Finding the best regression line using regression model

Table of content

* Introduction to linear regression
* Implementation of Linear regression using Python
  + Exploring dataset
  + Splitting dataset
  + Training the model
  + Visualizing the trained model
  + Testing the model
  + Visualizing the model on testing data
  + Evaluating the model
* Summary

**Introduction to Linear regression model**

In machine learning, we mainly deal with three types of datasets: classification, regression, and forecasting. As the name suggests, a linear regression model is used to make predictions for regression datasets. A dataset that contains continuous values as output is known as a regression dataset. In this article, we learn how we can solve regression problems using various evaluation matrices.

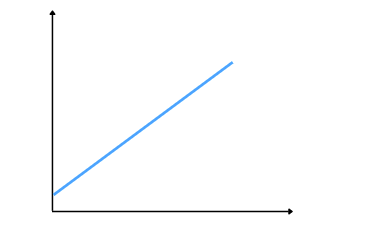
Linear Regression is one of the easiest and most popular Supervised Machine Learning algorithms. It is a technique for predicting a target value using independent factors. Linear Regression is mostly used for forecasting and determining cause and effect relationships among variables.

Based on the number of independent variables, a linear regression can be divided into two main categories.

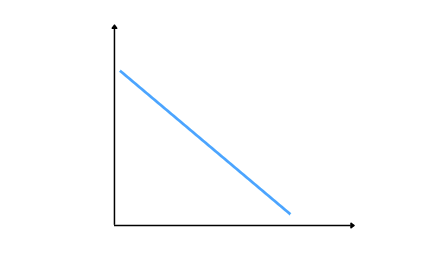
1. **Simple linear regression:** In simple Linear regression, where there is only one dependent and one corresponding independent variable. That means there is only one possible output Y for every input variable X.
2. **Multi-linear regression:** There are multiple independent variables and one corresponding dependent variable in multiple linear regression. That means for every output variable Y, there are more than one input variable X.

The next thing to understand is the type of regressions. Basically, there are two types of regressions. Positive regression and negative regression.

A **positive linear** relationship is when the dependent variable expands on the Y-axis while the independent variable increases on the X-axis. In simple words, as the input variables increase, the output variables also increase. The slope of such a linear relationship will be positive as shown below:



A negative linear relationship is when the dependent variable decreases on the Y-axis while the independent variable increases on the X-axis. In simple words, as the input variables increase, the output variables decrease. The slope of such a linear relationship will be negative as shown below:



**Implementation of linear regression using Python**

Now, we will use Python and its various built-in modules to implement the linear regression. But before going to the implementation parts, make sure that you have installed those required Python modules. In this article, we will be using the following modules.

* Sklearn
* Pandas
* Matplotlib

You install these modules on your system using pip command.

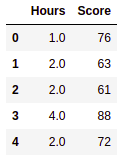
**Exploring the dataset**

For the demonstration purposes, we will be using a sample dataset about students. The input variable is the total number of hours a student spends on studying and the target variable is the score student obtained in exam out of 100. You can access the dataset from this link.

Let us first import the dataset using pandas module.

|  |
| --- |
| # importing pandas module  import pandas as pd    # importing dataset  data = pd.read\_csv('data.csv')    # heading  data.head() |

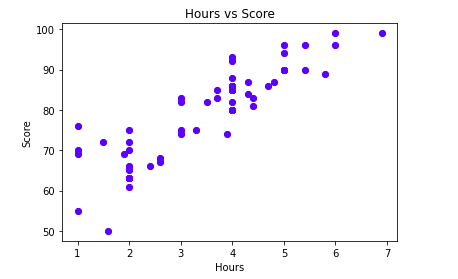
Output:



Let us also visualize the dataset to see the type of regression that the dataset has. We will use matplotlib module to visualize the dataset.

|  |
| --- |
| # importing the module  import matplotlib.pyplot as plt    #get a copy of dataset exclude last column  Input = data.iloc[:, :-1].values    #get array of dataset in column 2st  output = data.iloc[:, 1].values    # applying scttered graph  plt.scatter(Input, output, color='blue')  plt.title('Hours vs Score')    # X label and Y label  plt.xlabel('Hours')  plt.ylabel('Score')    # showing the graph  plt.show() |

Output:



As you can see, there is a positive regression relation between the input and output variables.

**Splitting dataset**

Now, we will split the dataset into testing and training parts. We will use the training parts to train the regression model and then will use a testing dataset to evaluate the model’s performance.

We will split the dataset into training and testing parts using sklearn module.

|  |
| --- |
| # importing dataset  from sklearn.model\_selection import train\_test\_split    # 70% dataset assigned to training part  X\_train, X\_test, y\_train, y\_test = train\_test\_split(Input, output, train\_size=.7) |

As you can see, we have assigned 70% of the dataset to the training part and the remaining 30% of the dataset to the testing part.

