

Lab 1

ID2223 / HT2024

.....



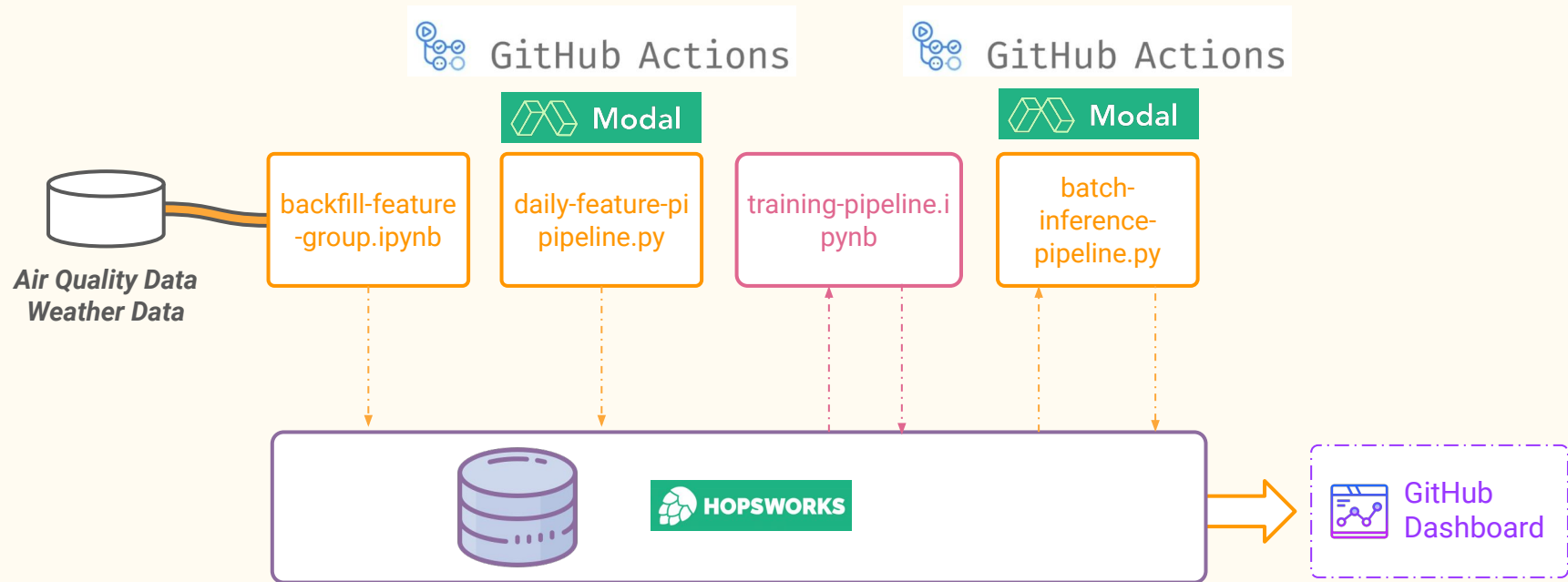
Air Quality Prediction Service

Course Material: Prof Jim Dowling

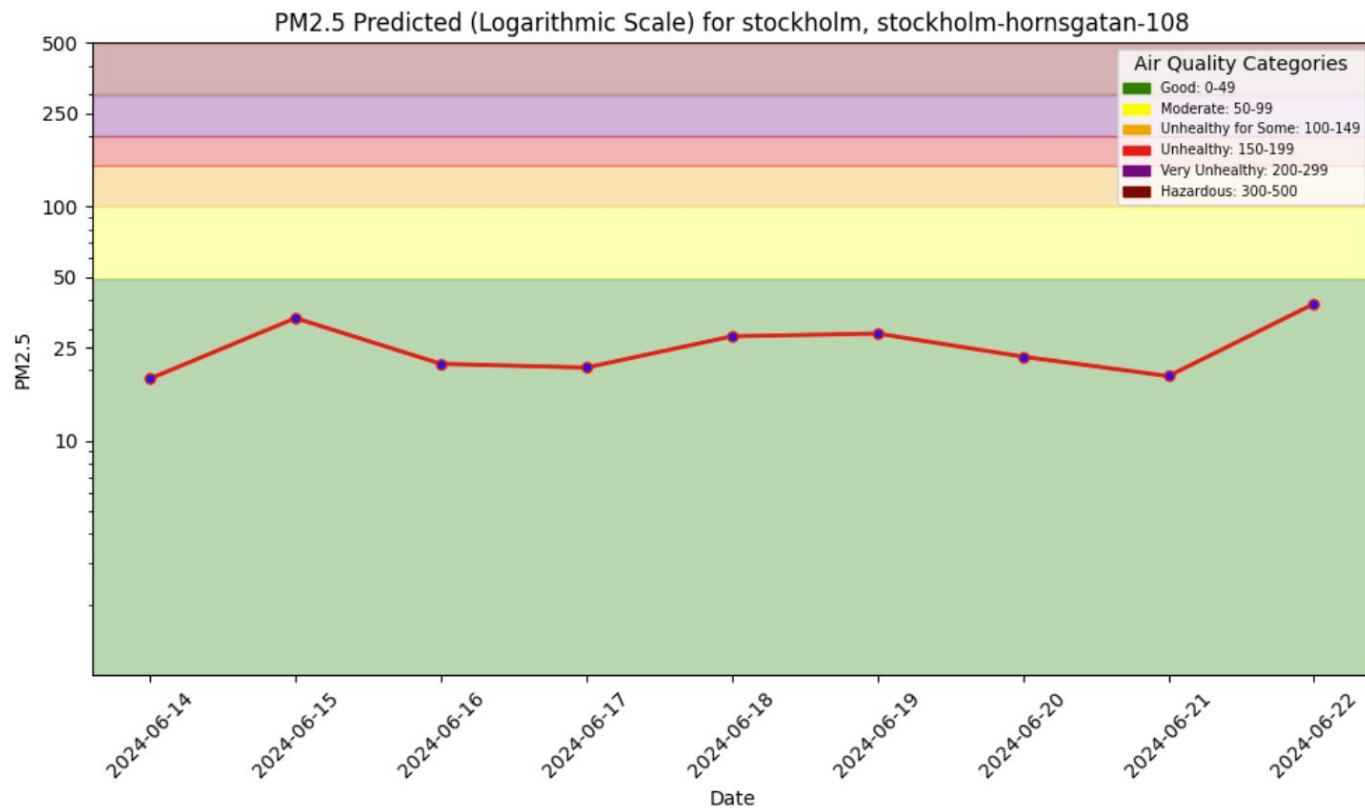
Source Code and References for Lab 1

- Source Code for this project Github
<https://github.com/featurestorebook/mlfs-book/>
- See Chapter 3 in the “[Building ML Systems with a Feature Store](#)” book on the course’s Github Page
