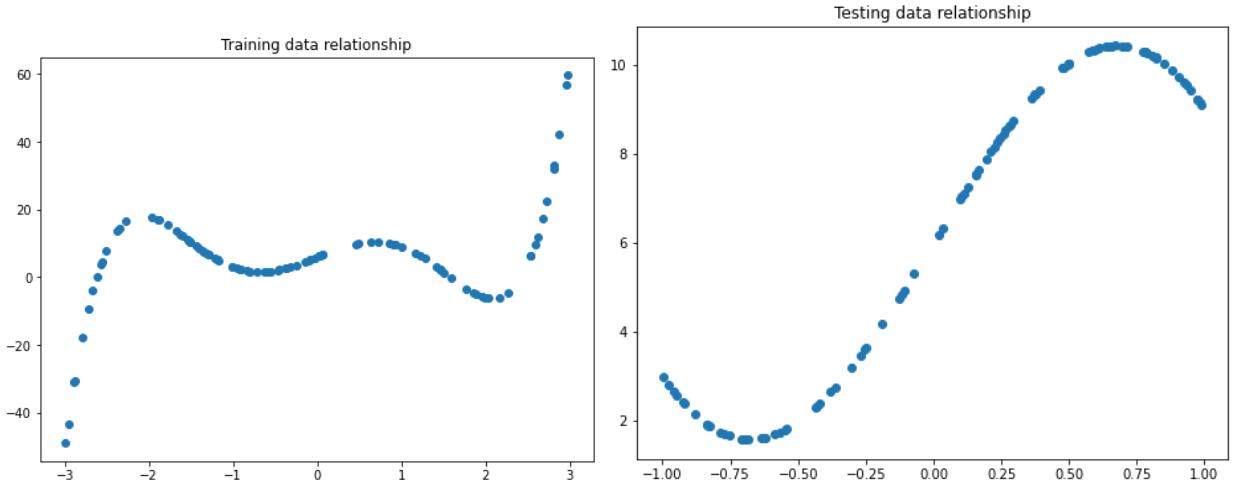
HIEU NGHIEM
CS5783
Assignment 1 Report

Question 1

1. Plot the relationship between training and testing data



2. The plots indicate that the relationship between X and Y for both training data and testing data is not linear. The relationship pattern between training and testing data are also not similar. Hence, we can try some approaches below:
   - Feature engineering: normalize the variable by subtracting every data point to mean(X) and divided by standard deviation of X.
   - Add polynomial features like x^2 or x^3

   Now we will try to implement a linear regression model based on gradient descent method. The parameters are chosen will be 0.001 for learning rate and 1000 iteration. For this step, I will:

   - Define cost function - using formula on P16 Lecture 3. This function will return the cost and the prediction error for every theta

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

$$h(x) = \sum_{i=0}^{n}\theta_i x_i = \theta^T x,$$

```
def cost_function(x,y,t):
    cost = np.dot(np.dot(x, t.T) - y, np.dot(x, t.T) - y) * (1/2)
    error = np.dot(x, t.T) - y
    return cost, error
```

- Define gradient descent function - using update rule on P23 Lecture 3. This function will return theta after every iteration

