```
• Discuss practical examples demonstrating the use of Surprise library
    for recommendation tasks
Introduction to the Surprise Library for
recommendation systems
The Surprise library, which stands for Simple Python Recommendation
System Engine, is an easy-to-use Python library designed for building and
evaluating recommendation systems. It provides various algorithms for
collaborative filtering, content-based filtering, and hybrid recommendation
systems. Surprise is particularly popular for its implementation of
traditional recommendation algorithms and evaluation metrics. Its key
Easy-to-use Interface: Surprise provides a user-friendly environment for
implementing recommendation algorithms, making it accessible to
developers and researchers alike.
Support for Various Algorithms: The library offers implementations of
various recommendation algorithms, including collaborative filtering,
content-based filtering, and hybrid methods. This empowers developers to
select the algorithm that best fits their particular use case.
Evaluation Metrics: Surprise includes built-in evaluation metrics for
assessing the performance of recommendation models. This simplifies the
process of comparing different algorithms and tuning hyperparameters.
Integration with Data Sets: It provides seamless integration with popular
data sets such as MovieLens, allowing users to quickly get started with
building recommendation systems without the need for extensive data
Advantages of Surprise over scikit-learn for
Recommendation Systems
Compared to traditional machine learning libraries like scikit-learn,
Surprise is specialized for recommendation tasks, offering dedicated
functionality and optimized implementations for these use cases. While
scikit-learn is a versatile machine learning library suitable for various
tasks, Surprise offers specialized support for recommendation systems with
dedicated functionality, evaluation metrics, and optimizations. For
developers focusing specifically on building recommendation systems,
Surprise provides a streamlined workflow and comprehensive tool set for
developing, evaluating, and deploying effective recommendation models.
Specialization: Surprise is tailored specifically for recommendation
systems, offering dedicated functionality and optimizations for these tasks.
This specialization allows developers to focus on building effective
recommendation models without the need for extensive customization or
preprocessing.
Simplified Workflow: Surprise provides a high-level interface for building
recommendation systems, streamlining the development process by
abstracting away low-level implementation details. This simplicity
accelerates prototyping and experimentation with recommendation
Evaluation Metrics: Surprise includes built-in evaluation metrics
designed specifically for recommendation tasks, enabling users to assess
the performance of recommendation models accurately. These metrics
provide insights into model effectiveness and guide algorithm selection and
Dataset Integration: Surprise offers seamless integration with popular
recommendation datasets like MovieLens, facilitating dataset loading and
preprocessing for recommendation tasks. This integration reduces the
overhead of data management and allows users to focus on model
Optimized for Sparse Data: Recommendation datasets often exhibit
sparsity, where users interact with only a small fraction of items. Surprise
includes optimizations for handling sparse data efficiently, ensuring
scalability and performance even with large-scale datasets.
Primary Use Cases of Surprise Library in
Recommender Systems
Surprise library finds primary use in different types of recommender
Matrix Factorization Methods
Matrix factorization methods, like Singular Value Decomposition (SVD)
and Non-negative Matrix Factorization (NMF), are commonly used in
collaborative filtering-based recommendation systems. Surprise provides
implementations of these algorithms, allowing developers to build accurate
and scalable recommendation systems based on user-item interactions. For
instance, the NMF algorithm can be used to suggest articles to users
depending on their past reading activities.
Neighborhood Methods
Neighborhood-based methods, such as K-Nearest Neighbors (KNN), are
another popular approach in recommendation systems. These methods rely
on finding similar users or items based on predefined similarity metrics.
Surprise's KNN algorithm, as demonstrated in the example below, allows
developers to build recommendation systems that leverage the preferences
of similar users or items to make personalized recommendations.
Practical Examples Demonstrating the Use of
Surprise Library for Recommendation Tasks
Example: Book Recommendation System using Surprise for KNN and
K-Nearest Neighbors (KNN) is a simple and intuitive algorithm used for
both classification and regression tasks. In the context of recommendation
systems, KNN is applied as a collaborative filtering technique. The
algorithm works by finding the K most similar users or items to a target
user or item based on similarity metrics such as cosine similarity or
Pearson correlation coefficient. It then predicts the rating or preference of
the target user for an item by aggregating the ratings of the nearest
Non-negative Matrix Factorization (NMF) is a matrix factorization
technique commonly used in recommendation systems. It decomposes the
user-item interaction matrix into two lower-dimensional matrices
representing users and items' latent features. NMF is particularly useful for collaborative filtering tasks, where it learns latent preferences and features
from the observed ratings data.
In this example, we will create a book recommendation system using the
Surprise library with the KNN and NMF algorithms.
1. Installing Surprise Library
Note: It is recommended to execute the following code snippet in
subsequent labs instead of the prior Week 2 lab environment. The surprise
library will work in Python version 2.7.
    !pip install scikit-surprise
This line installs the Surprise library using pip. Surprise can be installed
via pip or conda
2. Importing Required Libraries
    from surprise import Dataset, Reader
     from surprise.model_selection import train_test_split
    from surprise import KNNBasic
     from surprise import NMF
     from surprise import accuracy
This code imports necessary modules and classes from the Surprise library,
including Dataset, Reader, train_test_split, KNNBasic, and
3. Loading the Data Set
    # Load the dataset
    data = Dataset.load_builtin('ml-1m')
This code loads the MovieLens 1M data set, which contains 1 million
movie ratings, using the built-in method load_builtin().
4. Defining Reader and Train-Test Split
     import pandas as pd
    # Load the dataset into a Surprise Dataset object
    data = Dataset.load builtin('ml-1m')
    # Define the Reader object
    reader = Reader(rating_scale=(1, 5))
    # Optionally, convert the Surprise Dataset object into a pandas DataFrame
    df = pd.DataFrame(data.raw_ratings, columns=['user_id', 'item_id', 'rating', 'timestamp'])
    # Drop the 'timestamp' column if it's not needed
   df = df.drop(columns=['timestamp'])
# Load the data into Surprise format using load_from_df
    data = Dataset.load_from_df(df[['user_id', 'item_id', 'rating']], reader)
    # Split the data into test and train sets
    trainset, testset = train_test_split(data, test_size=0.2)
```

