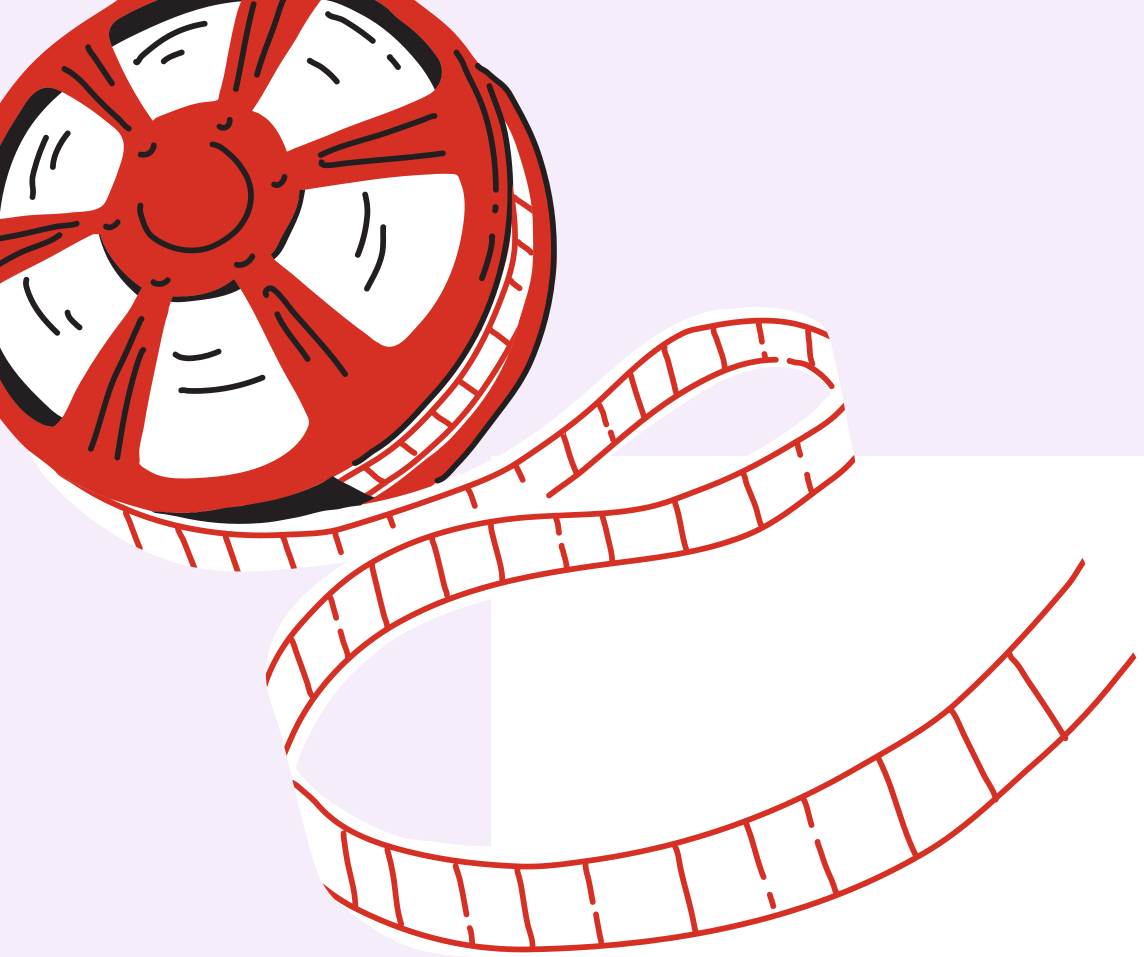
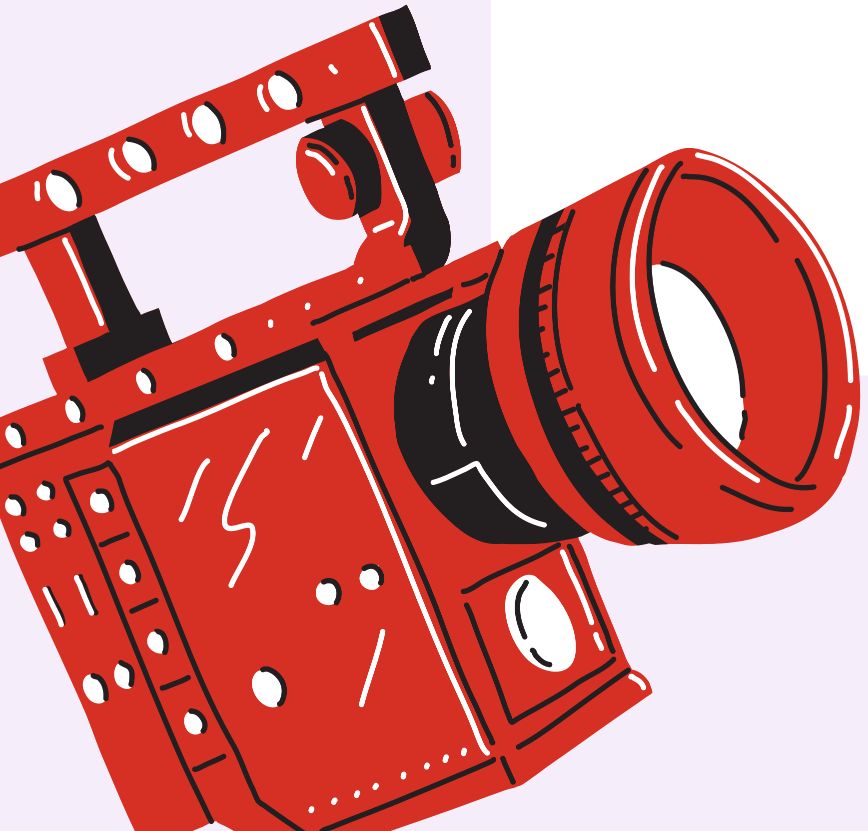


