Lab 1: Iris Flowers as a Serverless ML System

Iris Flower, blue and yellow, ultra-wide-angle
created with **Midjourney**

Course Material: Prof Jim Dowling

Warmup

- Course Repository on Github
  https://github.com/featurestoreorg/serverless-ml-course/

- Use Conda or virtual environments to manage your python dependencies on your laptop

- If you are new to Machine Learning, run and understand the following programs:

  Click on links to open a Colab notebook
  - src/00-intro/green-apples-vs-oranges.ipynb
  - src/00-intro/red-and-green-apples-vs-oranges.ipynb

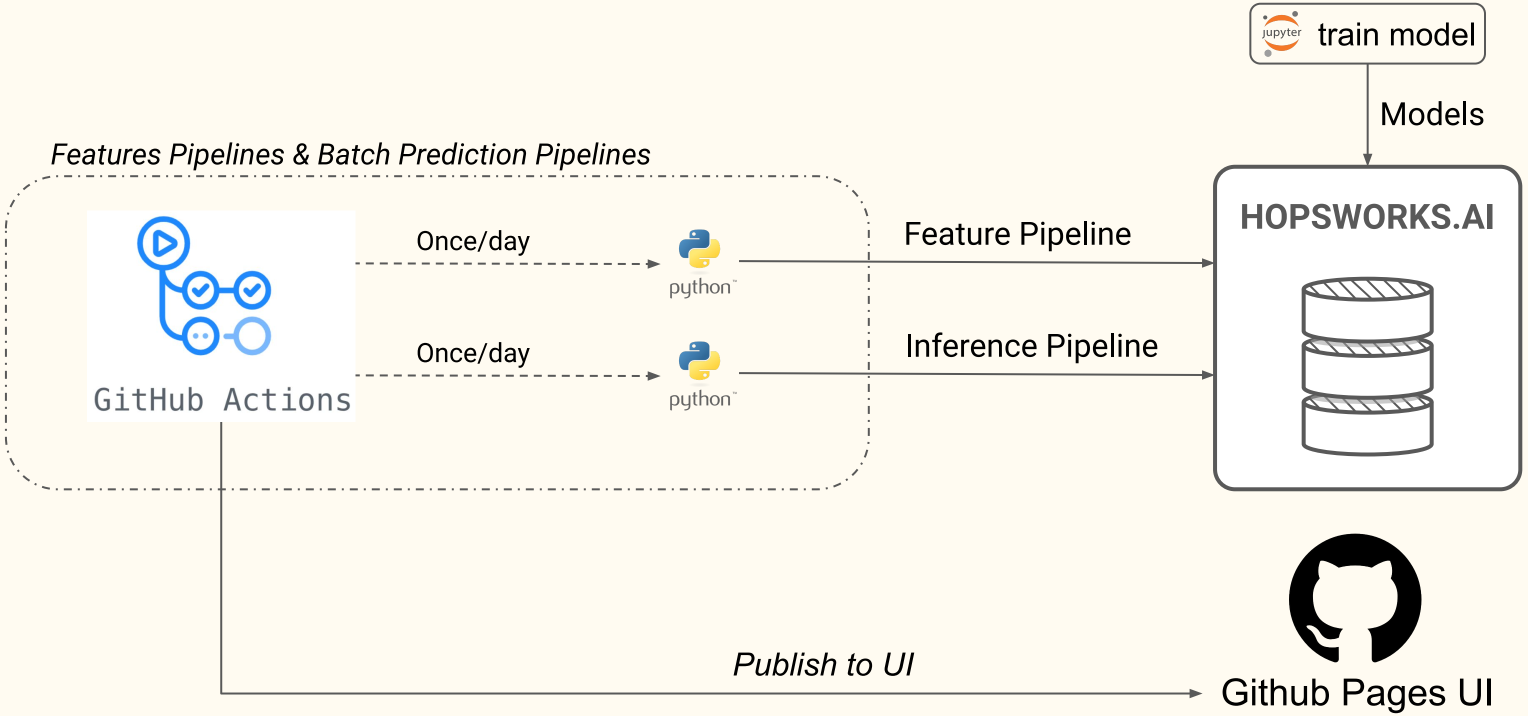
  You should run this streamlit example on your laptop:
  - conda activate <your_conda_env>
  - pip install streamlit
  - cd serverless-ml-course/src/00-intro
  - python -m streamlit run streamlit-example.py

