Lab 1
ID2223 / HT2022

Iris Flowers as a Serverless ML System

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

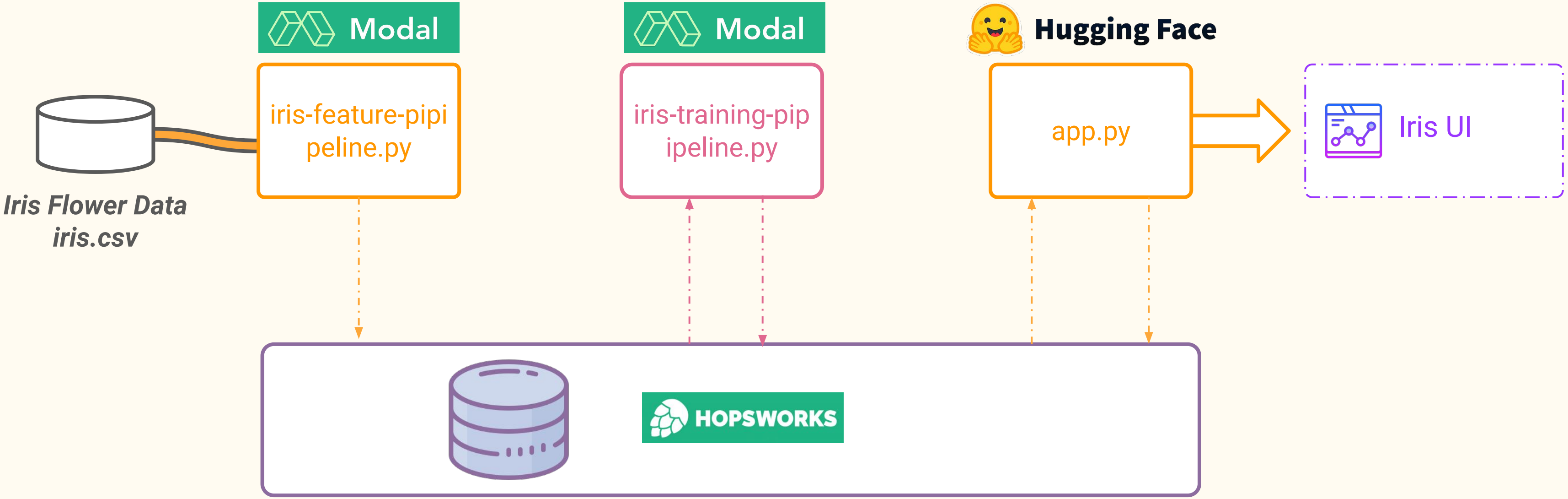
Course Material: Prof Jim Dowling

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

Iris as a serverless ML system with Modal, Hopsworks, and Hugging Face Spaces



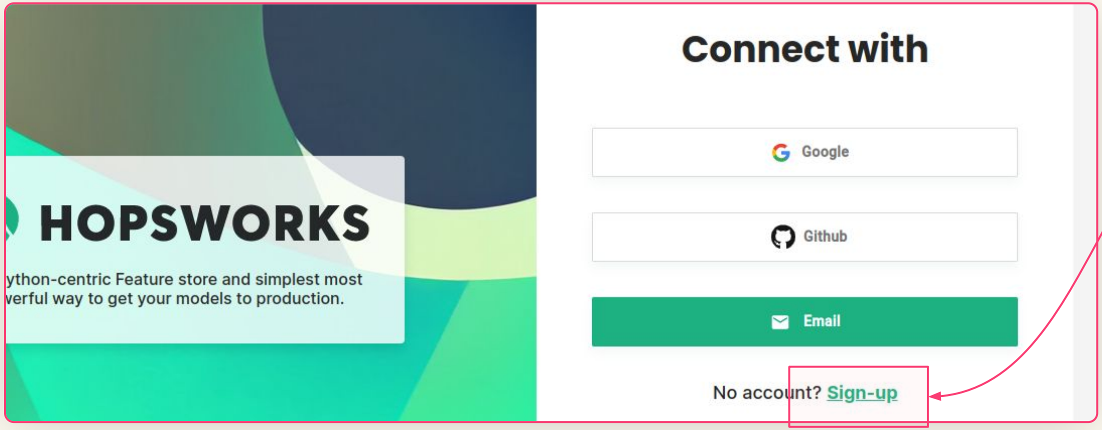
What will we cover in this part

- Case Study: Iris Flower Dataset

- **First Steps**
  a. Create a free account on hopsworks.ai
  b. Create a free account on modal.com
  c. Create a free account on huggingface.com

