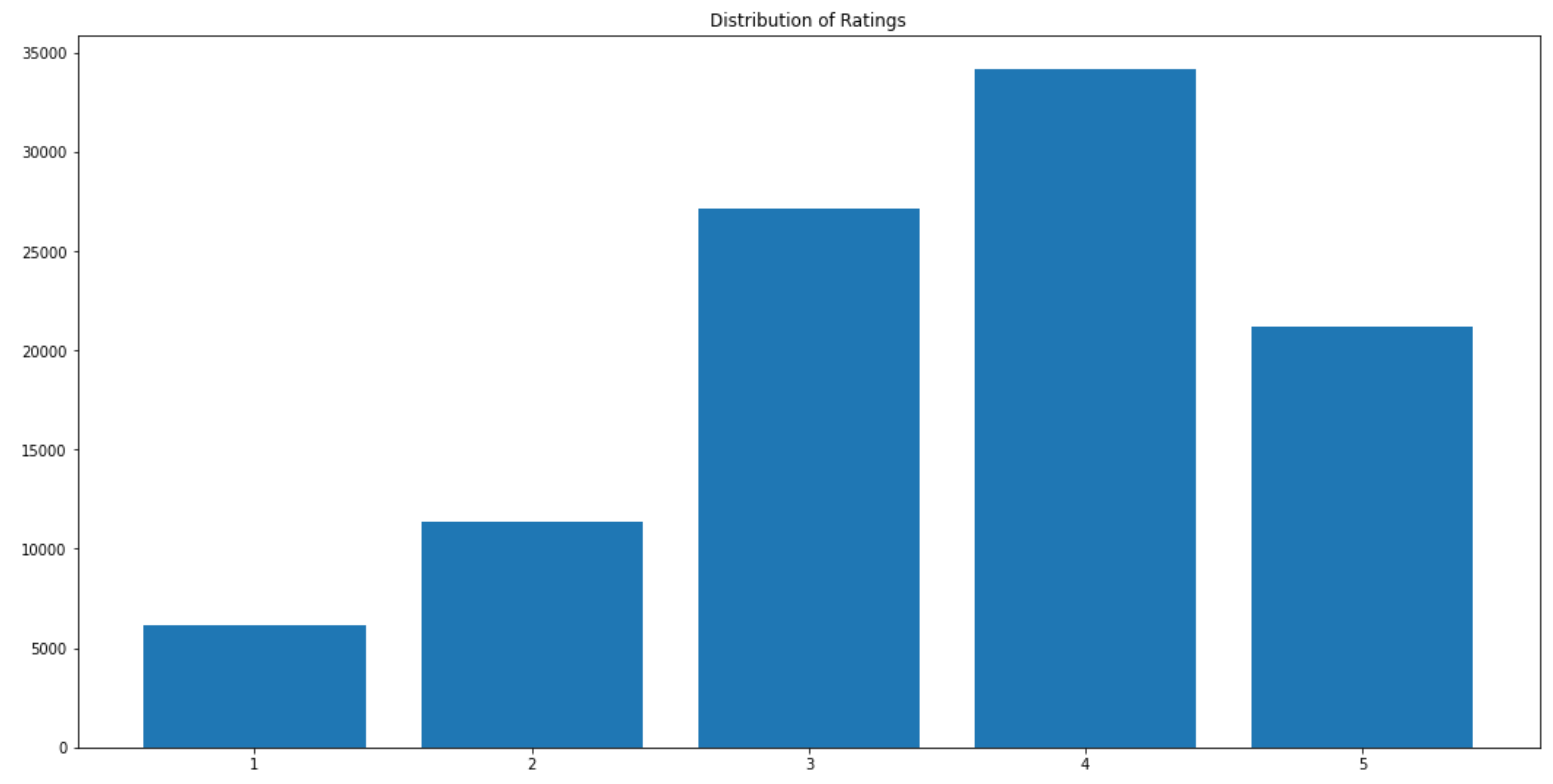
- The number of ratings in each category varies a lot.
- Some categories like action, comedy, drama, and thriller movies have more than 20 thousand ratings in total, but other categories like documentary movies have only fewer than 1000 ratings.
- 10 ratings of movie do not have a known category.
- predictions based on the categorical information about the movies in unpopular categories or even in an unknown category might not be reliable.

Distribution of Number of Ratings



- Number of ratings of each user ranges from 20 to approximately 800.
- Ghost users with fewer than 20 ratings were already deleted.
- The distribution of unimodal and strongly skewed to the right
- More than half of users having fewer than 100 ratings and few users having more than 300 ratings.

Distribution of Ratings



- The distribution of ratings is uni-model and skewed to the left, with a peak at a rating of 4.
- Most of the ratings have a score of 4 or 3, a few ratings have a score of 5, and only a small proportion of ratings have a score of 1 or 2.
- Fake users with all-5 or all all-1 ratings were already deleted.

Things Depend on Time

- Movies are merging.
- Users' preference change with time.
- Users' rating standard on movies change with time.
- Hot topics and movie trend change over time.

Complementary Sources of Data

The data we used here only contain title, release date, video release date, IMDB URL, genres about the movies, but additional information about movies like

- movie contents
- cast
- awards
- professional ratings
- popularity
- language

can be very useful in helping make better recommendation.

Similarly, since our data only has age, gender, occupation, zip code about users, some complementary sources can be users' favourite directors, living country, spoken language, etc.

In addition, users' search history and browsing history may also be useful in showing what they are interested in.

List of Factors that Could Affect the Recorded Data

Before building up a model, it is important to notice that rating in reality is very complicated, and there are many factors that could conceivably influence people's rating. Here are some of these factors:

- The overall rating and professional rating of the movie.
- Preference over some actor/actress in a movie.
- People's rating habits are different. Some give an overall high rating; some people tend to give low ratings.
- Popularity of the movie.
- Availability of the movie. People have easy access to some movies, while others are not.
- Social influence and reputation of the movies.
- Personal relevance to the movie.
- Culture similarity with the movie.
- The user's account being hacked.
- Some users are bots, and give fake ratings.
- Standard of 'high quality movies' has changed over time. For example, the IMAX-3D techonology in 2009 is very rare and new, movies like Avatar using IMAX-3D techonology received high ratings. But people are no longer feel so impressed by IMAX-3D techonology in 2021.
- Viewing condition of the user. Viewing a movie in computer and in cinema will have different visual effect and sound effect.
- Users' mood when watching a movie.
- Movie's relevance to the hot topics. For example, movie Flu shot in 2013 received higher ratings in 2020 than it was in 2013, since it is closely related to COVID-19.

Model Staircase

Here are some models we proposed.