What will we cover in this lab

- Case Study: Iris Flower Dataset

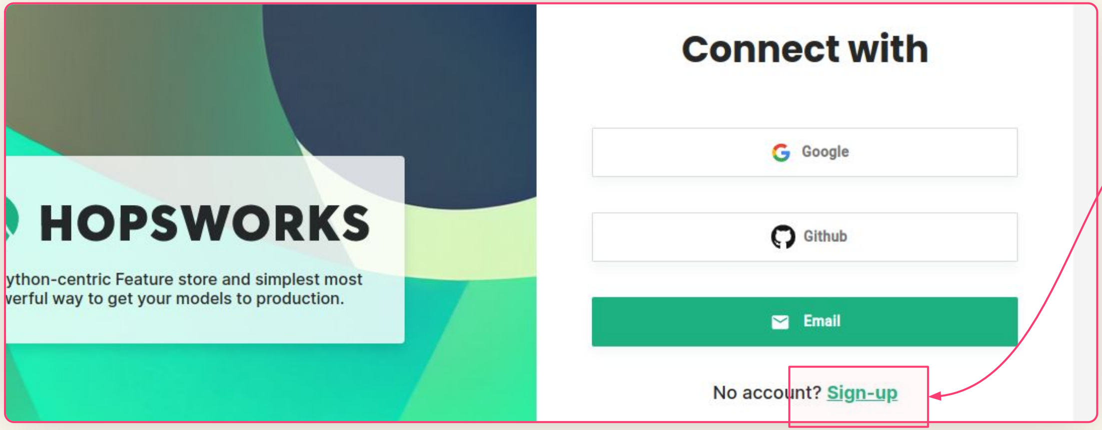
- **Steps**
  a. Add a user interface (Gradio) to an Iris Flower End-to-End ML Pipeline
  b. Refactor the ML Pipeline into feature, training, and inference pipelines.
  c. Use Github Actions to run a feature pipeline and a batch inference pipeline on a schedule.
  d. Add a Github Pages UI.

What we we will build today



Register and create an account on www.github.com (if you do not have an account already)

Register and Login to the Hopsworks Feature Store



1. First, create an account on https://app.hopsworks.ai

```
import hopsworks
proj = hopsworks.login()
fs = proj.get_feature_store()

Paste it here:

Copy your Api Key (first register/login): https://c.app.hopsworks.ai/account/api/generated
```

2.

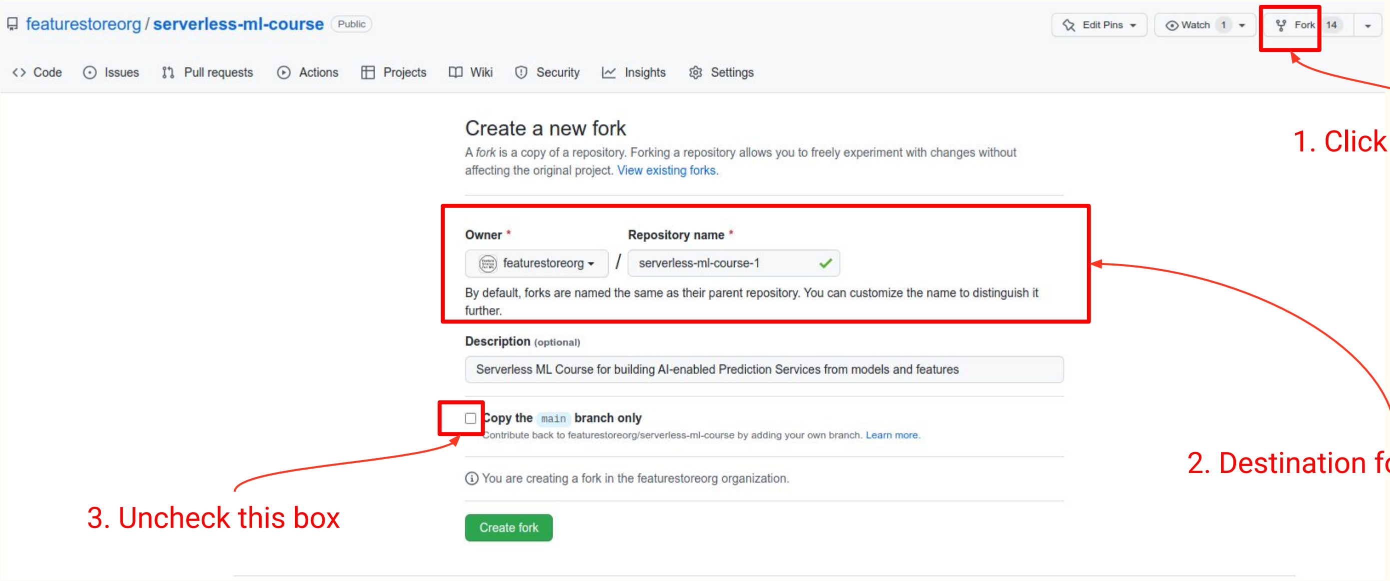
1.

