1. Problem Statement

1.1 Single variable regression

In this question:

1.1.1 First it needs to visualize the data and give a glance of it distribution.

1.1.2 It required us to using mathematical methods to achieve linear regression. And by applying training data and testing data, we could get the result from training set and testing set. It could be used to calculate the training error and testing error.

1.1.3 Applying python build in method to achieve linear regression and compare the performance between the mathematical method and build in method.

1.1.4 Applying mathematical methods to build polynomial model. By changing the parameter Order the model will have different performance and find the relatively best one.

1.1.5 Find the relationship between training set size and model accuracy.

1.2 Multivariate regression

In this question:

1.2.1 Using mathematical method to map the origin data into higher dimension by using combination of features.

1.2.2 Applying multivariate linear regression method to this higher dimension data and evaluate it performance by using mean square error.

1.2.3 By using matrix operation to calculate the theta and using iterative to do the same thing (gradient descent). And compare the theta and performance between these two methods.

1.2.4 Applying Gaussian kernel function regression such as the mapping function and compare the time and accuracy (precision-recall).

2 Proposed solution

2.1 First question basically using linear regression method.

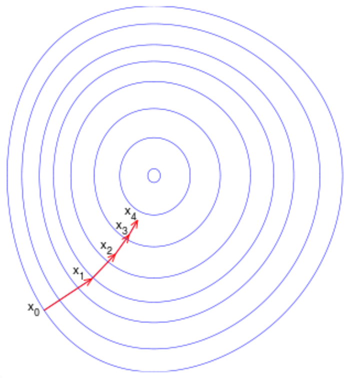
$$\boldsymbol{Y \sim N(a+bx,\sigma^2)}$$
$$\boldsymbol{Y = a+bx+\varepsilon,\quad \varepsilon \sim N(0,\sigma^2)}$$

linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).[1] The very simplest case of a single scalar predictor variable x and a single scalar response variable y is known as simple linear regression. The extension to multiple and/or vector-valued predictor variables (denoted with a capital X) is known as multiple linear regression, also known as multivariable linear regression. Nearly all real-world regression models involve multiple predictors, and basic descriptions of linear regression are often phrased in terms of the multiple regression model. Note, however, that in these cases the response variable y is still a scalar. Another term, multivariate linear regression, refers to cases where y is a vector, i.e., the same as general linear regression. [2]

2.2 Second question using relatively complex model. It contain kernel method: kernel methods are a class of algorithms for pattern analysis, Kernel methods can be thought of as instance-based learners: rather than learning some fixed set of parameters corresponding to the features of their inputs, they instead "remember" the *i*-th training example (x_i,y_i) and learn for it a corresponding weight w_i. Prediction for unlabeled inputs, i.e., those not in the training set, is treated by the application of a similarity function *k*, called a kernel, between the unlabeled input x' and each of the training inputs x_i.

$$\hat{y} = \operatorname{sgn} \sum_{i=1}^{n} w_i y_i k(\mathbf{x}_i, \mathbf{x}'),$$

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes step s proportional to the negative of the gradient (or approximate gradient) of the



function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent.

The Gaussian kernel is defined in 1-D, 2D and N-D respectively as:

$$G_{1D}(x;\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}},\ G_{2D}(x, y;\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},\ G_{ND}(\vec{x};\sigma) = \frac{1}{\left(\sqrt{2\pi}\,\sigma\right)^N} e^{-\frac{|\vec{x}|^2}{2\sigma^2}}$$

The theta determines the width of the Gaussian kernel.

3 Implementation details

3.1 Single variable regression

Read the data from folder to array. Applying different size of data user should change the reading array size. And applying NumPy method train_test_split method to split data randomly into two sets, training and testing. User could change the rate of the training data/ testing data which could be used in the question 1.e. Then convert the data in to matrix, applying mathematical methods (matrix operation) to calculate theta, mention here, KX should know be 0. Using the training data and theta to calculate the result of training data. Applying in the same way of testing data. Here it gives the error of every individual data and the average error of training set and testing set.

Applying python build in method to calculate the linear regression.

Applying polynomial models to data by manual set the order and using iterative way to calculate the coefficient matrix, plot the curve. Here, if we want to have bigger order, user should change the poly_mat manually, because it should match the order of the polynomial.