- Use mean rating of the movies rated by each user assuming each user will give same rating to all movies.
- Use KNN to predict movie rating by finding the Nearest Neighbours that the user has rated.
- Use different user and movie latents for female and male.
- Use movie side info instead of movie latents. (This does not predict well, and the final testing rmse is around 0.78. So only use movie side info, i.e., 19 genres is not enough here.)
- Add temporal features to the PMF. (Variant 1: Temporal)
- Use side info in addition to latents. (Variant 2: Side Info)
- Prediction function is a non-linear matrix factorization. (Variant 3: Non-linear)

Baseline Model (PMF)

The baseline model provided for us uses PMF.

$$r_{ui} = P_u Q_i^T,$$

where r_{ui} is the rating of User u to Item i, and P is user latent factor matrix with dimension $n \times k$, Q is item latent factor matrix with dimension $m \times k$.

And we will implement three variants of PMF below.

```
0 1.2531515 0.9336393 0.9329738
100 0.8892034 0.7436105 0.7620929
200 0.86772126 0.7312353 0.75058323
300 0.8594228 0.7260171 0.74692917
400 0.85295147 0.7220692 0.7453697
500 0.84596014 0.71796274 0.7446068
600 0.8390361 0.71379507 0.74394524
700 0.8327416 0.70990884 0.74290997
800 0.82677925 0.70623773 0.74132067
900 0.82080483 0.7026735 0.7392549
1000 0.81487006 0.6991699 0.73682266
1100 0.8094192 0.6959069 0.7344423
1200 0.8048929 0.6931623 0.7324936
1300 0.8014111 0.69096863 0.73103917
1400 0.79883486 0.68928134 0.7299567
1500 0.79694986 0.68798125 0.7291591
1600 0.79556614 0.6869843 0.72858745
1700 0.79454064 0.6862012 0.7281751
1800 0.7937722 0.6855745 0.7278624
1900 0.7931895 0.6850708 0.7276199
2000 0.7927428 0.68466854 0.7274494
2100 0.7923968 0.6843394 0.7273101
2200 0.79212606 0.68406534 0.7272029
2300 0.79191214 0.68383664 0.72711736
2400 0.79174185 0.68364316 0.7270539
2500 0.7916051 0.6834785 0.72700113
2600 0.79149455 0.68333596 0.7269561
2700 0.7914044 0.6832126 0.7269146
2800 0.79133064 0.6831051 0.7268785
2900 0.7912697 0.68301237 0.72684914
3000 0.79121935 0.6829307 0.7268226
3100 0.7911774 0.6828592 0.72679764
3200 0.7911421 0.6827964 0.7267761
3300 0.7911126 0.68274146 0.7267567
3400 0.7910876 0.6826923 0.7267384
3500 0.7910666 0.68264836 0.7267213
3600 0.7910485 0.6826093 0.7267054
3700 0.7910333 0.68257433 0.72669077
3800 0.7910203 0.6825431 0.72667724
3900 0.79100895 0.68251514 0.72666544
4000 0.79099935 0.6824901 0.7266547
4100 0.79099107 0.68246764 0.7266452
4200 0.7909839 0.6824473 0.7266365
4300 0.7909777 0.682429 0.7266289
4400 0.79097235 0.6824125 0.7266217
4500 0.79096764 0.68239754 0.7266151
4600 0.7909636 0.68238384 0.7266089
4700 0.79096 0.6823713 0.7266032
4800 0.79095703 0.68236005 0.7265978
4900 0.79095423 0.6823497 0.7265929
```

Variant 1: Adding Temporal Features to PMF

In reality, user preference and movie popularity change over time. Therefore, we should account for the temporal effects reflecting the dynamic nature of user-item interactions. To accomplish this, we can add a temporal term that affects user preferences and, therefore, the interaction between users and items.

Recall our original PMF uses the latent of each user and the latent of each movie. We now add temporal latent accounting for the change in time and the user-temporal latent accounting for the change of user reaction over time. Thus, the user-time interaction is a dot product between the user temporal latent and the temporal latent.

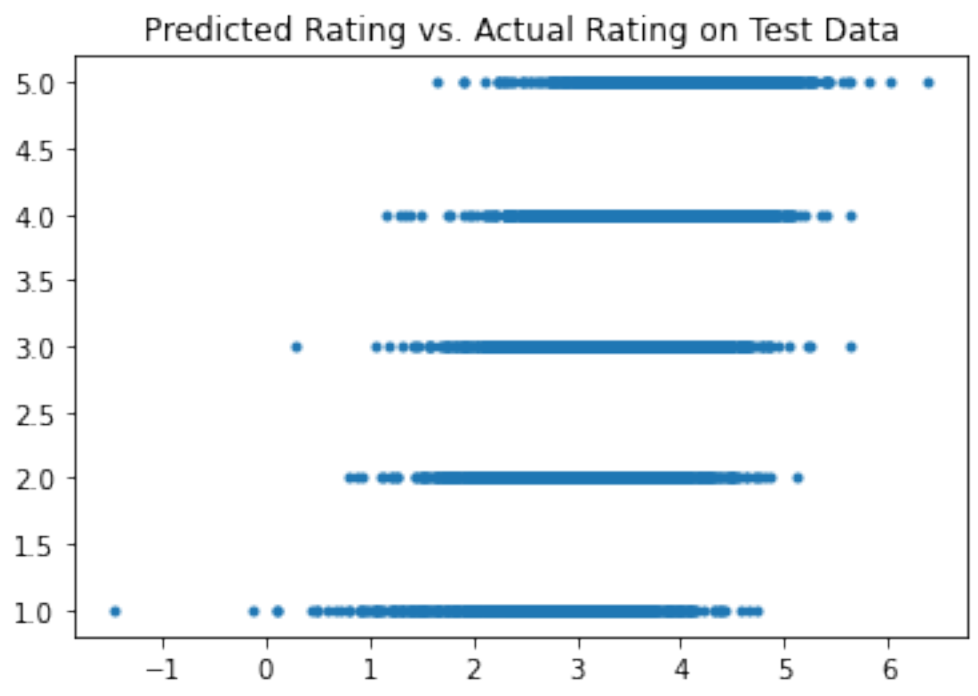
$$r_{ui} = P_u Q_i^T + P_u(t),$$

where r_{ui} is the rating of User u to Item i, and P is user latent factor matrix with dimension $n \times k$, Q is item latent factor matrix with dimension $m \times k$ (the same as the baseline PMF), and $P_u(t)$ takes user factors as a function of time.

