Programming Assignment 1 Report

- **Student Name: Hongyi Xu**
- **Student Number: 499173**

1. Introduction

Description/formulation of the problem

For this report, my goal is to design and implement a gradient decent algorithm for regression to predict the Concrete compressive strength with different features like Cement, Blast Furnace Slag, Coarse Aggregate. Then complete several different ways to solve the same dataset and compare their results. This program use the Concrete Compressive Strength dataset in the UCI repository. Through the results of this program results, we can get some intuitions or some conclusions to predict the Concrete compressive strength in the future. This may be very useful for some companies. And we can also use these conclusions to solve other predictions not only the Concrete compressive strength prediction.

The details of my algorithm

For all algorithm I set a stopping criterion that the program will stop when the MSE is lower than 40 because an MSE of 40 is already a relatively good result for this assignment, and with my model, it will take a very large number of iterations to further drop more MSE after 60. The learning rate is different according to different algorithms. The learning rates are listed below:

Models	learning rate (before standardized)	learning rate (after standardized)	initial m, b (before standardized)	initial m, b (after standardized)
Cement (uni-variate)	0.0000001	0.0000001	m=0, b=0	m=0, b=0
Blast Furnace Slag (uni-variate)	0.00000007	0.0000001	m=0, b=37	m0, b=37
Fly Ash (uni-variate)	0.00000001	0.0000001	m=0, b=36	m=0, b=36
Water (uni-variate)	0.00000001	0.0000001	m=0, b=37	m=0, b=67
Superplasticizer (uni-variate)	0.00000001	0.00000001	m=0, b=36	m=0, b=36
Coarse Aggregate (uni-variate)	0.00000001	0.00000001	m=0, b=36	m=0, b=55
Fine Aggregate (uni-variate)	0.00000001	0.00000001	m=0, b=46	m=0, b=55
Age (uni-variate)	0.00000001	0.00000001	m=0, b=36	m=0, b=36
all feature (multi-variate)	0.0000001	0.1	m=[0]*9	m=[0]*9
quadratic model	0.0000000000001	0.01	m=[0]*45	m=[0]*45

Description of bonus quadratic model

The quadratic model contains 36 quadratic terms and 8 linear terms, for a total of 45 (44 + the constant term) parameters. For the 36 quadratic terms, use combination rule to those 8 features (Xi^2 only counts once).

Pseudo-code of those algorithms

Uni-variate linear regression:

```
This program calculates unni-variance explained for the response variable

function gradient_desc(Argument one, Argument two, Argument three, Argument four,
Argument five, Argument six){
    update the m and b by Gradient Descent optimization algorithm

    Run this function repeatly until epochs reach the defined value

    return changed_m, changed_b
}
```

```
function compute_variance (Argument one, Argument two, Argument three){
    compute 1-MSE/Variance(observed)
}

{
In the main function
    Import dataset from a xls file

    Slice the first 900 rows as the training set
    Slice the remaining 130 rows as the testing set

    Chose one feature from the dataset
    Set the chosed feature, initial m, initial b, learning rate, epochs

    Send the training data, the initial m, initial b, learning rate, epochs into
gradient_desc function
    Send the testing data, the trained m, the trained b into compute_variance function
and print the varience result

    Plots of trained uni-variate models on top of scatterplots of the
training data used (plot the data using the x-axis for the predictor variable
and the y-axis for the response variable)
}
```

Multi-variate linear regression:

```
This program calculates variance explained for the multi-variable

function gradient_desc(Argument one, Argument two, Argument three, Argument four){
    update every m by Gradient Descent optimization algorithm

    Run this function repeatly until epochs reach the defined value

    return final_m, final_varience
}

function compute_variance (Argument one, Argument two){
    compute 1-MSE/Variance(observed)
}

{
In the main function
    Import dataset from a xls file

    Slice the first 900 rows as the training set
    Slice the remaining 130 rows as the testing set

    Set the initial m, learning rate, epochs
    Send the training data, the initial m, learning rate, epochs into gradient_desc
function and print the varicance result
    Send the testing data, the trained m into gradient_desc function and print the
varicance result
}
```

Multi-variate polynomial regression:

```
This program calculates variance explained for multi-variate quadratic regression
model

function gradient_desc(Argument one, Argument two, Argument three, Argument four){
    update every m by Multi-variate Quadratic Regression Model and Gradient Descent
optimization algorithm

    Run this function repeatly until epochs reach the defined value

    return final_m, final_varience
}

function compute_variance (Argument one, Argument two){
    compute 1-MSE/Variance(observed)
}

{
In the main function
    Import dataset from a xls file

    Slice the first 900 rows as the training set
    Slice the remaining 130 rows as the testing set

    Set the initial m, learning rate, epochs
    Send the training data, the initial m, learning rate, epochs into gradient_desc
function and print the varicance result
    Send the testing data, the trained m into gradient_desc function and print the
varicance result
}
```

