Recommendation Engine for ‘Life Altering Films’

Capstone project report for *Springboard* - *Data Science Intensive*

By Justin Barton

August 2016

PROJECT BRIEF

I want to find films for people that will: inspire them, challenge their view of the world, make them think and make them feel – 'Life Altering Films' (LAF).

INTRODUCTION

Many who seek LAFs typically suffer a few pitfalls:

1. Spend a lot of time searching through film websites, film databases, review sites, recommendation sites, and watching trailers.
2. Even with the most rigorous search and recommendation tools available, they often come back with a list of films to watch, the vast majority of which are not LAFs, and sometimes, not even 'good' at all.
3. Given the barriers to finding LAFs, it is likely many miss the opportunity to watch these LAFs. This is detrimental to both the filmmakers and the population – the filmmaker cannot spread their message, and the user is less likely to challenge their view of the world.

Recommendation engines have come a long way in the last decade, but I have yet to find any that reliably find LAFs for myself (similarly with many other film lovers I've spoken with). I suspect recommendation engines suffer common drawbacks:

1. Are driven by commercial incentive and will recommend films you are likely to be persuaded to watch, which may or may not be LAFs. This is a distinctly different to goal to connecting people to their LAFs.
2. Are not hand crafted enough to the specific user. They are tackling the problem first at scale, and trying to get as many recommendations for as many people as possible. User taste is a lot more subjective than is common opinion, which requires a lot more analysis at the user level.
3. Following on from the above, the difficulty of getting enough previously rated films from new users, from which to make predictions of their LAFs. This is working on the assumption that more data, typically trumps a slightly more tuned recommendation engine.
4. The challenge to getting more user data is gamifying the experience, and also feeding users the films they are likely to have seen (instead of vastness of all films of all time).

In this approach, the question here isn't 'can we create a near perfect model that it will recommend accurately?', but merely 'how much data (previously reviewed films) is required, such that a basic model, whose features are hand-crafted for the new user, will predict reasonably accurately?'

The technical approach to solving this problem is broken down in the following parts:

- Feature Engineering
  - The datasets
  - Merging the datasets & cleaning the data
  - Creating surveys for new users
  - Generating profiles for new users
  - Creating feature tables
- Creating Recommendations
  - Linear regression on film attributes – per new user
  - Linear regression on old user ratings – per new user
  - Finding old users with similar ratings to new users
- Results
- Future of the project

FEATURE ENGINEERING

The datasets

Three publicly available data sets were used. They and their fields of interest are shown below:

MovieLens – historical user ratings (films previously rated by 1000s of other users)

OMDB – budget, revenue, runtime, job titles (Director, Producer, Editor etc.)

IMDB (from pycon) – Actors (actor name, and # ranking in credits)

Across all 3 datasets, each has a unique film id, and each have the film's title and year of production.

I explored all 3 datasets initially to see what aspects might be useful, and all 3 were chosen as they each had unique information that could be used as features. The final list of features prepared were:

- year
- set of {historical user ratings}
- set of {cast/crew from all job titles}
- budget
- revenue
- runtime
- num_ratings
- average_rating

Merging the datasets & Cleaning the data

Given that none of the films had been linked across datasets, it was established that the best way to merge the 3 datasets was by using the film's title and year. The difficulty of merging came about from various incongruences in the ways the film's title and year were indentified for the same films across datasets. Some examples were:

1. *'City of God (Cidade de Deus)'* in MovieLens and *'City of God'* in OMDB
2. *'Birdman'* in MovieLens and OMDB and *'Birdman or (The Unexpected Virtue of Ignorance)'* in IMDB
3. *'V for Vendetta'* is a 2006 movie in MovieLens and OMDB, but its a 2005 movie in IMDB
4. *'Usual Suspects, The (1995)'* in MovieLens is *'The Usual Suspects'* (with year separated) in IMDB & OMDB
5. OMDB had dates stored as dd-mm-yy and MovieLens & IMDB as (yyyy)

Some experiments were undertaken to try to minimise these incongruences, however given the scope of the challenge to find and model them all, not all were corrected (incongruences of type 4 & 5 were fixed).

When all 3 data sets had been linked (despite some title and year issues remaining) there were 8832 films to use in extracting the features. In the future it is hoped to improve the film title matching algorithms to have a greater pool of films.

Creating surveys for new users

In order to find LAFs for users, I needed to capture their previous preferences through a rating system. It would be infeasible to have each user rate 8832 different films, as this would take a long time, and there is high likelihood that many of the films haven't been watched. To create a feasible list I experimented with some filtering parameters, and finally settled upon, films that:

- Were made on or after 1965
- Had English language
- Had at least 10 previous user ratings

I then sorted this list by average historical user rating, then by number of ratings. From this sorted list, I took the top 1000 films. Then for the number of new users ~ 10, I printed for each, an excel sheet with the top 500 films randomised, followed by the bottom 500 (of the top 1000) randomised. Each of these randomised sheets was then imported into a shared google sheet for each group of new users (ie friends from Barker, family etc), with a separate tab for each user. Here is an example for my own data:

https://docs.google.com/spreadsheets/d/1Vl6vStEdKToK3AGdHcN1mym39q0cpXFWgLZGlgt8I9A/edit#gid=554351319

See column 'out of 10' of sheet 'Reviewed' with my personal film ratings. (blank), '-' or '-1' denoting a film not seen.