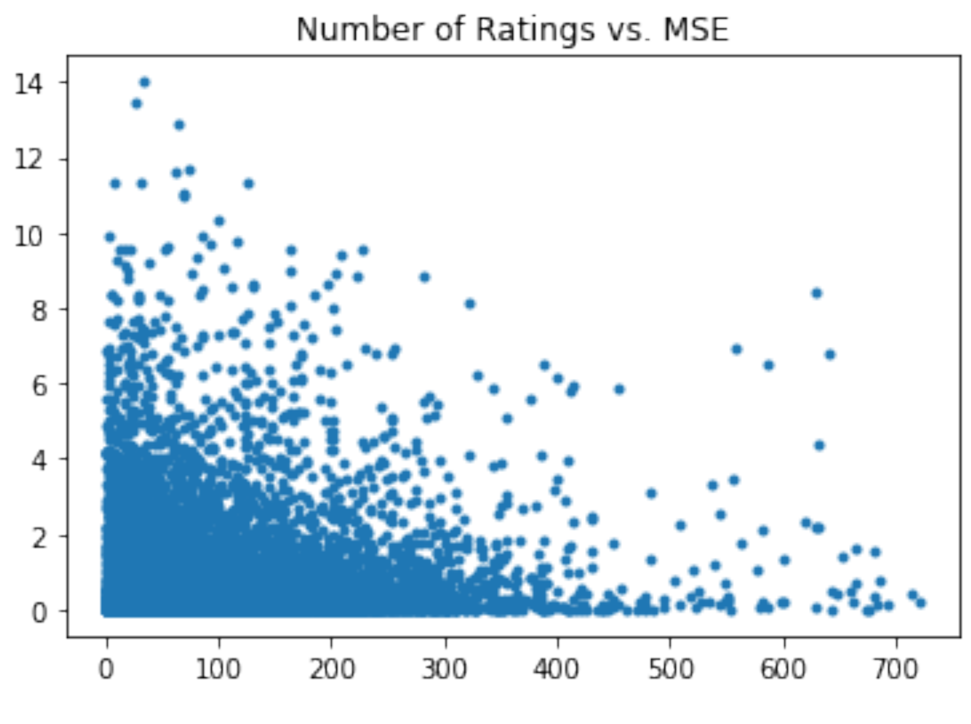
Model and Sanity Check

Number of iterations, training loss, training RMSE, testing RMSE using Variant 1.

```
0 1.2738315 0.94510305 0.94745064
100 1.1786981 0.9038757 0.92231274
200 1.0713223 0.83177096 0.8629265
300 0.9503269 0.7677249 0.80975646
400 0.867045 0.72243065 0.77153456
500 0.82006526 0.69554865 0.75337356
600 0.7926598 0.6790036 0.7440422
700 0.7759146 0.668286 0.73918927
800 0.76499814 0.66105455 0.73678064
900 0.7574418 0.65586495 0.73544556
1000 0.75194424 0.65195405 0.73470646
1100 0.74778193 0.64889616 0.73430705
1200 0.74453 0.64645743 0.7341772
1300 0.7419253 0.64446366 0.7341016
1400 0.7397969 0.64279395 0.7340288
1500 0.7380288 0.6413653 0.73393106
1600 0.73653966 0.64013547 0.73384213
1700 0.7352705 0.6390727 0.7337597
1800 0.73417735 0.6381424 0.73367184
1900 0.73322684 0.63732105 0.7335686
2000 0.7323937 0.6365943 0.7334614
2100 0.7316577 0.63594925 0.73334694
2200 0.7310028 0.6353814 0.7332312
2300 0.7304165 0.63487965 0.7330947
2400 0.72988844 0.63443 0.73293805
2500 0.7294104 0.6340248 0.7327798
2600 0.7289755 0.63365513 0.73261666
2700 0.72857815 0.6333142 0.7324604
2800 0.72821367 0.63300097 0.7322926
2900 0.7278782 0.6327138 0.73212326
3000 0.7275686 0.6324538 0.731955
3100 0.727282 0.63221884 0.7317795
3200 0.7270159 0.63200307 0.7316011
3300 0.7267685 0.63180274 0.73143315
3400 0.7265379 0.6316163 0.7312751
3500 0.7263224 0.6314425 0.73112696
3600 0.726121 0.6312799 0.73098546
3700 0.72593224 0.6311276 0.73084813
3800 0.7257551 0.63098925 0.7307133
3900 0.7255886 0.63086015 0.73058707
4000 0.7254319 0.6307379 0.7304627
4100 0.7252843 0.6306216 0.7303367
4200 0.72514486 0.6305127 0.73021275
4300 0.7250132 0.6304111 0.73009074
4400 0.72488856 0.630314 0.7299749
4500 0.7247707 0.6302223 0.72985834
4600 0.72465885 0.63013697 0.7297491
4700 0.72455275 0.6300566 0.7296424
4800 0.7244519 0.62997955 0.7295351
4900 0.724356 0.6299066 0.72942984
```



From the plot of predicted rating vs. actual rating on test data above, we can see a trend that the range of predicted ratings shifts to the right as the actual rating increases from 1 to 5, indicating that our prediction model follows the correct trend and functions reasonably in predicting users' ratings to the movies. However, the variance in predictions seems to be very large, and we want to investigate further to find out the possible reasons why such large variance exists.



One possible reason for such large variances is that the predicting power of users with less rating history is significantly weaker that that of users with numerous rating histories. Thus, we calcuate the prediction MSE for each user, and we plot the number of ratings vs. MSE.

From the plot above, we see that the variance in MSE decreases as the number of ratings by the user increases. This correponsds to our previous assumption that the prediction for users with many rating histories is more accurate.

```
Out[ ]:

count    10000.000000
mean        95.457200
std         92.991796
min          1.000000
25%         27.000000
50%         66.500000
75%        136.000000
max        720.000000
Name: numRate, dtype: float64
```

Table I: The Summary of the Number of Ratings

```
Out[ ]:

count    2.521000e+03
mean     9.176382e-01
std      1.398241e+00
min      2.698476e-08
25%      8.740047e-02
50%      3.904262e-01
75%      1.096429e+00
max      1.344413e+01
Name: mse, dtype: float64
```

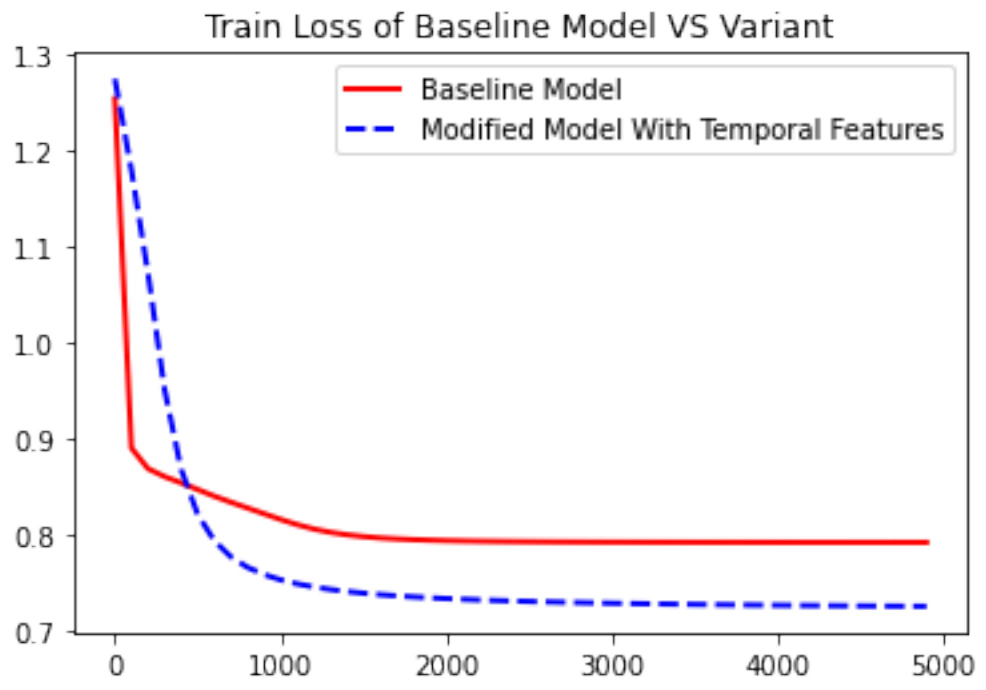
Table II: The MSE Summary for Users in 1st Quantile in Terms of the Number of Ratings

