1.1 Question (60%): Implement a simple Linear Regression on a toy dataset

Steps:
1. Consider this toy dataset

```
YearsExperience,        Salary
1.1,                    39343.00
1.3,                    46205.00
1.5,                    37731.00
2.0,                    43525.00
2.2,                    39891.00
2.9,                    56642.00
3.0,                    60150.00
3.2,                    54445.00
3.2,                    64445.00
3.7,                    57189.00
3.9,                    63218.00
4.0,                    55794.00
4.0,                    56957.00
4.1,                    57081.00
4.5,                    61111.00
4.9,                    67938.00
5.1,                    66029.00
5.3,                    83088.00
5.9,                    81363.00
6.0,                    93940.00
6.8,                    91738.00
7.1,                    98273.00
7.9,                    101302.00
8.2,                    113812.00
8.7,                    109431.00
9.0,                    105582.00
9.5,                    116969.00
9.6,                    112635.00
10.3,                   122391.00
10.5,                   121872.00
```

The first column is the independent variable (Years of Experience) and the second column is the dependent variable (Salary).
Read in this dataset (store it as .CSV or just .txt) and **Display** this toy dataset.

2. Training set and Testing set
Let's split the data to 80% training and 20% testing
(We could manually implement this easy function, or consider
sklearn.model_selection.train_test_split)

3. Training a linear regression model on the training set
Consider using sklearn.linear_model.linearregression

Optional Practice (not mandatory)
implement the linear regression model $y = wx+b$ and train it by implementing a gradient descent as discussed in class (Lecture 3 Regression).
end description of Optional Practice

Display the trained weights w and b.

4. Use the trained model to predict the testing set.
Thus, ($y = wx+b$), we input the Years of Experience in the testing set, and get the predicted Salary. We compare the prediction with the actual Salary in the dataset.
Plot (consider matplotlib) the Years of Experience (x-axis) v.s. the actual Salary along with the regression model line together.

Submission: a word/PDF report containing the following contents.

1. Runnable codes (70%)

2. The required displaying or printings or plots or outputs highlighted bold in the above description (15%)

3. Discussion of thinking and discovery (15%)

1.2 Question (40%): Implement Naïve Bayes classifier on a toy dataset

Consider this toy training dataset (3 features per sample, and there are 6 samples.)
Class 1 (C1): [1, 1, 1], [0, 1, 0], [1, 1, 0]
Class 2 (C2): [0, 0, 0], [1, 0, 1], [1, 0, 0]

Now we have testing data x: [0, 1, 1], write from scratch a Python program (sklearn or other machine learning packages are **not** allowed) to classify it.

(A good example of Naïve Bayes on a toy dataset can be found at Page 9, Lecture 5)
Display the results of P(C1), P(C2), P(x|C1), P(x|C2), and P(C1|x) and P(C2|x). Which class do we classify the testing data x based on the results?

Optional Practice (not mandatory)
Make sure that your implementation can handle the situation when these following three constraints are simultaneously satisfied: (i) the number of features per sample is a variable; (not fixed as 3 as in the previous example) (ii) the number of samples is a variable (not fixed as 6 as in the previous example); and (iii) classify any given testing data of the same number of features.
E.g.,
Class 1: [1, 1, 1, 1, 1], [0, 1, 0, 0, 0], [1, 1, 0, 1, 1], [1, 1, 0, 1, 0]
Class 2: [0, 0, 0, 0, 0], [0, 0, 0, 1, 1]
end description of Optional Practice

Submission: a word/PDF report containing the following contents.

1. Runnable codes (70%)

2. The required displaying or printings or plots or outputs as in the above description (15%)

3. Discussion of thinking and discovery (15%)