NETFLIX RECOMMENDATION SYSTEM-

1. Business Problem

1.1 Problem Description

Netflix is all about linking people to the movies they love. To help customers find those movies, they developed a superlative movie recommendation system: CinematchSM. Its job is to predict whether someone will enjoy a movie based on how much they liked or disliked other movies. Netflix use those predictions to make personal movie recommendations based on each customer's unique tastes. And while Cinematch is doing pretty well, it can always be made better.

Now there are a lot of interesting alternative approaches to how Cinematch works that Netflix haven't tried. Some are described in the literature, some aren't. We're curious whether any of these can beat Cinematch by making better predictions. Because, frankly, if there is a much better approach it could make a big difference to our customers and our business.

Credits: https://www.netflixprize.com/rules.html

1.2 Problem Statement

Netflix provided a lot of anonymous rating data, and a prediction accuracy bar that is 10% better than what Cinematch can do on the same training data set. (Accuracy is a measurement of how closely predicted ratings of movies match subsequent actual ratings.)

1.3 Sources

- https://www.netflixprize.com/rules.html
- https://www.kaggle.com/netflix-inc/netflix-prize-data
- Netflix blog: https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429 (very nice blog)

- surprise library: http://surpriselib.com/ (we use many models from this library)
- surprise library doc: http://surprise.readthedocs.io/en/stable/getting_started.html (we use many models from this library)
- installing surprise: https://github.com/NicolasHug/Surprise#installation
- Research paper: http://courses.ischool.berkeley.edu/i290-dm/s11/SECURE/a1-koren.pdf (most of our work was inspired by this paper)
- SVD Decomposition : https://www.youtube.com/watch?v=P5mlg91as1c

---

1.4 Real world/Business Objectives and constraints

Objectives:

1. Predict the rating that a user would give to a movie that he ahs not yet rated.
2. Minimize the difference between predicted and actual rating (RMSE and MAPE)

Constraints:

1. Some form of interpretability.
2. Latency requirements are minimum. Computation can be done when user is logged out too. Because a user does not watch more than 2 movies a day generally.

---

2. Machine Learning Problem

2.1 Data

2.1.1 Data Overview

Get the data from : https://www.kaggle.com/netflix-inc/netflix-prize-data/data

Data files :

- combined_data_1.txt
- combined_data_2.txt
- combined_data_3.txt
- combined_data_4.txt
- movie_titles.csv- contains movie_id and the movie_name.

The first line of each file [combined_data_1.txt, combined_data_2.txt, combined_data_3.txt, combined_data_4.txt] contains the movie id followed by a colon. Each subsequent line in the file corresponds to a rating from a customer and its date in the following format:

CustomerID,Rating,Date

MovieIDs range from 1 to 17770 sequentially.

CustomerIDs range from 1 to 2649429, with gaps. There are 480189 unique users. All customerIds do not have users thus.

Ratings are on a five star (integral) scale from 1 to 5.

Dates have the format YYYY-MM-DD.

2.1.2 Example Data point

1:

1488844,3,2005-09-06

822109,5,2005-05-13

885013,4,2005-10-19

30878,4,2005-12-26

823519,3,2004-05-03

893988,3,2005-11-17

124105,4,2004-08-05

1248029,3,2004-04-22

1842128,4,2004-05-09

2238063,3,2005-05-11

1503895,4,2005-05-19

2207774,5,2005-06-06

2590061,3,2004-08-12

2442,3,2004-04-14

543865,4,2004-05-28

1209119,4,2004-03-23

804919,4,2004-06-10

1086807,3,2004-12-28

1711859,4,2005-05-08

372233,5,2005-11-23

1080361,3,2005-03-28

1245640,3,2005-12-19

558634,4,2004-12-14

2165002,4,2004-04-06

1181550,3,2004-02-01

1227322,4,2004-02-06

427928,4,2004-02-26

814701,5,2005-09-29

808731,4,2005-10-31

662870,5,2005-08-24

337541,5,2005-03-23

786312,3,2004-11-16

1133214,4,2004-03-07

1537427,4,2004-03-29

1209954,5,2005-05-09

2381599,3,2005-09-12

525356,2,2004-07-11

1910569,4,2004-04-12

2263586,4,2004-08-20

2421815,2,2004-02-26

1009622,1,2005-01-19

1481961,2,2005-05-24

401047,4,2005-06-03

2179073,3,2004-08-29

1434636,3,2004-05-01

93986,5,2005-10-06

1308744,5,2005-10-29

2647871,4,2005-12-30

1905581,5,2005-08-16

2508819,3,2004-05-18

1578279,1,2005-05-19

1159695,4,2005-02-15

2588432,3,2005-03-31

2423091,3,2005-09-12

470232,4,2004-04-08

2148699,2,2004-06-05

1342007,3,2004-07-16

466135,4,2004-07-13

2472440,3,2005-08-13

1283744,3,2004-04-17

1927580,4,2004-11-08

716874,5,2005-05-06

4326,4,2005-10-29

2.2 Mapping the real world problem to a Machine Learning Problem

---

2.2.1 Type of Machine Learning Problem

For a given movie and user we need to predict the rating would be given by him/her to the movie.

The given problem is a Recommendation problem. Matrix Factorization concepts.

It can also be seen as a Regression problem. The data can be used to predict a regression rating value from 1 to 5. (D={xi,yi}).

2.2.2 Performance metric

- Mean Absolute Percentage Error: https://en.wikipedia.org/wiki/Mean_absolute_percentage_error
- Root Mean Square Error: https://en.wikipedia.org/wiki/Root-mean-square_deviation

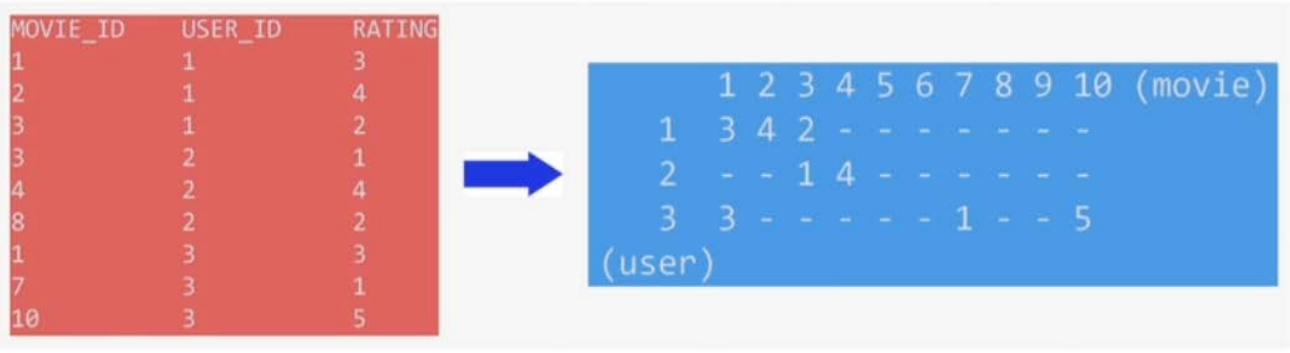
---

1. Minimize RMSE.
2. Try to provide some interpretability.

Steps followed-

1. Contents from data.txt is converted to data.csv. It is in the form {Mj, Ui, rating, date}. The dataframe is sorted in ascending order of date.
2. NaN values are checked using df.isnull().any().
3. Duplicates are checked.
4. The ascending date sorting was done to train-test-split the data temporally. (TBS). Suppose, we deploy the model today, the training will be done based

on data available before today. And the testing and future predictions will be done on data available from today. Thus, TBS is important.

5. Day of week is added as a feature.
6. Some EDA is performed on the train data.
7. Most ratings are either 3 or 4. # Ratings ==5 > #Ratings ==1. Hence, ratings are high on average.
8. No. of data points sharply increases post 2003. Netflix as a company grew during this period probably.
9. Thus, our test data is mostly of post 2003.
10. Now, no. of ratings of movies per user were extracted using groupby.
11. KDE plots (both PDF and CDF) were plotted. Most users rate a VERY FEW no of movies with few rating even above 5000 movies!
12. No. of movies rated per user was observed.
13. The quantiles (0.05 and 0.25) were plotted.
14. Now, no. of ratings of users per movie was extracted using groupby.
15. Distribution of ratings is skewed. Movies like Titanic have large number of viewership and rating.
16. Countplot for day of the week vs ratings was plotted. Friday, Saturday, Sunday receive least ratings maybe due to outdoor activities in foreign countries.
17. Boxplot was plotted for the same to check usefulness of day of week feature.
18. Boxplot gave very bad interpretation. And groupby['day of week'].ratings.mean() gave similar results for all week days.
19. Thus DoW was dropped.
20. The users and movies column was converted to a sparse matrix [based on ratings] using scipy.sparse.csr_matrix.
21. 

