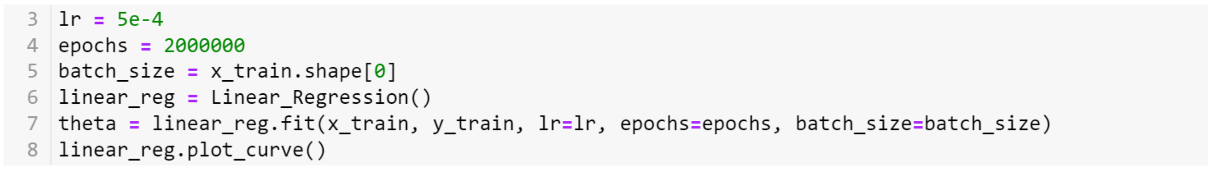
NYCU Pattern Recognition, Homework 1

109550090, 李以恩

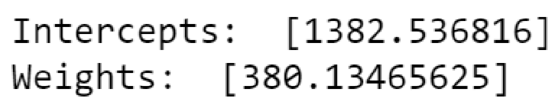
Part. 1, Coding (70%):

1. (0%) Show the learning rate and epoch you choose
   learning rate: __5e-4__
   epoch: __2000000__



```
lr = 5e-4
epochs = 2000000
batch_size = x_train.shape[0]
linear_reg = Linear_Regression()
theta = linear_reg.fit(x_train, y_train, lr=lr, epochs=epochs, batch_size=batch_size)
linear_reg.plot_curve()
```

2. (5%) Show the weights and intercepts of your linear model.
   weights: __1382.536816__
   intercepts: __380.13465625__

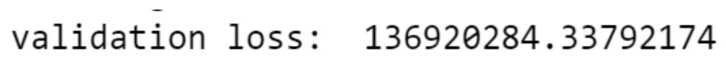


```
Intercepts:  [1382.536816]
Weights:  [380.13465625]
```

3. (5%) What's your final training loss (MSE)?
   training loss (MSE): __139562065.48342776__

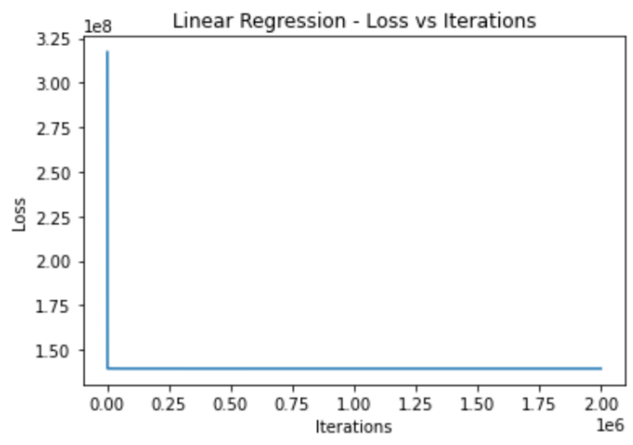
```
training loss:  139562065.48342776
```

4. (5%) What's the MSE of your validation prediction and validation ground truth?
   validation loss (MSE): __136920284.33792174__

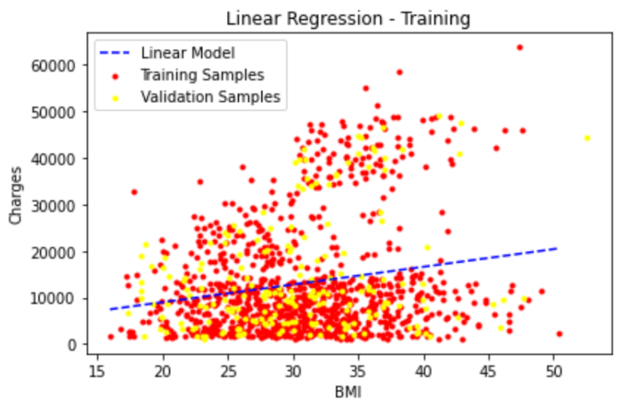


```
validation loss:  136920284.33792174
```

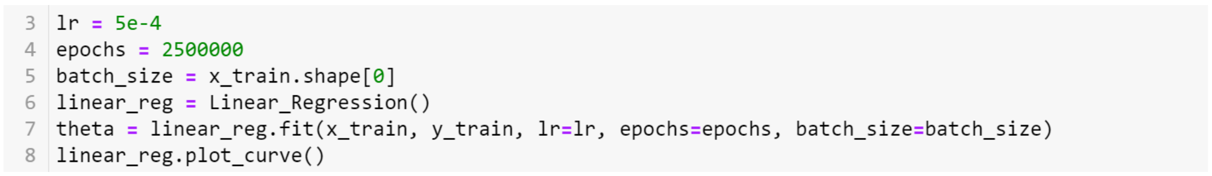
5. (5%) Plot the training curve. (x-axis=epoch, y-axis=loss)



6. (5%) Plot the line you find with the training and validation data.