Big Data Analysis

movie recommendation system

KOUFOUNDA Anthony, HRIRD Sihem, BOUTERA Killian, DAHIRI Idriss



Problem statement

Develop a movie recommendation system capable of suggesting relevant movies to users based on their past preferences.

Machine learning
Objectives and challenges

Objectives

- Build a model that analyzes user preferences
- Ensure scalability to handle large datasets
- Adapt the model to changing user preferences and trends

Challenges

- Correcting recommendation errors (false positives/negatives)
- Cold Start Problem (new users or unknown movies)

What is Pandas?

Pandas is essential in data science and machine learning because it enables:

- Import/export of data (CSV, Excel, SQL, etc.).
- Data cleaning and transformation.
- Efficient data analysis and manipulation.

When to use Pandas ?

1 Data Preprocessing & Cleaning
- Handling missing values
- Removing duplicates
- Formatting data types

2 Exploratory Data Analysis (EDA)
- Summary statistics
- Data visualization
- Checking correlations

3 Feature Engineering
- Creating new feature
- Encoding categorical data
- Normalizing data

4 Integrating with Machine Learning Models
- Preparing datasets for scikit-learn
- Splitting data
- Storing and retrieving large datasets efficiently

Understanding the Cosine Similarity Maxtrix

- Cosine Similarity measures how similar two items are by calculating the cosine of the angle between their vectors in a multi-dimensional space. The smaller the angle, the higher the similarity.

- It is widely used in recommendation systems to suggest items (e.g., movies, products) that are similar to those a user has interacted with, based on shared features like genres, ratings, or keywords.

