Lab 1
ID2223 / HT2023

Iris Flowers as a Serverless ML System

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

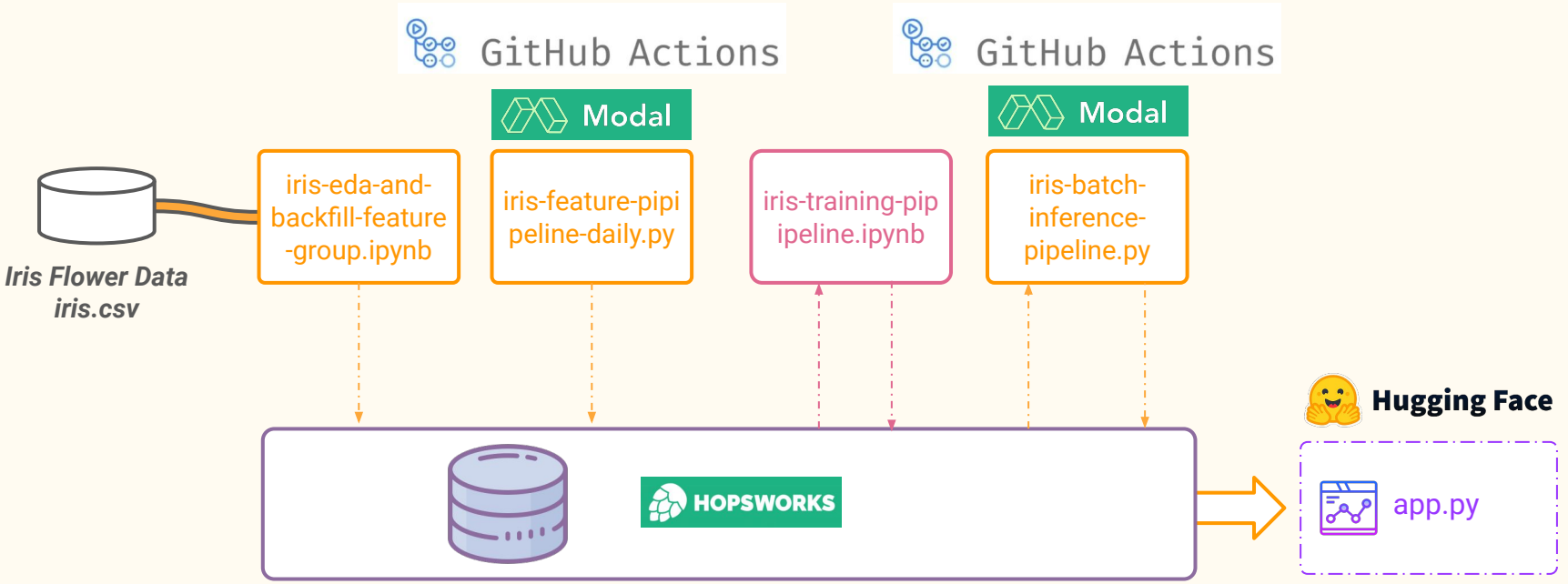
Course Material: Prof Jim Dowling

Source Code for Lab 1

- Source Code Github https://github.com/ID2223KTH/id2223kth.github.io/tree/master/src/serverless-ml-intro

- Use Conda or virtual environments to manage your python dependencies on your laptop. See more info on how to manage your Python environment here.

