514A-Programming Assignment1

Zijie Liu 513053

October 2022

1 Introduction

1.1 1.1 Dataset

My dataset is from Concrete Compressive Strength dataset , we regard the first 8 columns as our X values and the last columns as our Y values.
For pre-processing our dataset, we use the Standardizing and Normalizing separately, all in the Prepocess.py file.

$$x_{Normalizing} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

$$x_{Standardizing} = \frac{x - x_{min}}{x_{std}} \tag{2}$$

Here is a comparison between raw data distribution and processed data distribution:

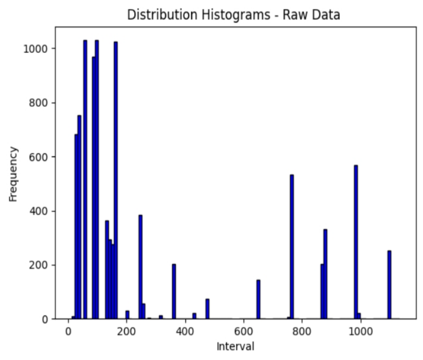


Figure 1: Raw Dataset

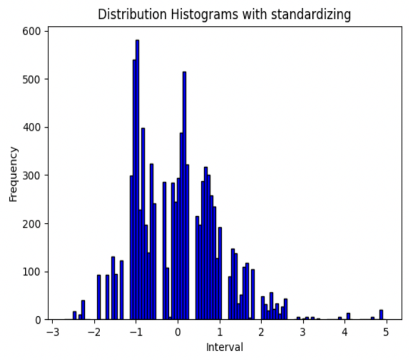


Figure 2: Standardizing Dataset

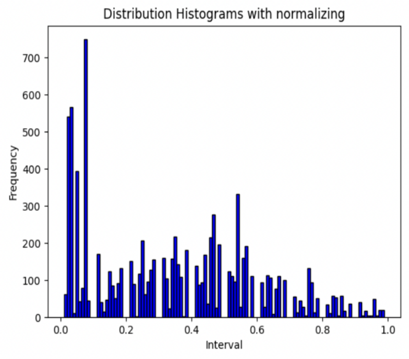
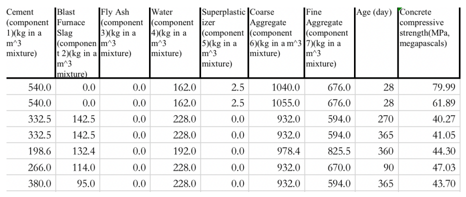


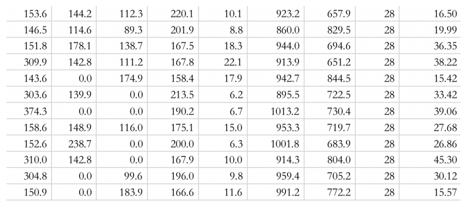
Figure 3: Normalizing Dataset

After that, we shuffle our dataset and splite them into Train dataset and Test dataset (9:1.3). Here is some sample of our train and test dataset:



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)	Concrete compressive strength(MPa, megapascals)
540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
266.0	114.0	0.0	228.0	0.0	932.0	670.0	90	47.03
380.0	95.0	0.0	228.0	0.0	932.0	594.0	365	43.70

Figure 4: Train Dataset



153.6	144.2	112.3	220.1	10.1	923.2	657.9	28	16.50
146.5	114.6	89.3	201.9	8.8	860.0	829.5	28	19.99
151.8	178.1	138.7	167.5	18.3	944.0	694.6	28	36.35
309.9	142.8	111.2	167.8	22.1	913.9	651.2	28	38.22
143.6	0.0	174.9	158.4	17.9	942.7	844.5	28	15.42
303.6	139.9	0.0	213.5	6.2	895.5	722.5	28	33.42
374.3	0.0	0.0	190.2	6.7	1013.2	730.4	28	39.06
158.6	148.9	116.0	175.1	15.0	953.3	719.7	28	27.68
152.6	238.7	0.0	200.0	6.3	1001.8	683.9	28	26.86
310.0	142.8	0.0	167.9	10.0	914.3	804.0	28	45.30
304.8	0.0	99.6	196.0	9.8	959.4	705.2	28	30.12
150.9	0.0	183.9	166.6	11.6	991.2	772.2	28	15.57

Figure 5: Test Dataset

1.2 Algorithm

For our Uni-Variant, here is eight feature in our x values, thus we use the for-loop to iterate those features and optimize their weights and bias, then combine them as a list. When optimizing them, we try both stochastic gradient decent and gradient decent, actually, the time cost will not have a huge difference because the limited parameters, but compared with optimize some data, just choose all errors will be more clear to show the decreasing of loss and them adjust hyper-parameters.
For our Multi-Variant, we use the matrix operation to calculate the predict values, we use the gradient decent as our optimization method.Here we combine the weights and bias in one matrix and add 1 in every X features.
For both Uni-Variant and Multi-Variant, I initialize the learning rate as 1e-7,because as normal, I first set the learning rate as 1e-5, but I found it's will occur gradient explosion thus I choose 1e-7, and then I use the learning rate decay, it means if the previous loss have the small difference with current loss, just down the learning rate by multiply 1e-1. More, when the previous loss equal to current, or the number of iteration attain 200000(this value can be changed in the config.py), we will stop the iteraion for this feature and then save the model.