Problem:

Hard to split data randomly ---- using NumPy method train_test_split method.

Calculate different order's polynomial model ---- using iterative way to calculate the coefficient matrix.

3.2 Multivariate regression

Read the data from folder to array. Applying different size of data user should change the reading array size. And applying NumPy method train_test_split method to split data randomly into two sets, training and testing. User could change the rate of the training data/ testing data.

Using combination of features by manually build the matrix, if user wants to apply different dimension of data user should rebuild the mapping function.

Applying theta function to calculate theta in calculate the Gram matrix. Using

$$\hat{\beta}_{(i)} = (X'X)^{-1}X'\mathbf{y}_{(i)}$$

the training data and theta to calculate the result of training data. Applying in the same way of testing data (data which used should have same dimension as the theta). Calculate the RSE.

Using the linear regression mathematics method to calculate the theta of linear regression and apply to both the training set and testing set. Calculate the RSE.

Applying stochastic gradient descent method to iterative calculate the theta.

```
for $i = 1$ to $m$ :
    $\theta_j = \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$
```

Applying gaussian 2 dimentional kernel method to calculate the model of theta.

Problem:

Combination of features ---- build the new data manually, but when the dimension of origin data is changing the matrix need be rebuilt.

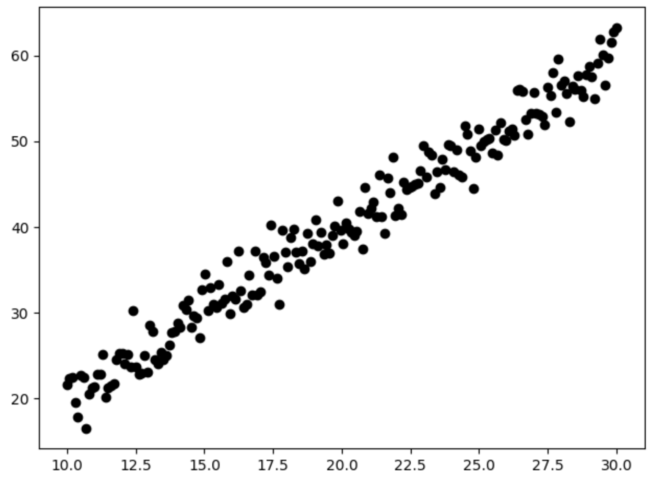
Iterative solution to regression problem ---- using gradient decent method (stochastic gradient descent method) by iterative calculation of the theta’s inverse.

Don’t know how to using Gaussian kernel function ---- find a kernel function in OpenCV method and trying to apply by myself.

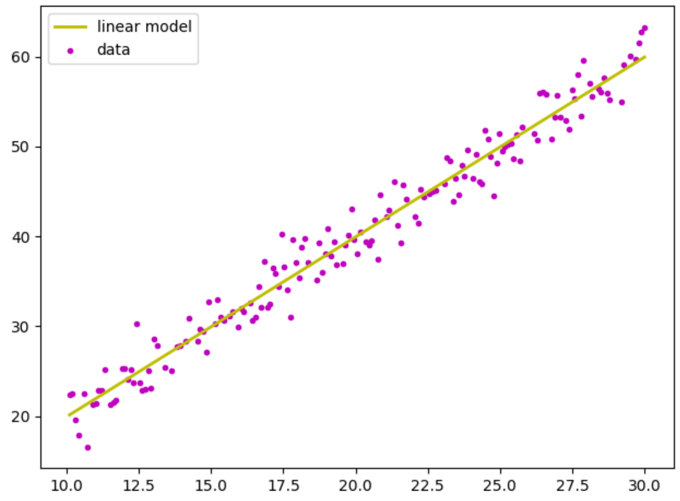
4 Result and discussion

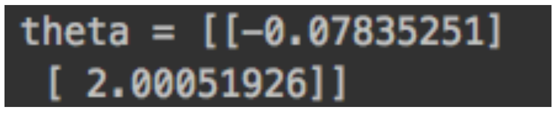
4.1 Single variable regression (using svar-set1.txt as example)

(a) Plot the data.