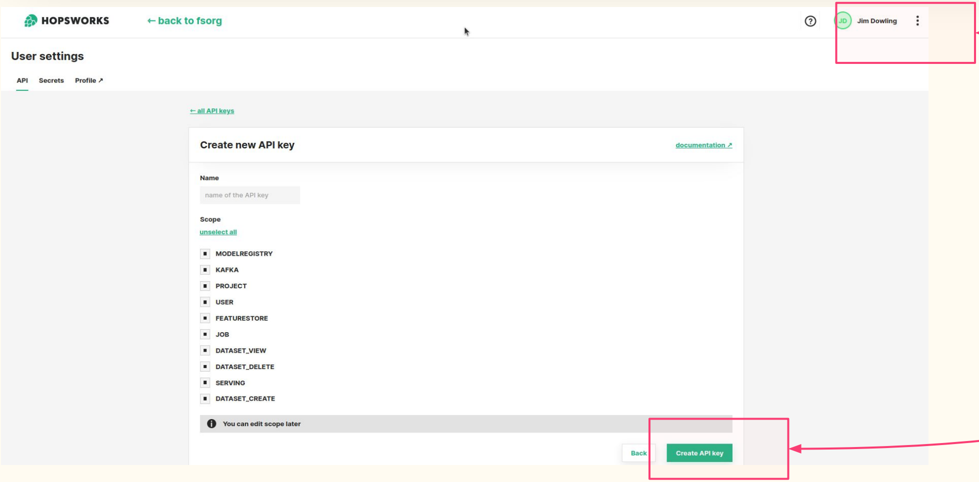
- **Tasks**
  a. Build and run a feature pipeline on Modal
  b. Build and run a training pipeline on Modal
  c. Build and run an inference pipeline with a Gradio UI on Hugging Face Spaces.

Register and Login to the Hopsworks Feature Store



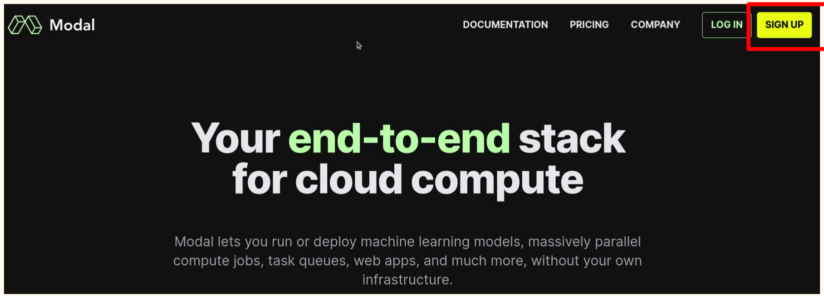
1. First, create an account on https://app.hopsworks.ai