Here is a sample for Uni-Variant Linear Regression and for the Multi-Variant, I use the matrix multiplication.

$$\boldsymbol{X} \times w + b = \boldsymbol{\hat{y}} \tag{3}$$

We both use MSE and MAE as our loss function separately, here is calculation step for each of them:
MSE:

$$Loss\ Function = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{4}$$

$$\frac{dL}{dw} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y_i}) \times X_i \tag{5}$$

$$\frac{dL}{db} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y_i}) \tag{6}$$

MAE:

$$Loss\ Function = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{7}$$

$$\frac{dL}{dw} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} -1 \times x_i & if\ y \geq \hat{y} \\ \frac{1}{n}\sum_{i=1}^{n} 1 \times x_i & if\ y < \hat{y} \end{cases} \tag{8}$$

$$\frac{dL}{db} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} -1 & if\ y \geq \hat{y} \\ \frac{1}{n}\sum_{i=1}^{n} 1 & if\ y < \hat{y} \end{cases} \tag{9}$$

1.3 Pseudo-code

Algorithm 1: Uni-Variant Linear Regression

```
    Data: train dataset $X$,$Y$, type $t$;
    Result: features parameters $W$,$B$, final response value $Y_{pre}$
 1  Initialize: $w \leftarrow 1$, $b \leftarrow 0$, $epoch \leftarrow 2e+7$, $lr \leftarrow 1e-7$, $loss_{pre} \leftarrow 0$;
 2  while $r \leq epoch$ and $loss_{pre} \neq loss$ do
 3      if $t = 1$ then
 4          loss = MSE();
 5          $partial_{weight} = \frac{\partial loss}{\partial weight}$;
 6          $w- = lr * partial_{weight}$;
 7          $partial_{bias} = \frac{dloss}{dbias}$;
 8          $b- = lr * partial_{bias}$;
 9      else if $t = 0$ then
10          loss = MAE();
11          $partial_{weight} = \frac{\partial loss}{\partial weight}$;
12          $w- = lr * partial_{weight}$;
13          $partial_{bias} = \frac{dloss}{dbias}$;
14          $b- = lr * partial_{bias}$;
15      if $loss_{pre} - loss \leq 1e-4$ then
16          $lr == lr * 1e-1$
17      $loss_{pre} \leftarrow loss$;
18      $r \leftarrow r+1$
19  end
```

Algorithm 2: Multi-Variant Linear Regression

```
    Data: train dataset $X$,$Y$, type $t$;
    Result: features parameters $W$,$B$, final response value $Y_{pre}$
 1  Initialize: $wb \leftarrow random$, $epoch \leftarrow 5e+5$, $loss_{pre} \leftarrow 0$, $lr \leftarrow 1e-7$;
 2  while $r \leq epoch$ do
 3      if $t = 1$ then
 4          loss = MSE();
 5          $partial_{w} = \frac{\partial loss}{\partial w}$;
 6          $wb- = lr * partial_{w}$;
 7      else if $t = 0$ then
 8          loss = MAE();
 9          $partial_{w} = \frac{\partial loss}{\partial w}$;
10          $wb- = lr * partial_{w}$;
11      if $loss_{pre} - loss \leq 1e-8$ then
12          $lr == lr * 1e-1$
13      $loss_{pre} \leftarrow loss$;
14      $r \leftarrow r+1$
15  end
```

2 Result

2.1 Variance explained for Uni-Variant

2.1.1 MSE

I have re-trained all of my model in Oct 12, all detailed information show in the ./Result/Text directory, there concludes all of weights, bias and R^2, I have generated.