Iris as a serverless ML system



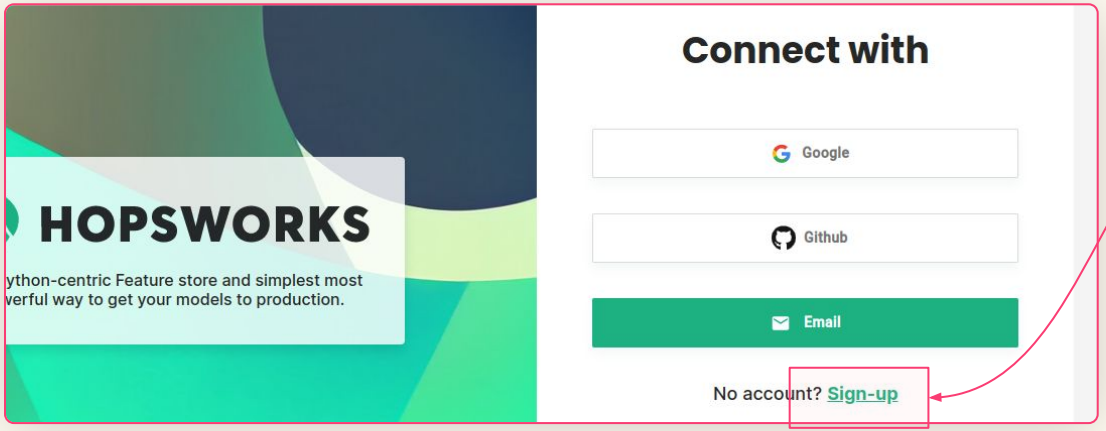
What will we cover in this part

- Case Study: Iris Flower Dataset

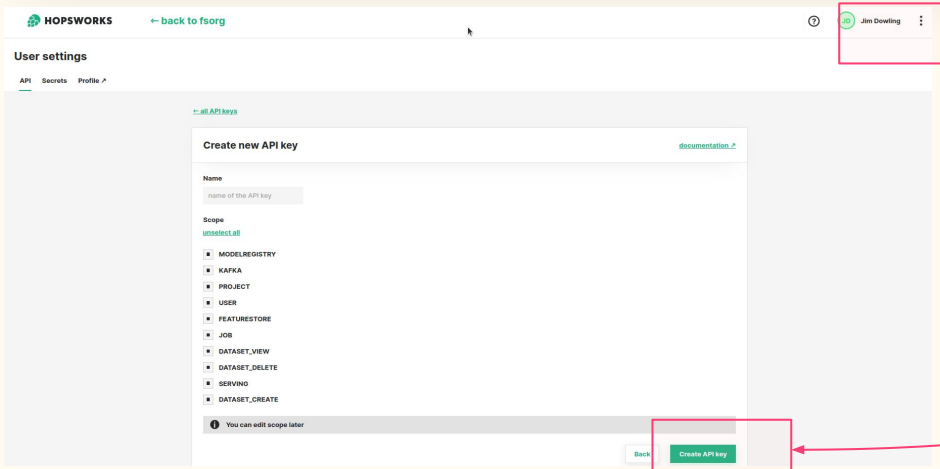
- **First Steps**
  a. Create a free account on hopsworks.ai
  b. Create a free account on modal.com or github.com
  c. Create a free account on huggingface.com
     (alternatively you can use streamlit.com)

- **Tasks**
  a. Build and run a feature pipeline on Modal or GithubActions
  b. Run a training pipeline
  c. Build and run an inference pipeline with a Gradio UI on Hugging Face Spaces.

Register and Login to the Hopsworks Feature Store



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



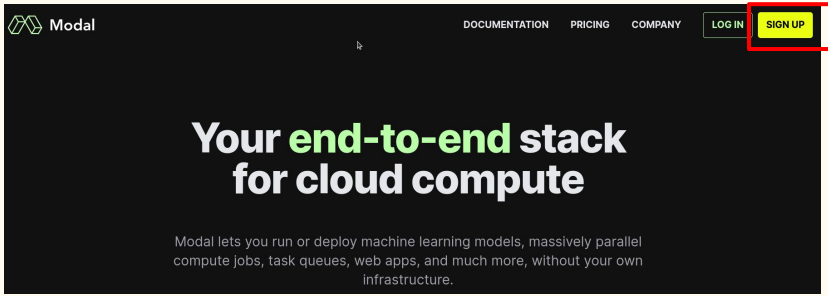
2. Click on “User Settings”

3. Create and Save an “API Key”

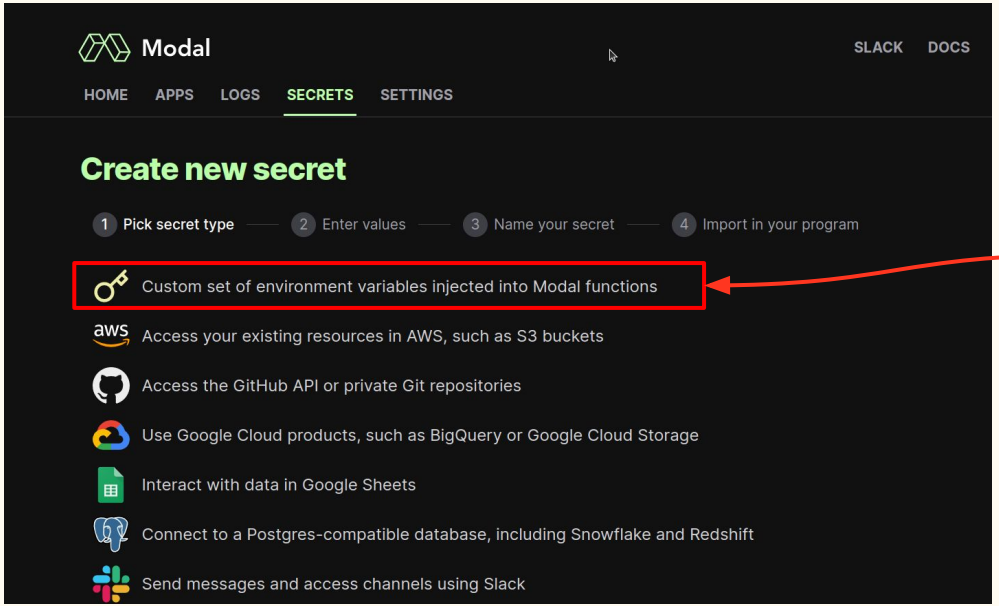
Choose how to run your serverless ML pipeline

Use either
(1) Modal - needs a credit card to register
(2) Github Actions - no credit card needed