7. (0%) Show the learning rate and epoch you choose.
learning rate: __5e-4__
epoch: __2500000__

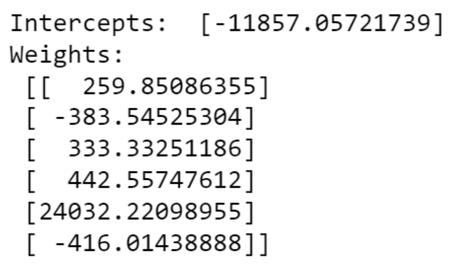


```
lr = 5e-4
epochs = 2500000
batch_size = x_train.shape[0]
linear_reg = Linear_Regression()
theta = linear_reg.fit(x_train, y_train, lr=lr, epochs=epochs, batch_size=batch_size)
linear_reg.plot_curve()
```

8. (10%) Show the weights and intercepts of your linear model.
weights:

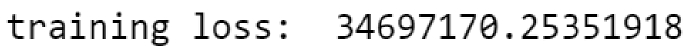
```
[[ 259.85086355]
 [ -383.54525304]
 [ 333.33251186]
 [ 442.55747612]
 [24032.22098955]
 [ -416.01438888]]
```

intercepts: __-11857.05721739__



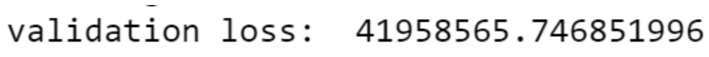
```
Intercepts:  [-11857.05721739]
Weights:
 [[  259.85086355]
 [ -383.54525304]
 [  333.33251186]
 [  442.55747612]
 [24032.22098955]
 [ -416.01438888]]
```

9. (5%) What's your final training loss?
training loss (MSE): __34697170.25351918__



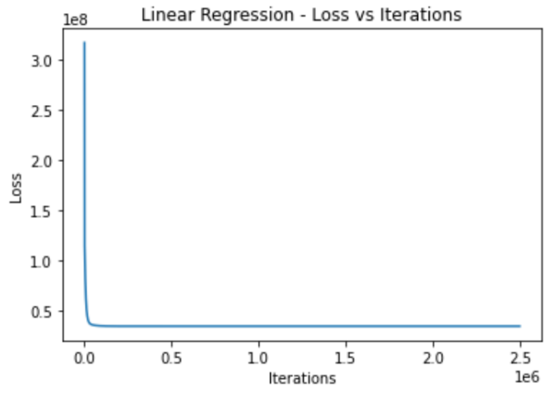
```
training loss:  34697170.25351918
```

10. (5%) What's the MSE of your validation prediction and validation ground truth?
validation loss (MSE): __41958565.746851996__



```
validation loss:  41958565.746851996
```

11. (5%) Plot the training curve. (x-axis=epoch, y-axis=loss)



12. (20%) Train your own model and fill the testing CSV file as your final predictions.