2. Click on "User Settings"

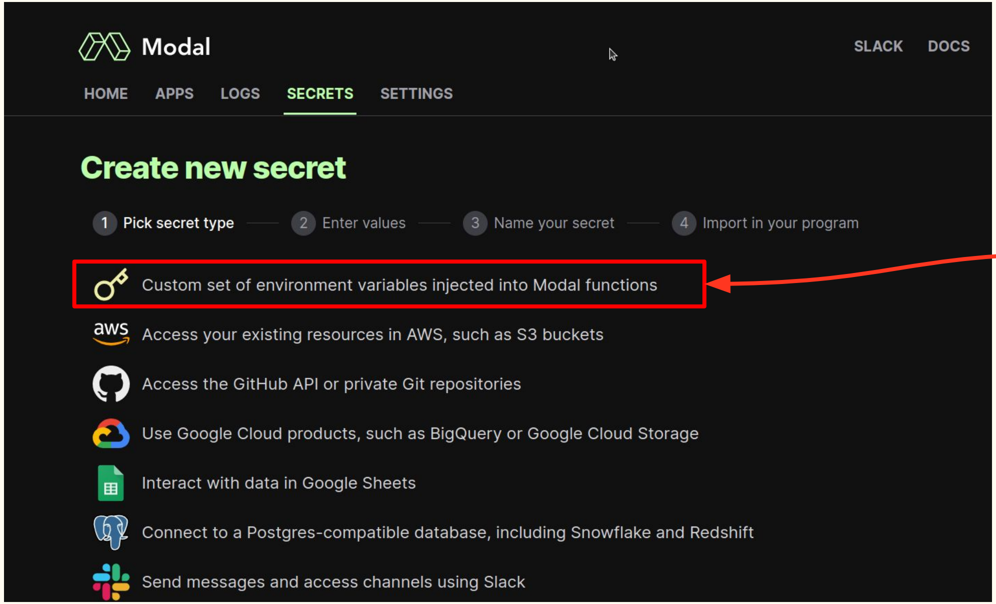


3. Create and Save an "API Key"

Register to Modal and Set up HOPSWORKS_API_KEY environment variable

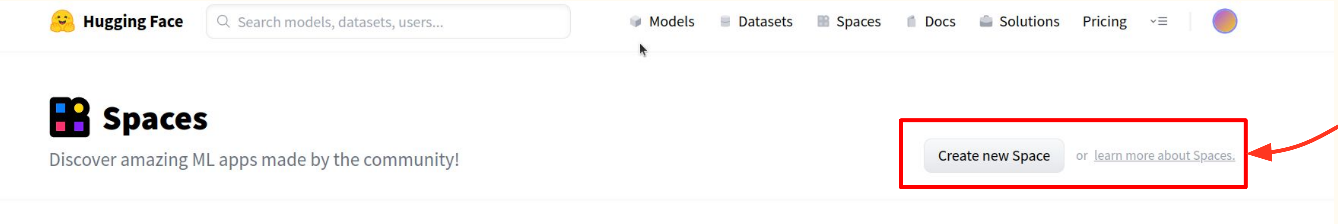


Create an account on Modal
(might need some time to be approved)

