← Go Back to Recommendation Systems

Course Content

New Package Introduction - Recommendation Systems Part 1

Let's now go through some of the common functions used in the LVC case studies.

Defaultdict

One of the issues that we face with Python dictionaries is the frequent missing KeyError. In those cases where dictionaries are frequently used, it can be a hindrance to deal with this error. To overcome this, there is a Python library called **collections** that implements specialized container types. One such example is the Python **defaultdict** type, which is an alternative to dict that's specifically designed to help with missing keys.

To import it the below code can be used.

```
from collections import defaultdict
```

To get more information on this function you can refer to this link.

Surprise

Surprise is an easy-to-use Python library that allows us to quickly build different kinds of recommender systems without having to write code from scratch. It's a single library that contains all the methods and functionalities to build many types of recommendation systems in Python. It has methods to build Collaborative Filtering-Based, Clustering-Based, and Model-Based recommendation systems. The following code is needed to install the package.

```
# Installing the surprise library
!pip install surprise
```

To get more information about this package you can refer to this link.

Reader

The **Reader** function in the Surprise library creates a different class to prepare the required format of the dataset to build recommendation systems. To import it, the below code can be used:

```
from surprise.reader import Reader
```

It is used in the below way to format a dataset.

```
# Instantiating Reader scale with expected rating scale
reader = Reader(rating_scale = (0, 5))

# Loading the rating dataset
data = Dataset.load_from_df(rating[['userId', 'movieId', 'rating']], reader)

# Splitting the data into train and test datasets
trainset, testset = train_test_split(data, test_size = 0.2, random_state = 42)
```

Here "Dataset" is used to format the pandas DataFrame into what's required in Surprise. The train_test_split is used to split the dataset into train and test sets.

To get more information about this function you can refer to this link.

KNNBasic

KNNBasic is an algorithm associated with the Surprise package. It is used to find the desired similar items among a given set of items.

The following code helps in importing this function.

```
from surprise.prediction_algorithms.knns import KNNBasic
```

The below code demonstrates one specific use of this function.

```
sim_user_user = KNNBasic(sim_options = sim_options, verbose = False, random_state = 1)
```

Here, sim_options contains the similarity options that need to be considered when measuring the similarity between two users or items. It contains the type of similarity measure we want to use, which may be a cosine similarity or some distance-based similarity like Manhattan distance or Euclidean distance.

To get more information about this function, you can refer to this link.

GridSearchCV

This is a special method in the Surprise library that is used to perform hyperparameter tuning in order to find the best set of hyperparameters. To import this, the below code can be used:

```
from surprise.model_selection import GridSearchCV
```

The following code shows one use case of this function.

```
gs = GridSearchCV(KNNBasic, param_grid, measures = ['rmse'], cv = 3)
```

Here, the param_grid is the set of values for each hyperparameter that needs to be optimized. The measures display the type of error that we consider to find the best hyperparameter set, cv tells us about the number of cross validations utilized.

To get more information about this function, you can refer to this link.

SVD

It is a function used to perform singular value decomposition over a matrix. It provides methods to create matrix factorization based Recommendation Systems. To import it, the below code can be used:

```
from surprise.prediction_algorithms.matrix_factorization import SVD
```

The following code shows one of the applications of this function.

```
# Using SVD with matrix factorization
svd = SVD(random_state = 1)
# Training the algorithm on the training dataset
svd.fit(trainset)
# Let us compute precision@k, recall@k, and f_1 score with k = 10
precision_recall_at_k(svd)
```

To get more information about this function, you can refer to this link.

Happy Learning!