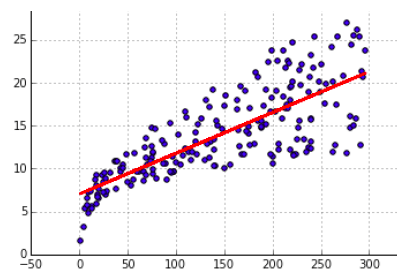
Simple Linear Regression

This blog focuses on how Simple Linear Regression works. We have tried to explain every concept in simple words. You can find the code [here](https://github.com/sinha-abhash/machine-learning/blob/master/Supervised%20Learning/Linear%20Regression/SimpleLinearRegression.ipynb). If you are familiar with the concepts, please feel free to skip to the next section.

**What is Simple Linear Regression?**

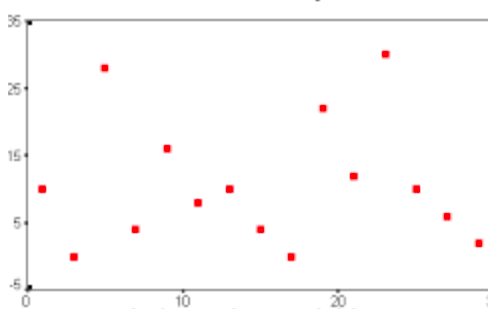
Simple Linear Regression is finding the best relationship between the input variable **x**(independent variable) and the expected variable **y**(dependent variable). The linear relationship between these two variables can be represented by a straight line called **regression line**.

To understand this concept more clearly, let’s look at the below plot of data.



Here we can see that the variable **y** is linearly dependent on the variable **x**. Hence, we can find a straight line which can best define the data.

But, consider the below plotting of data.



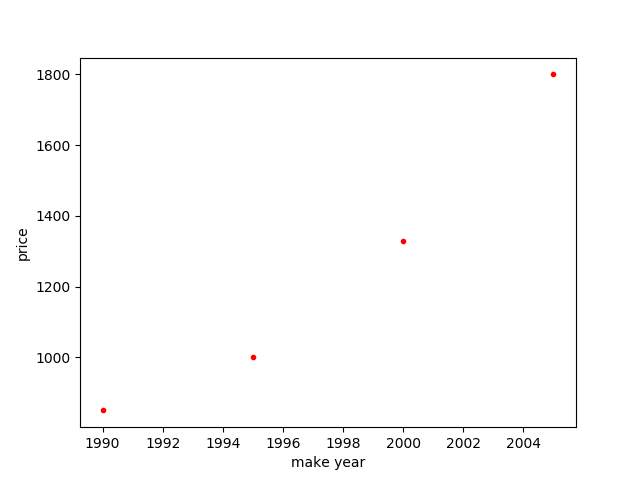
Here, the data is randomly distributed and there is no linear relationship between the variables and hence we can not find single straight line which can describe the relationship better.

**Example**

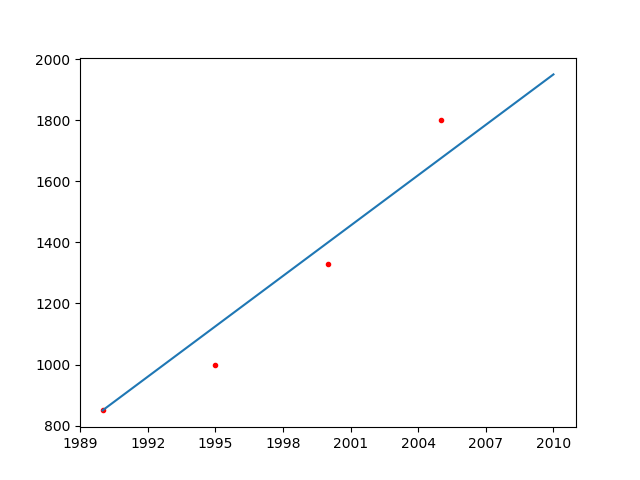
Let’s consider an example of predicting the price of a second-hand bike based upon its build year only based on the below given data.



Now, if we ask you to predict the price of a second-hand bike whose make year is 2010. The first thing you will do is to plot the above data.



Next, you will try to find a straight line which should be as close to every point as possible.



To predict the price of the second-hand bike for the make year 2010, you can simply extend this straight line further to check the y-axis value when x-axis value is 2010. This projected value of y-axis gives you the rough price of the bike which we are looking for. This straight line is **regression line**.

**Concepts:**

Let us start by studying the regression line in detail.

The equation of a straight line is represented by: **y** = **mx** + **c**

where **m** is the slope of the line also known as gradient and **c** is the point at which the line crosses the **y**-axis.