1. Click on link, copy the API key
2. Paste the API key into this text box, then press return

```
Logged in to project, explore it here https://c.app.hopsworks.ai:443/p/398
```

1. When you have logged in, you can click on this link to open your project in a new tab

With your Github account, fork the serverless-ml-course repository

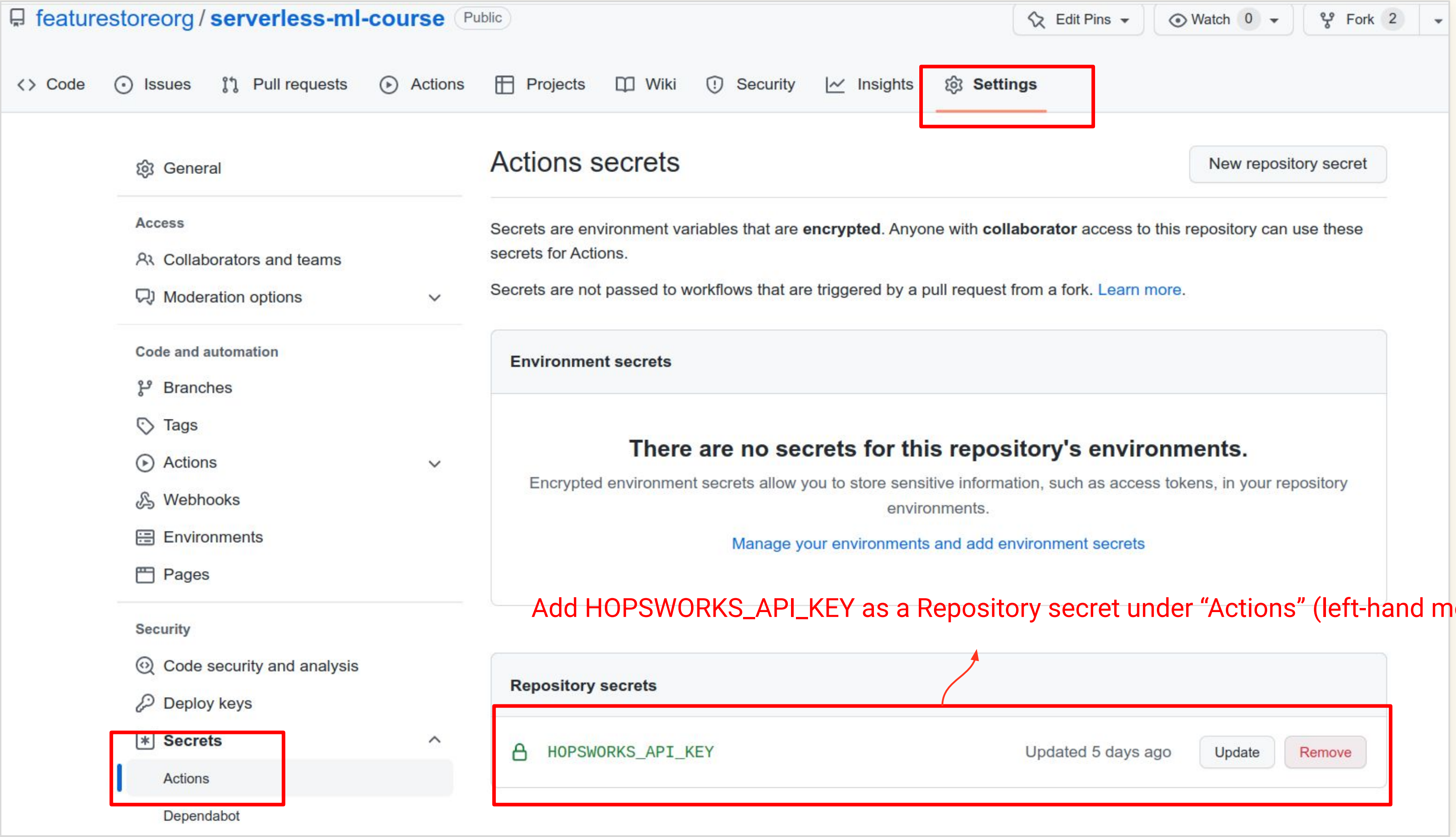


1. Click here to fork it

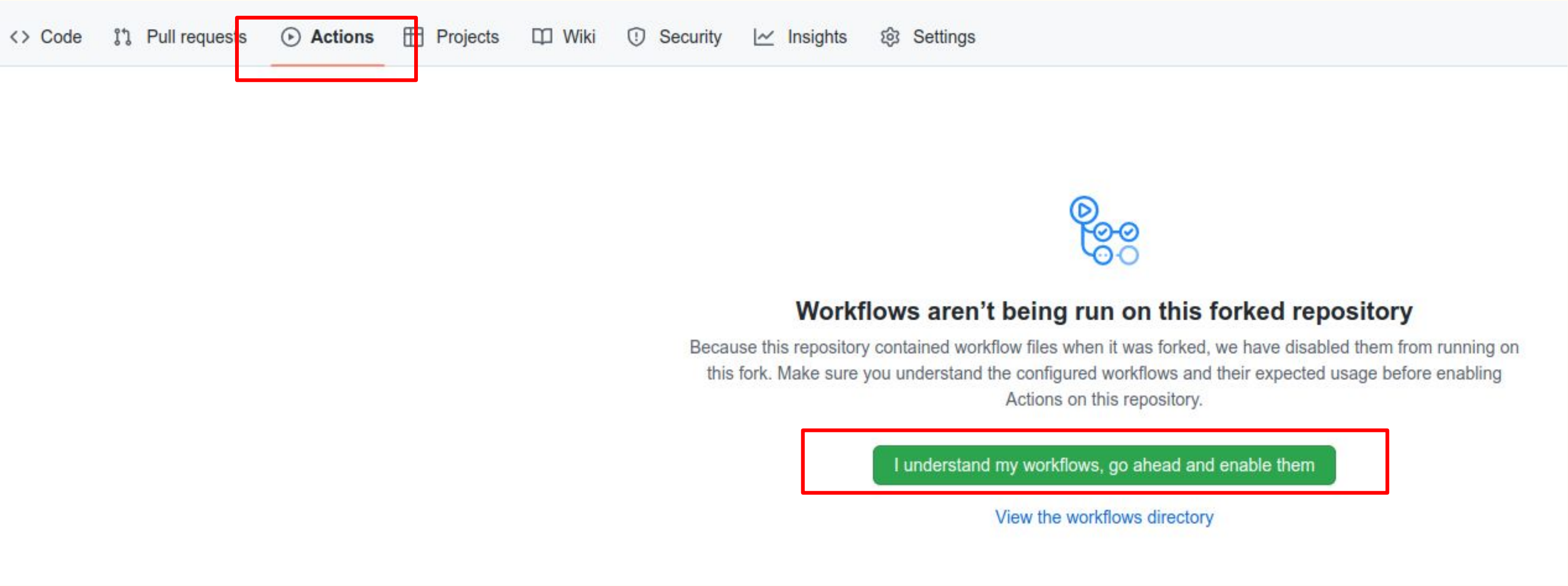