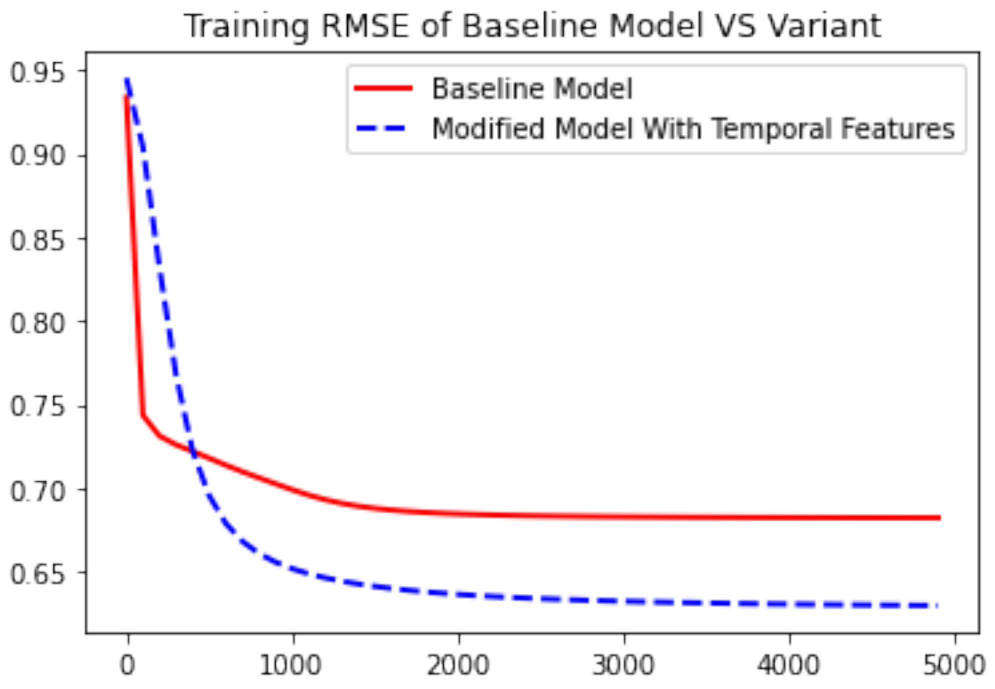
```
Out[ ]:

count    2.515000e+03
mean     8.390989e-01
std      1.349318e+00
min      4.800773e-08
25%      7.149035e-02
50%      3.197210e-01
75%      9.899698e-01
max      9.570035e+00
Name: mse, dtype: float64
```

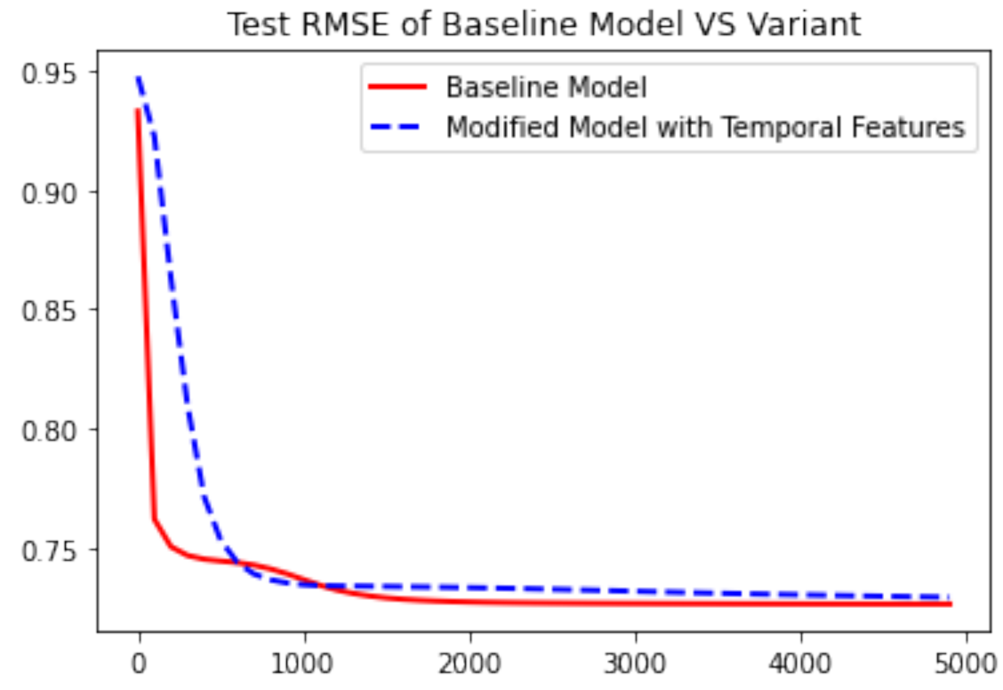
Table III: The MSE Summary for Users in 4th Quantile in Terms of the Number of Ratings

We want to look at the numbers more closely. The average MSE of predictions among users who are in the first quantile in terms of the number of ratings is around 0.9228. The average MSE of predictions among users who are in the first quantile in terms of the number of ratings is around 0.8143. This is an indication to our previous assumption.

Result and Limitation







From the above three plots, we can see that our variant has lowered training loss by 8.71%, and training RMSE by 7.78%. The test RMSE is still slightly higher than the baseline model at the 5000th iteration, but it seems to decrease faster than the baseline model and we can believe that it will at least perform as the same as the baseline model with more iterations.

Since we are using a Stochastic gradient descent to train our model, we can believe that our model reaches the maximum level with appropriate hyperparameters. To improve this model specifically, we might need to incorporate other models, such as other user demographics.

Variant 2: Using Side Info in Addition to Latents.

In the baseline model, both the user latent and the movie latent has two columns, one representing the latent factor and the other representing the bias for the mean. However, a user's preference can vary in movies genres, so the user latent for different movie genres can be different. Therefore, in this model, we add side info for movie genres as a variant to the baseline model.