- Use Conda or virtual environments to manage your python dependencies on your laptop. [See more info on how to manage your Python environment here.](#)

Serverless AI System that Predicts Air Quality for a Location



Dashboard for Serverless AI System that Predicts Air Quality for a Location



What will we cover in this part

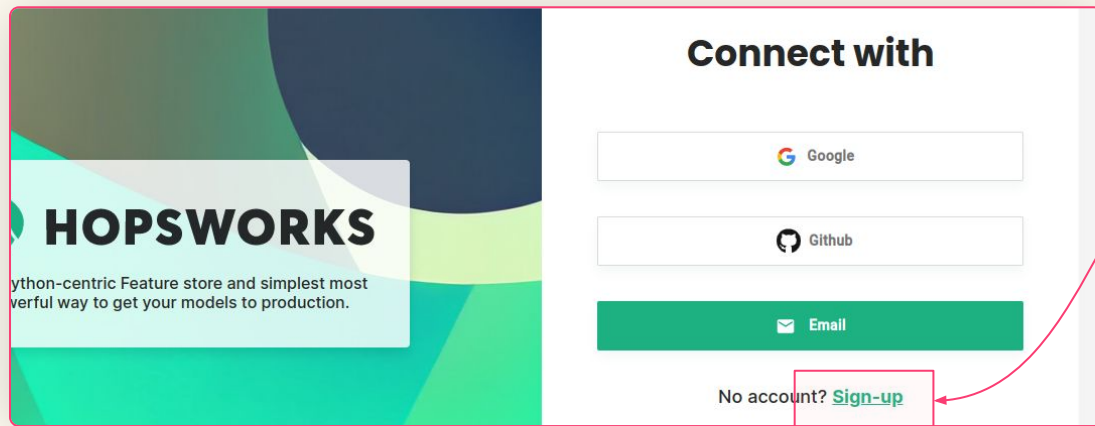
- **First Steps**

- a. Create a free account on hopsworks.ai
- b. Create a free account on github.com (and optionally modal.com)

