Alex Rios 801320278
ECGR-4105 Summer 2025
Report for assignment2

GitHub Repository: https://github.com/TheluckyEngineer101/AR-assignment2-sum25

Objective: Implement and evaluate linear regression using gradient descent to predict housing prices, exploring feature combinations, input scaling, and regularization.

1. Data Loading and Visualization

```python
# Alex Rios 801320278
# Assignment2 Summer 2025

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd

# go ahead load and find the path for my dataset
df = pd.read_csv('../Datasets/Housing.csv')
df.head() # To get first n rows from the dataset default value of n is 5
M=len(df)
M

print(f"Loaded {M} samples.") # rather than a random "100" I will add addtional info on what 100 means.
print(df.head()) # from the df.head() I went ahead and added print to make it show our first 4 rows
```

```
Loaded 545 samples.
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
```

Additional data loading and statistics
Approach: Used df.describe() this function is used for Generating descriptive stats for each column such as [count, mean, std, min/max and percentiles]
Result:

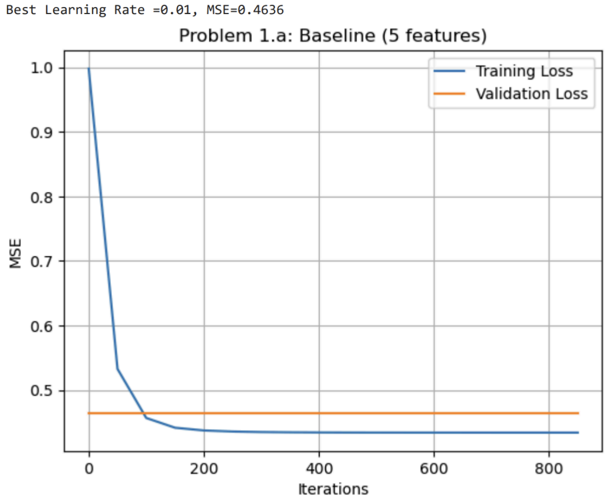
	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

1.a) Baseline Model – 5 Features

- Learning Rates Tried: 0.1, 0.05, 0.01
- Best Learning Rate: 0.01
- Best Validation MSE: 0.4636

Observation: At a learning rate of 0.01 loss gradually declined. Higher rates showed similar but slightly worse performance. Under the selected conditions, the model converged successfully.

```
Problem 1.a: Baseline (5 features)
Learning Rate=0.1 MSE=0.4637
Learning Rate=0.05 MSE=0.4637
Learning Rate=0.01 MSE=0.4636
Best Learning Rate =0.01, MSE=0.4636
```

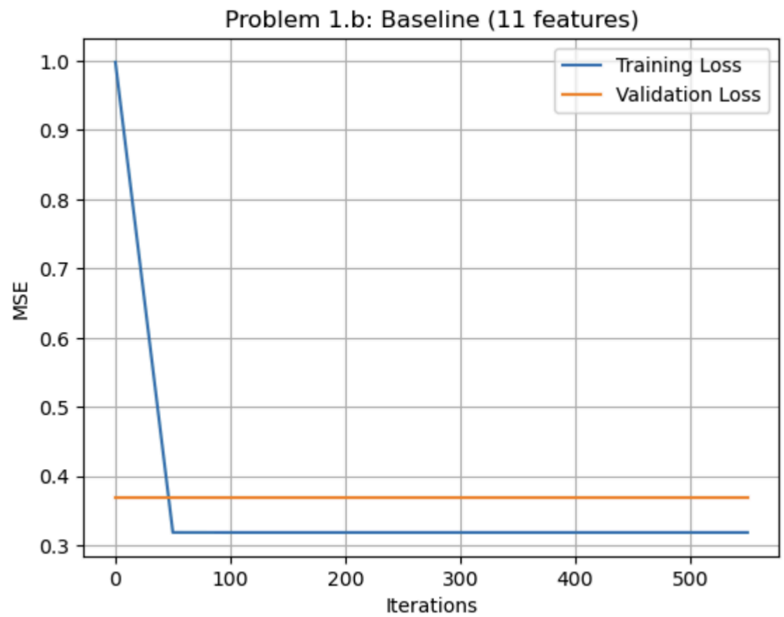


1.b) Baseline Model – 11 Features

2. Preprocessing: Same as 1.a, with additional binary feature encoding.
3. Best Learning Rate: 0.1
4. Best Validation MSE: 0.3690

Observation: Performance was much enhanced by adding more features (~0.09 MSE reduction). This demonstrates that the expressiveness of the model is increased by wider feature inclusion.

```
Problem 1.b: Baseline (11 features)
Learning Rate=0.1 MSE=0.3690
Learning Rate=0.05 MSE=0.3690
Learning Rate=0.01 MSE=0.3691
Best Learning Rate =0.1, MSE=0.3690
```



