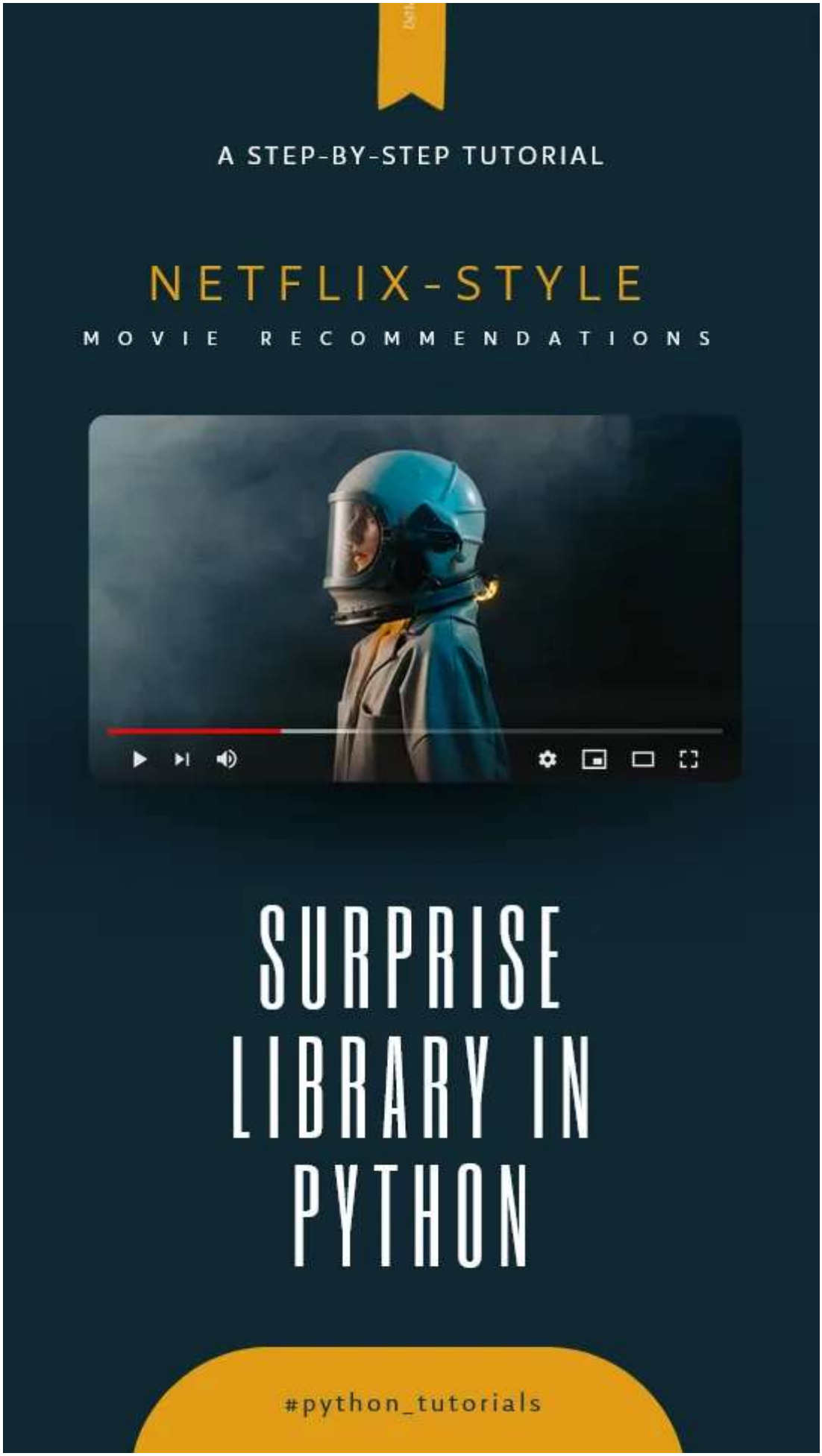
Published in Artificial Intelligence in Plain English

Waleed Mousa Follow

Feb 28 · 3 min read · Listen

Save

A Netflix-Style Movie Recommendations with the Surprise Library in Python


A STEP-BY-STEP TUTORIAL
NETFLIX-STYLE
MOVIE RECOMMENDATIONS
SURPRISE
LIBRARY IN
PYTHON
#python_tutorials

Surprise is a Python library that makes it easy to build and evaluate recommender systems.

