**What is Simple Linear Regression?**

**Simple linear regression** is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:

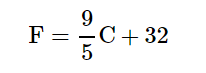
* One variable, denoted *x*, is regarded as the **predictor**, **explanatory**, or **independent** variable.
* The other variable, denoted *y*, is regarded as the **response**, **outcome**, or **dependent** variable.

Simple linear regression gets its adjective "simple," because it concerns the study of only one predictor variable. In contrast, multiple linear regression, gets its adjective "multiple," because it concerns the study of two or more predictor variables.

### **Types of relationships**

1. **Deterministic** (or **functional**) **relationships**

The relationship between degrees Fahrenheit and degrees Celsius is known to be:



That is, if you know the temperature in degrees Celsius, you can use this equation to determine the temperature in degrees Fahrenheit ***exactly*.**

Here are some examples of other deterministic relationships:

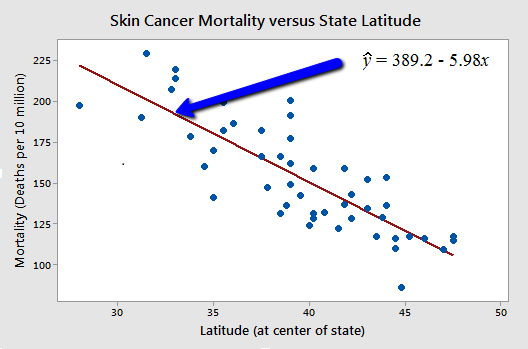
* Circumference = π × diameter
* Ohm's Law: *I* = *V*/*r*, where *V* = voltage applied, *r* = resistance, and *I* = current.

For each of these deterministic relationships, the equation ***exactly*** describes the relationship between the two variables.

1. **Statistical relationships**, in which the relationship between the variables is not perfect.

Here is an example of a statistical relationship.

The response variable *y* is the mortality due to skin cancer (number of deaths per 10 million people) and the predictor variable *x* is the latitude (degrees North) at the center of each of 49 states in the U.S.



You might anticipate that if you lived in the higher latitudes of the northern U.S., the less exposed you'd be to the harmful rays of the sun, and therefore, the less risk you'd have of death due to skin cancer.

The scatter plot supports such a hypothesis.

There appears to be a negative linear relationship between latitude and mortality due to skin cancer, but the relationship is not perfect.

Indeed, the plot exhibits some "**trend**," but it also exhibits some "**scatter**." Therefore, it is a statistical relationship, not a deterministic one.

Some other examples of statistical relationships might include:

* Height and weight — as height increases, you'd expect weight to increase, but not perfectly.
* Alcohol consumed and blood alcohol content — as alcohol consumption increases, you'd expect one's blood alcohol content to increase, but not perfectly.
* Driving speed and gas mileage — as driving speed increases, you'd expect gas mileage to decrease, but not perfectly.

**Regression models** describe the relationship between variables by fitting a line to the observed data.

Linear regression models use a straight line,

**while**

logistic and nonlinear regression models use a curved line.

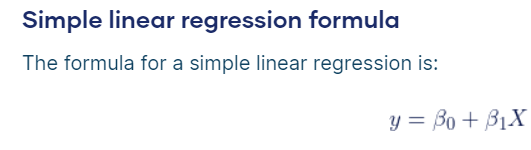
Regression allows you to estimate how a [dependent variable](https://www.scribbr.com/methodology/independent-and-dependent-variables/#dependent) changes as the independent variable(s) change.

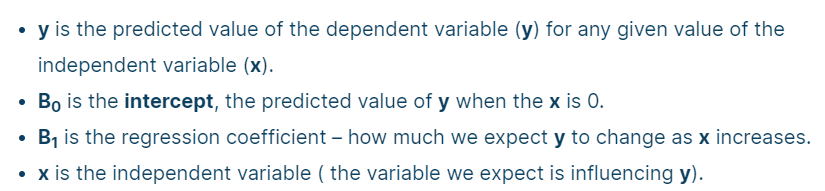
## **Assumptions of simple linear regression**

1. Homogeneity of variance (homoscedasticity): the size of the error in our prediction doesn’t change significantly across the values of the independent variable.
2. Independence of observations: the observations in the dataset were collected using statistically valid [sampling methods](https://www.scribbr.com/methodology/sampling-methods/), and there are no hidden relationships among observations.
3. Normality: The data follows a [normal distribution](https://www.scribbr.com/statistics/normal-distribution/).