Register to Modal and Set up HOPSWORKS_API_KEY environment variable

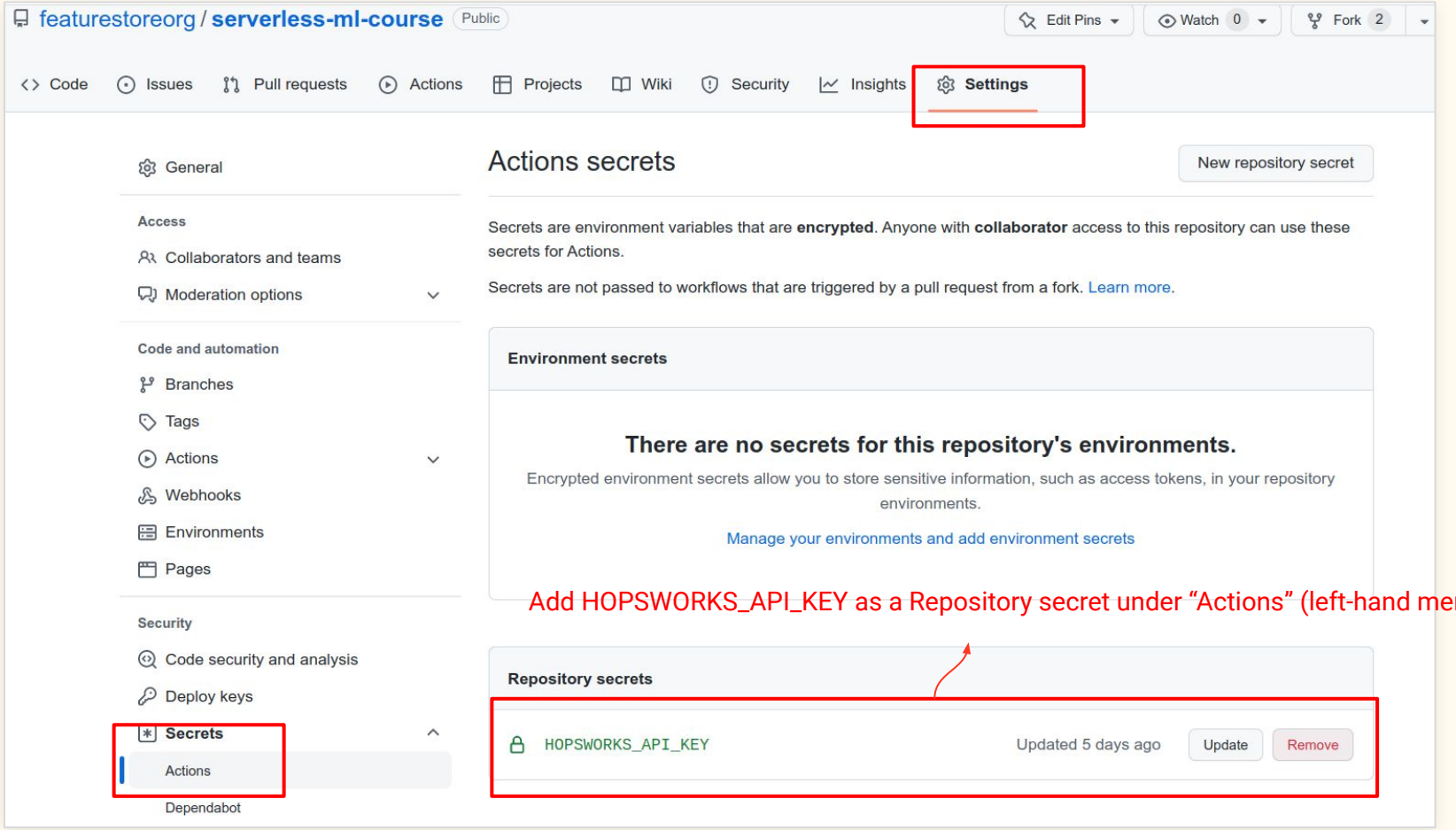


Create an account on Modal
(might need some time to be approved)