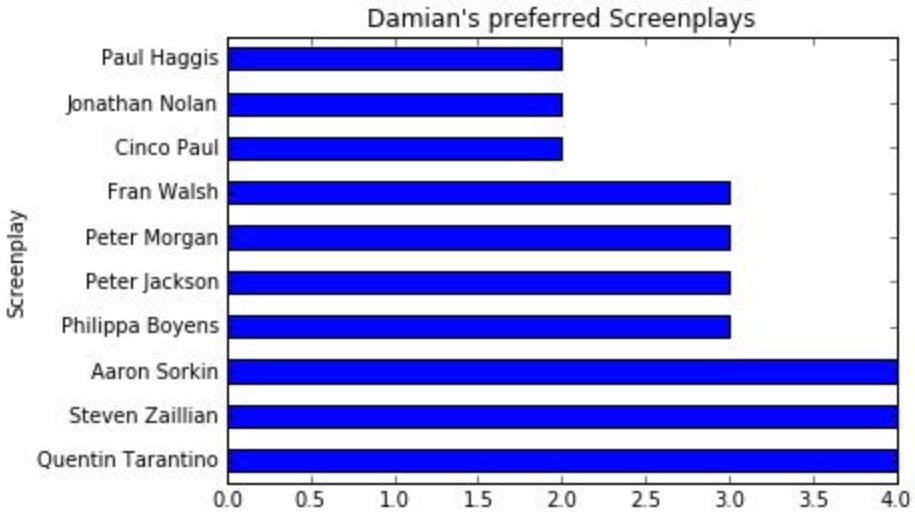
It was important to have the film titles randomised to remove bias, say for example if the films were sorted by highest rated first. Films the like e.g. Shawshank Redemption are well known to rate highly, this could alter one's rating of those higher on the list.

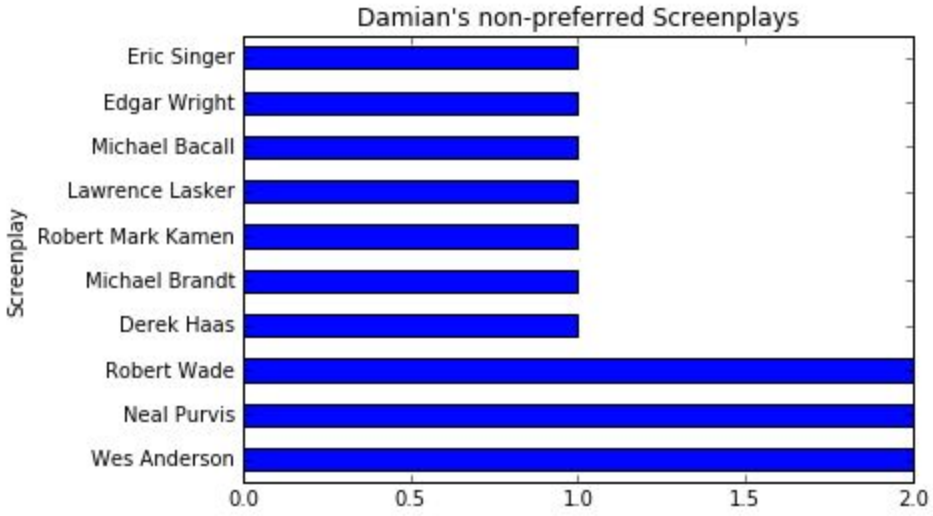
Connecting pandas to Google Sheets proved troublesome, as it was both slow and not fully documented on anywhere for the current version of pandas and google authentication. I had to experiment and combine a number of different resources and code examples to read new user data from Google Sheets. This was an important step as it meant the task was much less burdensome for

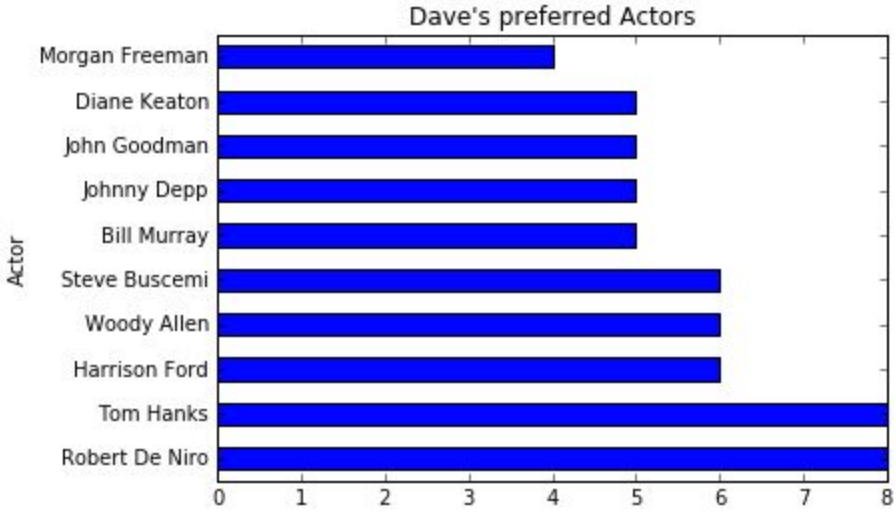
the users otherwise having to use excel and saving and email versions of complete/incomplete files. Once reviewed, references of dataframes for each new user's film ratings were read into a 2D dictionary of the format: users[sheet_name][user_name].