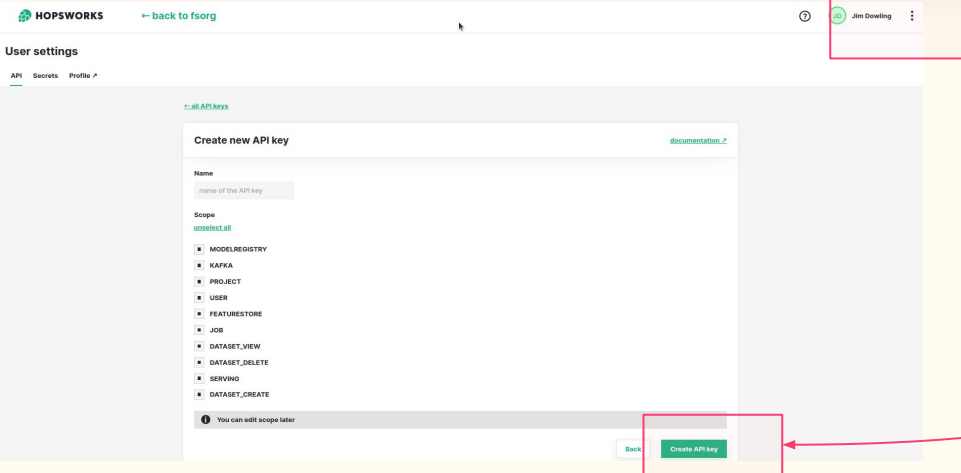
- **Tasks**

- a. Build and run a feature pipeline on Github Actions or Modal
- b. Run a training pipeline
- c. Build and run a batch inference pipeline on Github Actions or Modal
- d. Visualize your air quality predictions with a dashboard

Register and Login to the Hopsworks Feature Store



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



2. Click on "User Settings"

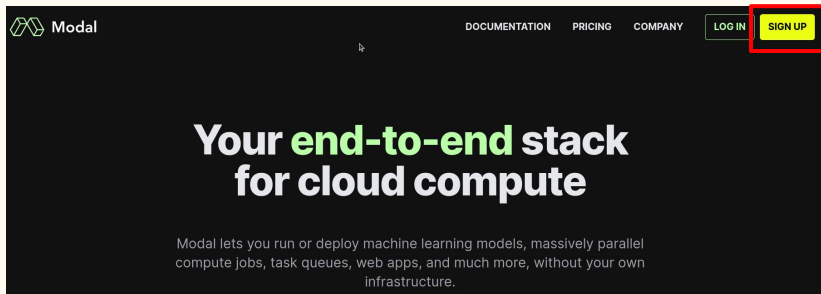
3. Create and Save an "API Key"

Choose how to run your serverless ML pipeline

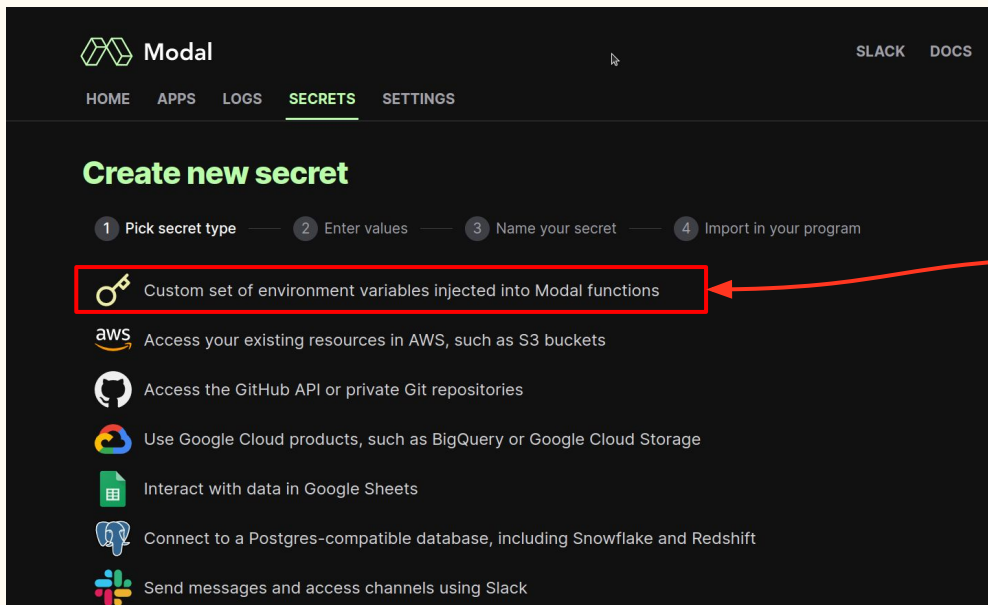
Use either

- (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

<> Code Issues Pull requests Actions Projects Wiki Security Insights **Settings**

General

Access

Collaborators and teams

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Pages

Security

Code security and analysis

Deploy keys

*** Secrets**

Actions

Dependabot

Actions secrets

New repository secret

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

Environment secrets

There are no secrets for this repository's environments.