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

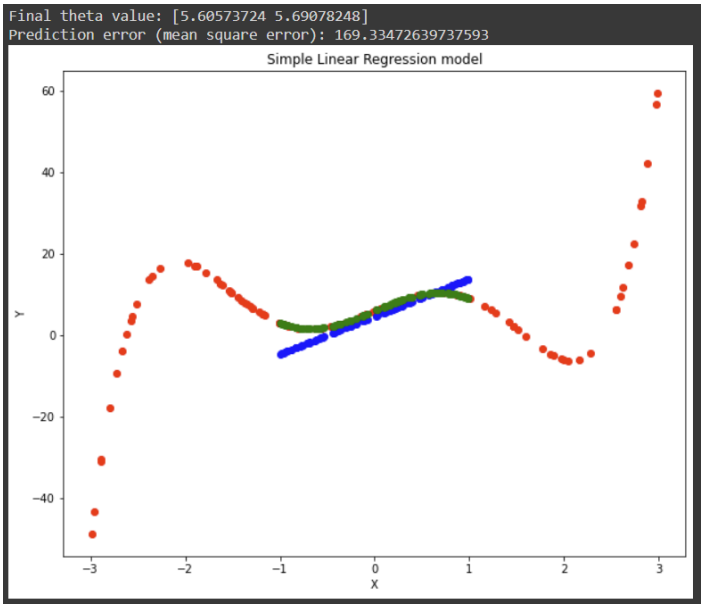
```
def gradient_descent(x,y,t,a, iteration):
    for i in range(iteration):
        c, e = cost_function(x, y, t)
        t = t - (a  * np.dot(x.T, e))
    return t
```

- Define prediction function - simply return y_predicted with values of x and theta

```
def prediction(x,t):
    y_predicted = np.dot(x, t.T)
    return y_predicted
```

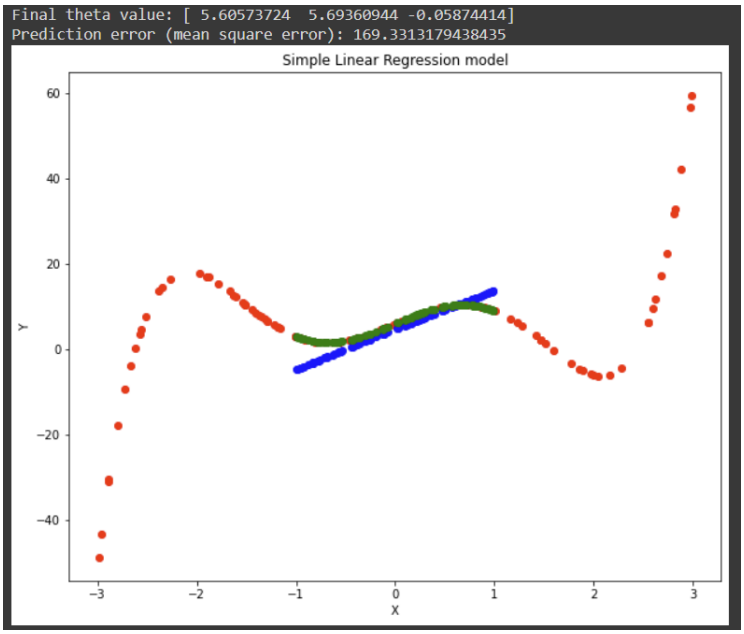
After importing training data, we will run the gradient descent in training data to find prediction equation and apply prediction function on the testing data. We have prediction model Y = 5.60573724 + 5.69078248 * X. Prediction error (use mean square error method) is 169.33

The red scatter points are training data, green scatter points are testing data and the prediction line is blue.

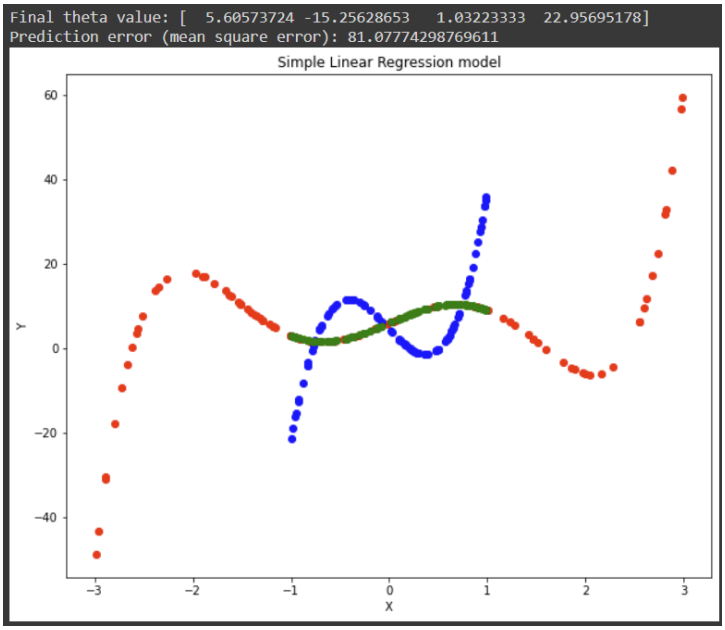


Now we will add the polynomial features for the non-linearity by using basis function X^2 and X^3 and build the model again.

