Movie Recommendation System

Jithin Joyson

CS 578

Problem

Since the dawn of time, humans have been plagued with decisions to make that will either break and make their day. From the remedial things such deciding what color underwear to wear to hygienic choices that impact people around them to survival decisions such sleep and food these decisions form how we go about in life. While some may be for the betterment of the society or their own health, less consequential choices are weighed by the same metric. It simply takes too much time to make choices and little bit longer to make the “right” choices. If the time wasted is not the worst part of this process, it will be the regret of making the wrong choice with that wasted time. As we have moved into the modern age, these choices have become more frequent with the choice space increasing exponentially. It seems predestined for humans to struggle with choices to make.

Often paying close attention to the finer details go a long way in choices made. Having context of the decision at hand helps but having someone who have gone through your ordeal to speak about their results is the best. Thus, recommendations for choices have existed as long as choices have existed. For every choice, there has been someone recommending the idea to become a clear contender for your time; recommendations are the reason humans are plagued with this way of being. They are also the key to making safer and faster choices. Having an accountable resource to do the recommending can make the decision making a very easy process. But we have credited this position to people we trust without thinking they have their own tastes. Here lies the solution to our problem. The only person that knows you well is you. But extrapolating that data can take time especially when the choices are exponential. Therefore, we will take advantage of the computation power the present has to offer and couple it with our prior choices to create a recommendation system.

Movie Recommendation

Though choosing which movies to watch may not be at the same importance level as a survival level choice, it is a problem that wastes a lot time and has an exponential version space. This is also a field that many systems have been created for. But the consistency of these systems varies from the producer to producer. Furthermore, there is a reasoning behind some of the choices being recommend. Granted these choices may expand your tastes like I have experienced with the Netflix recommendation system, it would put to rest the regret of making a bad decision if some sort of transparency was built into the system. Another problem with current systems is the lack of discussion that can be held of your choices. You make that one mistake to try a chick flick and now 1 in every 10 recommendation is a chick flick. Fixing the system of these mistakes are sometimes problematic since recommendations are not also being fixed. Therefore, an interactive aspect of the recommendation is also key in elevating the user experience.

Dataset

Several datasets were scavenged to find movies that had enough volume, user ratings and features to choose from. Contenders are:

1. Movie Lens
   A general-purpose dataset that holds many user ratings and reviews to serve as a foundation for recommendation systems. Movie sets range anywhere from 500 to 58,000 with tokens (keywords), genres and IMDB and TMDB ids. Richer datasets which have smaller volume give more information with respect to the users' background to create interesting correlations for models.

2. IMDB
   API based dataset that has request limits but up to date data with regard to movies. Often a hassle to make requests real time due to limits and must be used well in advanced to save and analyze meaningful data. Interestingly enough, it doesn't offer actors in a movie which is addressed further in the paper.

3. TMDB
   Another API based dataset that also has an annoying request limit that forces time managed data collection. Allows extraction of actors in a movie with a slight caveat; calls must be made with name of actor and existence of that actor in a given movie can only be confirmed by asking if the actor was present in the movie. This leads to a large number of requests to check if an actor in several movies (not the other way around unfortunately).

After several evaluations of the datasets, the Movie Lens dataset was chosen as the primary choice. This set allows for user rating flexibility while not constricting results to time. Scaling the model after test trails would be nearly impossible (unless the process began well in advance). Featurization of select attributes also drove the decision.

With that said, the latest and largest (ml-latest) dataset was used as the primary dataset. This set offered several data frames, but the selected frames were movies.csv (Figure 1), ratings.csv (Figure 2) and tags.csv (Figure 3).

movieId	title	genres
1	Toy Story (	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1	Adventure|Children|Fantasy
3	Grumpier (	Comedy|Romance
4	Waiting to	Comedy|Drama|Romance
5	Father of t	Comedy
6	Heat (1995	Action|Crime|Thriller
7	Sabrina (1	Comedy|Romance