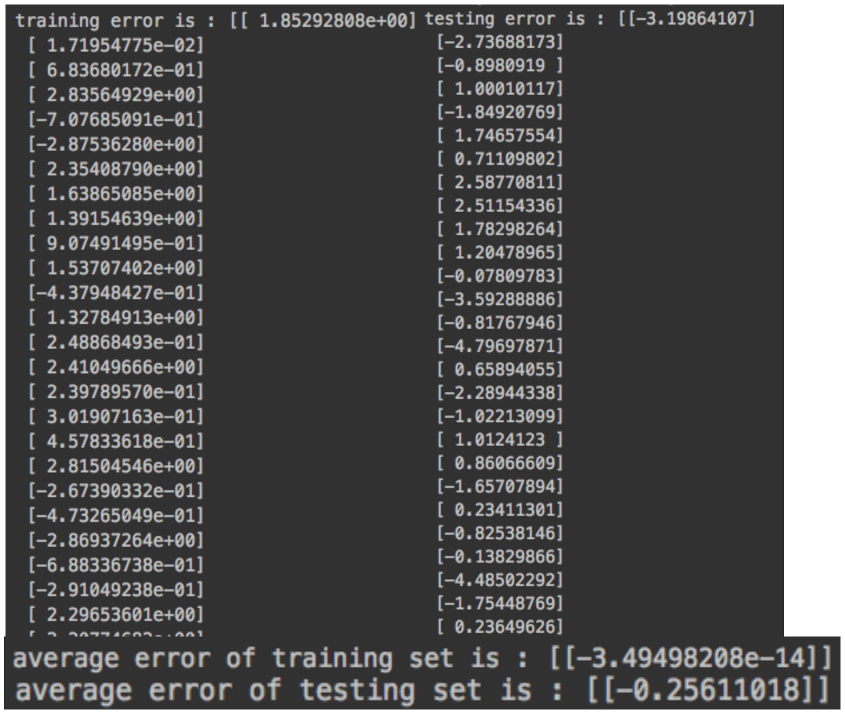
(b) By applying the training set with the theta, we have the linear regression curve in the plot.





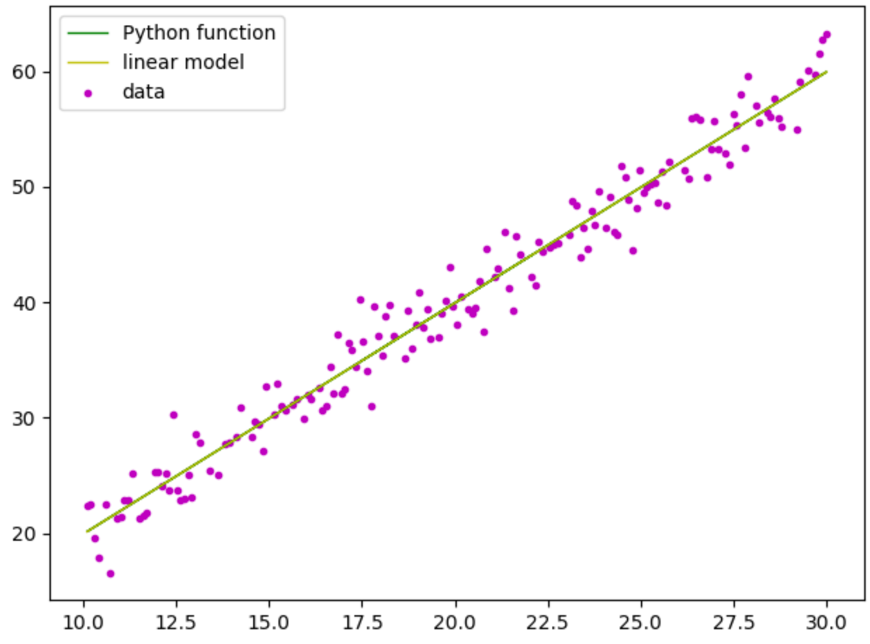
```
theta = [[-0.07835251]
 [ 2.00051926]]
```

The training error and testing error are:

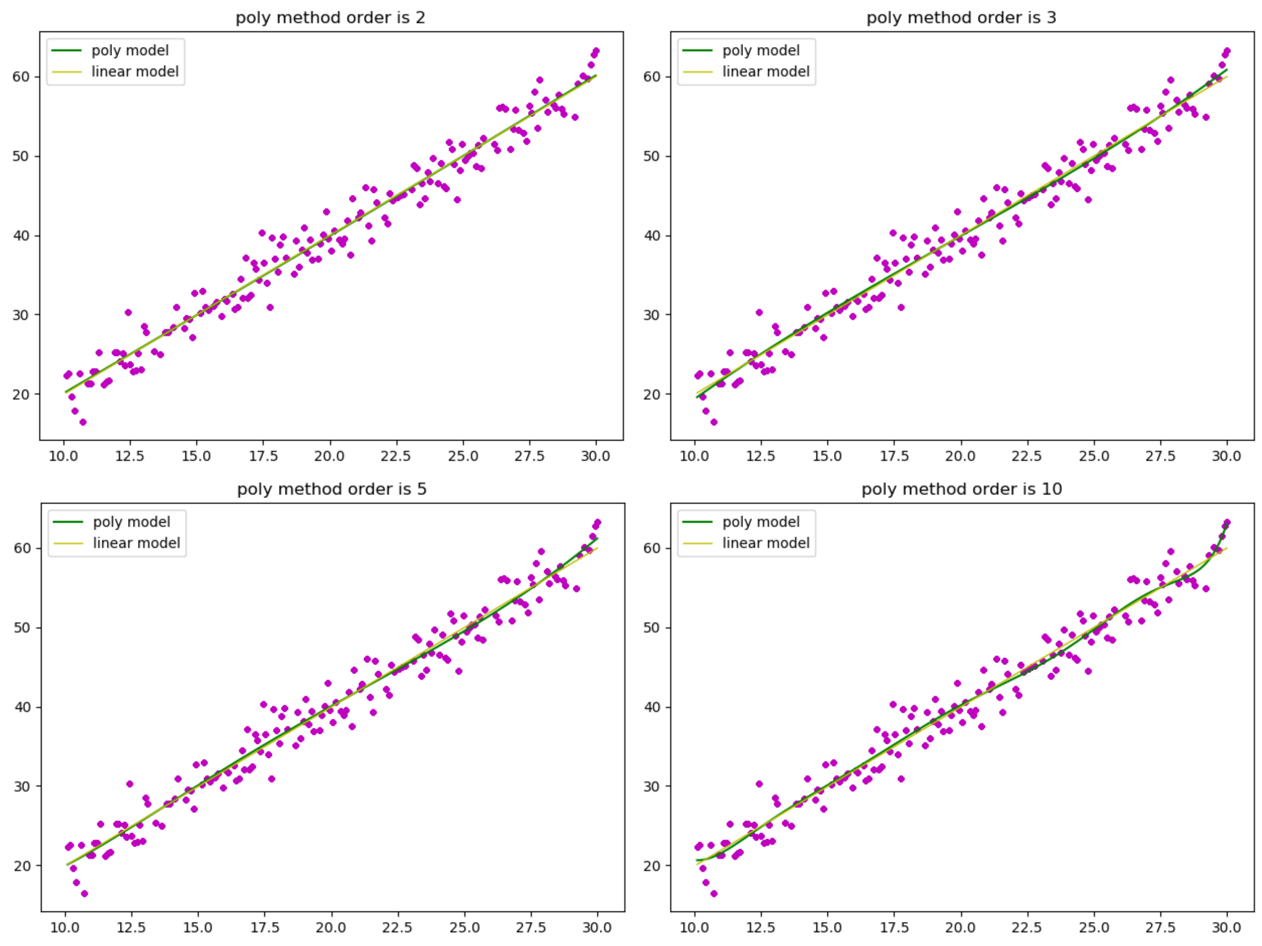


```
training error is : [[ 1.85292808e+00] testing error is : [[-3.19864107]
 [ 1.71954775e-02]                      [-2.73688173]
 [ 6.83680172e-01]                      [-0.8980919 ]
 [ 2.83564929e+00]                      [ 1.00010117]
 [-7.07685091e-01]                      [-1.84920769]
 [-2.87536280e+00]                      [ 1.74657554]
 [ 2.35408790e+00]                      [ 0.71109802]
 [ 1.63865085e+00]                      [ 2.58770811]
 [ 1.39154639e+00]                      [ 2.51154336]
 [ 9.07491495e-01]                      [ 1.78298264]
 [ 1.53707402e+00]                      [ 1.20478965]
 [-4.37948427e-01]                      [-0.07809783]
 [ 1.32784913e+00]                      [-3.59288886]
 [ 2.48868493e-01]                      [-0.81767946]
 [ 2.41049666e+00]                      [-4.79697871]
 [ 2.39789570e-01]                      [ 0.65894055]
 [ 3.01907163e-01]                      [-2.28944338]
 [ 4.57833618e-01]                      [-1.02213099]
 [ 2.81504546e+00]                      [ 1.0124123 ]
 [-2.67390332e-01]                      [ 0.86066609]
 [-4.73265049e-01]                      [-1.65707894]
 [-2.86937264e+00]                      [ 0.23411301]
 [-6.88336738e-01]                      [-0.82538146]
 [-2.91049238e-01]                      [-0.13829866]
 [ 2.29653601e+00]                      [-4.48502292]
                                        [-1.75448769]
                                        [ 0.23649626]
average error of training set is : [[-3.49498208e-14]]
average error of testing set is : [[-0.25611018]]
```

