The complete beginner's guide to machine learning: simple linear regression in four lines of code!

A clear and comprehensive blueprint for absolutely anyone who wants to build a simple machine learning model

Anne Bonner Follow

Mar 24 · 12 min read ★

Even you can build a machine learning model.

Seriously!

Good data alone doesn't always tell the whole story. Are you trying to figure out what someone's salary should be based on their years of experience? Do you need to examine how much you're spending on advertising in relation to your yearly sales? Linear regression might be exactly what you need!

GIF via GIPHY

What is linear regression?

> *Linear regression looks at the relationship between the data you have and the data you want to predict.*

Linear Regression is a basic and commonly used type of predictive analysis. It's the most widely used of all statistical techniques. It quantifies the relationship between one or more **predictor variables** and one **outcome variable**.

Linear regression models are used to show (or predict) the relationship between two variables or factors. **Regression analysis** is commonly used to show the correlation between two variables.

You could, for example, look at some information about players on a baseball team and predict how well they might do that season. You might want to examine some variables about a company and predict well their stock might do. You might even just want to examine the number of hours people study and how well they do on a test, or you could look at student's homework grades overall in relation to how well they might do on their tests. It's a seriously useful technique!

Photo by StockSnap via Pixabay

> *Just remember: **correlation is not causation**! Just because a relationship exists between two variables doesn’t mean that one variable caused the other variable! Regression analysis is not used to predict cause-and-effect relationships. It can look at how variables relate to each other. It can examine to what extent variables are associated with each other. It’s up to you to take a closer look at those relationships.*

A couple of important terms:

The variable that the equation in your linear regression model is predicting is called the **dependent variable**. We call that one **y**. The variables that are being used to predict the dependent variable are called the **independent variables.** We call them **X**.

You can think of it as though the prediction (**y**) is dependent on the other variables (**X**). That makes **y** the dependent variable!

In **simple linear regression analysis**, each observation consists of two variables. These are the independent variable and the dependent variable. **Multiple regression analysis** looks at two or more independent variables and how they correlate to the independent variable. The equation that describes how **y** is related to **X** is called the **regression model**!

Regression was first studied in depth by Sir Francis Galton, a man with a wide variety of interests. While he was a very problematic character with a lot of beliefs worth disagreeing with, he did write some books with cool information about things like treating spear wounds and getting your horse unstuck from quicksand. He also did some useful work with fingerprints, hearing tests, and even devised the first weather map. He was knighted in 1909.

While studying data on the relative sizes between parents and their children in plants and animals, he observed that larger-than-average parents have larger-than-average children, but those children will be less large in terms of their relative position within their own generation. He called it **regression towards mediocrity.** That would be **regression to the mean** in modern terms.

(I have to say, though, that there is a certain sparkle to the phrase, "regression towards mediocrity" that I need to work into my day-to-day life…)

GIF via GIPHY

To be clear, though, we're talking about **expectations** (predictions) and not absolute certainty!

What good are regression models?

Regression models are used for predicting a real value, for example, salary or height. If your independent variable is **time**, then you are forecasting future values. Otherwise, your model is predicting present but unknown values. Examples of regression techniques include:

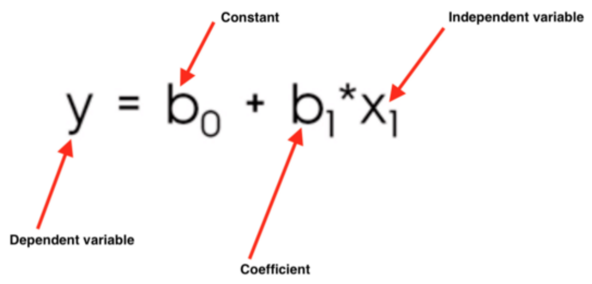
- Simple regression
- Multiple regression
- Polynomial regression
- Support Vector Regression

Let's say you're looking at some data that includes employee's years of experience and salary. You want to look at the correlation between those two figures. Maybe you're running a new business or small company that has been kind of setting the numbers randomly.

So how can you find the correlation between those two variables? In order to figure that out, we'll create a model that will tell us what is the best fitting line for this relationship.

