BUILDING A MOVIE RECOMMENDATION ENGINE

Jinny Kwon's Final Capstone @ Thinkful

WHY?

Recommendation engines are being used across many industries in all shapes and sizes – media (news), entertainment (streaming), e-commerce, social media, etc.

For companies with a flood of content and choices, it is crucial to bring a **personalized** and **curated** experience to users by suggesting selections that users will find interesting, and therefore, make the product **sticky**.

VALUES

- **Business decisions:** By predicting what users will like, the business can decide **which products to invest in**, ranging from content production to merchandising decisions.
  - Budget, specialized category, marketing strategy, etc.

- **User experience:** Through **a personalized experience**, users find what they want in a few clicks, having an easy and pleasing experience on the platform.
  - Cross-sell, upsell, new user acquisition, etc.

What is a recommendation system?

- **Information filtering system:** removes redundant or unwanted information
- **Recommendation system:** seeks to predict the "rating" or "preference"
  - Collaborative filtering (i.e. based on your activity, we think you'll like this)
    - Challenge: scalability, cold start, sparsity
  - Content-based filtering (based on description of the item and user's preference)
    - Challenge: more data/learning required
  - Hybrid approach

Source: Wikipedia

PROCESS

01

Get to Know the Data

02

Feature Engineering

03

Build Models & Train the Data

04

Evaluation

05

Iteration

01 GET TO KNOW DATA

userId:
Unique ID for each user, numeric

movieId:
Unique movie ID, numeric

tagId:
TagID to describe movie features, numeric

relevance:
importance/weight of the tags to movie

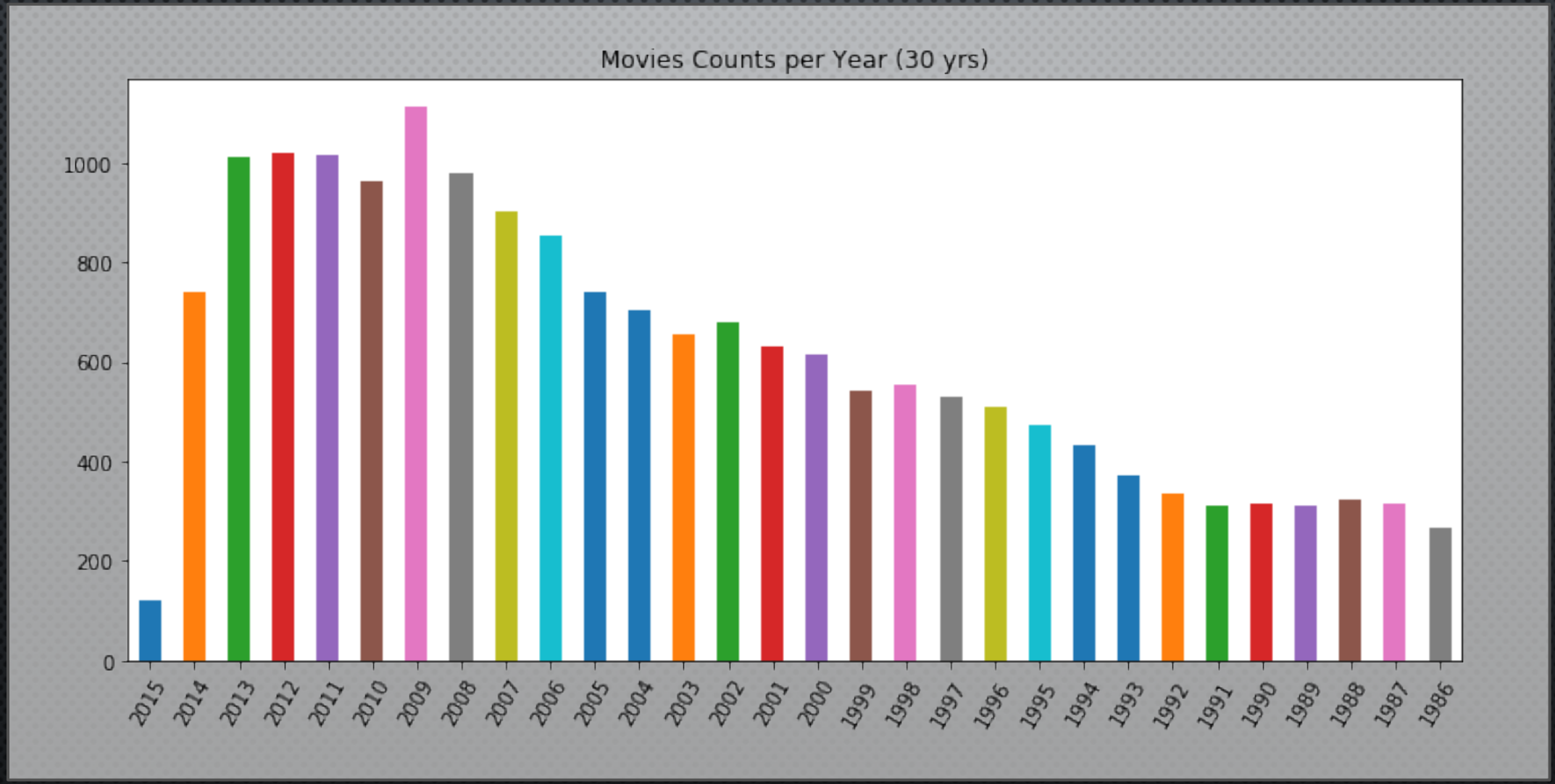
genres:
movie genres, need to transform it to categorical values **