Figure 1: Movies.csv

userId	movieId	rating	timestamp
1	307	3.5	1.26E+09
1	481	3.5	1.26E+09
1	1091	1.5	1.26E+09
1	1257	4.5	1.26E+09
1	1449	4.5	1.26E+09
1	1590	2.5	1.26E+09
1	1591	1.5	1.26E+09

Figure 2: Ratings.csv

userId	movieId	tag	timestamp
14	110	epic	1.44E+09
14	110	Medieval	1.44E+09
14	260	sci-fi	1.44E+09
14	260	space acti	1.44E+09
14	318	imdb top 2	1.44E+09
14	318	justice	1.44E+09
14	480	Dinosaurs	1.44E+09

Figure 3: Tags.csv

Features

Movie.csv and tags.csv were looked at closely to debate what features to make when creating a movie object. Definite green light was given to the genres attribute in Figure 1 since this was present in every single movie (no genre was a label) and there was a select range for this attribute. Thus, one hot encoding for this attribute resulted in 19 features.

The next ideal feature should be actors. Due to the limitations of the request calls when using the TMDB API and the work around to check a given actor is in a movie, it became quickly apparent that this feature was questionable. Another problem was the frequency of the feature. Movies that had lead actors that were not popular did not correlate in creating a good generalization for other movies. This is where Figure 3 helps. Tags.csv offers keywords for select movies. Some of these keywords are actors. This was a better way to sample actors into the features list since they are contributing to the decision of the movie while not creating a sparse representation of the data.

Now the tag.csv had many problems with it. Since these were user inputs, there was no standardization of the tags. Even proper names were subject to user error since spacing and special characters recreated the same feature in different light. Another option in the tag field was actual hashtags. These features were often in one cell of the csv and had to be split to create separate and unique meaning Thus, intensive data cleaning was done beforehand.

Finally, to force frequency of features such that movies could be subject to similarity tests in to the future was token frequency. Instead of being as aggressive as a tf-idf (token frequency – inverse document frequency) approach, features were subject to at least being frequent in 2 documents. Figure 4 entails the tokenization of the different tags. To add a final layer of generalization of features, special characters were removed from the features which created one-word features that were subject to repetition. This also allowed long duplicates such as ‘TOM hank’s’ and ‘Tom Hanks’ to normalize to ‘tomhanks’.

Figure 5 holds a transition from the initial genres to genre list to genre and token list to feature csr matrix.

```
def tokenize_tag(movies):
    toks = []
    for idx, row in movies.iterrows():
        try:
            tag = tags.query('movieId=='+str(row['movieId'])).iloc[0]['tag']
            for t in tag:
                t = t.lower();
                t = t.replace('-','')
                t = t.replace('/','')
                if len(t) > 0:
                    if t[0] == '#':
                        temp = t.split(' ')
                        for te in temp:
                            row['tokens'] = list(set(row['tokens'] + [str(te[1:])]))
                    else:
                        if len(t) > 0:
                            temp = tokenize_string(t)
                            temp2 = ''
                            for te in range(len(temp)):
                                temp2 += temp[te]

                            if not temp2.isdigit():
                                if not temp2 == '':
                                    row['tokens'] = list(set(row['tokens'] + [str(temp2)]))
        except:
            a = 1 #filler
        toks += [row['tokens']]
    movies['n_tokens'] = toks
    return movies
```

Figure 4: Tokenization

	movieId	title	genres	tokens	n_tokens	features
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	[adventure, animation, children, comedy, fantasy]	[rainydaywatchlist, want, friends, engaging, h...	(0, 0)\t2.8949294625059285\n (0, 1)\t2.5627...
1	2	Jumanji (1995)	Adventure|Children|Fantasy	[adventure, children, fantasy]	[rainydaywatchlist, animals, lebbat, comedy, s...	(0, 0)\t2.8949294625059285\n (0, 14)\t1.307...
2	3	Grumpier Old Men (1995)	Comedy|Romance	[comedy, romance]	[comedinhadevelhinhosengraãƒâada, comedy, good...	(0, 36)\t0.5548875094520286\n (0, 40)\t1.63...
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	[comedy, drama, romance]	[characters, drama, interracialrelationship, b...	(0, 36)\t0.5548875094520286\n (0, 40)\t1.63...
4	5	Father of the Bride Part II (1995)	Comedy	[comedy]	[gynecologist, baby, dianekeaton, pregnancy, c...	(0, 4)\t2.3785549086385926\n (0, 14)\t1.307...
5	6	Heat (1995)	Action|Crime|Thriller	[action, crime, thriller]	[want, whocaresdvds, electronicsoundtrack, rob...	(0, 1)\t2.5627640579164535\n (0, 40)\t1.630...
6	7	Sabrina (1995)	Comedy|Romance	[comedy, romance]	[fusion, remake, sexuality, love, comedy, panc	(0, 36)\t0.5548875094520286\n (0, 40)\t1.63

