Introduction

Concrete is one of the most common and most important materials in civil engineering, the data set that has been given is about things that contain in the concrete as long as the compressive strength of the concrete, it has 8 quantitative input variables and 1 quantitative output value, there's no missing values in the data set.

My goal of this problem is:

1) separate the data set into training data and testing data, the first 900 instances for training and the rest for testing
2) build an uni-variate linear regression model, a multi-variate linear regression model, and a multi-variate quadratic model separately
3) use steepest gradient descent to make sure the loss function(here is MSE) of all the models reaches its minimum when using the training data
4) use my testing data to test the performance(here is MSE and 1-MSE/Variance(observed)) of my final models
5) normalize each variable, and do things again from step 2), and this time use normalized data
6) compare the difference between using raw data and using normalized data, draw conclusions

Details of my algorithms:

	m initial values	b initial value(if exists)	Learning rate(before normalization)	Learning rate(after normalization)	Max iteration	Stop criterion	Stochastic gradient descent
Uni-variate	0	0	1e-06	1e-01	1000	1e-06	Yes
Multi-variate	All 0		1e-07	1e-01	1000	1e-06	Yes
Multi-variate quadratic	All 0		1e-13	1e-02	1000	1e-06	Yes

pseudo-code of the algorithms:

1) the uni-variate linear regression algorithm:

```
def uni_var_lr(x, y, m, b):
    sum1, sum2, sum3 = 0.0, 0.0, 0.0
    for i in range(length of the training data):
        sum1 += (predict y – observed y) ** 2
        sum2 += 2 * x * (predict y – observed y)
        sum3 += 2 * (predict y – observed y)
    mse = sum1 / length of the training data
    derivative of m = sum2 / length of the training data
    derivative of b = sum3 / length of the training data
    return mse, derivative of m, derivative of b
```

2) the gradient algorithm for uni-variate linear regression:

```
def grad_uni(x, y, m, b, step size=1e-06, max iteration times=1000,stop criterion=1e-06):
    i = 0
    previous loss = 0
    previous m = m
    previous b = b
```

```
    current loss, derivative of m, derivative of b = uni_var_lr(x, y, m, b)
    current m = previous m – step size * derivative of m
    current b = previous b – step size * derivative of b
    while i < max iteration times:
        if abs(current loss – previous loss) < stop crit:
            break
        else:
            previous loss = current loss
            current loss, derivative of m, derivative of b = uni_var_lr(x, y, m, b)
            current m = previous m – step size * derivative of m
            current b = previous b – step size * derivative of b
            i += 1
    return current loss, current m, current b
```

3) the multi-variate linear regression algorithm:

```
def multi_var_lr(x, y, m):
    sum1, sum2, sum3 = 0.0, 0.0, 0.0
    for i in range(length of the training data):
        for j in range(number of input features):
            sum1 += predict y
        sum2 += (predict y – observed y) ** 2
        sum3 += 2 * x * (predict y – observed y)
    mse = sum2 / length of the training data
    derivatives of m = sum3 / length of the training data
    return mse, derivatives of m
```

4) the gradient algorithm for multi-variate linear regression:

```
def grad_multi(x, y, m, step size=1e-07, max iteration times=100,stop_crit=1e-06):
    i = 0
    previous loss = 0
    previous m = m
    current loss, derivative of m = multi_var_lr (x, y, m, b)
    current m = previous m – step size * derivative of m
    while i < max iteration times:
        if abs(current loss – previous loss) < stop crit:
            break
        else:
            previous loss = current loss
            current loss, derivative of m, derivative of b = multi_var_lr(x, y, m)
            current m = previous m – step size * derivative of m
            i += 1
    return current loss, current m
```