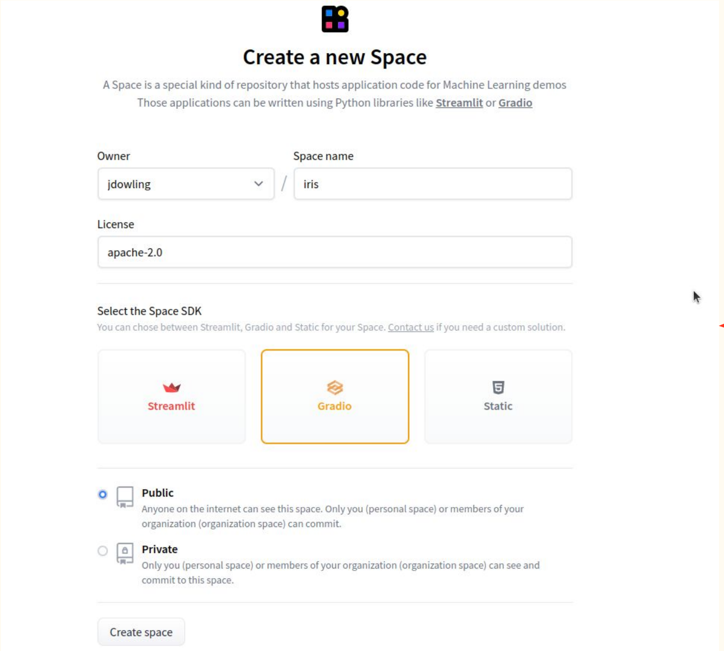


Add HOPSWORKS_API_KEY as a Environment variable secret

Register and Create a Hugging Face Space

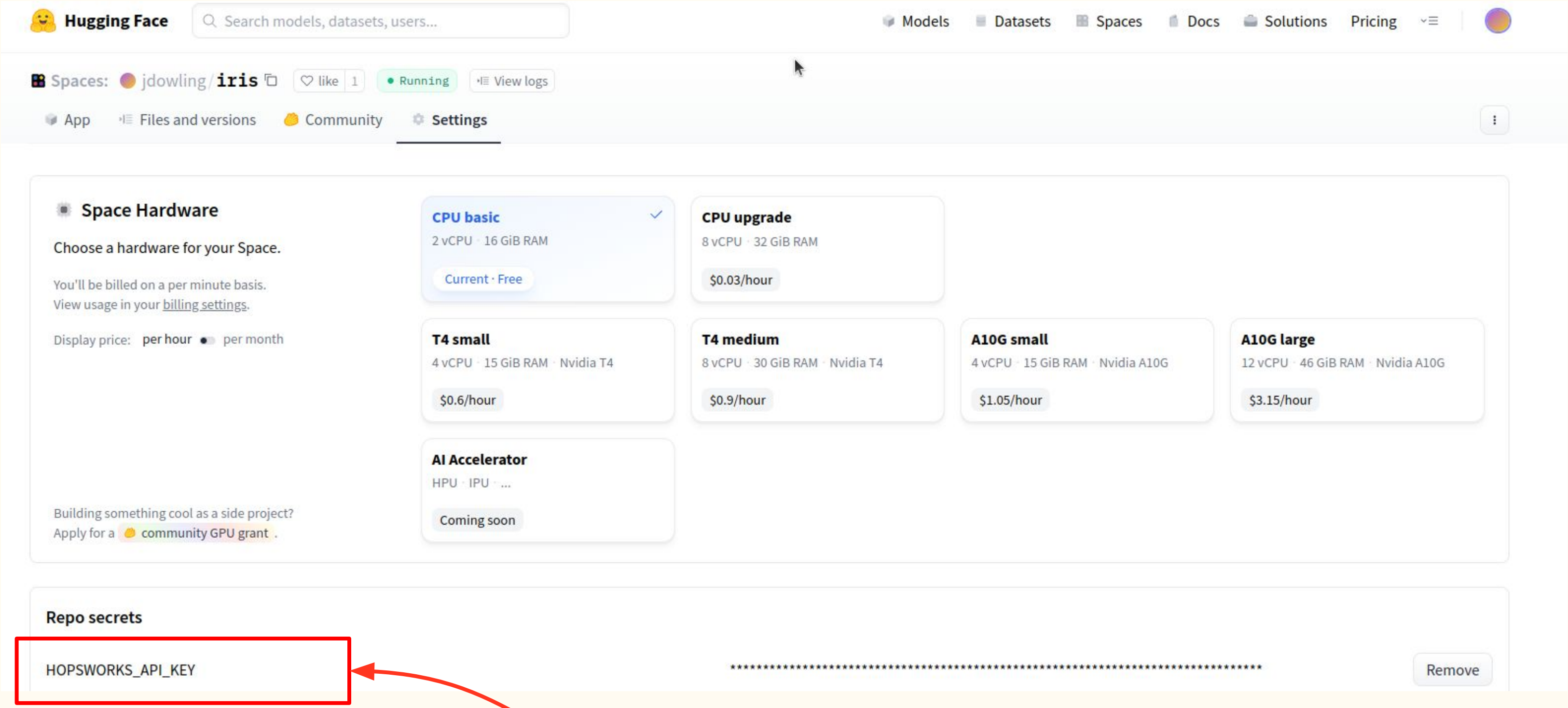


1. Create an account on Hugging Face
2. Create a "Space"

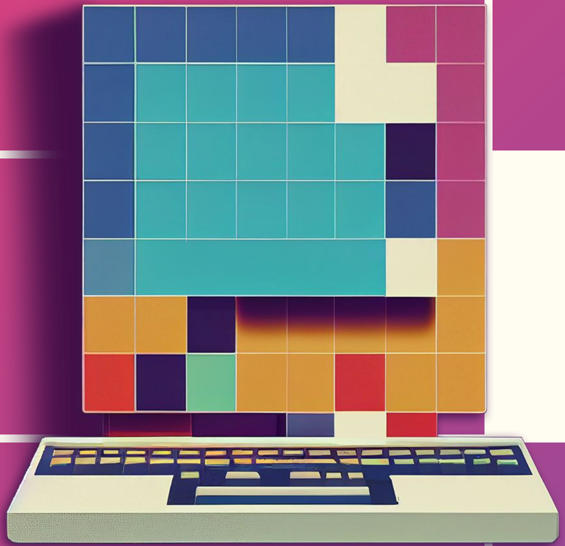


3. Create a Gradio App with the name Iris inside your account

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



1. Add your HOPSWORKS_API_KEY as a Repo Secret

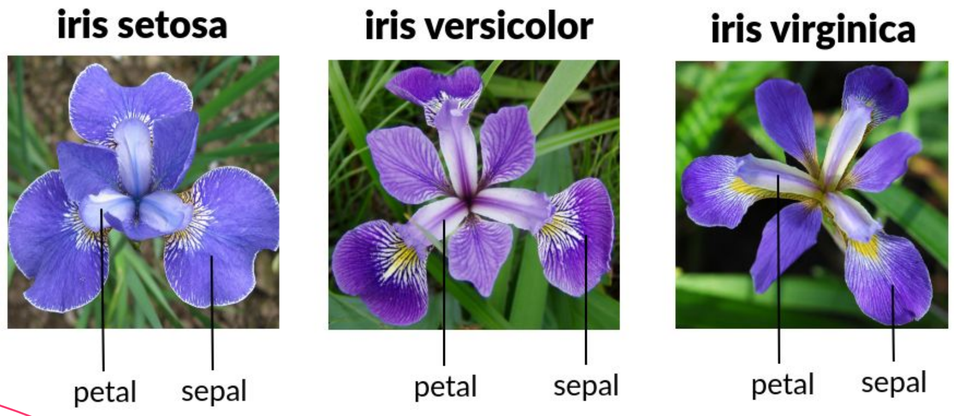