Figure 5: Features

Methodology

Now that the data has been fully processed and mined into ideal features, target classes and model choices need to be made to continue with the recommendation process. Since the Naïve Bayes and Logistic Regression classifiers were intensively discussed with regard to their transparency and interactivity, these classifiers were used along with slight tampering to the

parameters as the models for the recommendations. In the Naïve Bayes classifier, the prior parameter was (True or False) were tested and the L1 and L2 penalties were tested in the Logistic Regression classifier.

Before training the model, it is important to find a user in the Movie Lens dataset that had substantial amount of ratings. Ratings also needed to be normalized such that a binary classification can take for each movie. Finally, the learning method when the model is implemented over a huge test space needs to be ironed out such that the model learns the best movie object to improve its accuracy metric.

To start off, the user with the highest amount of ratings were learned. A threshold rating of 4 was used to judge if the movie was liked or disliked. Active learning was introduced with 3 different learning techniques: random sampling, uncertainty sampling and query by bagging. Figure 6 shows the accuracy metric as the movies were learned for user 199011 under the basic Logistic Regression model. Initially, a high accuracy on the model was good sign. But after further investigation, it was apparent there wasn't an equal distribution in the target class. Thus, the user 138483 was chosen who had 1000 recommendation which had 519 positives and 481 negatives.

Figure 6: Nonequal distribution of target class

Interaction

As interaction is one of the problems with current movie recommenders and a reason behind this project, it is important the model is mendable by the normal user. Thus, several steps were taken to ensure this feature.

Active Learning

Due to the repetition process of recreating the model each time with the user's input, it is important to ask the user right questions. Therefore, active learning was introduced into the model. Figure 7 shows both the Logistic Regression and Naïve Bayes base classifiers on the three different learning techniques.

