AI/ML for Climate Workshop

International Livestock Research Institute (ILRI)

---

hide: - toc

🤖 Introduction to Machine Learning for Weather & Climate

1. Learning Goals

By the end of this module, you will be able to:

- Understand what “machine learning” means in an operational climate / meteorological context.
- Recognize common supervised ML tasks in climate science (regression, classification).
- Explain the difference between training, validation, and testing.
- Describe typical inputs (predictors/features) and outputs (predictands/targets) in seasonal and subseasonal prediction.
- Train a simple ML model in Python using `scikit-learn`.

---

2. Why ML in Climate and Meteorology?

Traditional NWP (Numerical Weather Prediction) and physics-based seasonal models are powerful, but they have limitations:

- **Biases and systematic errors** in forecasts.
- **Resolution gaps** (global models are often too coarse for local decisions).
- **Cost / compute constraints** for running large ensembles in real time.

Machine learning can help with:

- **Bias correction & calibration**
  e.g. adjusting a raw seasonal forecast to better match observed rainfall over the Greater Horn of Africa.

- **Downscaling / localization**
  e.g. taking coarse 1° x 1° model output and predicting station-scale rainfall probability.

- **Predicting impacts, not just weather**
  e.g. "likelihood of below-normal rainfall affecting crop yield in eastern Ethiopia."

> ⚠ *ML does **not** replace physics.*
> *It helps learn statistical relationships between inputs (e.g. SST, winds, geopotential height) and outcomes (e.g. rainfall tercile).*

---

3. Core ML Terms (Climate Context)

Feature / Predictor

A variable we use as input to the model (e.g. Niño3.4 index, soil moisture anomaly, 850 hPa wind, previous month's rainfall).

Target / Label / Predictand

What we're trying to predict (e.g. rainfall category: below/normal/above for OND season in the GHA).

Regression vs Classification

- **Regression:** predict a number
  Example: "Forecast total mm of rainfall for next week in Addis Ababa."

- **Classification:** predict a category
  Example: "Is next OND season likely to be 'below normal', 'normal', or 'above normal'?"

Training / Validation / Testing

- **Training set:** we fit the model here.
- **Validation set:** we tune hyperparameters here.
- **Test set:** we evaluate final skill here.

In climate, we often split data **by time** (e.g. train on 1993–2015, test on 2016–2024) to simulate "forecasting the future."

---

4. Typical ML Workflow for Seasonal/Subseasonal Prediction

1. **Data collection**
2. Observations / reanalysis (e.g. CHIRPS rainfall, ERA5 winds).
3. Model forecasts / ensemble members (e.g. GCM seasonal forecasts).
4. Climate indices (e.g. IOD, ENSO).
5. **Preprocessing**
6. Compute anomalies (remove the climatological mean).
7. Spatial averaging (e.g. average rainfall over a GHA subregion).
8. Temporal aggregation (e.g. last 30 days, last 90 days).
9. **Feature engineering**
10. Build meaningful predictors, like:
    - Regional mean SST anomaly (western Indian Ocean)
    - Zonal wind shear
    - Previous month rainfall deficit
11. Scale / normalize as needed.
12. **Model training**
13. Fit a regression or classification model.
14. **Evaluation**
15. Use skill metrics (correlation, RMSE, ACC, Brier score, ROC AUC, etc.).
16. Check reliability and sharpness (is the forecast probabilistic and trustworthy?).
17. **Deployment**
18. Generate an operational forecast for “next period.”
19. Communicate uncertainty.

---

5. A Minimal Hands-On Example

Below is a **toy** supervised learning example in `scikit-learn`.
The goal: predict rainfall anomaly (mm/day) from a few climate indices.