Intuition

Here's a simple linear regression formula:



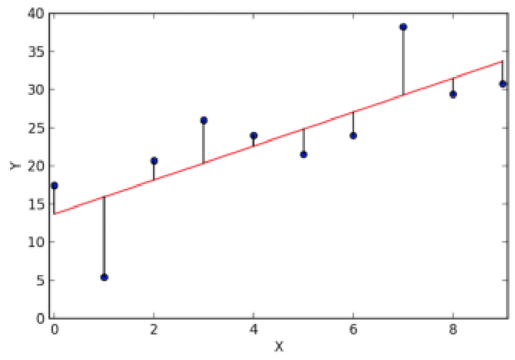
(You might recognize this as the equation for a slope or trend line from high school algebra.)

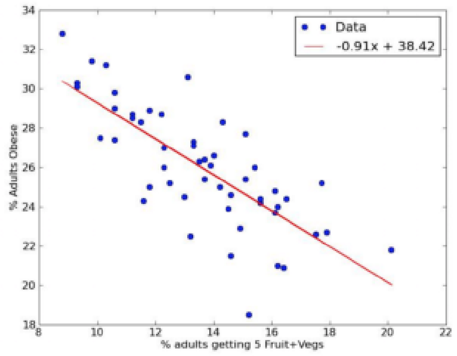
In this equation, **y** is the dependent variable, which is what you're trying to explain. For the rest of this article, **y** will be an employee's salary after a certain number of years of experience.

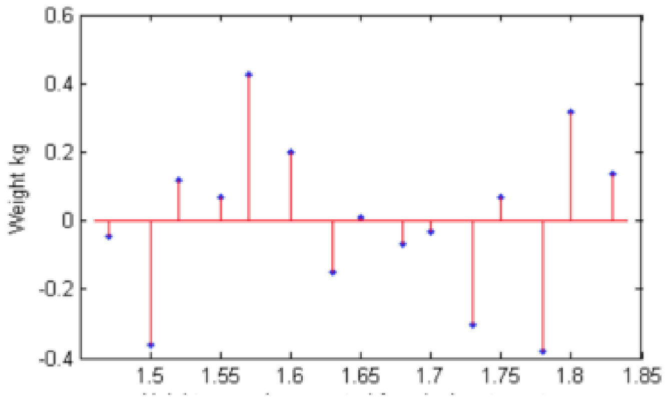
You can see the independent variable above. That's the variable that is associated with the change in your predicted values. The independent variable might be causing the change or simply associated with the change. Remember, **linear regression doesn't prove causation**!

The coefficient is how you explain that a change in your independent variable is maybe not totally equal to a change in y.

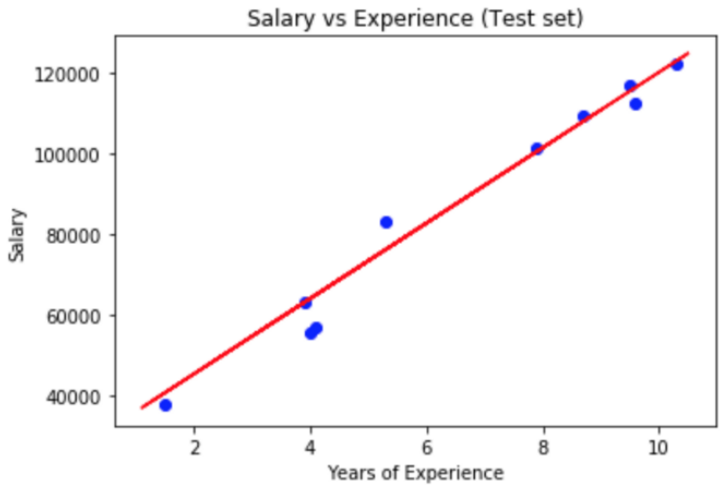
Now we want to look at the evidence. We want to put a line through our data that best fits our data. A regression line can show a positive linear relationship (the line looks like it's sloping up), a negative linear relationship (the line is sloping down), or really no relationship at all (a flat line).







