Multiple Linear Regression Model

Question 1 – Multiple Linear Regression Models

In the practical lab folder you will find a data file called boston.csv. This is a well-known regression dataset.

The following is a description of each of the features:

1. CRIM per capita crime rate by town
2. ZN proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS proportion of non-retail business acres per town
4. CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX nitric oxides concentration (parts per 10 million)
6. RM average number of rooms per dwelling
7. AGE proportion of owner-occupied units built prior to 1940
8. DIS weighted distances to five Boston employment centres
9. RAD index of accessibility to radial highways
10. TAX full-value property-tax rate per $10,000
11. PTRATIO pupil-teacher ratio by town
12. B 1000(Bk - 0.63)^2 where Bk is the proportion of black population
13. LSTAT % lower status of the population
14. MEDV Median value of owner-occupied homes in $1000's

Instructions:

1) The objective of this question is to build and assess the performance of a MLR model for the Boston house prices dataset. You will find a template for your code in the Blackboard folder. In the code you will notice that rather than having a single coefficient value (as was the case in linear regression), we design to code so that it can scale for multiple coefficients. One for each feature we read in from the dataset.

2) You should complete the hypothesis method. It takes in the following arguments:

    a. A 2D NumPy array containing all the feature data
    b. A array containing each of the coefficients for each feature in our dataset
    c. The bias (a single numerical value)

The objective of the function is to calculate the predicted output for each training example. Remember your hypothesis for an MLR is:

$$h(x) = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \ldots .. + \lambda_m x_m + b$$

3) Complete the gradient calculations in the gradient_descent function. The function calculates the gradient for the bias and the updated bias value. You should also update each of the coefficients using the same gradient descent update rule. Calculate the gradient and use this to update the coefficient. Once complete you should now be able to run your MLR algorithm.

When you train the model on the Boston dataset you should be obtaining an R2 of approximately 0.73.

Question 2 – Performance on Regression Data with Test Set

I have provided another dataset in a .zip file called Dataset.zip that contains both a train and test file.

Run your MLR (developed in Q1) on the training set to build your model.

Adjust your existing code to assess the accuracy of the model on the test set. You should be obtaining a very high R2 >0.98.

This is the same dataset we used previously for the kNN exercise last year. Notice that the MLR provides better accuracy. Unlike the kNN, which treated all features equally, the MLR attaches a weight to each feature.