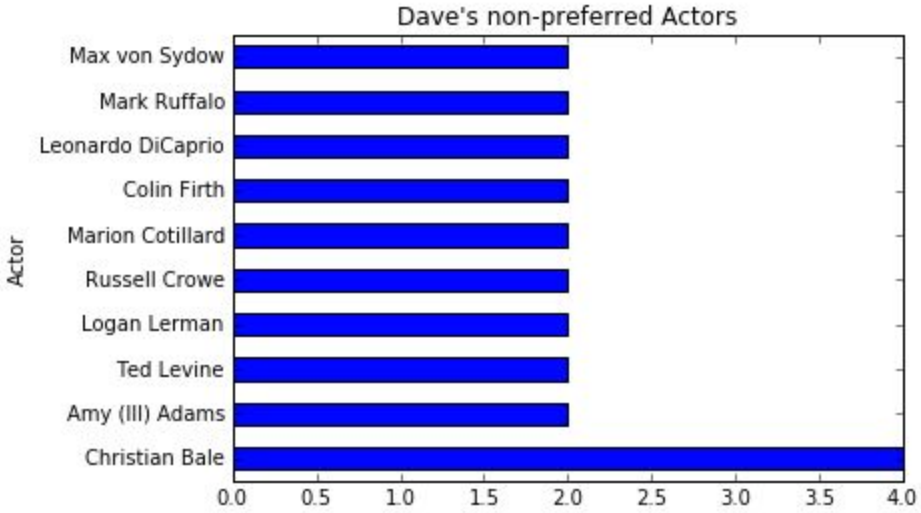
Generating profiles for new users

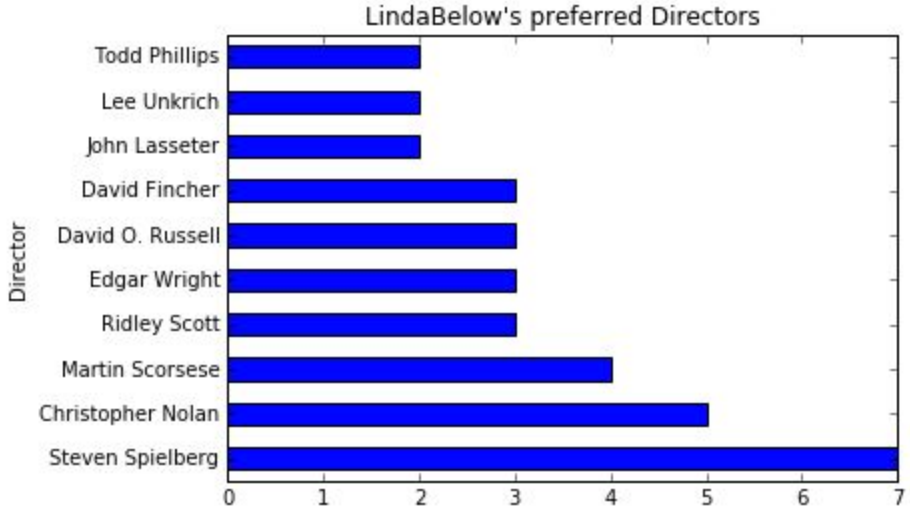
To get a sense for what information would be most relevant to the specific new user, and to make the recommendations more customised, I attempted to create a profile for each new user. First was to create a model of the features to be used, as features came in all shapes and sizes: stored in different dataframes, continuous/discrete, features spread over 2 columns, features with a vast number of instances (e.g. Actors). A dictionary of features structure was created taking into account these nuances, and then a function `add_features(df, features)` was created, which would add any combination features (excluding historical user ratings) to a dataframe dynamically.
Once these features were added, I wanted to visualise which features were most prominent for each user, this was through function `graph_top_discrete_features(sheet_name, user_name, feature, max_disp_results, bad_films)` . This function also called another function sort_by_feature_count. Here are some of the results:

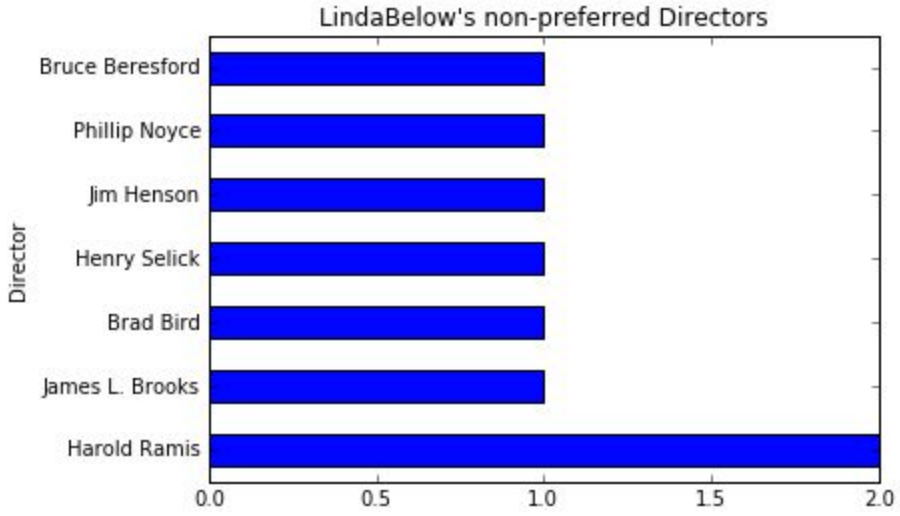

Damian's preferred Screenplays
Screenplay
Paul Haggis
Jonathan Nolan
Cinco Paul
Fran Walsh
Peter Morgan
Peter Jackson
Philippa Boyens
Aaron Sorkin
Steven Zaillian
Quentin Tarantino
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Damian's non-preferred Screenplays
Screenplay
Eric Singer
Edgar Wright
Michael Bacall
Lawrence Lasker
Robert Mark Kamen
Michael Brandt
Derek Haas
Robert Wade
Neal Purvis
Wes Anderson
0.0
0.5
1.0
1.5
2.0


Dave's preferred Actors
Actor
Morgan Freeman
Diane Keaton
John Goodman
Johnny Depp
Bill Murray
Steve Buscemi
Woody Allen
Harrison Ford
Tom Hanks
Robert De Niro
0
1
2
3
4
5
6
7
8


Dave's non-preferred Actors
Actor
Max von Sydow
Mark Ruffalo
Leonardo DiCaprio
Colin Firth
Marion Cotillard
Russell Crowe
Logan Lerman
Ted Levine
Amy (III) Adams
Christian Bale
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


LindaBelow's preferred Directors
Director
Todd Phillips
Lee Unkrich
John Lasseter
David Fincher
David O. Russell
Edgar Wright
Ridley Scott
Martin Scorsese
Christopher Nolan
Steven Spielberg
0
1
2
3
4
5
6
7


LindaBelow's non-preferred Directors
Director
Bruce Beresford
Phillip Noyce
Jim Henson
Henry Selick
Brad Bird
James L. Brooks
Harold Ramis
0.0
0.5
1.0
1.5
2.0

CREATING RECOMMENDATIONS:

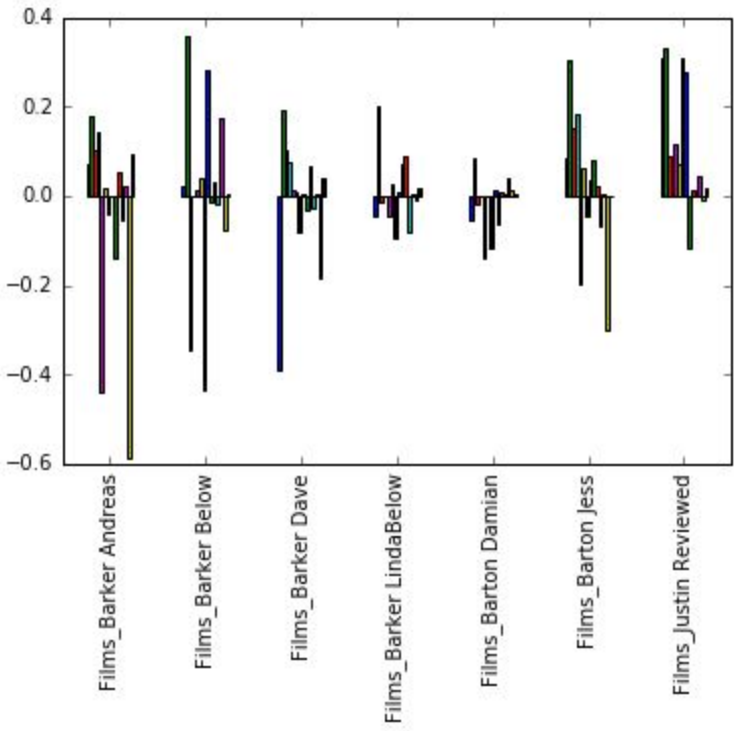
Linear regression on film attributes – per new user.

