Shane Canfield
Assignment 1
October 12, 2022

Content

Introduction

Concrete is wildly used in many fields such as civil engineering. During construction, the concrete used for the project might require specific properties in order to perform its function. It might be helpful to understand the relationship between the properties of concrete and its overall concrete compressive strength. We will be using a combination of univariate and multivariate linear regression using gradient descent to determine which factors contribute most to our response variable. The data for this endeavor will be using The Concrete Compressive Strength dataset published in 2007 from the Chung-Hua University. The data has 1030 rows, and 9 different attributes. Number 9, Concrete compressive strength (MPa, megapascals), being the datasets response variable. The attributes (kg in a m^3 mixture) are as follows:

1. Cement
2. Blast Furnace Slag
3. Fly Ash
4. Water
5. Superplasticizer
6. Coarse Aggregate
7. Fine Aggregate
8. Age (day)
9. Concrete compressive strength (MPa, megapascals)

Of these features, a random sample of 900 instances will be used for training, and the remaining 130 will be used for testing. Both univariate and multivariate gradient descent

models are explored in this report, standardized and non-standardized. The loss function of our gradient descent algorithm is Mean Squared Error (MSE).

For the univariate model implementations, a max iteration limit of 500 was chosen to prevent overfitting as much as possible. The difference between 400 and 500 iterations proved to be significant enough to allow success with two different univariate models able to explain at least 10% variance. Max iterations for multivariate gradient descent were increased to 1000 which proved to help the algorithm achieve the near-best performance. To investigate timing of one of the gradient descent methods, a threshold of 0.001 was added with a max iteration value set to 5,000. This was done to explicitly compare the speeds of the univariate models with the Stochastic Gradient Descent Method.

Learning rates were chosen based on what was appropriate given the data at hand. For example, non-standardized data required the learning rates for the *m* and *b* parameters to be orders of magnitude different in size to prevent the gradient from blowing up. Meanwhile, standardized data needed consistent and less precise learning rates for the two learning rates. Learning rates for multivariate gradient descent were treated similarly.

In this report, we will investigate stochastic and non-stochastic gradient descent algorithms and will be implementing standardized and non-standardized data simultaneously. This will help us get a better picture and gain more confidence of which properties of concrete are being identified with different variations of gradient descent. This implementation of stochastic gradient descent chooses a single random data point to adjust the partial derivatives of *m* and *b,* as described in the pseudocode below. Ideally, this will speed up training times, but be much less "smooth" during its descent as it chooses random datapoints.

Univariate Algorithm Pseudocode:

```
function univariateGradientDescent(X, y, m, b, lr_m, lr_b, is_stochastic):
        If is_stochastic:
                Grab single random sample from X
                Get corresponding sample's y
        Calculate yhat with X, m, and b using yhat = mx + b
        Calculate partial derivative of MSE with respect to m
        Calculate partial derivative of MSE with respect to b
        Update m and b with new partial derivatives and learning rates
        Return m and b

function linearRegression(X, y, lr_m, lr_b, threshold, epochs, is_stochastic):
        Initialize m and b to 0.0
        Create array for costs of empty zeros for each epoch
        If threshold:
                While threshold not met and total iterations < 10,000:
                        Run gradient descent algorithm
                        Assign m and b to output of univariateGradientDescent()
                        Recalculate yhat
                        Calculate cost and determine if threshold met
```

Else no threshold:
For number of epochs**:**
Run gradient descent algorithm
Assign m and b to output of **univariateGradientDescent**()
Recalculate yhat
Calculate cost
Return m, b, and cost

function runUnivariateRegression():
Parse data from CSV as X
Initialize hyperparameters lr_m, lr_b, epochs, is_stochastic
Collect random sample of 900 values from X
For each feature:
Create training and testing data
If using standardized data:
Standardize training data and testing data with training data mean and std
Calculate m, b, and cost using **linearRegression**()

