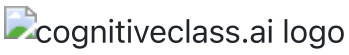


Build a course recommender app with streamlit

Estimated time needed: **60** minutes

Objectives

After completing this lab, you will be able to:

- Understand and run the provided skeleton recommender app
- Extend the recommender app by completing model training and prediction methods

Lab environment setup and preparation

If you plan to develop the app in your local development environments, you need to make sure you have the following:

- Your favorite IDE or text editor
- Python 3.7 - Python 3.9
- PIP
- We recommend you create a virtual environment for building the Streamlit apps.
- Click here to learn more about Python virtual environments.

Personalized Course Recommender app

Download and unzip the provided recommender app template:

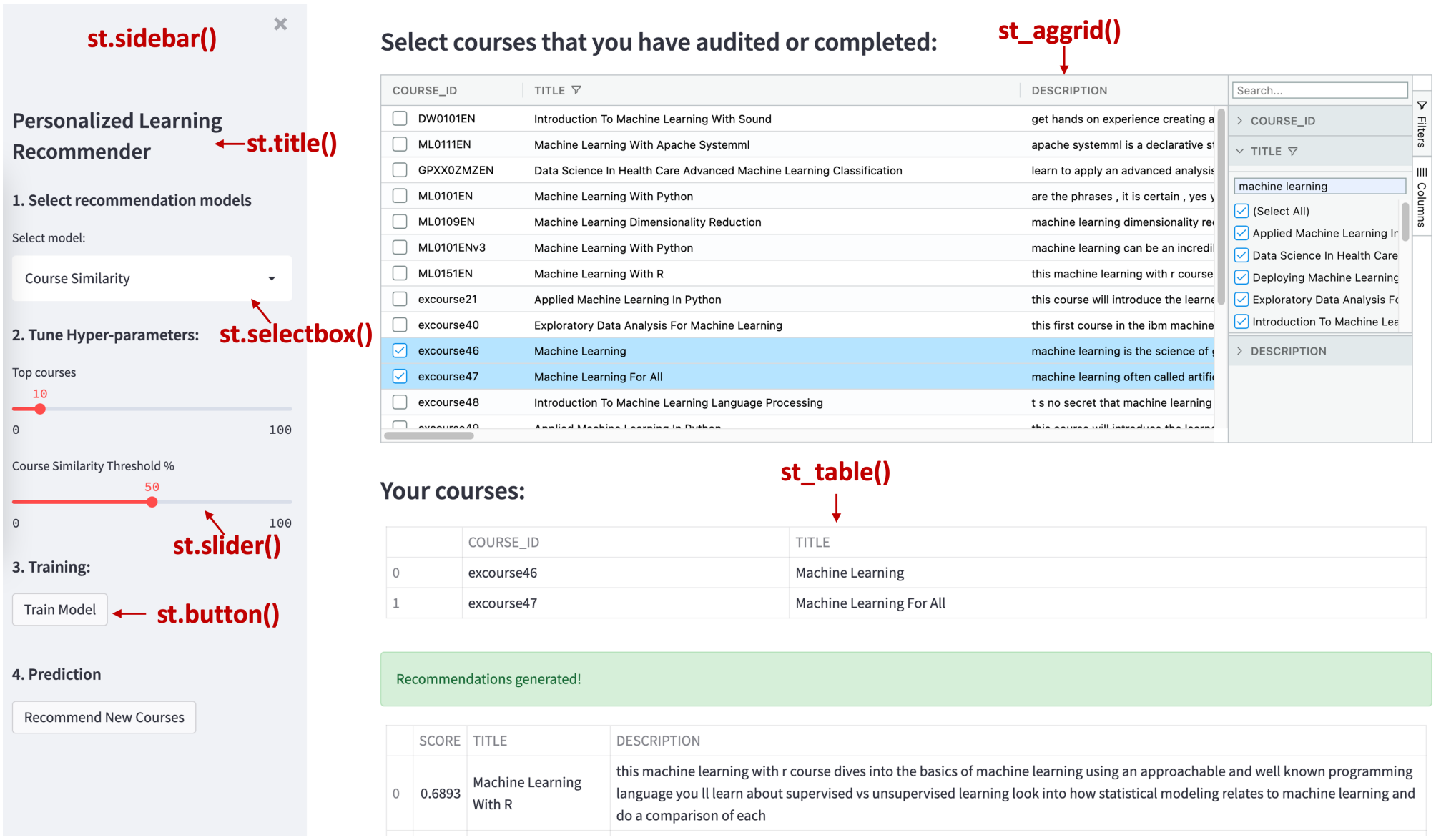
```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-ML321EN-SkillsNetwork/labs/module_5/app.zip
```

```
unzip app.zip
rm app.zip
```

In the unzipped app folder, we can see the following files and directories:

- `data` folder contains related datasets such as ratings, course content, similarity matrix, etc.
- `backend.py` contains backend machine learning functions such as loading datasets, model training, predictions, etc.
- `recommender_app.py` is the main `Streamlit` Python script mainly implements the user interactions with the backend machine learning code
- `requirements.txt` specifying the required Python libraries for the app. You can add more libraries whenever needed

