Project Overview: Predictive Modeling of Weather and Interpretable Neural Architectures

Part 1: Weather-Based Cloud Cover Forecasting Using Logistic Regression

This project focuses on building a **logistic regression model** to forecast **cloud cover** six hours into the future using **historical hourly weather data**. A long-term weather dataset (e.g., Hawaii, spanning approximately 40 years) is used for model training and evaluation.

Key steps include:

- Ingesting and preprocessing weather data from structured repositories (e.g., public datasets in CSV or JSON formats).
- Using **current atmospheric variables** (e.g., temperature, humidity, wind speed) to predict future cloud cover states.
- Ensuring **rigorous evaluation** using **ROC (Receiver Operating Characteristic) curves** and calculating the **AUC (Area Under the Curve)** to assess classification performance.
- Designing a robust **training-test split** to avoid data leakage and ensure temporal independence.
- Testing **model generalizability** by applying the Hawaii-trained model to a separate location's weather data (e.g., Stockholm) to assess cross-regional performance transferability.

Part 2: Interpretable Neural Networks for Classical Learning Problems

This part explores how simple neural network architectures can **reproduce well-known machine learning models**, with a focus on **single-neuron networks** and custom configurations.

Key tasks include:

- Designing neural networks (with appropriate choices of **activation functions**, **loss functions**, and **regularization terms**) that replicate:
  - **Linear Regression**
  - **Linear Regression with Regularization** (Lasso or Ridge)
  - **Binary Logistic Regression**
  - *(Optional)* **Perceptron Classifier**
  - *(Optional)* **Support Vector Machines** with linear kernels
- Implementing and training models using **Stochastic Gradient Descent (SGD)** or similar optimizers.
- Comparing learned parameters and predictive performance with those from **closed-form or convex-optimized solutions**.
- Using PyTorch, TensorFlow (or related wrappers like Keras) to implement and evaluate model behavior under consistent training conditions.