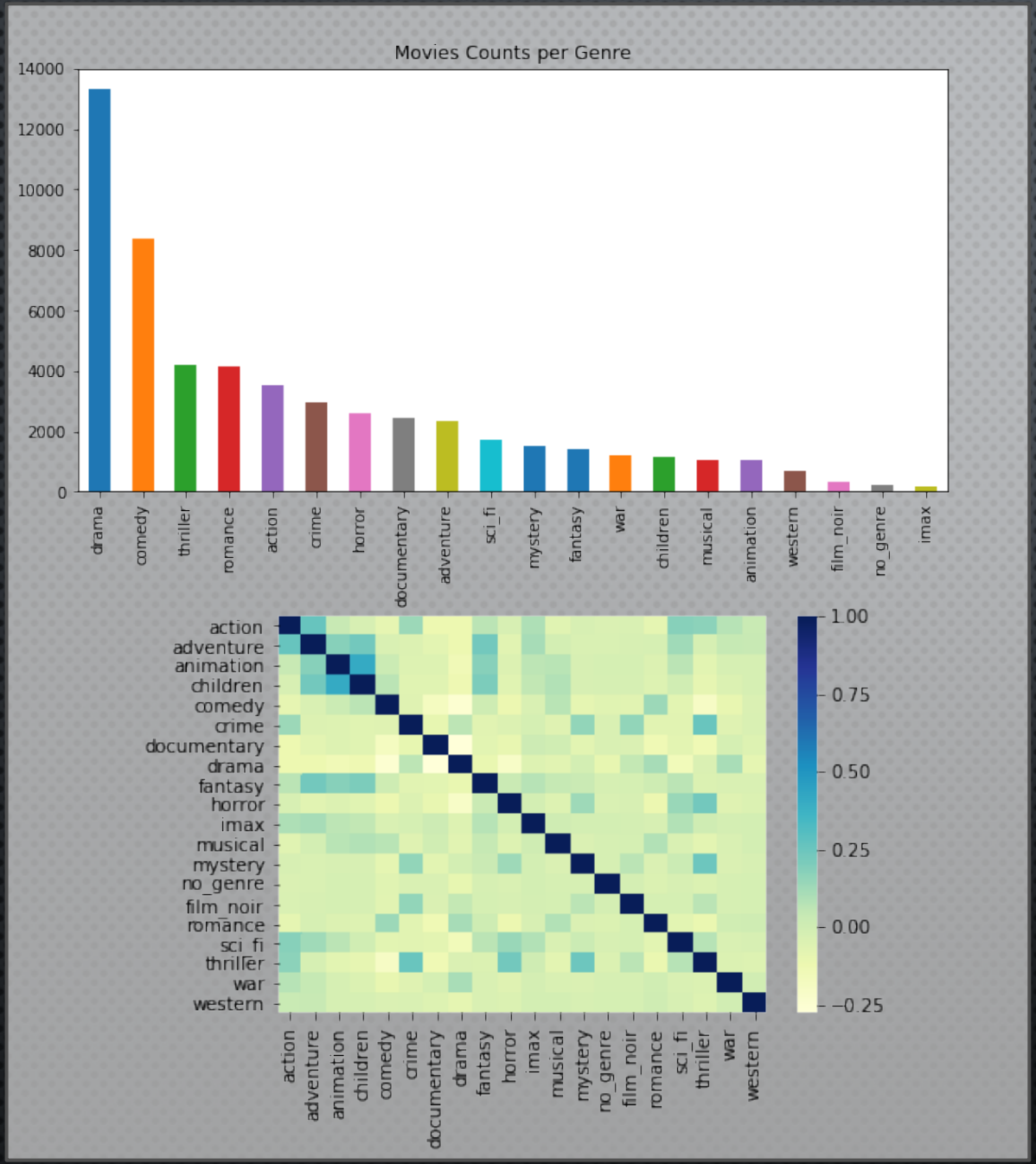
year:
need to extract year from the title **