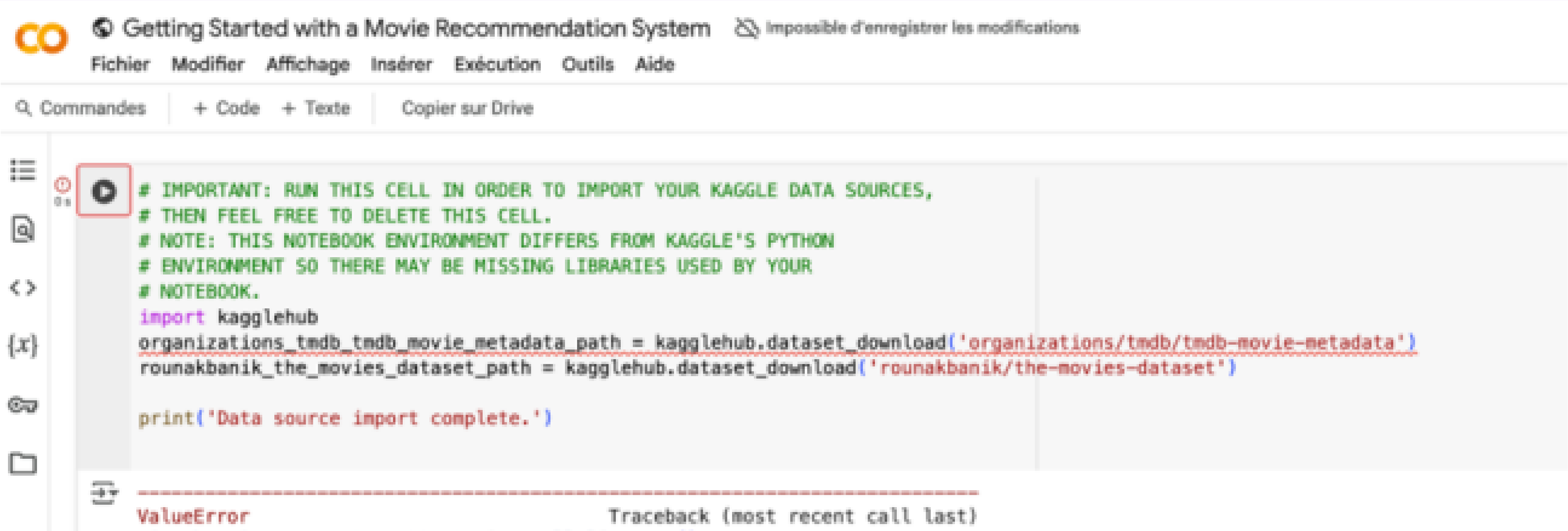
- In a movie recommendation system, Cosine Similarity helps identify movies that are similar to those you've already liked, improving personalized recommendations based on your viewing history.

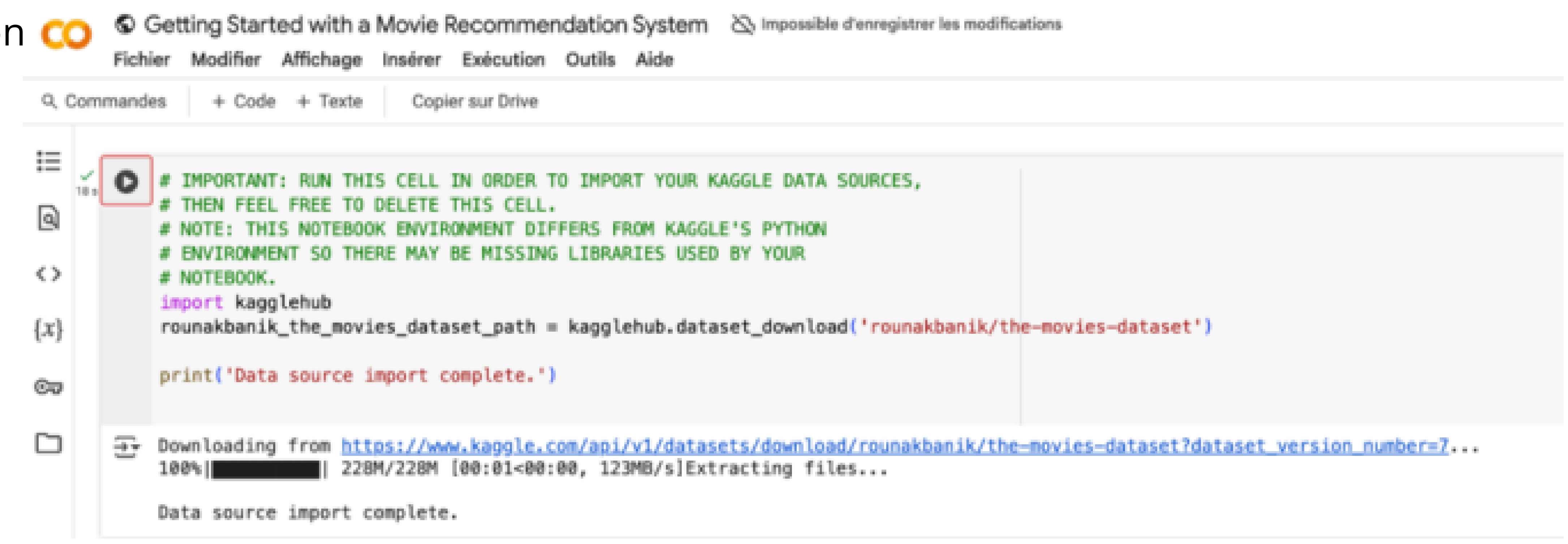
```python
# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel

# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```python
# Function that takes in movie title as input and outputs most similar movie
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title'].iloc[movie_indices]
```