features	weight	bias	R^2 train	R^2 test
feature1	0.12127397400050	0.16041198909187	0.9896430664100	0.9928602693495
feature2	0.21269186941583	1.890572147029	0.9980706712789	0.9869660150915
feature3	-0.028961245358351	37.53142554623	0.9975377358146	0.9831001467739
feature4	0.19190830401293	0.10028096202305	0.9970518269382	0.9836816450392
feature5	1.0640720236266	29.373535632244	0.9985955968068	0.989925284061
feature6	0.036521061672702	-0.000877140235222	0.9974164023425	0.981766710358
feature7	0.045745965296986	0.06639668501483	0.997415729083013	0.9814580639958
feature8	0.3282203763376	16.899242568763	0.9984694093022	0.9910851981305

features	norm weight	norm bias	norm R^2 train	norm R^2 test
feature1	0.3000000005651	0.2000000035170	0.9896430664100	0.9327263584994
feature2	0.30000000016763	0.200000004426	0.9870628199709	0.9092933917341
feature3	0.29999999855830	0.19999999832637	0.9852422735027	0.8985371855128
feature4	0.30000000073702	0.20000000263673	0.9835949297444	0.90130411997968
feature5	0.3000000004827	0.20000000375487	0.9888039630857	0.9192532268122
feature6	0.3000000001709	0.20000000028650	0.9848842861049	0.8919873953048
feature7	0.2999999998887	0.19999999983093	0.9846791410065	0.8998381139034
feature8	0.30000000030840	0.20000000415781	0.9879462743203	0.9397974063895

features	stand weight	stand bias	stand R^2 train	stand R^2 test
feature1	0.9520934712983	0.0015659900701958	0.9988099076240	0.9925147757129
feature2	0.9176837217986	-0.000881685835486	0.9978883470697	0.985772468211
feature3	0.8936105663080	0.0010715519138751	0.9972278555908	0.981060499304
feature4	0.87953761589981	0.0013774359342160	0.996625526133	0.98142670477298
feature5	0.9387112416709	-0.0001894213342220	0.9984990884202	0.9892499082757
feature6	0.88885961170349	0.0008592209669555	0.9970760776687	0.97940221219443
feature7	0.88922754888537	0.00128832757758611	0.99707337550737	0.97920210634888
feature8	0.93666906151332	0.000586482774245	0.9983567741686	0.99060340053408

2.1.2 MAE

features	weight	bias	R^2 train	R^2 test
feature1	0.11429264479159	0.9969960127901	0.9988702430171	0.9928609378417
feature2	0.16246776788421	7.123808666779	0.9980710148901	0.9869682822928
feature3	0.2059400453329	6.99586659994	0.9975382925115	0.9831038269138
feature4	0.17486156816913	0.9956894669135	0.997052621443	0.9836858234675
feature5	3.0547116821835	10.928910933397	0.9985957794304	0.9899265645342
feature6	0.03506756123456	0.9990184024691	0.9974170200389	0.9817710107557
feature7	0.042410442208643	0.9987705901234	0.99741633442448	0.9814620798956
feature8	0.35708511110920	18.639225044461	0.9984696155093	0.9910860775988

features	norm weight	norm bias	norm R^2 train	norm R^2 test
feature1	0.9949205336499	0.0021374621359222	0.9928286585596	0.9826597983044
feature2	0.9987598122504	0.027958840776700	0.9828511469068	0.9613853487017
feature3	0.9910656371901	0.01519854563106	0.9813544465463	0.9547244614979
feature4	0.9888368152631	-0.011902295145630	0.9810405409341	0.9579395910405
feature5	1.0036598927884	0.0346003922330	0.9846708453297	0.9602805011122
feature6	0.9875712220092	-0.010207761165044	0.9831768824060	0.9552593386904
feature7	0.9906346673337	-0.007285310679612	0.9838809702673	0.9541452685840
feature8	1.0025217930544	0.04449364077672	0.9778710811566	0.9532715939676

features	stand weight	stand bias	stand R^2 train	stand R^2 test
feature1	0.992352331912	3.208543689e-05	0.966093243708	0.919309080724
feature2	0.99046962961	0.0013890543	0.94194666158859	0.8548624037658
feature3	0.987208217026	-0.00041219417	0.926106095530	0.80866167012
feature4	0.986318462346	-0.002088001941747	0.91171562393	0.813795657266
feature5	0.992275374825	-6.9254368932e-05	0.957689438181	0.8891468247645
feature6	0.987554766590	-0.00099133786407	0.9221849944187	0.79825782287195
feature7	0.987926390	-0.001345906796116	0.92295835607	0.780888653548
feature8	0.996592526669	-0.000312217475	0.954993441210	0.885605064852