In this variant, while the movie latent remains to be a $(2 \times n)$ matrix as in the baseline model, we now define the user latent as a $(k + 1 \times n)$ matrix, where k is the number of genres and n is the number of users. Then, we also define a matrix indicate the genres that a movies belong to.

The baseline model predicts the rating based on the movie latent and the user latent in general, and based on the two new matrices we designed, the variant model predicts the rating based on the movie latent and the user latent for the specific genres of the movie.

Model and Sanity Check

Number of iterations, training loss, training RMSE, testing RMSE using Variant 2.

```
0 1.2498736 0.9289224 0.9326297
100 0.8847966 0.74276423 0.7610215
200 0.8600893 0.72953135 0.7495494
300 0.84720314 0.7225243 0.74611557
400 0.8317387 0.7143693 0.7450587
500 0.8127457 0.704211 0.74516636
600 0.7946565 0.69411385 0.74560666
700 0.77919996 0.68519014 0.74668473
800 0.7661655 0.67751616 0.74853015
900 0.7550875 0.6709756 0.7507244
1000 0.74567044 0.66531503 0.75286776
1100 0.7376694 0.66041386 0.7550562
1200 0.7308481 0.65615135 0.75722015
1300 0.7249889 0.6524084 0.7592253
1400 0.7199077 0.64911574 0.7610682
1500 0.715462 0.64623016 0.76281106
1600 0.71154946 0.643664 0.76442295
1700 0.7080948 0.64136153 0.76601696
1800 0.7050393 0.63931453 0.7674964
1900 0.7023325 0.6374722 0.76884866
2000 0.69992745 0.63581496 0.77012664
2100 0.6977811 0.63431895 0.77133673
2200 0.6958541 0.6329705 0.7724469
2300 0.69411385 0.63173604 0.77348214
2400 0.69253254 0.6305969 0.774425
2500 0.6910877 0.6295634 0.7752891
2600 0.68976116 0.62861353 0.77605873
2700 0.6885373 0.6277329 0.77677906
2800 0.6874034 0.62691194 0.777452
2900 0.6863487 0.62614703 0.7780622
3000 0.6853639 0.62543035 0.7786106
3100 0.68444115 0.62475985 0.7791055
3200 0.68357384 0.6241351 0.77955675
3300 0.6827561 0.62354374 0.7799621
3400 0.6819831 0.6229781 0.7803277
3500 0.68125063 0.62244374 0.7806621
3600 0.68055534 0.6219331 0.7809704
3700 0.6798942 0.6214465 0.781236
3800 0.6792652 0.62098163 0.78147537
3900 0.6786665 0.6205401 0.78170925
4000 0.67809653 0.6201188 0.7819334
4100 0.67755437 0.6197173 0.7821384
4200 0.6770391 0.6193342 0.7823306
4300 0.6765499 0.6189644 0.7825119
4400 0.6760861 0.6186096 0.7826895
4500 0.6756465 0.6182692 0.7828566
4600 0.6752301 0.61794704 0.78301275
4700 0.6748358 0.6176404 0.7831788
4800 0.67446244 0.6173475 0.78333175
4900 0.6741086 0.61706597 0.78348565
```

```
Out[ ]:
```

	count
count	19.000000
mean	11189.210526
std	11236.960723
min	10.000000
25%	2729.500000
50%	7182.000000
75%	16607.000000
max	39895.000000

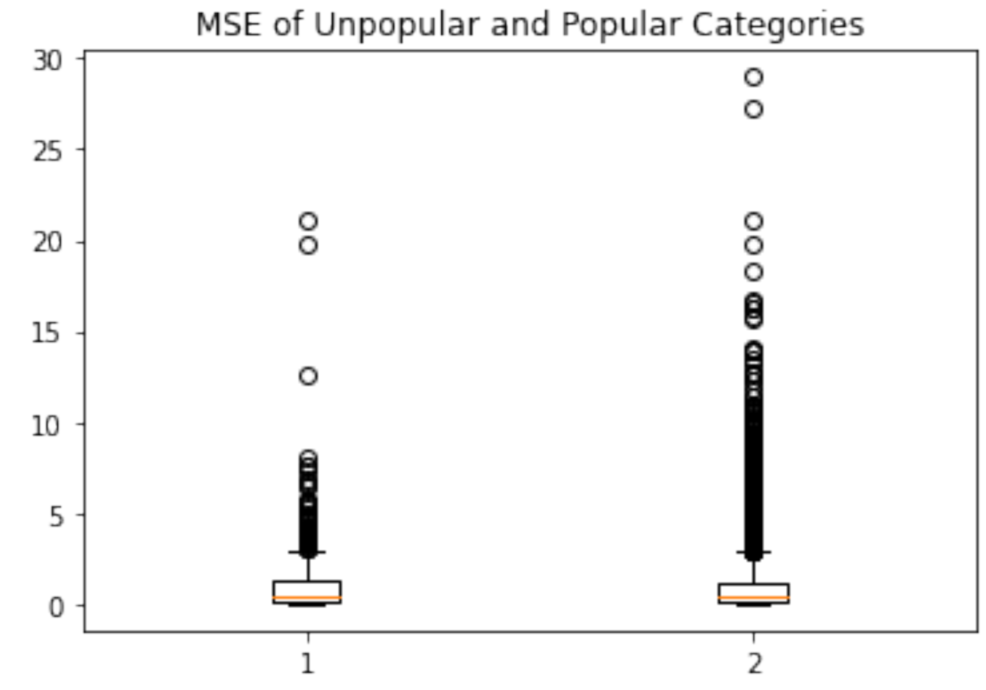
Table 1: The 1st quantile of the number of ratings in each category is 2729.5. And the 3rd qunatile is 16607.

```
Out[ ]:

count    5.340000e+02
mean     1.059674e+00
std      1.896921e+00
min      2.593479e-07
25%      8.927646e-02
50%      4.083659e-01
75%      1.249443e+00
max      2.114979e+01
Name: mse, dtype: float64
```

Table 2: The average MSE of ratings in unpopular categories is 1.059674

```
Out[ ]:

count    9.135000e+03
mean     1.021210e+00
std      1.697306e+00
min      2.313777e-08
25%      9.132229e-02
50%      4.076968e-01
75%      1.210387e+00
max      2.899513e+01
Name: mse, dtype: float64
```

Table 3: The average MSE of ratings in popular categories is 1.021210.



