Exploring Machine Learning Algorithms — Simple Linear Regression

Eshita Goel
Aug 13 · 8 min read

As we dive into the world of Machine Learning and Data Science, one of the easiest and fun ways is to explore the various machine learning algorithms. They can be intimidating, especially if you're just starting out. One of the simplest algorithms that we can explore with very basic knowledge of data science is the Linear Regression algorithm.

About Linear Regression

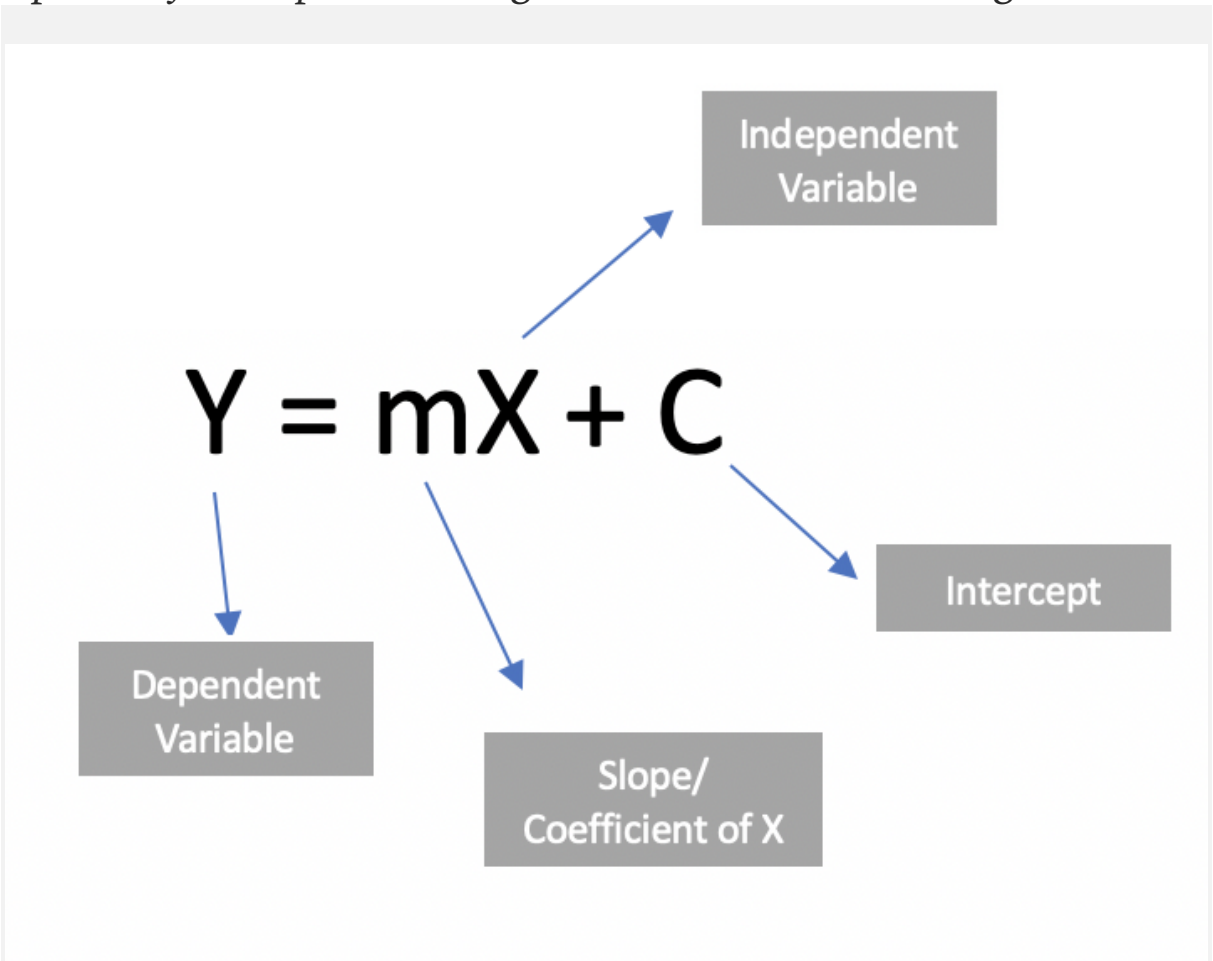
Linear Regression is mainly used in predicting continuous values. We deal with 2 kinds of variables:

1. **Independent Variables:** *The variables whose values are not affected by any other variables in our dataset. Like the age of a person or the gender of an individual.*
2. **Dependent Variables***: If the value of a variable is influenced by the value of one or more variables, then that variable is a dependent variable. For example, the designation of a person will influence his/her salary, or the number of bedrooms in a house will influence the cost of the house.*

In any form of regression, our aim is to investigate the relationship between the independent and dependent variables.