2.2 Variance explained for Multi-Variant

parameter	raw MAE	norm MAE	stand MAE
weight1	0.29301839	0.47249839	0.55141839
weight2	-0.20051532	0.38251861	0.03525374
weight3	0.70089084	0.16185356	0.02137758
weight4	-0.25002195	0.02019285	-0.07509269
weight5	-0.12692341	-0.43944048	-0.13911924
weight6	-0.00140425	0.21303529	0.26461744
weight7	0.18747376	-0.12647046	0.42295981
weight8	-0.29864238	-0.19347871	-0.28269915
bias	2.10059961	0.61542321	0.17966743
R^2 train	0.9919332641239	0.990080303701	0.9976698756889
R^2 test	0.991784544908815	0.975189956903	0.98402229339183

parameter	raw MSE	norm MSE	stand MSE
weight1	-0.08176004	-0.09657336	0.55141839
weight2	0.19065589	0.30868802	0.03525374
weight3	-0.16218132	-0.10811035	0.02137758
weight4	-0.31533922	-0.28043478	-0.07509269
weight5	0.01139219	0.14711726	-0.13911924
weight6	0.22610405	0.21203294	0.26461744
weight7	-0.15579702	-0.33198362	0.42295981
weight8	0.16608142	0.3464343	-0.28269915
bias	0.2932415]	0.53705839	0.17966743
R^2 train	0.9987929565630	0.99859978174542	0.9985997817454
R^2 test	0.9919332641239	0.9903727161523	0.99037271615235

2.3 Plot the data

In the last pages conclude all feature maps

3 Discussion

3.1

In Train dataset, if the number of iterations are same and not totally converge, the Uni-Variant Linear Model will have a higher accuracy, this is because, we have a small dataset, in every iteration, we only need to update one weight parameter and one bias parameter, it's more easier than update 8 weight parameters and 1 bias parameters, but I thought the Multi-Variant Linear Model will have a higher robustness.
For the same model, here are some differences between the Train dataset and Test dataset, I thought this is because I just pursuit the converge in the train

dataset but not consider the generalization ability of our Linear Model, especially for this simple model, and for the limited number of data, we hardly promise the generalization ability of our model, thus the accuracy in the Test dataset will be lower than in Train dataset. So, if a model can perform well in the train dataset not means it still work in the test dataset.
More, I found that when I pre-process data, like standardizing or normalizing the data before input in our model, the model will converge faster than the raw data, thus in the limited iteration, they will have a higher accuracy and perform well in the test dataset. Significantly, they have a higher R^2 , which those pre-processed data be easier to learn for a Linear Model.

3.2

For this two models, Uni-Variant Linear Model will have a higher accuracy in Train dataset and lower accuracy in Test dataset, the Multi-Variant Linear Model is reverse, this is because the Multi-Variant Linear Model is a more complex model than the Uni-Variant Linear Model, thus it needs more iteration to converage but can perform well in the out of Train dataset.
Actually, the reason why I takes the long time to train the Uni-Variant Linear Model is I use the for-loop when I iterate different features, I did use the matrix dot product operation to finish them, not the for-loop, it actually cost me a long time to wait for the results. Compared with the Multi-Variant Linear Model, which originally takes long time to train, but matrix multiply operation save lots of time.But in fact, the complex should take a long time to train compared with the simple model, thus theoretically the Uni-Variant did use the less time to train well.
I have try different hyper-parameters in those two kinds of Linear Regression Model, because the huge bias in our original dataset, so the initial learning rate need help the model quickly close to the optimized status, and I use the learning rate decay to adjust the step according to the specific situation of loss decreasing. More importantly, when we use the Standardized or Normalized Dataset, we need to initialize a small learning rate like 1e-7, cause those method decrease the gap in our dataset thus the data be more compacted, when we continue to use a large learning rate, it will quickly close to the optimized status but hardly to converge due to the large stride when gradient decent.

3.3

From our weight table, I think the feature2(Blast Furnace Slag (component 2)(kg in a m3 mixture)), feature5(Superplasticizer (component 5)(kg in a m3 mixture)), feature8(Age (day)) will influence a lot for predicting the Concrete compressive strength, this is because they all have a higher weight and R^2 scores, compared to other features.
I think, if you want to make the hardest possible concrete, you should improve those features proportion, then down others.