It provides a number of **built-in algorithms**, as well as tools for evaluating the accuracy of those algorithms.

In this tutorial, we'll be building a **simple movie recommendation system similar to Netflix.** Here's an outline of the steps we'll follow:

1. Installing the Surprise Library
2. Importing Required Libraries
3. Loading the Dataset
4. Splitting the Dataset
5. Defining the Recommender System Algorithm
6. Training the Recommender System
7. Testing the Recommender System
8. Making Predictions
9. Recommending Movies
10. Conclusion

Let's get started!

1. Installing the Surprise Library

Before we can build our recommender system, we need to install the Surprise library. You can install it using pip with the following command:

```
!pip install scikit-surprise
```

2. Importing Required Libraries

Once we have the Surprise library installed, we can import it, as well as any other libraries we'll need.

For this tutorial, we'll be using Pandas to load and manipulate the dataset.

```
import pandas as pd
from surprise import Dataset
from surprise import Reader
from surprise import SVD
from surprise.model_selection import train_test_split
from surprise import accuracy
```

3. Loading the Dataset

We'll be using the MovieLens 100K dataset for this tutorial. You can download the dataset from this link: https://grouplens.org/datasets/movielens/100k/

```
ratings = pd.read_csv('path/to/ratings.csv')
```

4. Splitting the Dataset

Before we can train our recommender system, we need to split the dataset into a training set and a testing set. We'll use 80% of the data for training and 20% for testing.

```
trainset, testset = train_test_split(data, test_size=.2)
```

5. Defining the Recommender System Algorithm

Next, we need to define the algorithm we'll be using for our recommender system.

For this tutorial, we'll be using **Singular Value Decomposition (SVD)**, which is a popular algorithm for recommendation systems.

```
algo = SVD()
```

6. Training the Recommender System

Now we're ready to train our recommender system. We'll use the fit() method of our algorithm object to train it on the training set.

```
algo.fit(trainset)
```


7. Testing the Recommender System

After training our recommender system, we need to evaluate its accuracy on the testing set.

We'll use the test() method of our algorithm object to get predictions for the test set, and then use the accuracy() function to compute the accuracy metrics.

```
predictions = algo.test(testset)
accuracy.rmse(predictions)
```

8. Making Predictions

Now that we've trained and tested our recommender system, we can use it to make predictions for individual users and movies.

We'll create a function that takes a user ID and a movie ID as input, and returns a predicted rating.

```
def predict_rating(user_id, movie_id):
    prediction = algo.predict(user_id, movie_id)
    return prediction.est
```

9. Recommending Movies

Finally, we can use our recommender system to recommend movies to users. We'll create a function that takes a user ID as input, and returns a list of the top 10 movies that the user is most likely to enjoy.

```
def recommend_movies(user_id):
    # Get a list of all the movies the user has not rated
    all_movies = ratings['movieId'].unique()
    user_movies = ratings[ratings['userId'] == user_id]['movieId'].unique()
    new_movies = list(set(all_movies) - set(user_movies))

    # Predict the ratings for the new movies
    predictions = [algo.predict(user_id, movie_id) for movie_id in new_movies]

    # Sort the predictions by estimated rating
    predictions.sort(key=lambda x: x.est, reverse=True)

    # Get the top 10 recommendations
    top_recommendations = [prediction.iid for prediction in predictions[:10]]

    return top_recommendations
```

10. Conclusion

And that's it! With just a few lines of code using the **Surprise library**, we were able to build a recommender system that can make accurate predictions and provide movie recommendations to users.

Of course, this is just a **basic example**, and there are many ways to customize and improve the recommender system.

But hopefully this tutorial has given you a good starting point for building your own recommendation engine using the Surprise library in Python.

If you enjoyed reading this tutorial and found it helpful, please consider supporting me on Buy Me a Coffee.

*More content at **PlainEnglish.io**.*

*Sign up for our **free weekly newsletter**. Follow us on **Twitter**, **LinkedIn**, **YouTube**, and **Discord**.*

***Interested in scaling your software startup**? Check out **Circuit**.*

Computer Science Data Science Machine Learning Artificial Intelligence Python