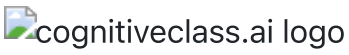

Introduction to `Streamlit`

Estimated time needed: **45** minutes

Objectives

After completing this lab, you will be able to:

- Describe the core concepts of `streamlit` framework
- Build your first `Streamlit` app

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. Note we are developing a Python app with script files so notebooks like Jupyter Notebook is not very suitable for developing app.
- 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.
- Download the sample app from here:

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

Install required Python libraries and test the app

- In your terminal, make sure you have Python installed or activated any Python virtual environments.

First cd to the unzipped `first_app` directory:

```
cd first_app
```

and install required Python libraries specified in `requirements.txt` file for this app:

```
pip3 install -r requirements.txt
```

First, let's run the first app by the following command:

```
streamlit run app.py
```

If you are on your local environment,

- Open your browser and enter URL `http://localhost:8501`

If you are using Skills Network Labs environment (TBD):

Introduction to `Streamlit`

`Streamlit` is an open-source app framework which allows data scientists and machine learning engineers to quickly build interactive machine learning and data analytics web apps. It is so easy to use that you could turn your Python data and modeling scripts into shareable web apps in minutes, and no web front-end experience required at all.

Core concepts

Streamlit allows you to write apps the same way as you write Python scripts.

- Streamlit reruns your entire Python script from top to bottom if you updated your source code or there is

a user interaction (like clicking a button)

- Streamlit can easily display data, charts, maps, which is similar to Jupyter notebook.
- Streamlit provides a set of UI Widgets like text label, button, slider, checkbox, selectbox, which you can use

to interact with your data and models

- Streamlit uses cache to avoid recomputing heavy functions such as loading large csv files.
- Streamlit supports multiple cloud deployment methods. Once you have built and tested a Streamlit app, you can easily deploy it on many cloud environments to showcase your

work

A simple example app

Now let's build a very simple Streamlit app together to better understand how Streamlit works.

- Open the `app.py` file and you can see an empty script file with only two imports `streamlit` and `pandas`
- We can now run the empty app to see. In the terminal, run the following command:

```
streamlit run app.py
```

and you should see an empty page.

- Then under comment #1, copy the following code snippet to add a Streamlit title widget:

```
st.title("This is a sample app")
```

- Refresh the page or click the `rerun` button on the page, and you should see a newly added title
- Under comment #2, copy the following code snippet to add a button title widget:

```
button1 = st.button("Click to show a dataframe")
print(button1)
```

As you can see in the console, the value of button1 is `False` which means it is not clicked. and if we click the button, The Streamlit app will re-run and the button1 becomes `True`

```
button1 = st.button("Click to show a dataframe")
print(button1)
```

Then, we can add the following code snippet to show a simple Pandas dataframe if button1 is clicked:

```
if button1:
    df = pd.DataFrame({
    'column1': [1, 2, 3, 4],
    'column2': [10, 20, 30, 40]
    })
    # Show the Pandas dataframe using st.dataframe() method
    st.dataframe(df)
    # Visualize the column1 series using st.line_chart() method
    st.line_chart(df['column1'])
```

Refresh the page and click the button, and you should see a dataframe is displayed as a table using `st.dataframe()` method and visualized as a line chart using `st.line_chart()`

- Next, under comment #3, let's add two sliders to receive some numerical values from users:

```
slider1 = st.slider("Slider1", min_value=1, max_value=10, value=1)
print(slider1)
slider2 = st.slider("Slider2", min_value=1, max_value=10, value=2)
print(slider2)
```

- Refresh the page and we can see two sliders are added. From the console, we can see the value of slider1

is 1 and value of slider2 is 2. If we scroll the sliders, their values will be updated accordingly (shown in the console).

- Lastly, under comment #4, we can add a text label to show the sum of two sliders:

```
# Create a streamlit subheader widget
st.subheader("The sum of slider1 and slider2 is: ")
st.text(slider1 + slider2)
```

Reference

You can check more details in here:

- Streamlit documentation

Summary

By now, you should have some basic understanding about how `Streamlit` works. Next, we can take a look at our course recommender app.

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