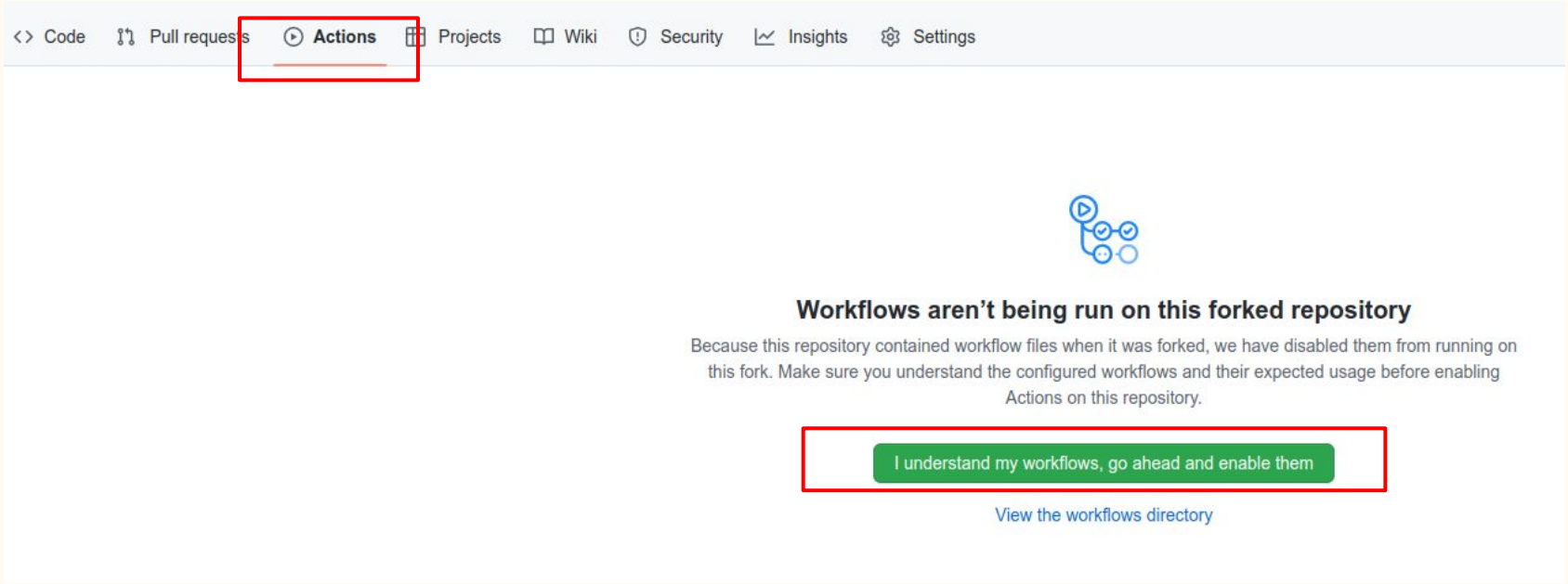


Add HOPSWORKS_API_KEY as a Environment variable secret

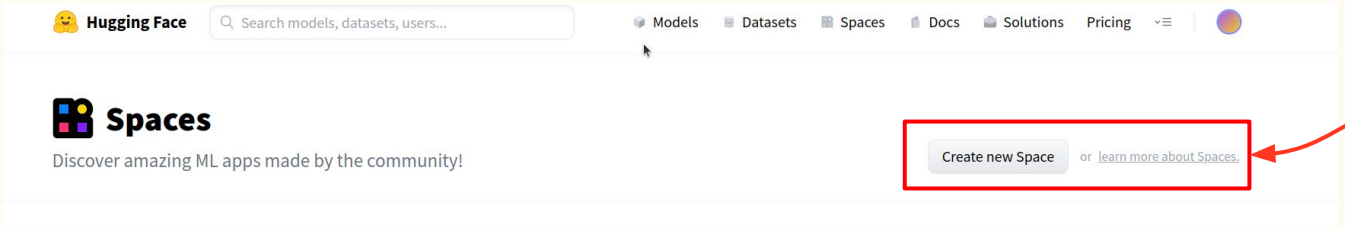
Add a HOPSWORKS_API_KEY as a secret for your Github Action



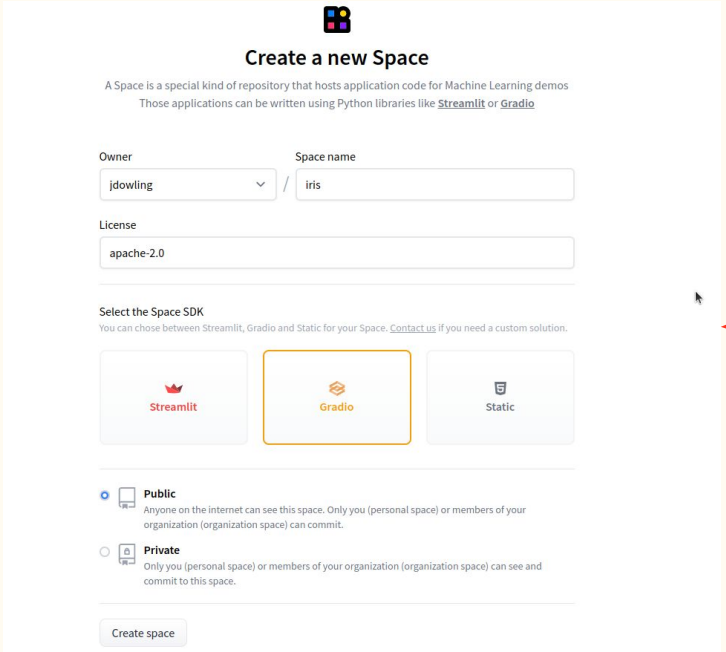
Enable the Github Actions for your Repository