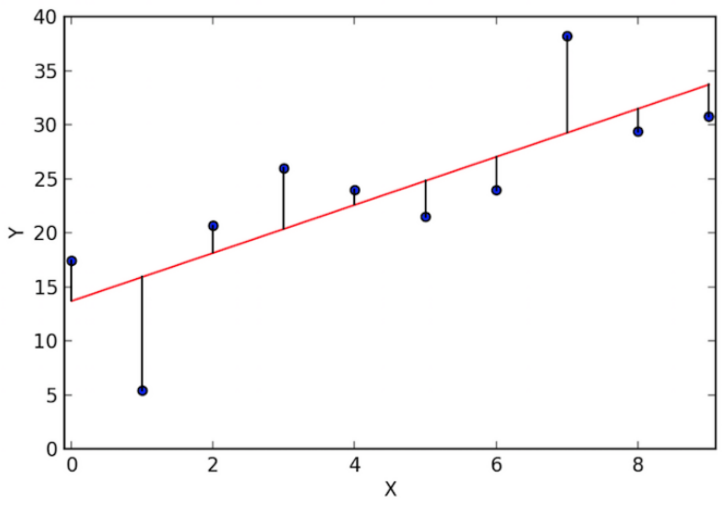
The constant is the point where the line crosses the vertical axis. For example, if you looked at 0 years of experience in the graph below, your salary would be around $30,000. So the constant in the chart below would be about $30,000.



The steeper the slope, the more money you get for your years of experience. For example, maybe with 1 more year of experience, your salary (y) goes up an additional $10,000, but with a steeper slope, you might wind up with more like $15,000. With a negative slope, you’d actually lose money as you gained experience, but I really hope you won’t be working for that company for long…

How does simple linear regression find that line?

When we look at a graph, we can draw vertical lines from the line to our actual observations. You can see the actual observations as the dots, while the line displays the model observations (the predictions).



The line that we drew is the difference between what an employee is actually earning and what he’s modeled (predicted) to be earning. We would look at the **minimum sum of squares** to find the best line, which just means that you'd take the sum of all the squared differences and find the minimum.

That's called the **ordinary least squares** method!

So how do we do that?

First the imports!

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Now let's preprocess our data! If you don't know much about data cleaning and preprocessing, you might want to check out this article. It will walk you through importing libraries, preparing your data, and feature scaling.

The complete beginner's guide to data cleaning and preprocessing

How to successfully prepare your data for a machine learning model in minutes

towardsdatascience.com

We're going to copy and paste the code from that article and make two tiny changes. We'll need to change the name of our dataset, of course. Then we'll take a look at the data. For our example, let's say for our employees we have one column of years of experience and one column of salaries and that’s it. Keeping in mind that our index starts at 0, we will go ahead and separate the last column from our data for the dependent variable, just like we already have set up. This time, however, we’d be grabbing the second column for our independent variable, so we’d make a minor change to grab that.

```
dataset = pd.read_csv('salary.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

Now X is a matrix of features (our independent variable) and y is a vector of the dependent variable. Perfect!

It’s time to split our data into a training set and a test set. Normally, we would do an 80/20 split for our training and testing data. Here, though, we’re working with a small dataset of only 30 observations. Maybe this time we’ll split up our data so that we have 20 training observations and a test size of 10.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 1/3, random_state = 0)
```

You have an X_train, X_test, y_train, and y_test! You’re ready to go! (Never forget that there are about a million things to learn about, change, and improve at every step of this process. The power of your model depends on you and everything that you put into it!)

Photo by Thomas William on Unsplash

We set a random state of 0 so that we can all get the same result. (There can be random factors in calculations, and I want to make sure we're all on the same page so that nobody gets nervous.)

We'll train our model on the training set and then later predict the results based on our information. Our model will **learn** the correlations on the training set. Then we will test what it learned by having it predict values with our test set. We can compare our results with the actual results on the test set to see how our model is doing!

A**lways split your data into training and testing sets**! If you test your results on the same data you used to train it, you'll probably have really great results, but your model isn't good! It just memorized what you wanted it to do, rather than learning anything that it can use with unknown data. That's called overfitting, and it means that you **did not build a good model**!

Feature scaling

We actually don't need to do any feature scaling here!

Photo by Gift Habeshaw on Unsplash

Linear regression

Now we can fit the model to our training set!

