Linear Regression

Ankita Bhanushali

Pursuing M.Sc. Statistics

Consultant, Kantar

INSAID , March 2020 GCD Cohort.

Linear Regression

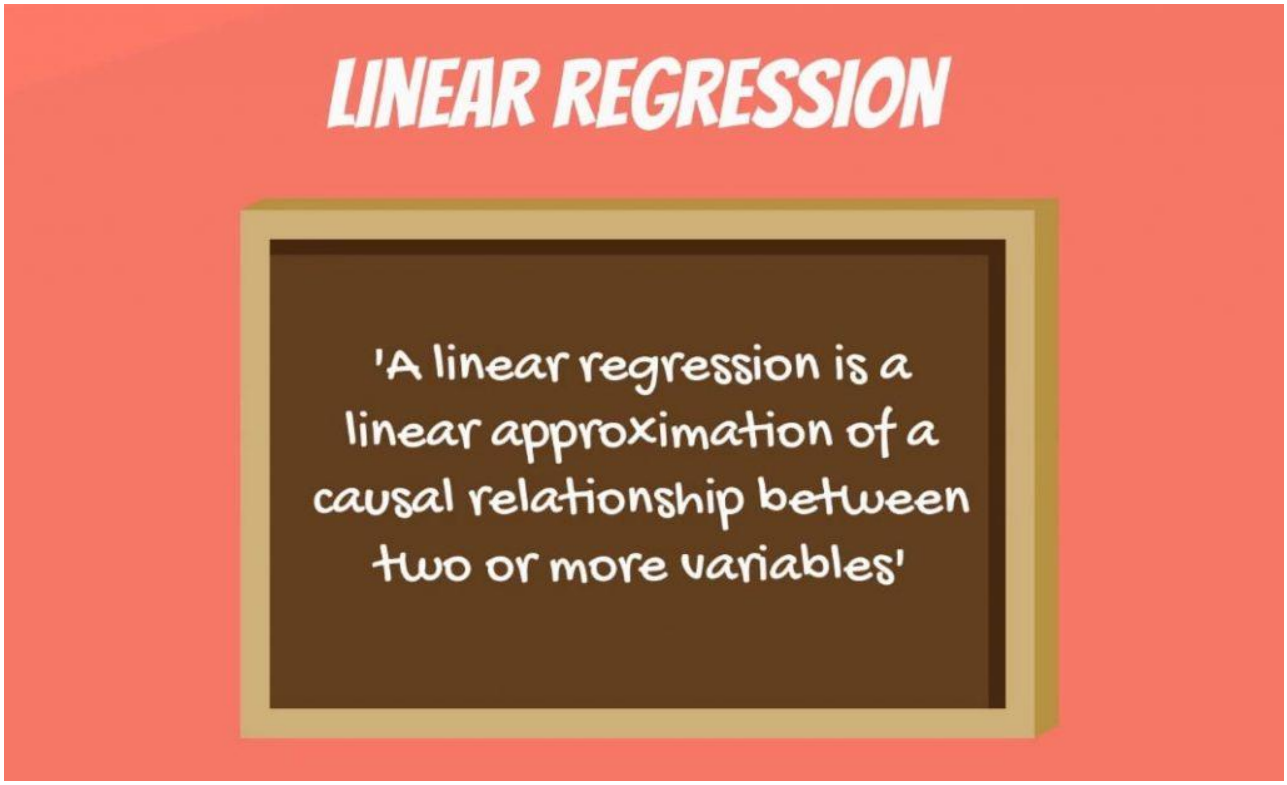
Regression analysis **is one of the most widely used methods for prediction. It is applied whenever we have a causal relationship between variables.**

Regression Analysis

We will use our typical step-by-step approach. We'll start with the **simple linear regression model**.

