Final Project — Columbia Engineering Data Analytics Bootcamp

Vipul Aggarwal, Kannika Phadoungxath, June Wang, Liliana Joya

MOVIE RECOMMENDATIONS!

A website that gives customized recommendations based on each user

INTRODUCTION

THE DATA

movieId	title	genres	userId	rating	rating_timestamp
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	2	3.5	2006-03-03 19:57:00
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3	4.0	2015-08-13 13:23:35
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	4	3.0	2019-11-16 22:44:12
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	1997-03-17 19:12:29
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	8	4.0	1998-03-21 15:01:57

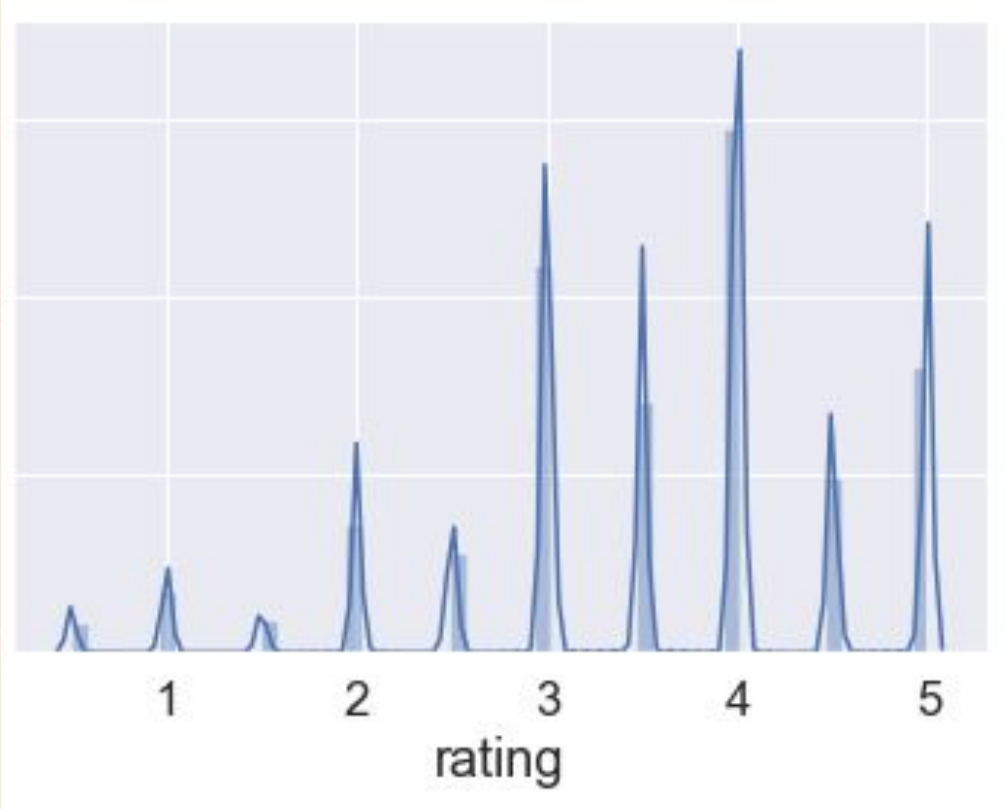
Source: MovieLens 25M Dataset
https://grouplens.org/datasets/movielens/

OUR SCOPE

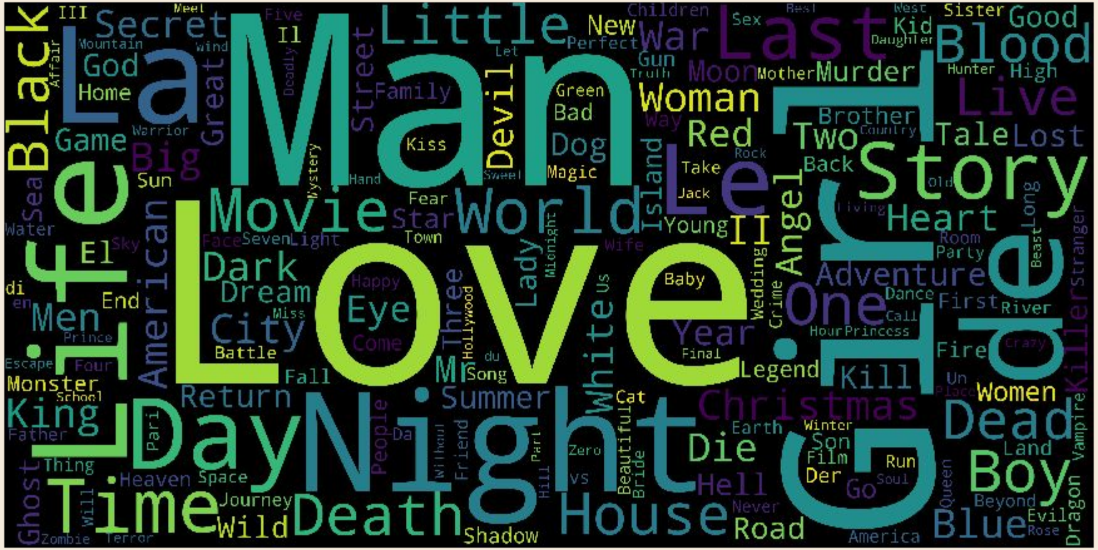
1. Pull data visualizations from the dataset and find fun facts about the data
2. Build machine learning models with a hybrid approach to predict movie recommendations based on various features including: genre, ratings and user rating history.
3. Create an interactive site that will provide the user recommendations after login.