2. Destination for forked repo

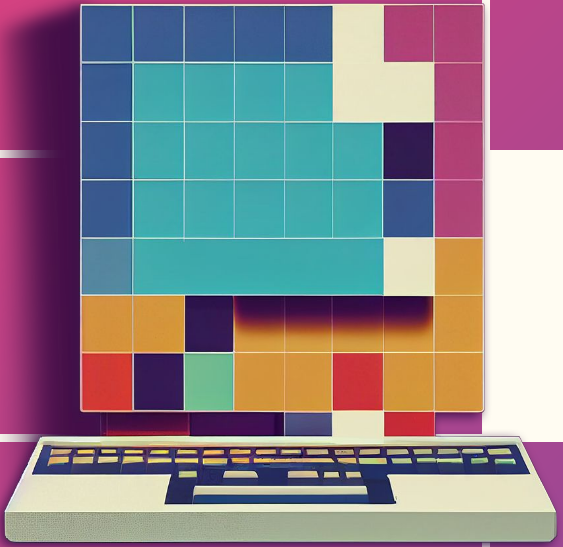
3. Uncheck this box

Add a HOPSWORKS_API_KEY as a secret for your Github Action



Enable the Github Actions for the forked Repository



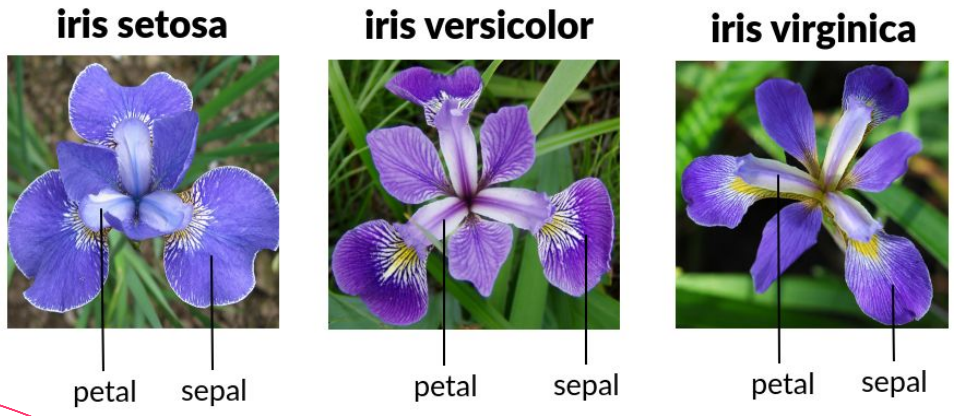


Case Study:
Iris Flower Dataset

Iris Flower Dataset

<u>Prediction Problem:</u>
Predict the *variety*, given the length and width of the petal and sepal.