Data pre-processing

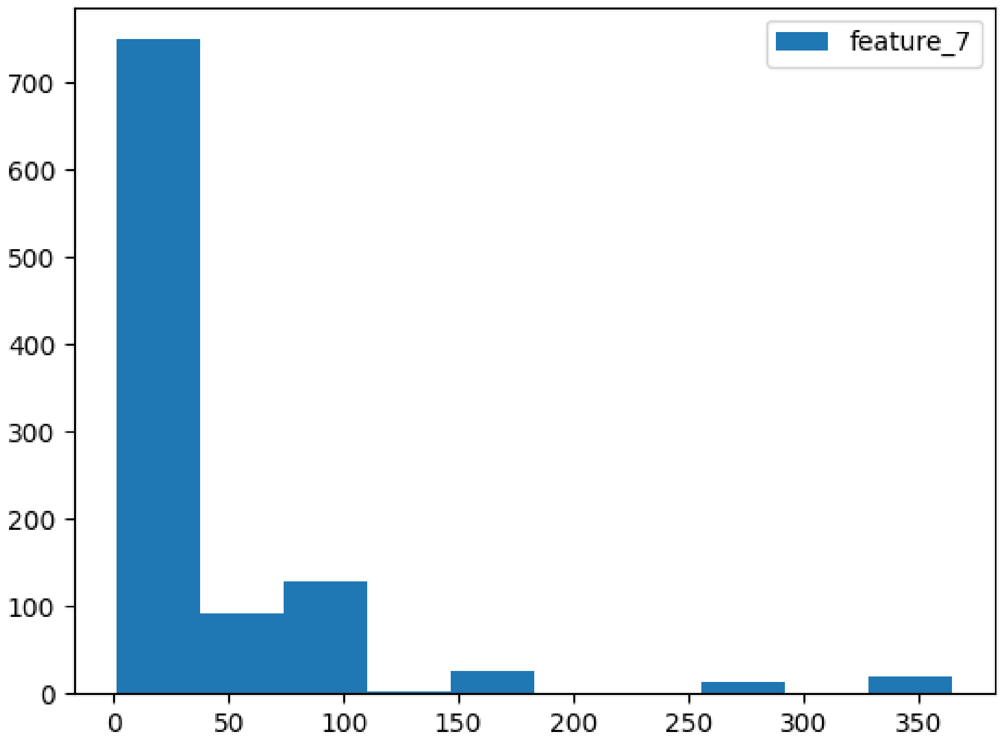
I have standardized data before training. It subtracts the mean and divides by the standard deviation of the dataset along a given axis so that these data can be fitted in a smaller range. The formula for pre-processing data is as follow:

$$\frac{X - mean}{std}$$

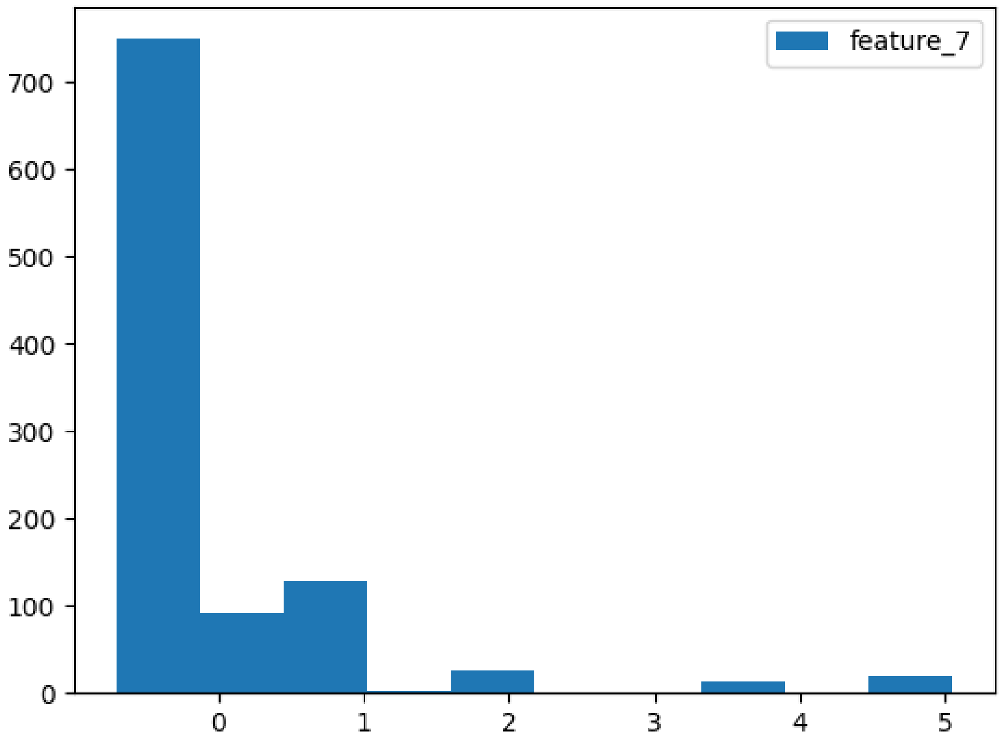
For this work, I have plotted the distributions of data before and after standardization as histograms.