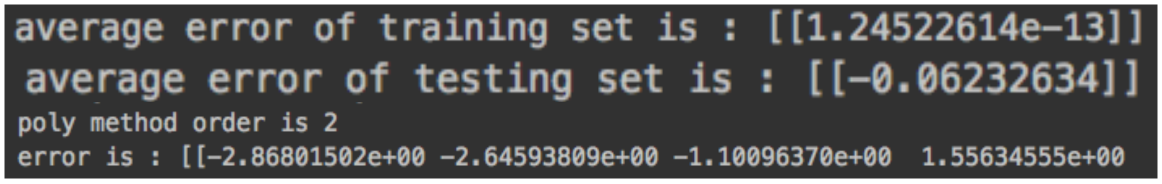
(c) Comparing to the python function there seems no big difference between these two linear regressions.



(d) As the order of the polynomial rises, the fitting situation increases significantly, but as the order increases excessively, over-fitting occurs and the calculation time becomes longer.

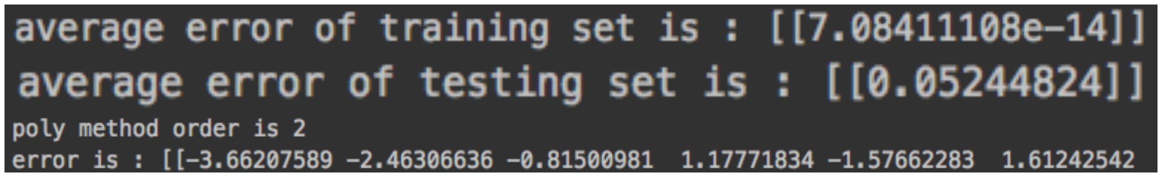


(e) When there are 50-50 of training and testing:



```
average error of training set is : [[1.24522614e-13]]
average error of testing set is : [[-0.06232634]]
poly method order is 2
error is : [[-2.86801502e+00 -2.64593809e+00 -1.10096370e+00  1.55634555e+00
```

When there are 25-75 of training and testing:

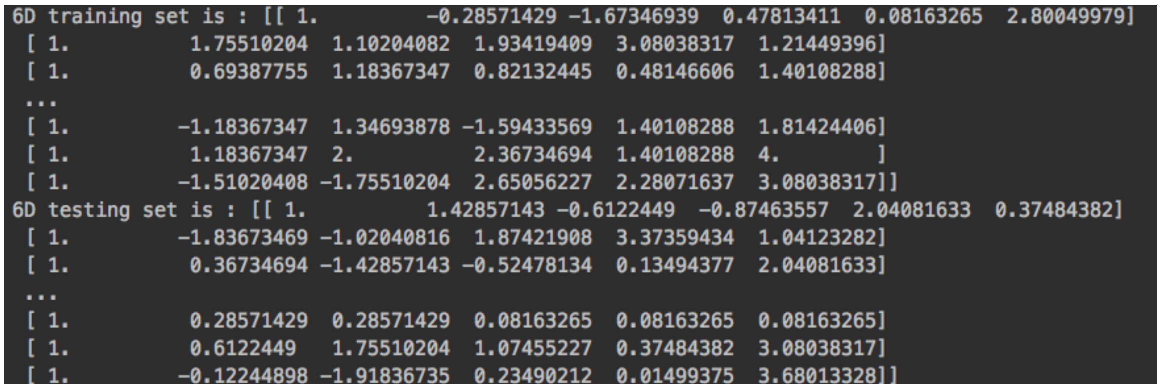


```
average error of training set is : [[7.08411108e-14]]
average error of testing set is : [[0.05244824]]
poly method order is 2
error is : [[-3.66207589 -2.46306636 -0.81500981  1.17771834 -1.57662283  1.61242542
```

Both of the training error and testing error of linear regression and polynomial regression is increading.

4.2 Multivariate regression

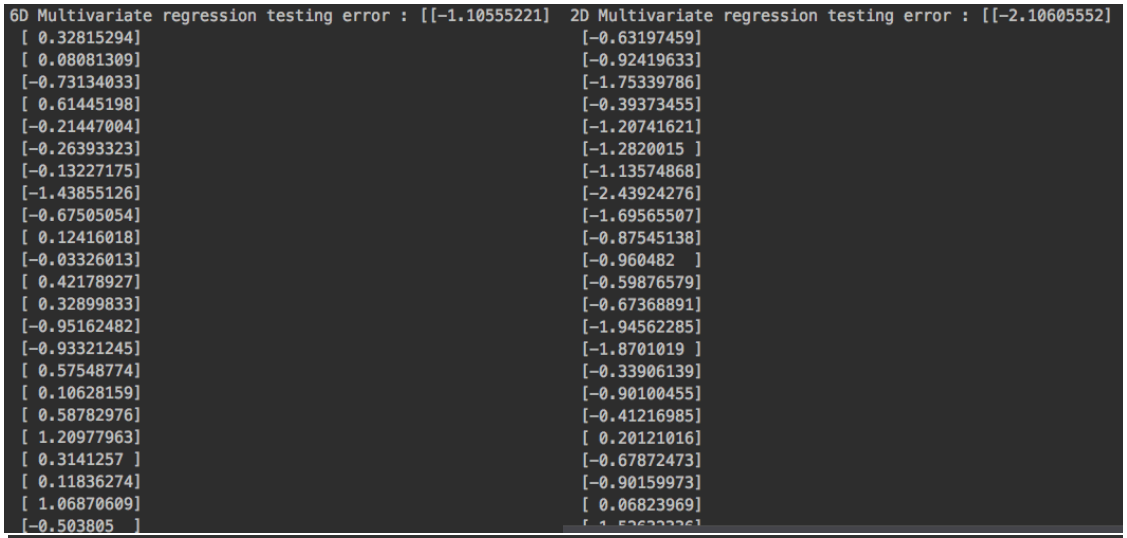
(a) Combination feature to higher dimension



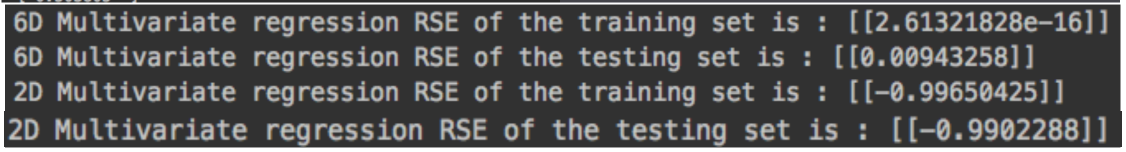
```
6D training set is : [[ 1.         -0.28571429 -1.67346939  0.47813411  0.08163265  2.80049979]
 [ 1.          1.75510204  1.10204082  1.93419409  3.08038317  1.21449396]
 [ 1.          0.69387755  1.18367347  0.82132445  0.48146606  1.40108288]
 ...
 [ 1.         -1.18367347  1.34693878 -1.59433569  1.40108288  1.81424406]
 [ 1.          1.18367347  2.          2.36734694  1.40108288  4.        ]
 [ 1.         -1.51020408 -1.75510204  2.65056227  2.28071637  3.08038317]]
6D testing set is : [[ 1.          1.42857143 -0.6122449  -0.87463557  2.04081633  0.37484382]
 [ 1.         -1.83673469 -1.02040816  1.87421908  3.37359434  1.04123282]
 [ 1.          0.36734694 -1.42857143 -0.52478134  0.13494377  2.04081633]
 ...
 [ 1.          0.28571429  0.28571429  0.08163265  0.08163265  0.08163265]
 [ 1.          0.6122449   1.75510204  1.07455227  0.37484382  3.08038317]
 [ 1.         -0.12244898 -1.91836735  0.23490212  0.01499375  3.68013328]]
```