Multivariate Algorithm Pseudocode:

function multivariateGradientDescent(X, y, w, lr_w, is_stochastic)
If is_stochastic:
Grab single random sample from X
Get corresponding sample's y
Calculate yhat with X and w using yhat = w * X
Calculate partial derivative of MSE with respect to w
Update w with partial derivatives and learning rate
Return w

function MVlinearRegression(X, y, lr_m, lr_b, threshold, epochs, is_stochastic):
Initialize m and b to 0.0
Create array for costs of empty zeros for each epoch
For number of epochs**:**
Assign w to output of **multivariateGradientDescent** ()
Calculate yhat with X and w using yhat = w * X
Calculate cost
Return w and cost

function runMultivariateRegression():
Parse data from CSV as X
Initialize hyperparameters lr_w, epochs, is_stochastic
Collect random sample of 900 values from X
Create training data
Create testing data
If using standardized data:
Standardize testing data with training data mean and std
Calculate w and cost using **MVlinearRegression** ()
Make predictions with train and test data
Calculate R2 for train and test data

Use of Standardization:

Standardization of feature values allows there to be a smaller distribution of values between feature to feature. The following formula was used to calculate the standardized data:

$$Z = \frac{X - \mu}{\sigma}$$

where Z becomes our standardized vector after we subtract mean μ from vector X and dividing by standard deviation σ.

Below, we can see the difference between non-standardized data and the default data from the dataset. Due to the nature of our features, the values could be completely different from one another, so standardization removes this and focuses exclusively on the standardized data.

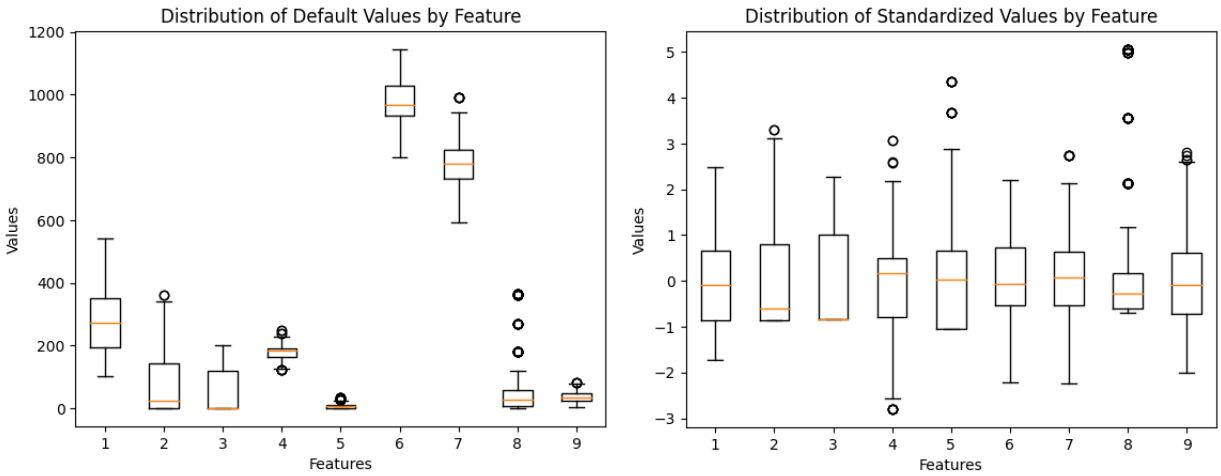

Figure 1. Distributions of default (left) and standardized input values by feature. Refer to list of features above.

In order to standardize X_test and y_test values during the Testing phase, the means and standardization of the Training phase were used to prevent our model from using clues of ideally not yet seen future data. This was done because in the real world as unforeseen data is processed by our model, it's impossible to get the true mean and standard deviation of all the future data until the very end.

Results

The results of generated univariate and multivariate models are below. For Univariate models, training/testing MSE and R-Squared (R2) values, as well as final m and b values are provided in the snapshots below of models that successfully explained >10% of variance explained for two of the features. Each table and figure pair represent the different variations of the gradient descent algorithm that were performed and contain 3 more pages of results than was expected on this assignment. The highest performing model is then focused on during the Discussion section on univariate results. Multivariate results for each of the models are at the end.

Univariate Results (Basic GD, SGD, GD Standardized, SGD Standardized):

Basic Gradient Descent (models over 10% VE in bold) – 500 epochs

	Feature (kg in a m^3 mixture)	trainMSE	trainR2	testMSE	testR2	m	b
1	**Cement**	**211.019**	**0.258**	**214.094**	**0.112**	**0.091**	**9.707**
2	Blast Furnace Slag	280.224	0.014	229.171	0.050	0.026	33.683
3	Fly Ash	281.445	0.010	239.977	0.005	-0.018	36.447
4	Water	313.919	-0.105	262.870	-0.090	0.110	15.164
5	Superplasticizer	280.079	0.015	237.785	0.014	0.057	35.403
6	Coarse Aggregate	303.900	-0.069	261.869	-0.086	0.032	4.600
7	Fine Aggregate	309.355	-0.089	270.661	-0.122	0.037	6.647
8	**Age (day)**	**253.750**	**0.107**	**214.380**	**0.111**	**0.092**	**31.443**

Table 1. Train/test MSE and R-Squared values with m and b values for each feature of Basic Gradient Descent. Each ran 500 epochs.