22. The compressed sparse percentage has 99.82 percent sparse values.
23. A csr matrix is also created for test data.

24. From the train csr matrix, global average rating for all the movies by all users is computed. Vectors of User average and movie averages too.
25. The pdf and cdf of the user average vector and movie average vector are plotted
26. The Cold Start problem in recommender systems is encountered. Since TBS is done, there will be some users and movies in our test data which were not part of our training data.
27. The problem is not very severe in movies but severe in users.
28. Similarity matrices (user-user and movie-movie) will crash the RAM (using all data). it is 405k*405k dimensional and is not sparse.
29. I planned to convert 17k dimensions to 500 using Truncated SVD. However, intuitively, it would take a long time too as the new 500-dim matrix is dense and not sparse like csr.
30. A DP approach was taken. Instead of computing the entire u-u sim matrix, we compute a list containing all u-u pair values for the required user using scikit cosine similarity.. And then argsort to contain the top n no. of users.

-

An alternative is to compute similar users for a particular user, whenenver re quired (**ie., Run time**)

- We maintain a binary Vector for users, which tells us whether we already computed or not..

- ***If not*** :

- Compute top (let's just say, 1000) most similar users for this given user, and add this to our datastructure, so that we can just access it(similar users) w ithout recomputing it again.

-

- ***If It is already Computed***:

- Just get it directly from our datastructure, which has that information.

- In production time, We might have to recompute similarities, if it is com puted a long time ago. Because user preferences changes over time. If we coul d maintain some kind of Timer, which when expires, we have to update it (re compute it).

-

- ***Which data-structure to use:***

- It is purely implementation dependant.

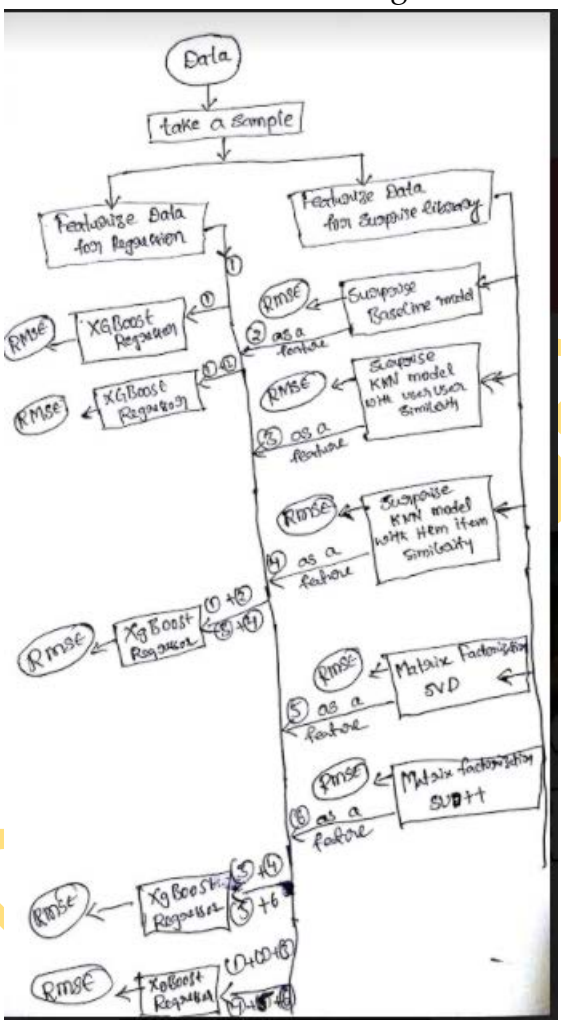
- One simple method is to maintain a **Dictionary Of Dictionaries**.

-

- **key :** _userid_

- __value__: _Again a dictionary_

- **__key__** : *_Similar User_*
- **__value__**: *_Similarity Value_*

31. Surprise Library was used to implement various RS algorithms. The library was inspired by Yehuda Koren's (winner of Netflix Prize) seminal paper.
32. Given below is my modelling strategy. First a sample of the data is taken for computational limitations. (approx. 10k*1k out of 405k*17k dimensions of the csr matrix). Then 13 handcrafted features are added to an XGBRegressor. The Surprise Library baseline models, KNN with u-u similarity and KNN with m-m similarity, SVD and SVD++ are used to add more features cumulatively. These are fed to the XGBRegressor.
33. 