**Training the model**

Let us now train the model using linear regression algorithm. We will first import the regression model from sklearn module and then will initialize it.

|  |
| --- |
| # Importing linear regression form sklear  from sklearn.linear\_model import LinearRegression    # initializing the algorithm  regressor = LinearRegression() |

Once we initialize the regression model, we can use the training dataset to train the model.

|  |
| --- |
| # Fitting Simple Linear Regression to the Training set  regressor.fit(X\_train, y\_train) |

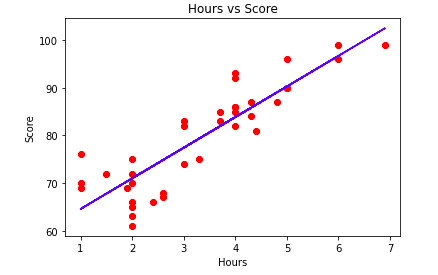
Now, our model is trained, we can check the best fitted model by visualizing the model.

**Visualizing the trained model**

Let us now visualize the trained model to the best fitted regression line.

|  |
| --- |
| # ploting the training dataset in scattered graph  plt.scatter(X\_train, y\_train, color='red')    # ploting the testing dataset in line line  plt.plot(X\_train, regressor.predict(X\_train), color='blue')  plt.title('Hours vs Score')    # labeling the input and outputs  plt.xlabel('Hours')  plt.ylabel('Score')    # showing the graph  plt.show() |

Output:



As you can see, the red dots are the actual data points, and the blue line is the best fitted regression line of the model. We can clearly see that the model has performed well and was able to follow the trend.

**Testing the model**

Now we will use the testing dataset to to make predictions. This time we will only use the input values and the model will predict the output values for us based on the above training.

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

The predictions of the model are stored in a variable y\_pred. We can now print both the actual values and the predicted values of the model to see how well our model is predicting the output values.

|  |
| --- |
| print("predicted values ",y\_pred)  print("\nAcutal values values ",y\_test) |

Output:

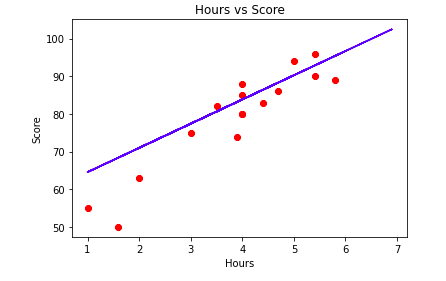
predicted values [77.41280329 83.84234185 92.84369584 90.27188042 64.55372617 92.84369584  
 95.41551127 83.84234185 88.34301885 70.98326473 83.84234185 68.41144931  
 80.62757257 83.84234185 86.41415728 83.199388 ]  
  
Acutal values values [75 80 96 94 55 90 89 80 86 63 85 50 82 88 83 74]

**Visualizing the testing part**

Now, we will use the same method to visualize the performance of model on testing dataset.

|  |
| --- |
| # ploting the training dataset in scattered graph  plt.scatter(X\_test, y\_test, color='red')    # ploting the testing dataset in line line  plt.plot(X\_train, regressor.predict(X\_train), color='blue')  plt.title('Hours vs Score')    # labeling the input and outputs  plt.xlabel('Hours')  plt.ylabel('Score')    # showing the graph  plt.show() |

Output:



This graph shows the best fitted line of the model for the testing dataset.

**Evaluating the model**

We already know that Linear Regression tries to fit a line that produces the smallest difference between predicted and actual values. We get the best fit line from the training dataset. Different performance evaluation methods are used to know how close the predicted line is to the actual values. These methods help us calculate the linear regression model’s accuracy, precision, and f-score on the specific dataset. In this section, we will use the Mean Absolute error, the Mean Square error, and R-square error to evaluate the performance of the regression model.

The **Mean Absolute Error (MAE)** is the simplest regression error metric. It represents the average absolute difference between the actual and predicted values in the dataset. And measures the average of the residuals in the dataset. We can use sklearn module to find the MAE of our model.

|  |
| --- |
| # Importing sklearn module  from sklearn.metrics import mean\_absolute\_error    # printing the mean absolute error  print(mean\_absolute\_error(y\_test, y\_pred)) |

Output:

5.239616618825823

The **Mean Square Error (MSE)** is similar to the Mean Absolute Error (MAE), but instead of taking the absolute value, it squares the difference before adding them together. The MSE will always be larger than the MAE because we squared the difference. Let us also calculate the MSE of our model.

|  |
| --- |
| # Importing sklearn module  from sklearn.metrics import mean\_squared\_error    # printing the mean squared error  print(mean\_squared\_error(y\_test, y\_pred)) |

Output:

45.29548665818075

The **R-Squared Error** method is also known as the coefficient of determination. This metric indicates how well a model fits a given dataset. Or in simple words, it indicates how close the regression line is to the actual data values. The R-Squared value lies between 0 and 1, where 0 indicates that this model doesn’t fit the given data and 1 means that the model perfectly fits the dataset provided. Let us also calculate the R-square score of the model.

|  |
| --- |
| # importing the module  from sklearn.metrics import r2\_score    # applying r square error  R\_square = r2\_score(y\_test, y\_pred)  print(R\_square) |

Output:

0.7254559005470624

As we can see that the scores of evaluation matrices show that our model has performed well on the given dataset.

**Summary**

A regression model provides a function that describes the relationship between one or more independent variables and a response. In this article, we discussed the regression model and we implemented it on a sample dataset. We also learned about some of the evaluation matrices of a regression model.