5) the multi-variate quadratic regression algorithm:

```
def multi_var_poly_lr(x, y, m):
    sum1, sum2, sum3 = 0.0, 0.0, 0.0
    for i in range(length of the training data):
```

```
            for j in range(number of input features):
                sum1 += predict y
            sum2 += (predict y – observed y) ** 2
            sum3 += 2 * x * (predict y – observed y)
        mse = sum2 / length of the training data
        derivatives of m = sum3 / length of the training data
        return mse, derivatives of m
```

6) the gradient algorithm for multi-variate quadratic regression:

```
def grad_multi_poly(x, y, m, step size=1e-13, max iteration times=100,stop_crit=1e-06):
    i = 0
    previous loss = 0
    previous m = m
    current loss, derivative of m = multi_var_poly_lr (x, y, m, b)
    current m = previous m – step size * derivative of m
    while i < max iteration times:
        if abs(current loss – previous loss) < stop crit:
            break
        else:
            previous loss = current loss
            current loss, derivative of m, derivative of b = multi_var_poly_lr (x, y, m)
            current m = previous m – step size * derivative of m
            i += 1
    return current loss, current m
```

7) the normalization algorithm:

```
def normalize_(x):
    mean = np.mean(x, 0)
    standard deviation = np.std(x, 0)
    return (x – mean) / std
```

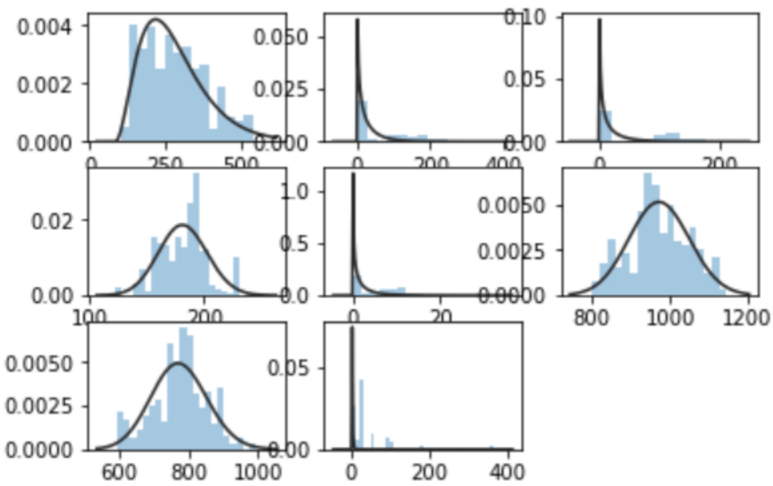
As for my quadratic model, it has 45 variables, which is:

1) for x_i * x_j(i != j), there are 28 variables
2) for x_i ^ 2, there are 8 variables
3) for x_i, there are 8 variables
4) there is also a constant variable

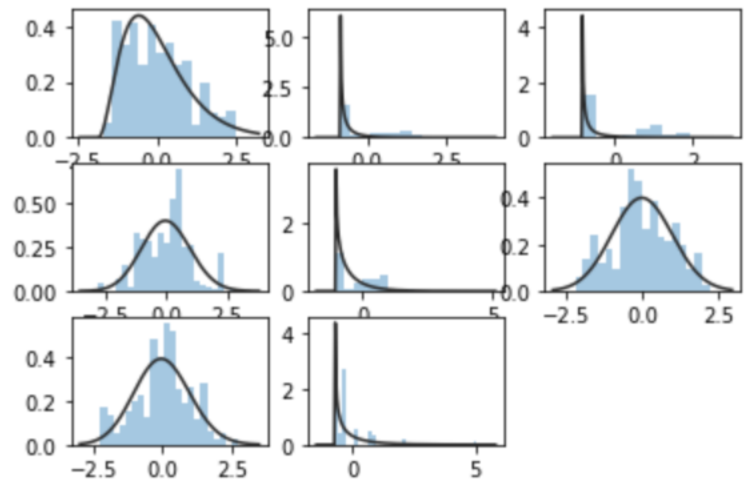
As for the normalization part, what I did is to make sure all the 8 variables have a mean of 0 and a standard deviation of 1, the following chart is **the description of the variables after normalization.**