ETL + DATA VISUALIZATIONS

Rating distribution shows that users tend to rate movies highly.

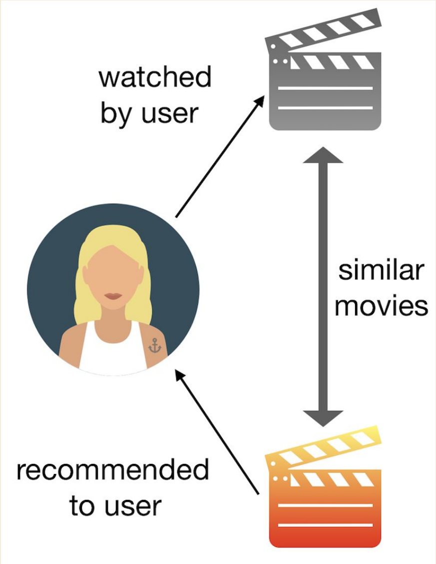


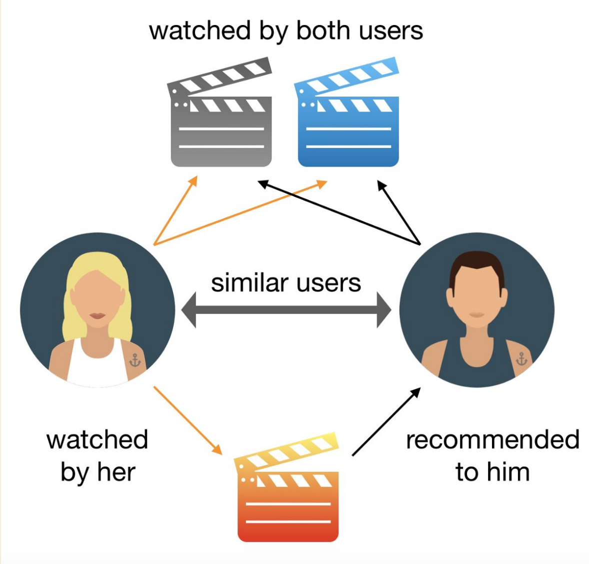
Having fun with the data…



Titles and Genres

CONTENT BASED MODEL VS COLLABORATIVE FILTERING MODEL WITH ALS

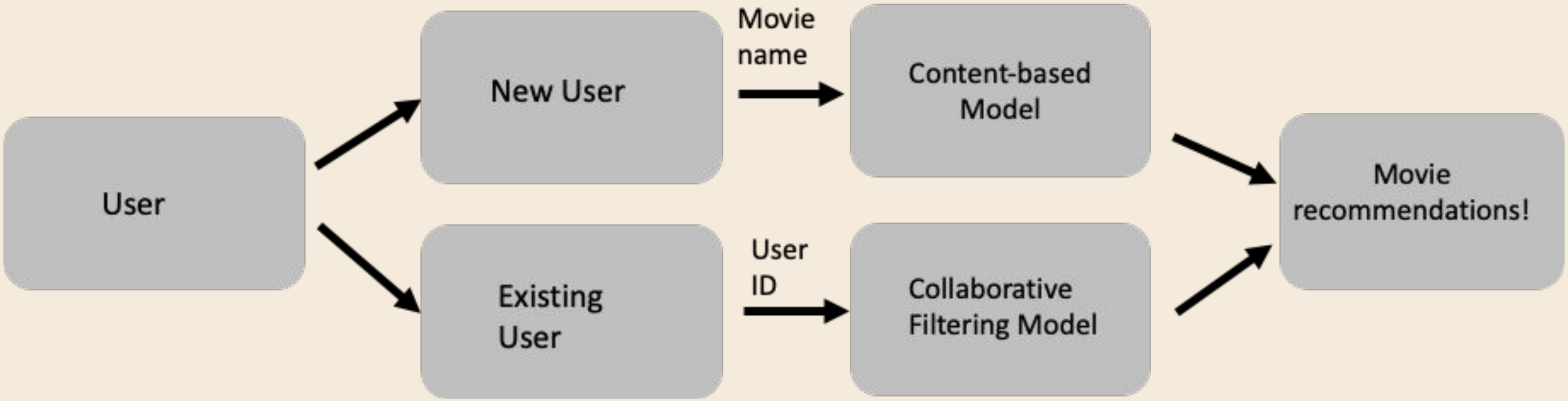




Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).