We’ll use Scikit-learn learn for this. First, we’ll import the linear model library and the linear regression class. Then we’ll create an object of the class—the regressor. We’ll use a method (the fit method) to fit the regressor object that we create to the training set. To create the object, we name it, then call it using the parenthesis. We can do all of that in about three lines of code!

Let’s import linear regression from Scikit-Learn so that we can go ahead and use it. Between the parenthesis, we’ll specify which data we want to use so our model knows exactly what we want to fit. We want to grab both X_train and y_train because we’re working with all of our training data.

You can look at the documentation if you want more details!

Now we’re ready to create our regressor and fit it to our training data.

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

There it is! We’re using simple linear regression on our data and we’re ready to try out our predictive ability on our test set!

This is machine learning! We created a machine, the regressor, and we had it learn the correlation between years of experience and salary on the training set.

Now it can predict future data based on the information that it has. Our machine is ready to predict a new employee’s salary based on the number of years of experience that the employee has!

Let’s use our regressor to predict new observations. We want to see how the machine has learned by looking at what it does with new observations.

We’ll create a vector of predicted values. This is a vector of predictions of dependent variables that we’ll call y_pred. To do this, we can take the regressor we created and trained and use the predict method. We need to specify which predictions to make, so we want to make sure we include the test set. For our input parameter in regressor.predict, we want to specify the matrix of features of new observations, so we’ll specify X_test.

```
y_pred = regressor.predict(X_test)
```

Seriously. That takes a single line of code!

Now y_test are the real salaries of the 10 observations in the test set and y_pred are the predicted salaries of these 10 employees predicted by our model.

You did it! Linear regression in four lines of code!

GIF via GIPHY

Visualization

Let's visualize the results! We need to see what the difference is between our predictions and the actual results.

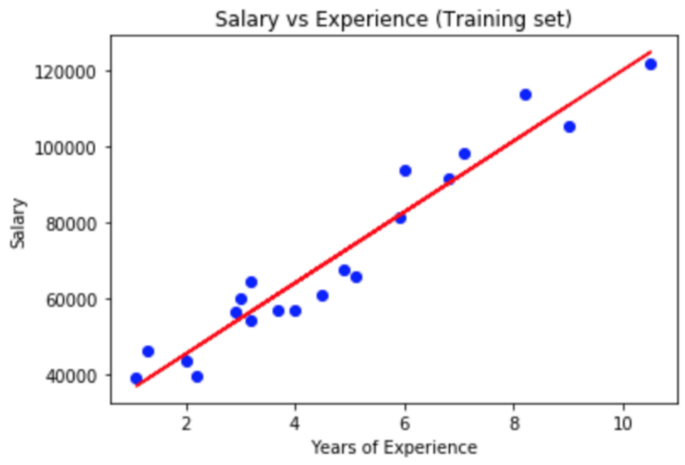
We can plot the graphs in order to interpret the result. First, we can plot the real observations using plt.scatter to make a scatter plot. (We imported matplotlib.pyplot earlier as plt).

We'll look at the training set first, so we'll plot X_train on the X coordinates and y_train on y coordinates. Then we probably want some color. We'll do our observations in blue, and our regression line (predictions) in red. For the regression line we'll use X_train again for the X coordinates, and then the predictions of the X_train observations.

Let's also fancy it up a little with a title and labels for the x-axis and y-axis.

```
plt.scatter(X_train, y_train, color = 'blue')
plt.plot(X_train, regressor.predict(X_train), color = 'red')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

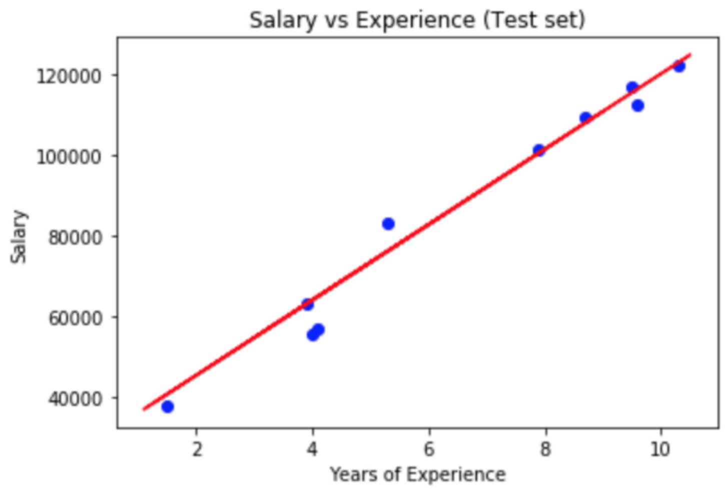
Now we can see our blue points, which are our real values and our predicted values along the red line!



Let's do the same for the test set! We'll change the test set title and change our "train" to "test" in the code.