Register and Create a Hugging Face Space

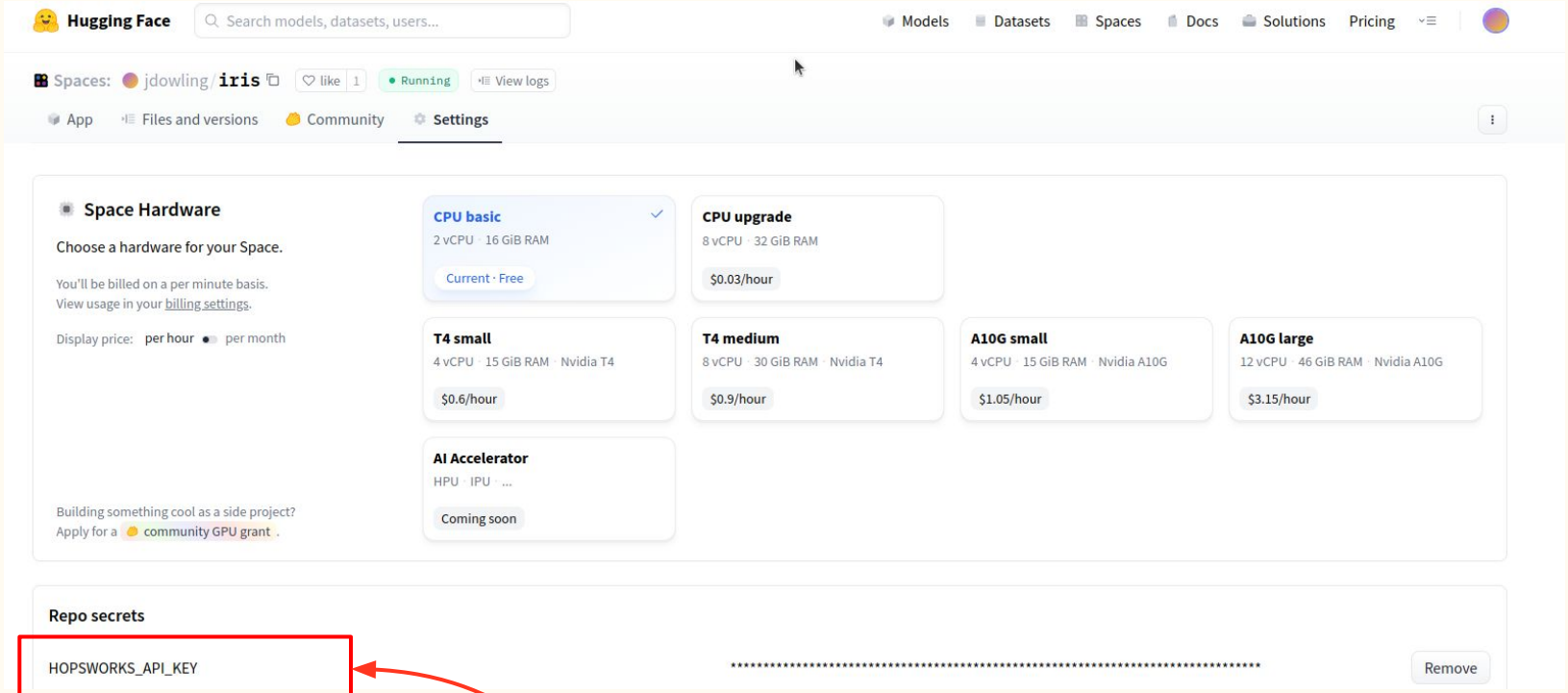


1. Create an account on Hugging Face
2. Create a “Space”

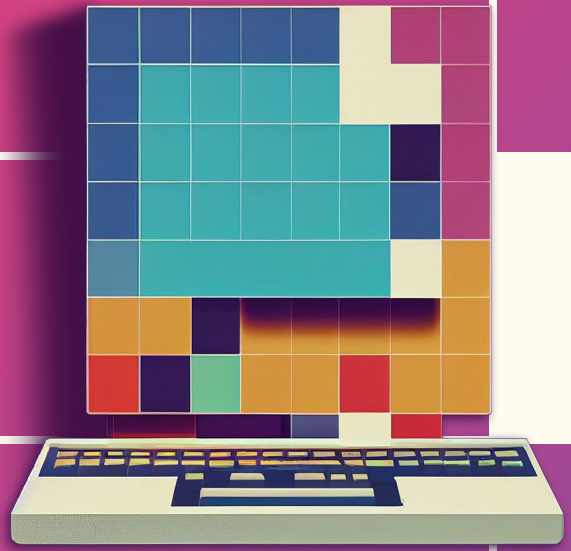


3. Create a Gradio App with the name Iris inside your account

Add a HOPSWORKS_API_KEY as a secret in your “iris” Space



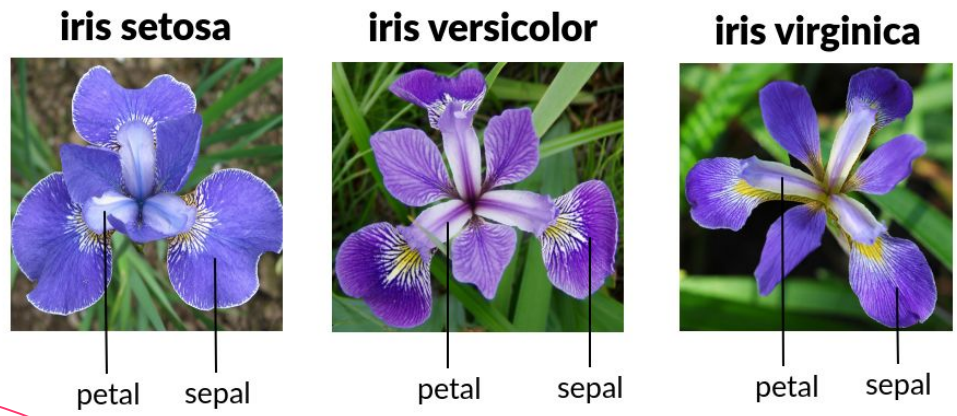
1. Add your HOPSWORKS_API_KEY as a Repo Secret