I choose the feature 8 as one of example of uni-variate model data:

feature 8 (shows feature_7 since the array index) data before standardization:

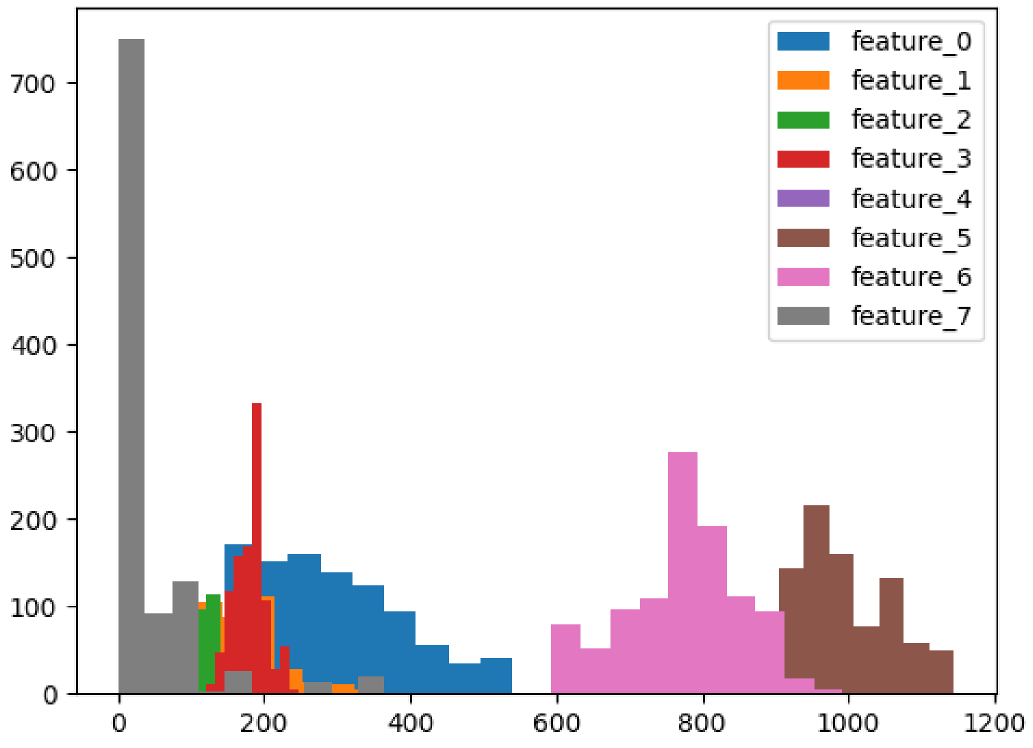


feature 8 (shows feature_7 since the array index) data after standardization:

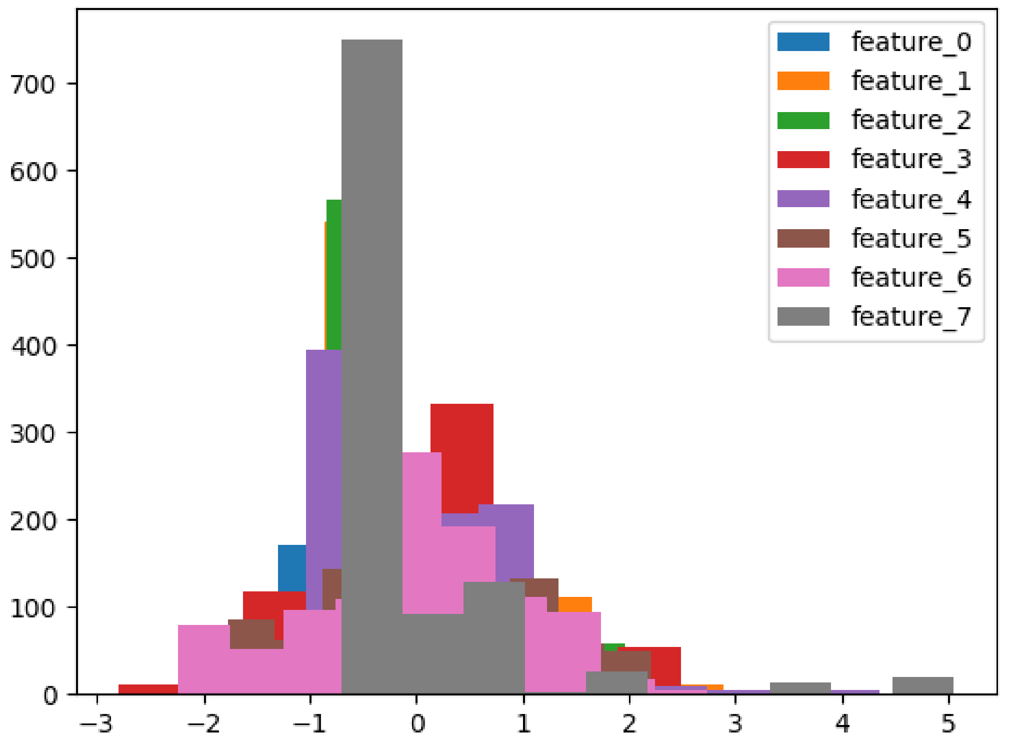


Then I plot the distributions of multi-variate model data before and after standardization as histograms:

multiple features data before standardization:



multiple features data after standardization:



From these pictures, we can see that the distribution of the original data (both for the uni-variate model data and the multiple features data) will not be changed if I do the standardization. In other words, I have not lost information in this standardization process. So the standardization process is successful and effective.

