Netflix Recommender System:

Business Problem:

Netflix is all about connecting people to the movies they love. To help customers find those movies, they developed world-class 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.

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.)

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:

Some form of interpretability.

Data

Data files : combined_data_1.txt combined_data_2.txt combined_data_3.txt combined_data_4.txt movie_titles.csv

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 users. Ratings are on a five star (integral) scale from 1 to 5. Dates have the format YYYY-MM-DD.

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
It can also seen as a Regression problem

Performance Metric:

Mean Absolute Percentage Error
Root Mean Squared Error

Split Data:

Convert / Merge whole data to required format: u_i, m_j, r_ij.Split whole data into 80:20 ratio(Temporal split)

Train Data:

No of movies in train data: 17424
No of users in train data: 405041
no of ratings in train data: 80384405

Test Data:
No of movies in test data: 17757
No of users in test data: 349312
no of ratings in test data: 20096102

Exploratory Data Analysis:

1. Distribution of ratings:Typically freom the plot it seems that most of the people has rated either 3 or 4.
2. Number of ratings given every day : Adding extra column(weekday) to data.It is notable that on saturdays and sundays predicted count goes low.Plotting train data vs month,we could see that the plot has a sharp peak at 2005.
3. Average raings given by user for a movie:
a. On looking the percentile values : 25th,50th,75th and 100th percentile-34,89,245,17112, we could come to an understanding that the 100th percentile value is way higher than the 75th.
b. No of ratings at last 5 percentile : 20305
4. Average number of users rated for a given movie:
a. Just like average raings given by user for a movie,average number of users rated for a given movie is skewed.
b. There are some movies (which are very popular) which are rated by huge number of users.
c. But most of the movies(like 90%) got some hundereds of ratings.

Creating sparse matrix:

1. Creating sparse train matrix from train data frame
2. Creating sparse test matrix from test data frame

Featurizations:

Since we have data in u_i, m_j, r_ij format,it is possible to extract features like:

1. Global average of all movie ratings
2. Global average rating per user
3. Global average rating per movie

Along with these features we could also take features like:

1. movie-movie similarity : top 5 similar movies rated by user
2. user-user similarity : top 5 similar users who rated that movie.

Features:

- **GAvg** : Average rating of all the ratings
- **Similar users rating of this movie**:
  - sur1, sur2, sur3, sur4, sur5 (top 5 similar users who rated that movie..)
- **Similar movies rated by this user**:
  - smr1, smr2, smr3, smr4, smr5 (top 5 similar movies rated by this movie..)
- **UAvg** : User's Average rating
- **MAvg** : Average rating of this movie
- **rating** : Rating of this movie by this user.

Surprise Model:

- We can't give raw data (movie, user, rating) to train the model in Surprise library.
- They have a saperate format for TRAIN and TEST data, which will be useful for training the models like SVD, KNNBaseLineOnly ..etc.. in Surprise

KNNBaseLineOnly....etc..,in Surprise.

- We can form the trainset/testset from a file, or from a Pandas DataFrame.

Models:

1. XGBoost with initial 13 features:
Using those base 13 features we train model.

2.Suprise BaselineModel :
Base Line model says the predicted value is the sum of global value of all ratings,bias for each user and bias for each movie.

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

- $\pmb \mu $: Average of all ratings in training data.
- $\pmb b_u$: User bias
- $\pmb b_i$: Item bias (movie biases)

Optimization function (Least Squares Problem)

> $ \large \sum_{r_{ui} \in R_{train}} \left(r_{ui} - (\mu + b_u + b_i)\right)^2 + \lambda \left(b_u^2 + b_i^2 \right).\text { [mimimize } } {b_u, b_i]}$

3. XGBoost with initial 13 features + Surprise Baseline predictor:
we use XGBoost with base 13 features and also we use results from surprise base line model.

4.Surprise KNN predictors:

\begin{align} \hat{r}_{ui} = b_{ui} + \frac{ \sum\limits_{v \in N^k_i(u)} \text{sim}(u, v) \cdot (r_{vi} - b_{vi})} {\sum\limits_{v \in N^k_i(u)} \text{sim}(u, v)} \end{align}

- $\pmb{b_{ui}}$ - *Baseline prediction* of (user,movie) rating
- $ \pmb {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 base line predictions instead of mean rating of user/item)

5. XGBoost with initial 13 features + Surprise Baseline predictor + KNNBaseline predictor(movie-movie and user-user):
We train whole data using XGBoost with those 13 base features,results from surprise base line model and predictions form both knn models(that uses User_User and Item_Item similarities along with our previous features).

6. SVD:

- Predicted Rating :
  - $ \large \hat r_{ui} = \mu + b_u + b_i + q_i^Tp_u $
    - $\pmb q_i$ - Representation of item(movie) in latent factor space
    - $\pmb p_u$ - Representation of user in new latent factor space

7. SVD with implicit feedback:

- Predicted Rating :
  - $ \large \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) $
    - $ \pmb{I_u}$ --- the set of all items rated by user u
    - $\pmb{y_j}$ --- Our new set of item factors that capture implicit ratings.
    - Optimization problem with user item interactions and regularization (to avoid overfitting) -$ \large \sum_{r_{ui} \in R_{train}} \left(r_{ui} - \hat{r}_{ui} \right)^2 + \lambda\left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 + ||y_j||^2\right) $

8. XgBoost with 13 features + Surprise Baseline + Surprise KNNbaseline + MF
We train whole data using XGBoost with those 13 base features,results from surprise base line model ,predictions form both knn models(that uses User_User and Item_Item similarities along with our previous features) and results MF techniques(svd and svd with implicit)

9.XgBoost with Surprise Baseline + Surprise KNNbaseline + MF Techniques:
We train whole data using XGBoost and features are results from surprise base line model ,predictions form both knn models(that uses User_User and Item_Item similarities along with our previous features) and results MF techniques(svd and svd with implicit)

Conclusion:

Model	Train_rmse	Test_rmse
first_algo	0.8131655657201271	1.0736360638377618
bsl_algo	0.9408643608703676	1.0735808085059688
xgb_bsl	0.8206049195071748	1.073068397290458
knn_bsl_u	0.26375340495634786	1.0729740633728826
knn_bsl_m	0.20738563427680795	1.0729630032608906
xgb_knn_bsl	0.7634494751495742	1.086218472877775
svd	0.6486886147912374	1.0730035598473069
svdpp	0.5795854676897524	1.0730045699350494
xgb_final	0.8068454624363004	1.0734467307804618
xgb_all_models	1.0606454093218731	1.0752397685718569