Objective

Implement Ridge regression on the California Housing dataset using closed-form and gradient descent methods. Evaluate and compare performance, experiment with hyperparameters, and demonstrate additional skills through additional subtasks that test UI integration, problem-solving, and creativity.

Dataset

The California Housing dataset contains 20,640 samples with 8 numerical features and a target variable (median house value). It's accessible via scikit-learn and suitable for local computation.

Tasks

Section 1: Data Preparation and Model Implementation (Mandatory)

- **Data Preparation**:
  - Load the dataset using scikit-learn.
  - Handle missing values (if any) and standardize features.
  - Split into 80% training and 20% test sets.
- **Closed-Form Solution**:
  - Implement Ridge regression using the normal equation with L2 regularization.
  - Train on the training set with a regularization parameter (lambda) of 1.0.
- **Gradient Descent**:
  - Implement Ridge regression using gradient descent, including the L2 regularization term.
  - Choose a learning rate and number of iterations (or use early stopping).
  - Train on the training set with lambda = 1.0.

Section 2: Model Evaluation and Comparison (Mandatory)

- Evaluate both implementations on the test set using Mean Squared Error (MSE) and R-squared metrics.
- Compare results with scikit-learn's Ridge regression using lambda = 1.0.
- Present results in a clear table or plot.

Section 3: Hyperparameter Experimentation (Mandatory)

- Experiment with at least three lambda values (e.g., 0.1, 1.0, 10.0).
- Plot MSE and R-squared against lambda to show regularization effects.
- For gradient descent, test three learning rates (e.g., 0.001, 0.01, 0.1).
- Plot loss over iterations for each learning rate to demonstrate convergence.

Section 4: Advanced Tasks (Choose at Least One)

Complete as many as time allows to demonstrate additional skills:

- **Interactive Visualization**:
  - Use Plotly to create interactive plots for hyperparameter experiments, allowing users to explore performance metrics dynamically.
- **Feature Engineering**:
  - Perform feature selection (e.g., based on correlation) or create new features (e.g., polynomial terms).
  - Train the model on the modified features and compare performance.
- **Alternative Regularization**:
  - Implement Lasso regression (L1 regularization) from scratch.
  - Compare its performance with Ridge regression.
- **Cross-Validation**:
  - Implement 5-fold cross-validation to select the optimal lambda.
  - Report the chosen lambda and test set performance.
- **Model Interpretability**:
  - Use SHAP to analyze feature importance.
  - Discuss which features most influence predictions.
- **Interactive Interface**:
  - Use ipywidgets to create an interactive tool in the notebook.
  - Allow users to adjust lambda or input feature values to see predictions.
- **Creative Extension**:
  - Propose and implement an innovative improvement (e.g., handling outliers, non-linear modeling).
  - Justify the approach and evaluate its impact.

Deliverables

- A well-documented Jupyter notebook containing:
  - Code for all mandatory tasks (Sections 1-3).
  - Results, plots, and a brief discussion (3-5 sentences) on Sections 1-3 outcomes.
  - Code and explanations for chosen advanced tasks (Section 4).
- Submit via a GitHub repository.

Allowed Libraries

- NumPy for math operations.
- Pandas for data handling.
- Matplotlib for static plots.
- scikit-learn for data loading, splitting, and comparison.
- For Section 4: Plotly, SHAP, ipywidgets, or others as needed.

Notes

- Prioritize Sections 1-3. Section 4 is for showcasing additional skills if time permits.
- Implementations must be from scratch for Ridge and Lasso (if chosen), using only NumPy for math.
- Do not use scikit-learn’s Ridge or Lasso for your implementations, only for comparison.
- Ensure code is well-commented and the notebook is organized.