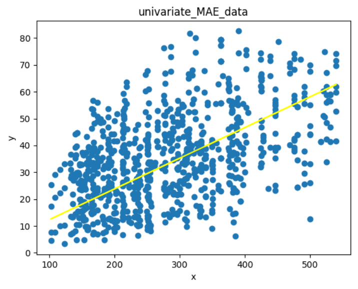


Figure 6: MAE Raw Feature 1

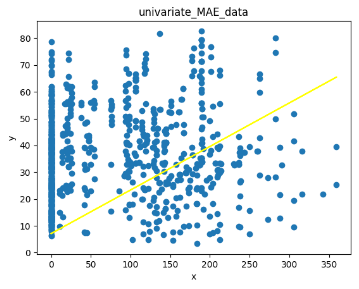


Figure 7: MAE Raw Feature 2

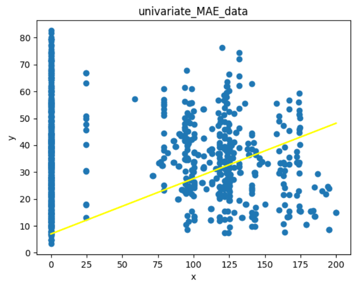


Figure 8: MAE Raw Feature 3

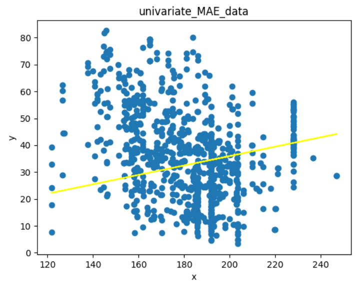


Figure 9: MAE Raw Feature 4

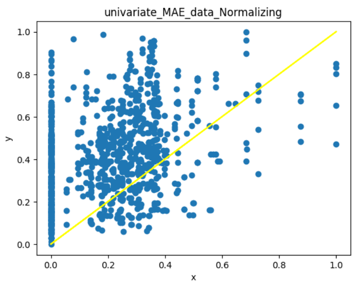


Figure 10: MAE Raw Feature 5

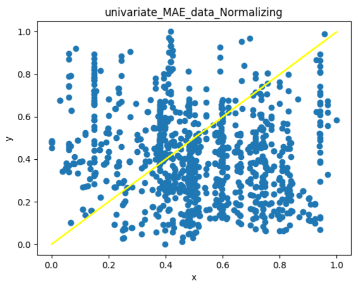


Figure 11: MAE Raw Feature 6

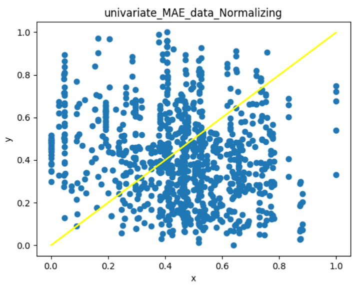


Figure 12: MAE Raw Feature 7

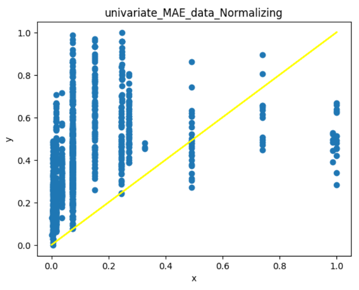


Figure 13: MAE Raw Feature 8

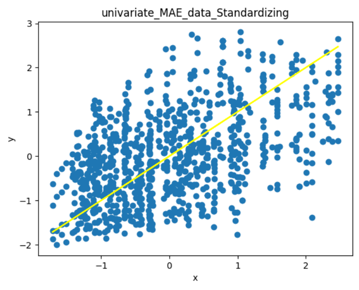


Figure 14: MAE Stand Feature 1

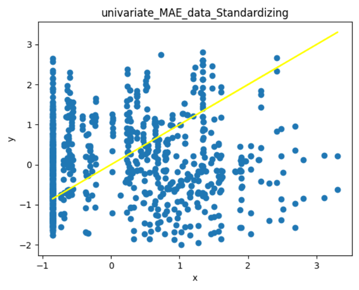


Figure 15: MAE Stand Feature 2

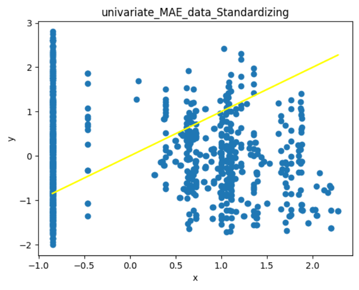


Figure 16: MAE Stand Feature 3

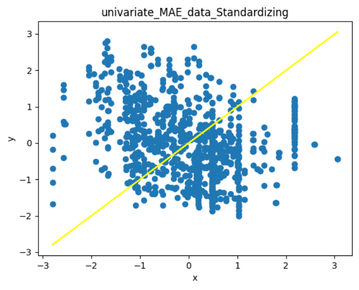


Figure 17: MAE Stand Feature 4

