PRML PROJECT

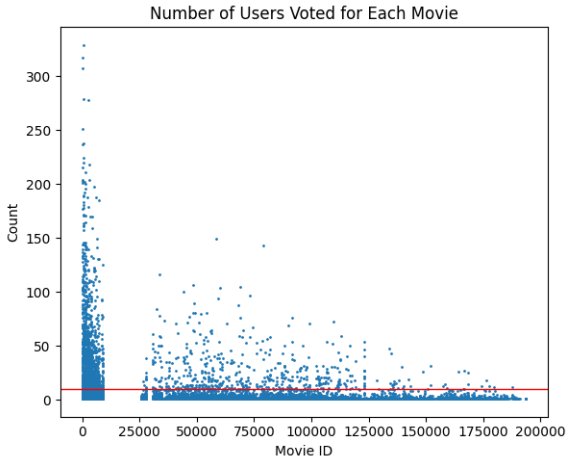
GROUP ID - 30

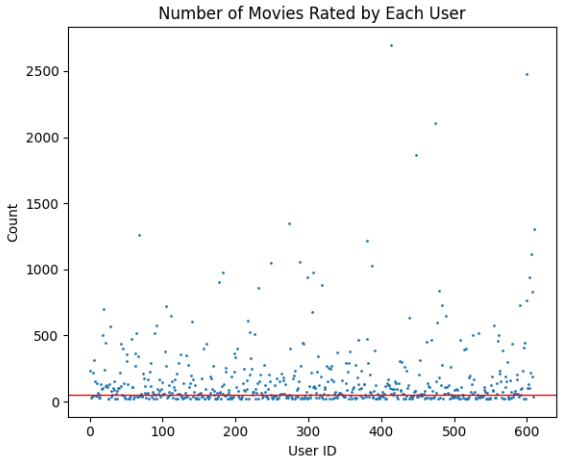
April 20, 2024

1 DATA PRE-PROCESSING AND VISUALISATION

1.1 USER-RATING

1. Consider the 'ratings.csv' file. The minimum rating any user has given in 0.5 and the maximum being 5.
2. Let's convert it into a pivot table and replace the Nan with 0. So any movie with rating 0 to a specified user is not watched.
3. Not all the users and the movies are trustworthy. What I mean by that is say there is a movie that is only watched and rated by 2 or 5 users it is not so popular meaning it won't interest the remaining mass.
4. Similar thing can be said for the sake of users where a user only rated 10 movies he might not be such a trust worth critique whose opinion matters or is valuable.
5. Hence we will consider only those movies which have been rated by atleast 10 users and users who have atleast rated 50 movies.





userId	1	4	6	7	10	11	15	16	17	18	...	600	601	602	603	604	605	606	607	608	610
movieId																					
1	4.0	0.0	0.0	4.5	0.0	0.0	2.5	0.0	4.5	3.5	...	2.5	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	5.0
2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	...	4.0	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0
3	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
5	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.5	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
6	4.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0	0.0	4.0	...	0.0	0.0	3.0	4.0	3.0	0.0	0.0	0.0	0.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
174055	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
176371	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
177765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
187593	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 1: final user-rating data

1.2 Features Of a Movie

1. Consider the data mentioned in mid-report. There are multiple features of a movies like genre, Year, Director and stars.
2. But there are multiple entries in each feature for a movie. So what we have decided to do was make multiple instances of the same movie consider all different possible features.
3. Later when we consider recommending we will take average of all those instances and result accordingly.
4. Here we have considered only those years in which atleast 10 movies have released, directors who have atleast directed 5 movies, stars who have atleast acted in 5 movies.

	movieId	genres	Year	Director	Stars
0	1	adventure	1995	johnlasseter	tomhanks
1	1	adventure	1995	johnlasseter	timallen
2	1	adventure	1995	johnlasseter	donrickles
3	1	adventure	1995	johnlasseter	jimvarney
4	1	animation	1995	johnlasseter	tomhanks
...	...	...	...	...	...
96868	187593	comedy	2018	davidleitch	juliandennison
96869	187593	sci-fi	2018	davidleitch	ryanreynolds
96870	187593	sci-fi	2018	davidleitch	joshbrolin
96871	187593	sci-fi	2018	davidleitch	morenabaccarin
96872	187593	sci-fi	2018	davidleitch	juliandennison

