Linear Regression HW

Learning Objectives

- Implement linear regression from scratch using:
  - Gradient descent with Sum of Squared Errors (SSE) cost function.
  - Normal equation (closed-form solution).
- Explore the role of cost functions in optimization.
- Design modular, robust classes following good software engineering and machine learning practices (e.g., scikit-learn-style API, input validation).
- Compare your implementation's performance to scikit-learn's `LinearRegression`.

Assignment Description

You will develop a linear regression library with:

1. A base class defining a common interface (e.g., `fit`, `predict`, `score`).
2. Two subclasses:
   - `LinearRegressionGD`: Implements gradient descent with SSE cost function.
   - `LinearRegressionNE`: Implements the normal equation.

Your implementation must:

- Use NumPy for computations; avoid external ML libraries for core fitting logic.
- Be evaluated on the provided `housing_toy_data.csv` dataset (features: size, bedrooms, age; target: price in thousands).

*A **sample Jupyter notebook template (https://uvu.instructure.com/courses/616319/files/129072770?wrap=1)** **(https://uvu.instructure.com/courses/616319/files/129072770/download?download_frd=1)** is provided with skeleton code, optional unit tests, comparison code, and visualizations. You may follow this template or design your own solution (e.g., as a Python script), as long as it meets the requirements.*

Dataset (https://uvu.instructure.com/courses/616319/files/129072769?wrap=1) (https://uvu.instructure.com/courses/616319/files/129072769/download?download_frd=1)

Load the dataset from `housing_toy_data.csv` **(https://uvu.instructure.com/courses/616319/files/129072769?wrap=1)** **(https://uvu.instructure.com/courses/616319/files/129072769/download?download_frd=1)** :

```
import pandas as pd
df = pd.read_csv('housing_toy_data.csv')
X = df[['size', 'bedrooms', 'age']].values
y = df['price'].values
```

Train and evaluate your models on this dataset. Compare results to scikit-learn's `LinearRegression`.

Requirements

1. **Implementation**:
   - **Base Class**: Define a base class (e.g., `BaseLinearRegression`) with methods `fit`, `predict`, and `score` (returns R^2). Include `fit_intercept` parameter and attributes like `coef_` and `intercept_`.
   - **Gradient Descent (GD)**: Implement using SSE cost function $J(\theta) = \frac{1}{2m}\sum(h_\theta(x) - y)^2$ J(θ)=2m1∑(hθ(x)−y)2. Support parameters:
     - `learning_rate` (float, default=0.01): The step size for weight updates.
     - `max_iter` (int, default=1000): The maximum number of iterations.
     - `random_state` (int, optional): Seed for random number generation.
     - other hyperparameters like `batch_size` (int, default=32) for mini-batch gradient descent, etc.

     Track cost history for analysis.
   - **Normal Equation (NE)**: Implement using $\theta = (X^T X)^{-1} X^T y$θ=(XTX)−1XTy.
   - Follow good practices: modular design, clear documentation, and type hints.
2. **Comparison**:
   - Compare your GD and NE models to scikit-learn's `LinearRegression` on the dataset.
   - Report coefficients, intercept, R^2 R2, MSE, and fit time.
   - Visualize: actual vs. predicted scatter plots for each model and cost history for GD.
3. **Analysis**:

- **Learning Rate**: Analyze how different learning rates affect convergence and final metrics.
- **Batch Size**: Investigate the impact of different batch sizes on training stability and performance.
- **Cost Function**: Discuss the choice of cost function (SSE vs. MSE) and its implications for model training and evaluation.

- **Coefficients**: Analyze how the coefficients learned by your models compare to those from scikit-learn.
- **Implementation Details**: Reflect on the challenges encountered during implementation and how they were addressed.

Grading Rubric

Component	Points
Implementation (60 points)	
Base Class	5
Gradient Descent	30
Normal Equation	25
Comparison & Visualization (20 points)	
Metrics	10
Visualizations	10
Analysis (20 points)	
Learning Rate & Batch Size	10
Cost Function	5
Coefficients	5
Implementation Details	5

Submission Guidelines

- Submit a Jupyter notebook (or Python script) containing:
  - Complete implementation (base class, GD, NE).
  - Comparison code with metrics and visualizations.
  - Markdown cells answering analysis questions.

Resources

- NumPy: **numpy.org/doc/stable/ (https://numpy.org/doc/stable/)**

- Scikit-learn API: **scikit-learn.org/stable/developers (https://scikit-learn.org/stable/developers)**