In simple linear regression, we try to model the data using a straight line that best fits our data. We then make the predictions using this line of best fit. Our predictions are usually the value of the independent variable and for this to work properly, our independent variable must be continuous rather than discrete.

If you don't want to go into the math too much, Linear Regression is simply expressing the relationship between the dependent and independent variables in a linear equation. For example, the equation for simple linear regression will look something like this:



Plugging in the various values of X, we can easily get the corresponding values of Y.

For Multiple Linear Regression, we have more than one independent variables that influence our dependent variable. Here we form an equation like -

$$Y = m_1 X_1 + m_2 X_2 + ... + m_n X_n + C$$

Here again, by plugging in the various values of the independent variables, we get our corresponding value of Y.

$$m = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$$

$$\text{and } c = \frac{\Sigma y - m\Sigma x}{n}$$

Image via: https://tinyurl.com/y55aen6h

The formula for calculating the coefficients m and the intercept c isn't something we need to memorize as, during our modelling

stage, it would be automatically done for us when we train our data.

Uses of Linear Regression

1. *Predicting continuous data or trends, like economic growth or sales estimates*
2. *Predicting the prices of commodities based on their various characteristics, for example, calculating the price of a house based on the number of rooms, square footage, location etc.*
3. *Sports predictions*

Step by Step Linear Regression

A good way to practice linear regression for the first time is to take a dataset having 2 variables, an independent variable and a dependent variable, and test the algorithm.

I will be taking a dataset consisting of 2 columns — "Hours" and "Scores". The "Hours" depict the number of hours studied by a student for a test and the "Scores" depicts the percentage scored by the student on that test. We can imagine that the more a student studies, the higher scores he/she is likely to get. Of course, there can be other factors that we should be considering, but let's focus on just these two variables for now.

