In this video, we will talk about how to partition a dataset to train a recommender system.

Before we start talking about data set partitioning, we need to recap how recommender systems work from a high level point of view.

A recommender algorithm is

based on two mathematical functions,

f and g. One possible input of function f is the URM,

the User Rating Matrix.

Each row corresponds to a user,

and each column corresponds to an item.

At the intersection, there is

the rating of that user for that item.

The matrix is very sparse,

which means with a lot of zeros,

since each user only rates a few items.

The output is what we call the model,

it is a representation of the user preferences.

The function g takes us,

inputs the model just created,

and the user profile in order to estimate some ratings.

**Next Slide**

Let's consider an example.

The function f takes as input in URM that

contains several information about users and movies.

It builds the model according to the fact that

the movie Top Gun is similar to the movie The Avenger.

We know that the user,

Timy really likes Top Gun.

What would happen if we combine these two information,

the model and the user profile?

We would end up with a recommendation

for the user, Timy.

According to our recommender system,

he should watch the movie The Avengers.

At the end, we have to compare

the recommendation with the true opinion of the user.

**Next Slide**

Let's now analyze one of

the most used partitioning methods.

We are discussing about the whole out of

ratings technique according to which we

extract some interactions from

the data set and keep them away.

The goal is to use them to evaluate

the quality of our recommendations,

once we have estimated the ratings.

As shown in the table,

we have a URM which is quite sparse.

We extract some ratings highlighted in green in

the image and calls that

the set made up of these ratings.

This set can be built in many ways.

For example, by randomly choosing the 20% of the ratings,

or by selecting only one interaction per user.

The last mentioned technique is called leave one out.

The Z set is called test set,

as it will allow us to test

the goodness of our estimations.

Then we call all the remaining ratings X,

and we use them as the input for the function

f.

Since we build the model with the X set,

this one is named, training set.

Finally, we select the Y set as shown in the matrix,

which represents the user profile

set that was introduced before.

To make some recommendations,

we need to feed the function g

with the model and the set Y.

It is important to say

that with the whole layout of the ratings,

the user profiles belong to the training.

Mathematically, Y is a subset of X.

At the end, we have to compare

the estimated results with the ones

that are present in the test set Z.

This approach is the hold out of the rating approach.

Now lets move towards overfitting