514 Programming 1 Report

Name: Zhuoyun Chen

SIS ID: 499214

1. Introduction

1.1. Description of the problem

There are 1030 instances in the dataset of Concrete Compressive Strength with 9 features. We use the last feature as the response variable, and the other 8 features as variables. We use 900 instances as training data and 130 instances as testing data.

In this project, we try to find the relationship between the response variable and other variables. We use uni-variate regression and multi-variate regression to find the target function and computing loss values.

After seeing the data in the dataset, I use normalization to make the data more regular.

1.2. Details of Algorithm

As of Uni-variate linear regression, the function formula is h(x) = mx + b.

As of Multi-variate linear regression, the function formula is

$h(x_i) = m_1x_1 + m_2x_2 + m_3x_3 + m_4x_4 + m_5x_5 + m_6x_6 + m_7x_7 + m_8x_8 + b$

The loss function is MSE, and the function is

$$MSE = \sum_{i=1}^{n}(y_i - h(x_i))^2$$

The function of gradient-descent is:

$$m_{new} = m_{old} - \frac{\alpha}{n}\sum_{i=1}^{n} -2x_i(y_i - (m_{old}x_i + b_{old}))$$

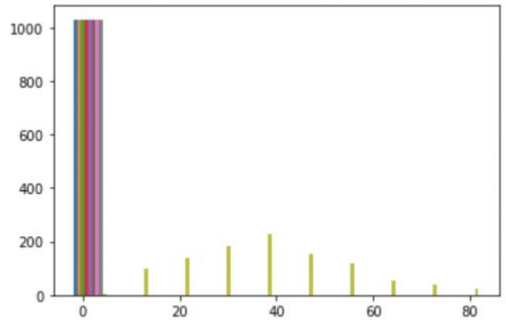
1.3. Pseudo-code of algorithm

```
def h(x,theta):
    return np.matmul(x,theta)

def cost_function(x, y, theta):
    return ((h(x, theta)-y).T@(h(x, theta)-y))/y.shape[0]

def gradient_descent(x, y, theta, learning_rate=0.1, num_epochs=2000):
   m = x.shape[0]
   J_all = []
  for _ in range(num_epochs):
       h_x = h(x, theta)
       cost_ = (2/m)*(x.T@(h_x - y))
```

```
        theta = theta - (learning_rate)*cost_
        J_all.append(cost_function(x, y, theta))

    return theta, J_all
```

R-square value = 1-J/np.var(y)

1.4. The histogram of normalized data



2. **Result(All these result are based on the normalized data)**

2.1. Variance explained for the response variable in the training dataset

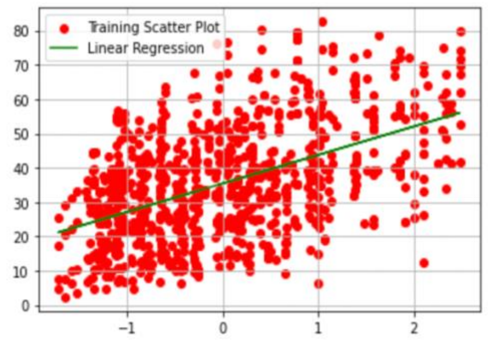
Var_explained_1 = 0.253
Var_explained_2 = 0.014
Var_explained_3 = 0.009
Var_explained_4 = 0.069
Var_explained_5 = 0.136
Var_explained_6 = 0.026
Var_explained_7 = 0.029
Var_explained_8 = 0.105
Var_explained_9(multi) = 0.585

2.2. Variance explained for the response variable in the test dataset

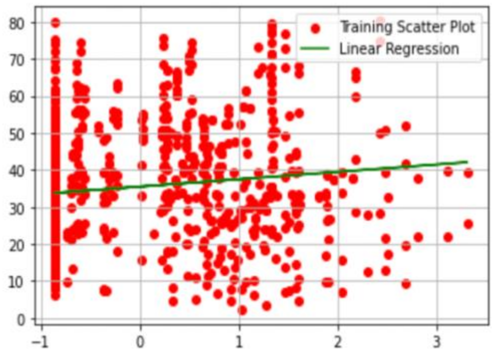
Var_explained_test_1 = 0.206
Var_explained_test_2 = 0.037
Var_explained_test_3 = 0.019
Var_explained_test_4 = 0.154
Var_explained_test_5 = 0.101
Var_explained_test_6 = 0.028
Var_explained_test_7 = 0.018
Var_explained_test_8 = 0.124
Var_explained_test_9(multi) = 0.639