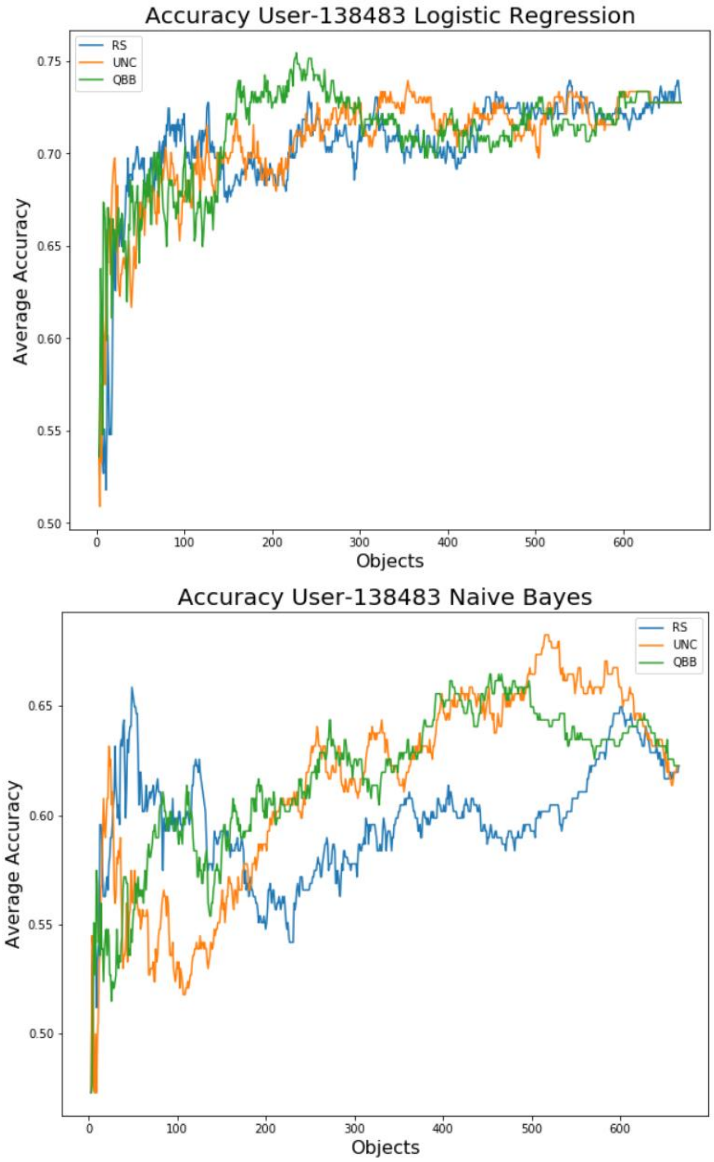

Figure 7: Accuracy Metric on Logistic Regression and Naïve Bayes

The accuracy has been steadily going up as the model is being improved in the Logistic Regression classifier in all 3 learning techniques which means we are going in the right direction. Query by bagging does very well after 150 objects or so before uncertainty sampling takes over and finally random sampling in the last 50 objects. The Naïve Bayes classifiers peaks early with random sampling and uncertainty sampling does all the way through. A concerning trend is the decline of accuracy as the more objects are added to the train set. This is an indicator that the classifier does not do well which will be further discussed in the evaluation section.

Drawing conclusions from the observations, it is assumed to start the learning with query by bagging, then switch to uncertainty sampling and end with random sampling after a select threshold. But the sample space of each method is jeopardized when combining all the techniques. Thus, query by bagging is the best learning technique if results have to be replicated since it dispenses accurate results in the beginning and makes up for its inaccuracies in the middle as a lot more objects are added. When it comes to movie recommendations, it is important to produce accurate results in the beginning such that the user invests in the system to produce more accurate results in the long run.

Ratings

Movie Lens has been a very rich dataset in the sense that it allows us to play with the rating threshold to be flexible to a user's taste. The rating attribute it offers is out of 5 which allows the user to draw the line when it comes to what ratings are defined as "best". This can be an added layer in the interaction portion where the user does not feel the recommendation system is aggressive enough with the recommendations and can move up or down as he pleases. The threshold is set to 4 initially since it is a good heuristic measure and created target class space that had almost equal distribution for the test user (138483).

User Input

Finally, the best way to interact with the user is through asking user feedback on recommendations. This is a straightforward act of interaction and can be confirmed to be an input without noise. Figure 8 shows the interaction step of just the Logistic Regression classifier.

```
Here are the top 10 recomendations for now
1. Rear Window (1954)
Negative Evidence: -2.1188104440126714 Positive Evidence: 12.83606546485242
What would you rate this movie? Out of 5. Input enter to skip
2. Vertigo (1958)
Negative Evidence: -4.839411599316044 Positive Evidence: 12.463522929912937
What would you rate this movie? Out of 5. Input enter to skip
3. Three Colors: Blue (Trois couleurs: Bleu) (1993)
Negative Evidence: -2.232796304506745 Positive Evidence: 8.597366563355775
What would you rate this movie? Out of 5. Input enter to skip
```

Figure 8: User Input

Transparency

The second problem we are trying to solve through this system is the transparency of the model to the users using it. Sure, users can interact with the model, but they do not understand why the recommendations they are asked to evaluate is being asked in the first place. Here lies the transparency attribute which classifier specific and explain different results is different ways.