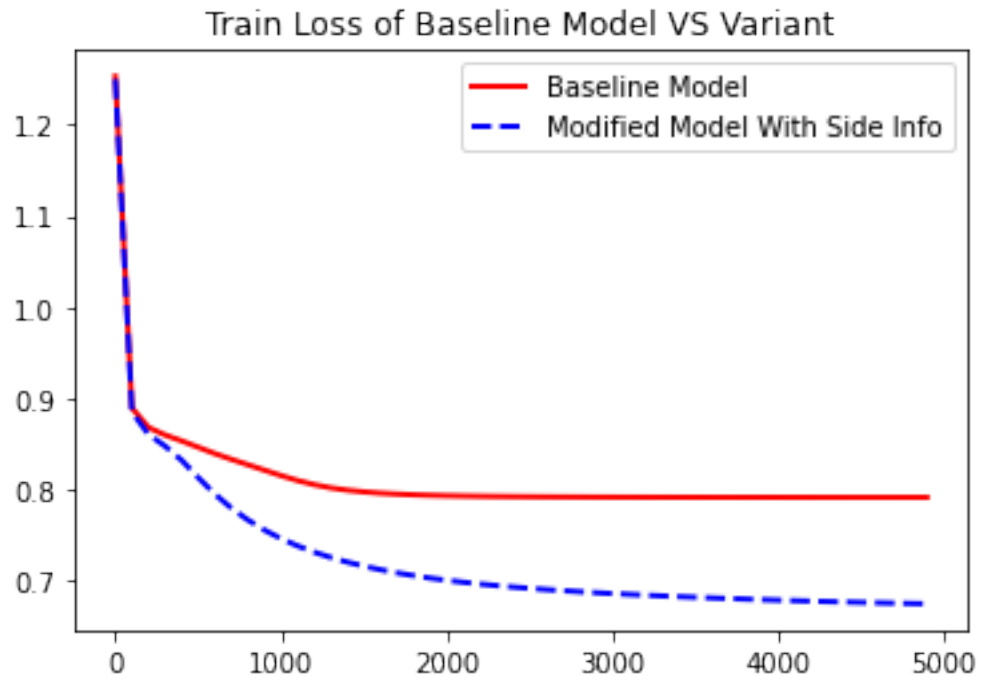
Overall, the MSE of ratings in the unpopular movie categories is slightly higher than that in the popular movie categories, probably because the user latent parameter trained based on fewer ratings in the unpopular categoreis are less reliable.

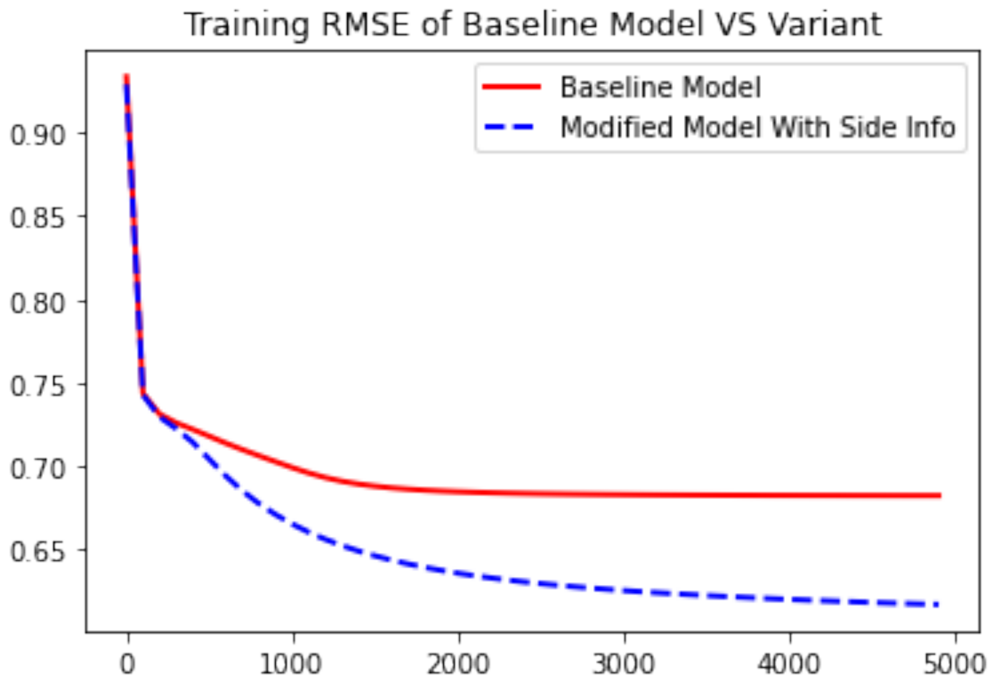
However, the variance of the MSE of ratings in the popular movie categories are larger thann that in the unpopular movie categories. A possible explanation is that a user's preferences for movies in a niche category are believed to be similar, but the preferences of the user for movies in popular categories can vary a lot more.

Result and Limitation

The main question that this model is answering is that a user's preference of movies can vary by genres, so the user latents for different genres can be different.

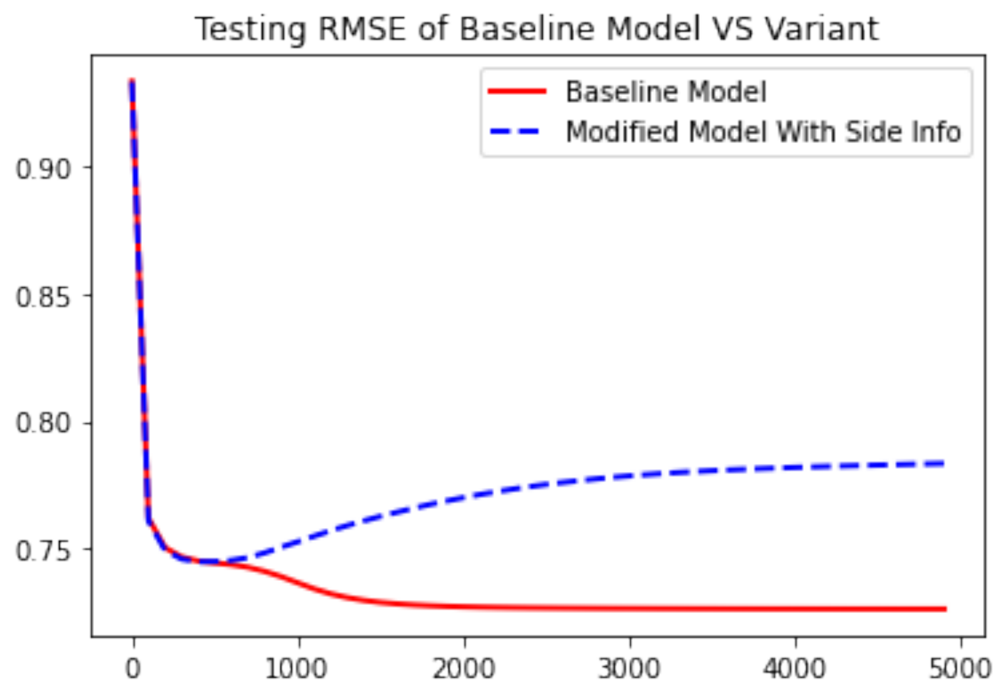
The way to split the training and testing sets is the same as that in the basline model, so the training data is still the first 80000 observations, and the testing data is the last 10000 observations. Therefore, the spliting predicts behavior of a random existing user after the training data was gathered.


