**Problem Statement**

Iris is Pluralsight’s learning intelligence platform, an innovative and unique user experience, whose aim is to use data to create a smarter, personalized learning journey.

In this project I build a recommender system, which recommends a user courses based on his preferences.

Also, displays users who are similar to a particular user.

**Answers to the given questions:**

1. Tell us about your similarity calculation and why you chose it.

I used the technique called model-based collaborative filtering systems. With these systems you build a model from user ratings, and then make recommendations based on that model. This offers a speed and scalability that's not available when you're forced to refer back to the entire dataset to make a prediction.

I calculated the **Pearson r correlation coefficient**, for every user pair in the resultant matrix. With correlation being based on similarities between user preferences.

2. We have provided you with a relatively small sample of users. At true scale, the  number of users, and their associated behavior, would be much larger. What  considerations would you make to accommodate that?

We can also build a microservice that exposes a REST API that receives input and can predict an outcome. Basically, instead of running an ad-hoc batch program to do predictions, or use streaming, you could build an API that can be used by other systems:

Docker would be used to build the container

The container is stateless and can be horizontally scaled over multiple nodes to grow with the demand of predictions

Even online learning could be added to the capabilities. In that case the model’s data would have to be persisted outside the container (e.g. in an object store)

Python could be used together with spark or scikitlearn to answer to prediction requests. The API could be exposed using Flask. All these libraries/packages will have to be built within the container. It helps in running projects on different servers, thereby helping in tracking and collaboration of machine learning projects.

3. Given the context for which you might assume an API like this would be used, is  there anything anything else you would think about? (e.g. other data you would like to collect)

User feedback should be collected on each course. For example, after completion a particular course the user should be asked to rate his experience out of 10.

We can use these ratings, by suggesting new users more of these higher rated courses.

Also, we try to log how frequently a user is accessing a particular course and get the measure of how much interesting a particular course is.

Certain courses tend to get seasonal demand and these can be identified using time series analysis. These courses can be recommended more in their peek season.

**Machine Learning Theory and Background:**

I used the technique called **model-based collaborative filtering systems.**With these systems you build a model from user ratings, and then make recommendations based on that model. This offers a speed and scalability that's not available when you're forced to refer back to the entire dataset to make a prediction.

In the demo for this segment, you're going see truncated singular value decomposition. You're also going to see something called a utility matrix. Utility matrix is also known as user item matrix. These matrices contain values for each user, each item, and the rating each user gave to each item.

Another thing to note is that utility matrices are sparse because every user does not view every item. Actually, only a few users provide views for a few courses. So in these matrices, we are likely to see mostly null values.

**SVD** is a linear algebra method that you can use to decompose a utility matrix into three compressed matrices.

It's useful for building a model-based recommender because you can use these compressed matrices to make recommendations without having to refer back to the complete and entire dataset. With SVD, you uncover latent variables. These are inferred variables that are present within and affect the behavior of a dataset. Although these variables are present and influential within a dataset, they're not directly observable.

We'll calculate the Pearson r correlation coefficient, for every movie pair in the resultant matrix. With correlation being based on similarities between user preferences.