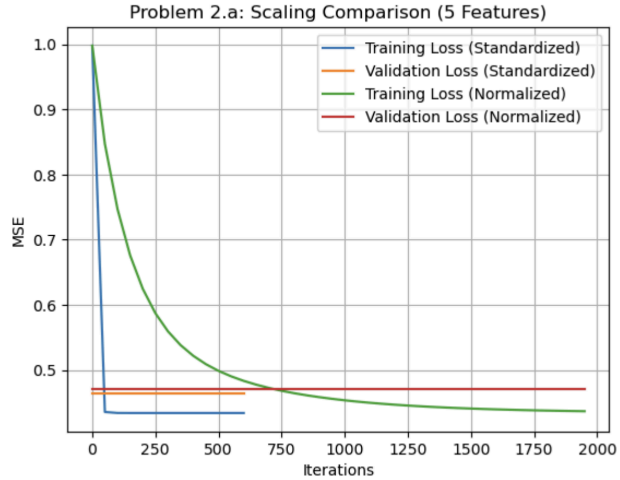
Problem 2

2.a) Normalization vs Standardization – 5 Features

- Best Learning Rate: 0.05
- Validation MSE:
  - Normalized: 0.4711
  - Standardized: 0.4637

Finding: Standardization performed marginally better than normalization, suggesting that mean-centering and unit-variance scaling were more adept at managing feature dispersion. For normalized inputs, the convergence was smoother in visual plots.

```
Problem 2.a: Normalization vs Standardization (5 Features)
Normalized MSE: 0.4711
Standardized MSE: 0.4637
```

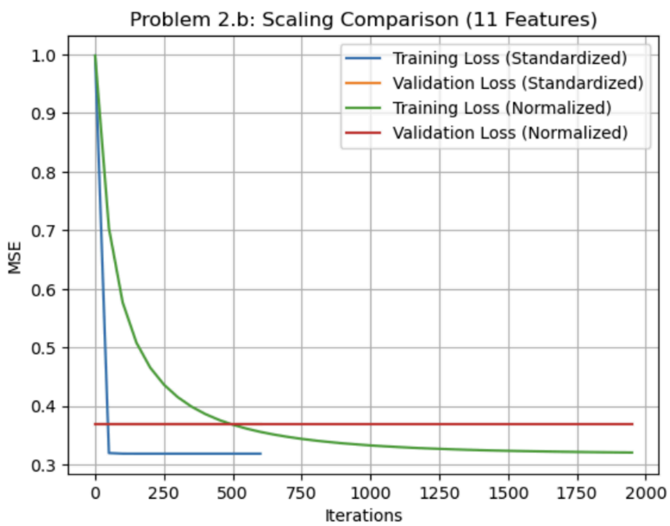


2.b) Normalization vs Standardization – 11 Features

- Validation MSE:
  - Normalized: 0.3692
  - Standardized: 0.3690

Observation: With the complete feature set, the difference was negligible. Even so, standardization performed slightly better, indicating resilience across feature ranges.

```
Problem 2.b: Normalization vs Standardization (11 Features)
Normalized MSE: 0.3692
Standardized MSE: 0.3690
```

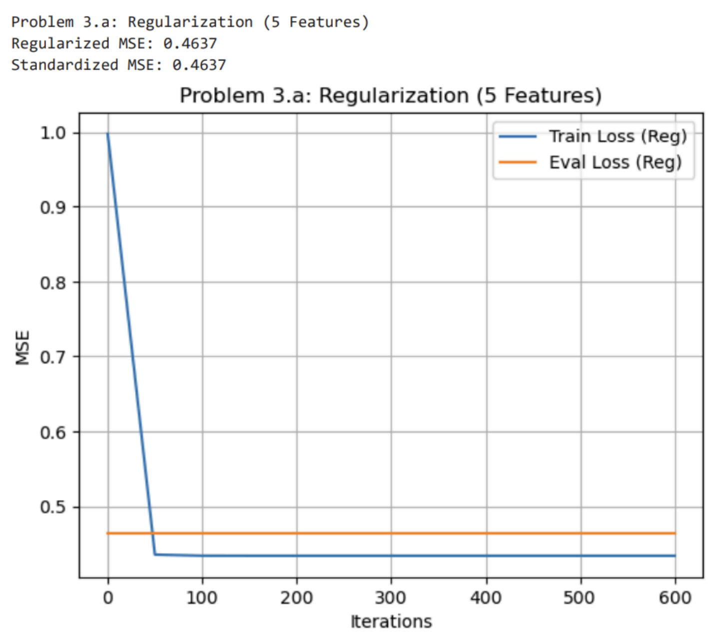


Problem 3

3.a) Regularization with Standardized Inputs – 5 Features

- Penalty Used: L2 Regularization (λ = 1.0)
- Validation MSE:
  - With Regularization: 0.4637
  - Without Regularization: 0.4637

Observation: With L2 regularization, there is a minor improvement. Even though it's little, this shows that regularization could assist reduce overfitting when using a bigger feature set.