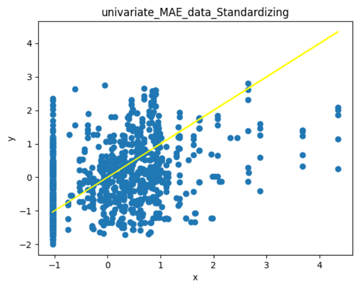


Figure 18: MAE Stand Feature 5

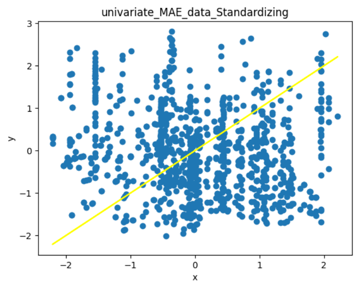


Figure 19: MAE Stand Feature 6

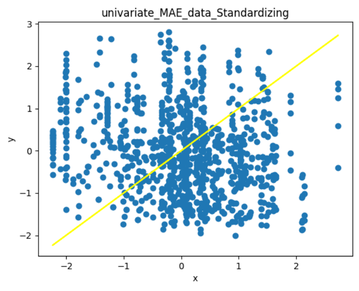


Figure 20: MAE Stand Feature 7

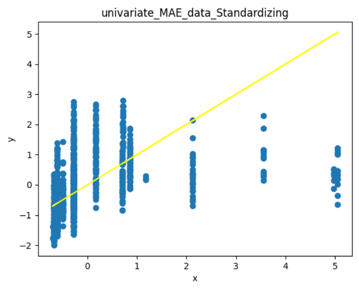


Figure 21: MAE Stand Feature 8

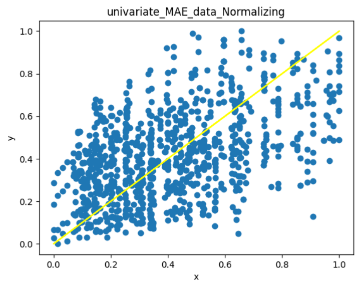


Figure 22: MAE Norm Feature 1

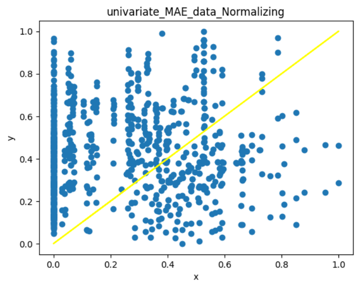


Figure 23: MAE Norm Feature 2

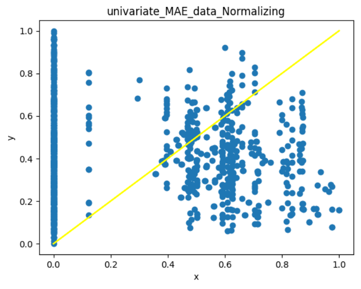


Figure 24: MAE Norm Feature 3

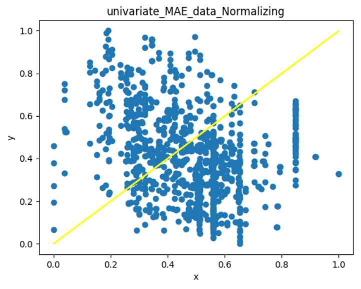


Figure 25: MAE Norm Feature 4

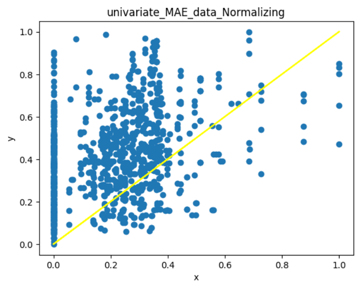


Figure 26: MAE Norm Feature 5

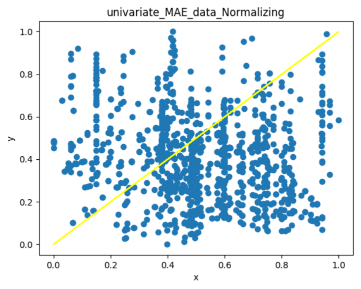


Figure 27: MAE Norm Feature 6

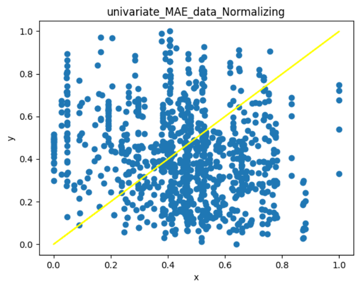


Figure 28: MAE Norm Feature 7

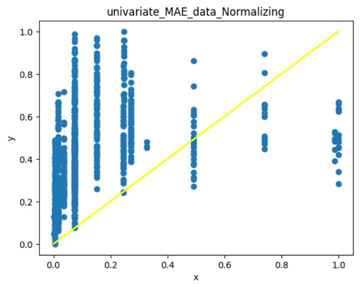


Figure 29: MAE Norm Feature 8