To find the best line for our data, we need to find the best set of slope **m** and y-intercept’s **c**values. In simple words, we already have **x**, we have to find **m** and **c** to predict the value of **y**.

Pheww!! Enough of maths for now!

**Diving into the code**

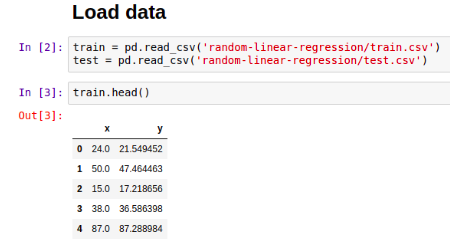
Dividing the code into steps for better understanding:

1. Download the dataset.
2. Visualize the data.
3. Training Simple Linear Regression Model.

**Step 1: Download the dataset.**

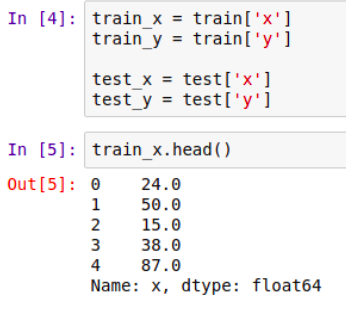
Alright then let’s start our baby steps with the dataset. We have downloaded the dataset from Kaggle from the [link](https://www.kaggle.com/andonians/random-linear-regression/data) specified. The data is just a toy example and there is no significance as such.

We have train.csv and test.csv now. Since the dataset is in CSV format, read\_csv method of Pandas is used. We load and read the data (x and y values) in the below snippet.

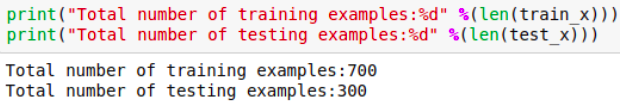


Here, x is the independent variable and y is the dependent variable which will be predicted based on the value of x. Therefore, we have separated x and y values of train and test file into train\_x, train\_y, test\_x and test\_y.

Further, we can see the values of train\_x in the below snippet:

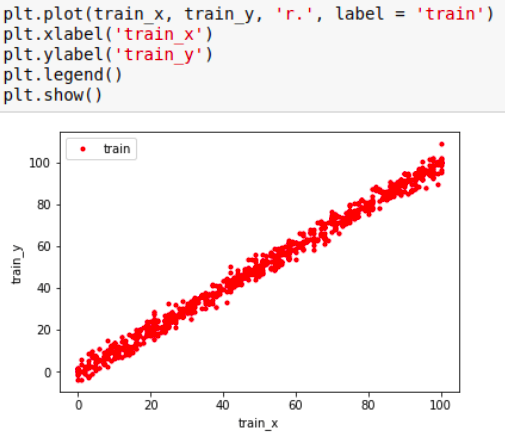


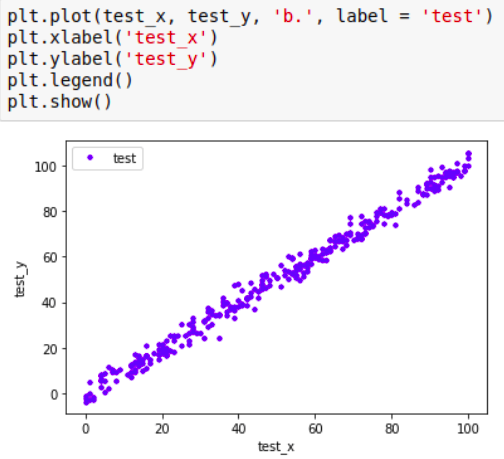
To make the dataset more clear, we will have a look at the number of train and test samples.



**Step 2: Visualize the data.**

Plotting the data of train and test set for better understanding.





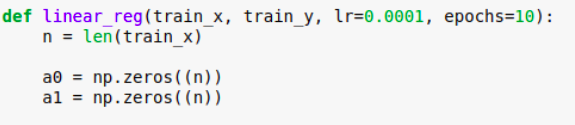
Ok, now we hope the dataset is pretty clear by now. We will move to the next step now!

**Step 3: Training Simple Linear Regression Model.**

Before starting with this step, lets get a high level idea of what is going to happen so that we are not lost in between.

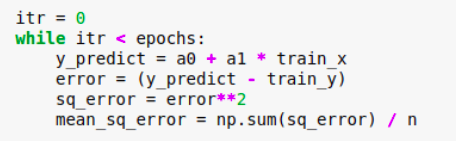
High level idea: The model is trained based on the values of **x** and **y** from the train dataset. The model learns the best fit by learning the values of **m** and **c**of the straight line or regression line equation. Once these values are learnt, the model uses these learnt values for prediction on the test dataset.