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
count	1.030000e+03	1.030000e+03	1.030000e+03	1.030000e+03	1.030000e+03	1.030000e+03	1.030000e+03	1.030000e+03
mean	2.988252e-15	8.768606e-16	-1.798992e-16	7.400768e-16	4.690962e-16	6.068932e-15	1.225958e-14	7.546283e-16
std	1.000486e+00	1.000486e+00	1.000486e+00	1.000486e+00	1.000486e+00	1.000486e+00	1.000486e+00	1.000486e+00
min	-1.715253e+00	-8.568879e-01	-8.471439e-01	-2.800211e+00	-1.039143e+00	-2.212138e+00	-2.240917e+00	-7.073594e-01
25%	-8.500535e-01	-8.568879e-01	-8.471439e-01	-7.808939e-01	-1.039143e+00	-5.265174e-01	-5.319697e-01	-6.123314e-01
50%	-7.915193e-02	-6.017783e-01	-8.471439e-01	1.608294e-01	3.271508e-02	-6.329352e-02	7.386739e-02	-2.797331e-01
75%	6.589606e-01	8.007446e-01	1.002278e+00	4.887927e-01	6.691307e-01	7.267605e-01	6.291661e-01	1.637312e-01
max	2.477915e+00	3.310675e+00	2.281084e+00	3.065647e+00	4.353642e+00	2.214224e+00	2.733062e+00	5.057677e+00

The distribution of the 8 feature variables **before normalization:**



The distribution of the 8 feature variables **after normalization:**



Results

Variance explained of my models on the **training dataset:**

Model = Uni	column= "Cement"	column= "Blast Furnace Slag"	column= "Fly Ash"	column= "Water"	column= "Superplasticizer"	column= "Coarse Aggregate"	column= "Fine Aggregate"	column= "Age"
Before normalization 1-MSE/Variance (Observed)	0.158	-2.439	-2.826	-0.209	-3.682	-0.088	-0.125	-2.144
After normalization 1-MSE/Variance (Observed)	0.229	0.018	0.003	0.088	0.175	0.039	0.033	0.112

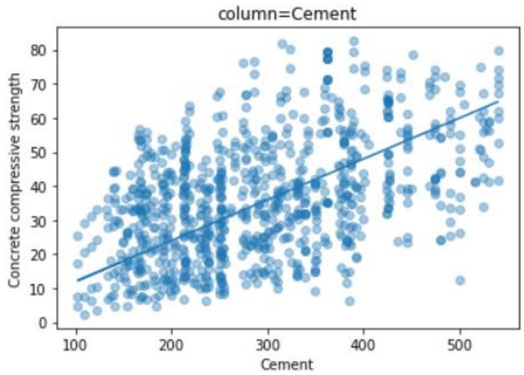
Model	Multi	Multi-quadratic
Before normalization 1-MSE/Variance(Observed)	0.419	0.501
After normalization 1-MSE/Variance(Observed)	0.613	0.783

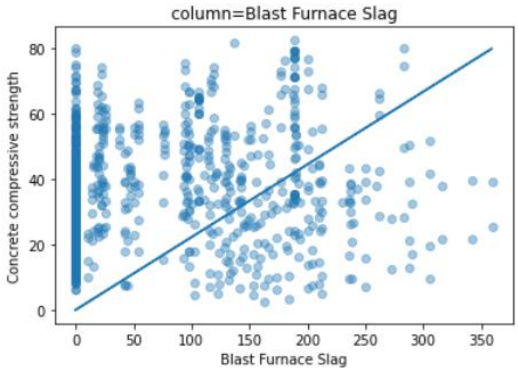
Variance explained of my models on the **testing dataset:**