timestamp:
use the timestamp to divide the train data so that I can introduce the data incrementally

rating:
user's rating to movie, it's the target!

MOVIES MADE BY YEAR





MOVIES PER GENRE

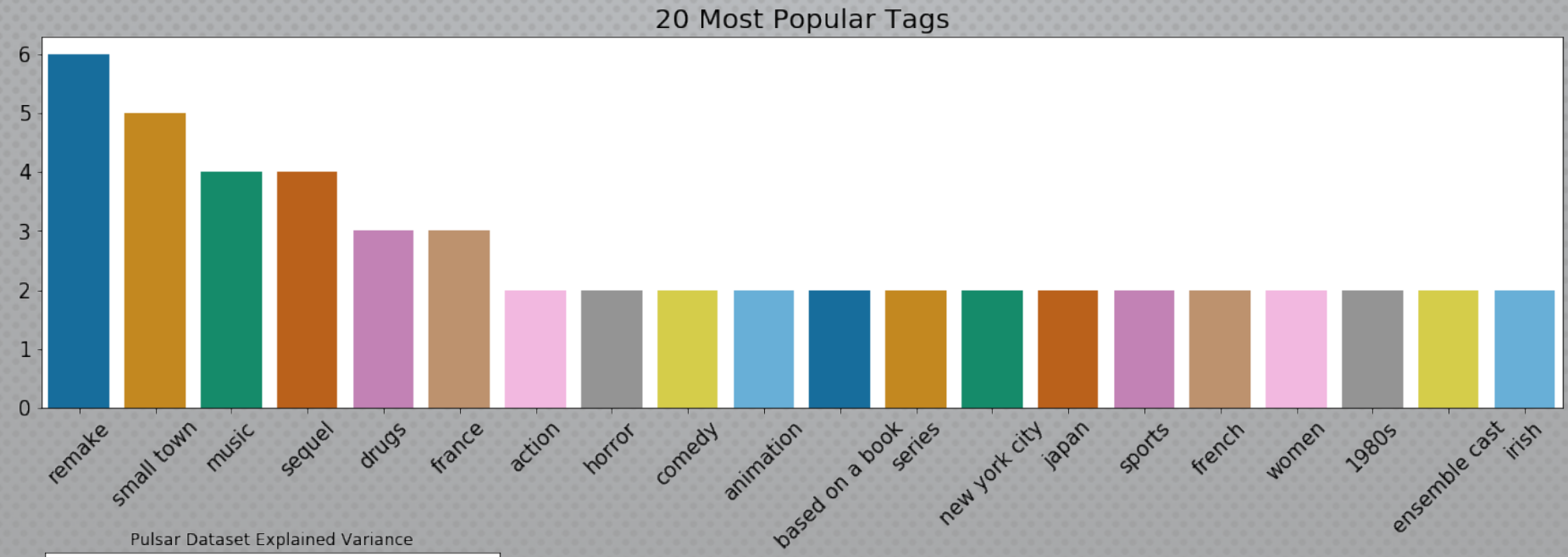
The most common pairs are:

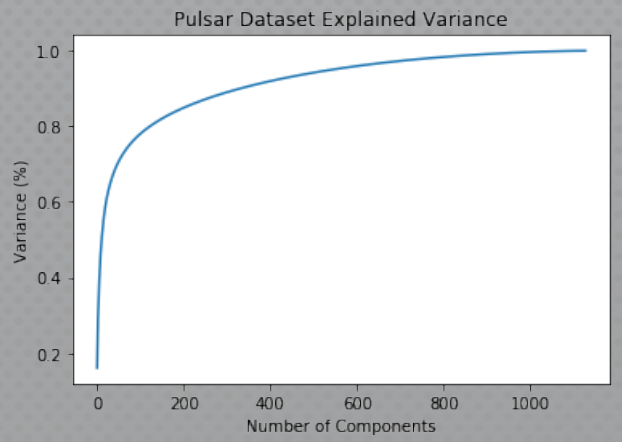
- Drama + Comedy
- Drama + Documentary
- Drama + Romance
- Comedy + Documentary
- Comedy + Romance

```
from collections import Counter
## Just out of curiosity, I want to see the most common pairs

bigrams = zip(genres, genres[1:])
counts = Counter(bigrams)
print(counts.most_common()[:20])
```