2. Results

The formula for calculating the variance in this program:

$$1 - \frac{MSE}{Variance(Observed)}$$

Variance explained of models on the training dataset

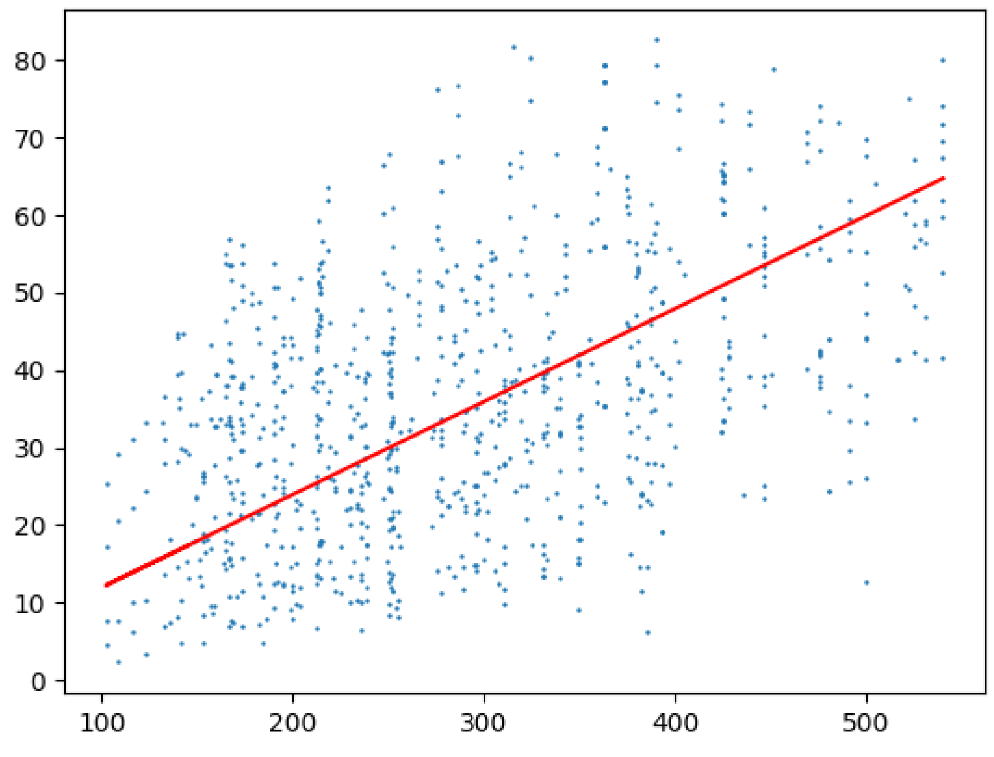
Training Models	Variance (before standardized)	Variance (after standardized)	epochs
feature 1 (uni-variate)	0.15717638859690852	0.157147030739604	10000
feature 2 (uni-variate)	0.005069651370609329	0.005000442221225088	10000
feature 3 (uni-variate)	-0.000037200113339963536	0.0017575341018032375	10000
feature 4 (uni-variate)	0.003770100329733017	0.08063335421578433	10000
feature 5 (uni-variate)	0.0008832630743116354	0.0024044626796211332	10000
feature 6 (uni-variate)	-0.0004298422730273366	0.025808263057028435	10000
feature 7 (uni-variate)	0.01779335500287349	0.02794919201598922	10000
feature 8 (uni-variate)	0.07484930564501457	0.07484930564501457	10000
all feature (multi-variate)	0.5991850718543389	0.6127199839081706	10000
quadratic model	0.6416685138134328	0.8105326492721665	10000

Variance explained of models on the testing data points

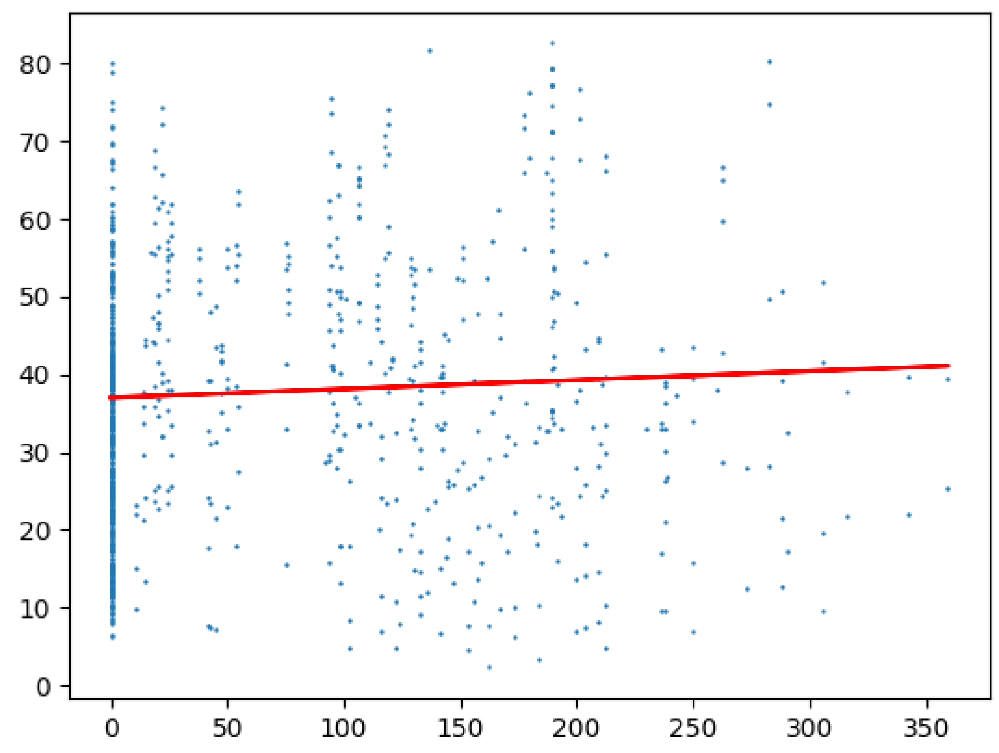
Testing Models	Variance (before standardized)	Variance (after standardized)	epochs
feature 1 (uni-variate)	0.2281118438813876	0.22802391475841455	10000
feature 2 (uni-variate)	-0.23121998638223817	-0.22472820628369083	10000
feature 3 (uni-variate)	-0.0782104219209423	-0.07624649372547676	10000
feature 4 (uni-variate)	-0.0966714946687095	-0.0628655349228382	10000
feature 5 (uni-variate)	-0.18226492417774143	-0.11881194370071402	10000
feature 6 (uni-variate)	-0.11980533484645117	-0.17409782499812732	10000
feature 7 (uni-variate)	-0.12375618951504941	-0.14122913282635574	10000
feature 8 (uni-variate)	-0.1931872248827482	-0.1931872248827482	10000
all feature (multi-variate)	0.567844566043439	0.5789472286349375	10000
quadratic model	0.6027999258131181	0.6901423500151679	10000