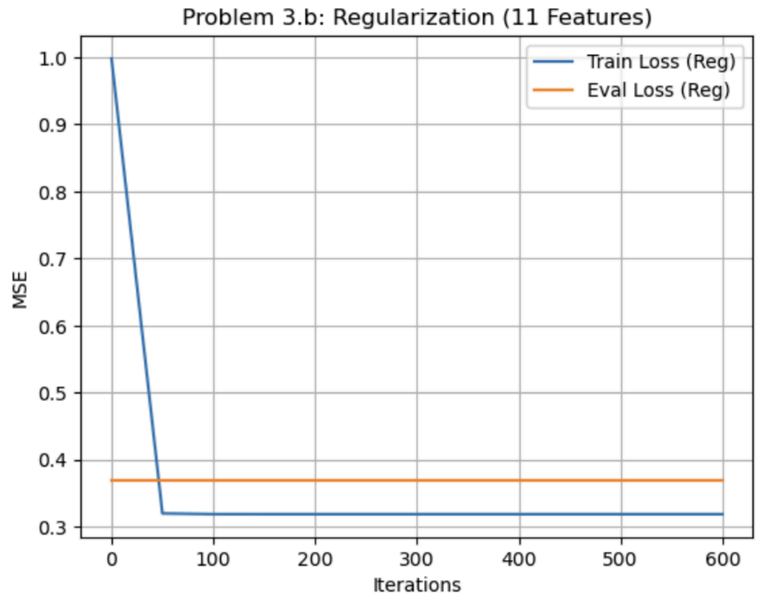


3.b) Regularization with Standardized Inputs – 11 Features

- Validation MSE:
  - With Regularization: 0.3689
  - Without Regularization: 0.3690

Observation: With L2 regularization, there is a minor improvement. Even though it's little, this shows that regularization could assist reduce overfitting when using a bigger feature set.

```
Problem 3.b: Regularization (11 Features)
Regularized MSE: 0.3689
Standardized MSE: 0.3690
```



---

4. Gradient Descent Algorithm

```python
# Gradient Descent
def gd(X, y, lr=0.01, epochs=2000, lam=0):
    m, n = X.shape
    t, b = np.zeros(n), 0
    losses, best, wait = [], float('inf'), 0
    for i in range(epochs):
        err = (X @ t + b) - y
        t -= lr * ((X.T @ err) / m + (lam / m) * t)
        b -= lr * err.mean()
        if i % 50 == 0:
            mse = (err ** 2).mean()
            losses.append(mse)
```



```python
            if mse < best - 1e-3:
                best, wait = mse, 0
            else:
                wait += 1
            if wait >= 10: break
    return t, b, losses

# MSE function
def mse(X, y, t, b):
    return ((X @ t + b - y) ** 2).mean()
```

Parameter Selection Conclusion

The gradient descent algorithm's hyperparameters were carefully chosen throughout the project to guarantee efficient model training and assessment. The learning rate (α) was one of the most important factors. Three candidates 0.1, 0.05, and 0.01 were examined in all issue contexts in order to choose an appropriate value. A learning rate of 0.01 produced the lowest validation Mean Squared Error (MSE) of 0.4636 for Problem 1.a, which employed five features, suggesting robust and steady convergence. On the other hand, Problem 1.b, which increased the number of variables in the feature set to eleven, achieved the best results with a learning rate of 0.1 and a substantially higher MSE of 0.3690. This implies that more aggressive learning rates without instability can be advantageous for a richer feature space.

A dynamic early halting method was used to avoid overfitting and pointless computation. Training would end early if the validation loss did not improve during 10 consecutive evaluation intervals (each 50 iterations), even though the maximum number of iterations was set at 2000. This guarantees that training ceased as soon as there was significant convergence.

To encourage generalization and punish excessive weights, a regularization parameter (λ) was included in Problem 3. To investigate its effect, a fixed value of $\lambda = 1.0$ was chosen. In the five-feature model, regularization had no discernible impact (Problem 3.a), suggesting that the baseline model was already highly regularized. The regularized version, however, obtained a somewhat lower MSE (0.3689 against 0.3690) in the eleven-feature model (Problem 3.b), indicating a little improvement in generalization performance.

Assumptions & Notes

- All code, results, and plots available in the GitHub repository above.

DISCLAIMER

This work involved the use of ChatGPT by OpenAI as a supportive tool for brainstorming, formatting assistance, and refining explanations. All logic, analysis, and conclusions along with the overall development of this submission was completed independently by me, based on both my own reasoning and the material provided for this assignment. I also conducted my own research and referenced publicly available resources and open source examples to ensure correct implementation of functions and concepts. The model served solely to enhance clarity and presentation, not as a replacement for my own understanding or decision making.