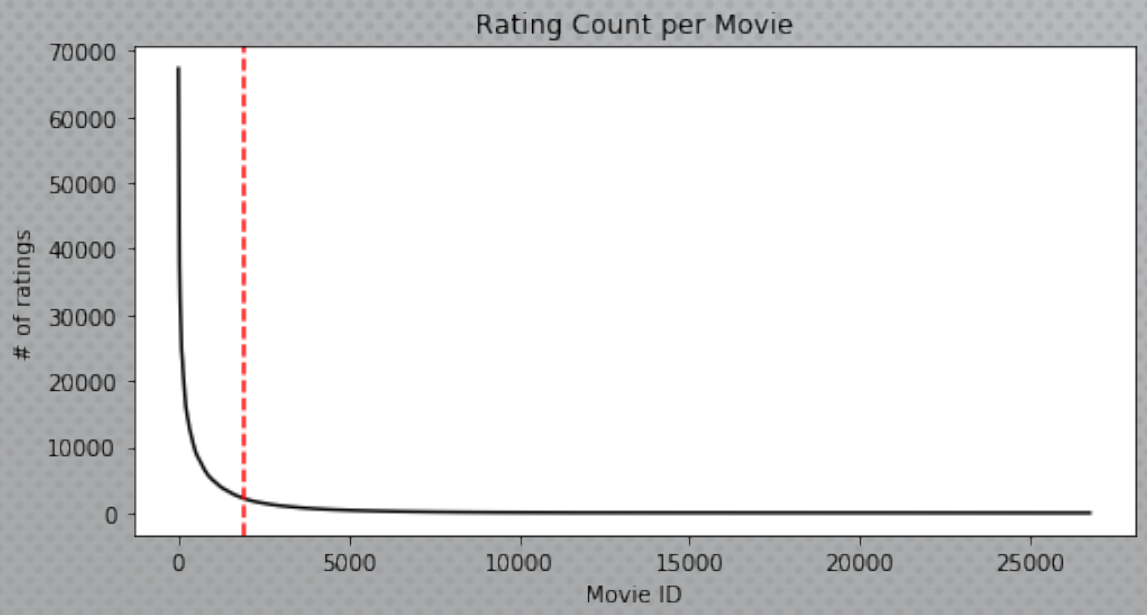
POPULAR TAGS (PCA)

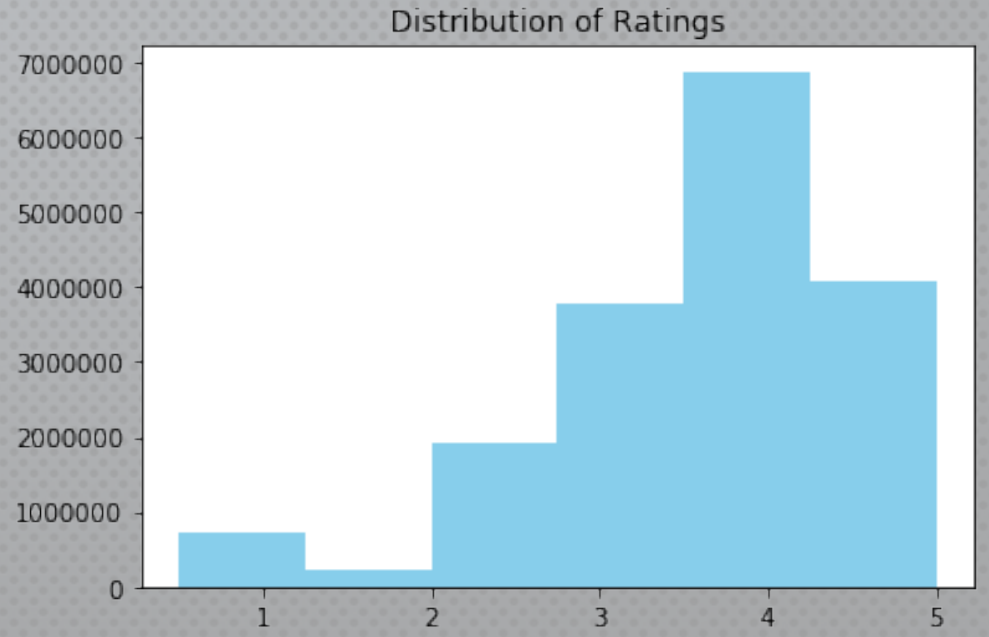




- To eliminate redundant tags, I used **PCA** (ttl tags: 1128).
- I chose **n_components=100** to explain **78%** of the variance.
- For **scalability**, we want to reduce the number of tags in production.