> *In the real workshop, you will replace the dummy `X` and `y` arrays with: - Predictors from reanalysis / model output - Target from observed rainfall over your domain*

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

# ----------------------------
# Example: X are "features"
#   col 0: ENSO index (e.g. Niño3.4 SST anomaly)
#   col 1: Indian Ocean Dipole index
#   col 2: previous month's regional rainfall anomaly
#
# y is the rainfall anomaly we're trying to predict for next month
# ----------------------------

# Fake data for illustration only
X = np.array([
    [1.2, -0.3,  5.1],
    [0.8, -0.1,  4.7],
    [0.1,  0.4, -2.0],
    [-0.5, 0.9, -3.1],
    [-1.0, 1.2, -4.0],
    [0.4, -0.2, 2.2],
    [1.1, -0.4, 4.9],
    [-0.8, 1.1, -3.6],
])

y = np.array([
    6.2,
    5.9,
    -1.8,
    -2.5,
    -3.2,
    2.4,
    5.8,
    -2.9
])

# Split train/test (for demo we'll just cut in half)
X_train, X_test = X[:5], X[5:]
y_train, y_test = y[:5], y[5:]

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Predictions:", y_pred)
print("Truth:", y_test)
```

```
print("R² score:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

Discussion:

- Does the model capture the sign (wet vs dry anomaly)?
- Is skill (R^2, RMSE) "good enough" to trust it in operations?
- Which predictors seem most important?

We can inspect learned coefficients:

```
for name, coef in zip(["ENSO", "IOD", "PrevRain"], model.coef_):
    print(f"{name}: {coef:.3f}")

print("Intercept:", model.intercept_)
```

This tells you how strongly each climate signal influences the predicted rainfall anomaly in this simple linear model.

---

6. Classification Example (Tercile Forecasting)

Seasonal forecast centers often issue tercile forecasts: - **Below Normal** - **Normal** - **Above Normal**

Below is a simple example using `LogisticRegression` for classification.

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Let's say classes are:
# 0 = Below Normal
# 1 = Normal
# 2 = Above Normal
#
# X = predictors (same idea: ENSO, IOD, previous rainfall anomaly)
# y = rainfall category next season

X = np.array([
    [1.2, -0.3,  5.1],
    [0.8, -0.1,  4.7],
    [0.1,  0.4, -2.0],
    [-0.5, 0.9, -3.1],
    [-1.0, 1.2, -4.0],
    [0.4, -0.2, 2.2],
    [1.1, -0.4, 4.9],
    [-0.8, 1.1, -3.6],
```

```
])

y = np.array([2, 2, 1, 0, 0, 1, 2, 0])

X_train, X_test = X[:5], X[5:]
y_train, y_test = y[:5], y[5:]

clf = LogisticRegression(multi_class="multinomial", max_iter=500)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print("Predicted classes:", y_pred)
print("True classes     :", y_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("
Full report:
", classification_report(y_test, y_pred))
```

Operational relevance:

- You can map class probabilities (e.g. 70% chance “below normal”) over the Greater Horn of Africa.
- That turns into a forecast product that NMHSs and ICPAC can communicate.

---

7. Limitations and Good Practice

Limitations - Small sample sizes (seasonal data = maybe only 30–40 years of hindcasts). - Climate is non-stationary (warming trend, changing teleconnections). - Risk of overfitting if you add too many predictors.

Good practice - Use cross-validation that respects time order. - Always compare against a baseline (e.g. climatology, persistence). - Communicate uncertainty (probabilities, confidence).

---

8. Exercises / Your Turn

1. **Build your own dataset**
2. Choose 2–4 predictors you believe affect OND rainfall in your country.

3. Construct `X` and `y` arrays from historical data.

4. **Train a regression model**

5. Fit `LinearRegression` and report RMSE.

6. **Train a classification model**

7. Define categories (below/normal/above).
8. Fit `LogisticRegression`.

9. Report accuracy.

10. **Interpretability**

11. Which predictors were most important?

12. Does that match known physical drivers (ENSO, IOD, etc.)?

13. **Bonus (if time):**

14. Plot forecast vs observed on a time series.
15. Make a simple reliability diagram.

---

9. Takeaways

- ML in climate is not magic — it's statistics + domain knowledge.
- The value comes from combining:
- physically meaningful predictors,
- careful validation,
- and clear communication of uncertainty.
- You'll build on this in the next module: **"ML Workflow for Weather & Climate"** (`day4/04-ml-workflow.md`),
  which focuses on making this operational and reproducible.