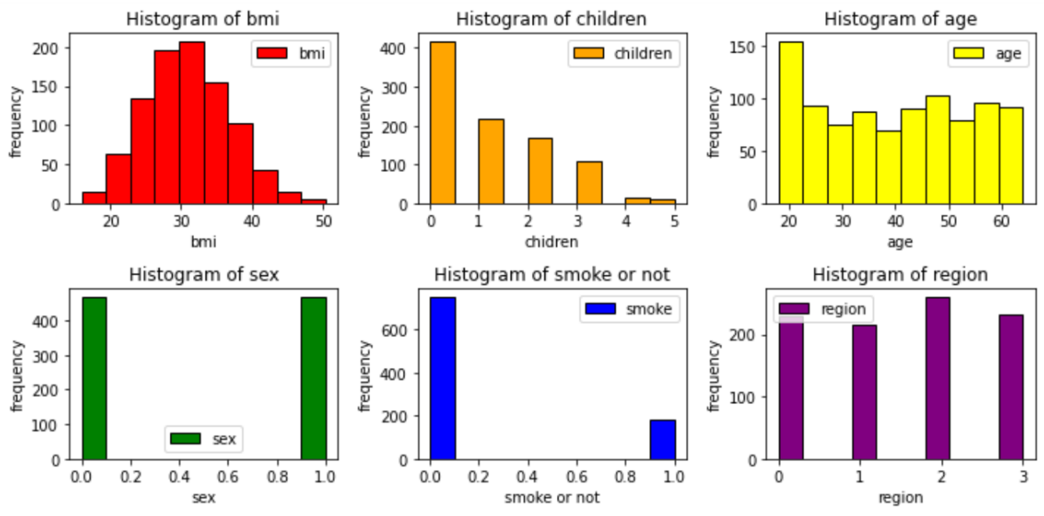
learning rate: 5e-4
epoch: 1000000
batch_size: x_train.shape[0]
Used features: age, bmi, children, female, male, smoker_no, smoker_yes, northeast, northwest, southeast, southwest, age*smoker_yes, bmi*smoker_yes, bmi_bins_smoker_yes

(1) Data Analysis

a. Histogram_Plot

After doing the Label_Encoding( ), I plotted the histograms to see the distributions of each feature. We can observe that:

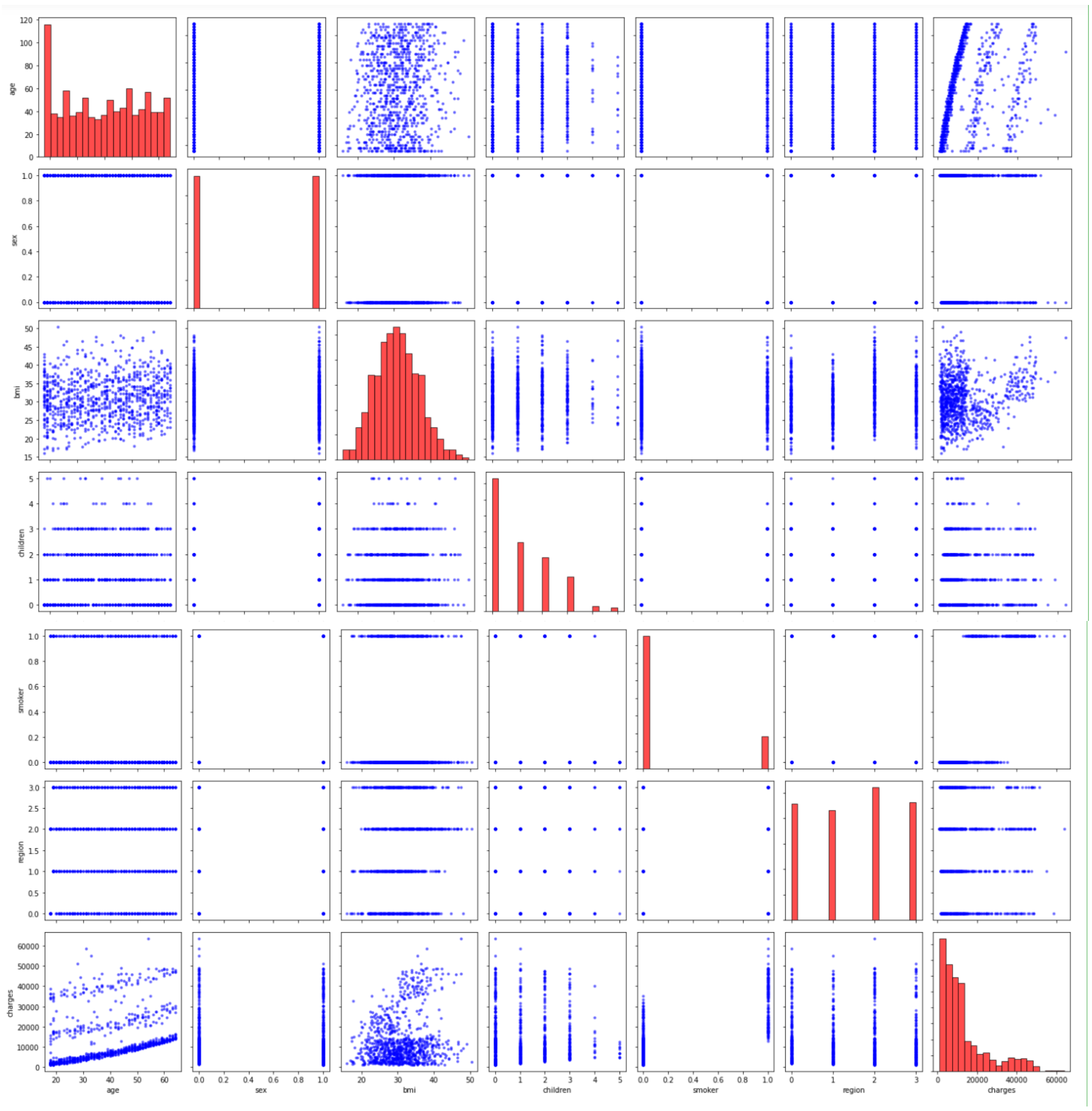
- The bmi distribution is nearly normal distributed.
- There is little difference between the percentage of male and female.
- There is little difference between the percentage of each region.
- Most people in the training dataset do not smoke.