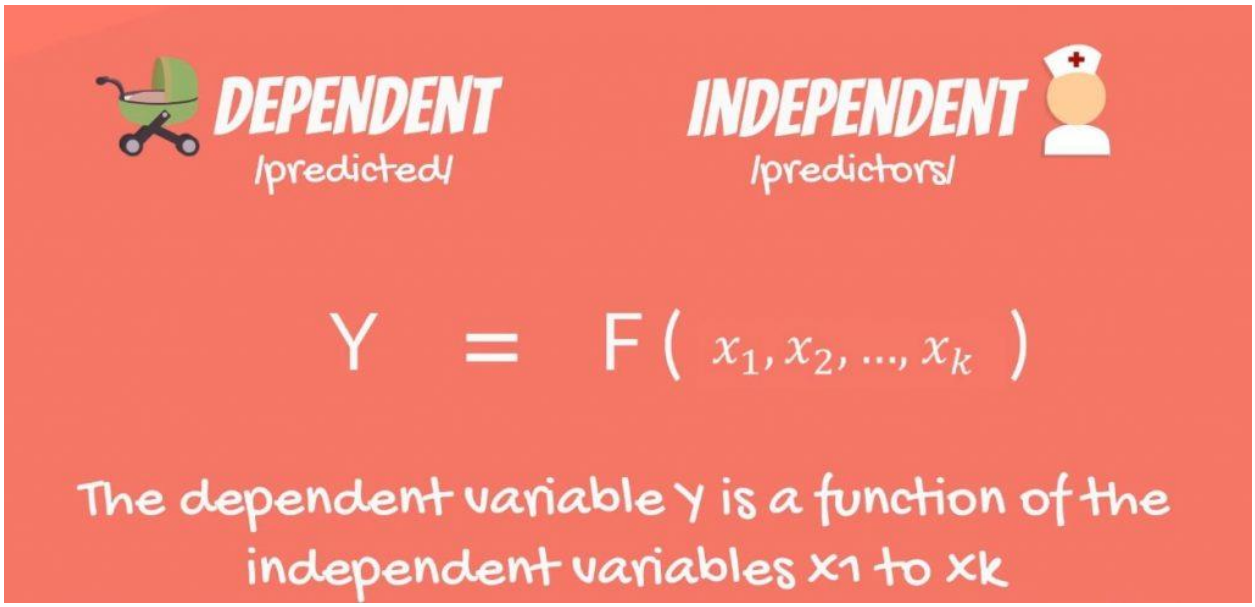
What is a Linear Regression

Let's start with some dry theory. A **linear regression** is a linear approximation of a causal relationship between two or more variables.



Regression models are highly valuable, as they are one of the most common ways to make inferences and predictions.

There is a dependent variable, labeled *Y*, being predicted, and independent variables, labeled *x1*, *x2*, and so forth. These are the predictors. *Y* is a function of the *X* variables, and the **regression model** is a linear approximation of this function.