Serverless ML with
Iris Flower Dataset

Iris Flower Dataset

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

This column is the Pandas Index



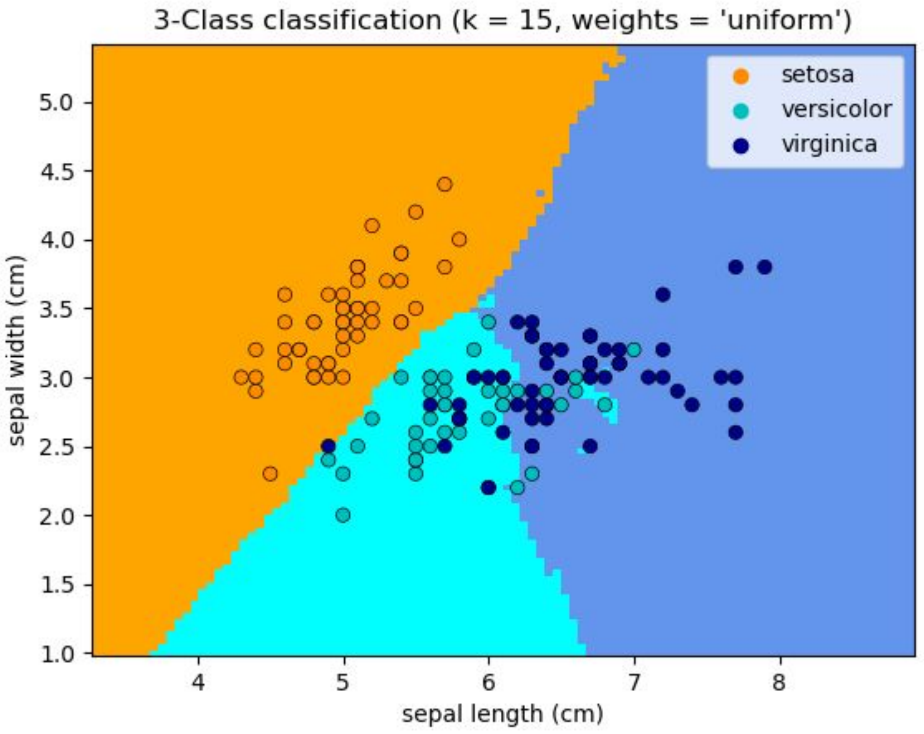
	sepal_length	sepal_width	petal_length	petal_width	variety
133	6.3	2.8	5.1	1.5	Virginica
48	5.3	3.7	1.5	0.2	Setosa
26	5.0	3.4	1.6	0.4	Setosa
134	6.1	2.6	5.6	1.4	Virginica
115	6.4	3.2	5.3	2.3	Virginica
15	5.7	4.4	1.5	0.4	Setosa
52	6.9	3.1	4.9	1.5	Versicolor

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*).