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

[Manage your environments and add environment secrets](#)

Repository secrets

HOPSWORKS_API_KEY	Updated 5 days ago	Update	Remove
-------------------	--------------------	--------	--------

Add HOPSWORKS_API_KEY as a Repository secret under "Actions" (left-hand menu)

Enable the Github Actions for your Repository

<> Code  Pull requests  **Actions**  Projects  Wiki  Security  Insights  Settings

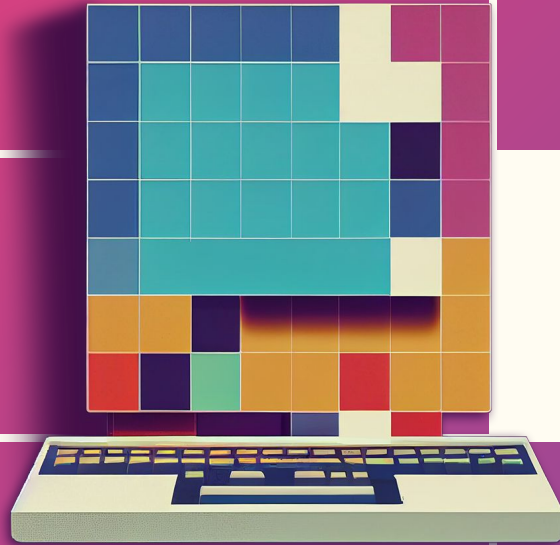


Workflows aren't being run on this forked repository

Because this repository contained workflow files when it was forked, we have disabled them from running on this fork. Make sure you understand the configured workflows and their expected usage before enabling Actions on this repository.

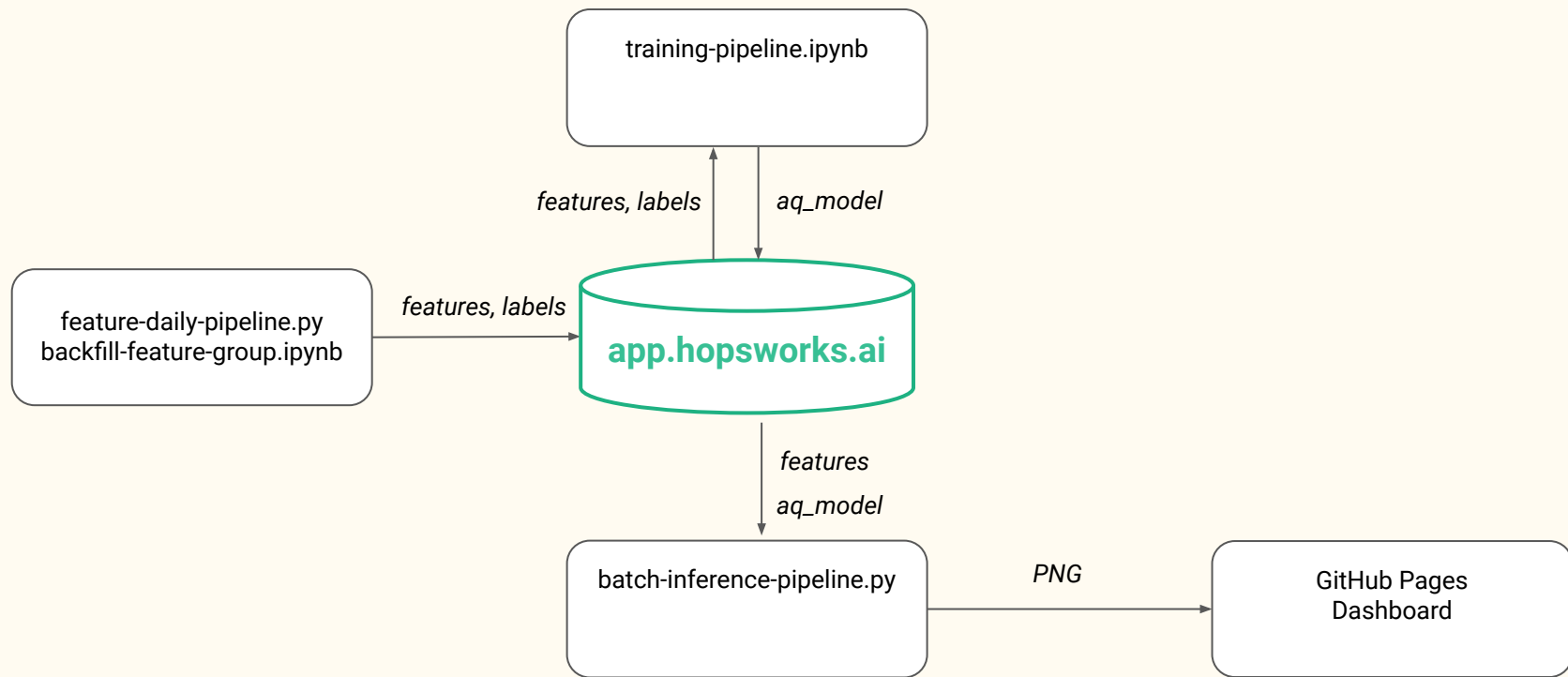
I understand my workflows, go ahead and enable them