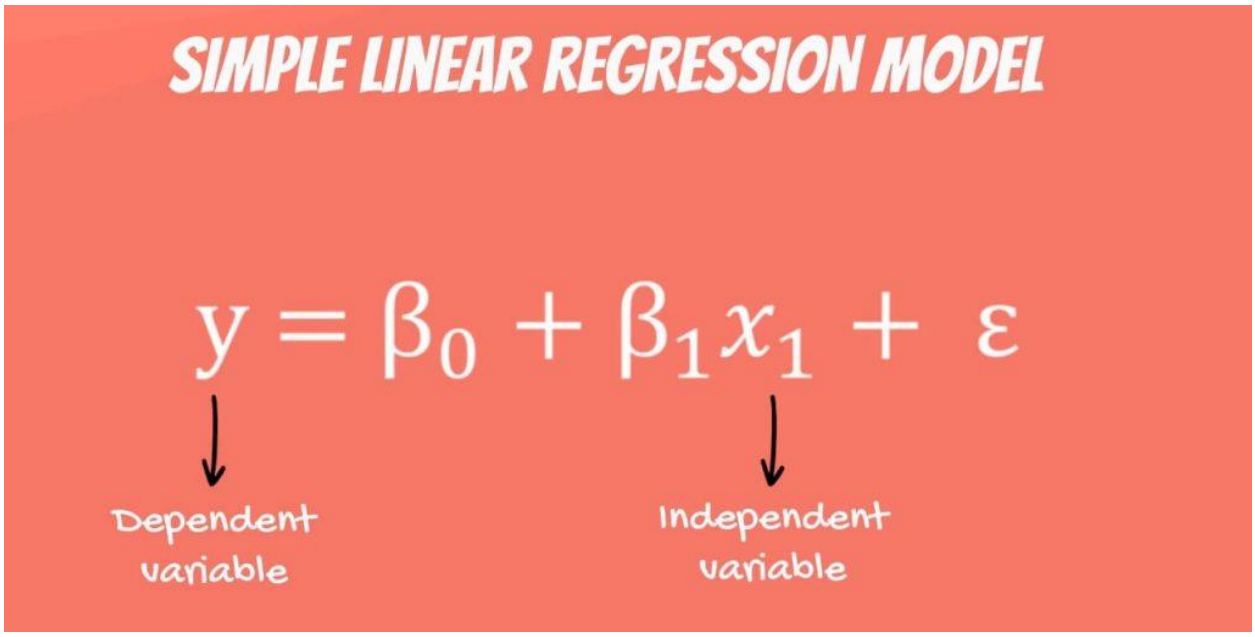


The Simple Linear Regression

The easiest regression model **is the** simple linear regression**:**

$Y = \beta_0 + \beta_1 * x_1 + \varepsilon$.

Let's see what these values mean. *Y* is the variable we are trying to predict and is called the *dependent variable*. *X* is an *independent variable*.



The Regression Line

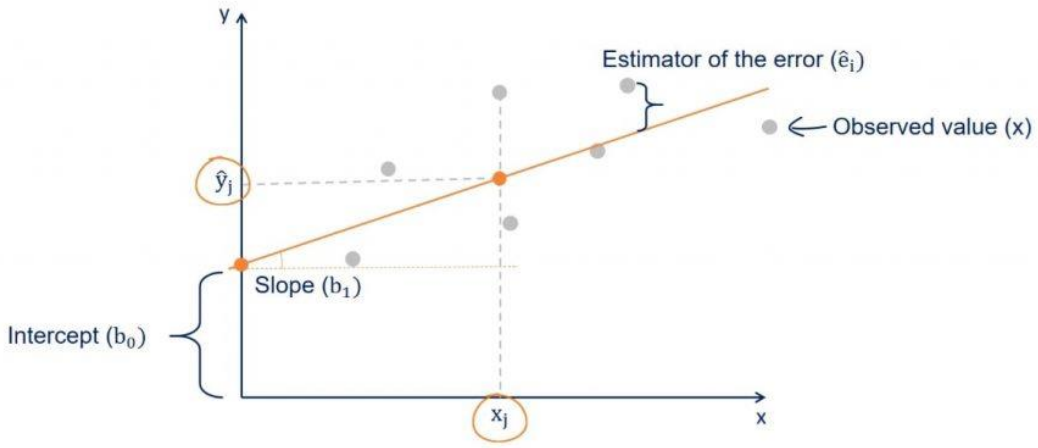
You may have heard about the **regression line**, too. When we plot the data points on an *x-y* plane, the **regression line** is the best-fitting line through the data points. You can take a look at a plot with some data points in the picture above. We plot the line based on the **regression equation**. The grey points that are scattered are the observed values. B_0, as we said earlier, is a *constant* and is the intercept of the **regression line** with the y-axis.B_1 is the slope of the **regression line**. It shows how much *y* changes for each unit change of *x*.

The Estimator of the Error

The distance between the observed values and the **regression line** is the *estimator of the error term epsilon*. Its **point estimate** is called residual. Now, suppose we draw a perpendicular from an observed point to the **regression line**. The intercept between that perpendicular and the **regression line** will be a point with a *y* value equal to $\hat{y}$. As we said earlier, given an *x*, $\hat{y}$ is the value predicted by the **regression line**.

Linear regression model. Geometrical representation

$$\hat{y}_i = b_0 + b_1 x_i$$



Linear Regression in Python Example

We believe it is high time that we actually got down to it and wrote some code! So, let's get our hands dirty with our first **linear regression** example in Python.

Understanding the Dataset

Before we get started with the Python linear regression hands-on, let us explore the dataset. We will be using the Boston House Prices Dataset, with 506 rows and 13 attributes with a target column. Let's take a quick look at the dataset.

Let's take a quick look at the dataset.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
14	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
15	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
16	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
17	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
18	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1
19	0.7842	0	8.14	0	0.538	5.99	81.7	4.2579	4	307	21	386.75	14.67	17.5

In this Python Linear Regression example, we will train two models to predict the price.