Predictions over Training Data Plots:

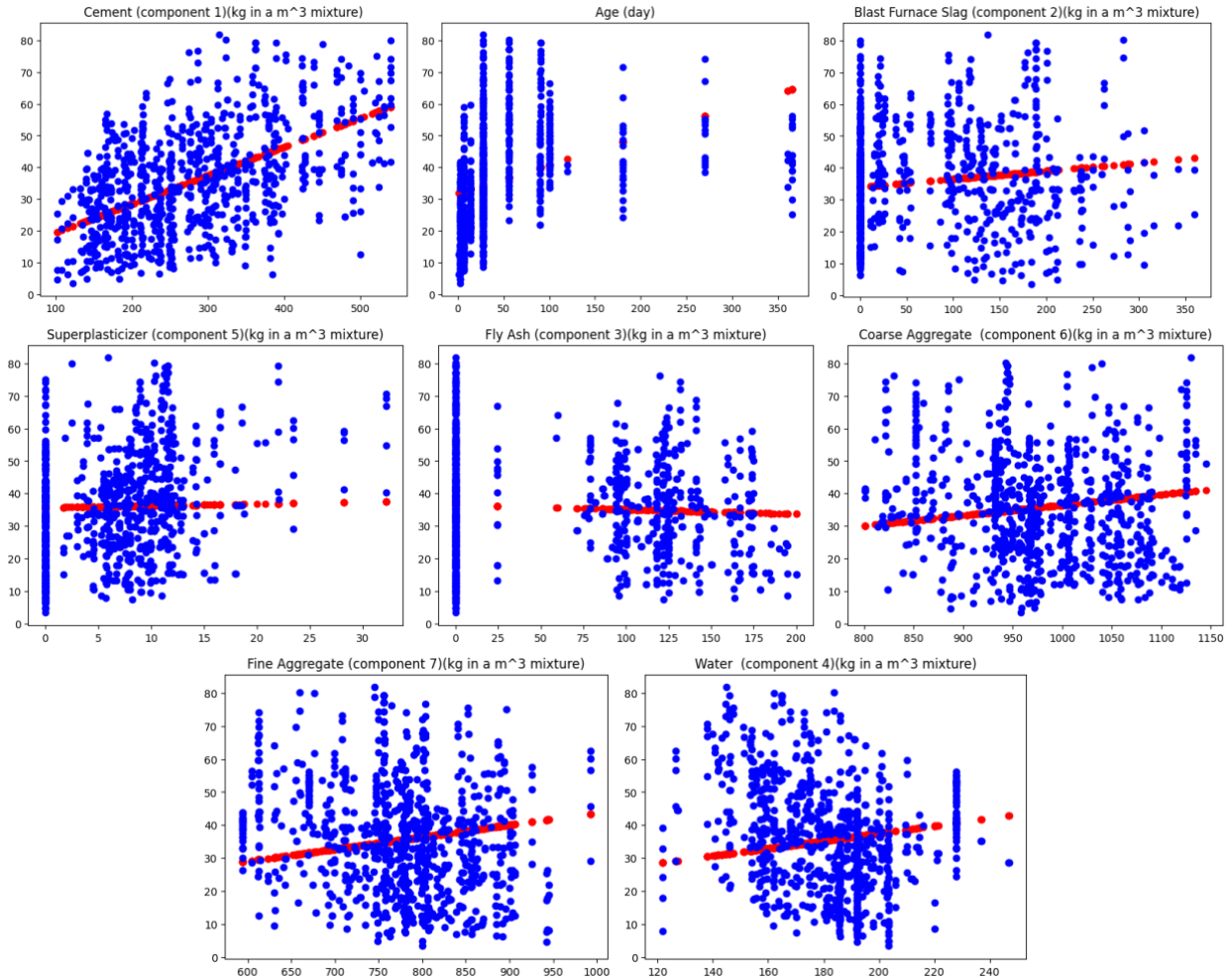

Figure 2. Predictions (red) over Training (blue) data points for Basic Gradient Descent