b. **Pair_Plot**

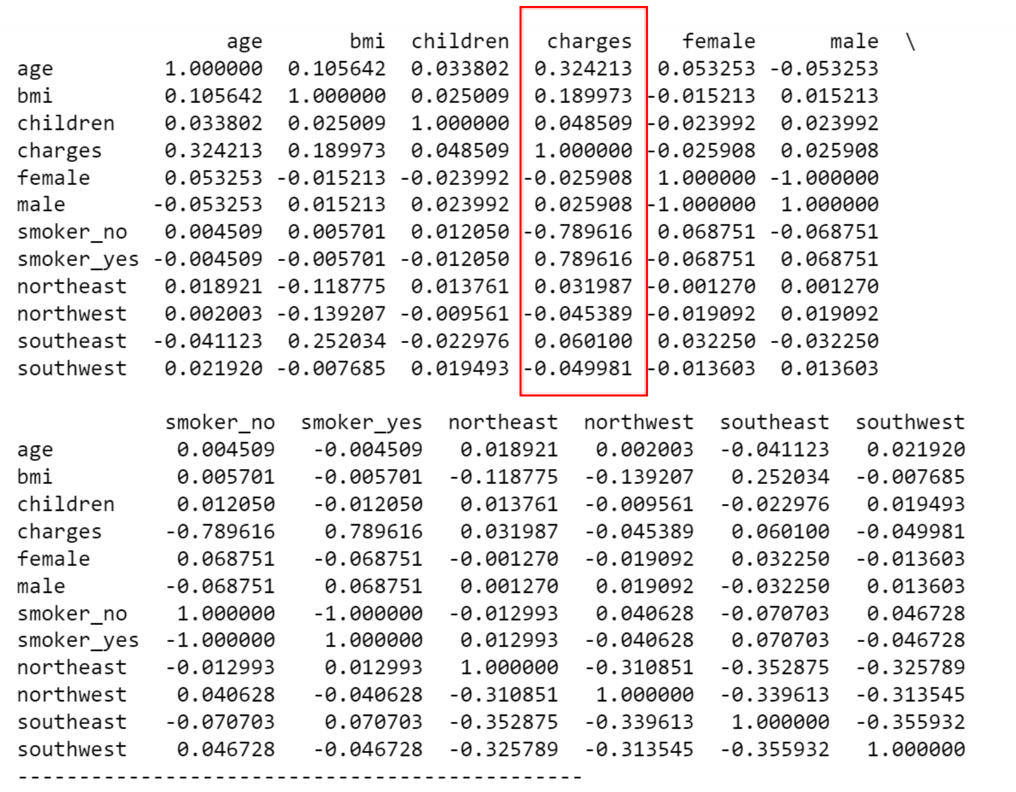
After doing the Label_Encoding( ), I plotted the pair correlation between each feature. We can observe that:

- The higher the age, the higher the insurance value.
- The higher the bmi, the higher the insurance value.
- A non-smoker has less insurance than a smoker.
- The amount of insurance according to gender is similar.
- The value of the insurance according to the region is also similar.



c. **Correlation Analysis**

After doing the Dummy_Encoding( ), I printed the correlation coefficient between each feature by calling df.corr( ). We can observe that **charges** has positive correlation with **age**, **bmi**, and especially high positive correlation with **smoker_yes**. Therefore, I considered these features to do the feature engineering.