Mistakes in the model

- The dataset 'organizations/tmdb/tmdb-movie-metadata' returned an error because it is either invalid, non-existent, or private. We replaced it with rounakbanik/the-movies-dataset, the main dataset used in the notebook analysis. It contains valuable information for movie recommendations.

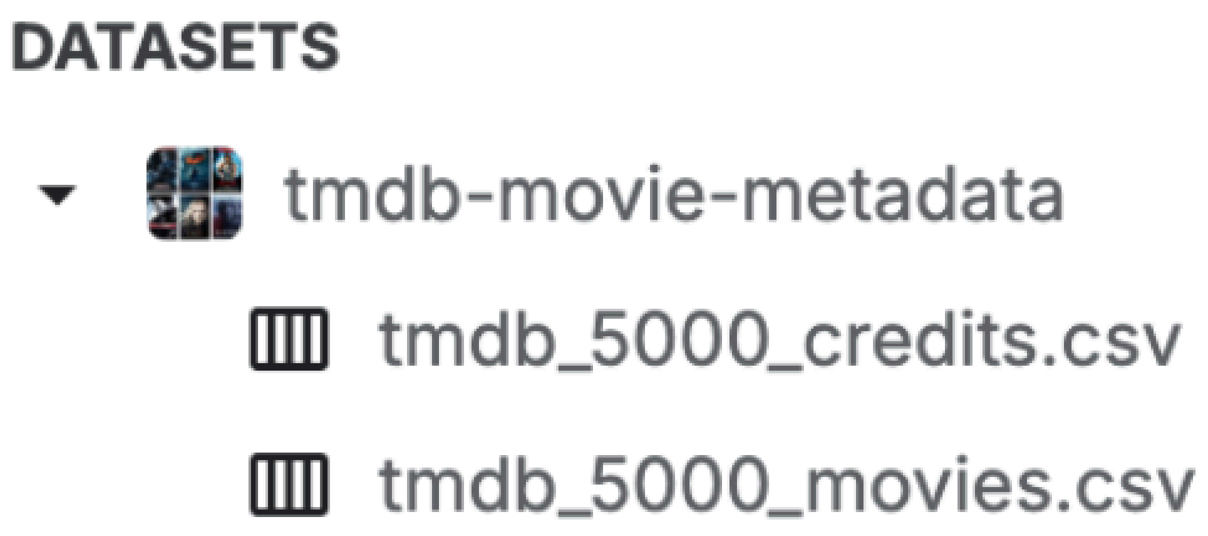




Mistakes in the model

We encountered a new error because the notebook tries to read two files:

- tmdb_5000_credits.csv
- tmdb_5000_movies.csv
- These files were missing in Colab. We downloaded the dataset with both files and uploaded them to Colab.



- To read the zip files, we replaced the cell with: df1 = pd.read_csv('/content/tmdb_5000_credits.csv')

Mistakes in the model

```
from surprise import Reader, Dataset, SVD, evaluate
reader = Reader()
ratings = pd.read_csv('../input/the-movies-dataset/ratings_small.csv')
ratings.head()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-53-b22091582d52> in <cell line: 0>()
----> 1 from surprise import Reader, Dataset, SVD, evaluate
      2 reader = Reader()
      3 ratings = pd.read_csv('../input/the-movies-dataset/ratings_small.csv')
      4 ratings.head()

ModuleNotFoundError: No module named 'surprise'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

1

Note that in this dataset movies are rated on a scale of 5 unlike the earlier one.

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
data.split(n_folds=5)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-39-7ac32b723610> in <cell line: 0>()
      1 data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
----> 2 data.split(n_folds=5)

AttributeError: 'DatasetAutoFolds' object has no attribute 'split'
```

3

Étapes suivantes : Expliquer l'erreur

```
[40] svd = SVD()
     evaluate(svd, data, measures=['RMSE', 'MAE'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-40-8cec4f0b0ff7> in <cell line: 0>()
      1 svd = SVD()
----> 2 evaluate(svd, data, measures=['RMSE', 'MAE'])

NameError: name 'evaluate' is not defined
```

```
!pip install numpy==1.24.3 --quiet
!pip install scikit-surprise --quiet
```

2

Note that in this dataset movies are rated on a scale of 5 unlike the earlier one.