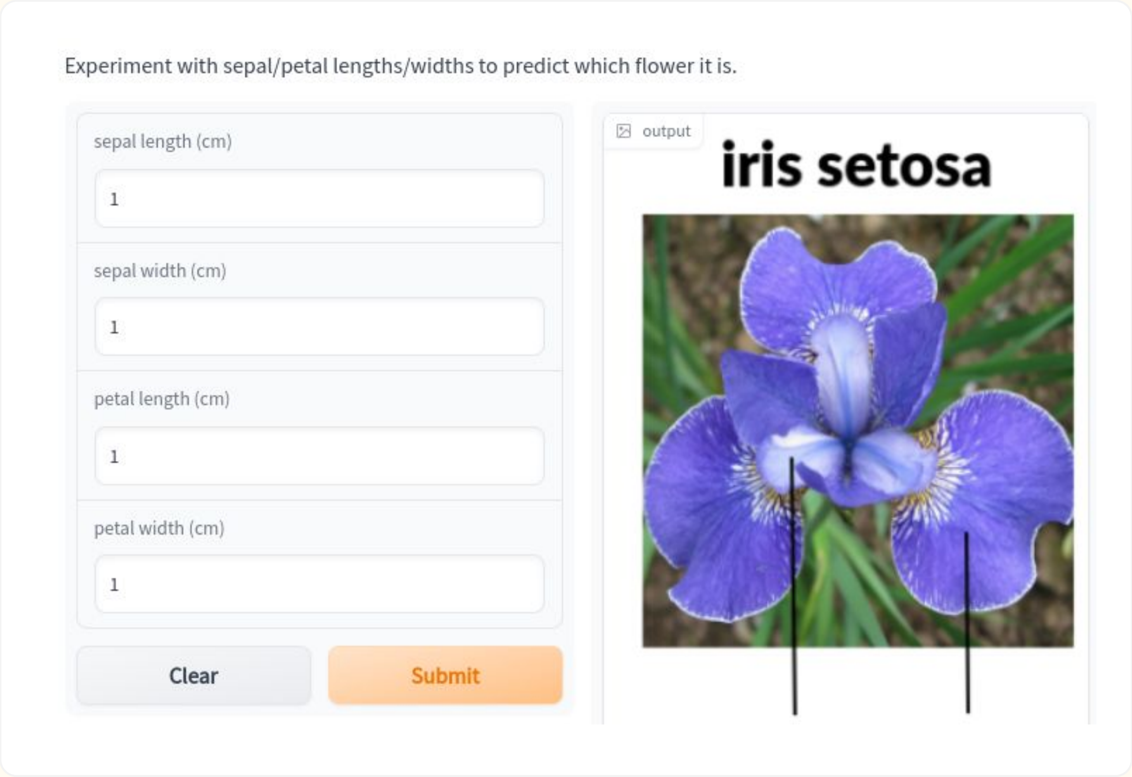


[Image from https://scikit-learn.org/stable/modules/neighbors.html#classification]