This column is the Pandas Index



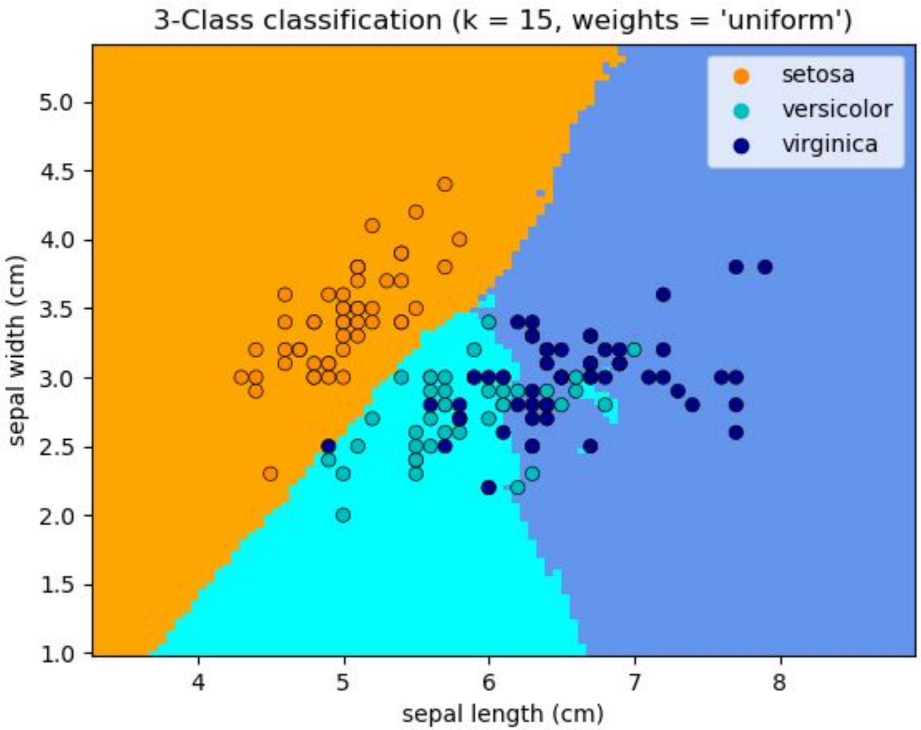
	sepal_length	sepal_width	petal_length	petal_width	variety
133	6.3	2.8	5.1	1.5	Virginica
48	5.3	3.7	1.5	0.2	Setosa
26	5.0	3.4	1.6	0.4	Setosa
134	6.1	2.6	5.6	1.4	Virginica
115	6.4	3.2	5.3	2.3	Virginica
15	5.7	4.4	1.5	0.4	Setosa
52	6.9	3.1	4.9	1.5	Versicolor

<u>Tabular Data</u>
Features
- sepal length
- sepal width
- petal length
- petal width

Target (label)
- variety

Classify Iris Flowers with K-Nearest Neighbors

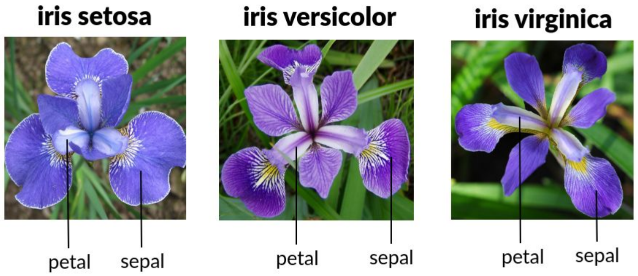
As we can see here two features (*sepal_length* and *sepal_width*) is not enough features to separate the three different varieties (*setosa, versicolor, virginica*).



[Image from https://scikit-learn.org/stable/modules/neighbors.html#classification]

Iris Flower End-to-end ML pipeline for Feature Engineering, Training, and Inference

1. Read raw data
2. Split into features/labels and train/test sets
3. Define Model Architecture
4. Train Model
5. Evaluate Model on the test set
6. Query the model using a Gradio UI



Let’s look at the Iris Flower Classification Problem

module-01/src/iris_end_to_end_ml_pipeline.ipynb

Case Study: The Iris Flower Dataset End-to-end ML pipeline

- Open this notebook in Colab: https://colab.research.google.com/github/featurestoreorg/serverless-ml-course/blob/main/src/01-module/iris_end_to_end_ml_pipeline.ipynb

- Loads Iris Flower Dataset into a Dataframe, random splits into train and test sets

- Visually analyzes the four input features (sepal/petal length/width)