Attempting to create predictions was started by looking at some continuous features:

- budget
- revenue
- runtime
- num_ratings
- average_rating

For each new user, I ran a linear regression model on each feature separately, then all features together. Additionally for each model, data was split into training and test sets for cross validation. Function `add_features` as discussed above was used here to generate the results, which were added to this table, and graphed below:

	Films_Barker Andreas	Films_Barker Below	Films_Barker Dave	Films_Barker LindaBelow	Films_Barton Damian	Films_Barton Jess	Films_Justin Reviewed
All_feats_R^2_Test	0.072815	0.022327	-0.388948	-0.042749	-0.055600	0.084337	0.310399
All_feats_R^2_Train	0.177296	0.357523	0.193866	0.203595	0.084781	0.302849	0.330246
Films_rated	51.000000	154.000000	331.000000	189.000000	263.000000	107.000000	657.000000
budget_R^2_Test	0.102802	-0.346707	0.102257	-0.011296	-0.017850	0.150916	0.087844
budget_R^2_Train	0.143881	0.000038	0.076786	0.000943	0.000637	0.185917	0.066561
num_ratings_R^2_Test	-0.439209	0.011847	0.012464	-0.046410	-0.139055	-0.197996	0.116308
num_ratings_R^2_Train	0.015780	0.041469	0.009665	0.028631	0.001529	0.061344	0.072530
rating_R^2_Test	-0.039717	-0.433138	-0.082072	-0.091975	-0.117211	-0.045790	0.309029
rating_R^2_Train	0.000559	0.280157	0.006475	0.009824	0.012547	0.034182	0.279770
revenue_R^2_Test	-0.136811	-0.015461	-0.031494	0.070669	-0.062199	0.082619	-0.115738
revenue_R^2_Train	0.053273	0.030761	0.066343	0.090529	0.006923	0.023592	0.014501
runtime_R^2_Test	-0.055114	-0.018808	-0.027157	-0.078824	0.005122	-0.066380	-0.000505
runtime_R^2_Train	0.024358	0.174816	0.002878	0.005865	0.040640	0.002906	0.046204
year_R^2_Test	-0.587887	-0.076100	-0.182196	-0.008378	0.014322	-0.301905	-0.007043
year_R^2_Train	0.095605	0.005002	0.041145	0.016562	0.004580	0.000752	0.016598



Unfortunately there were no significant results that came out of this as none of the model's test R^2 (performance on future data) were even above 50%. This was including myself, who looked at over 6000 films to rate 657 of those, hence having 657 data points. Perhaps with more data, there might have been some patterns found, but it is probably infeasible to have users offer more ratings. Results may also improve with Principal Component Analysis (PCA). However based on my intuitive understanding, if PCA were to help, we'd expect to see some correlation between single feature models, which wasn't the case.

Linear regression on old user ratings – per new user

The next angle, was to try to find linear relationships between each new user, and historical user's ratings. For this, I created a few functions to deal with all new and old users ratings dynamically:

`find_hist_users(sheet_name, user_name, only_match_good, max_num_users, min_common_films)` - find all historical users, who have rated the same films as the new user

`create_hist_user_features(hist_users_with_new, max_features)` - convert `hist_users_with_new` into a user features dataframe.

`pivot_and_keep_cols(df, values, index, columns, keep_cols = [], dropna = True)` - make each userId a column name, with said historical user's film ratings.

`run_linreg_hist_users(sheet_name, user_name, max_num_users, only_match_good, results)` - run multiple linear regression models for each new user, on old users ratings as features

Using the above functions, I ran linear regressions models for all users, against previous users, tweaking both:

- The number of users to use as features (1, 2, 5, 10, 20, 50, 100, 200, 500), and
- either match on good films or not, defined by films with rating at least good_films_threshold = 7.

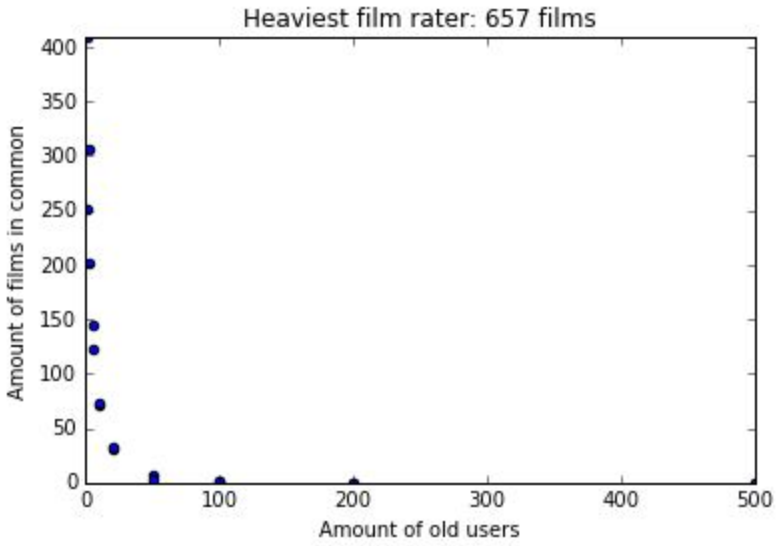
An excerpt of the results table looked like this:

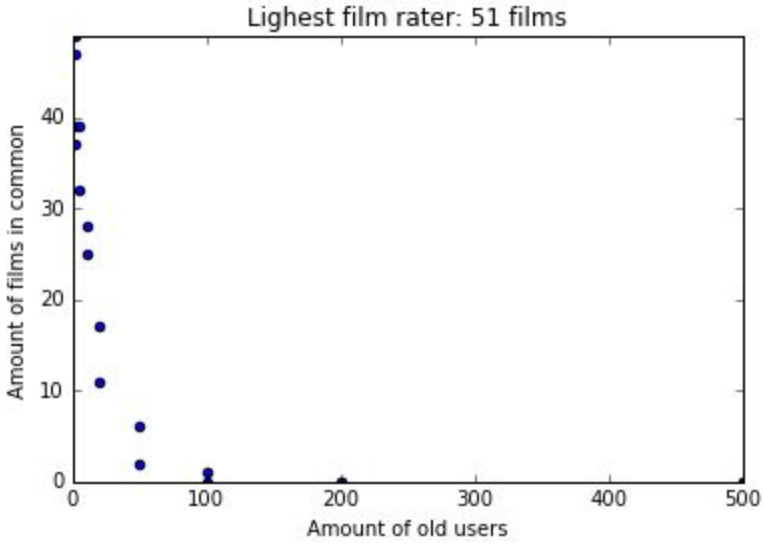
	User	Amount rated by user	Amount of old users	Amount of films in common	Good films only	R^2_Train	R^2_Test
0	Films_Barker LindaBelow	189	1	130	True	0.029359	0.031410
1	Films_Barker LindaBelow	189	1	179	False	0.000076	-0.003335
2	Films_Barker LindaBelow	189	2	111	True	0.025492	0.114096
3	Films_Barker LindaBelow	189	2	170	False	0.017230	-0.104352
4	Films_Barker LindaBelow	189	5	78	True	0.039774	0.016383
5	Films_Barker LindaBelow	189	5	150	False	0.062413	-0.072245

The R^2 results for each are summarised below. Some were quite high (perhaps overfit), and others were wildly inaccurate. In all, the results although inconsistent for different models, showed some indication (in some of the models) that there are similar user raters out there. See excerpt below:

			R^2_Test	R^2_Train
User	**Amount rated by user**			
Films_Barker Andreas	**51**	**count**	12.000000	12.000000
		mean	-2.646864	0.500902
		std	4.890542	0.422835
		min	-16.853163	0.044251
		25%	NaN	NaN
		50%	NaN	NaN
		75%	NaN	NaN
		max	1.000000	1.000000
Films_Barker Below	**154**	**count**	14.000000	14.000000
		mean	-0.672296	0.468655
		std	1.321154	0.435006
		min	-4.556207	0.006015
		25%	NaN	NaN
		50%	NaN	NaN
		75%	NaN	NaN
		max	0.165805	1.000000
Films_Barker Dave	**331**	**count**	12.000000	12.000000
		mean	-0.049265	0.319033
		std	0.375584	0.381777
		min	-0.510247	0.000824
		25%	NaN	NaN
		50%	NaN	NaN
		75%	NaN	NaN
		max	0.958468	1.000000

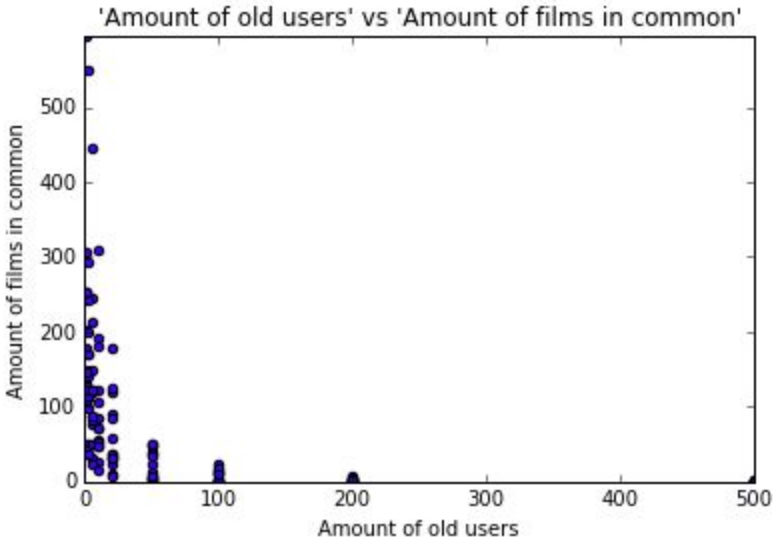
Upon analysing these results further, it became evident that those new users who had rated more films, had a substantially higher pool of old raters from which to find similar raters. See below a graph of ‘amount of old users’ vs ‘amount of films in common’ for both the heaviest new user rater, and the lightest new user rater:

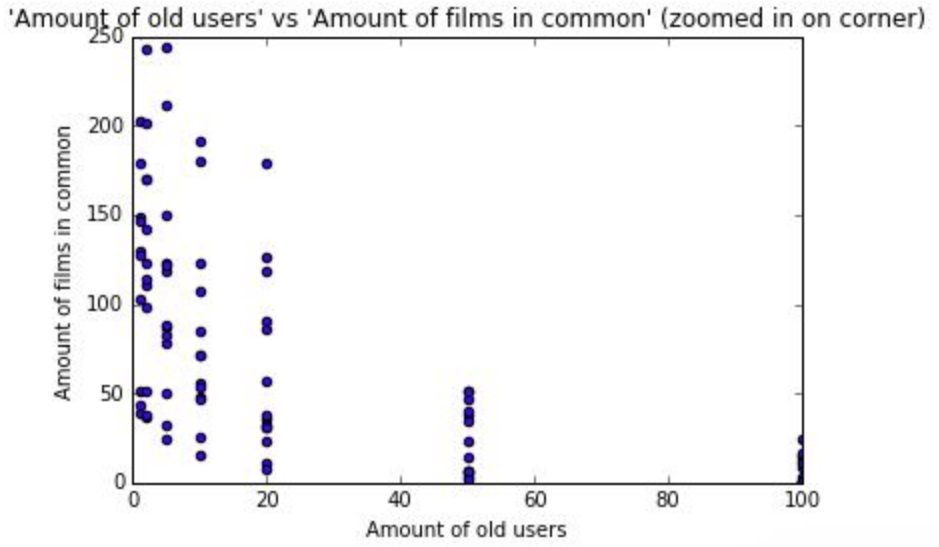




As seen on the graphs, the sweet spot (the knee) of having a both a large number of both of (‘Amount of old users’, ‘Amount of films in common’) was around: (50, 50) for heaviest film rater, compared with (50, 5) for the lightest film rater. The conclusion here is that, with the more films a new user rates, there is significantly more opportunity to find data (films both users have rated) that could be used to create a model comparing the two.

When comparing a results from all new users of ‘Amount of old users’ & ‘Amount of films in common’, the results follow a similar locus:





Finding old users with similar ratings to new users

A different, and perhaps more exploratory alternative to recommending came about by testing the theory that there exists similar users to the new users. In order to find similar historical (old) users to new users, there had to be some kind of scoring system/metric to do so. This requires first finding old users who have rated the same films as the new user, in order to generate the results from the scoring system. The algorithm created is described below:

1. Find old users who have rated the same films
2. For each old rater, with each new rater, reduce the list to films only both parties have rated.
3. Of this list calculate the mean & standard deviation (std) for each of the old, and new user.
4. Compare the new (mean, std) with old (mean, std), then order the by the ratio difference in std, and then by ratio difference in mean.
5. Then remove those old users in the list whose difference in mean and std exceed a certain tolerance (by default 10%)

This was achieved by the following function that was created:

```
find_similar_hist_users(sheet_name, user_name, max_num_users,
only_match_good, tolerance, similiar_users)
```

This function was called in a loop that iterrated over all new users. After various attempts, it was discovered that the processing and memory constraints caused the program to crash. I optimised this by converting one of the 1.7GB dataframes into a 670MB data frame, which helped slightly. Then instead of calculating the results for all users, I tested only on myself (since I had the most data). Additionally I had to cut the maximum number of old users to search for to 200. See results below for similar users to myself. 'out of 10' column denotes new user rating (in this case me), and 'userId' column denotes old user's rating as per row index:

	count		mean		std		min		25%		50%		75%		max		mean	std
userId	**out of 10**	**userId**	**out of 10**	**userId**	**out of 10**	**userId**	**out of 10**	**userId**	**out of 10**	**userId**	**out of 10**	**userId**	**out of 10**	**userId**	**out of 10**	**userId**	**diff**	**diff**
2669	184.0	184.0	7.413043	7.869565	0.594183	0.639627	7.0	7.0	7.0	7.0	7.0	8.0	8.0	8.0	9.0	10.0	0.061584	0.076483
33575	162.0	162.0	7.456790	7.728395	0.611472	0.610719	7.0	7.0	7.0	7.0	7.0	8.0	8.0	8.0	9.0	10.0	0.036424	0.001231
52260	165.0	165.0	7.393939	7.606061	0.611844	0.650487	7.0	7.0	7.0	7.0	7.0	8.0	8.0	8.0	9.0	10.0	0.028689	0.063159
91349	158.0	158.0	7.455696	7.683544	0.624295	0.608733	7.0	7.0	7.0	7.0	7.0	8.0	8.0	8.0	9.0	9.0	0.030560	0.024928
110847	178.0	178.0	7.320225	7.898876	0.556153	0.563383	7.0	7.0	7.0	8.0	7.0	8.0	8.0	8.0	9.0	10.0	0.079048	0.012999
114270	157.0	157.0	7.369427	7.675159	0.602203	0.612022	7.0	7.0	7.0	7.0	7.0	8.0	8.0	8.0	9.0	10.0	0.041487	0.016305
118205	209.0	209.0	7.363636	7.928230	0.581624	0.604166	7.0	7.0	7.0	8.0	7.0	8.0	8.0	8.0	9.0	10.0	0.076673	0.038757

In the future, I'm looking to optimise this algorithm to increase the search capability. Additionally, I want to experiment with different scoring systems. Once some more patterns emerge I can take this information to define a classification problem to creating recommendations.

