“Comparison of Built-in Linear Regression and Pseudoinverse Linear Regression with Optimal Weight Initialization using PLA in Python”

Prepared for

Dr. Chia-Ling Tsai
Professor, CSCI 325: Machine Learning, CUNY Queens College

By

Loyd Flores
Senior @ CUNY Queens College

Due
October 13, 2024

I. Abstract

This paper presents an exploration of linear classification techniques using the breast cancer dataset from the *sci-kit learn library*. The objective is to classify whether a tumor is benign or malignant based on various features related to cell nuclei characteristics. The analysis begins with exploratory data analysis (EDA), where histograms and a correlation matrix are utilized to examine feature distributions and relationships. Standardization of the dataset is applied to ensure uniformity in feature scales. Subsequently, the dataset is split into training and test sets for the classification task. The paper employs a linear classifier to predict the outcome, evaluates model performance, and provides insights into the factors influencing prediction accuracy. This study highlights the importance of preprocessing and feature analysis in improving classification results.

II. Preprocessing

The breast cancer dataset is straightforward to work with due to its manageable size of **569 rows and 30 features**, providing a sufficient amount of data for a model to learn from. An additional advantage is that the dataset contains no missing values, simplifying the preprocessing steps.

The main requirement is to standardize the feature values, as they vary greatly in magnitude. For instance, the *mean of mean_radius* is 17.99, while *mean_compactness* is 0.188, and *mean_perimeter* is 122.80. Without standardization, these numerical differences could cause certain features to disproportionately influence the model just because their values are larger.

Standardization transforms the data so that all features have a mean of 0 and a standard deviation of 1, ensuring that each feature contributes equally during model training.

Importantly, this process does not affect the actual significance of the features, as the model will determine their importance through the learned coefficients. Finally, it is essential to clearly separate the training features (X) from the target label (y) for proper model training.

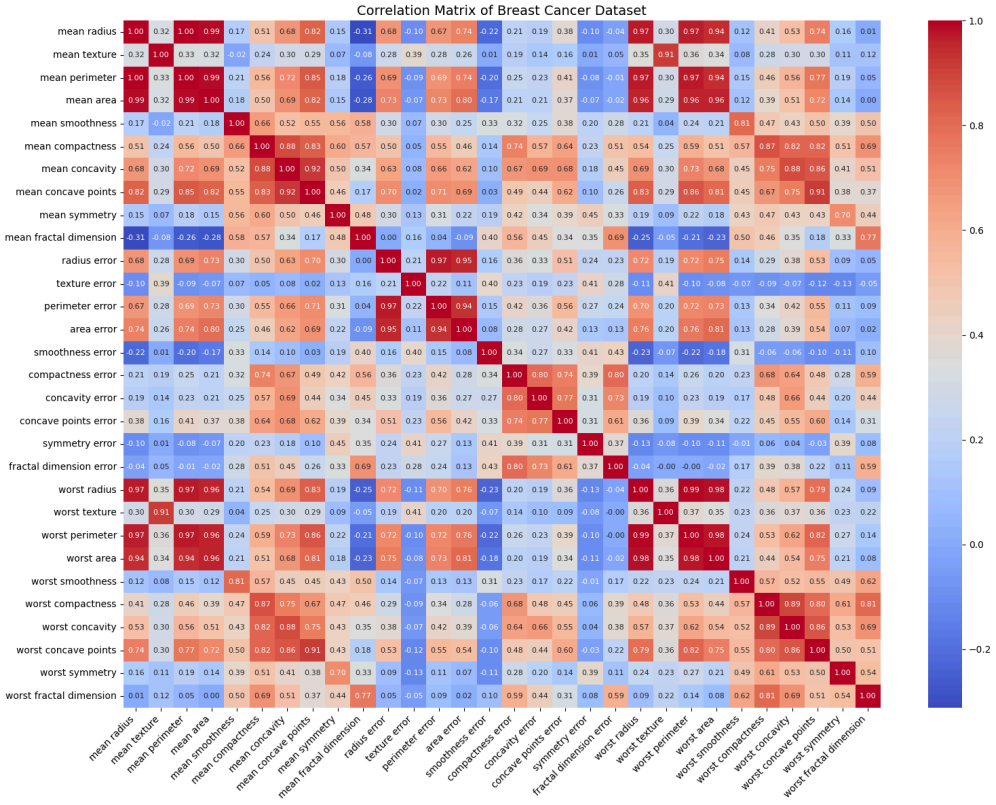

Figure 1: Correlation Matrix. Red means more correlated; Blue means less correlated