With X^2, the prediction equation is Y = (5.60573724 + 5.69360944 * X – 0.05874414 * X^2) with prediction error of 169.33. You can see that the weight of X^2 is quite small and the error is the same as previous model so the quadratic feature does not contribute much to the model.



With X^3, the prediction equation is Y = (5.60573724 -15.25628653 * X + 1.03223333 * X^2 + 22.95695178 X^3) with prediction error is 81 which is significantly lower than previous models. You can see that the weight of all quadratic and cubic features contribute to the model.



Adding basis functions reduce the prediction error, however if we add higher terms on prediction function, overfitting may happen.

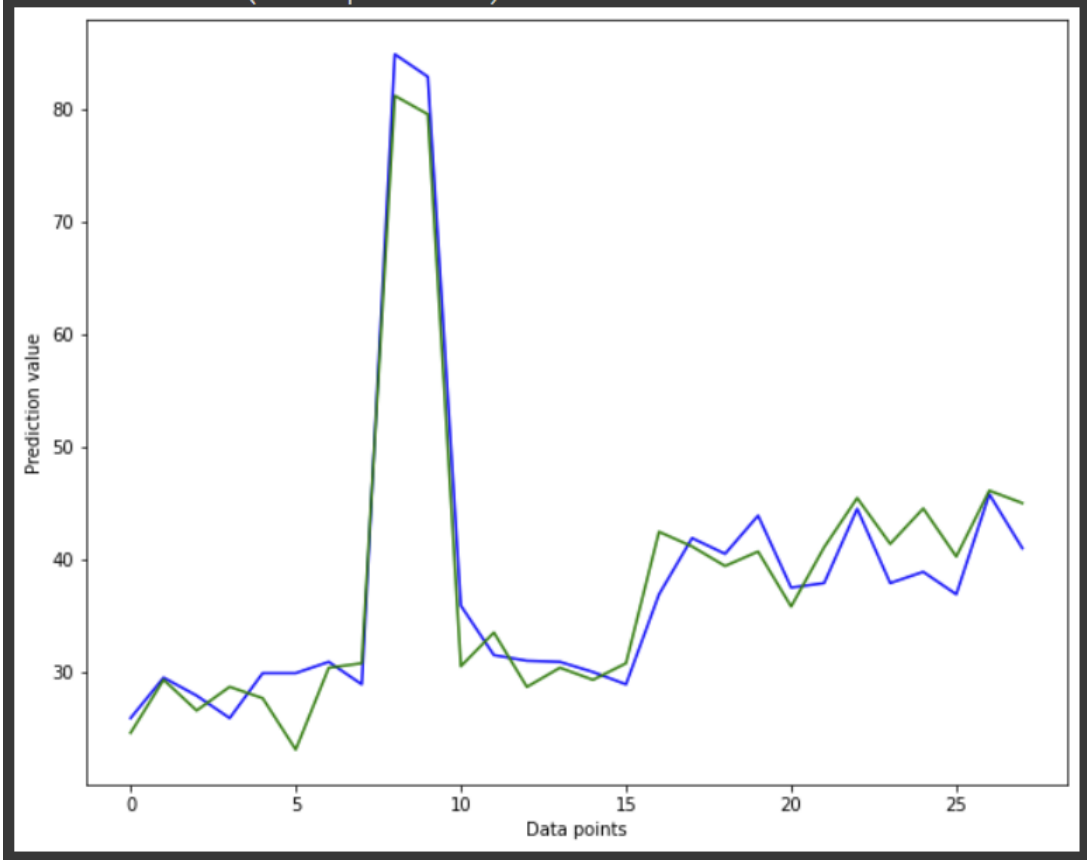
Question 2

1. Using simple linear regression model implementation with no polynomial basis function in Q1 with input variables as below columns:

```
X = data[['Local Price', 'Bathrooms', 'Land Area', 'Living area',
       '# Garages', '# Rooms', '# Bedrooms', 'Age of home',
       'Construction type', 'Architecture type', '# Fire places']]
```

There are only 28 records of data, so I will try to build a model with all data points as training set and predicting on the same set of data. That's not a good practice but there is no testing dataset so that's what I can do, and the performance of the model may not be reflected correctly. The corresponding theta values for every feature are shown in below figures, and the mean square error (or average square error) is 9.35 for this model. The blue line indicates the real y value and green line indicates the predicted y value.

```
Final theta value: [38.15714286  2.60936371  4.18877713  0.50473755  6.69492157  1.15834822
 -0.89176102 -0.34147193 -0.9178142   1.21154324  0.80110827  1.22174215]
Prediction error (mean square error): 9.352750812533968
```



Now, we find correlation between X and predict_Y and Y to see what the most important features and least important features are. The higher correlation between X and Y, the more important this feature is and vice versa.

```
Correlation between predict_y and Local Price 0.7506631299734747
Correlation between predict_y and Bathrooms 0.7230169050866526
Correlation between predict_y and Land Area 0.49534349948783823
Correlation between predict_y and Living area 0.7101088944026925
Correlation between predict_y and # Garages 0.4136395257687919
Correlation between predict_y and # Rooms 0.5093687109152628
Correlation between predict_y and # Bedrooms 0.4428119602683544