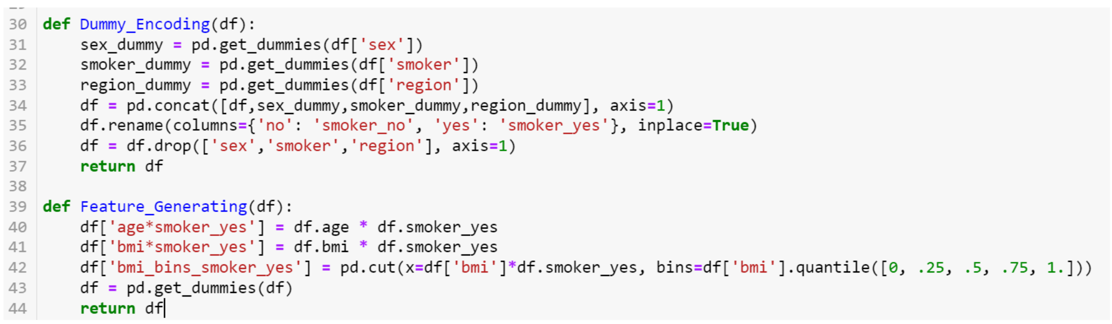


```
                 age       bmi  children   charges    female      male  \
age         1.000000  0.105642  0.033802  0.324213  0.053253 -0.053253
bmi         0.105642  1.000000  0.025009  0.189973 -0.015213  0.015213
children    0.033802  0.025009  1.000000  0.048509 -0.023992  0.023992
charges     0.324213  0.189973  0.048509  1.000000 -0.025908  0.025908
female      0.053253 -0.015213 -0.023992 -0.025908  1.000000 -1.000000
male       -0.053253  0.015213  0.023992  0.025908 -1.000000  1.000000
smoker_no   0.004509  0.005701  0.012050 -0.789616  0.068751 -0.068751
smoker_yes -0.004509 -0.005701 -0.012050  0.789616 -0.068751  0.068751
northeast   0.018921 -0.118775  0.013761  0.031987 -0.001270  0.001270
northwest   0.002003 -0.139207 -0.009561 -0.045389 -0.019092  0.019092
southeast  -0.041123  0.252034 -0.022976  0.060100  0.032250 -0.032250
southwest   0.021920 -0.007685  0.019493 -0.049981 -0.013603  0.013603

            smoker_no  smoker_yes  northeast  northwest  southeast  southwest
age          0.004509   -0.004509   0.018921   0.002003  -0.041123   0.021920
bmi          0.005701   -0.005701  -0.118775  -0.139207   0.252034  -0.007685
children     0.012050   -0.012050   0.013761  -0.009561  -0.022976   0.019493
charges     -0.789616    0.789616   0.031987  -0.045389   0.060100  -0.049981
female       0.068751   -0.068751  -0.001270  -0.019092   0.032250  -0.013603
male        -0.068751    0.068751   0.001270   0.019092  -0.032250   0.013603
smoker_no    1.000000   -1.000000  -0.012993   0.040628  -0.070703   0.046728
smoker_yes  -1.000000    1.000000   0.012993  -0.040628   0.070703  -0.046728
northeast   -0.012993    0.012993   1.000000  -0.310851  -0.352875  -0.325789
northwest    0.040628   -0.040628  -0.310851   1.000000  -0.339613  -0.313545
southeast   -0.070703    0.070703  -0.352875  -0.339613   1.000000  -0.355932
southwest    0.046728   -0.046728  -0.325789  -0.313545  -0.355932   1.000000
----------------------------------------------
```