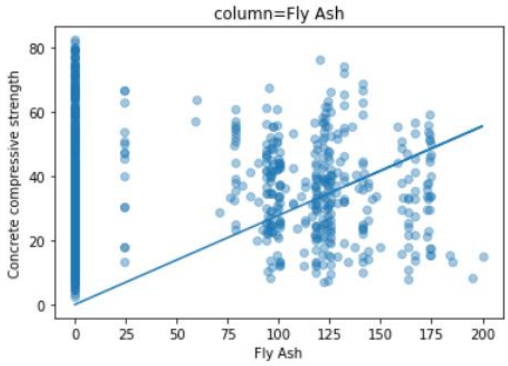
Uni	column= "Cement"	column= "Blast Furnace Slag"	column= "Fly Ash"	column= "Water"	column= "Superplasticizer"	column= "Coarse Aggregate"	column= "Fine Aggregate"	column= "Age"
Before normalization 1-MSE/Variance (Observed)	0.234	-1.417	-4.263	-0.290	-5.319	-0.145	-0.184	-3.598
After normalization 1-MSE/Variance (Observed)	0.439	-0.102	-0.031	-0.070	-0.624	-0.313	-0.166	-0.0420

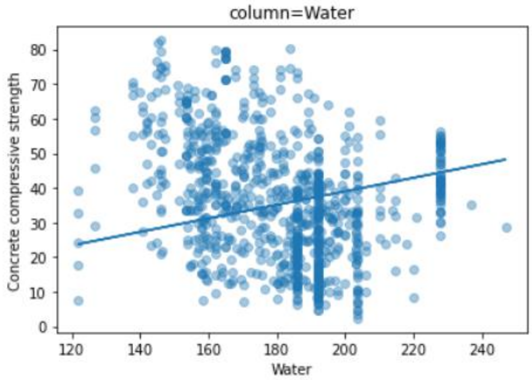
Model	Multi	Multi-quadratic
Before normalization 1-MSE/Variance(Observed)	0.599	0.629
After normalization 1-MSE/Variance(Observed)	0.581	0.693

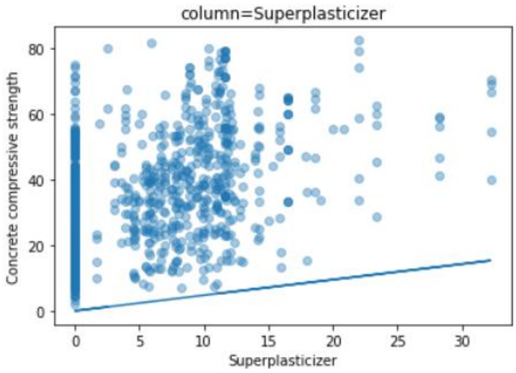
Plots of my trained uni-variate models **before normalization:**