1. *We start off with importing our libraries. This is very important as it helps us perform the various tasks further on.*

```
# Importing the libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

The **pandas** *library allows us to work with dataframes*
The **numpy** *library allows us to work with arrays*
matplotlib.pyplot *will help us in plotting graphs to visualize our data.*
seaborn *is a library made on top of matplotlib that also allows us in data visulaization.*

2. Now import your data. The data I worked with can be found on this link[link]It was provided to me when I was working on this mini-project as part of a Data Science internship with The Sparks Foundation. I will use pandas to import my data into a dataframe.

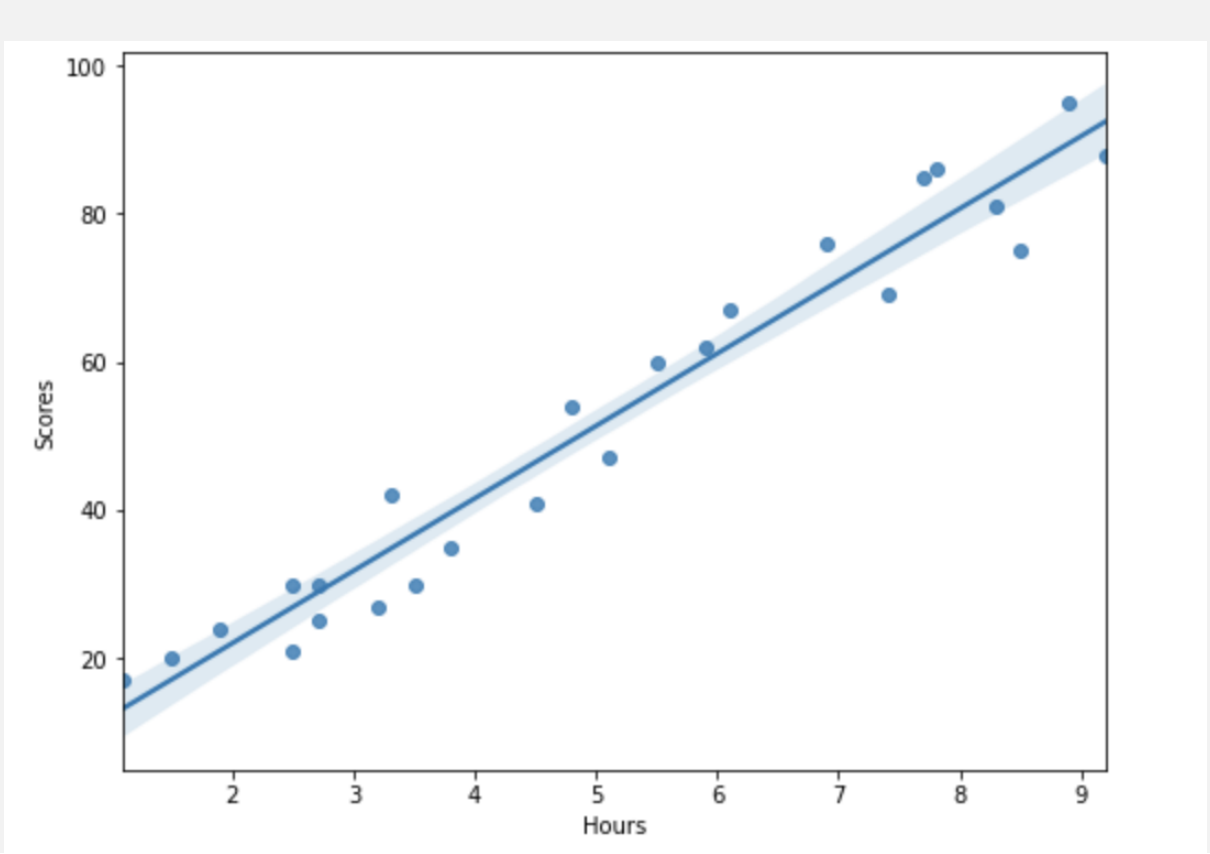
```
In [24]: df = pd.read_csv("http://bit.ly/w-data")
         df
```

Out[24]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88

3. **Explore your data***: Before you actually use linear regression to make predictions, you should see if your data actually has a linear relationship or not. We can use many visualization techniques to check this. My favourite is using seaborn's regplot that not only plots our data, but also shows a regression line. This makes it easy to see if our data is has a linear relationship or not.*

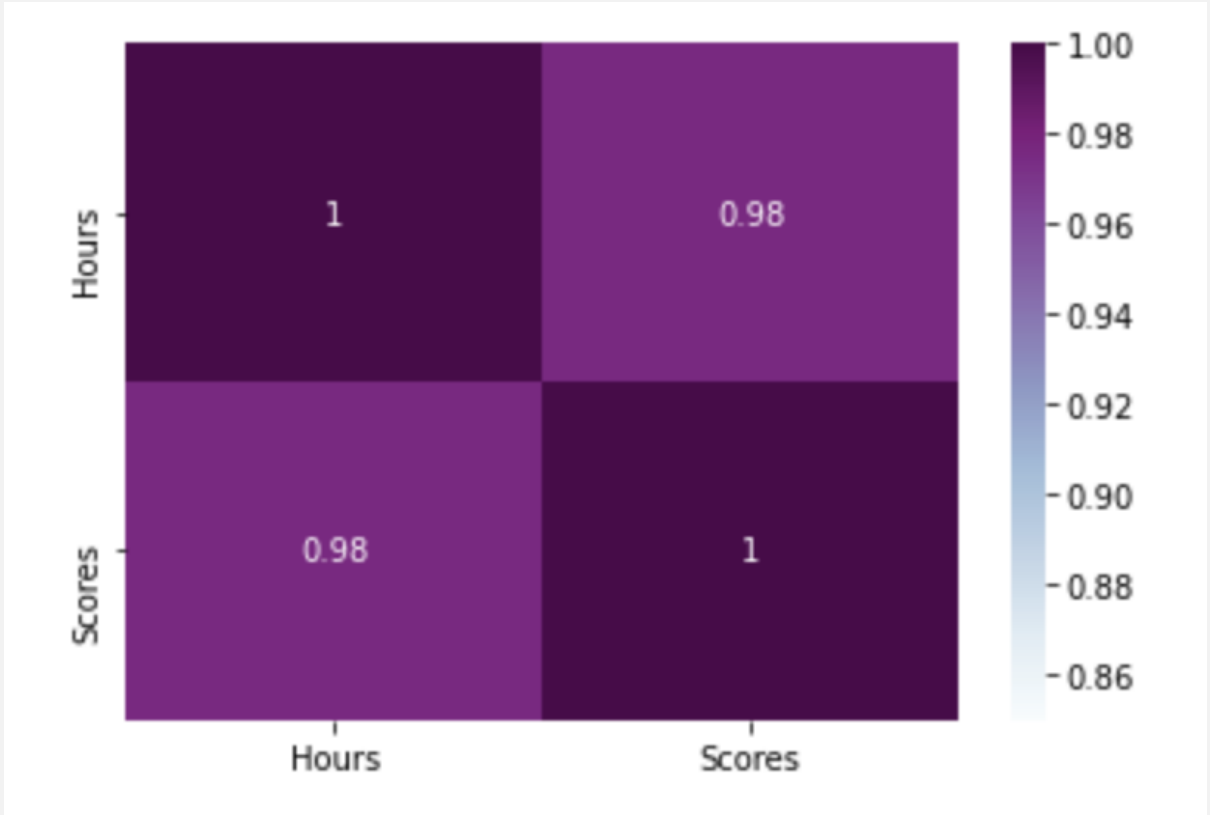
```
In [25]: plt.figure(figsize=(8,6))
         sns.regplot("Hours","Scores",data=df)
```



We can see an obvious positive linear relationship between the Hours and the Scores. As the Hours are increasing, so are the scores and the increase is linear.

Such a regplot is good if you have less number of independent variables. But it's quite inconvenient to plot this for all variables if you have many of them. In that case, I recommend a Heat Map. Plotting the correlation between each independent variable with the dependent variable using a heatmap gives us information about the whole data in practically just 1 line of code.