Here comes the villain now! Lets attack it line by line:



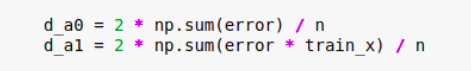
In this function we are passing 4 parameters: 2 parameters (train\_x, train\_y) and 2 hyper-parameters (lr, epochs) with their default values specified. lr: learning rate.

What is being done in the above snippet is : the parameters **a0**and **a1** are initialized to 0 vectors of size n which is equal to the total number of samples in the training set. Values can be randomly initialized as well. Goal is to iteratively reach the minimum loss value by updating these parameters at every epoch.

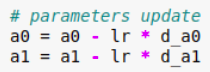


In order to achieve this, we define a straight line equation (a0 + a1\*train\_x) to predict the value followed by finding the error. This error is the distance between predicted value (y\_predict) and ground truth value (train\_y). Square this error then find the mean across the whole dataset. The distance is squared to ensure that it is positive and to make our error function differentiable. Remember our goal? To minimize error? we have to minimize this squared error! If we minimize this function, we will get the best line for our data.

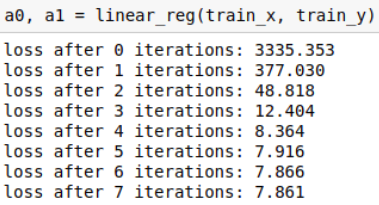
Now, you must be thinking what needs to be done to minimize the error? Don’t worry, its simple. We have to find the gradient or differentiation of loss function. If you are unfamiliar with Gradient Descent, then it is highly recommended to check it out [here](https://medium.com/machinelearningalgorithms/machine-learning-basics-742257fa8d49) before you proceed further.



We use this differentiated (d\_a0, d\_a1) value to update the initialized parameters a0, a1 which will be updated by subtracting the product of learning rate and the differentiated value from the previous value of the parameter (as seen in the below image). This step is repeated for every epoch until we get the minimum loss value.

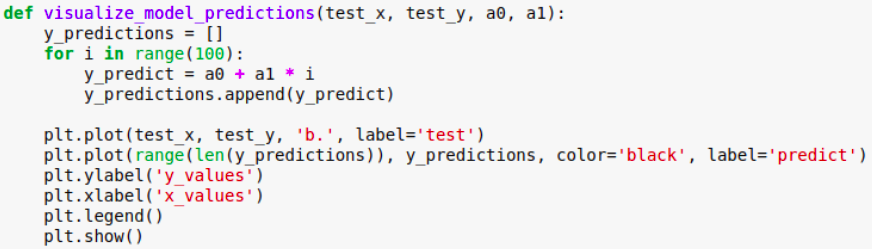


Voila! We can see that our model is learning below.

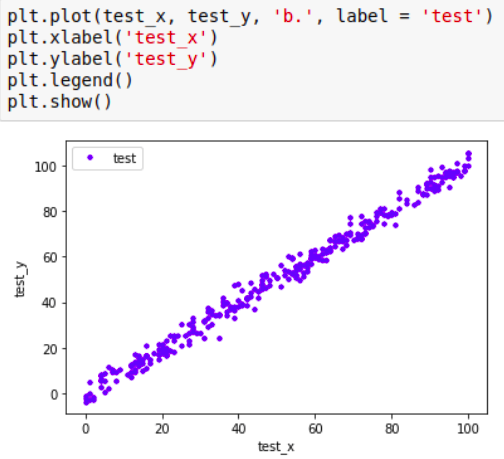


The loss is decreasing at every epoch. It started with big drop in the loss and once it reaches the minima of the loss function, decrement in the loss value is less; settling with the final value.

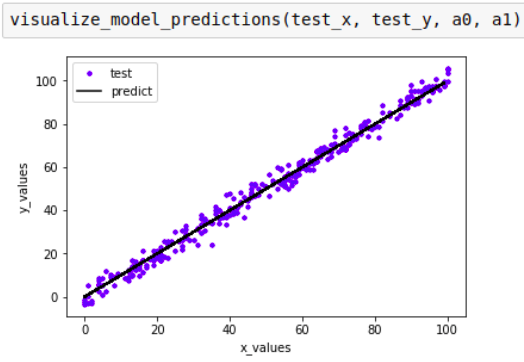
**Visualizing:**



For visualization purpose, we will first plot the test dataset again.



We will then use these learnt parameters (a0 and a1) to draw that infamous line which we have been talking all along. The line has been represented in black.



That concludes Simple Linear Regression for now!

Source Link: <https://medium.com/machinelearningalgorithms/simple-linear-regression-775e4d54bfed>