- Trains a K-nearest neighbors classifier model on the train set

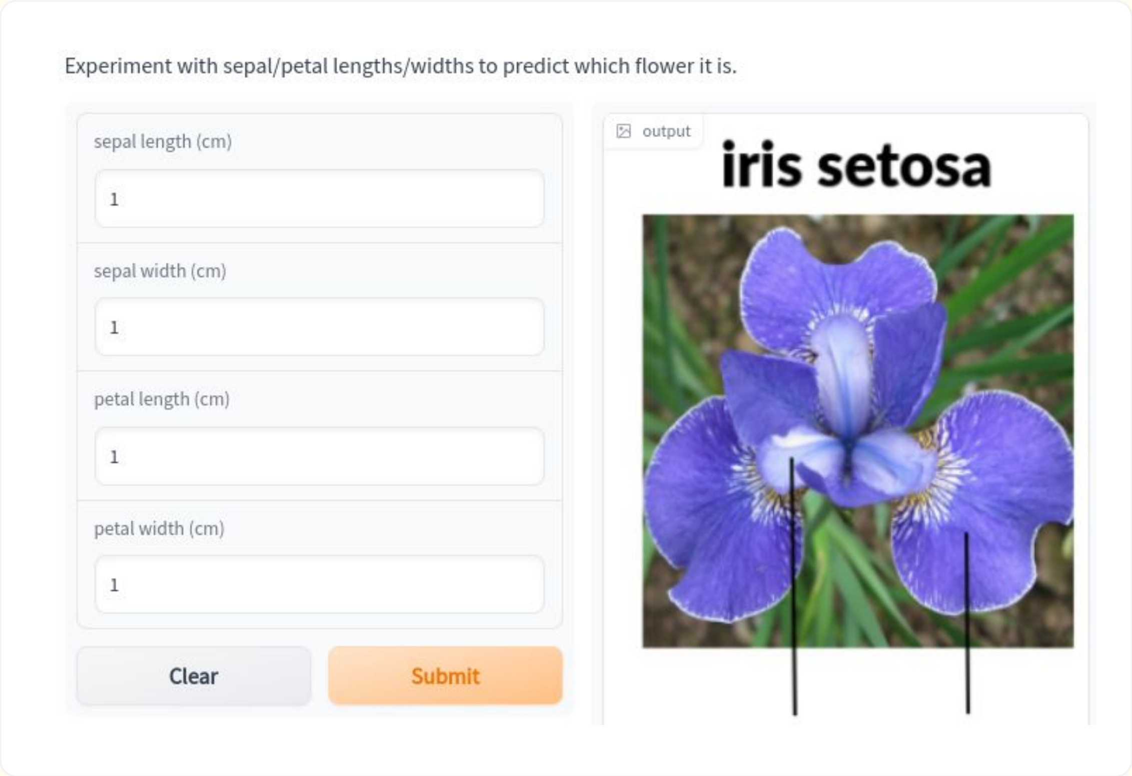
- Evaluates model performance on the test set

- Adds a UI using Gradio to interactively interact with the model
    - **Ask what-if questions, such as what type of Iris flower would you expect if we input these petal/sepal lengths/widths?**

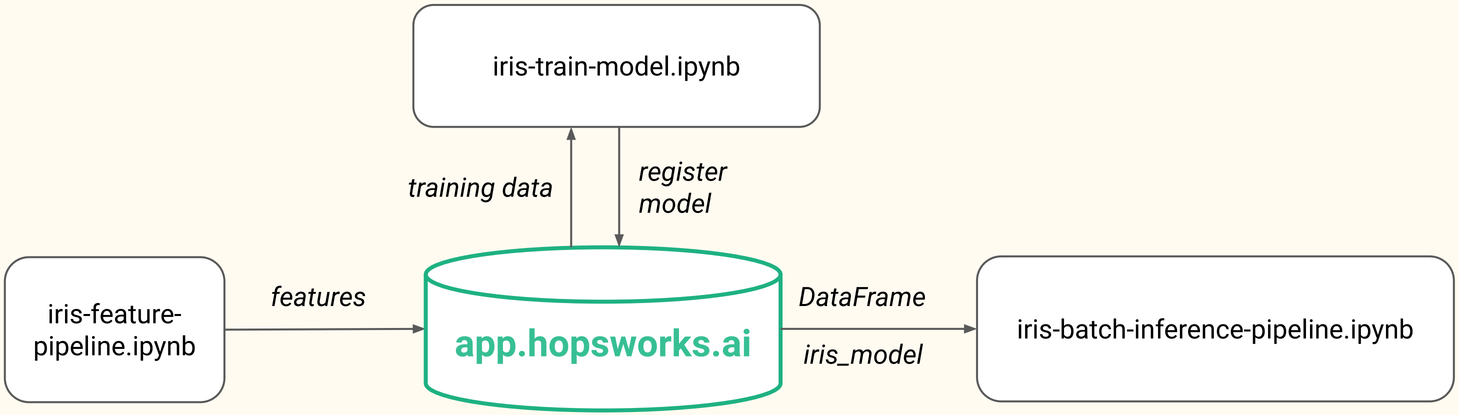
Communicate the value of your model with a UI (Gradio)

- Communicate the value of your model to stakeholders with an app/service that uses the ML model to make value-added decisions

- Here, we design a UI in Python with Gradio
  - Enables “predictive analytics” where a user can use the model to as “what-if” i had an Iris Flower with this sepal/petal width/length?



Step 2: Refactor the Monolith into Feature, Training, Inference Pipelines



Refactor Iris End-to-End ML Pipeline into:
Feature, Training, Batch Inference Pipelines

- New input data should continually arrive for a production system
    - We will create a synthetic data source to produce “Iris Flower” examples
- Our feature pipeline needs to process both the historical data (iris.csv) and the new input data
    - Our feature pipeline will run in either “BACKFILL” mode or in “NEW DATA” mode
    - BACKFILL mode will read data from iris.csv and write a DataFrame to the feature store
    - NEW DATA mode (BACKFILL == False) will read a DataFrame from our synthetic data source and write it to the feature store
- The model training pipeline will read training data from the feature store
- The batch inference pipeline will read new input data from the feature store and write its output to both Github Pages UI and the Feature Store
- We will run the Feature and Inference Pipelines on a schedule
    - Github Actions

Refactor Iris End-to-End ML Pipeline into:
Feature, Training, Batch Inference Pipelines