```
sns.heatmap(df.corr(), cmap="BuPu", vmin=0.85, vmax = 1)
```



The correlation between Hours and Scores is very high, Linear Regression to make predictions seems like a good idea.

4. **Making the Model:** *Now that we know we can use linear regression for this data, we make our model.*

Firstly, we store our values for independent and dependent variables in X and Y, respectively.

```
X = df[["Hours"]].values
Y = df[["Scores"]].values
```

Now we make use of the sklearn library and using model_selection we import train_test_split. This will allow us to split our data into 2 parts: a training part and a testing part. This is useful as we can train our data on one part and use the other part to test our

accuracy. If we were to use all the data for training, we would have no way of knowing if any subsequent predictions made by us are accurate or not.

```
from sklearn.model_selection import  train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

Now we make the model. From sklearn.linear_model, we import LinearRegression. We then create 'lm' which will be used for Linear regression.

We fit and train our data and then store the predictions for the testing data in yhat. The r2_score is a metric that allows us to check the accuracy of our data. Here we can see that our data has a r2 score of about 0.9454, which is a very good accuracy score keeping the kind of data we have. We will explore what r2 score is in the further paragraphs.

```
from sklearn.linear_model import LinearRegression
from  sklearn.metrics import r2_score

lm = LinearRegression()
lm.fit(X_train,y_train)
yhat = lm.predict(X_test)
r2_score(y_test,yhat)
```

```
0.9454906892105356
```