Linear regression makes one additional assumption:

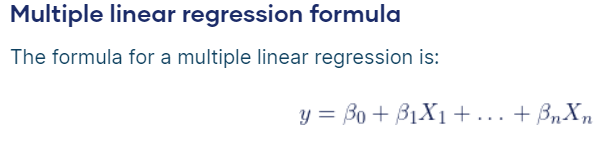
1. The relationship between the independent and dependent variable is linear: the line of best fit through the data points is a straight line (rather than a curve or some sort of grouping factor).

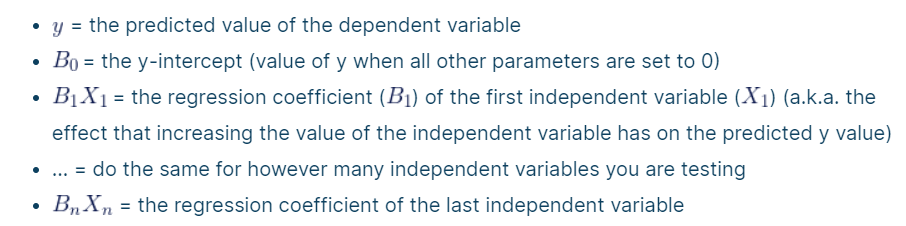




**Multiple linear regression** is used to estimate the relationship between two or more independent variables and one dependent variable. You can use multiple linear regression when you want to know:

1. How strong the relationship is between two or more [independent variables](https://www.scribbr.com/methodology/independent-and-dependent-variables/#independent) and one dependent variable (e.g. how rainfall, temperature, and amount of fertilizer added affect crop growth).
2. The value of the [dependent variable](https://www.scribbr.com/methodology/independent-and-dependent-variables/#dependent) at a certain value of the independent variables (e.g. the expected yield of a crop at certain levels of rainfall, temperature, and fertilizer addition).





a.k.a stands for also known as

<https://medium.com/@shuv.sdr/simple-linear-regression-in-python-a0069b325bf8>

# Implement Simple Linear Regression in Python

In this example, we will use the [salary data](https://www.kaggle.com/datasets/karthickveerakumar/salary-data-simple-linear-regression) concerning the experience of employees. In this dataset, we have two columns *YearsExperience* and *Salary*

## **Step 1: Import the required python packages**

We need *Pandas* for data manipulation, *NumPy* for mathematical calculations, and *MatplotLib, and Seaborn*for visualizations. *Sklearn* libraries are used for machine learning operations

# Import libraries  
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from pandas.core.common import random\_state

from sklearn.linear\_model import LinearRegression

**sklearn** doesn’t work so install as seen below

py -m pip install **scikit-learn**

## **Step 2: Load the dataset**

**# Get dataset**

df\_sal = pd.read\_csv('C:/biju 23-24 new laptop/TY IDS/ch 6/Salary\_Data.csv')

df\_sal.head()



## **Step 3: Data analysis**

# Describe data

df\_sal.describe()



Here, we can see Salary ranges from 37731 to 122391 and a median of 65237.  
We can also find how the data is distributed visually using *Seaborn* distplot

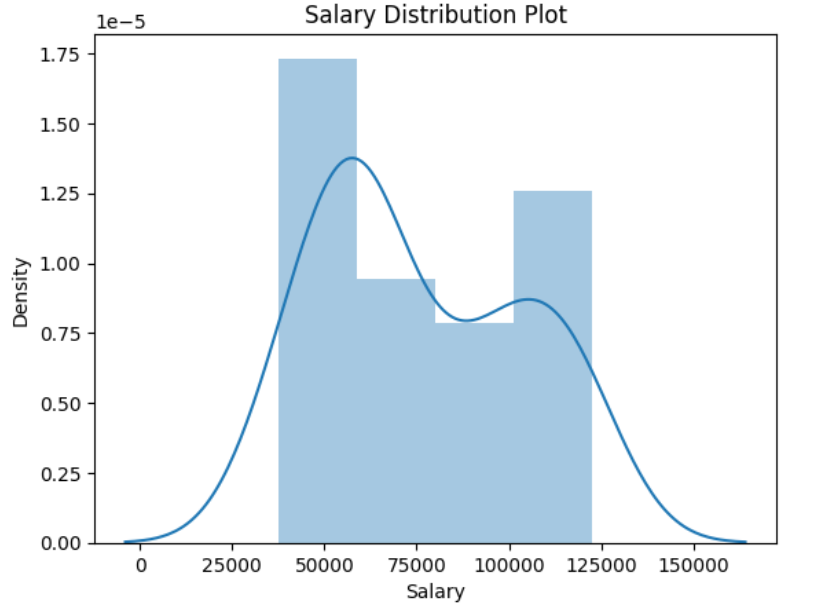
A distplot or distribution plot shows the variation in the data distribution.  
It represents the data by combining a line with a histogram.