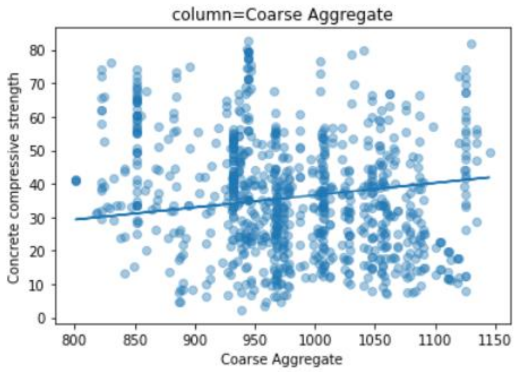


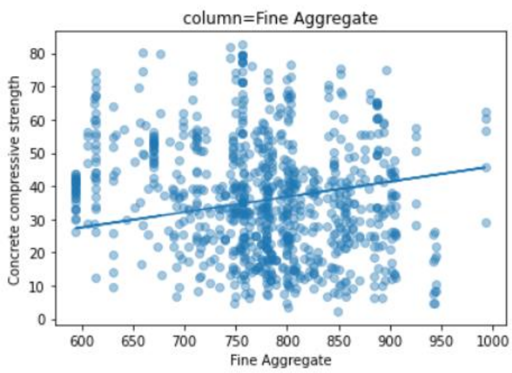


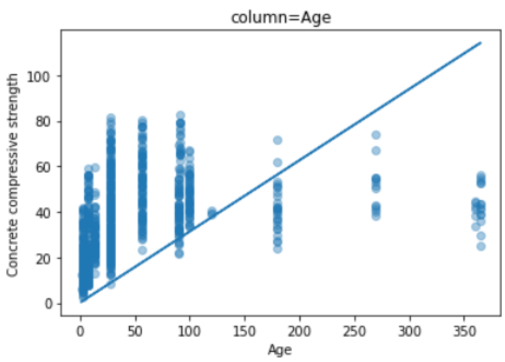




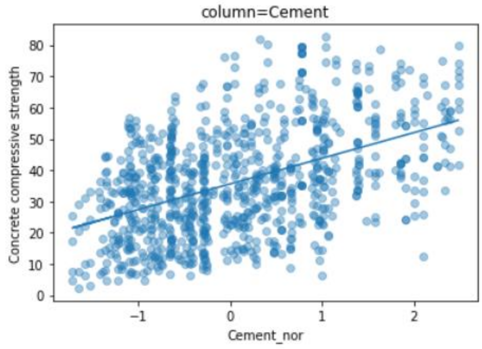


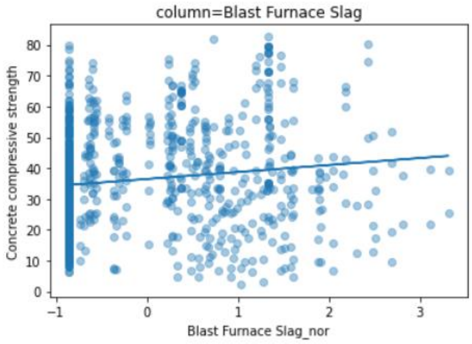


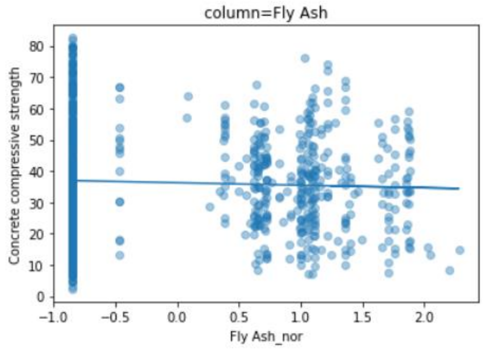


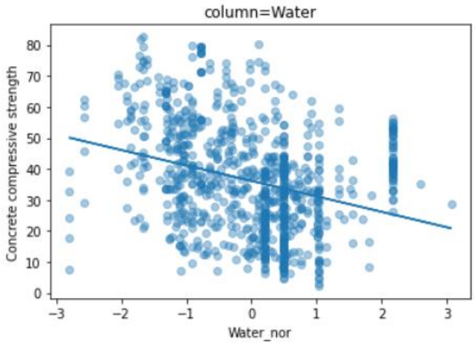


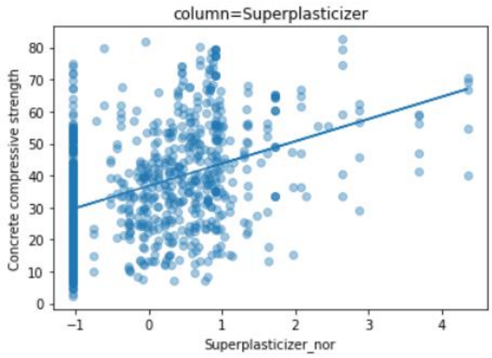
Plots of my trained uni-variate models **after normalization:**