Naïve Bayes

This classifier deals with probabilities of events and the best approach it to find the tokens the have the most influence on the object being classified as a positive and the least influence on the object being classified as a negative. Figure 9 shows the Naïve Bayes probabilities for feature importance. In context of movie recommendation, the positive class shows keywords you look for in a movie and negative class shows keywords you don't want in a movie.

```
Positive Class:

#       Probability             Feature #       Feature
-------------------------------------------------------------------------
1.      -29.277754813106732     13784           shannonelizabeth
2.      -29.277754813106732     18706           dominican
3.      -29.277754813106732     18705           digitalvideo
4.      -29.277754813106732     18704           bitch
5.      -29.277754813106732     18703           daycare
6.      -29.277754813106732     18702           kindergarten
7.      -29.277754813106732     18701           bahmanghobadi
8.      -29.277754813106732     18700           ethic
9.      -29.277754813106732     18699           6.4filmaffinity
10.     -29.277754813106732     18698           betterthanfirstpart

Negative Class:

#       Probability             Feature #       Feature
-------------------------------------------------------------------------
1.      -29.201718200046635     27568           modernsherlock
2.      -29.201718200046635     16784           techniquevoiceover
3.      -29.201718200046635     16783           talkingbaby
4.      -29.201718200046635     16782           abevigoda
5.      -29.201718200046635     16781           johnflynn
6.      -29.201718200046635     16780           blooddonation
7.      -29.201718200046635     16779           violenceinschools
8.      -29.201718200046635     16778           schoolkids
9.      -29.201718200046635     16777           obedience
10.     -29.201718200046635     16776           rules
```

Figure 9: Transparency Naïve Bayes (Prior = True)

Logistic Regression

Feature importance for a Logistic Regression classifier is contributed by the coefficients of each feature. This weight the classifier assigns to each feature based on the training process the model had to adhere to and useful information the user can use when learning about he classifier. Figure 10 shows the Logistic Regression coefficient values to deem if the feature is important in the classification process.

#	Coeficient	Feature #	Feature
1.	1.1277612705310878	213	tumeysdvds
2.	1.0506367376429124	389	bdr
3.	1.011557121321135	1785	directorialdebut
4.	-0.9752723500438648	218	soundtrack
5.	0.9321905091327999	80	nationalfilmregistry
6.	0.8601689643523025	229	ensemblecast
7.	0.8430152838093726	66	erlendsdvds
8.	0.7993657035820964	2174	irish
9.	0.7715151242716862	1551	criterion
10.	0.7638650182712202	524	oscarnomineebestpicture

Figure 9: Transparency Logistic Regression (No Penalty)

Once again, the features are keywords the user finds important when making choices. Users can learn more about themselves and the model as the recommendation are being made.

Another way to be transparent is in the recommendation process. Feature importance explains the keywords in the model, but evidence is needed to explain why a choice was made. Figure 8 shows the evidence assigned to the recommendation made by the Logistic Regression classifier. This takes into account the keywords discussed earlier and computes the final score for each class. The larger score points toward what class the movie is assigned and the difference in score ranking shows the ordering of the recommendation.

Evaluation

Whenever training a model it is important to evaluate the results by using a metric or user interaction as detailed in the section above. Since we are dealing with a recommender system, an error metric (RSME) would be the ideal way to measure this. But this is a process that evaluates the model (metrics like accuracy, precession, recall, f1, …) Figure 10. There needs to be a way to evaluate the recommendations as a whole such that it is both interactive and transparent to the user. Thus, the recommendations made by the classifiers will be compared to the basic recommendation systems that accurately recommends movies as a staple.