Serverless ML with
Iris Flower Dataset

Iris Flower Dataset

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



This column is the Pandas Index

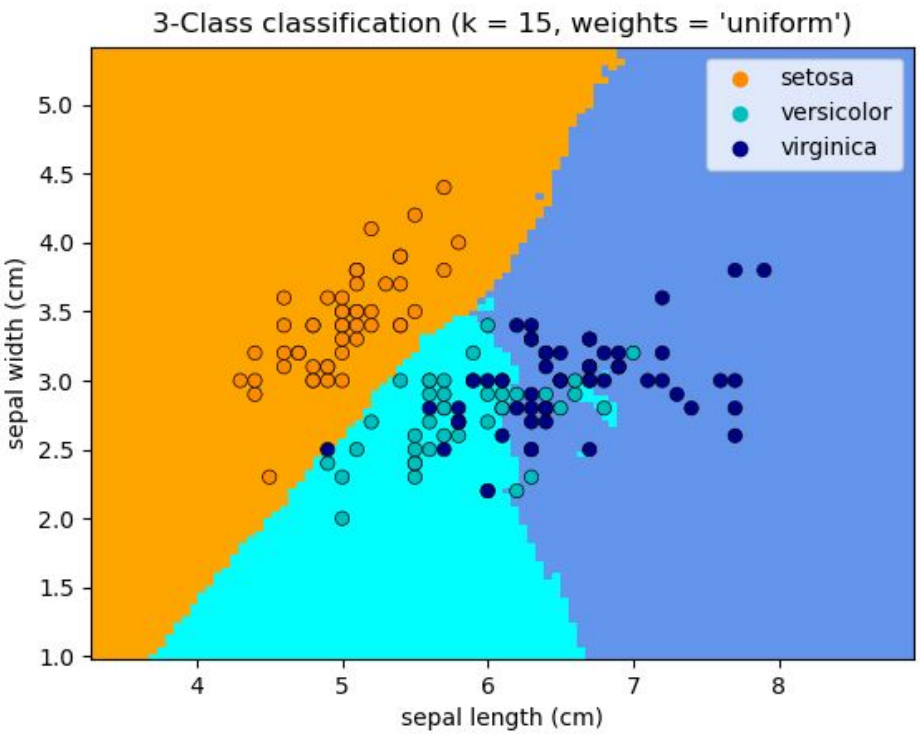
	sepal_length	sepal_width	petal_length	petal_width	variety
133	6.3	2.8	5.1	1.5	Virginica
48	5.3	3.7	1.5	0.2	Setosa
26	5.0	3.4	1.6	0.4	Setosa
134	6.1	2.6	5.6	1.4	Virginica
115	6.4	3.2	5.3	2.3	Virginica
15	5.7	4.4	1.5	0.4	Setosa
52	6.9	3.1	4.9	1.5	Versicolor

Tabular Data
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*).

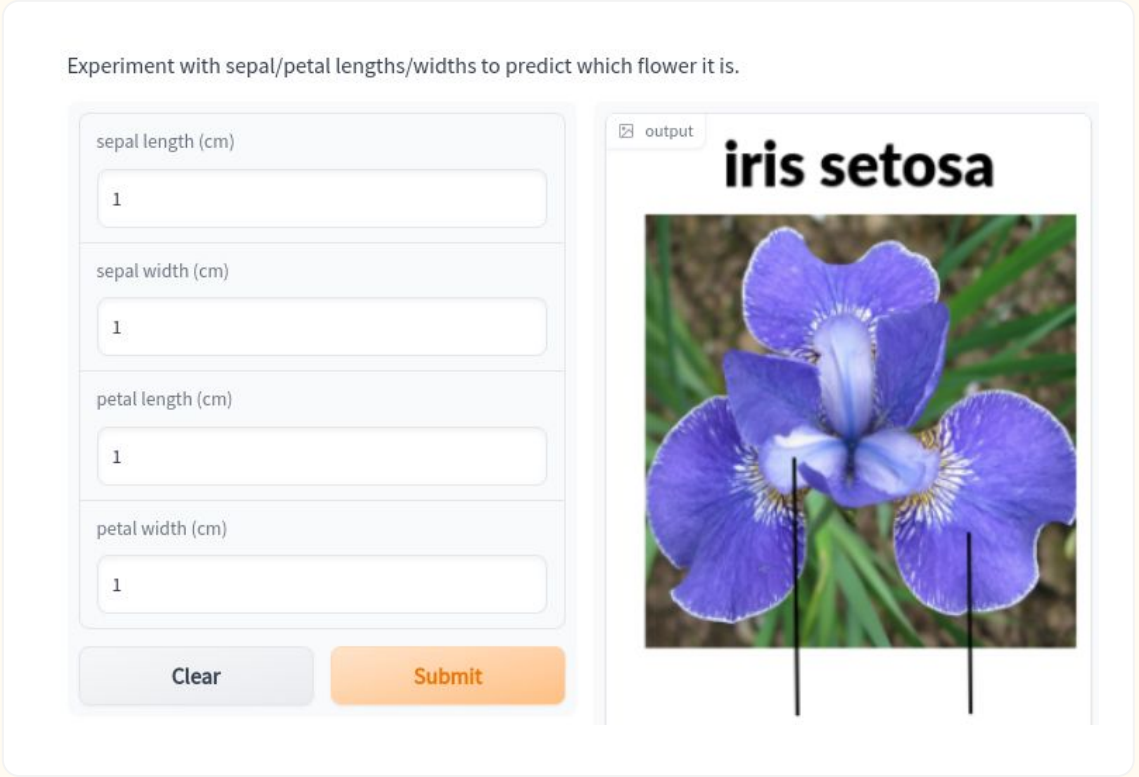


[Image from https://scikit-learn.org/stable/modules/neighbors.html#classification]