Figure 2: featurised movie data

- Here the movieId is not a feature but will be useful later in fetching related data.
- The remaining features will be undergoing categorical encoding for faster processing.
- The target for these will the rating the user has given. So our goal is to predict the rating that a user will be giving based on the ratings he has given to movies he has watched and then recommend him the top predicted rating movies.
- Unlike in item based collaborative system here we will be needing to mention the USER_ID of the user as input.
- As there are rating ranging from 0.5 - 5 with separation of 0.5 there are too many classes. Hence when ever we will be using a classifier we will first apply ciel to rating. If its a regression then there is nothing to worry about.

2 TRAIN-TEST DATA SPLITTING

- Based on the userId we will be getting the movies which the user has watched and those which are yet to be.
- For the training data we will just be adding the ratings as target and leave the testing data to be no target in the form of dataframes later converted to numpy arrays.

genres	Year	Director	Stars	target
16	59	920	129	3.0
1	59	852	1643	3.0
14	61	21	37	4.0
16	60	387	926	5.0
5	61	860	373	4.0

Figure 3: training data for userId = 6

movieId	genres	Year	Director	Stars
1921	16	64	176	2870
2640	0	44	756	2492
1288	4	50	798	2160
118696	8	80	715	2503
1214	10	45	769	3713

Figure 4: testing data for userId = 6

3 TRAINING MODEL AND RECOMMENDING

3.1 KNN

1. First we will calculate user similarity based on movie ratings. Meaning consider the movies to be dimensions and the ratings to be the values on those axes.
2. Here we will be using cosine similarity. Then we can consider top 20 similar user (neighbors).
3. Then average the ratings given by neighbors on testing data.
4. Finally recommend the top 5 movies.

userId	1	4	6	7	10	11	15	16	17	18
userId										
1	1.000000	0.222592	0.154845	0.172873	0.020610	0.143281	0.171103	0.181110	0.277343	0.247073
4	0.222592	1.000000	0.104152	0.130424	0.039595	0.061612	0.079262	0.182762	0.158323	0.147400
6	0.154845	0.104152	1.000000	0.090671	0.027330	0.261448	0.078993	0.057677	0.104441	0.145904
7	0.172873	0.130424	0.090671	1.000000	0.156559	0.208292	0.242816	0.167518	0.255873	0.310400
10	0.020610	0.039595	0.027330	0.156559	1.000000	0.032733	0.127905	0.116169	0.084233	0.146908
...	...	...	...	...	...	...	...	...	...	...
605	0.173354	0.105924	0.138146	0.258716	0.130196	0.125683	0.153551	0.095777	0.199894	0.203548
606	0.227410	0.267954	0.147205	0.274294	0.131048	0.100483	0.238728	0.255188	0.304728	0.354333
607	0.294444	0.151207	0.191082	0.203150	0.012793	0.292494	0.183870	0.115673	0.243977	0.220684
608	0.320707	0.171938	0.214309	0.352245	0.095104	0.179816	0.256767	0.196047	0.238493	0.380322
610	0.196737	0.146852	0.078277	0.257108	0.181875	0.111424	0.319280	0.257147	0.297043	0.498099