If we run the sample app, you should see the following page pops-up:



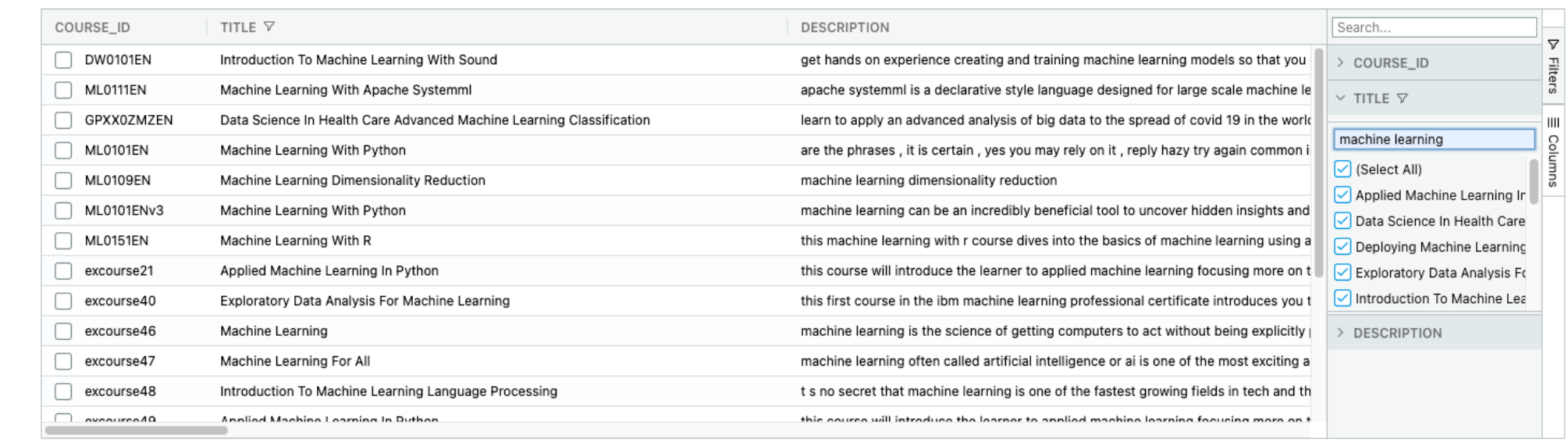
We also labelled the `Streamlit` widgets on the above screenshot, and you can check their details in the source code.

If take a look at the course selection table, here we used a `streamlit-aggrid` plugin to create a st_aggrid() widget to better interact with the dataframes. For example, you can easily select, filter, and search the courses you want.

The user interactions in this app are pretty straightforward:

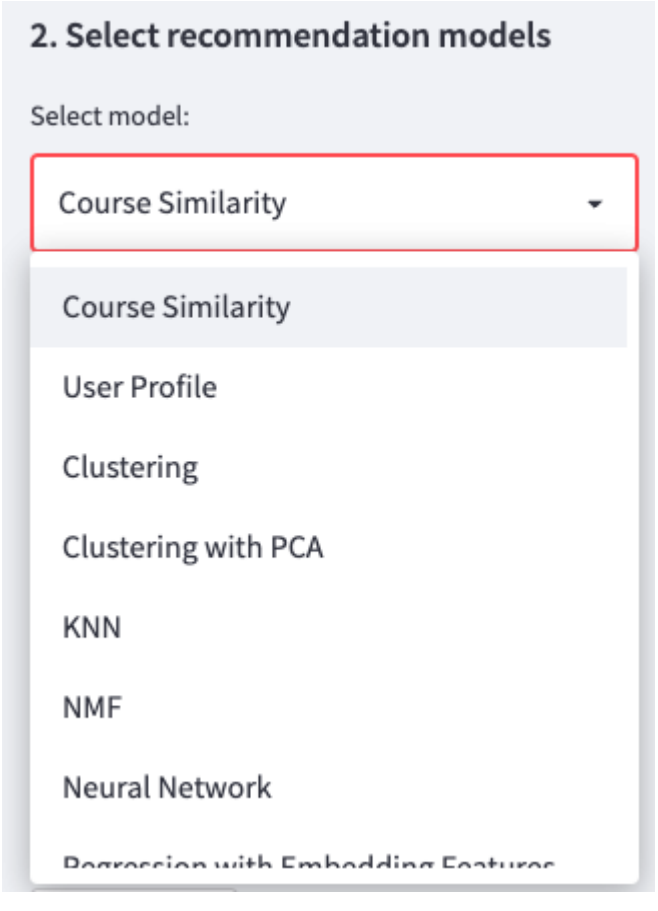
1. As a test user, you first need to search and select the courses you have audited or completed. You can filter courses

via the Filters and Columns menu items on the right side of the `aggrid`.