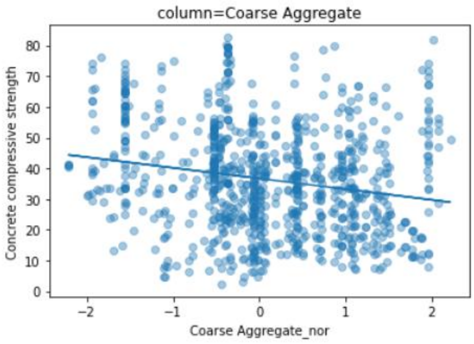


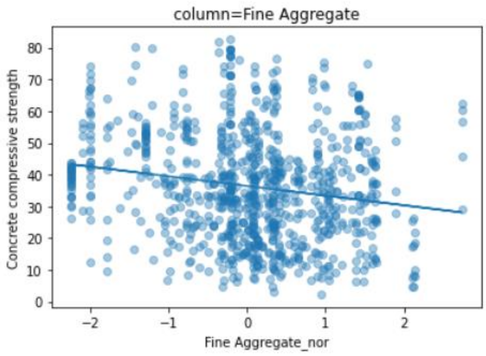


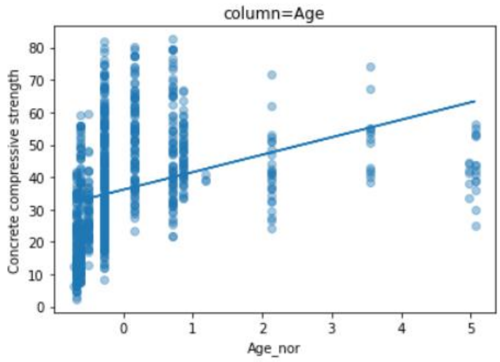












Discussion

MSE of my models on the **training dataset:**

Model = Uni	column= “Cement”	column= “Blast Furnace Slag”	column= “Fly Ash”	column= “Water”	column= “Superplasticizer”	column= “Coarse Aggregate”	column= “Fine Aggregate”	column= “Age”
Before normalization MSE	249.363	1018.742	1133.462	358.190	1387.206	322.349	333.437	947.544
After normalization MSE	228.334	290.775	295.265	270.100	244.314	284.485	286.315	262.914

Model	Multi	Multi-quadratic
Before normalization MSE	172.000	147.716
After normalization MSE	114.566	64.262

MSE of my models on the **testing dataset:**

Uni	column= “Cement”	column= “Blast Furnace Slag”	column= “Fly Ash”	column= “Water”	column= “Superplasticizer”	column= “Coarse Aggregate”	column= “Fine Aggregate”	column= “Age”
Before normalization MSE	110.928	350.083	762.278	186.887	915.181	165.957	171.519	666.028
After normalization MSE	81.134	81.134	149.443	149.443	235.263	190.242	168.989	150.915

Model	Multi	Multi-quadratic
Before normalization MSE	58.000	53.672
After normalization MSE	60.559	44.371

According to the MSE above, as for performance on the training data, before normalization, the uni-variate linear model that uses column “Cement” as its input value is the best, because it has the lowest MSE value of 249.363, after normalization, the multi-variate linear regression model is the best, which has the MSE value of 133.069.

Also, those MSE values shows that the same models that performed well on the training data also do well on the testing data, and actually those models even perform better on the testing data.

The coefficients of the uni-variate models when using training data:

	column= "Cement"	column= "Blast Furnace Slag"	column= "Fly Ash"	column= "Water"	column= "Superplasticizer"	column= "Coarse Aggregate"	column= "Fine Aggregate"	column= "Age"
Before normalization m value	0.119	0.222	0.278	0.195	0.476	0.036	0.045	0.312
Before normalization b value	0.003	0.043	0.047	0.476	0.069	0.00019	0.001	0.044
After normalization m value	8.208	2.268	-0.803	-4.981	6.947	-3.473	-3.044	-3.044
After normalization b value	35.635	36.515	36.309	36.176	36.832	-3.473	-3.044	36.155