Figure 5: user similarities

3.2 NAIVE BAYES

1. Instantiate a GaussianNB classifier and fit the model.
2. Store predictions, probabilities related to each class.
3. Gather the instances of the same movie and average out the probabilities of each class and finally classify the movie.
4. It is possible that user has no interest in giving 5 to any movie then in that case you have to consider top 4 rated.
5. Instead of ratings here we recommend based on probabilities.

```
array([[6.39936841e-002, 2.51177186e-001, 1.15958269e-001,
        7.85052482e-002, 4.90365612e-001],
       [6.47377307e-002, 2.50374124e-001, 1.15951309e-001,
        7.86218260e-002, 4.90315011e-001],
       [1.29538881e-001, 2.71519534e-001, 1.09452054e-001,
        1.06159046e-001, 3.83330485e-001],
       ...,
       [0.00000000e+000, 5.04251123e-112, 5.03372805e-001,
        4.96627195e-001, 1.66103067e-011],
       [0.00000000e+000, 5.09874794e-112, 5.29330587e-001,
        4.70669413e-001, 1.84453732e-011],
       [0.00000000e+000, 5.04125293e-112, 5.04769539e-001,
        4.95230461e-001, 1.67084030e-011]])
```

Figure 6: probability arrays

- The predicted rating here need to accurately be the predicted class as we are not considering the weights of other classes.
- So rating will be the weighted average over all classes.

3.3 SVM

1. Instantiate a SVC classifier and fit the model with probability=True.
2. Store predictions, probabilities related to each class.
3. Gather the instances of the same movie and average out the probabilities of each class and finally classify the movie.
4. It is possible that user has no interest in giving 5 to any movie then in that case you have to consider top 4 rated.
5. Instead of ratings here we recommend based on probabilities.

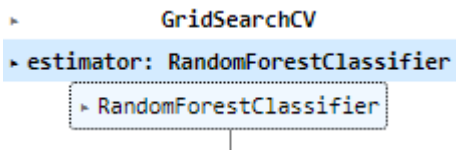
```
array([[0.01580483, 0.04872667, 0.35499562, 0.27764465, 0.30282822],
       [0.0157861 , 0.04704475, 0.34908654, 0.27820137, 0.30988123],
       [0.011533  , 0.048145  , 0.34800223, 0.38625831, 0.20606146],
       ...,
       [0.02131059, 0.04909425, 0.38890476, 0.32788792, 0.21280248],
       [0.02877056, 0.04456411, 0.36477521, 0.32523984, 0.23665028],
       [0.02191807, 0.04889781, 0.38921074, 0.32711248, 0.2128609 ]])
```

Figure 7: probability arrays

- The predicted rating here need to accurately be the predicted class as we are not considering the weights of other classes.
- So rating will be the weighted average over all classes.

3.4 Random Forest

1. Instantiate a RandomForestClassifier and fit.
2. We will be doing hyperparameter tuning using GridSearchCV.



3. Parameters being :
   - 'max_depth': [2,3,5,10,20]
   - 'min_samples_leaf': [5,10,20,50,100,200]
   - 'n_estimators': [10,25,30,50,100,200]
4. For userId = 6 the grid_search.best_score_ = 0.45317170777556526, max_depth = 20, min_samples_leaf = 5, n_estimators = 10.

```
RandomForestClassifier
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=10,
                       n_jobs=-1, random_state=42)
```

5. predict the classes of test data and get any 5 top rated movies.

3.5 LINEAR REGRESSION

1. Instantiate a LinearRegression and fit.
2. Here we just need to take average of same movie instances and recommend top 5.
3. It is possible that a prediction may go more than 5. It just means it has very high rating prediction.

```
[(86882, 3.934078513549464),
 (142488, 3.931624313823602),
 (109374, 3.922640855860871),
 (88405, 3.921094906048568),
 (122904, 3.918952784605416)]
```

Figure 8: top 5 regressions for userId = 6

3.6 BAG OF WORDS

1. This is the only item based system which requires movie name rather than userId.
2. We will be using the data mentioned in mid-report.
3. First we will get rid of all the movies which have no genre, no director, no keyword or proper year.
4. Then combine all the columns into a single content column.
5. One interesting thing I have done here is using **stemming** and **lemmatization** methods to make similarity checking even more meaningful and easier.
6. Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.
7. Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma.
8. percentage difference : 0.609115731989078 between stemming and lemmatization methods
9. Considering the top 5000 words in all the contents of movies we will using the **text vectorisation** and convert each movie into a vector.
10. Late we will use cosine similarity to get 5 most closest to a movie of our choice.
11. The more close a movie is its expected similarity / rating is high as well.