- New input data should continually arrive for a production system
    - We will create a synthetic data source to produce “Iris Flower” examples
- Our feature pipeline needs to process both the historical data (iris.csv) and the new input data
    - Our feature pipeline will run in either “BACKFILL” mode or in “NEW DATA” mode
    - BACKFILL mode will read data from iris.csv and write a DataFrame to the feature store
    - NEW DATA mode (BACKFILL == False) will read a DataFrame from our synthetic data source and write it to the feature store
- The model training pipeline will read training data from the feature store
- The batch inference pipeline will read new input data from the feature store and write its output to both Github Pages UI and the Feature Store
- We will run the Feature and Inference Pipelines on a schedule
    - Github Actions

Refactor Iris End-to-End ML Pipeline into:
Feature, Training, Batch Inference Pipelines

- New input data should continually arrive for a production system
  - We will create a synthetic data source to produce “Iris Flower” examples
- Our feature pipeline needs to process both the historical data (iris.csv) and the new input data
  - Our feature pipeline will run in either “BACKFILL” mode or in “NEW DATA” mode
  - BACKFILL mode will read data from iris.csv and write a DataFrame to the feature store
  - NEW DATA mode (BACKFILL == False) will read a DataFrame from our synthetic data source and write it to the feature store
- The model training pipeline will read training data from the feature store
- The batch inference pipeline will read new input data from the feature store and write its output to both Github Pages UI and the Feature Store
- We will run the Feature and Inference Pipelines on a schedule
  - Github Actions

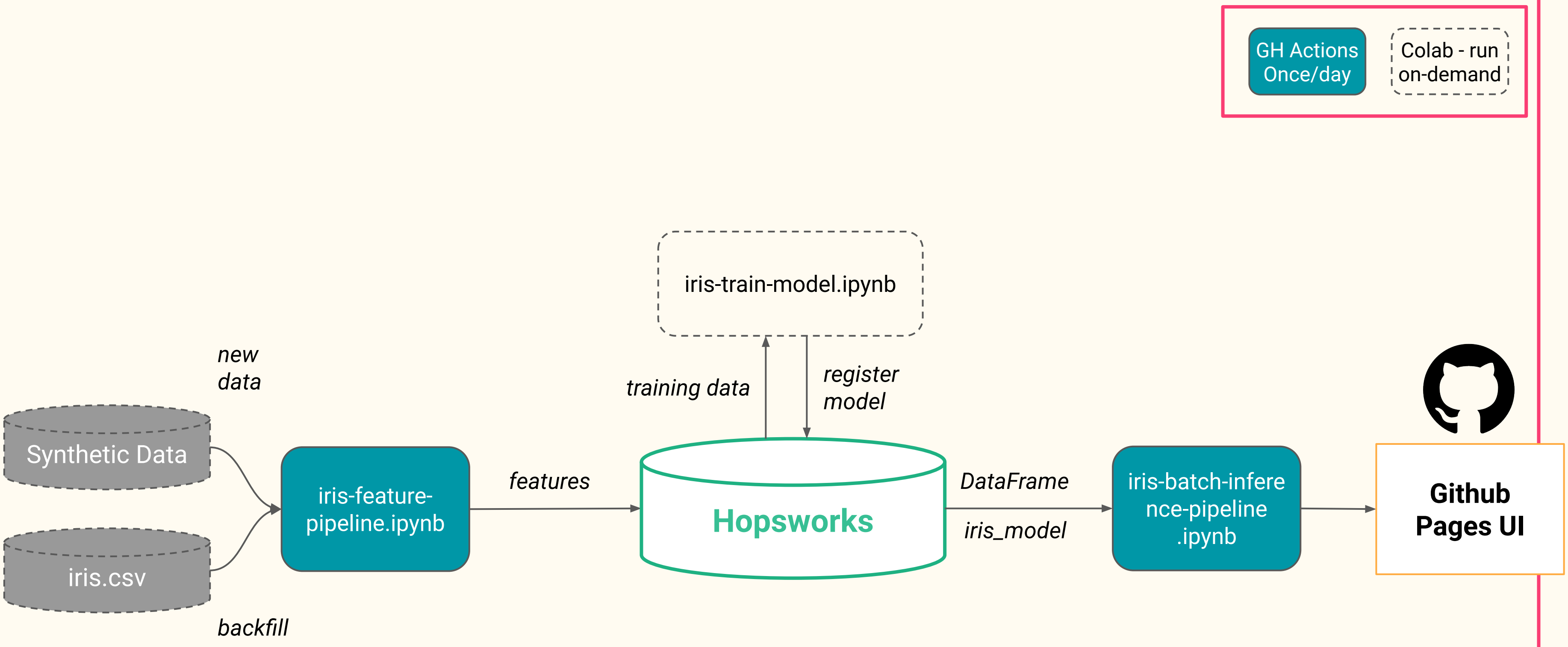
Refactor Iris End-to-End ML Pipeline into:
Feature, Training, Batch Inference Pipelines