(b) Performance of 2D and 6D:

In higher dimension the more variant of data feature it could explain more implicit information between more data.

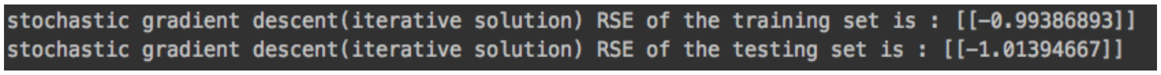


```
6D Multivariate regression testing error : [[-1.10555221]  2D Multivariate regression testing error : [[-2.10605552]
 [ 0.32815294]                                             [-0.63197459]
 [ 0.08081309]                                             [-0.92419633]
 [-0.73134033]                                             [-1.75339786]
 [ 0.61445198]                                             [-0.39373455]
 [-0.21447004]                                             [-1.20741621]
 [-0.26393323]                                             [-1.2820015 ]
 [-0.13227175]                                             [-1.13574868]
 [-1.43855126]                                             [-2.43924276]
 [-0.67505054]                                             [-1.69565507]
 [ 0.12416018]                                             [-0.87545138]
 [-0.03326013]                                             [-0.960482  ]
 [ 0.42178927]                                             [-0.59876579]
 [ 0.32899833]                                             [-0.67368891]
 [-0.95162482]                                             [-1.94562285]
 [-0.93321245]                                             [-1.8701019 ]
 [ 0.57548774]                                             [-0.33906139]
 [ 0.10628159]                                             [-0.90100455]
 [ 0.58782976]                                             [-0.41216985]
 [ 1.20977963]                                             [ 0.20121016]
 [ 0.3141257 ]                                             [-0.67872473]
 [ 0.11836274]                                             [-0.90159973]
 [ 1.06870609]                                             [ 0.06823969]
 [-0.503805  ]
```

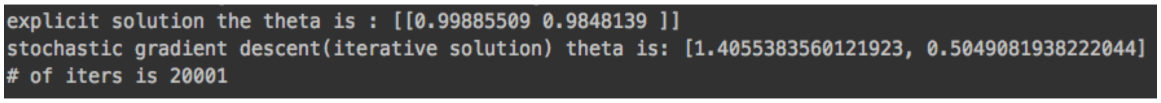


```
6D Multivariate regression RSE of the training set is : [[2.61321828e-16]]
6D Multivariate regression RSE of the testing set is : [[0.00943258]]
2D Multivariate regression RSE of the training set is : [[-0.99650425]]
2D Multivariate regression RSE of the testing set is : [[-0.9902288]]
```

(c) this method have relatively higher RSE in both testing and training sets.



```
stochastic gradient descent(iterative solution) RSE of the training set is : [[-0.99386893]]
stochastic gradient descent(iterative solution) RSE of the testing set is : [[-1.01394667]]
```



```
explicit solution the theta is : [[0.99885509 0.9848139 ]]
stochastic gradient descent(iterative solution) theta is: [1.4055383560121923, 0.5049081938222044]
# of iters is 20001
```

5 Reference

[1]wikipedia.Linearregression[EB/OL].https://en.wikipedia.org/wiki/Linear_regression#Simple_and_multiple_linear_regression.

[2]wikipedia.Kernel method[EB/OL]. https://en.wikipedia.org/wiki/Kernel_method.

[3]wikipedia.Gradientdescent[EB/OL].https://en.wikipedia.org/wiki/Gradient_descent.