Once historical users were found. I then found all of the films that they had rated as at least 8/10, removed from that list, the films I'd seen, then printed the results. See results excerpt below:

	index	title	year	genres	lead actor
0	203878	The Replacements	2000	Comedy	Keanu Reeves
1	19974	Transformers: Revenge of the Fallen	2009	Action|Adventure|Sci-Fi|IMAX	Shia LaBeouf
2	50556	Bigger Than Life	1956	Drama|Mystery|Thriller	James Mason
3	749	High Noon	1952	Drama|Western	Gary Cooper
4	221052	The Greatest Show on Earth	1952	Drama	Betty Hutton
5	94358	Being There	1979	Comedy|Drama	Peter Sellers
6	93350	Patton	1970	Drama|War	George C. Scott
7	10839	The Borrowers	1997	Adventure|Children|Comedy|Fantasy	John Goodman
8	201720	Stanley & Iris	1990	Drama|Romance	Jane Fonda
9	18052	Tender Mercies	1983	Drama|Romance|Western	Robert Duvall
10	72700	Friday Night Lights	2004	Action|Drama	Billy Bob Thornton
11	213938	Geronimo: An American Legend	1993	Drama|Western	Jason Patric
12	142338	All the President's Men	1976	Drama|Thriller	Dustin Hoffman
13	202441	Freaks	1932	Crime|Drama|Horror	Wallace Ford

On inspection of the list, there are some films, I'd already seen, but not yet rated as they were not part of the survey. This effect would need to be built into the workflow of new users; they review films, then receive recommendations, which may be already watched films. The already watched films can form part of the next survey to add to the user's predictors data set.

I'm looking forward to watching some of these to road test the system. After this, with a leap of faith or perhaps with more refinement, I shall send recommendations to my early users for feedback.

FUTURE OF THE PROJECT

As mentioned in each of the sections, various areas can be optimised, or explored differently as this several month project is just the tip of the iceberg. Thankfully there is enough evidence to say that similar users likely exists, and their previous ratings can be used to recommend films to new users on an individual level. Here is a summary of the future experiments proposed:

- Running clustering algorithms to see which patterns exist
- Creating a classification model, predicting good/bad films for users based on a threshold
- Creating a classification model, that has classes: 7, 8, 9, 10 for ratings out of 10

- Creating a classification model that uses each new users preferred features (as graphed above - e.g. favourite directors/producers/screenplays etc
- Principal Component Analysis to improve accuracy of the models, both new and old.
- Creating a front-end to allow users to make advanced searches, recording their usage with google analytics to get insight into what aspects in films they find most important.
- Gamifying the experience for new users to increase the amount of new user data.
- Ensemble methods to combine the best aspects of various models.
- Improving the film title matching algorithm for the 3 datasets to create a larger master dataset.
- Continue approach in last section by improving upon the scoring system with old users.
- Merging the recommendations for new users, to get shared recommendations for a group.

CONCLUSION

Copious amounts of data have been churned into a useful structure to enable people to rate films. There has been a dent made in the goal of finding LAFs for new users, and a lot of new ideas for continued or different experiments have arose as a result of this work. It will be interesting to see this project grow in the future, and who else might find it useful other than the current new users. Concrete examples of how users can currently use this information are the various features that are described in the user profiles generated; preferred/non-preferred: Director, Screenplay, Producer, Director of Photography, Editor, Original Music Composer, music Executive Producer. Similarly for the recommended films on the similar historical users to new user scoring algorithm described above. I can produce a list of basic recommendations for each new user (as I have myself).

Linear models are limited, and time consuming to run. The limitations are that linear relationships don't always exist, regardless of the amount of data. In this instance, it could be that ratings (the predicted item) is not helpful to be thought of as continuous (similarly for old user ratings as predictors). Classification models with new user's preferred and nonpreferred people (classes of features such as Directors/Actors/Screenplays) can be explored to see how accurately a model on these can predict new user's LAFs. Similarly for old user ratings, and using classes 7, 8, 9, 10 for example.

Not all features are going to be useful, and too many will add to the complexity and computer processing power needed. Getting this right balance will be a continued challenge. I'm confident though, that with enough data, and a tuned model or collection of models, that new user's LAFs can be found. The continued question is, how much new user data is needed to find their LAFs?