- New input data should continually arrive for a production system
  - We will create a synthetic data source to produce “Iris Flower” examples
- Our feature pipeline needs to process both the historical data (iris.csv) and the new input data
  - Our feature pipeline will run in either “BACKFILL” mode or in “NEW DATA” mode
  - BACKFILL mode will read data from iris.csv and write a DataFrame to the feature store
  - NEW DATA mode (BACKFILL == False) will read a DataFrame from our synthetic data source and write it to the feature store
- The model training pipeline will read training data from the feature store
- The batch inference pipeline will read new input data from the feature store and write its output to both Github Pages UI and the Feature Store
- We will run the Feature and Inference Pipelines on a schedule
  - Github Actions

Refactor Iris End-to-End ML Pipeline into:
Feature, Training, Batch Inference Pipelines

- New input data should continually arrive for a production system
    - We will create a synthetic data source to produce “Iris Flower” examples
- Our feature pipeline needs to process both the historical data (iris.csv) and the new input data
    - Our feature pipeline will run in either “BACKFILL” mode or in “NEW DATA” mode
    - BACKFILL mode will read data from iris.csv and write a DataFrame to the feature store
    - NEW DATA mode (BACKFILL == False) will read a DataFrame from our synthetic data source and write it to the feature store
- The model training pipeline will read training data from the feature store
- The batch inference pipeline will read new input data from the feature store and write its output to both Github Pages UI and the Feature Store
- We will run the Feature and Inference Pipelines on a schedule
    - Github Actions

Iris Serverless Analytical ML System Architecture



Refactor Iris End-to-End ML Pipeline into:
Feature, Training, Batch Inference Pipelines

- New input data should continually arrive for a production system
    - We will create a synthetic data source to produce "Iris Flower" examples
- Our feature pipeline needs to process both the historical data (*iris.csv*) and the new input data
    - Our feature pipeline will run in either "BACKFILL" mode or in "NEW DATA" mode
    - BACKFILL mode will read data from iris.csv and write a DataFrame to the feature store
    - NEW DATA mode (BACKFILL == False) will read a DataFrame from our synthetic data source and write it to the feature store
- The model training pipeline will read training data from the feature store
- The batch inference pipeline will read new input data from the feature store and write its output to both Github Pages UI and the Feature Store
- We will run the Feature and Inference Pipelines on a schedule
    - Github Actions

Steps to build and run our serverless Iris Flower Analytical ML System

1. Run the `iris-feature-pipeline.ipynb` notebook with BACKFILL=True.
   Check in Hopsworks that this added 150 rows of features to the `iris` Feature Group.
   Revert BACKFILL=False and save this notebook.

2. Run the `iris-train-pipeline.ipynb` notebook.
   Check in Hopsworks that your model is in the Hopsworks Model Registry.

3. Run the `iris-batch-inference-pipeline.ipynb` notebook
   Check that an `iris-predictions` Feature Group was created with 1 row.
   Run the `iris-batch-inference-pipeline.ipynb` as many times as needed to insert 3 different iris flowers in your `iris_predictions` feature group. Then the notebook will start saving the *confusion_matrix.png* file.

4. Add the github actions secret *HOPSWORKS_API_KEY* with *your API key*

5. Enable/run the github actions workflow, *features-and-predictions.yml*, from the Github UI

6. Change to your Github Pages repo, *gh-pages*, edit the *index.md* and *_config.xml* files, and commit/push those changes back to your *gh-pages* repo.

Homework

1. https://forms.gle/2p5odBdpAqvavH1T7 **(deadline is 1 week from the lecture delivery)**
2. **Please enter these values** for the iris features in the Gradio UI (sepal_length=6.5, sepal_width=3, petal_length=5, petal_width=2) and identify the flower predicted by the original model.

3. In the iris training pipeline notebook, **change n_neighbors=5** in the KNN model, and then report the "weighted avg for the F1 score" for this model.

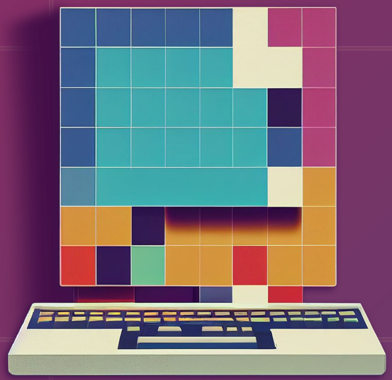
4. Give us your **Github URL** for your Github Pages UI, add the github action, and add a badge to your README.md to show that your action is "passing".

Optional

5. Design a Streamlit UI for the Iris Flower Dataset. Enter the URL for your **streamlit** cloud service.
6. Improve our github action (less code, cleaner) and create a PR with your proposed implementation. Enter the URL for your PR.
7. Rewrite *iris-batch-inference-pipeline.ipynb* to also store the features in the *iris_predictions* feature group. Add some new analyses, like feature-importance, to your Github Pages UI.

Homework Submission Here

https://forms.gle/2p5odBdpAqvavH1T7



SERVERLESS ML

www.serverless-ml.org

https://github.com/featurestoreorg/serverless-ml-course ⭐

Show love with a star!