OUR APPROACH



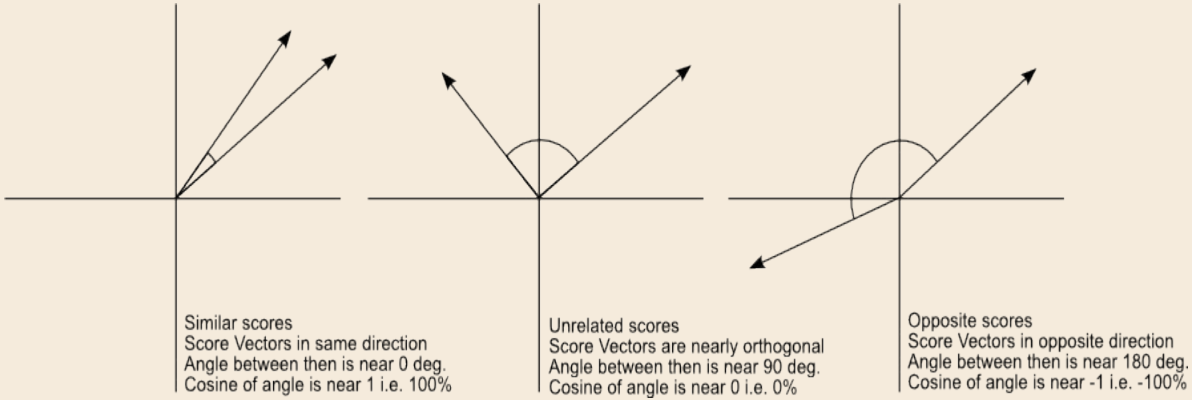
CONTENT BASED RECOMMENDATION MODEL FOR NEW USERS

DataSet Sampling using below conditions:

- Remove Movies with Unknown Genre
- Keep movies with rating greater than or equal to 4
- Keep movies with rating equal or greater than 3.2 and released in 1995 or later

Create Count Vector for feature column and calculate **cosine similarity** between each count vector to calculate similarity between movies based on feature

Recommend movies similar to another movie based on the similarity matrix created between movies.




```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

cv = CountVectorizer()
count_matrix = cv.fit_transform(samples_movie_df['feature'])
#print(count_matrix)
#creating a similarity score matrix
sim = cosine_similarity(count_matrix)
print(sim)
#print(movies_df['comb'])
```

```
[[1.         0.61237244 0.23570226 ... 0.         0.         0.18257419]
 [0.61237244 1.         0.         ... 0.         0.         0.2236068 ]
 [0.23570226 0.         1.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 1.         0.         0.        ]
 [0.         0.         0.         ... 0.         1.         0.31622777]
 [0.18257419 0.2236068  0.         ... 0.         0.31622777 1.        ]]
```

Collaborative Filtering Recommendation Model for Returning Users

- Sample dataset taking only rating for movies having average rating greater than 4 and not having Unknown Genres and random sampling to get 50K records.
- Divide the original data into train and test data
- Create the basic model with ALS
- Fit cross validator to the 'train' dataset to find the best model
- Get rating predictions
- Evaluate the model with RMSE

```
#Create Basic Model
als = ALS(nonnegative=True)\
.setMaxIter(5)\
.setRegParam(0.01)\
.setUserCol("userId")\
.setItemCol("movieId")\
.setRatingCol("rating")\


# Confirm that a model called "als" was created
type(als)
```

```
pyspark.ml.recommendation.ALS
```

```
alsModel = als.fit(training)
```

```
predictions = alsModel.transform(test)
```

userId	movieId	rating	prediction
63474	471	5.0	NaN
78436	471	3.0	NaN
3917	471	3.0	NaN
155398	471	4.5	NaN
73492	833	3.0	0.24391115
6779	1088	3.0	NaN
33357	1088	4.0	1.6808618
65092	1088	4.0	NaN
110826	1088	3.0	NaN
84752	1342	2.0	2.2990913
72055	1342	3.5	NaN
45029	1580	3.5	NaN
132461	1580	5.0	0.084586374
45583	1580	3.0	NaN

```
# View the predictions
test_predictions = alsmodel.transform(test)
RMSE = evaluator.evaluate(test_predictions)
print(RMSE)
```

```
1.6720604233002658
```

LAST STEP: BRIDGE THE BACK END AND THE FRONT END WITH FLASK ...

DEMO TIME

FUTURE PROJECTS

Add more features
to our model to yield
better predictions

Add learning and
memory capabilities
to the model, based
on user input

Improve
user interface with
more movie information
and images

THANK YOU!

QUESTIONS?