34. Now, Handcrafting the features is done. We have only the (ui,mj) to create the feature from. The rating is the prediction label.
35. The first feature is global rating of all users for all movies. The next feature is the average rating given by the user. The next is the average rating received by the movie. The next is the ratings given by top 5 similar users to the movie (such that ratings are non-void). The next is ratings received by top 5 similar movies by the user (which are non-void).
36. RMSE was around 1 and MAPE was 34% (this is on a SAMPLE of the data)

37. The features are passed on to XGBRegressor. Feature importance shows UserAvg and MovieAvg are most important.
38. Now, modelling with Surprise Library.
39. Given below is the predictions of the baseline model.
40. $$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$
    - μ : Average of all trainings in training data.
    - b_u : User bias
    - b_i : Item bias (movie biases)
41. Here, u is the global average of all ratings. bu is related with user-average and bi resembles movie avg. Our task is to optimize bu and bi by least squares method with L2-regularization. The baseline model does that
42. $$\sum_{r_{ui} \in R_{train}} (r_{ui} - (\mu + b_u + b_i))^2 + \lambda \left(b_u^2 + b_i^2\right) . \text{ [mimimize } b_u, b$$
43. The baseline model is RUN with SGD optimization and 0.001 learning rate perameters.
44. Now the 13 features are added with the bslpr predictions (as a feature). And then XGBRegressor is run. Interestingly, bslpr shows least feature importance.
45. The Surprise KNN Baseline features are very similar to the 5 similar user ratings (for same movie) and 5 similar movie ratings (by same user).
46. KNN model's predicted rating formula is shown below. bui is the prediction from Baseline Model.
47. - **predicted Rating** : (_ **based on User-User similarity** _)
    $$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$
    - b_{ui} - *Baseline prediction* of (user,movie) rating
    - $N_i^k(u)$ - Set of **K similar** users (neighbours) of **user (u)** who rated **movie(i)**
    - *sim (u, v)* - **Similarity** between users **u and v**
        - Generally, it will be cosine similarity or Pearson correlation coefficient.
        - But we use **shrunk Pearson-baseline correlation coefficient**, which is based on the pearsonBaseline similarity (we take line predictions instead of mean rating of user/item)
48. Here, sim (u,v)= cosine/Pearson/shrunk Pearson correlation similarity between u and v. rvi- rating given by user v to movie i and bvi-baseline prediction of the rating.

49. Hyperparams are set as No. of NNs=40 and shrinkage=100. The KNN baseline (both user-based=True and False) overfits but gives better accuracy than the simple baseline model
50. The KNN-added features did not add much value since we already have them in handcrafted features.

- **Predicted Rating :**
  - $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$
    - q_i - Representation of item(movie) in latent factor space
    - p_u - Representation of user in new latent factor space

51.

- **Optimization problem with user item interactions and regularization (to avoid overfitting)**
  - $\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2 \right)$

52.
53. So, predicted rating is also dependent on baseline rating here. Added with SVD matrix factorization.
54. pi and qu are n-dimensional. Here n is the no. of factors and is a h-param kept 100 in our case.
55. Next, we use SVD++ algorithm which uses the concept of implicit feedback.
56. Ratings given are cardinal numbers and act as explicit feedback. Implicit feedback is derived from the behaviour of the user. For example, the very fact that an Amazon customer spent some time on a product page shows that he was interested in it. In our case, the very fact that a user gave some rating to a movie implicitly feedbacks that he spent some time with it and was interested in it.

57.

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

- I_u --- the set of all items rated by user u
- y_j --- Our new set of item factors that capture implicit ratings.

58. SVD++ algo features (n- new- factors=50) combined with XGBRegressor is run.
59. Next, all the 13 features combined with Baseline+KNN_Baseline_user+KNN_baseline_items+SVD+SVD++ features and fed into the regressor. SVD and SVD++ features show least importance.
60. Now the 13 features are excluded and only the surprise Library features are included. Feature importance shows SVD predictions have max value and SVDpp the least.
61. Comparing all the models stored in our dictionary SVD performs best with RMSE=1.072 approx and MAPE=34 percent approx..
62. This was done with small subset of data and NO hyperparameter tuning.
63. Future improvements-

1.Instead of using 10K users and 1K movies to train the above models, use 25K users and 3K movies (or more) to train all of the above models. Report the RMSE and MAPE on the test data using larger amount of data and provide a comparison between various models as shown above..

2.Tune hyperparamters of all the Xgboost models above to improve the RMSE.