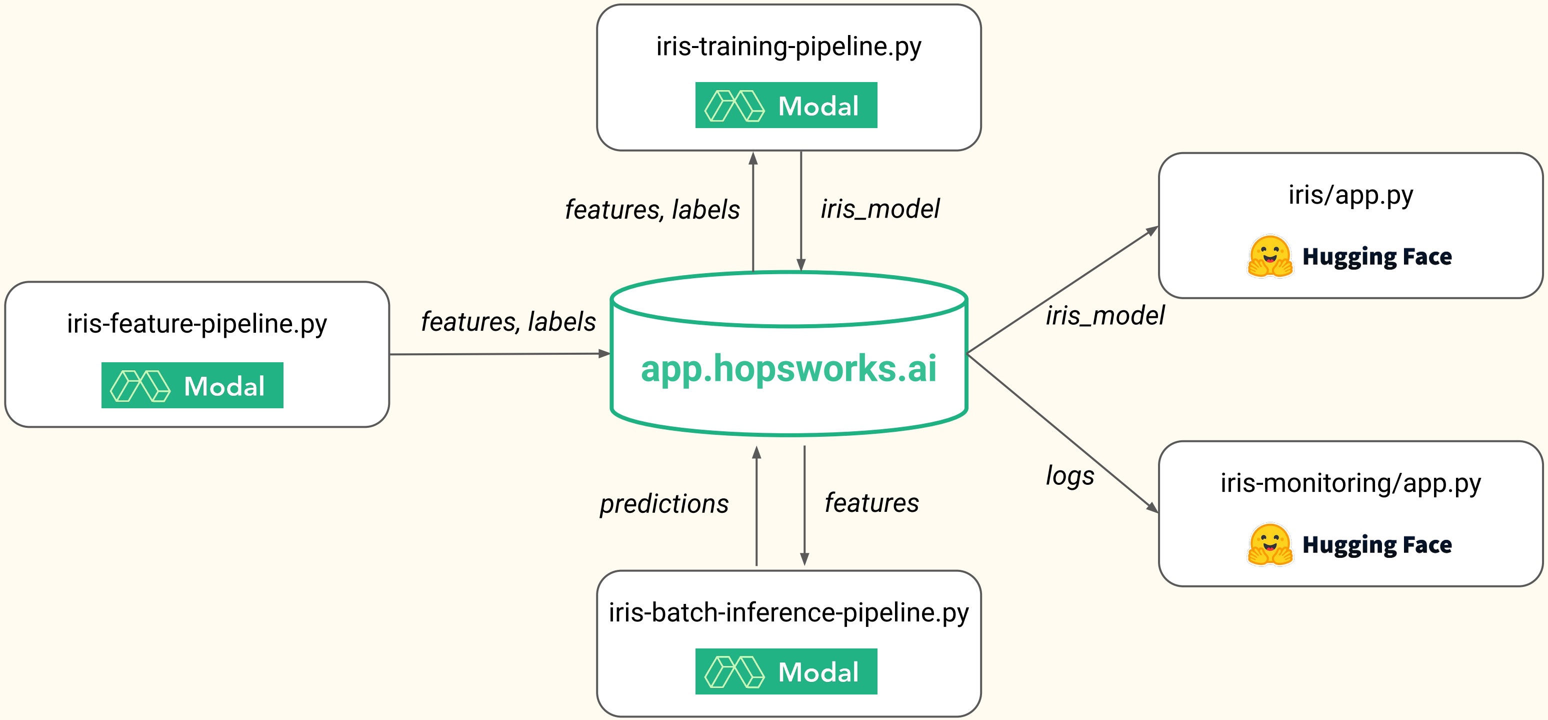
Communicate the value of your model with a UI (Gradio)