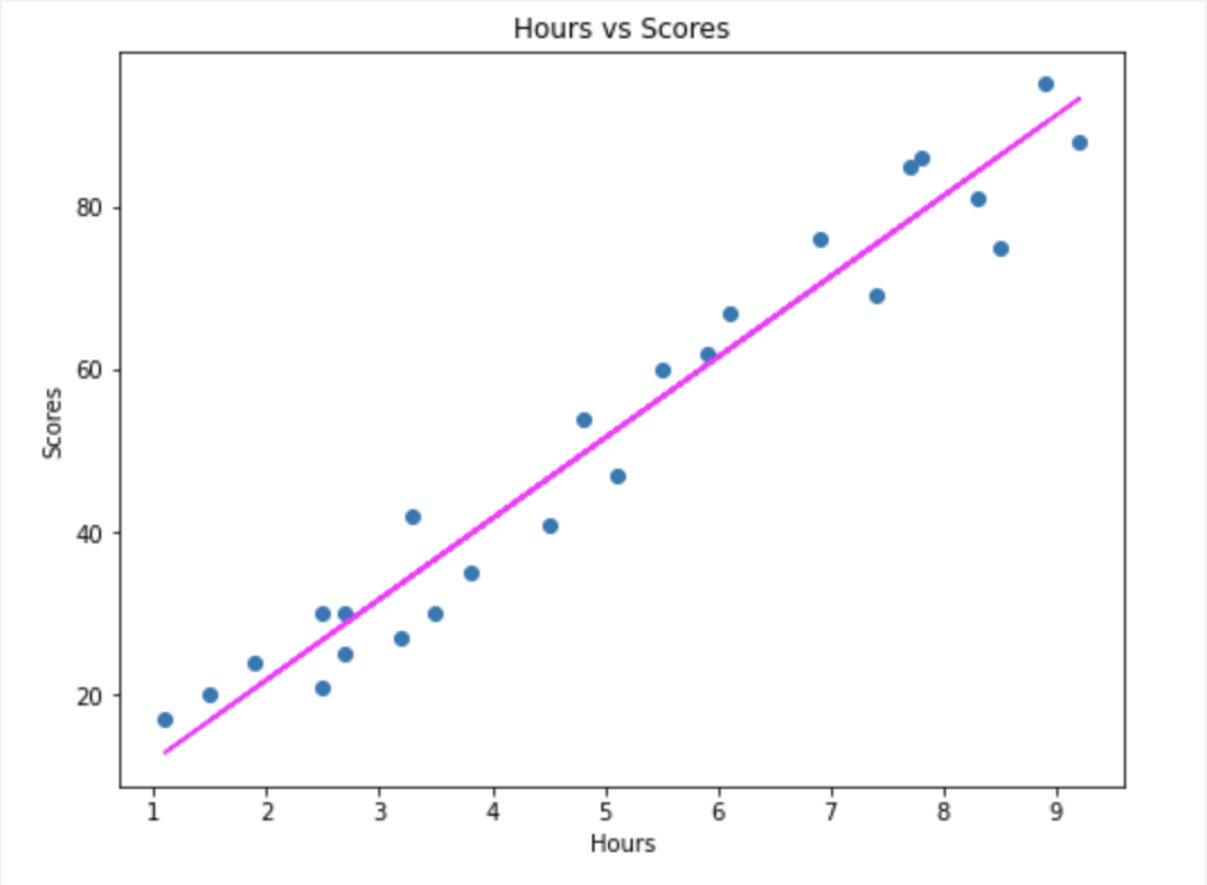
If we recall, Linear Regression was about finding a straight line that best fits our data points. Our model does just that. Using the coef_ and intercept_ commands we can access the coefficient of the X values and the intercept for our line.

```
print("The coefficient is : ",lm.coef_)
print("The intercept is : ",lm.intercept_)
```

```
The coefficient is :  [[9.91065648]]
The intercept is :  [2.01816004]
```

Let us plot the line using these values of coefficient and intercept to see if it fits our data or not.

```
coef = lm.coef_[0][0]
inter = lm.intercept_[0]
plt.figure(figsize=(8,6))
plt.scatter("Hours","Scores",data=df)
plt.plot(X,coef*X + inter,c="magenta")
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Hours vs Scores")
```



Our model has given us a well-fitted line. This line is what is being used for the prediction of our data. For each value of x that we provide, the model calculates the corresponding value of y on this line and that is our predicted value.

Evaluating our Model

We will be using a metric called the **R-Squared Value** *to check the accuracy of our predictions and how well our model works.*

The R-Squared value is a statistical measure of how close our data is to the predicted line of best fit. A value close to 1 tells us that our model works well and a value close to 0 means that our dependent variable is not at all dependent on our other variables.
The R-squared value can also be negative. This happens when our prediction is actually worse than just using the mean value. What this means is — using the regression algorithm, our predictions are more inaccurate than just using the mean value for the independent variable.