Plots of trained uni-variate models on top of scatterplots of the training data used

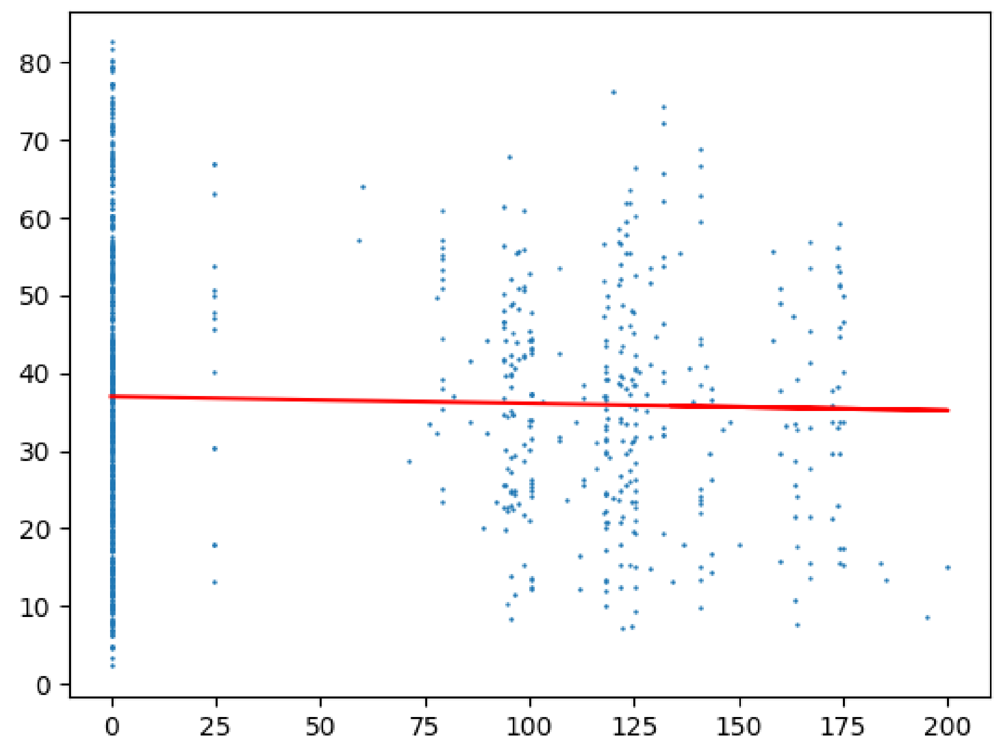
feature 1 model (uni-variate) plot



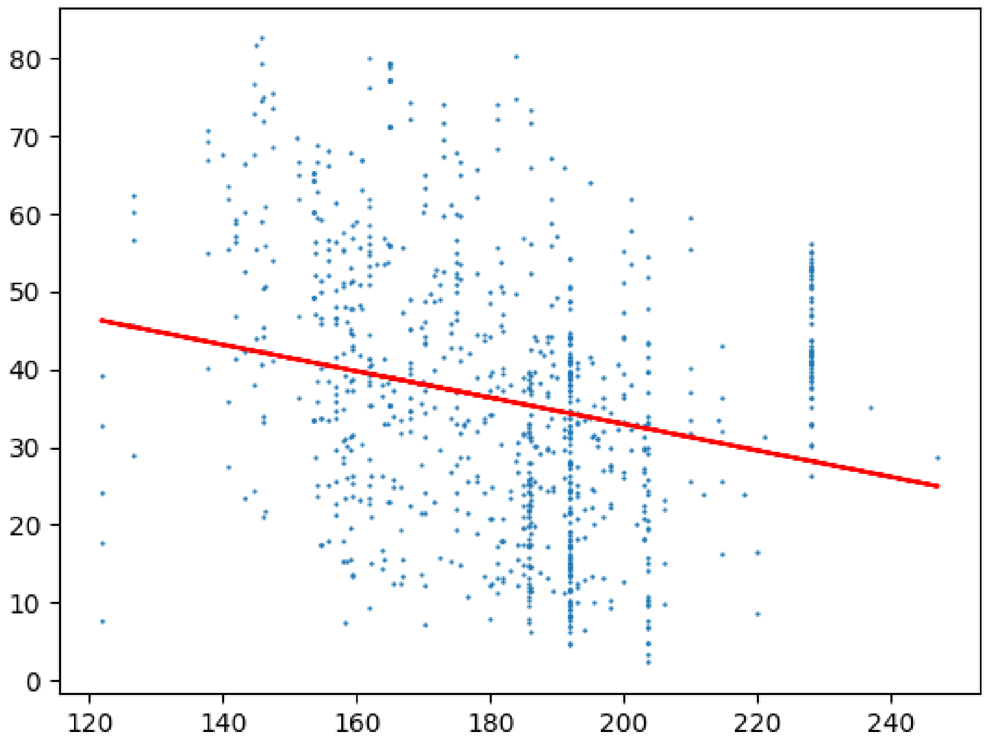
feature 2 model (uni-variate) plot



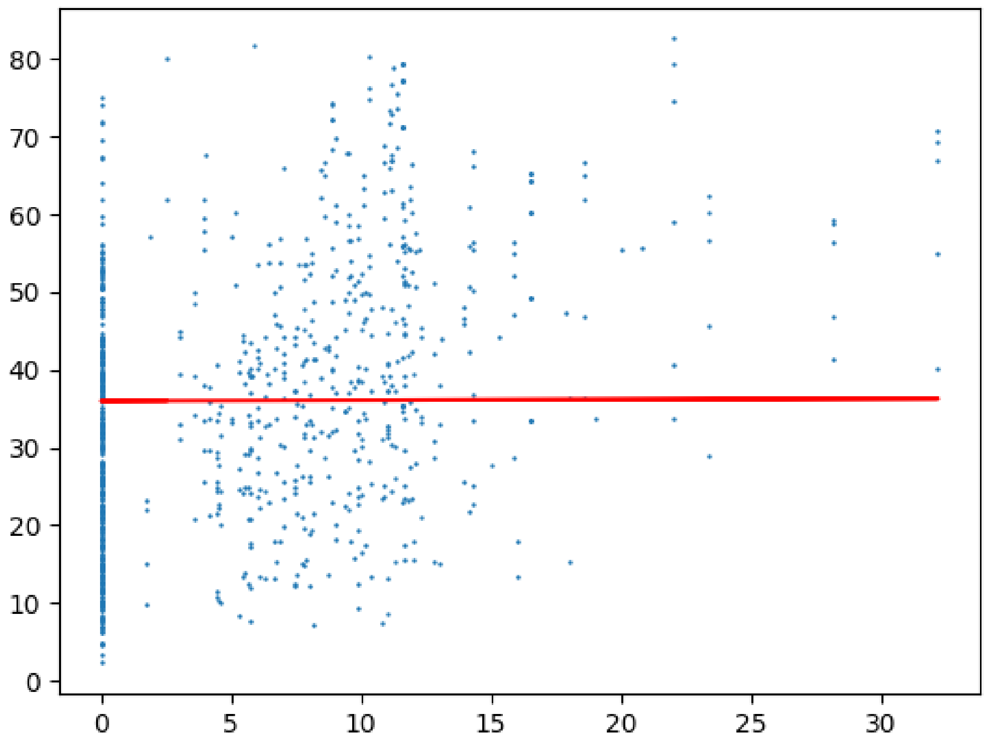
feature 3 model (uni-variate) plot



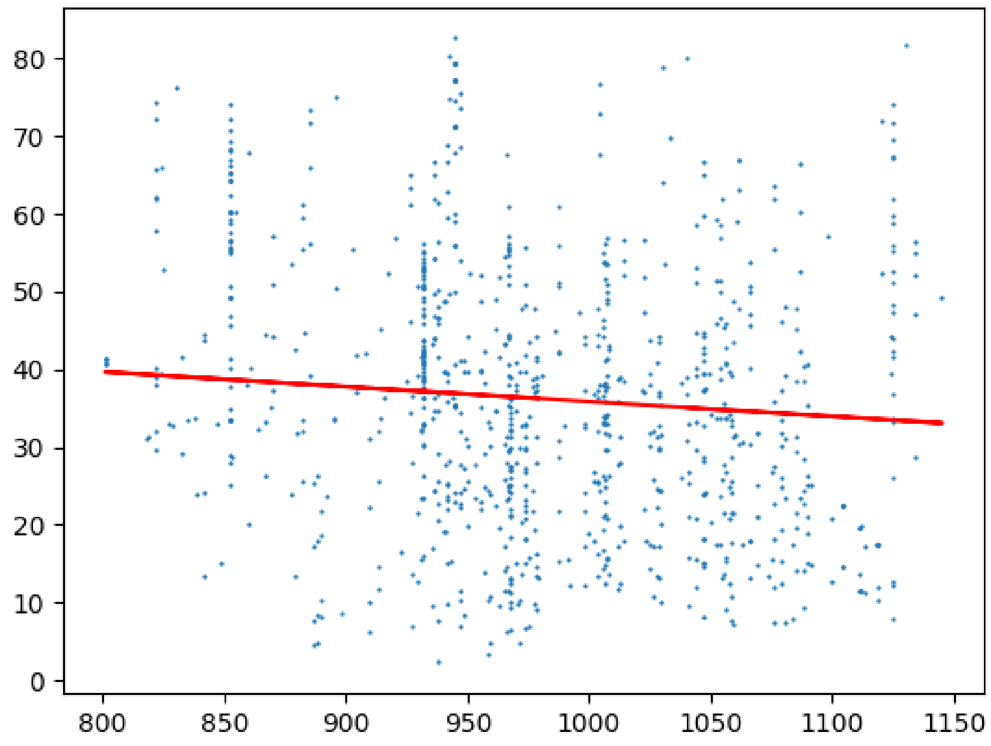
feature 4 model (uni-variate) plot



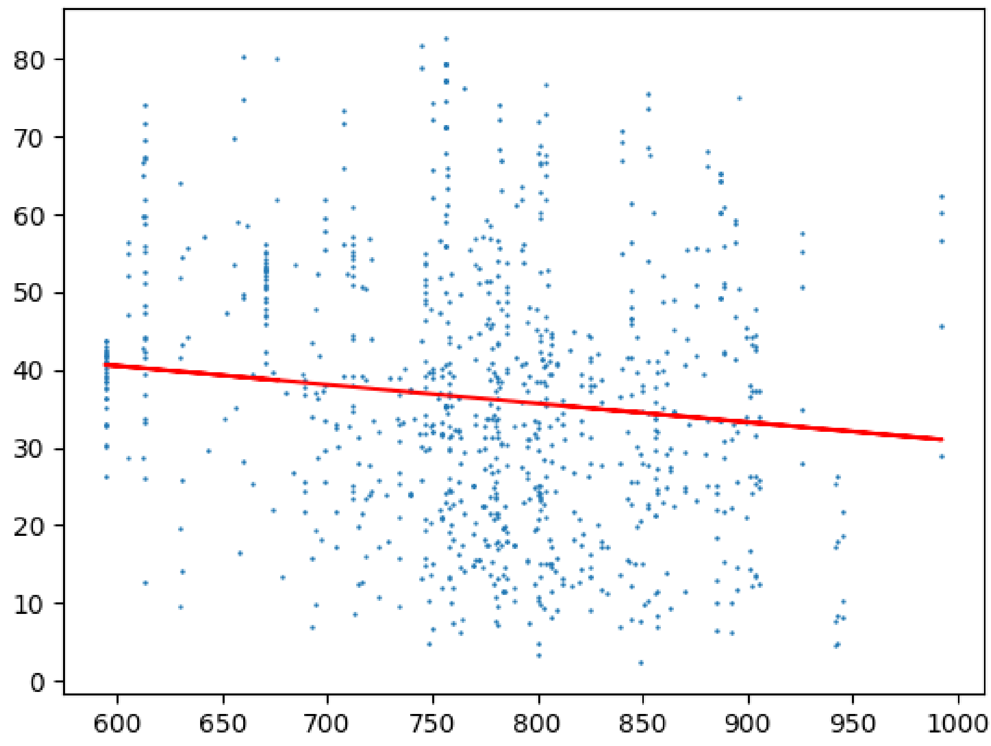
feature 5 model (uni-variate) plot



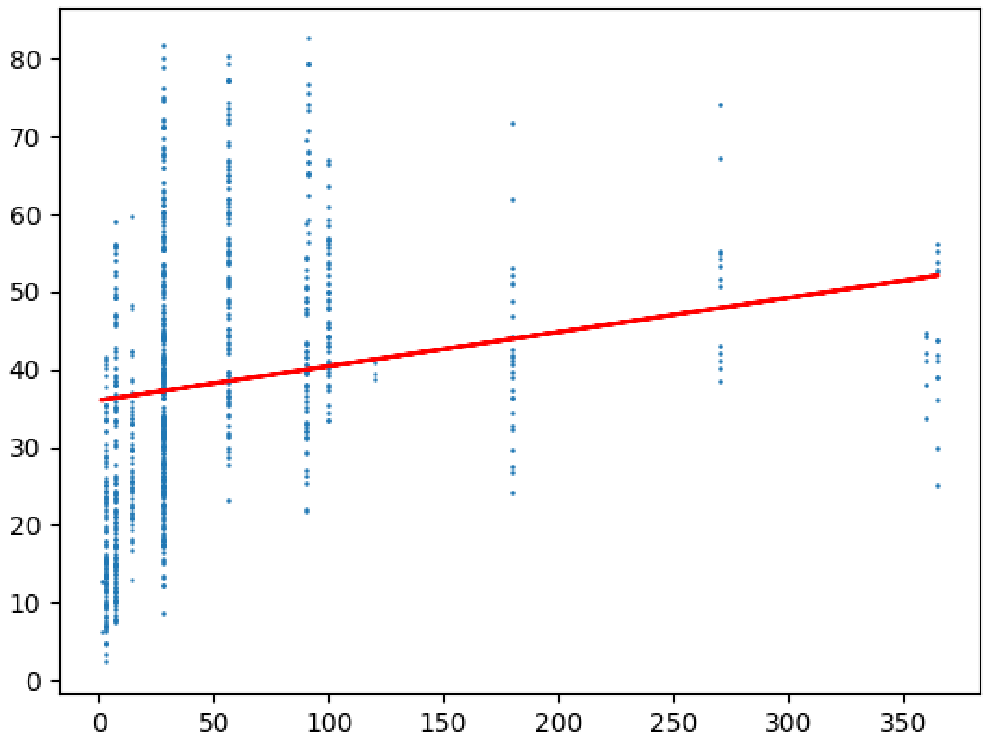
feature 6 model (uni-variate) plot



feature 7 model (uni-variate) plot



feature 8 model (uni-variate) plot



3. Discussion

Some comparations

For the performance on the training data, from the result, we can find that the quadratic model get the best variance than other model. Multi-variate model also performs better than uni-variate models. The variances of the uni-variate models show