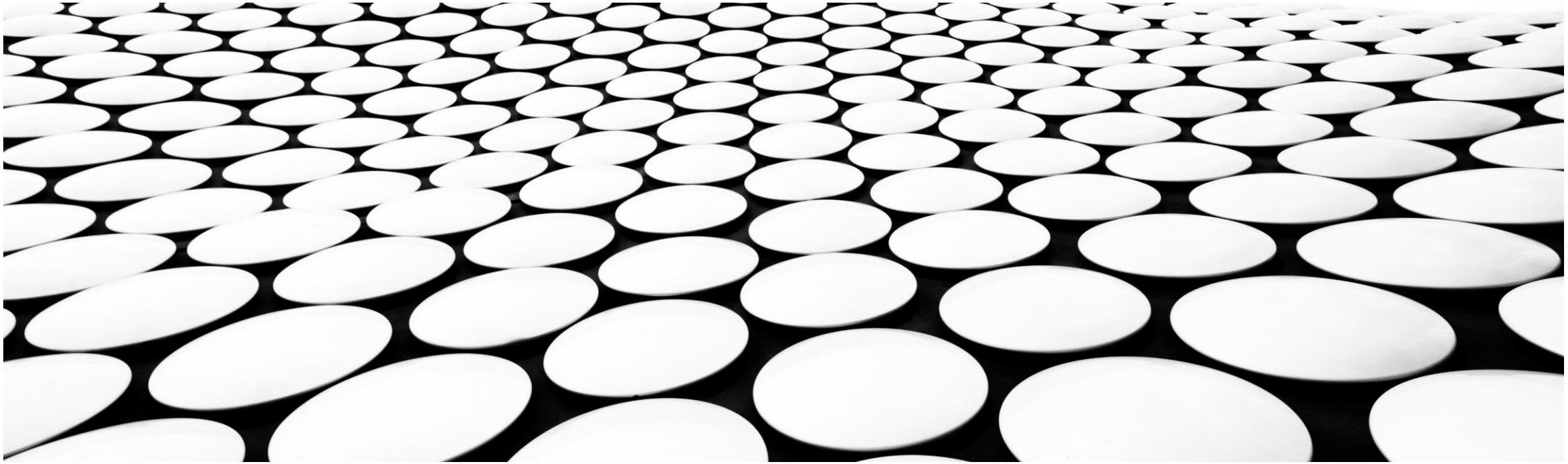
MATRIX COMPLETION FOR RECOMMENDATION SYSTEM

GROUP 12：110511312 劉曉迪、110550143 洪巧芸、110651079 徐翊昕、111511271 黃庭豐、111705015 張詠翔



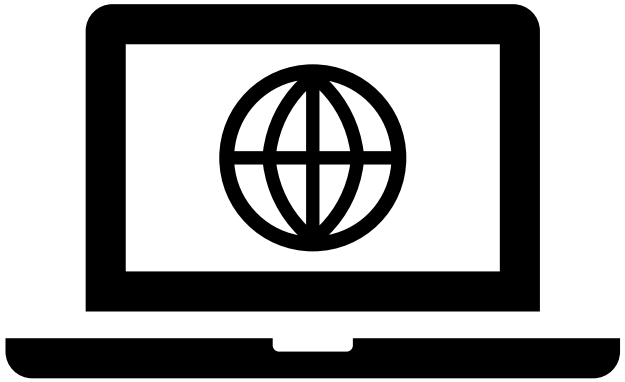
HERE IS OUR VIDEO

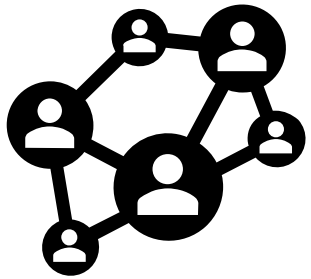
- https://www.youtube.com/watch?v=ZPE14o_cPzA

PROCESS

WHAT IS RECOMMENDATION SYSTEM

- An information filter system makes consumers choose what they want more efficiently.
- NETFLIX, SHOPEE, and some online platform.





