Pokémon and Machine Learning: Mega-evolve Prediction

December 11, 2025
Machine Learning Operations Final Project

Group 7 – Bradley Stoller, Samuel Martinez Koss, Xigang Zhang, and Zhiwei Guo

THE UNIVERSITY OF CHICAGO | MS-APPLIED DATA SCIENCE

Agenda

1 **Dataset Introduction and EDA Insights**

Brief overview of the data and key insights from the EDA

2 **Pipeline Staging and Evaluation Framework**

Review of pipeline and metrics used for evaluation

3 **Modeling Methodology**

Summary of AutoML and Mlflow results

4 **Model Monitoring Approach**

Overview of model monitoring setup and usage method

5 **Data Change**

Discussion of data changes for drift detection confirmation

6 **Results and Demo**

Review of all results and model monitoring drift detection demo

Agenda

1 **Dataset Introduction and EDA Insights**

Brief overview of the data and key insights from the EDA

2 **Pipeline Staging and Evaluation Framework**

Review of pipeline and metrics used for evaluation

3 **Modeling Methodology**

Summary of AutoML and Mlflow results

4 **Model Monitoring Approach**

Overview of model monitoring setup and usage method

5 **Data Change**

Discussion of data changes for drift detection confirmation

6 **Results and Demo**

Review of all results and model monitoring drift detection demo

Introduction – Why Pokémon?

Data Background

- From Niantic, the company developed Pokémon Go
- It contains 21 features (variables) spanning 721 Pokémon (N=721)

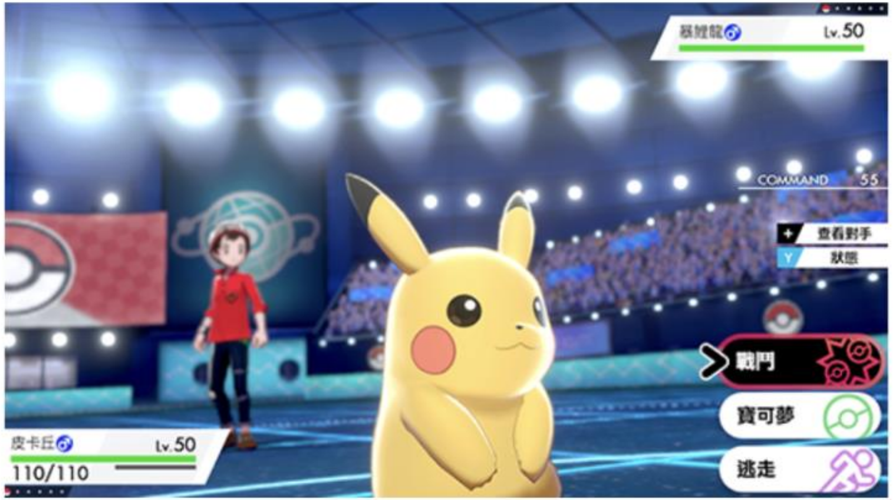
Analysis Objective

- Strongest primary type (species)
- Features that correlated with Mega-evolve
- Build a **predictive model** to determine whether a Pokémon can **Mega-evolve**

Analysis Significance

- For the Pokémon game itself, provide data-driven insights about Pokémon stats to guide future balance and design
- Enable Pokémon-related companies to use data-driven decision making for designing new Pokémon and potentially increasing revenue
- For Pokémon players (like me!), this helps us understand which Pokémon are most likely to succeed in battle

What is a Mega Evolution?



- Increased attack/defense
- Bonus skills
- Exclusive features

High-level Dataset Introduction: Pokémon Characters

721	21	13	7	2
Total Rows	Features Available	Numerical Features	Categorical Features	Prediction Classes