RATINGS (LONG TAIL PLOT)





- 80% of the ratings are from 1,950 movies - 8% of all movies in the dataset.
- For the recommendation engine, we will only recommend the top 3,000 rated movies.

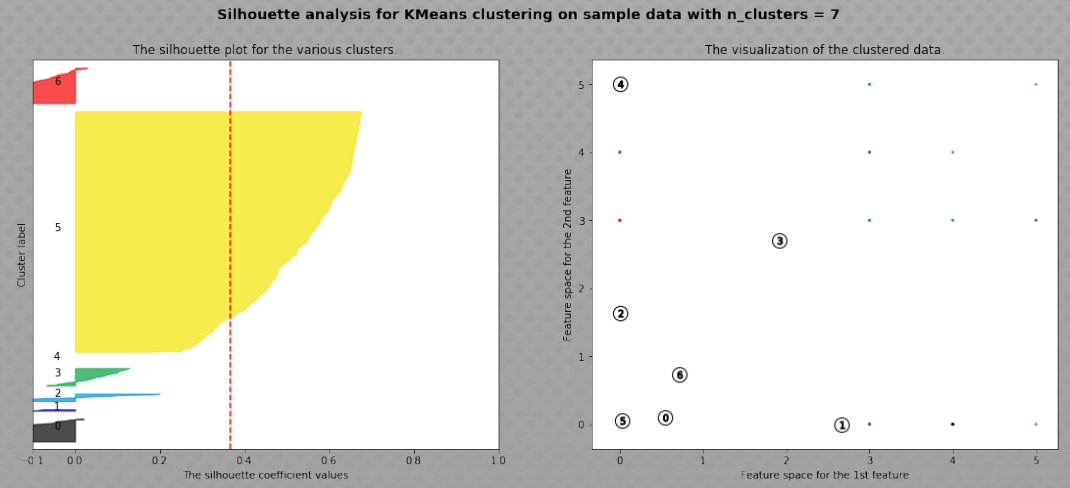
MODEL I - NEAREST NEIGHBORS (COSINE SIMILARITY)

- USING USER RATINGS

```
People who liked *Misérables, Les (1995)* also liked:

1: Ed Wood (1994), with similarity score of 0.6082635908182086
2: Menace II Society (1993), with similarity score of 0.6315810626988685
3: Indian in the Cupboard, The (1995), with similarity score of 0.6329531255620646
4: Boxing Helena (1993), with similarity score of 0.6590087675832442
```

```
For n_clusters = 3 The average silhouette_score is : 0.3706924758536127
For n_clusters = 4 The average silhouette_score is : 0.3431701042573199
For n_clusters = 5 The average silhouette_score is : 0.32491394348499375
For n_clusters = 7 The average silhouette_score is : 0.3673961546013058
For n_clusters = 9 The average silhouette_score is : 0.26443981621798907
```