PREPARATION
IMPORT LIBRARY

- To load datasets, define models, train and test, data analysis and so on.
- Reader: parse files
- SVD: matrix factorization

```
pip install scikit-surprise
```

```
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.10/dist-packages (1.1.3)
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.3.
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.23
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.11.
```

```
from surprise import Dataset, Reader
from surprise import SVD
from surprise.model_selection import train_test_split
from surprise import accuracy
```

```
import numpy as np
import pandas as pd
```

PREPARATION
FILTERING DATA

- Read data from dataset
- Filter the data that corresponds to movies to focus on analysis

```
movie = pd.read_csv('/content/movie.csv')
rating = pd.read_csv('/content/rating_df.csv')
genome_scores = pd.read_csv('/content/genome_scores.csv')
link = pd.read_csv('/content/link.csv')
tag = pd.read_csv('/content/tag.csv')
test_data = pd.read_csv('/content/test_v2.csv')
```

```
# select the column 'movieId' from the test data
movieId = test_data['movieId']

# filter down the data to include only the information relevant to the movies in the test data
movie = movie[movie.movieId.isin(movieId)]
rating = rating[rating.movieId.isin(movieId)]
genome_scores = genome_scores[genome_scores.movieId.isin(movieId)]
tag = tag[tag.movieId.isin(movieId)]
link = link[link.movieId.isin(movieId)]
```

PREPARATION
MERGE

- 'how = "left" : specify the column you want to the left, and "movieId" is specified to be performed.
- Merge the data to check whether rows match rating

```
train_data = movie.merge(rating, how = "left", on = "movieId" )
train_data.head()
```

	movieId	title	genres	userId	rating	timestamp
0	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	107	4.5	2014-04-12 21:27:49
1	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	108	3.0	1996-07-05 12:26:24
2	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	109	4.0	2003-07-10 17:48:13
3	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	110	5.0	2007-02-10 04:02:30
4	356	Forrest Gump (1994)	Comedy|Drama|Romance|War	116	4.0	2005-11-23 02:06:25

PREPARATION

- Specify the rating scale
- Split the data into training and testing sets

```
# Create a Reader object to specify the rating scale
reader = Reader(rating_scale=(1, 5))

# Load the data into a Surprise Dataset
train_dataset = Dataset.load_from_df(train_data[['userId', 'movieId', 'rating']], reader)

# Specify the desired test set size
test_size = 12340

# Split the data into training and testing sets
Trainset, Testset = train_test_split(train_dataset, test_size=test_size, random_state=42)
```

REGRESSION

Root Mean Square Error (RMSE)

- Lower the great number of MSE to make better model performance
- ROOT
- $\sqrt{\frac{\sum_{i=0}^{n}(y_{prediction}-y_{actual})^2}{n}}$

$n =$ sample size

$y_{prediction} =$ the predicted data value

$y_{actual} =$ the actual data value

Mean Absolute Error (MAE)

- Just calculate the error and take average
- $\frac{\sum_{i=0}^{n} | y_{prediction}-y_{actual} |}{n}$

$n =$ sample size

$y_{prediction} =$ the predicted data value

$y_{actual} =$ the actual data value

SVD

- Matrix factorization to filter the recommendation system
- Compute the predicted rating and actual rating with RMSE and MAE to evaluate the accuracy of regression model
- Result!

```
# SVD Model Training
svd_model = SVD()
svd_model.fit(Trainset)
predictions = svd_model.test(Testset)
```

```
 # Calculate RMSE and MAE
rmse = accuracy.rmse(predictions)
mae = accuracy.mae(predictions)
```

```
RMSE: 0.9282
MAE:  0.7174
```

EVALUATE

- Set a threshold and convert predicted rating to binary labels based on it.
- Get the true labels from Testset.
- Calculate precision, recall and F1 score with the consideration of the imbalance in the class distribution.

```
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
# Convert predictions to binary labels based on a threshold
threshold = 4.0
binary_labels = [1 if pred.est >= threshold else 0 for pred in predictions]

# Get the true labels from Testset
true_labels = [int(test_tuple[2]) for test_tuple in Testset]


# Calculate precision, recall, and f1_score
precision = precision_score(true_labels, binary_labels, average='weighted')
recall = recall_score(true_labels, binary_labels, average='weighted')
f1 = f1_score(true_labels, binary_labels, average='weighted')

# Print the metrics
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
Precision: 0.000335837297179915
Recall: 0.016693679092382497
F1 Score: 0.0006579898700990305
```

EVALUATE

Then here comes the result!

```
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
# Convert predictions to binary labels based on a threshold
threshold = 4.0
binary_labels = [1 if pred.est >= threshold else 0 for pred in predictions]

# Get the true labels from Testset
true_labels = [int(test_tuple[2]) for test_tuple in Testset]


# Calculate precision, recall, and f1_score
precision = precision_score(true_labels, binary_labels, average='weighted')
recall = recall_score(true_labels, binary_labels, average='weighted')
f1 = f1_score(true_labels, binary_labels, average='weighted')

# Print the metrics
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

```
Precision: 0.000335837297179915
Recall: 0.016693679092382497
F1 Score: 0.0006579898700990305
```

THANKS FOR READING