The coefficients of the multi-variate models when using training data:

Model	Multi	Multi-quadratic
Before normalization m value	[3.790327243640112e-05, 0.08086785968175879, 0.042071073519139635, 0.022220504160116176, -0.014110288180942418, 0.008001481161360251, 0.002876080547145025, 0.007841241152307594, 0.05146968947985563]	[6.25302469687737e-11, 1.5970274059632005e-06, 3.271399585260447e-05, 5.430828560723212e-06, 4.211526984725497e-05, 1.966238653423182e-06, -2.32249829456496e-06, 5.468981035836778e-07, 5.7146733784934444e-06, 6.9328390049009715e-06, 9.932114003763194e-08, 7.108036025619448e-06, -9.220739570364251e-06, -1.2198106436119545e-05, 1.1589200372421938e-06, -3.98252041358304e-06, 3.383235471344772e-06, 2.294734806683279e-05, 2.9052844023801458e-05, 3.2739591932856567e-07, 4.710722991484205e-06, 3.850459487282615e-06, 5.518943760087005e-06, 3.148212617380582e-05, 3.849366252658102e-05,

		8.661279718354981e-07, 6.336622807458082e-06, 2.0439425902035464e-06, 5.820747629226773e-06, 1.3355869362121036e-05, 4.3306889523000073e-05, 4.815681965565363e-05, 2.5287307565729075e-06, 5.402499275441506e-06, 8.299803251642924e-06, 1.855399450553925e-05, 3.2791938672640706e-05, 6.157442036995578e-08, 4.223879033850882e-08, 2.985144785279869e-08, -6.284927036747852e-09, 7.309564422459924e-09, 2.8728229981900896e-08, 2.7728323863751732e-08, 4.3602101662834474e-08]
After normalization m value	[35.83369375998126, 11.571005552628092, 8.15650791559382, 5.493777838561696, -3.7613771190525522, 1.8692987371552356, 1.0194565404008968, 0.8168865601983376, 7.282783733758603]	[36.037965362019975, -10.27605384374023, 4.562331595662635, -1.3780001646469, 3.149386604222147, 0.44099778838969683, -0.4320140890822067, 2.357674731052929, 0.31611972717603865, 1.6393178168920268, -4.30368014326168, 2.045183493030496, -1.0326746279457137, -2.1640910323728377, 0.976358890495911, -0.6940090284985427, 1.9450796153186125, 0.9600291597612411, 0.2033614872160939, -0.5733124768718575, -0.0832646559353098, -1.2680728837476705, 2.3480440799773947,

		1.9676596385008127, 0.6295303809656915, 1.3964144599407597, 0.9445238053711656, 0.7985967770411262, -1.4632658090047541, 1.6274530541367977, 1.9056955594390812, 3.0784868372335454, -0.08202169697824342, 0.31621324542789037, 1.1596815031334955, 0.9101352017391915, 1.5474921902457461, 2.196340023711101, 0.9077932318260713, 0.09051885105090007, -1.2100517093034928, 0.6348069565298674, -0.09263859533483805, -0.21044490271986932, 3.1127664269469157]

So according to those coefficients of the uni-variate models and the multi-variate model, the coefficients of the uni-variate models failed to predict the coefficients in the multi-variate model.

About factors predict concrete compressive strength, according to those previous models, all the 8 quantitative input variables can predict concrete compressive strength to some extent.

To make the hardest possible concrete, through the entire data set, I found the largest value of the output feature is 82.6, and the combination of input variables of this particular output value is the following, and this combination may be able to make the hardest possible concrete.

Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5)(kg in a m^3 mixture)	Coarse Aggregate (component 6)(kg in a m^3 mixture)	Fine Aggregate (component 7)(kg in a m^3 mixture)	Age (day)
389.9	189	0	145.9	22	944.7	755.8	91