Simple Linear regression with Python

**Introduction to regression**

Simple linear regression (SLR) is first algorithm which generally ML enthusiast try their hands. SLR can be simply used where one independent variable is directly driving a value of a dependent variable. Linear Regression assumes a linear or straight line between relationship between two variables.

**Math Behind it**

This relationship for a SLR is simply written as:

A picture containing businesscard, screenshot, text

Description generated with high confidence

Dependent Variable ( Y ) – Dependent variable is something whose value we trying to predict or explain. This is the output variable.

Independent Variable ( X ) – This is the variable which is deriving the value of dependent variable. This is the input variable for the equation. In real world one variable not always impact the output but for the learning let’s assume X is directly impacting the Y.

Coefficient – It’s degree which determines, how a unit change in X will impact the value of Y.

Constant – It’s that minimum value which represents the value of dependent variable even in case of zero value of dependent variable.

Now let’s use this mathematical equation to solve some real-world problem.

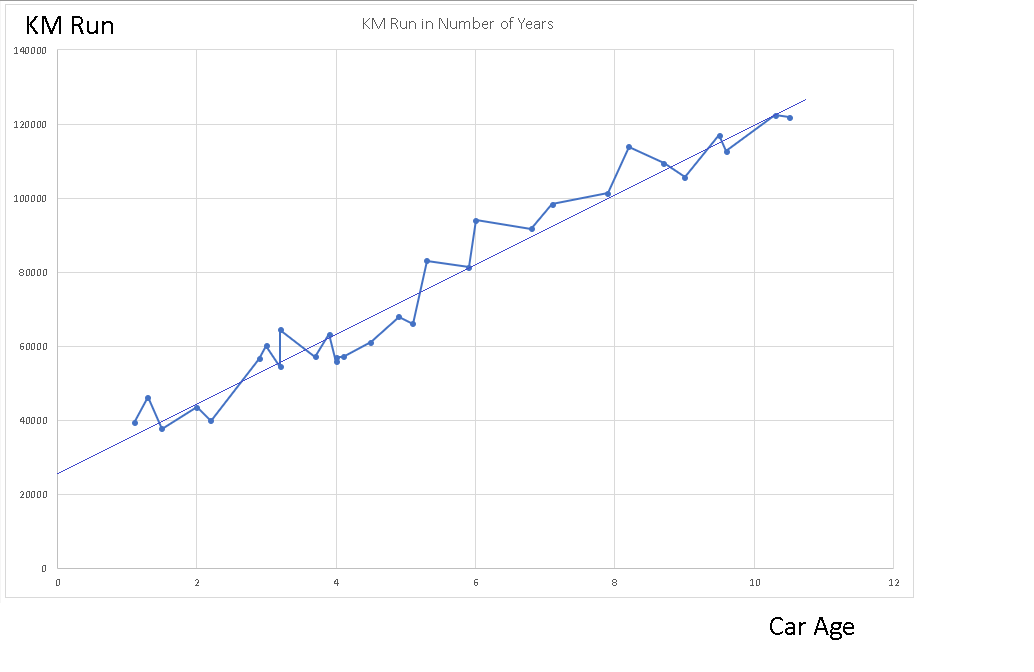
Problem Statement

Let’s say we have data set of car age (how old is the car) in years as independent variable and number of Kilometer driven as dependent variable.

A drawing of a face

Description generated with high confidence

And we want to detect total KM run on the bases of car age. Below is the graph plotted for KM Run vs Car age.



As you can in above graph, it is impossible to draw a straight line between the data points. Now our task is to identify the best fit in above plot. So that we can predict the response of any new feature. This best fit line is called Regression Line.

Let’s implement Linear Regression using Python sklearn library.

**Implementation using Python**

To implement the SLR into Python, below is step by step implementation:

1. Import three Python libraries and dataset:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  #Import data set  dataset =pd.read\_csv("Car\_age\_data.csv") |

1. Set independent and dependent variable. As observed into dataset, first column is our dependent variable whose value we want to predict and all other columns are independent variable.

|  |
| --- |
| #Set independent variable  X = dataset.iloc[:,:-1].values  #set dependent variable  y=dataset.iloc[:,1].values |

1. For splitting the data into training and test set, we have used sklearn.cross\_validation class.

For data splitting , you will always face a question. In which ratio one should divide the data between training and test set. The de facto standard of the diving ratio is somewhere between 80/20 to 70/30. I have seen people goes up to 65/30 but not more than that. For this exercise we have taken the 1/3rd of data as test data and rest has been used for training purpose.

|  |
| --- |
| #spliting data into training and test set  from sklearn.cross\_validation import train\_test\_split  X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=1/3,random\_state=0) |

1. Fitting simple linear regression to training set. This simple line of code train our model with the training data.

|  |
| --- |
| #fitting simple linear regression to training set  from sklearn.linear\_model import LinearRegression  regressor=LinearRegression()  regressor.fit(X\_train,y\_train) |

1. Moment of truth. Let’s predict the test set results with our trained model.

|  |
| --- |
| # Predicting the Test set results  y\_pred = regressor.predict(X\_test) |

Let’s compare the test result vs predicted results. As you can observe, our model has made few very good predictions e.g. Observation no. 4 (on index 3) is almost hits the bulls eye. As the predicted value is almost equal to test value. But on the other hand, few predictions are not good, predicted value of observation no. 2 get deviated with big margin. In coming chapters, we will learn how to improve the accuracy,

A screenshot of a cell phone

Description generated with very high confidence

1. Let’s Visualizing the results into graphics

|  |
| --- |
| # Visualising the Training set results￼  plt.scatter(X\_train, y\_train, color = 'red')  plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')  plt.title(KM Run vs Age (Training set)')  plt.xlabel('Car Age')  plt.ylabel('KM Run')  plt.show()  # Visualising the Test set results  plt.scatter(X\_test, y\_test, color = 'red')  plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')  plt.title('KM Run vs Age (Test set)')  plt.xlabel('Car Age')  plt.ylabel('KM Run')  plt.show() |

1. Graph of both the results are attached in below table:

|  |  |
| --- | --- |
| KM vs Age (Test Set) | KM vs Age (Training Set) |
| A close up of a map  Description generated with high confidence | A close up of a map  Description generated with very high confidence |

**Conclusion**

We have observed into given implementation, using Python sklearn library we can easily implement SLR. Being a machine learning soul, one should always try to resolve the ML problems with different methods and technology. Above problem is attempted using Python sklearn but same problem can be easily resolved using R also. This blog is contributed by [Amit Saini](https://www.linkedin.com/in/amsaini) . Please write comment if you find anything incorrect or you want to share some more information about the topic.