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
```

```
[46] svd = SVD()
     cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8983  0.8927  0.8977  0.8956  0.8959  0.8960  0.0020
MAE (testset)     0.6899  0.6886  0.6908  0.6884  0.6899  0.6895  0.0009
Fit time          1.37    1.84    1.51    1.36    1.37    1.49    0.18
Test time         0.13    0.18    0.13    0.12    0.26    0.16    0.06
{'test_rmse': array([0.89834895, 0.8926532 , 0.89768   , 0.89555296, 0.89588433]),
 'test_mae': array([0.6898602 , 0.6886363 , 0.69077994, 0.68841849, 0.68993053]),
 'fit_time': (1.369171142578125,
  1.843813180923462,
  1.5063469409942627,
  1.357344150543213,
  1.3740580081939697),
 'test_time': (0.12675762176513672,
  0.1777021884918213,
  0.12642955780029297,
  0.11530637741088867,
  0.26496458053588867)}
```

4

We get a mean Root Mean Sqaure Error of 0.89 approx which is more than good enough for our case. Let us now train on our dataset and arrive at predictions.

Understanding the machine learning: RMSE and MAE model

Analysis:

- RMSE (Root Mean Square Error): Measures the average squared difference between predicted and actual ratings. The closer to 0, the more accurate the model.
- MAE (Mean Absolute Error): Measures the average absolute difference between predictions and actual ratings, giving a direct idea of the average deviation.

Note that in this dataset movies are rated on a scale of 5 unlike the earlier one.

Loading...

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
```

```
[45] svd = SVD()
     cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8923  0.8915  0.8979  0.8920  0.9085  0.8964  0.0065
MAE (testset)     0.6875  0.6904  0.6924  0.6872  0.6968  0.6909  0.0035
Fit time          3.58    1.67    1.39    1.38    1.40    1.89    0.86
Test time         0.66    0.11    0.11    0.15    0.31    0.27    0.21
{'test_rmse': array([0.89231086, 0.89145974, 0.89787191, 0.8919773 , 0.90850295]),
 'test_mae': array([0.68752892, 0.6904173 , 0.69243425, 0.68723129, 0.69676396]),
 'fit_time': (3.5830914974212646,
  1.6695563793182373,
  1.3947563171386719,
  1.3799316883087158,
  1.3998298645019531),
 'test_time': (0.6566212177276611,
  0.1069033145904541,
  0.10759663581848145,
  0.1472320556640625,
  0.308164119720459)}
```

We get a mean Root Mean Sqaure Error of 0.89 approx which is more than good enough for our case. Let us now train on our dataset and

- Average RMSE: **0.8964** (low error, accurate predictions)
- Average MAE: **0.6909** (small gap between predictions and actual ratings)

Discussion about the Notebook

Widi Satriaji
Posted 6 years ago · Posted on Version 8 of 11

2

Sorry, but I don't understand. Why did you use `popularity` when you've just computed the `score` on demographic filtering? Thanks!

Reply React

Ibtesam Ahmed Posted 6 years ago · Posted on Version 8 of 11 TOPIC AUTHOR

1

In `score` we calculate the best rated movies and I'm using popularity for `Under the Trending Now tab of these systems we find movies that are very popular and they can just be obtained by sorting the dataset by the popularity column.`

Reply React

Discussion about the Notebook

Hamza El Bouatmani
Posted 6 years ago · Posted on Version 8 of 11

▲ 2

Thank you for this well-written and informative Kernel Ibtessam.

I have one question: You said that because you used TF-IDF, you can use linear_kernel instead of cosine similarity. Can you please (or anyone who reads this) explain it ?