Train Loss of Baseline Model VS Variant
Baseline Model
Modified Model With Side Info
1.2
1.1
1.0
0.9
0.8
0.7
0
1000
2000
3000
4000
5000


Training RMSE of Baseline Model VS Variant
Baseline Model
Modified Model With Side Info
0.90
0.85
0.80
0.75
0.70
0.65
0
1000
2000
3000
4000
5000

```
Out[ ]:
```

```
<matplotlib.legend.Legend at 0x7f66712f9780>
```



From the above three curves, this variant model has 11.68% lower training loss and 6.53% lower training RMSE than the baseline model. However, this model has 5.69% higher testing RMSE than the baseline model, probably because the user latents of each genre are trained from ratings in the training sets and is able to capture the features of the preferences of the existing behaviors better.

Moreover, since there are more parameters (user_latents) in this variant, the bias of the model decreases, but as a trade-off, the variance of the model increases. Therefore, the model works better for the training set, but worse for the testing data.

Variant 3: Non-linear Prediction Function

In the original PMF, we used a linear factorization of a user-item matrix. And by using the linear prediction function, we assumed that user's preference is consistent over all the movies. However, in reality, user's ratings are far more complicated. And their preference may change for different items. For example, a user's interest in history decumentaries may not be correlated with their interest in romantic comedies.

And in this variant, we make a combination between global preference and specific taste. First, we model each user with T latent vectors, and all of them correspond to a single latent variable of the item. In addition, since some users have not rated many movies, they have not provided enough information about their T different tastes, so we also use the original PMF as our 'global preference'.

$r_{ui} = P_u Q_i^T + max_{t=1,\ldots,T} W_{ut} Y_i^T$,

where r_{ui} is the rating of User u to Item i, and P is user latent factor matrix with dimension $n \times k$, Q is item latent factor matrix with dimension $m \times k$ (the same as the baseline PMF). And W $(n \times k \times T)$ is the user latent factor matrix representing T specific tastes of the user. And $Y(m \times k)$ is the item latent factor matrix for taste-specific component.

Every time, the specific taste among T different tastes that matches best is captured using max. And this model is first introduced by Santosh Kabbur and George Karypis.

Model and Sanity Check

Number of iterations, training loss, training RMSE, testing RMSE using Variant 3.

```
0 1.2343854 0.93208015 0.9366645
100 0.8886361 0.74384 0.7619097
200 0.86869806 0.7321538 0.75036067
300 0.8625817 0.72799706 0.7467962
400 0.86002976 0.72600085 0.74543107
500 0.8587565 0.7248883 0.7448065
600 0.8580302 0.7241874 0.74449277
700 0.8575417 0.7236973 0.74431115
800 0.857055 0.72327733 0.7441884
900 0.85607487 0.7226706 0.7440445
1000 0.85311055 0.72114116 0.7436525
1100 0.8454443 0.7171823 0.74244034
1200 0.83398706 0.7107973 0.7401285
1300 0.8229741 0.70442355 0.7371639
1400 0.8136789 0.6990736 0.7340649
1500 0.8062243 0.6947087 0.73152196
1600 0.7999901 0.69097126 0.7297533
1700 0.7941456 0.687358 0.7282906
1800 0.78850675 0.6836443 0.7269828
1900 0.7834191 0.68011045 0.72584945
2000 0.7790727 0.6769505 0.72503966
2100 0.77554643 0.6742992 0.7244073
2200 0.7728058 0.67216164 0.72392386
2300 0.7706731 0.670431 0.7235191
2400 0.768984 0.6690053 0.7231762
2500 0.76762426 0.66781676 0.7229147
2600 0.766518 0.6668035 0.72269887
2700 0.7656089 0.66593504 0.72250897
2800 0.76485586 0.6651882 0.72235304
2900 0.7642273 0.6645376 0.7222197
3000 0.7637002 0.66397506 0.7221007
3100 0.7632553 0.6634838 0.7220038
3200 0.7628776 0.663049 0.72192746
3300 0.76255554 0.66266537 0.72186625
3400 0.7622794 0.6623277 0.72181743
3500 0.7620418 0.66202956 0.721785
3600 0.76183516 0.66176265 0.721756
3700 0.76165736 0.66152406 0.7217345
3800 0.7615017 0.6613096 0.72172076
3900 0.7613649 0.6611172 0.7217081
4000 0.761245 0.66094345 0.7216983
4100 0.7611393 0.6607861 0.7216903
4200 0.76104444 0.6606421 0.7216811
4300 0.7609604 0.66051084 0.72167087
4400 0.76088536 0.66039085 0.72166395
4500 0.7608026 0.6602723 0.7216749
4600 0.7607039 0.66015786 0.72167736
4700 0.76062006 0.66005164 0.72167784
4800 0.76055515 0.65995646 0.72168
4900 0.7604997 0.6598727 0.721687
```

```
Out[ ]:
```

	count	userID
count	943.000000	943.000000
mean	106.044539	472.000000
std	100.931743	272.364951
min	20.000000	1.000000
25%	33.000000	236.500000
50%	65.000000	472.000000
75%	148.000000	707.500000
max	737.000000	943.000000

Table 4: The 1st quantile of the number of movies rated by each user is 33. And the 3rd qunatile is 148.

```
Out[ ]:

count    601.000000
mean       1.001852
std        1.423218
min        0.000032
25%        0.104482
50%        0.483597
75%        1.245067
max        8.281295
Name: mse, dtype: float64
```

Table 5: The average MSE of obervations rated by user who has fewer ratings is 0.999263.

```
Out[ ]:

count    5.492000e+03
mean     8.087752e-01
std      1.291245e+00
min      1.099853e-07
25%      6.939643e-02
50%      3.337879e-01
75%      9.858964e-01
max      1.548185e+01
Name: mse, dtype: float64
```

Table 6: The average MSE of obervations rated by user who has many ratings is 0.8154928.