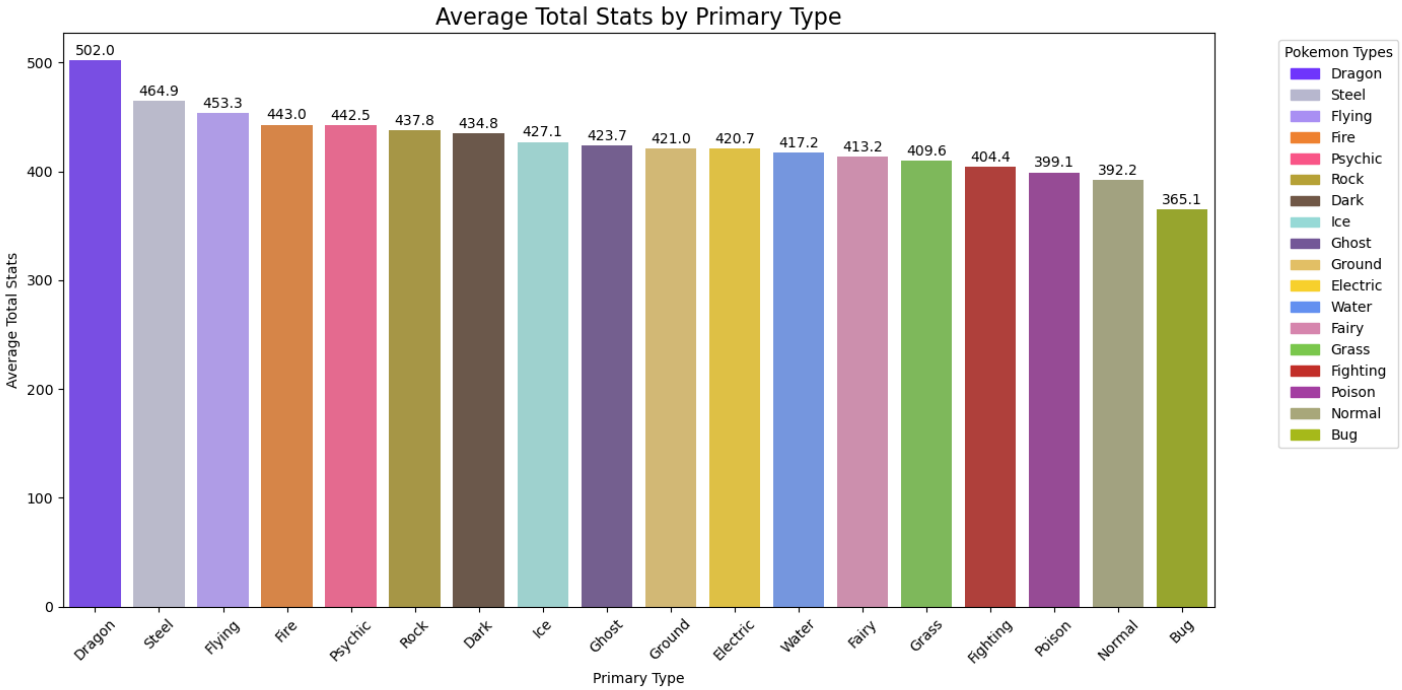
Dataset Overview:

- This dataset focuses on the stats and features of Pokémon from Generations 1 through 6
- It contains 21 features (variables) spanning 721 Pokémon

Target Variable:

hasMegaEvolution

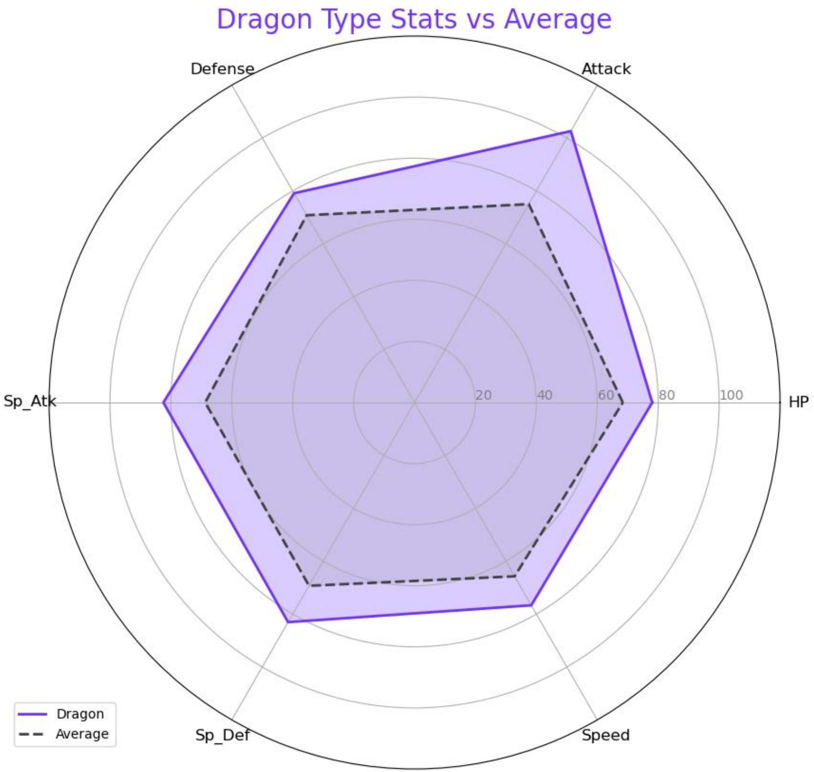
Overall, Dragon is the most powerful Primary Type across the dataset



Dragon has highest average total scores

Only Dragon has 500+ average total score

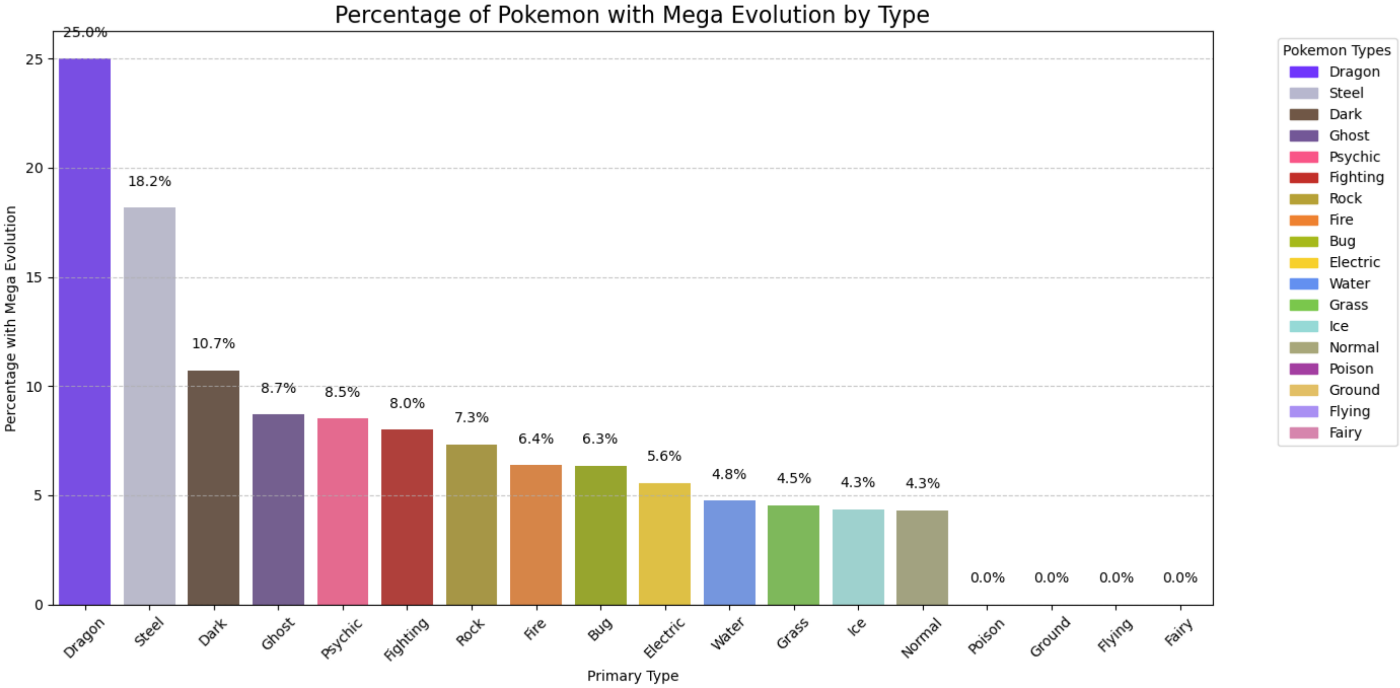
Specifically, Dragon's stats is higher than all other types' stat