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

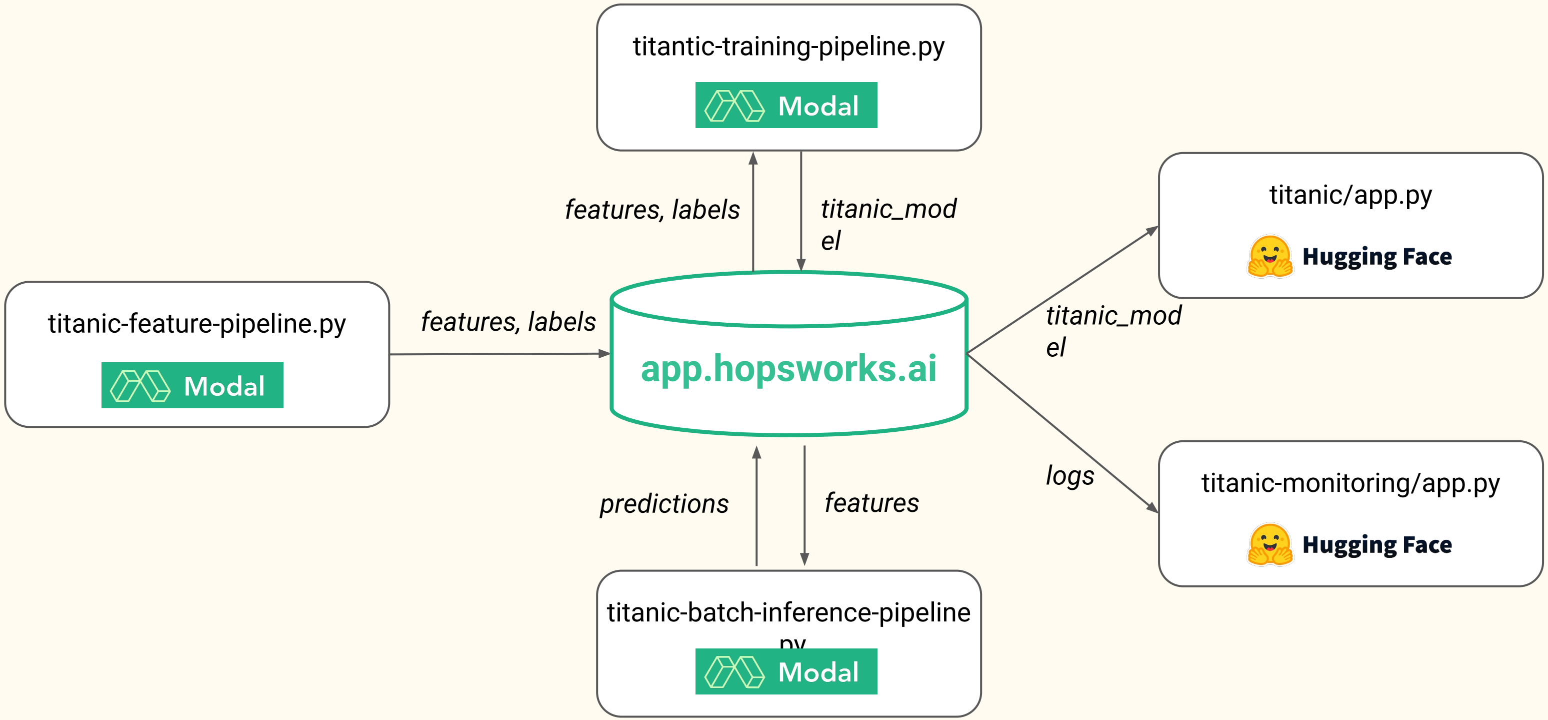
- Here, we design a UI in Python with Gradio
    - Enables “predictive analytics” where a user can use the model to as “what-if” i had an Iris Flower with this sepal/petal width/length?



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



Task 2: Build a Serverless ML system for the Titanic Dataset



Titanic Survival Dataset - Needs some Feature Engineering

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

The raw Titanic 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

Task 2: Serverless Titanic Survival Tasks

1. The Titanic Dataset:
    a. https://raw.githubusercontent.com/ID2223KTH/id2223kth.github.io/master/assignments/lab1/titanic.csv
2. Write a feature pipeline that registers the titantic dataset as a Feature Group with Hopsworks. You are free to drop or clean up features with missing values.
3. Write a training pipeline that reads training data with a Feature View from Hopsworks, trains a **binary classifier model** to predict if a particular passenger survived the Titanic or not. Register the model with Hopsworks.
4. Write a Gradio application that downloads your model from Hopsworks and provides a User Interface to allow users to enter or select feature values to predict if a passenger with the provided features would survive or not.
5. Write a synthetic data passenger generator and update your feature pipeline to allow it to add new synthetic passengers.
6. Write a batch inference pipeline to predict if the synthetic passengers survived or not, and build a Gradio application to show the most recent synthetic passenger prediction and outcome, and a confusion matrix with historical prediction performance.

References: https://www.kaggle.com/competitions/titanic/data
https://www.ritchieng.com/pandas-scikit-learn/

Deliverables

- Deliver your source code as a Github Repository
- Deliver your lab description as a README.md file in the root of your Github repository
- Deliver a Hugging Face Spaces public URL for the 2 Gradio Applications
  (1) Interactive UI for entering feature values and predicting if a passenger would survive the titanic or not
  (2) Dashboard UI showing a prediction of survival for the most recent passenger added to the Feature Store and the outcome (label) if that passenger survived or not. Include a confusion matrix to show historical model performance.

Deadline midnight 24th November.