[View the workflows directory](#)



Weather and Air Quality Data Sources

Air Quality Prediction: Feature, Training, Batch Inference Pipelines



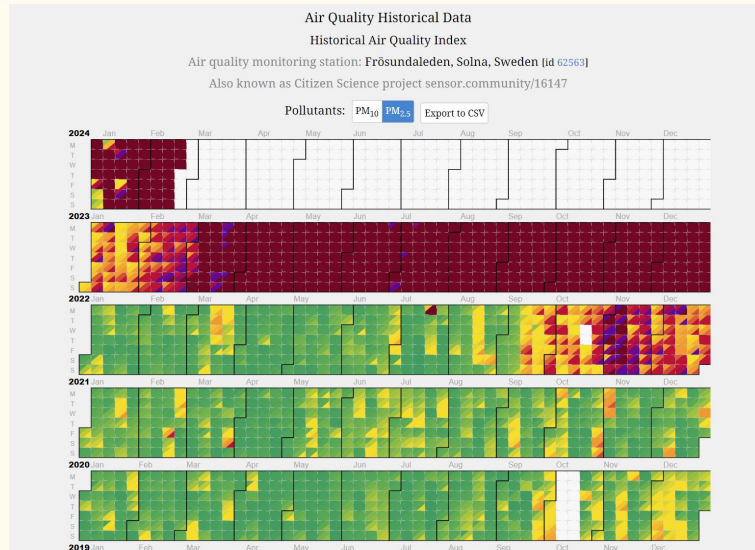
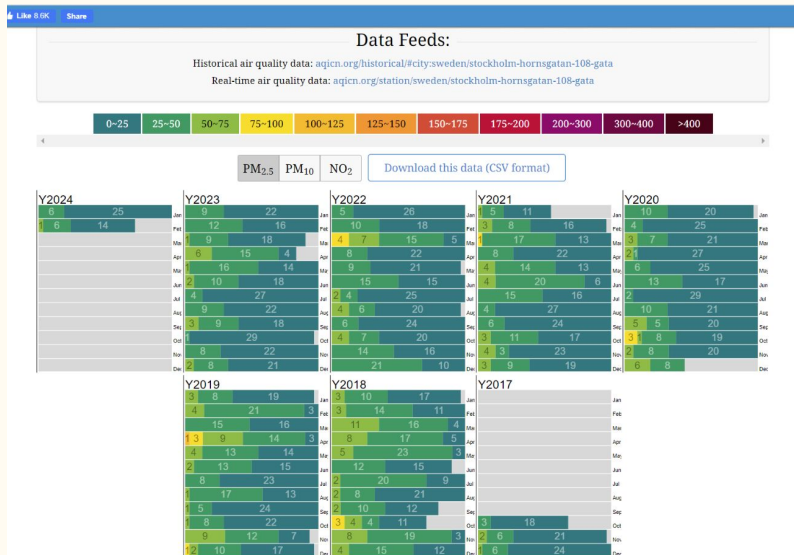
Air Quality Data Source: <https://aqicn.org>



<https://aqicn.org>

You should pick a sensor on this page - not the one in the book. Near your childhood home, country house, favorite place, wherever. It should have enough good quality measurements.

Pick an Air Quality Sensor with Good Quality Data





Free Weather API

Open-Meteo is an open-source weather API with free access for non-commercial use. No API key is required. You can use it immediately!