Model Building

Now that we are familiar with the dataset, let us build the Python linear regression models.

Simple Linear Regression in Python

Consider 'lstat' as independent and 'medv' as dependent variables

Step 1: Load the Boston dataset

```
In [1]: import pandas as pd
        data = pd.read_csv('Boston1.csv')
```

Step 2: Have a glance at the shape

```
In [4]: data.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

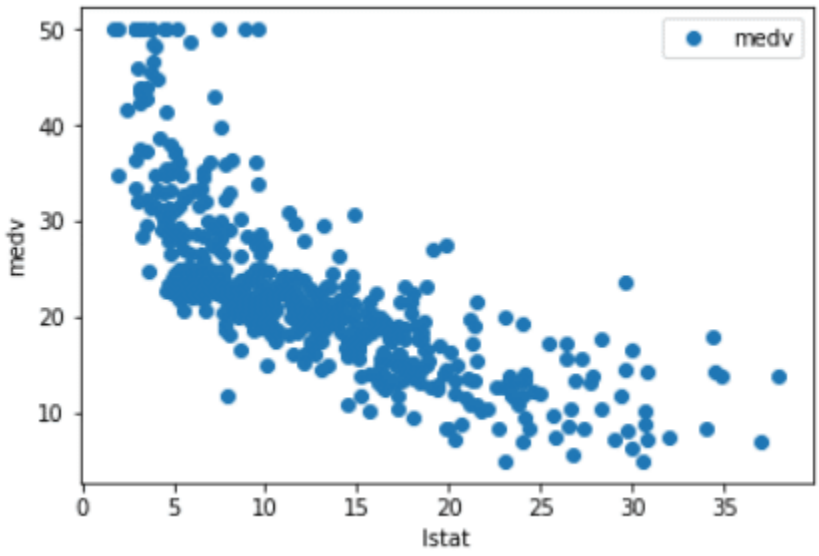
Step 3: Have a glance at the dependent and independent variables

```
In [6]: data_ = data.loc[:,['lstat','medv']]
        data_.head(5)
```

	lstat	medv
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

Step 4: Visualize the change in the variables

```
In [10]: import matplotlib.pyplot as plt
         data.plot(x='lstat',y='medv',style='o')
         plt.xlabel('lstat')
         plt.ylabel('medv')
         plt.show()
```



Step 5: Divide the data into independent and dependent variables

```
In [13]: X = pd.DataFrame(data['lstat'])
         y = pd.DataFrame(data['medv'])
```

Step 6: Split the data into train and test sets

```
In [14]: 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=1)
```

Step 7: Shape of the train and test sets

```
In [16]: print(X_train.shape)
         print(X_test.shape)
         print(y_train.shape)
         print(y_test.shape)
```

```
(404, 1)
(102, 1)
(404, 1)
(102, 1)
```

Step 8: Train the algorithm

```
In [11]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train, y_train)
```

Step 9: Retrieve the intercept

```
In [12]: print(regressor.intercept_)
```

```
[34.33497839]
```

Step 10: Retrieve the slope

```
In [13]: print(regressor.coef_)
```

```
[[-0.92441715]]
```

Step 11: Predicted value

```
In [23]: y_pred
```

Predicted
27.374117
27.697663
16.955936
26.847199
24.915168
24.055460
29.990218

Step 12: Actual value

```
In [25]: y_test
```

medv
28.2
23.9
16.6
22.0
20.8
23.0
27.9

Step 13: Evaluate the algorithm

```
In [26]: from sklearn import metrics
         import numpy as np
         print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
         print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
         print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 5.078127727696938
Mean Squared Error: 46.99482091954711
Root Mean Squared Error: 6.855276866731723
```

What Did We Learn?

We embarked on it by first learning about what a **linear regression** is. Then, we went over the process of creating one. We also went over a **linear regression** example. Afterwards, we talked about the **simple linear regression** where we introduced the **linear regression equation**. By then, we were done with the theory and got our hands on the keyboard and explored another **linear regression** example in Python! We imported the relevant libraries and loaded the data. We cleared up when exactly we need to create **regressions** and started creating our own. The process consisted of several steps which, now, you should be able to perform with ease.

Thank You!