```
plt.scatter(X_test, y_test, color = 'blue')
plt.plot(X_train, regressor.predict(X_train), color = 'red')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Make sure you notice that we aren’t changing X_train to X_test in the second line. Our regressor is already trained by the training set. When we trained, we obtained one unique model equation. If we replace it, we’ll obtain the same line and we’ll probably build new points of the same regression line.



This is a pretty good model!

Our model is doing a nice job of predicting these new employee salaries. Some of the actual observations are the same as the predictions, which is great. There isn't a 100% dependency between the **y** and **X** variables, so some of the predictions won't be completely accurate.

You did it! You imported libraries, cleaned and preprocessed data, built and trained a simple linear regressor, used it to make predictions, and you even visualized the results!

Congratulations!!!

Photo by Free-Photos via Pixabay

Want more?

Multiple linear regression is up next!

The Complete Beginner's Guide to Machine Learning: Multiple Linear Regression in 4 Line...

Conquer the basics of multiple linear regression (and backward elimination!) and use your data t...

towardsdatascience.com

Keep learning!

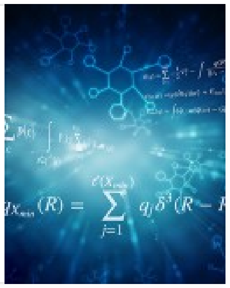
Machine learning is built on statistics and you can't begin to understand machine learning without concepts like the simple linear regressor. But that doesn't mean that statistics and machine learning are the same things! A linear regressor is very much a tool of statistics (and data science), in addition to being a part of the basic building blocks of machine learning.

If you're interested in learning more about the differences between statistics and machine learning, I recommend you check out this fantastic piece by Matthew Stewart, PhD Researcher! He does a beautiful job of clarifying these concepts and I highly recommend you take some time to read through his article.

The Actual Difference Between Statistics and Machine Learning

No, they are not the same. If machine learning is just glorified statistics, then architecture is just...

towardsdatascience.com



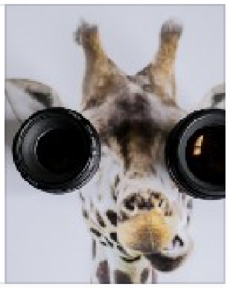
As always, if you're doing anything cool with this information, let people know about it in the responses below or reach out any time on LinkedIn @annebonnerdata!

You might want to check out some of these articles too:

Getting started with Git and GitHub: the complete beginner's guide

Git and GitHub basics for the curious and completely confused (plus the easiest way to...

towardsdatascience.com



How to effortlessly create a website for free with GitHub

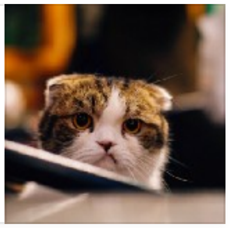
Getting started with GitHub Pages: the unbelievably quick and easy guide for creating a...

towardsdatascience.com

Getting Started With Google Colab

A Basic Tutorial for the Frustrated and Confused

towardsdatascience.com



Intro to Deep Learning

Neural networks for newbies, novices, and neophytes.

towardsdatascience.com

WTF is image classification?

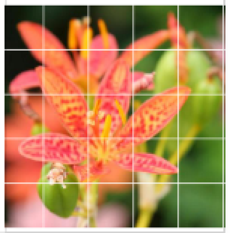
Conquering convolutional neural networks for the curious and confused

towardsdatascience.com

How to build an image classifier with greater than 97% accuracy

A clear and complete blueprint for success

medium.freecodecamp.org



The brilliant beginner's guide to model deployment

A clear and simple roadmap for getting your machine learning model on the Internet and doi...

heartbeat.fritz.ai

Thanks for reading!