different between different features. Some features show better than other features. For example, The feature 1 (Cement (component 1)(kg in a m^3 mixture)) shows better variance than all the other 7 features model. Feature 3 (Fly Ash (component 3)(kg in a m^3 mixture)) and feature 6 (Coarse Aggregate (component 6)(kg in a m^3 mixture)) do not have good performance in comparation with other features. For data with pre-processing, most of the modals' variance show better than the original data and the overall performance between different models do not change (quadratic model > Multi-variate model > uni-variate models).

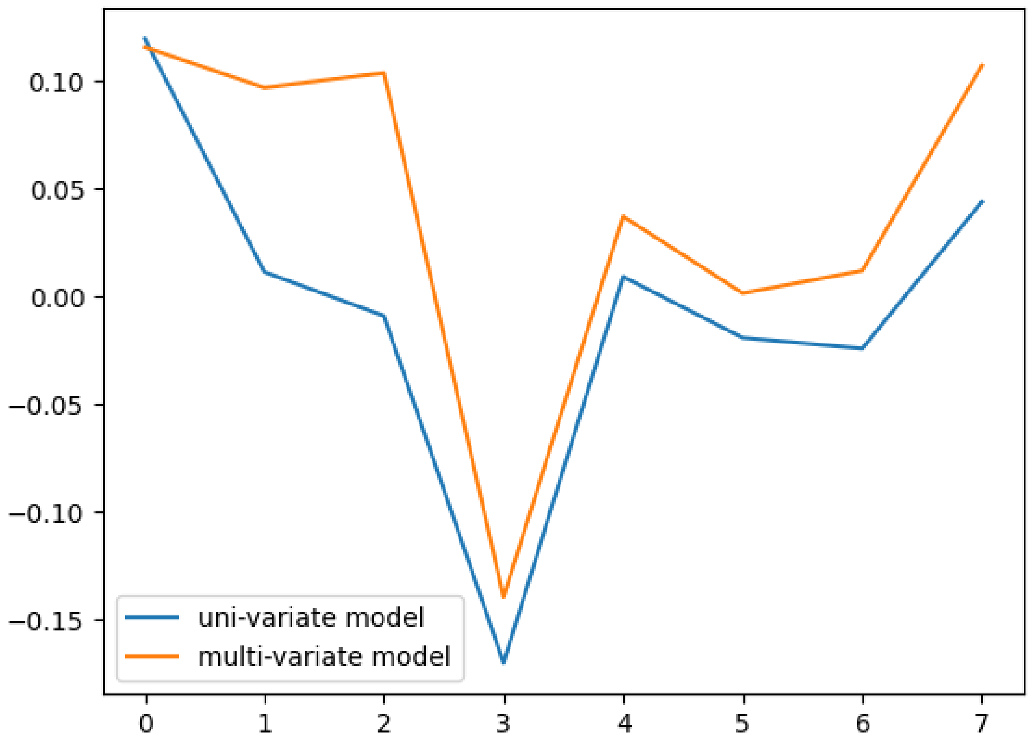
for the performance on the testing data, from the result, we can find that although every variance are less than every training data's variance, they still show consistent performance with their training data performance. That means same models