<https://open-meteo.com/en/docs/air-quality-api>

Overview of Lab

Dynamic Data Sources	Prediction Problem	UI or API	Monitoring
Air Quality Sensor Data: https://aqicn.info Weather Forecasts: https://open-meteo.com	Daily forecast of the level of PM _{2.5} for the next 7 days at the position of an existing air quality sensor.	A web page with graphs and a LLM-powered UI in Python.	Hindcast graphs show prediction performance of our model.

Weather Data Ingestion into Hopsworks

```
weather_df = # 1. read today's data in as a Pandas DataFrame

# 2. create features for in Pandas DataFrame

weather_fg = fs.get_or_create_feature_group(name="weather",
                                             version=1,
                                             description="Weather Daily Updates",
                                             primary_key=['city'],
                                             event_time='date'
)
weather_fg.insert(weather_df) # 3. write Pandas DataFrame to Feature Group

# ...
```

Weather Data in a Feature Group

Feature Group - **weather**

city_name	date	wind_speed_max	wind_direction_dominant	wind_gusts_max	temp_max	
<entity_id>	<event_time>	<numerical feature>	<categorical feature>	<numerical feature>	<numerical feature>	
string	datetime	double	string	double	double	
berlin	2022-01-01	14.3	ne	22.4	22.7	
dublin	2022-04-01	9.3	n	18.2	25.4	
seattle	2022-07-01	11.1	nw	15.2	20.8	
tacoma	2022-10-01	1.3	w	2.5	28.4	

Feature Types

Row

entity_id and event_time uniquely identify each row. They are not features.

Feature value.
Store unencoded to maximize reuse over many models.

Feature vector.
Set of feature values with the same primary key.

Air Quality Data Ingestion into Hopsworks

```
air_quality_df = # 1. read the most recent air quality observations

# 2. create features for in Pandas DataFrame

air_quality_fg = fs.get_or_create_feature_group(name="air_quality",
                                                version=1,
                                                description="City Air Quality Data",
                                                primary_key=['city'],
                                                expectation_suite=expectation_suite,
                                                event_time='date'
)
air_quality_fg.insert(air_quality_df) # 3. write DataFrame to Feature Group

# ...
```

Air Quality Data in a Feature Group

Feature Group - **air_quality**

city_name	date	pm2_5
<entity_id>	<event_time>	<numerical feature>
string	datetime	double
berlin	2022-01-01	5.3
dublin	2022-04-01	2.3
seattle	2022-07-01	3.1
tacoma	2022-10-01	4.3

Label

Column is the target for the prediction problem

Scheduling a Feature Pipeline to run daily with Modal

```
stub = modal.Stub("air_quality_daily")

image = modal.Image.debian_slim().pip_install(["hopsworks"])

@stub.function(image=image, schedule=modal.Period(days=1),
secret=modal.Secret.from_name("jim-hopsworks-ai"))
def g():
    ...

if __name__ == "__main__":
    stub.deploy("air_quality_daily")
    with stub.run():
        g()
```

Scheduling a Feature Pipeline to run daily with Github Actions

```
name: air-quality-daily
on:
  workflow_dispatch:
    schedule:
      - cron: '11 6 * * *'
jobs:
  schedule_pipelines:
    runs-on: ubuntu-latest
  ...
  steps:
  ...
    - name: install python packages
      run: |
        cd notebooks/ch03
        python -m pip install --upgrade pip
        pip install -r requirements.txt
    - name: execute python workflows from bash script
      env:
        HOPSWORKS_API_KEY: ${ secrets.HOPSWORKS_API_KEY }
      run: |
        cd notebooks/ch03
        jupyter nbconvert --to notebook --execute 2_air_quality_feature_pipeline.ipynb
```

Training Pipeline

```
fg_air_quality = fs.get_feature_group(name="air_quality", version=1)
fg_weather = fs.get_feature_group(name="weather", version=1)

selected = fg_air_quality.select(['pm2_5']).join(fg_weather.select_all())

fv = fs.create_feature_view(name="air_quality_fv",
                             version=1,
                             description="Weather and Air Quality",
                             labels=['pm2_5'],
                             query=selected
                             )
```


Training Pipeline

```
X_train, X_test, y_train, y_test = fv.train_test_split(test_size=0.2)

categorical_transformer=Pipeline(steps=[("encoder",
OneHotEncoder(handle_unknown="ignore"))])

preprocessor = ColumnTransformer(transformers=[ \
    ("cat", categorical_transformer, categorical_feature_ids)])

clf = Pipeline(steps=[("preprocessor", preprocessor), ("regressor",
XGBRegressor())])
clf.fit(X_train, y_train)
```