Dragon’s average stats is higher than all other characters

Attack is the most significant one

However, beyond Dragon, we see that the data presents a class imbalance



Severe class imbalance present

Used SMOTE for effective modeling

Agenda

1 **Dataset Introduction and EDA Insights**

Brief overview of the data and key insights from the EDA

2 **Pipeline Staging and Evaluation Framework**

Review of pipeline and metrics used for evaluation

3 **Modeling Methodology**

Summary of AutoML and Mlflow results

4 **Model Monitoring Approach**

Overview of model monitoring setup and usage method

5 **Data Change**

Discussion of data changes for drift detection confirmation

6 **Results and Demo**

Review of all results and model monitoring drift detection demo

For the evaluation metric, we chose accuracy since it is the most wholistic

Predicted

Actual		Evolution	No Evolution
	Evolution	**TP**	**FN**
	No Evolution	**FP**	**TN**

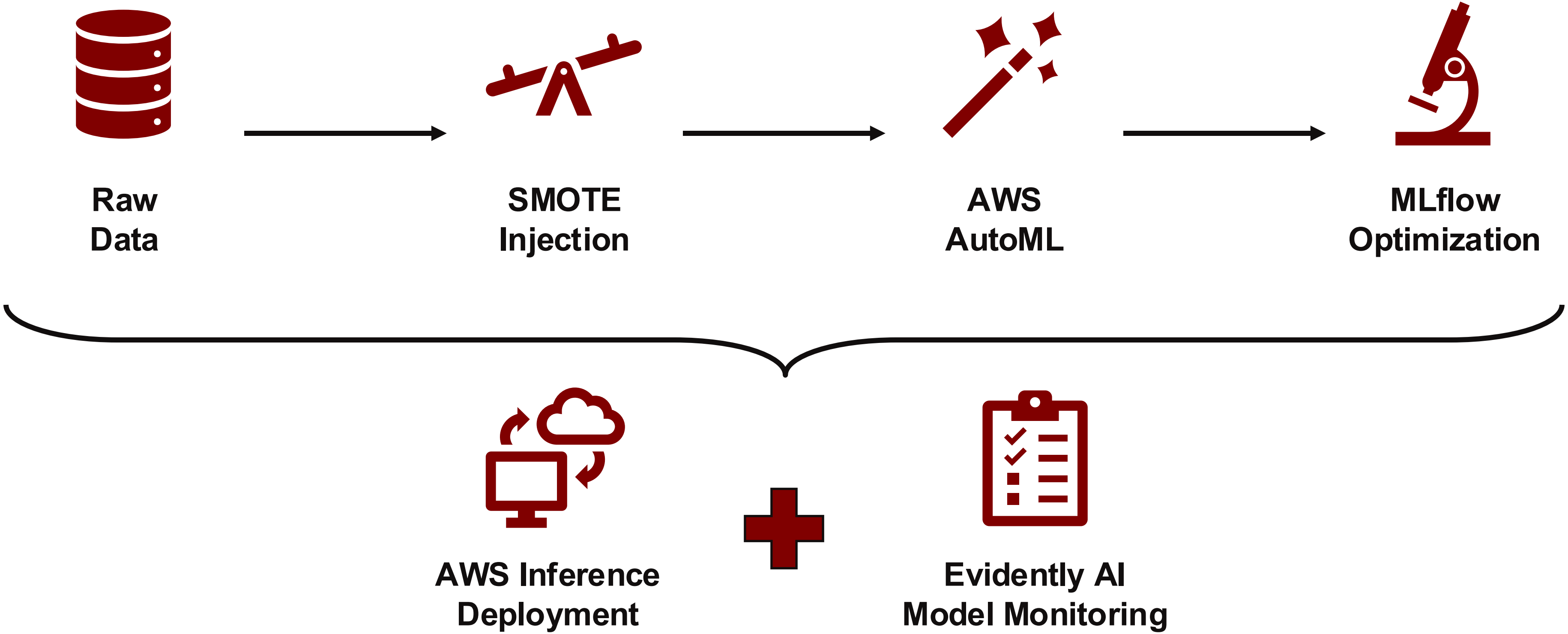
Accuracy:

$$\frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Overall Correctness

- Balanced representation of both classes after SMOTE
- Easy to interpret with regards to character evolution
- Appropriate given the equal cost of FPs and FNs