that performed well on the training data still did well on the testing data. Standardization can also improve the performance and increase the variance for the testing data.

The coefficients of the uni-variate models and the multi-variate model(s)

The coefficients of the uni-variate models and the multi-variate models is as below:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
uni-variate	0.1198	0.0125	-0.0034	-0.0067	0.0081	0.0001	-0.0127	0.0440
multi-variate	0.1159	0.0971	0.1039	-0.1396	0.0372	0.0016	0.0120	0.1073

From this table we can see that although there are some features that show similar coefficients between uni-variate model and multi-variate model like feature 1 (Cement) and feature 5 (Superplasticizer), the uni-variate model's coefficients fail to predict the exactly coefficients' value of the multi-variate model. Most of their coefficients show quite different. But most of their coefficients' value show the same trend like the following picture shows:



Conclusion

From the performance of those models, we can find that for the uni-variate model, different features have different levels to predict concrete compressive strength. This may be caused by the numerical variety. For example, The feature 1 (Cement), which shows better variance than all the other 7 features model, shows better correspondence changes with different concrete compressive strength without many duplicate value like other features show. Some features show bad performance since there are many repeated values for different Concrete compressive strength like the feature 3 (Fly Ash) that has too many 0 value and the feature 8 (Age) that has too many duplicate values. One ideal way to get the hardest possible concrete is to use the good performance features like Cement and avoid using the bad performance features.