Training Pipeline

```
joblib.dump(clf, 'air_quality_model/xgboost_pipeline.pkl')
```

```
input_schema = Schema(X_test)
```

```
output_schema = Schema(y_test)
```

```
aq_model = mr.sklearn.create_model("air_quality_model",  
    metrics={'accuracy': accuracy},  
    input_example=X_test.sample().to_numpy(),  
    model_schema=ModelSchema(input_schema=input_schema,  
output_schema=output_schema))
```

```
fraud_model.save('air_quality_model')
```

Batch Inference Pipeline

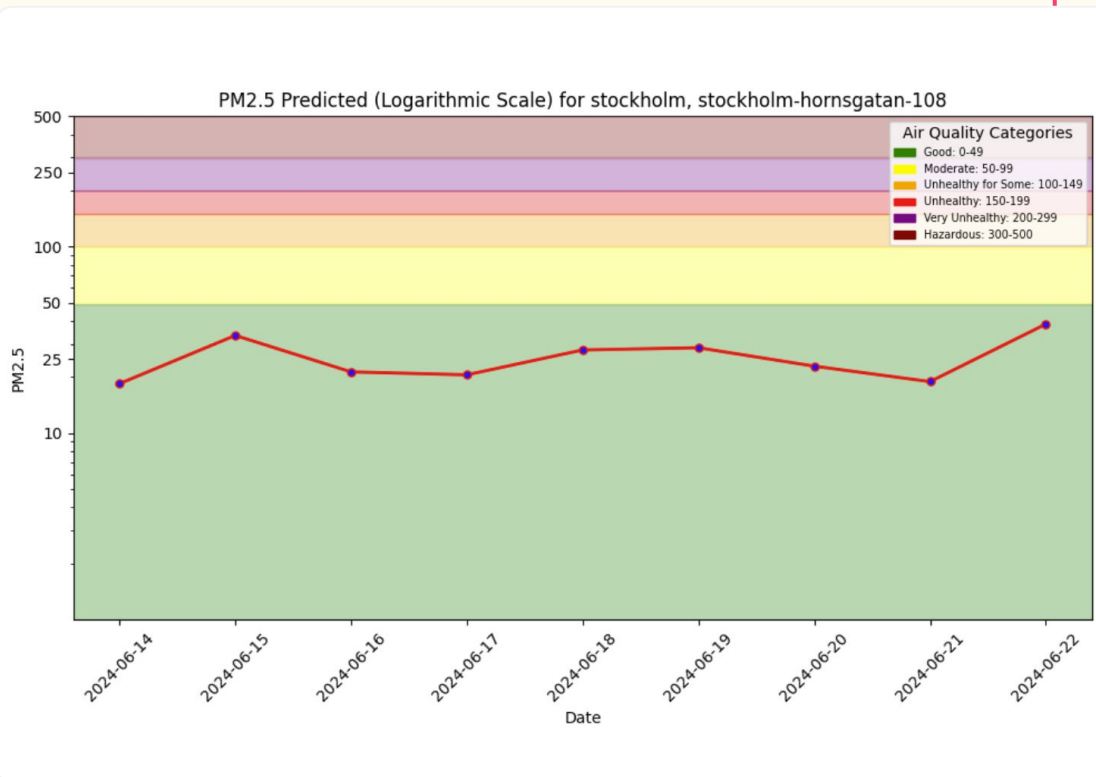
```
fv = fs.get_feature_view(name="air_quality_fv", version=1)
df = feature_view.get_batch_data(start_time=today)

mr = project.get_model_registry()
model = mr.get_model("lending_model", version=1)
model_dir = model.download()
model = joblib.load(model_dir + "/air_quality_model.pkl")

predictions_df = model.predict(df)
```

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 as a Github Page (a PNG file on a webpage - we use matplotlib to generate the PNG file)
 - Shows predictions of air quality for a place for the next few days



Task: Air Quality Prediction using Weather Features

1. Write a backfill feature pipeline that downloads historical weather data (ideally >1 year of data), loads a csv file with historical air quality data (downloaded from <https://aqicn.org>) and registers them as 2 Feature Groups with Hopsworks.
2. Schedule a daily feature pipeline notebook that downloads yesterday's weather data and air quality data, and also the weather prediction for the next 7-10 days and update the Feature Groups in Hopsworks. Use GH Actions or Modal.
3. Write a training pipeline that (1) selects the features for use in a feature view, (2) reads training data with the Feature View, trains a **regression or classifier model** to predict air quality (*pm25*). Register the model with Hopsworks.
4. Write a batch inference pipeline that creates a dashboard. The program should download your model from Hopsworks and plot a dashboard that predicts the air quality for the next 7-10 days for your chosen air quality sensor.
5. Monitor the accuracy of your predictions by plotting a hindcast graph showing your predictions vs outcomes (measured air quality).
6. Update your Model by adding a new feature, lagged air quality for the previous 1-3 days.