To calculate the R-squared value, we use the following mathematical formula:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Image via: https://medium.com/@erika.dauria/looking-at-r-squared-721252709098

Here, we are calculating the residual error (SSres) as the sum of the squares of the errors for each prediction and the total sum of the squared errors (SStot) as the sum of the squares of the distance between our data points and the mean.

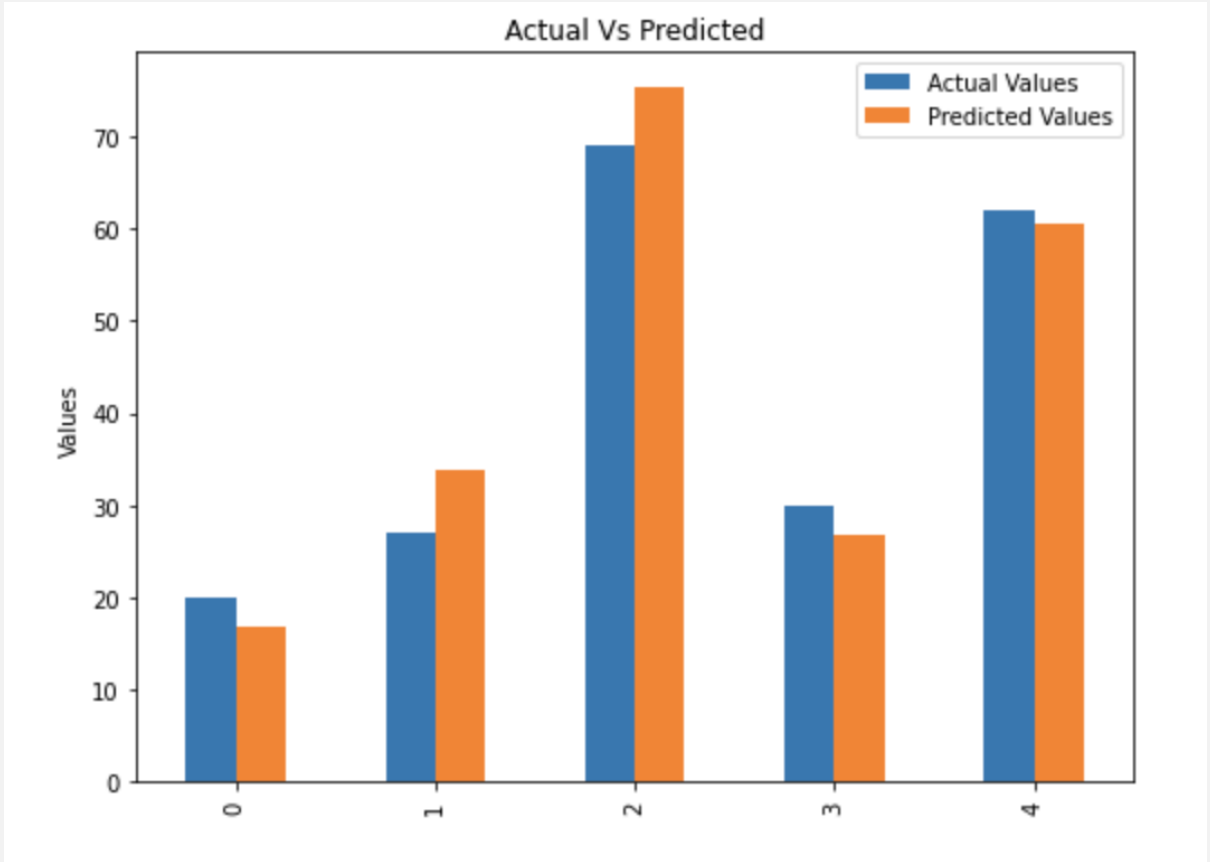
```
score = float(r2_score(y_test,yhat))
print("Our Accuracy is : ",score*100, "%")
```

```
Our Accuracy is :  94.54906892105356 %
```

The advantage of splitting our data into training and testing datasets is that now wee can easily compare our predicted values with the already available values of Scores in our test data. We create a datafrane to compare our values:

	Actual Values	Predicted Values
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

The predicted values are close to our actual values, we can visualise this using a bar graph where we plot the actual values and the predicted values simultaneously.



We can see from the bar graph that using Linear Regression, the prediction made by us are very close to the actual values!