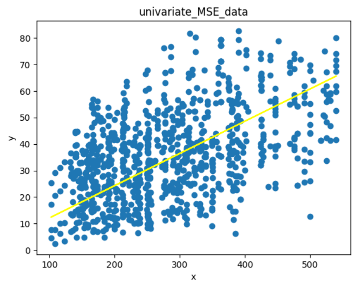


Figure 30: MSE Raw Feature 1

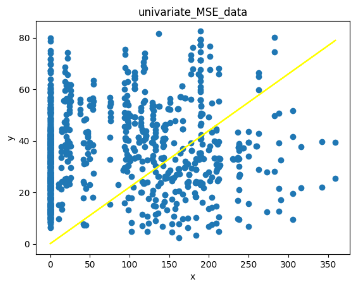


Figure 31: MSE Raw Feature 2

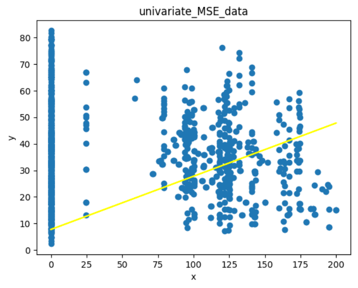


Figure 32: MSE Raw Feature 3

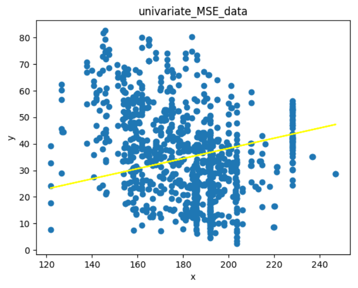


Figure 33: MSE Raw Feature 4

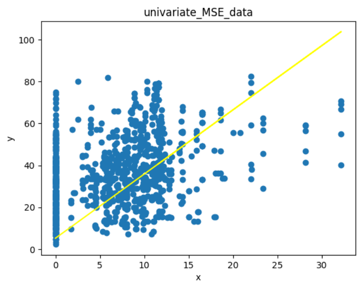


Figure 34: MSE Raw Feature 5

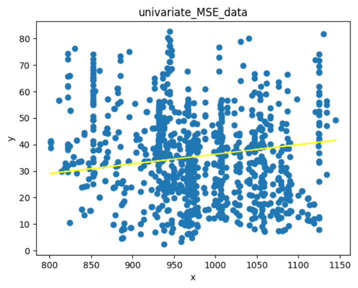


Figure 35: MSE Raw Feature 6

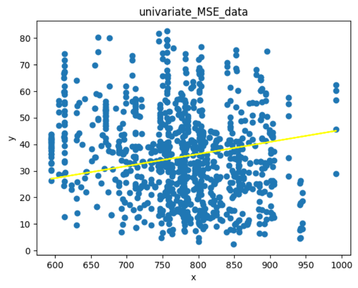


Figure 36: MSE Raw Feature 7

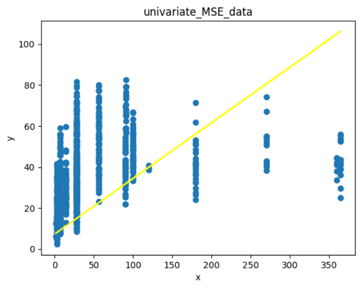


Figure 37: MSE Raw Feature 8

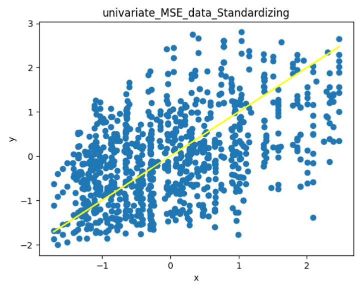


Figure 38: MSE Stand Feature 1

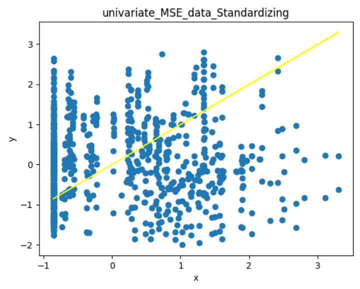


Figure 39: MSE Stand Feature 2

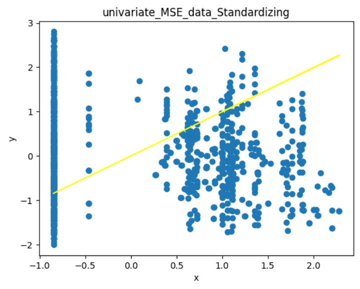


Figure 40: MSE Stand Feature 3

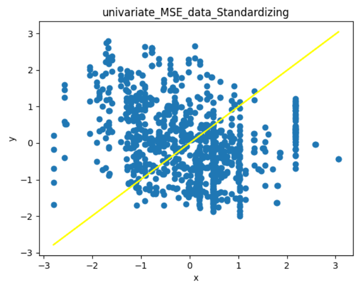


Figure 41: MSE Stand Feature 4