Deliverables

- Deliver your source code as a Github Repository.
- Deliver your lab description as a README.md file in the root of your project directory in your Github repository
- Deliver a public URL for the Dashboard.

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.

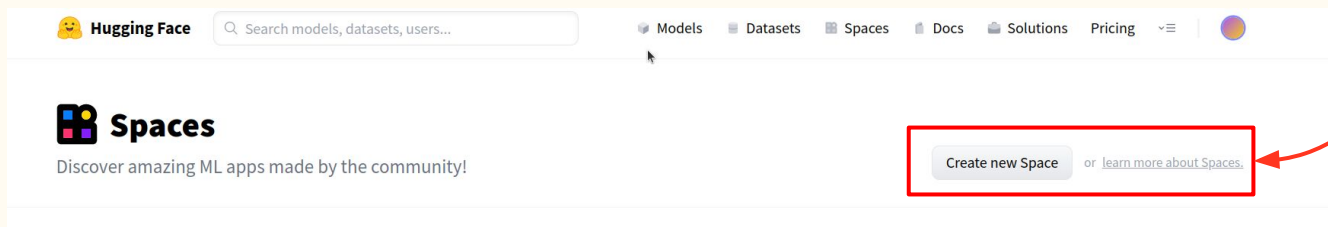
Grading

- The grade for this lab will be awarded if you (1) complete the tasks below, (2) and answer our questions during the grading defence.
- As a guide to your grade, assuming you answer questions successfully, you can expect:
 - Steps 1-4: Grade C
 - Steps 1-5: Grade B
 - Steps 1-6: Grade A

Alternative Dashboard developed in Python

- You can develop your own UI in Python if you prefer over GitHub Pages
 - Good frameworks are Gradio or Streamlit
 - You can deploy them for free on a serverless platform like HuggingFace or Streamlit Cloud

Optional: Register and Create a Hugging Face Space



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

A screenshot of the 'Create a new Space' form on Hugging Face. The form includes fields for 'Owner' (set to 'jdwoling') and 'Space name' (set to 'iris'). The 'License' is set to 'apache-2.0'. Under 'Select the Space SDK', three options are shown: Streamlit, Gradio (which is highlighted with a yellow border), and Static. At the bottom, there are radio buttons for 'Public' (selected) and 'Private'. A 'Create space' button is at the bottom left. A red arrow points from a text box on the right towards the 'Gradio' option.

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

Add a HOPSWORKS_API_KEY as a secret in your Space

The screenshot shows the Hugging Face Spaces interface. At the top, there's a navigation bar with links for Models, Datasets, Spaces, Docs, Solutions, and Pricing. Below this, the page title is 'Spaces: jdownling/iris', with a 'like' button and a 'Running' status indicator. The 'Settings' tab is selected, showing options for App, Files and versions, Community, and Settings. The 'Space Hardware' section is active, displaying various hardware configurations like 'CPU basic', 'CPU upgrade', 'T4 small', 'T4 medium', 'A10G small', and 'A10G large'. Below the hardware section, the 'Repo secrets' section is visible, containing a single secret named 'HOPSWORKS_API_KEY'. A red box highlights this secret name, and a red arrow points from it to a text box at the bottom of the slide.

Space Hardware

Choose a hardware for your Space.

You'll be billed on a per minute basis.
View usage in your [billing settings](#).

Display price: per hour ● per month

CPU basic (Current · Free)
2 vCPU · 16 GiB RAM

CPU upgrade
8 vCPU · 32 GiB RAM
\$0.03/hour

T4 small
4 vCPU · 15 GiB RAM · Nvidia T4
\$0.6/hour

T4 medium
8 vCPU · 30 GiB RAM · Nvidia T4
\$0.9/hour

A10G small
4 vCPU · 15 GiB RAM · Nvidia A10G
\$1.05/hour

A10G large
12 vCPU · 46 GiB RAM · Nvidia A10G
\$3.15/hour

AI Accelerator
HPU · IPU · ...
Coming soon

Building something cool as a side project?
Apply for a [community GPU grant](#).

Repo secrets

HOPSWORKS_API_KEY

Remove

1. Add your HOPSWORKS_API_KEY as a Repo Secret