2.3. Scatterplots

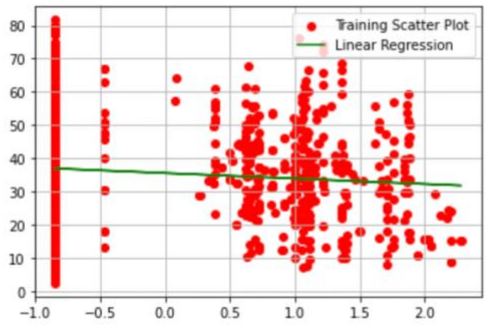
Response variable v.s. Cement



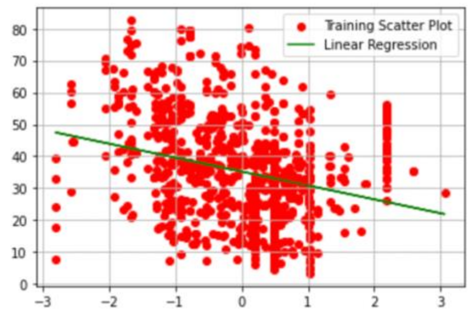
Response variable v.s. Blast Furnace Slag



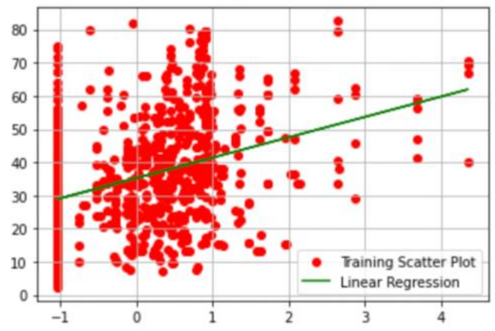
Response variable v.s. Fly Ash



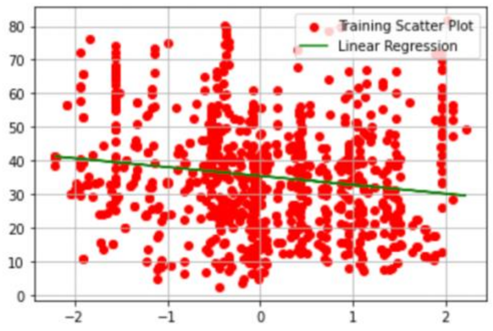
Response variable v.s. Water



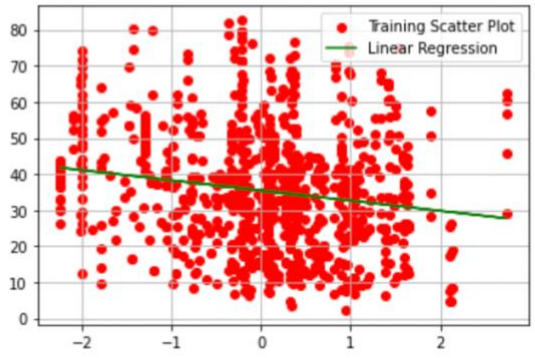
Response variable v.s. Superplasticizer



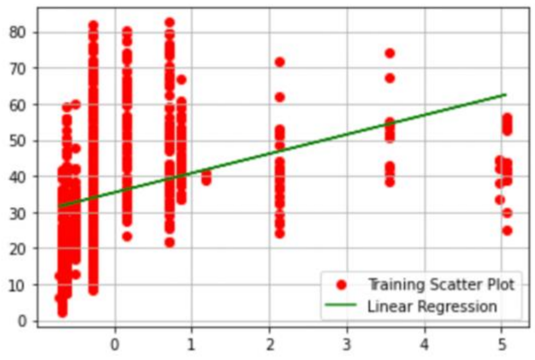
Response variable v.s. Coarse Aggregate



Response variable v.s. Fine Aggregate



Response variable v.s. Age



3. Discuss

3.1. Describe models and compare models

From the variance-explained, we can find that multi-variate linear regression has a much higher variance-explained value than other models. I think more features give the model itself more information, hence, making the model with higher R-squared value.

Considering uni-variate model, it is easy to find that some models' variance-explained value are really low, like model 3 (Fly Ash), model 6 (Coarse Aggregate) and model 7 (Fine Aggregate).

Also, we can find that some models perform good in training but worse in testing, like model 7 (Fine Aggregate). Meanwhile, some models perform better in testing, like the multi-variate model.

3.2. Describe how the different models compared to train/test. Did different models take longer to train? Did you have to use different hyperparameter values for different models?

When we look at these models, we can find that for each model, the difference of variance-explained value exists in train/test. And like I mentioned before, some models are better when using training dataset, but some are better when using testing data. And some are similar.

I used time in python to calculate the time model used for training. It turns out that of each model, the training model is about same. The difference between the shortest and the longest is less than 0.01. When iterations increases, the training time will increase. If I increase the iteration times from 20 to 2,000. The time will increase from 0.08 to 0.46.

I did not use different hyperparameters for different models. Because when you change the learning rate or iteration times, it does not that matter, it will finally converge.

3.3. Conclusion

Cement, Water and Superplasticizer predict concrete compressive strength. To make the hardest possible concrete, we should increase Cement and Superplasticizer, decrease Water.

3.4. Comparison between normalization and original

We can take a look at the original value, some features are not in the same level. Some features have average of thousand, but some are tens. Hence, when I run the model with the original data, the loss value of some models is up to 1,000. And the variance-explained value is negative, which is unusual. With normalization, we can find that the loss value are much smaller, the variance-explained looks right.