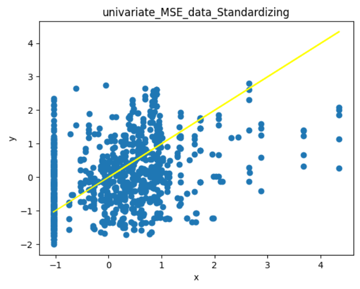


Figure 42: MSE Stand Feature 5

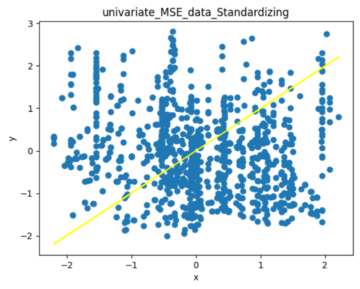


Figure 43: MSE Stand Feature 6

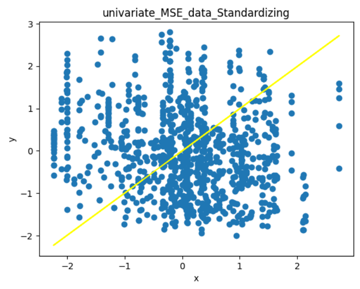


Figure 44: MSE Stand Feature 7

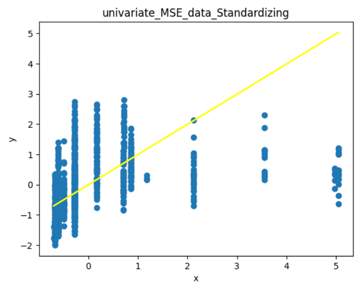


Figure 45: MSE Stand Feature 8

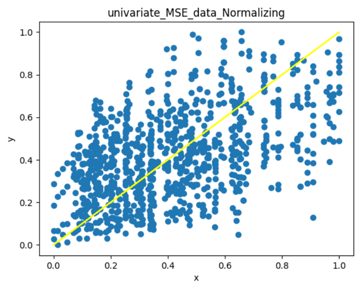


Figure 46: MSE Norm Feature 1

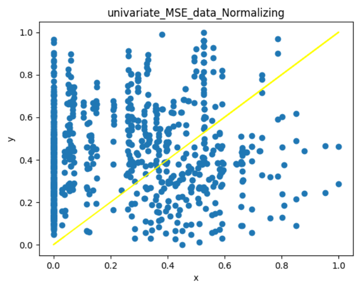


Figure 47: MSE Norm Feature 2

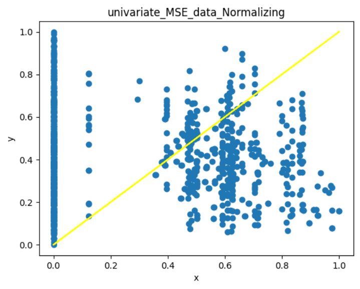


Figure 48: MSE Norm Feature 3

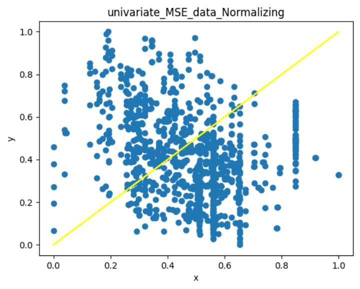


Figure 49: MSE Norm Feature 4

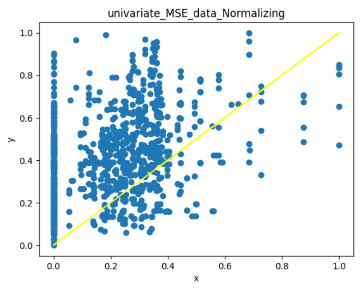


Figure 50: MSE Norm Feature 5

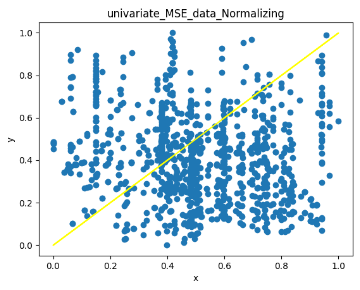


Figure 51: MSE Norm Feature 6

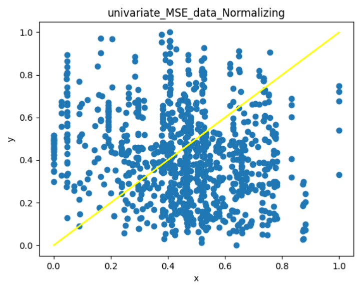


Figure 52: MSE Norm Feature 7

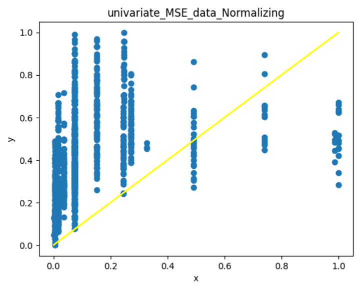


Figure 53: MSE Norm Feature 8