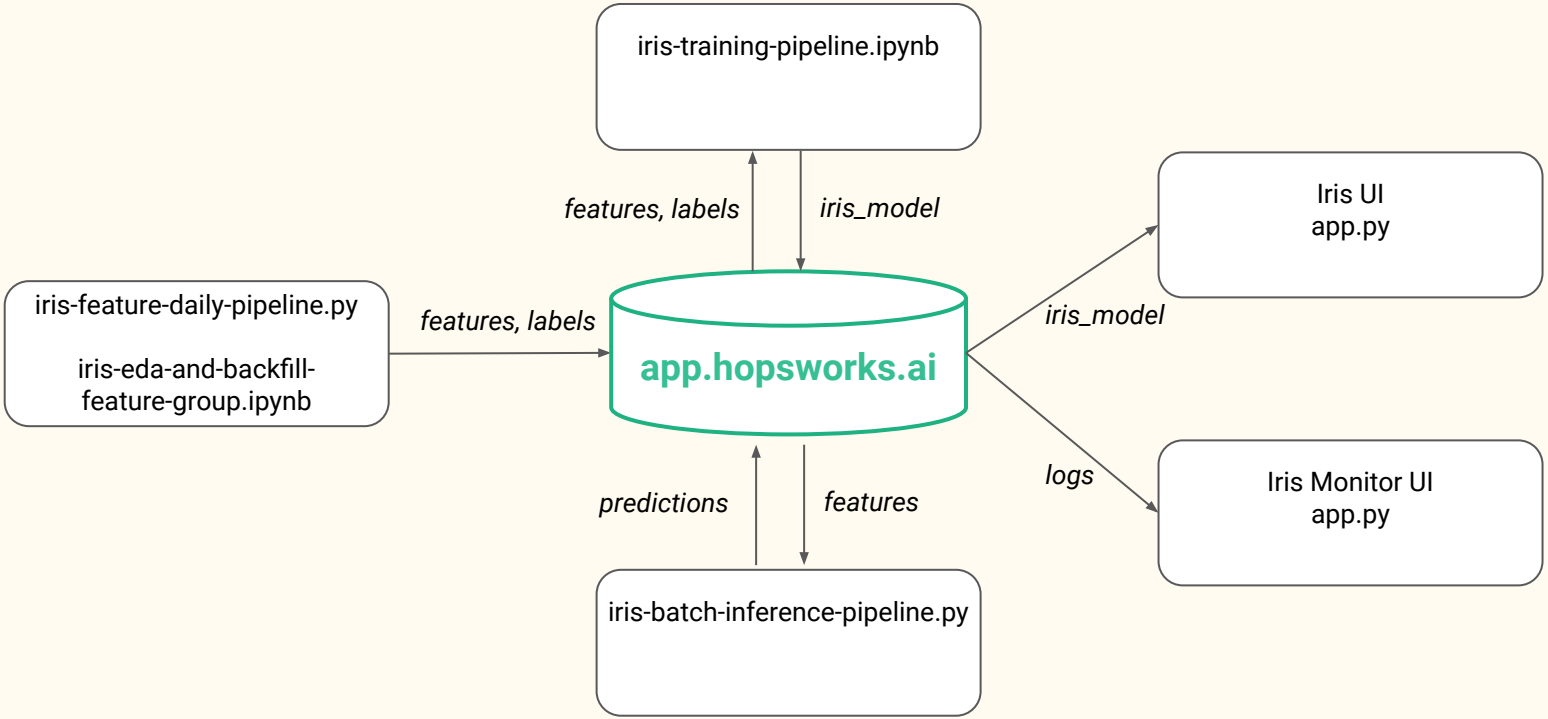
Communicate the value of your model with a UI (Gradio)

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

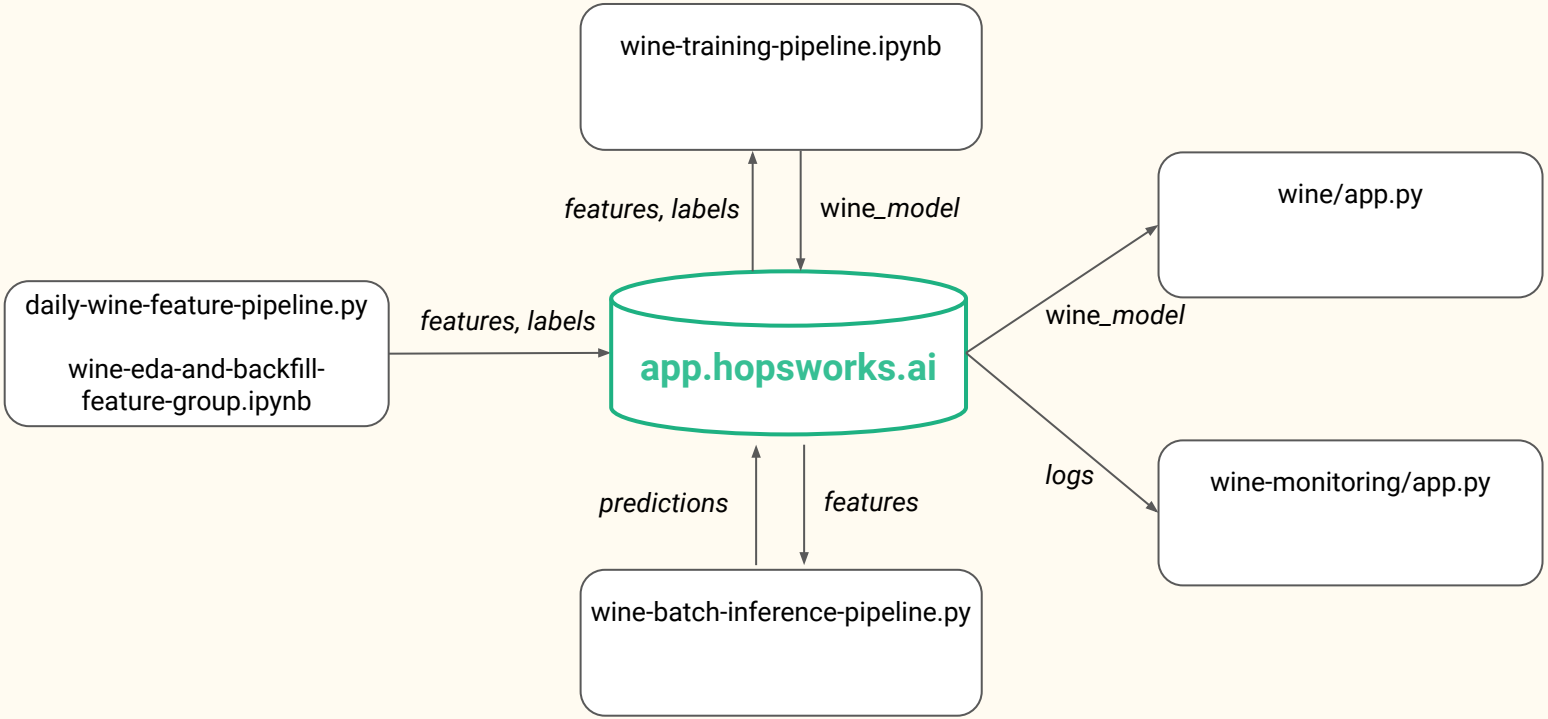
- Here, we design a UI in Python with Gradio
  - Enables “predictive analytics” where a user can use the model to as “what-if” i had an Iris Flower with this sepal/petal width/length?