Stochastic gradient descent (models over 10% VE in bold) – 500 epochs

	Feature (kg in a m^3 mixture)	trainMSE	trainR2	testMSE	testR2	m	b
1	**Cement**	214.936	**0.228**	234.761	**0.164**	0.101	5.721
2	Blast Furnace Slag	279.474	-0.004	275.397	0.020	0.044	34.255
3	Fly Ash	296.626	-0.065	293.793	-0.046	-0.019	32.336
4	Water	371.233	-0.333	369.284	-0.314	0.213	2.362
5	**Superplasticizer**	247.457	**0.111**	251.336	**0.105**	0.600	31.756
6	Coarse Aggregate	6912.700	-23.822	6855.633	-23.403	-0.051	3.573
7	Fine Aggregate	1179.722	-3.236	1187.340	-3.226	0.089	-4.946
8	**Age (day)**	257.098	0.077	236.827	**0.157**	0.125	30.518

Table 2. Train/test MSE and R-Squared values with m and b values for each feature of Stochastic Gradient Descent. Each ran 500 epochs.

Predictions over Training Data Plots:

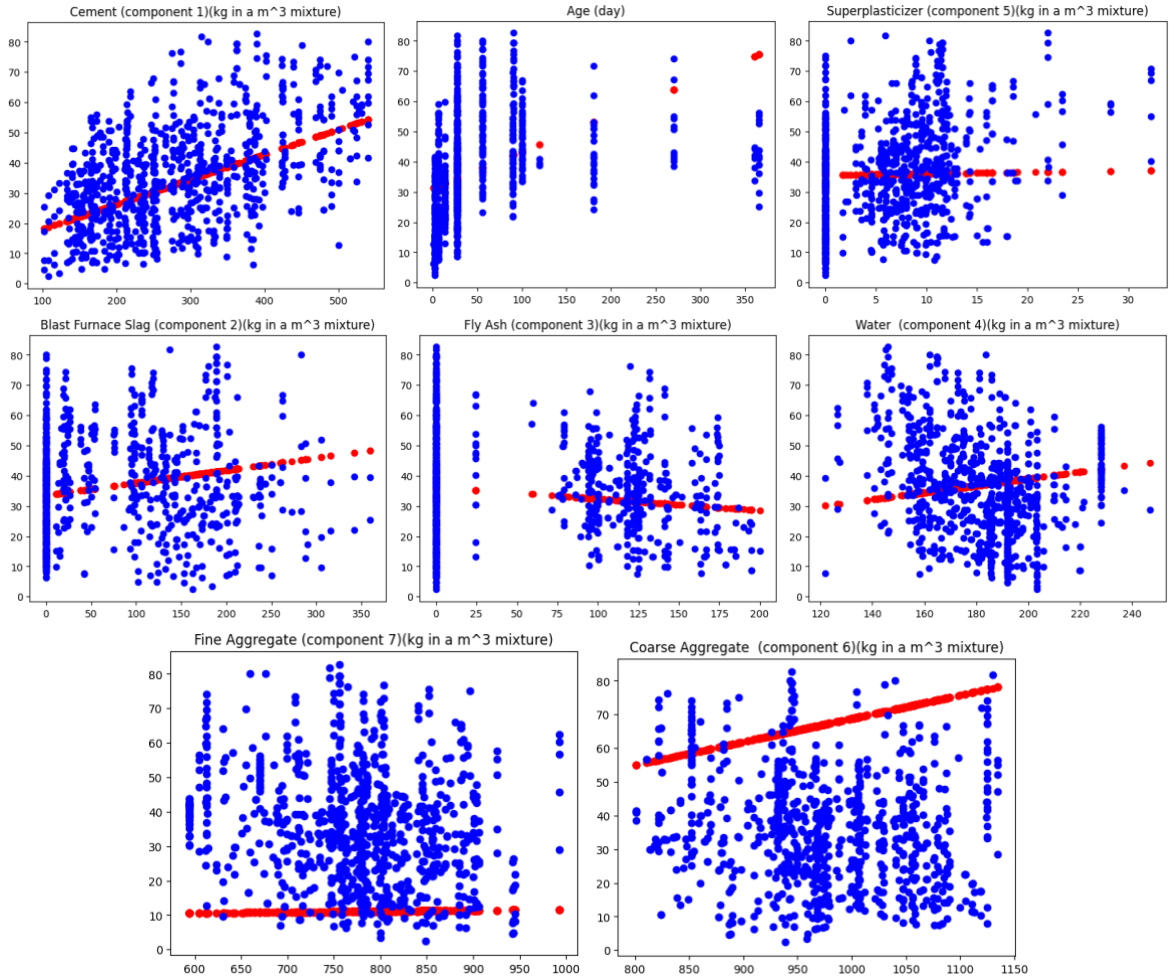

Figure 3. Predictions (red) over Training (blue) data points for Stochastic Gradient Descent

Basic Gradient Descent: Standardized Data (models over 10% VE in bold) – 500 epochs

	Feature (kg in a m^3 mixture)	trainMSE	trainR2	testMSE	testR2	m	b
1	**Cement**	0.802	**0.197**	1.075	**0.124**	0.302	1.674E-16
2	Blast Furnace Slag	0.979	0.020	1.236	-0.007	0.096	1.542E-16
3	Fly Ash	0.994	0.005	1.197	0.024	-0.049	1.545E-16
4	Water	0.916	0.083	1.223	0.003	-0.196	1.521E-16
5	**Superplasticizer**	0.859	**0.140**	1.218	0.007	0.254	1.544E-16
6	Coarse Aggregate	0.962	0.037	1.232	-0.004	-0.130	9.791E-17
7	Fine Aggregate	0.982	0.017	1.223	0.003	-0.088	1.633E-16
8	**Age (day)**	0.910	0.089	1.084	**0.116**	0.203	1.601E-16

Table 3. Train/test MSE and R-Squared values with m and b values for each feature of Standardized Basic Gradient Descent. Each ran 500 epochs.

Predictions over Training Data Plots:

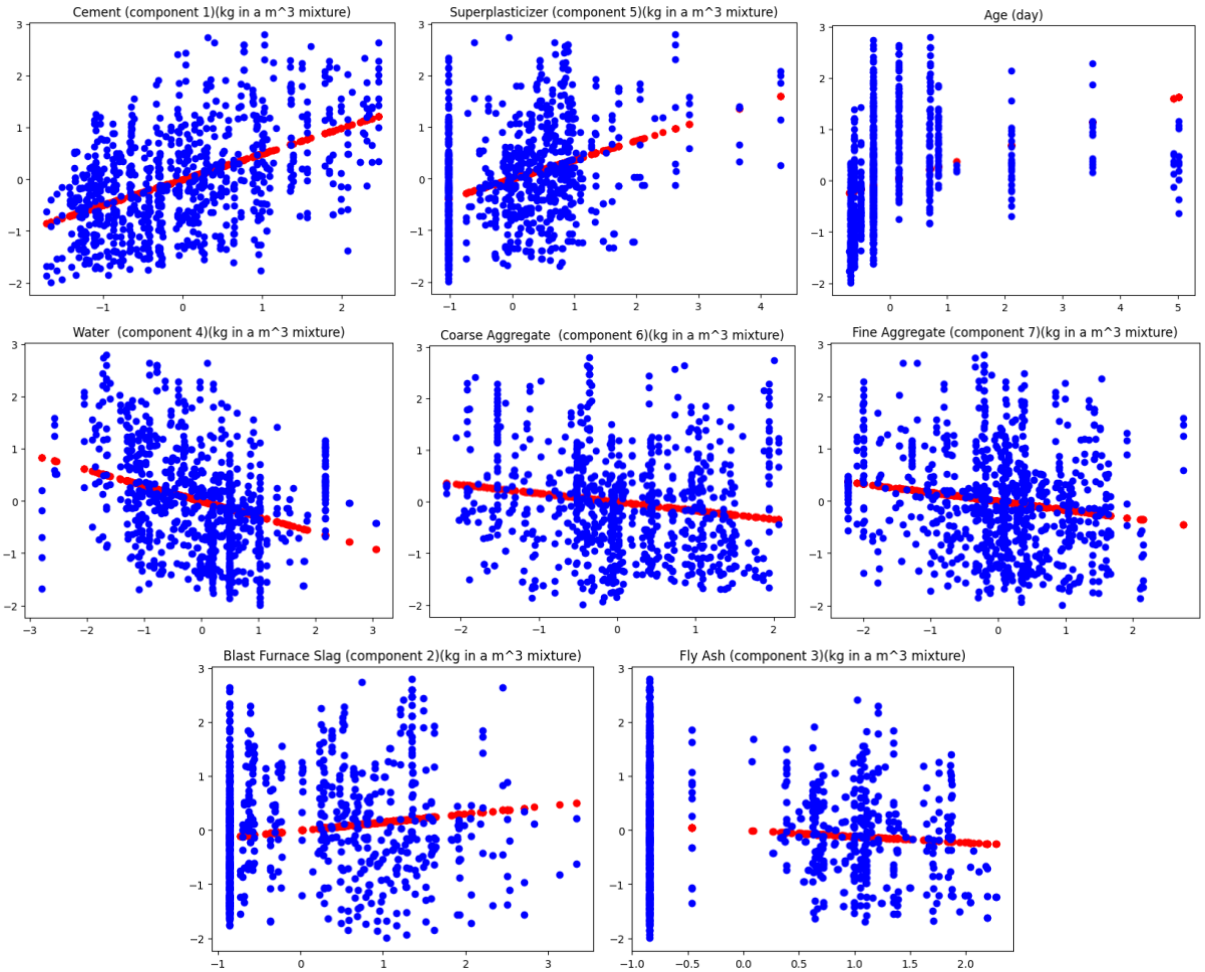

Figure 4. Predictions (red) over Training (blue) data points for Standardized Basic Gradient Descent

Stochastic Gradient Descent: Standardized Data (models over 10% VE in bold) – 500 epochs

	Feature (kg in a m^3 mixture)	trainMSE	trainR2	testMSE	testR2	m	b
1	**Cement**	**0.759**	**0.240**	**0.925**	**0.117**	**0.437**	**-0.085**
2	Blast Furnace Slag	1.028	-0.030	1.057	-0.009	0.133	-0.214
3	Fly Ash	1.021	-0.022	1.054	-0.006	-0.252	0.093
4	Water	0.964	0.035	1.024	0.022	-0.516	-0.049
5	**Superplasticizer**	**0.872**	**0.127**	**0.930**	**0.112**	**0.423**	**0.080**
6	Coarse Aggregate	0.997	0.002	1.037	0.009	-0.308	-0.002
7	Fine Aggregate	0.994	0.005	1.051	-0.004	-0.218	0.136
8	Age (day)	0.938	0.061	0.978	0.066	0.106	0.020

Table 4. Train/test MSE and R-Squared values with m and b values for each feature of Standardized Stochastic Gradient Descent. Each ran 500 epochs.

Predictions over Training Data Plots:

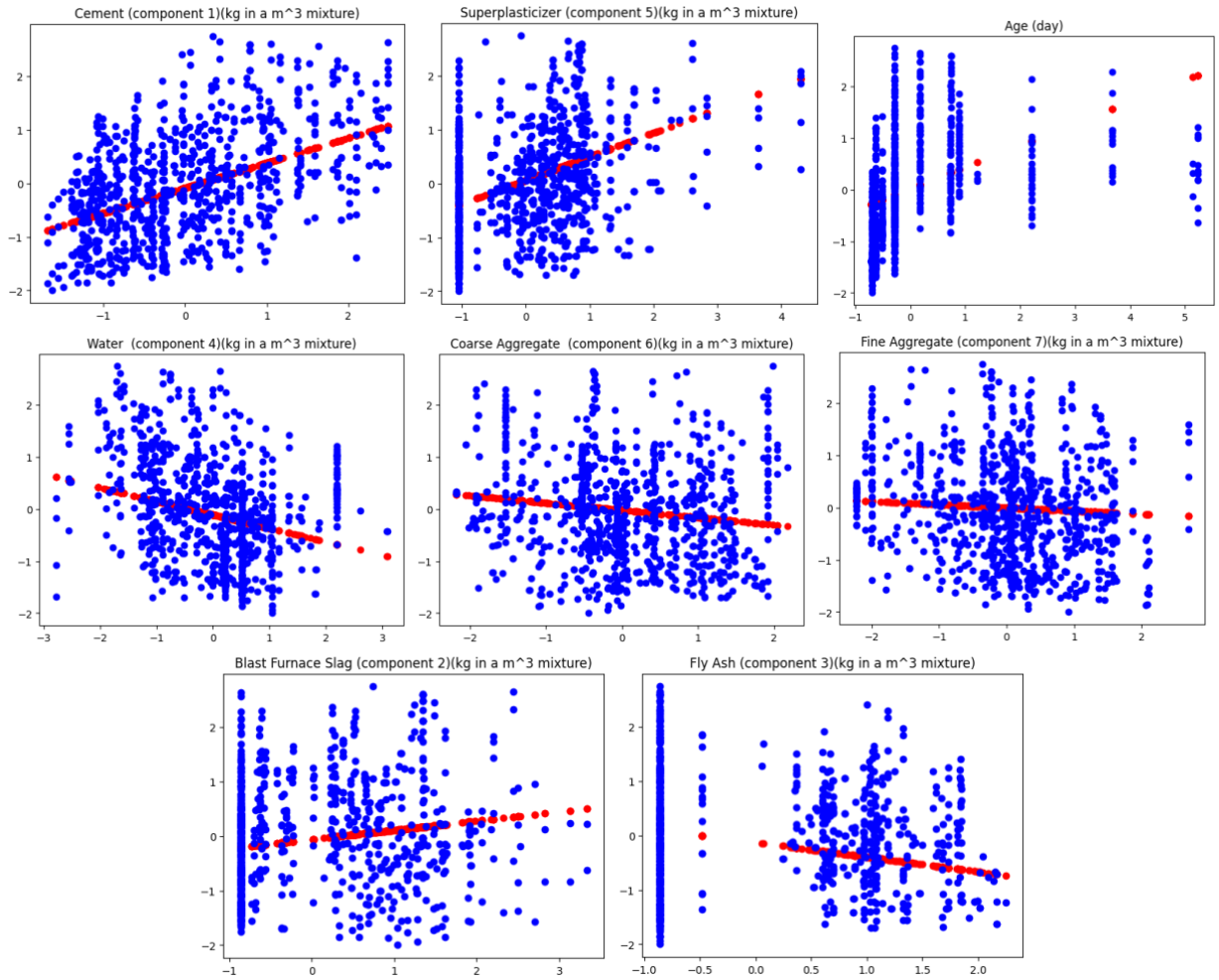

Figure 5. Predictions (red) over Training (blue) data points for Standardized Stochastic Gradient Descent

Results of Gradient Descent with and without Stochastic Selection and Standardized Data:

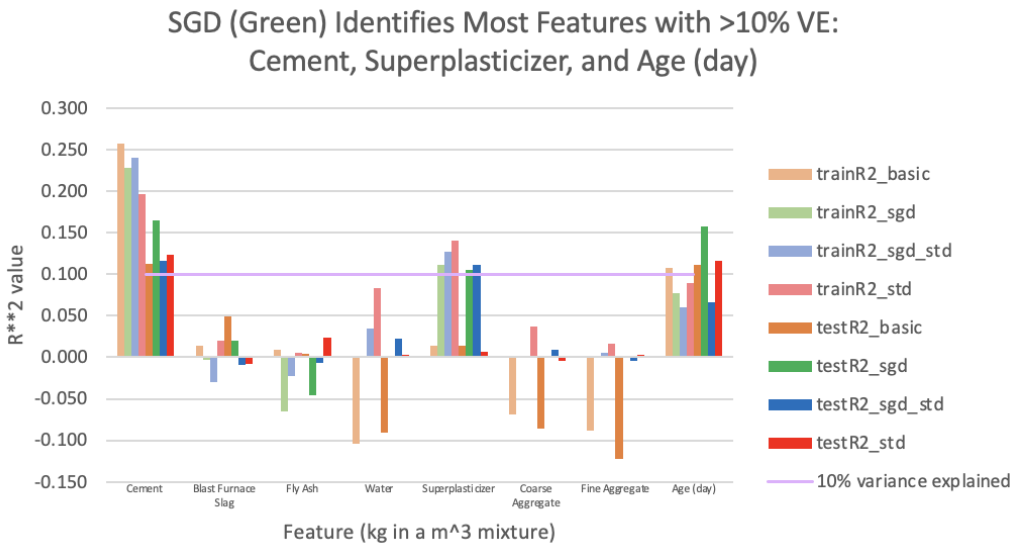

Figure 6. R-Squared values of the training (lighter shade) and testing (darker shade) methods of gradient descent. Outliers with insignificantly low R-Squared values omitted to highlight method similarities/differences.