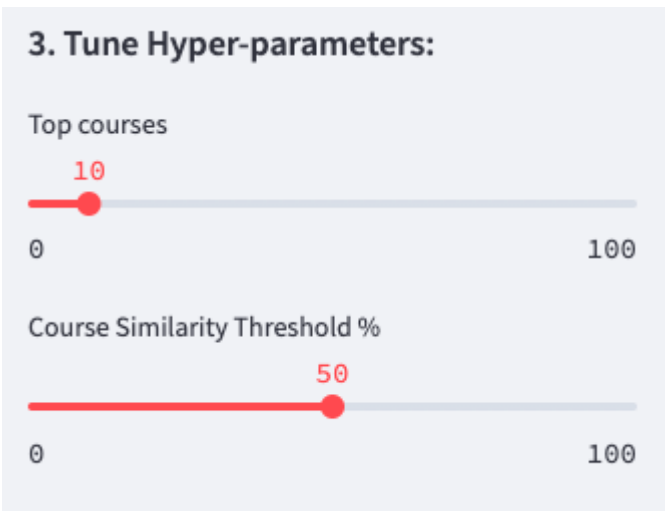


3. Then, once you have created a course enrollment list, you can choose which recommender model you want to use to

generate course recommendations. Here in this sample app, we only provided the course similarity based model for demo purpose. You will need to port the model implementation from the notebooks you have completed in the previous labs

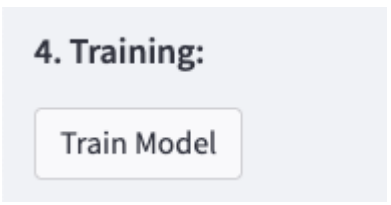


4. Depends on your model, you may add different hyper-parameters UI widgets and determine the hyper-parmeters for training and prediction



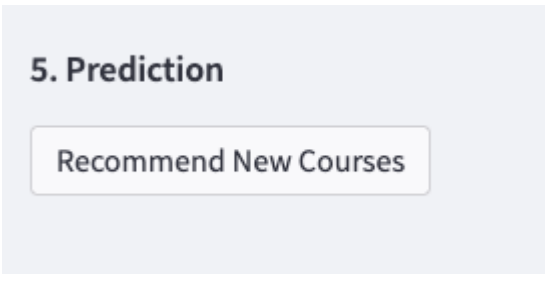
4. Depends on your model, you may need to re-train it with new data been added in Step 1.

You will need to port the model training code from the notebooks you have completed earlier.



5. Once you have your model trained, you need to determine the test data for the model to make predictions.

You need to figure out all the unseen/unselected for the test user and estimate the rating using the model.



Extend the recommender app by completing model training and prediction methods

Now you can extend this app by porting the model training and prediction code you have done in the previous notebooks. More specifically, you need to modify three places in the app source code:

1. Add more hyper-parameters widgets.

In the `recommender_app.py`, find the code area below comment `# Hyper-parameters for each model`. This is the area where we render different UI widgets for receiving hyper-parameter values.

For example, we added two sliders bar for only showing the top courses and the threshold to determine if two courses are similar or not.

```
params = {}
# Course similarity model
if model_selection == backend.models[0]:
    # Add a slide bar for selecting top courses
    top_courses = st.sidebar.slider('Top courses',
                                    min_value=0, max_value=100,
                                    value=10, step=1)
    # Add a slide bar for choosing similarity threshold
    course_sim_threshold = st.sidebar.slider('Course Similarity Threshold %',
                                             min_value=0, max_value=100,
                                             value=50, step=10)
    params['top_courses'] = top_courses
    params['sim_threshold'] = course_sim_threshold
```

All selected parameters will be stored in a Python dictionary called `params` and be passed into training and prediction functions.

```
params = {}
```

2. Add model-specific training code. If you find the code area under `# Training UI`, you should see the

following code snippet:

```
st.sidebar.subheader('4. Training: ')
training_button = st.sidebar.button("Train Model")
training_text = st.sidebar.text('')
# Start training process
if training_button:
    train(model_selection, params)
```

Basically, what it does is when you click the `training_button` button, it calls the `backend.train()` method via the `train()` method with a model name and hyper-parameters.

Depends on which model is selected, the actual training code can be different.

3. Add model-specific prediction code. If you find the code area under `# Prediction UI`, you should see the

following code snippet:

```
# Prediction UI
st.sidebar.subheader('5. Prediction')
# Start prediction process
pred_button = st.sidebar.button("Recommend New Courses")
if pred_button and selected_courses_df.shape[0] > 0:
    res_df = predict(model_selection, params)
    st.table(res_df)
```

Basically, what it does is when you click the `pred_button` button, it calls the `backend.predict()` method via the `predict()` method with a model name and hyper-parameters.

Depends on which model is selected, the actual prediction code can be different.

Reference

You can check the following links for more details about `Streamlit`:

- Streamlit documentation
- streamlit-aggrid

Summary

In this lab, you have learned how to extend the recommender template app to include more recommendation models you have built in the previous notebooks.

In addition, you are encouraged to modify and improve the template app to better serving and demonstrating your recommendation models.

Authors

Yan Luo

Other Contributors

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-03-22	1.0	Yan Luo	Created initial version of the lab