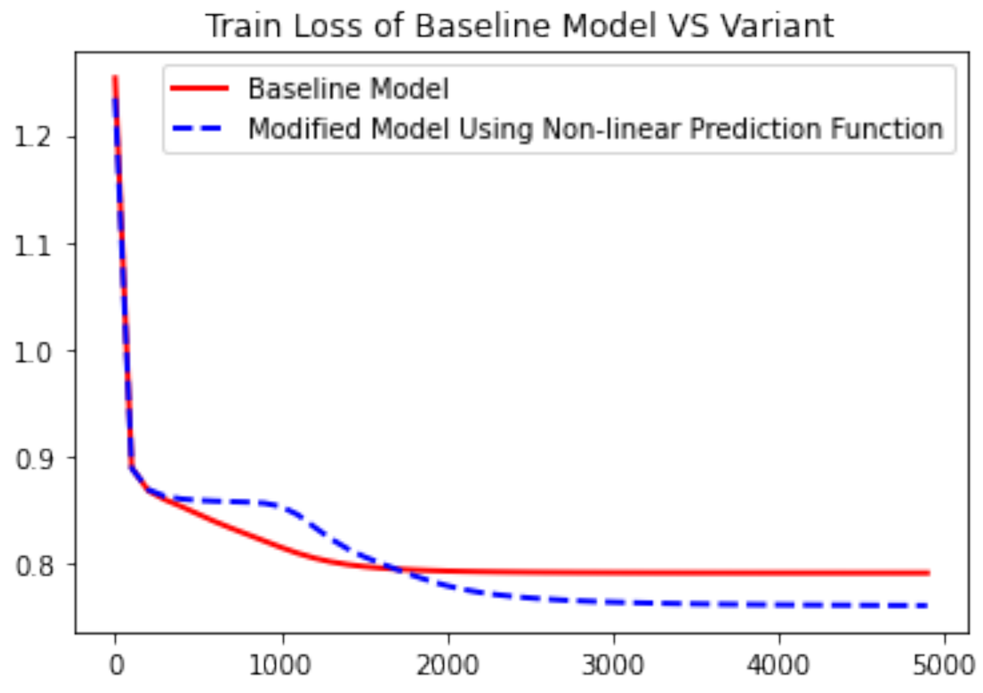
However, since for both 'global preference' and 'specific taste', we need many data to discover user's latent and their different tastes. So we would expect that users with fewer ratings would have perform worse than users with many ratings.

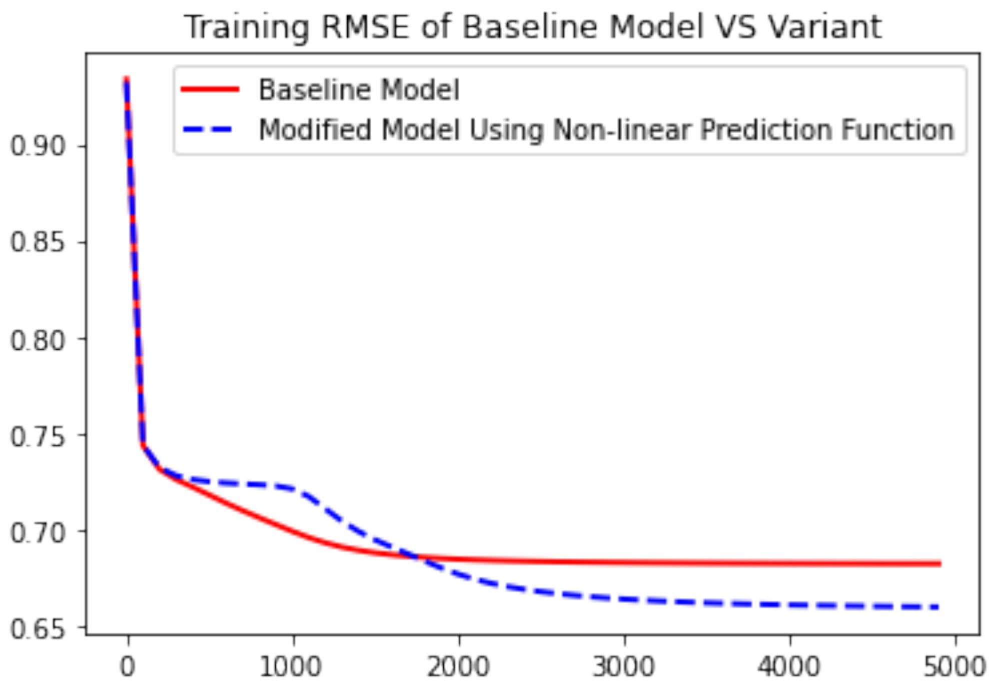
And this is proved. The average MSE of predictions among users who have watched less than 33 (first quantile of all 943 users) is around 1. And the average MSE of predictions among users who have watched more than 148 (third quantile of all 943 users) is around 0.815. And first quantile, median, third quantile of MSE among users who have rated many movies are all smaller than that among users who have rated fewer movies.

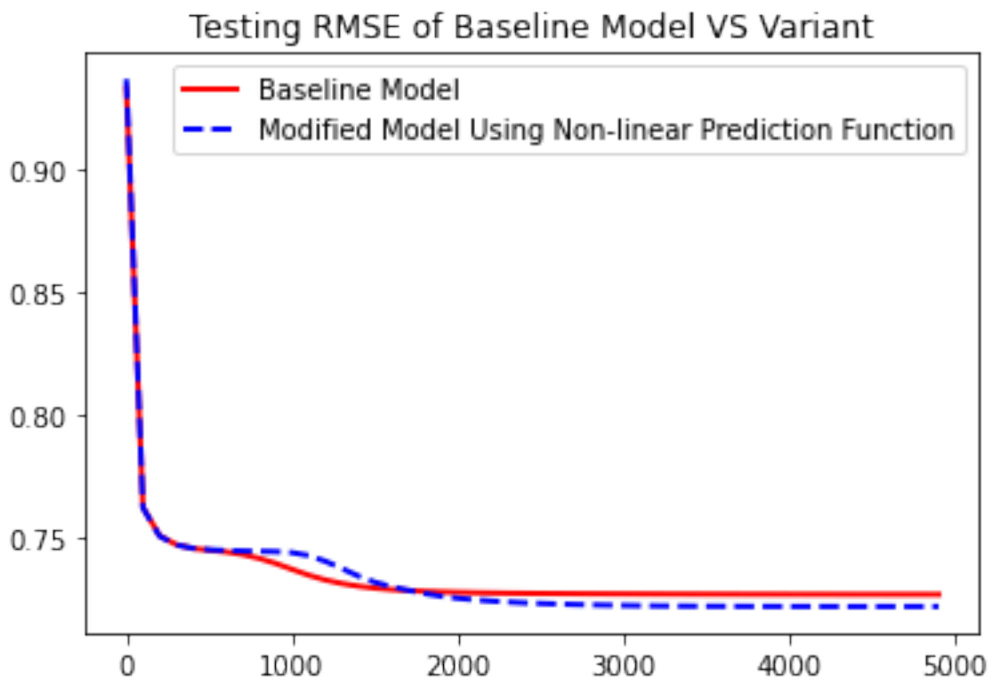
Result and Limitation

The main problem that this variant has solved is that user's preference is not consistent over all the movies, and 'a user's interest in history decumentaries may have little correlation with their interest in romantic comedies'.

And we did not change the way to split training and testing data in this variant. So the training data is still the first 80000 observations, and the testing data is the last 10000 observations. So it predicts behavior of a random existing user after the training data was gathered.







From the above three plots, we can see that our variant has lowered training loss by 3.850%, training RMSE by 3.294%, and testing RMSE by 0.675% than the simple PMF.

However, since our variant also needs many data to train and learn users' latents. In addition, it also assumes that user has many tastes. So if many new users joining, the accuracy will be lower. Besides, if many users only rate few movies, the accuracy will be lower too. But if many users stick around and rate more movies, the accuracy will increase over time.