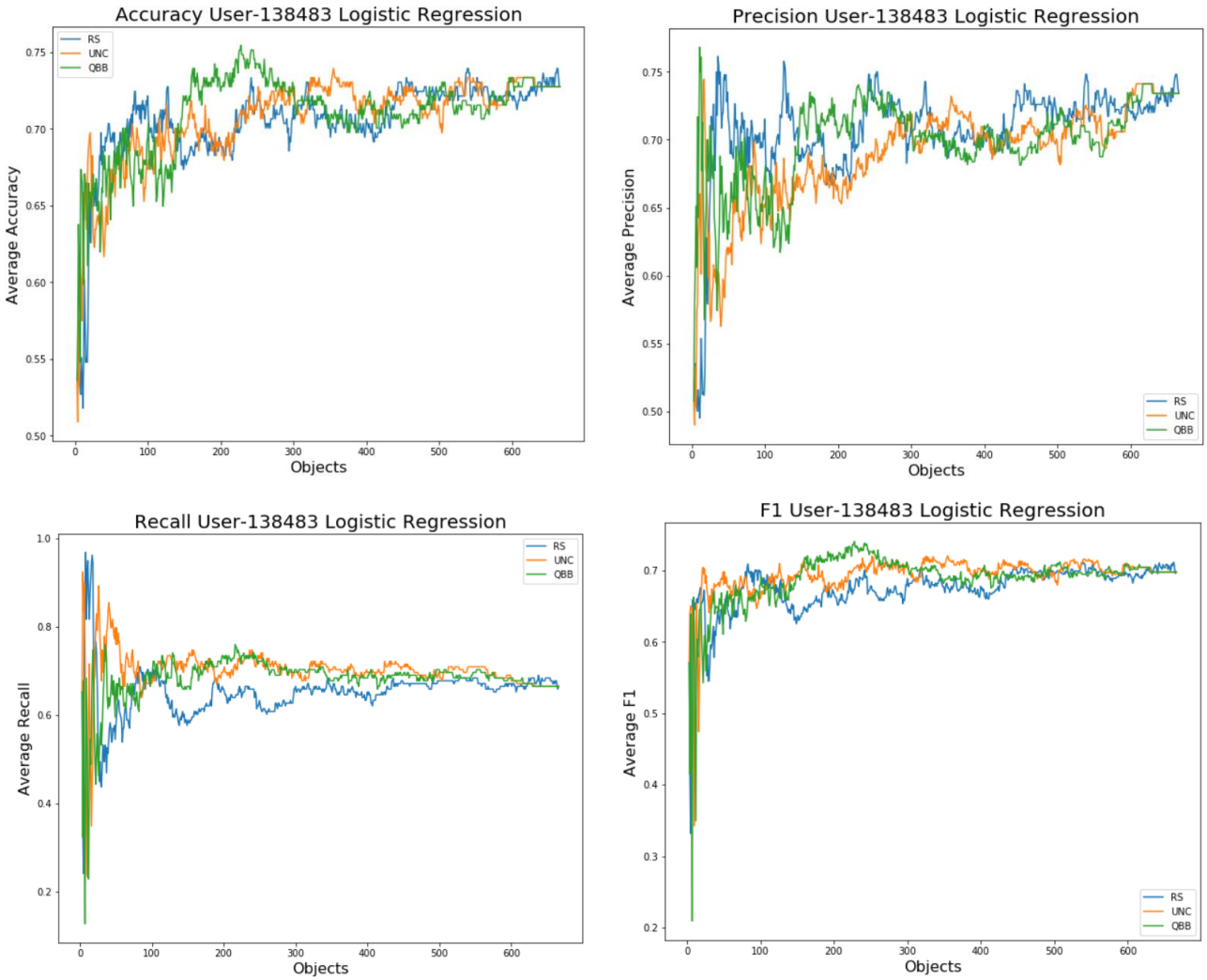

Figure 10: Base Logistic Regression Metrics

Content Based

This recommendation system looks at a given movie and compares that movie to all the other movies the user has watched. During the comparison, the cosine similarity of the movies is calculated (by transferring the features into TF-iDF scores Figure 10) and a recommendation is made based on the summation of the similarity and known ratings. Figure 10 shows the cosine similarity formula and the content-based recommendation calculations.

f_{ij} = frequency of term (feature) i in doc (item) j

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

n_i = number of docs that mention term i
N = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

TF-IDF score: $w_{ij} = TF_{ij} \times IDF_i$

$$\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$$

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

Figure 10: TF-iDF and cosine similarity ($\cos(w_x, w_y)$)

The recommendation scheme was used on user 138483 and top 10 movies were recommended. In the evaluation step, the recommendations were compared with the Logistic Regression models (Base, L1, L2) and the Naïve Bayes models (True and False priors). Figure 11 holds the results were the common movies are shown.