For prediction, we used a multi-stage pipeline with AutoML, MLflow, and AWS



Specifically, our MLflow pipeline tests 10 models via random search

n_estimators	learning_rate	max_depth	min_samples_split
2 – 7	0.01 – 0.1	2 – 4	5 – 20

RANDOM SEARCH

1. n_estimators: 2, learning_rate: 0.054, max_depth: 3, min_samples_split: 11
2. n_estimators: 7, learning_rate: 0.080, max_depth: 2, min_samples_split: 6
3. n_estimators: 5, learning_rate: 0.086, max_depth: 3, min_samples_split: 15
4. n_estimators: 7, learning_rate: 0.022, max_depth: 3, min_samples_split: 11
5. n_estimators: 5, learning_rate: 0.010, max_depth: 2, min_samples_split: 16
6. n_estimators: 5, learning_rate: 0.092, max_depth: 3, min_samples_split: 11
7. n_estimators: 3, learning_rate: 0.065, max_depth: 2, min_samples_split: 18
8. n_estimators: 2, learning_rate: 0.093, max_depth: 2, min_samples_split: 14
9. n_estimators: 2, learning_rate: 0.045, max_depth: 3, min_samples_split: 6
10. n_estimators: 7, learning_rate: 0.099, max_depth: 3, min_samples_split: 16

Agenda

1 **Dataset Introduction and EDA Insights**

Brief overview of the data and key insights from the EDA

2 **Pipeline Staging and Evaluation Framework**

Review of pipeline and metrics used for evaluation

3 **Modeling Methodology**

Summary of AutoML and Mlflow results

4 **Model Monitoring Approach**

Overview of model monitoring setup and usage method

5 **Data Change**

Discussion of data changes for drift detection confirmation

6 **Results and Demo**

Review of all results and model monitoring drift detection demo

From AWS AutoML, we found that gradient boosting was the best algorithm

AutoML Model Search

Best Model Selected

Gradient Boosting (XG-Boost)

Best Candidate Model

- **Accuracy:** 97%
- **Objective Metric:** Validation Accuracy
- **Detected Algorithm Family:** XG-Boost/Gradient Boosting

Best Candidate Model Hyperparameters

- **max_depth:** 4
- **eta:** 0.66
- **num_round:** 364
- **colsample_bytree:** 0.96
- **gamma:** 0.00035
- **lambda:** 0.96

After MLflow experimentation, our deployed model achieves 88% train accuracy

88%

Train Accuracy

n_estimators:
7

learning_rate:
0.099

max_depth:
3

min_samples_split:
16

Agenda

To monitor data and prediction drift, we used Evidently AI via the AWS Suite

SageMaker Model → Evidently AI Script → S3 Record Archive → Monitoring Dashboard

EVIDENTLY AI

aws

→ **Evidently integrates seamlessly with AWS workflow**

→ **S3 centralizes records for reliable audits and traceability**

Agenda

We made four key changes to X_test to test the model monitoring effectiveness

Change #1:	Swapped column 0 and column 1
Change #2:	Swapped column 2 and column 7
Change #3:	Randomized column 2
Change #4:	Randomized column 4

New Accuracy 74%
(16% decrease)

Agenda

1 **Dataset Introduction and EDA Insights**

Brief overview of the data and key insights from the EDA

2 **Pipeline Staging and Evaluation Framework**

Review of pipeline and metrics used for evaluation

3 **Modeling Methodology**

Summary of AutoML and Mlflow results

4 **Model Monitoring Approach**

Overview of model monitoring setup and usage method

5 **Data Change**

Discussion of data changes for drift detection confirmation

6 **Results and Demo**

Review of all results and model monitoring drift detection demo

Demo – Model Monitoring In Action!