Task 1: Run the Feature, Training, Online/Batch Inference Pipelines



Task 2: Build a Serverless ML system for the Wine Quality Dataset



Wine Quality Dataset - Needs some Feature Engineering

The wine quality dataset has a mix of numerical and categorical variables. You will need to do some data cleaning and feature engineering, including possibly some of these steps:

- Fill missing data with either random data or a category corresponding to "Unknown"
- Transform categorical variables into numerical variables
- Drop columns that do not have predictive power
- Write the features to the feature store as a Feature Group
- Read the features split the data into training and testing sets

```
df.describe().T
```

Output:

	count	mean	std	min	25%	50%	75%	max
fixed acidity	6487.0	7.216579	1.296750	3.80000	6.40000	7.00000	7.70000	15.90000
volatile acidity	6489.0	0.339691	0.164649	0.08000	0.23000	0.29000	0.40000	1.58000
citric acid	6494.0	0.318722	0.145265	0.00000	0.25000	0.31000	0.39000	1.66000
residual sugar	6495.0	5.444326	4.758125	0.60000	1.80000	3.00000	8.10000	65.80000
chlorides	6495.0	0.056042	0.035036	0.00900	0.03800	0.04700	0.06500	0.61100
free sulfur dioxide	6497.0	30.525319	17.749400	1.00000	17.00000	29.00000	41.00000	289.00000
total sulfur dioxide	6497.0	115.744574	56.521855	6.00000	77.00000	118.00000	156.00000	440.00000
density	6497.0	0.994697	0.002999	0.98711	0.99234	0.99489	0.99699	1.03898
pH	6488.0	3.218395	0.160748	2.72000	3.11000	3.21000	3.32000	4.01000
sulphates	6493.0	0.531215	0.148814	0.22000	0.43000	0.51000	0.60000	2.00000
alcohol	6497.0	10.491801	1.192712	8.00000	9.50000	10.30000	11.30000	14.90000
quality	6497.0	5.818378	0.873255	3.00000	5.00000	6.00000	6.00000	9.00000

Some descriptive statistical measures of the dataset

Task 2: Wine Quality Prediction Tasks

1. The Wine Quality Dataset:
   a. https://raw.githubusercontent.com/ID2223KTH/id2223kth.github.io/master/assignments/lab1/wine.csv
2. Write a feature pipeline notebook that registers the wine quality dataset as a Feature Group with Hopsworks.
3. Write a training pipeline that reads training data with a Feature View from Hopsworks, trains a **regression or classifier model** to predict if a wine's quality. Register the model with Hopsworks.
4. Write a Gradio or Streamlit application that downloads your model from Hopsworks and provides a User Interface to allow users to enter or select feature values to predict the quality of a wine for the features you entered.
5. Write a synthetic wine generator function and write a new "daily" feature pipeline that runs once per day to add a new synthetic wine.
6. Write a batch inference pipeline to predict the quality of the new wine(s) added, and build a Gradio or Streamlit application to show the most recent wine quality prediction and outcome, and a confusion matrix with historical prediction performance.

References: https://www.kaggle.com/datasets/rajyellow46/wine-quality
https://www.ritchieng.com/pandas-scikit-learn/

Deliverables

- Deliver your source code as a Github Repository for Task 2.
- Deliver your lab description as a README.md file in the root of your Github repository
- Deliver a public URL for the 2 Gradio or Streamlit Applications:
  (1) Interactive UI for entering feature values and predicting the wine quality
  (2) Dashboard UI showing the most recent wine added to the Feature Store and the predicted quality (label) for that wine. Include a confusion matrix to show historical model performance.

Deadline midnight 20th November.

The lab will be graded during a defence of your lab held over Zoom in the week of November 20th. Available Zoom slots for defence will be published in Canvas.

Grading

- Maximum points for this lab will be awarded if you (1) complete all the tasks - including a realistic wine simulation function and a reasonably performing wine model, (2) answer our questions during the grading defence.
- A passing grade will require that you complete task 1 and make a good attempt at task 2.