(2) Feature Engineering

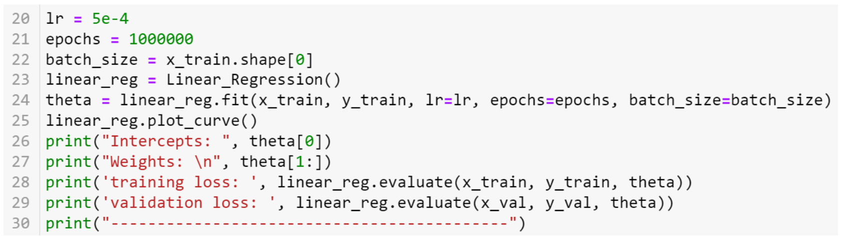
I have added three features. For **age*smoker_yes** and **bmi*smoker_yes**, I multiplied the two specific features, since they are all positively correlated with **charges**. For **bmi_bins_smoker_yes**, I used discretization of the feature to split it into multiple new features which can compile significant interactions to our feature matrix.



```python
def Dummy_Encoding(df):
    sex_dummy = pd.get_dummies(df['sex'])
    smoker_dummy = pd.get_dummies(df['smoker'])
    region_dummy = pd.get_dummies(df['region'])
    df = pd.concat([df,sex_dummy,smoker_dummy,region_dummy], axis=1)
    df.rename(columns={'no': 'smoker_no', 'yes': 'smoker_yes'}, inplace=True)
    df = df.drop(['sex','smoker','region'], axis=1)
    return df

def Feature_Generating(df):
    df['age*smoker_yes'] = df.age * df.smoker_yes
    df['bmi*smoker_yes'] = df.bmi * df.smoker_yes
    df['bmi_bins_smoker_yes'] = pd.cut(x=df['bmi']*df.smoker_yes, bins=df['bmi'].quantile([0, .25, .5, .75, 1.]))
    df = pd.get_dummies(df)
    return df
```

(3) Parameter Tuning



```python
lr = 5e-4
epochs = 1000000
batch_size = x_train.shape[0]
linear_reg = Linear_Regression()
theta = linear_reg.fit(x_train, y_train, lr=lr, epochs=epochs, batch_size=batch_size)
linear_reg.plot_curve()
print("Intercepts: ", theta[0])
print("Weights: \n", theta[1:])
print('training loss: ', linear_reg.evaluate(x_train, y_train, theta))
print('validation loss: ', linear_reg.evaluate(x_val, y_val, theta))
print("-----------------------------------------")
```

Part. 2, Questions (30%):

(7%) 1. What's the difference between Gradient Descent, Mini-Batch Gradient Descent, and Stochastic Gradient Descent?

- Gradient Descent:

  When doing every gradient descent process, **all the training data** is taken into consideration to compute the gradient and take a single step.

- Mini-Batch Gradient Descent:

  When doing every gradient descent process, **a batch of a fixed number of training examples** is taken into consideration to compute the gradient and take a single step.

- Stochastic Gradient Descent:

  When doing every gradient descent process, **only one example** is taken into consideration to compute the gradient and take a single step.

(7%) 2. Will different values of learning rate affect the convergence of optimization? Please explain in detail.

Yes, the value of learning rate can affect the convergence of optimization greatly. If the learning rate is **too small**, the optimization process may take a long time to converge or even get stuck in local minimum. If the learning rate is **too large**, the optimization may oscillate around the minimum or diverge. A proper learning rate should be chosen based on the problem specification and the optimization algorithm which is being used. If the convergence is too slow, the learning rate can be increased; if the optimization diverges, the learning rate should be decreased.

(8%) 3. Suppose you are given a dataset with two variables, X and Y, and you want to perform linear regression to determine the relationship between these variables. You plot the data and notice that there is a strong nonlinear relationship between X and Y. Can you still use linear regression to analyze this data? Why or why not? Please explain in detail.

If there is a strong **nonlinear relationship** between X and Y, linear regression may not be the best method to analyze the data since linear regression assumes that there is a linear relationship between the X and Y. However, if the nonlinearity is not severe, it may still be possible to use linear regression by transforming the variables or by adding polynomial terms to the model. If the nonlinearity is too strong, it may be necessary to use nonlinear regression techniques or other methods which can model more complex relationships between variables.

(8%) 4. In the coding part of this homework, we can notice that when we use more features in the data, we can usually achieve a lower training loss. Consider two sets of features, A and B, where B is a subset of A. (1) Prove that we can achieve a non-greater training loss when we use the features of set A rather than the features of set B. (2) In what situation will the two training losses be equal?

1) For linear regression, we are solving the problem by minimizing MSE = (1/n) * sum[(Wx+b) - t]^2. Since B is a subset of A, using the features of set A can cover the case of using the features of set B. Therefore, the model trained with the features of set A can achieve the same or lower training loss than the model trained with the features of set B, since set A has provided more information while training.
2) When the features in set A and set B have exactly the same fitting ability to the training data (ex: set B contains all the features in feature set A and no additional features) or when the weights of features in set(A-B) is zero, then the training losses of the models trained with feature set A and feature set B will be equal.

   It is possible that using all the features in set A may lead to a lower training loss rather than set B. However, using additional features can also add more noise and overfitting to the model.