Exploring Surprise Library and

• Describe the Surprise library for recommendation systems

• List the advantages of Surprise over scikit-learn for recommendation

• Explain the primary use cases of Surprise library in recommender

KNN Model

Objectives

Estimated time: 15 Minutes

After this reading, you will be able to:

In this code: • A Reader object is defined to specify the rating scale of the dataset. In this case, the ratings are on a scale from 1 to 5. The Reader object allows you to specify the rating scale used in the dataset. This is important because different datasets may use different rating scales. For example, some datasets may use a scale from 1 to 5, while others may use a scale from 1 to 10. By specifying the rating scale, the Reader object ensures that the Surprise library interprets the ratings correctly during model training and evaluation. • Optionally, the Surprise Dataset object is converted into a pandas DataFrame for further manipulation. The raw ratings are extracted from the Dataset object and stored in the DataFrame with columns: user_id', 'item_id', 'rating', and 'timestamp'. • The timestamp column, which may not be needed for the recommendation task, is dropped from the DataFrame. • The data is loaded back into Surprise format using the load_from_df method. This method takes the DataFrame and the Reader object as • The data is split into training and test sets using the train_test_split function from the Surprise library. Here, 20% of the data is used for testing. 5. Training the KNN Model # Build the KNN model sim_options = {'name': 'pearson', 'user_based': True} model_KNN = KNNBasic(sim_options=sim_options) # Train the model model_KNN.fit(trainset) This code builds a KNN model with Pearson correlation similarity and user-based approach, then trains the model on the training data. sim_options parameter is a dictionary that specifies the options for computing similarities between users or items. In this dictionary:

name specifies the similarity metric to use. Here, 'pearson' correlation coefficient is chosen. Pearson correlation assesses the linear relationship between two variables, specifically in this context, between users' ratings.

for computing similarities. When set to True, the model computes

model_KNN is an instance of the KNNBasic class, which is a basic

By default, the KNNBasic algorithm considers 40 neighbors if not

a parameter to specify similarity options.

setting the 'k' parameter in sim_options.

6. Build and Train NMF Model

Build the NMF model
model NMF = NMF()

between items.

similarities between users; when set to False, it computes similarities

user_based specifies whether to use a user-based or item-based approach

implementation of the KNN algorithm in Surprise. It takes sim_options as

specified otherwise. This default number of neighbors can be changed by

Train the model
model_NMF.fit(trainset)

An NMF (Non-negative Matrix Factorization) model is instantiated and trained on the training set.

7. Making Predictions and Evaluating with KNN model

Make predictions on the test set predictions = model_KNN.test(testset)
Evaluate the model
accuracy.rmse(predictions)

Finally, these lines make predictions on the test set using the trained model and evaluate the models performance using RMSE.

Code Output

RMSE: 0.9608
0.9607743656303924

The output provided indicates the Root Mean Squared Error (RMSE) of the KNN model's predictions on the test set. RMSE:0.9608 indicates the RMSE value calculated for the predictions made by the KNN model on the test set. RMSE serves as a widely adopted metric for assessing the precision of a model's predictions, notably in regression tasks such as recommendation systems. It quantifies the average disparity between the predicted ratings and the actual ratings, where smaller values signify superior predictive accuracy.

8. Making Predictions and Evaluating NMF model

Make predictions on the test set with the NMF model predictions = model_NMF.test(testset)

Evaluate the model
accuracy.rmse(predictions)

Finally, these lines make predictions on the test set using the trained model

and evaluate the models performance using RMSE.

Code Output

RMSE: 0.9167

The output provided indicates the Root Mean Squared Error (RMSE) of the NMF model's predictions on the test set. RMSE:0.9167 indicates the RMSE value calculated for the predictions made by the NMF model on the test set. RMSE serves as a widely adopted metric for assessing the precision of a model's predictions, notably in regression tasks such as recommendation systems. It quantifies the average disparity between the predicted ratings and the actual ratings, where smaller values signify superior predictive accuracy.

Conclusion:

The Surprise library is a valuable resource for developers and researchers involved in recommendation systems, providing various features and tools for building, evaluating, and deploying advanced recommendation models.

Whether it's building collaborative filtering models with KNN or leveraging matrix factorization methods like NMF, Surprise facilitates the development of personalized and accurate recommendation systems across various domains and applications.

Author(s)

Pratiksha Verma

Other Contributors

Malika Singla