Univariate with SGD timing:

For this data, SGD was given a threshold of 0.001 and a max iteration value of 5,000 to allow the linear regression algorithm to perform gradient descent until it achieves a change smaller than the threshold.

Feature (kg in a m^3 mixture)	Train Time(s)
Cement	0.0318
Blast Furnace Slag	0.0813
Fly Ash	0.0404
Water	0.0225
Superplasticizer	0.0480
Coarse Aggregate	0.8497
Fine Aggregate	0.4445
Age (day)	0.3065

Table 5. Train times for Stochastic Gradient Descent

Multivariate Results:

Final weights provided in the Supporting Information section.

Multivariate Approach	R2_train	R2_test	epochs	lr_w
Basic Gradient Descent	0.521	0.556	1000	0.0000001
Stochastic Gradient Descent	0.176	0.221	1000	0.0000001
Standardized Basic	0.601	0.589	1000	0.01
Standardized Stochastic	0.598	0.525	1000	0.01

Table 6. Multivariate results for each gradient descent method

Discussion

In this report various methods of gradient descent are compared in order to discover the most promising model with highest explained variance. Figure 5 shows each univariate model and reveals the features most predictive of Concrete Compressive Strength: Cement, Superplasticizer, and Age (days). For the univariate models, Stochastic Gradient Descent was able to identify all three of these features where it could explain >10% of the variance.

In most cases, the models that accurately predicted the training data also accurately predicted the testing data. In stochastic gradient descent, the model was able to explain >10% variance, or "sufficient performance", in both training and test data for Cement and Superplasticizer, however, had sufficient performance in only the testing data for Age (days). 7 times out of 8, when the model had sufficient performance in the training data it was also able to replicate that performance with testing data. There were only two cases when Age (day) did not achieve sufficient variance explanation in the training data, but were sufficiently explained in the testing data, which were with SGD and Standardized Basic Gradient Descent.

SGD's multivariate model performed better in the testing data than any univariate model. However, the performance of Basic, SGD, and SGD with Standardization performed better on the training data when looking specifically at Concrete than the multivariate was able to do using all features. Being that Concrete was consistently the most predictive of Concrete compressive strength than any other feature, perhaps the introduction of the other features introduced noise. Although, I would hesitate to make that claim as it still performs better on testing data than all other univariate model.

Regarding the runtime if the models, to understand the difference in univariate models of a specific method a simulation with SGD was performed with a threshold value of 0.001 and max iteration value of 5,000. Water trained the fastest at 0.0225 s and Coarse Aggregate trained the slowest taking 0.8497 s. SGD simulations generally completed faster and allowed fewer epochs for sufficient modeling. Regarding the non-standardized univariate models, select hyperparameters allowed me to achieve two models with greater than >10% variance explained, without needing to use different ones. These hyper parameters were 500 epochs, 0.000001 and 0.01 learning rates for m and b, respectively. Similarly, using standardized I just needed 0.01 as the learning rate for both m and b to achieve the desired results. The effects of standardization on the data were relatively minimal in the univariate models. However, in the multivariate model, the models that were standardization were the highest performing. Specifically, the standardized variation of non-stochastic gradient descent was the highest performer in training and testing data.

Figure 6 shows each univariate model and reveals the features most predictive of Concrete compressive strength: Cement, Superplasticizer, and Age (days). None of these are much of a surprise as cement is the main component of concrete, superplasticizers are additives used in making high-strength concrete, and age being known as a key indicator to concrete strength provided there's sufficient moisture. Making the hardest possible concrete would involve maximizing the amount of Cement and Superplasticizers as much as possible.

Supporting Information

W weights from Multivariate Model:

Basic Gradient Descent

w = [0.10059846, 0.06899172, 0.05212576, -0.04025558, 0.01452645, -0.00262097
  0.00762081, 0.07436207]

Stochastic Gradient Descent

w = [4.39939066e-02, 1.80166129e-02, -2.07715974e-03, 4.58711712e-05,
  2.28166319e-03, 1.39340604e-02, 1.49815966e-02, 1.56350888e-02]

Standardized Basic

w = [0.43850898, 0.23451984, 0.04304745, -0.33478448, 0.17223219, -0.08891151,
 -0.16161638, 0.41234137]

Standardized Stochastic

w = [0.40772148, 0.26478236, 0.01933092, -0.34541429, 0.22796623, -0.05889749,
 -0.17016951, 0.48684769]