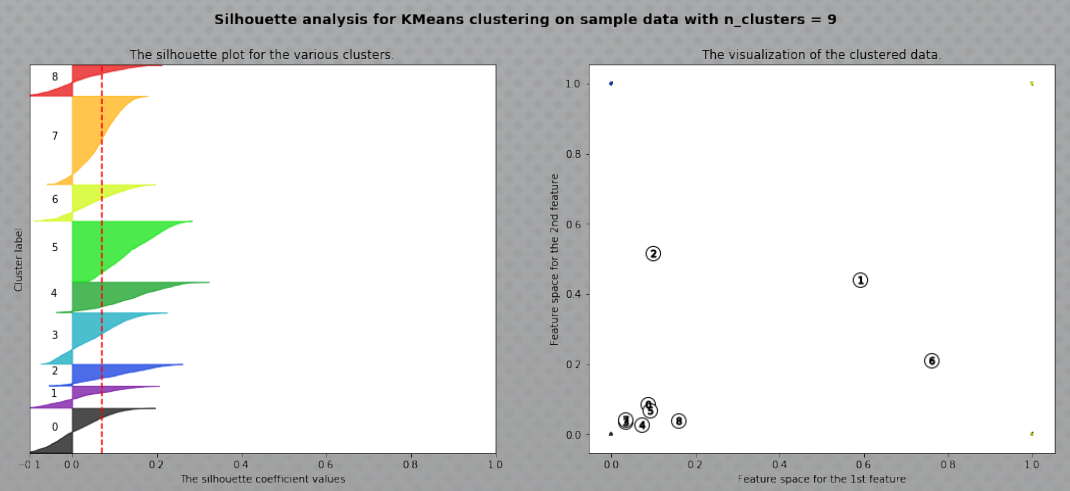


- USING MOVIE FEATURES (TAGS)

```
For n_clusters = 3 The average silhouette_score is : 0.07410225432406302
For n_clusters = 4 The average silhouette_score is : 0.06445806171589429
For n_clusters = 5 The average silhouette_score is : 0.07494724305163554
For n_clusters = 7 The average silhouette_score is : 0.06397409050457016
For n_clusters = 9 The average silhouette_score is : 0.06999511555497398
```

```
If you liked *Stepfather II (1989)*, you might also like:

1: Child's Play 2 (1990), with similarity score of 0.11632292724744109
2: Psycho III (1986), with similarity score of 0.11733792119260478
3: Halloween: The Curse of Michael Myers (Halloween 6: The Curse of Michael Myers) (1995), with si
milarity score of 0.1245249073680077
4: Children of the Corn IV: The Gathering (1996), with similarity score of 0.15550727274309284
```



KERAS

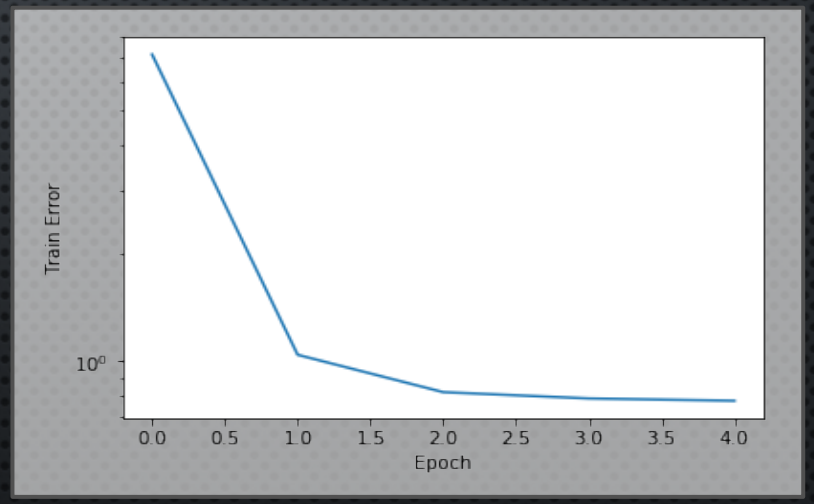
- PREDICT USER RATINGS USING EMBEDDING METHOD
  - PREDICTING ACCURATE RATINGS WILL HELP THE ENGINE TO DECIDE WHICH MOVIES TO RECOMMEND

```
import keras
from IPython.display import SVG
from keras.optimizers import Adam
from keras.utils.vis_utils import model_to_dot

movie_input = keras.layers.Input(shape=[1],name='Item')
movie_embedding = keras.layers.Embedding(n_movies + 1, 6, name='Movie-Embedding')(movie_input)
movie_vec = keras.layers.Flatten(name='FlattenMovies')(movie_embedding)

user_input = keras.layers.Input(shape=[1],name='User')
user_vec = keras.layers.Flatten(name='FlattenUsers')(keras.layers.Embedding(n_users + 1, 6, name=
'User-Embedding')(user_input))

prod = Dot(name="Dot-Product", axes=1)([movie_vec, user_vec])
model = keras.Model([user_input, movie_input], prod)
model.compile('adam', 'mean_squared_error')
```



- y_pred vs. y_actual
  - MAE: 0.62
  - RMSE: 0.83

PRODUCT ROADMAP

1

Improve deep learning model to predict ratings more accurately

2

Narrow down popular movies (less than 2,000)

3

Implement **candidate selection**: Combine user rating & movie feature

4

Monitor conversion & **iterate**