Logistic Regression (Base)

```
Chosen Movies
-------------------------------------------------
Rear Window (1954)
Eyes Without a Face (Yeux sans visage, Les) (1959)
```

Logistic Regression (L1)

```
Chosen Movies
-------------------------------------------------
It Happened One Night (1934)
Rear Window (1954)
Eyes Without a Face (Yeux sans visage, Les) (1959)
```

Logistic Regression (L2)

```
Chosen Movies
-------------------------------------------------
Rear Window (1954)
Eyes Without a Face (Yeux sans visage, Les) (1959)
```

Naive Bayes Prior = T

```
Chosen Movies
-------------------------------------------------
No Movies
```

Naive Bayes Prior = F

```
Chosen Movies
-------------------------------------------------
No Movies
```

Figure 11: Content Based Evaluation

Collaborated Filtering (Item-Item)

Collaborated filtering can be done in two cases were the comparison can be user to user or item to item. To keep the consistency of the content based, item-item filtering was used. In this system, movies are compared with other movies the user has watched but the ratings other users have given for the movie is used to calculate the similarity. The similarity vectors are normalized by the mean of the vectors and then cosine similarity is conducted. This is known as the Pearson correlation and the equations are present tin Figure 12.

- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users x and y

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

Figure 12: Pearson Similarity

Once again, the recommendation scheme was used on user 138483 and top 10 movies were recommended. In the evaluation step, the recommendations were compared with the Logistic Regression models (Base, L1, L2) and the Naïve Bayes models (True and False priors). Figure 13 holds the results were the common movies are shown.

Logistic Regression (Base)

```
Chosen Movies
--------------------------------------------------
Rear Window (1954)
Vertigo (1958)
Wild Bunch, The (1969)
```

Logistic Regression (L1)

```
Chosen Movies
--------------------------------------------------
Rear Window (1954)
Vertigo (1958)
It Happened One Night (1934)
Wild Bunch, The (1969)
```

Logistic Regression (L2)

```
Chosen Movies
--------------------------------------------------
Rear Window (1954)
Vertigo (1958)
Wild Bunch, The (1969)
```

Naive Bayes Prior = T

```
Chosen Movies
--------------------------------------------------
No Movies
```

Naive Bayes Prior = F

```
Chosen Movies
--------------------------------------------------
No Movies
```

Figure 13: Collaborated Filtering Evaluation

It is important to understand the dataset held about 58,000 movies and around 27,000,000 ratings. To reduce the space complexity, only the first 1000 movies were used as comparison during the collaborated filtering. Furthermore, movies were evaluated until a movie that met the threshold rating proposed by the user was produced. Since the computations are static for the similarity matrix, all the similarities will be calculated and saved when fully implemented such that the best results are produced.

Conclusion

To conclude on a classifier, several tests that the evaluation step proposed were done on the recommendations. As seen in Figure 7, the Logistic Regression classifier outputted higher accuracy scores. These scores were replicated in the other metrics (located in the folder) and they were more consistent. Furthermore, moving towards the recommendation evaluation, the results proposed by Logistic Regression were present in both content based (~3/10) and collaborative filtering (~4/10). Naïve Bayes classifiers both had underwhelming outputs were none of their recommendations were in common Figure 11 and Figure 13.

The Logistic Regression classifier were tampered with for the penalty attribute when making its recommendations. Comparisons with the staple classifiers produced Logistic Regression with L1 penalty as the clear winner since it predicted 1 more movie in common than the rest of the classifiers in its track. Furthermore, the L1 classifier has a special feature in which it drops weights of features to 0 as a result of penalty. Figure 14 shows the non-zero weighted feature counts (features the contribute to the classification). This approach normalizes noisy features in the model and produces substantial results. Looking forward, the C value of these classifiers will be played with to get a better classifier that does not under shoot the feature counts.

Logistic Regression (Base)
None zero features: 11502

Logistic Regression (L1)
None zero features: 324

Logistic Regression (L2)
None zero features: 11502

Figure 14: Feature Counts