Reply React

Ibtesam Ahmed Posted 6 years ago · Posted on Version 8 of 11 TOPIC AUTHOR

▲ 1

I use linear kernel instead of cos sim because it's faster since you only have to compute the dot product unlike cos sim.

Reply React

CONCLUSION

- Building an effective recommendation system

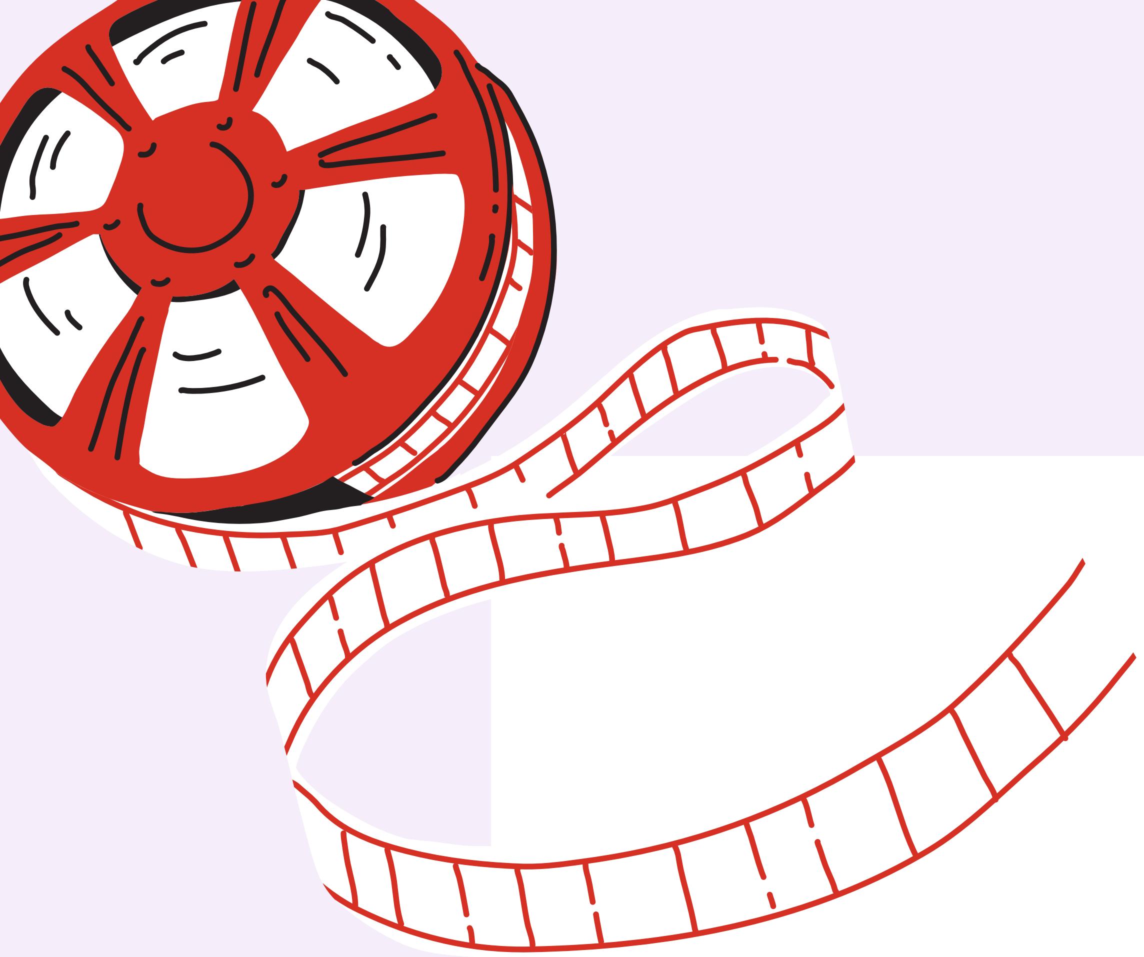
→ Used SVD with a RMSE of 0.8960 and MAE of 0.6895, ensuring accurate predictions.

- Handling errors and incompatibilities

→ Fixed obsolete functions, adapted to library updates, and improved debugging skills.

- Comprehensive machine learning approach

→ Covered data processing, model implementation, cross-validation, and performance analysis.



Thanks for your attention