Correlation between predict_y and Age of home -0.25139634595843136
Correlation between predict_y and Construction type 0.03994105363369855
Correlation between predict_y and Architecture type -0.020601048104984192
Correlation between predict_y and # Fire places 0.3741578617526924


correlation between y and Local Price 0.7583613569718661
correlation between y and Bathrooms 0.680784170118376
correlation between y and Land Area 0.487937410382764
correlation between y and Living area 0.49264118839353266
correlation between y and # Garages 0.4324258551098054
correlation between y and # Rooms 0.4267862228519455
correlation between y and # Bedrooms 0.3091685533418898
correlation between y and Age of home -0.3047847549545271
correlation between y and Construction type 0.05258044913676442
correlation between y and Architecture type 0.08814089405208615
correlation between y and # Fire places 0.4044181578593355
```

2. As in the correlation figure above, Local Price and Bathrooms are the most important features. I use the same model implementation with only two variables as input – the prediction error is double compared to the model with all variables:

```
Final theta value: [38.15714286  6.84236895  7.06409294]
Prediction error (mean square error): 17.82024292906855
```

To answer the question, yes, we can absolutely use only these features on predict the house price, but the model may have a higher prediction error. It is because only these features may not reflect the house price correctly.

3. As in the correlation figure, least important features are Construction type and Architecture type. Using the same implementation with these variables removed – we have the prediction error slightly increase compared to the model with all variables, but the difference seems to be negligible. Performance may reduce but it's in acceptable range. That makes sense because it will happen when any factor contributes to the house price are removed from the model, regardless of how important they are.

```
Final theta value: [38.15714286  3.58987729  3.82409656  0.22903811  6.45282679  1.07998927
 -0.39679781 -0.78556299 -0.94276587  0.9789259 ]
Prediction error (mean square error): 10.819509639599946
```

Question 3

Implementing a locally weighted linear regression model using Normal Equation approach. We are going to define two functions: one is for building the weight matrix W which are diagonal, and the elements are computed by the below equation:

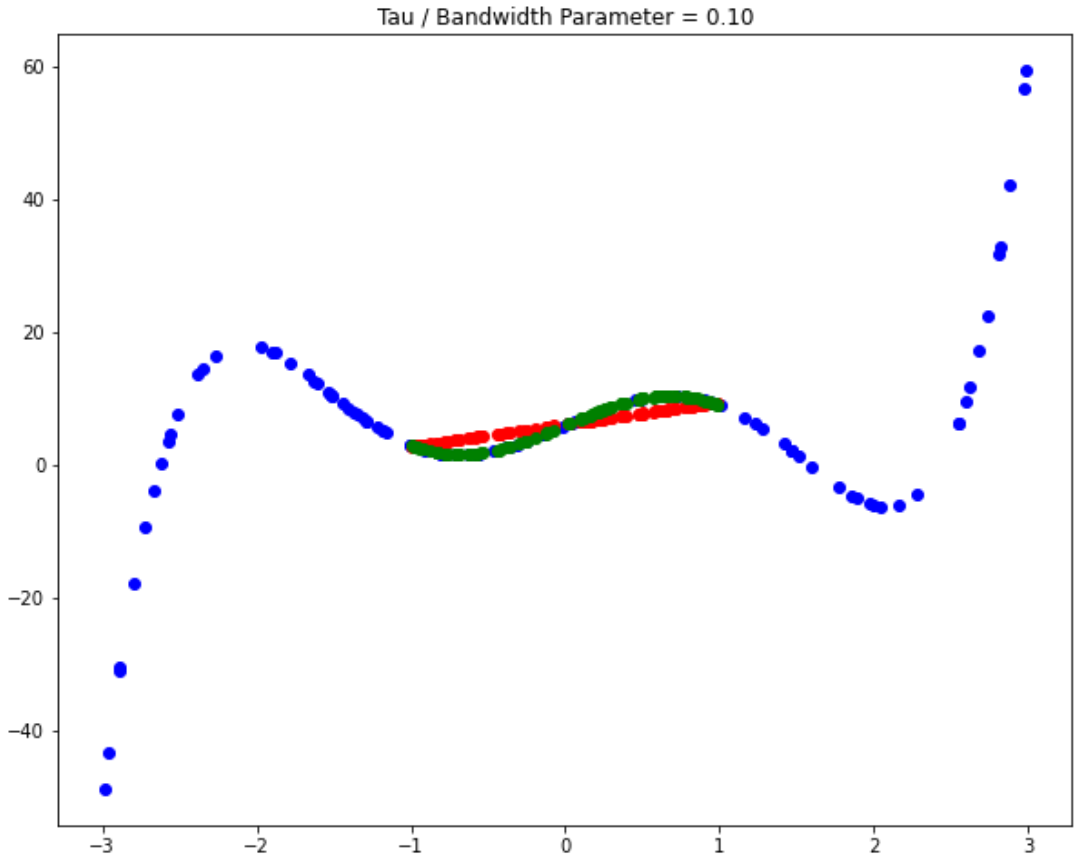
$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

Then I build prediction function which will calculate theta and make predict with every data points. The theta calculation formula and prediction formula are below:

$$\theta = (X^T W X)^{-1} X^T W Y$$

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Choosing Tau or bandwidth = 0.1, I put the model in the testing dataset. In below plot, red scatter points indicate the prediction line, green scatter points indicate the testing data and blue points indicate the training data.



1. We might not need a basis function using locally weighted approach. It is a non-parametric model so prediction will be made per data point by calculating theta for each of them so it can reflect the data points good enough in the model. If we add more basis function like adding higher terms, the computational resources dedicated will be a problem.
2. The main differences in implementing Q1 and this model are:
    - Q1 use gradient descent method, this model uses locally weighted method
    - Q1 is parametric model so training data used once to calculate theta, and we use the same theta to predict on testing data. This model is non-parametric model so for every data points prediction, we calculate corresponding theta and use this value to predict.
    - Calculation effort for this model may be drastically increase with more features, compared to Q1 model