Notebooks > ml_ops_final_project

ml_ops_final_project

Add description

Run all

Last saved: 1 minute ago

5b. Deploy the native XG-Boost model to AWS for inference

1 minute ago | Elapsed time: 3s

Python

```python
# Set S3 configurations for model saving
MODEL_KEY = 'model/model.tar.gz'
MODEL_S3_PATH = f's3://{BUCKET_NAME}/{MODEL_KEY}'
ENDPOINT_NAME = 'pokemon-model'

# Save the model
xgb_model.save_model('xgboost-model')

# Convert to tar.gz format (SageMaker requirement)
with tarfile.open('model.tar.gz', 'w:gz') as tar:
    tar.add('xgboost-model')
print("Model packaged as model.tar.gz")

# Upload to S3
S3_CLIENT.upload_file(
    Filename='model.tar.gz',
    Bucket=BUCKET_NAME,
    Key= MODEL_KEY
)
print(f"Model uploaded to: {S3_PATH}")

xgb_model_sm = XGBoostModel(
    model_data= MODEL_S3_PATH,
    role= ROLE_ARN,
    framework_version= '1.7-1',
    sagemaker_session= SESSION
)

print("\nDeploying model to SageMaker endpoint...")

```

Python 3.11 | 2 vCPU • 4 GiB | 4.0% | 53.9% | 2.0% | Ready

Drift Report Comparison: X_test

1	1	0	0	0
Tests	Success	Warning	Fail	Error

All tests

Number of Drifted Features

The drift is detected for 2 out of 21 features. The test threshold is lt=3.

Details

Feature name	Stattest	Drift score	Threshold	Data Drift
Type_2	K-S p_value	0.017	0.05	Detected
Has_Type_2	Z-test p_value	0.002	0.05	Detected
prediction	Z-test p_value	0.163	0.05	Not detected
Type_1	K-S p_value	0.562	0.05	Not detected
Total	K-S p_value	0.053	0.05	Not detected
Speed	K-S p_value	0.217	0.05	Not detected
Sp_Def	K-S p_value	0.08	0.05	Not detected
Sp_Atk	K-S p_value	0.468	0.05	Not detected
Height_m	K-S p_value	0.17	0.05	Not detected
Weight_kg	K-S p_value	0.545	0.05	Not detected
Has_Egg_Group_2	Z-test p_value	0.077	0.05	Not detected
Generation	K-S p_value	0.945	0.05	Not detected
Egg_Group_2	K-S p_value	0.545	0.05	Not detected
Egg_Group_1	K-S p_value	0.999	0.05	Not detected
Defense	K-S p_value	0.056	0.05	Not detected

Drift Report Comparison: X_test_modified

1	0	0	1	0
Tests	Success	Warning	Fail	Error

All tests

Number of Drifted Features — Details

The drift is detected for 7 out of 21 features. The test threshold is lt=3.

Feature name	Stattest	Drift score	Threshold	Data Drift
prediction	Z-test p_value	0.0	0.05	Detected
Attack	K-S p_value	0.0	0.05	Detected
Type_2	K-S p_value	0.0	0.05	Detected
Type_1	K-S p_value	0.0	0.05	Detected
Total	K-S p_value	0.0	0.05	Detected
Sp_Def	K-S p_value	0.005	0.05	Detected
Has_Type_2	Z-test p_value	0.002	0.05	Detected
Speed	K-S p_value	0.217	0.05	Not detected
Sp_Atk	K-S p_value	0.468	0.05	Not detected
Height_m	K-S p_value	0.17	0.05	Not detected
Has_Egg_Group_2	Z-test p_value	0.077	0.05	Not detected
HP	K-S p_value	0.295	0.05	Not detected
Generation	K-S p_value	0.945	0.05	Not detected
Egg_Group_2	K-S p_value	0.545	0.05	Not detected

GitHub Link:

https://github.com/bhstoller/ml-ops-fp/tree/main

Questions?