III. Methods

1. **Linear Regression; Using Pseudo Inverse to obtain the starting weight**
   Making predictions using Linear Regression under the hood is as simple as:

   $$Xw = y$$

   Where X represents the features, w represents the weights, and the product of X and w gives the predicted value y. In this method we are using the built in Linear Regression to find the most optimal weight w.

2. **Pocket Algorithm ;improve the weight vector for optimization**
   The Pocket Learning Algorithm is an extension of the Perceptron Learning Algorithm designed to handle cases where the data is not linearly separable. In simple terms, Perceptron is not optimized if a line cannot be drawn to separate the data in half and Pocket is capable. The algorithm works by continuously updating the weight vector based on misclassified points until all points are correctly classified. The pocket algorithm maintains the best solution found so far, and even if the algorithm does not converge. The “best” solution is stored in a “pocket”, hence the name.

IV. Experiment

To identify the performance of the Pseudo-Inverse Implementation of Linear Regression this study compares it to the built in implementation in Python. The two models will be tested on the same X or features, and will predict on the same y or labels. This gives an equivalent comparison. Train-test-split is also implemented to properly measure the accuracy of the model on Eout or unseen data.

V. Results

Mean Squared Error (MSE), measures how far the predicted values are from the actual values by calculating the average of the squared distances. Lower MSE indicates better predictions.

Linear Regression:
MSE (Ein): 0.206
MSE (Eout): 0.256

Pocket PLA:
MSE (Ein): 0.389
MSE (Eout): 0.211

Accuracy, The percentage of correct predictions out of all predictions made.

Linear Regression:
Accuracy (Ein): 0.962
Accuracy (Eout): 0.956

Pocket PLA:
Accuracy (Ein): 0.903
Accuracy (Eout): 0.947

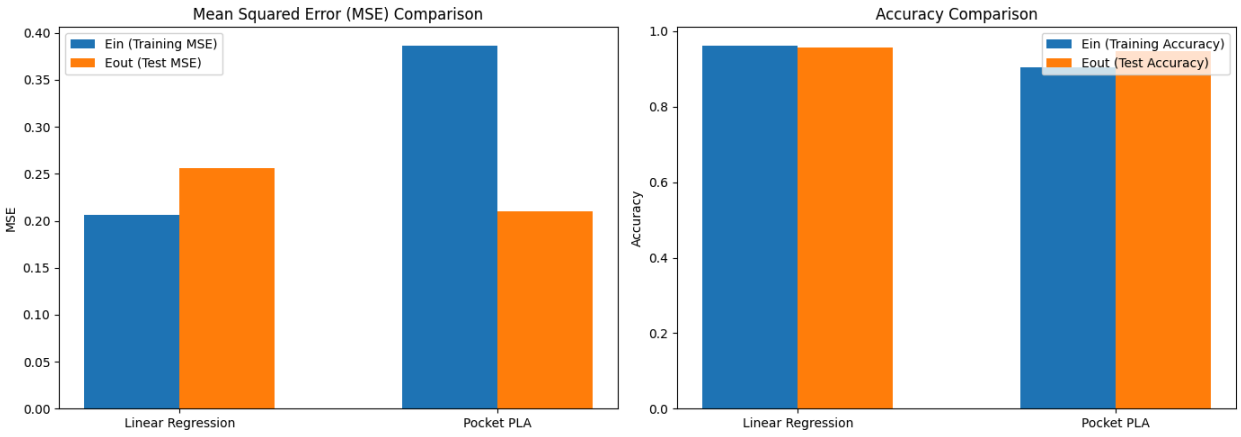

Figure 2: Result of the experiment

VI. Conclusion

It seems that the Python implemented Linear regression out performs the manual implementation of Linear Regression using Pseudo-Inverse.