# Data distribution

plt.title('Salary Distribution Plot')

sns.distplot(df\_sal['Salary'])

plt.show()



In the resulting plot, the histogram shows the discrete distribution of data points, while the smooth curve (kernel density estimate) provides a continuous representation of the data distribution.

Then we check the relationship between *Salary* and *Experience –*

# Relationship between Salary and Experience

plt.scatter(df\_sal['YearsExperience'], df\_sal['Salary'], color = 'green')

//color can be 'red', 'green', 'blue', 'yellow' etc.

plt.title('Salary vs Experience')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.box(False)

// plt.box(False) to hide the box around a plot.

plt.show()



It is clearly visible now, our data varies linearly. That means, that an individual receives more *Salary* as they gain *Experience*.

**Step 4: Split the dataset into dependent/independent variables**

*Experience* *(X)* is the independent variable  
*Salary* *(y)* is dependent on experience

# Splitting variables

X = df\_sal.iloc[:, :1] # independent

* The first colon : before the comma (:) refers to all rows in the DataFrame.
* The second part :1 after the comma (:) refers to the columns. It selects all columns from the beginning (index 0) up to, but not including, index 1.

y = df\_sal.iloc[:, 1:] # dependent

* The first colon : before the comma (:) refers to all rows in the DataFrame.
* The second part 1: after the comma (:) refers to the columns. It selects all columns starting from index 1 to the end of the DataFrame.

## **Step 4: Split data into Train/Test sets**

Further, split your data into training (80%) and test (20%) sets using ***train\_test\_split***

# Splitting dataset into test/train

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

The function train\_test\_split is used to split your data into two sets: one for training a machine learning model and the other for testing or evaluating the model. The parameters are:

* **X:** The features.
* **y:** The target variable.
* **test\_size:** This parameter determines the proportion of the dataset to include in the test split. In this case, test\_size=0.2 means that 20% of the data will be used for testing, and the remaining 80% will be used for training.
* **random\_state:** This is an optional parameter that allows you to set a seed for the random number generator. This ensures reproducibility. If you use the same random\_state, you will get the same split every time you run the code.

After running this code, you will have four sets:

* X\_train: The training set of features.
* X\_test: The testing set of features.
* y\_train: The training set of target values.
* y\_test: The testing set of target values.

You can then use X\_train and y\_train to train your machine learning model and evaluate its performance on the unseen data using X\_test and y\_test.

## **Step 5: Train the regression model**

Pass the *X\_train* and *y\_train* data into the regressor model by *regressor.fit*to train the model with our training data.

# Regressor model

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

## **Step 6: Predict the result**

Here comes the interesting part, when we are all set and ready to predict any value of *y (Salary)* dependent on *X (Experience)* with the trained model using *regressor.predict*

# Prediction result

y\_pred\_test = regressor.predict(X\_test) # predicted value of y\_test

y\_pred\_train = regressor.predict(X\_train) # predicted value of y\_train

## **Step 7: Plot the training and test results**

Its time to test our predicted results by plotting graphs

* **Plot training set data vs predictions**  
  First we plot the result of training sets *(X\_train, y\_train)* with *X\_train* and predicted value of *y\_train* *(regressor.predict(X\_train))*

# Prediction on training set

plt.scatter(X\_train, y\_train, color = 'lightcoral')

So, the scatter plot will display individual data points where each point is represented by a marker at coordinates (X\_train[i], y\_train[i]). The x-axis will have the values from X\_train, and the y-axis will have the corresponding values from y\_train. The markers will be light coral in color. Scatter plots are often used to visualize the distribution and relationships between two variables.

plt.plot(X\_train, y\_pred\_train, color = 'green')

// creates a line plot using the plot function

So, the line plot will display the relationship between X\_train and y\_pred\_train using a green line. The x-axis will have the values from X\_train, and the y-axis will have the corresponding predicted values from y\_pred\_train. This kind of plot is often used to visualize how well a model's predictions align with the actual data.

plt.title('Salary vs Experience (Training Set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.legend(['X\_train/Pred(y\_test)', 'X\_train/y\_train'], title = 'Sal/Exp', loc='best', facecolor='white')

plt.box(False)

plt.show()

The list ['X\_train/Pred(y\_test)', 'X\_train/y\_train'] is a collection of labels that you would typically use in the legend of a plot to identify the different series or data sets that have been plotted.

1. 'X\_train/Pred(y\_test)':
   * This label likely represents a series of data points or a line plot derived from the training data (X\_train) and the corresponding predictions made by a model on the test data (Pred(y\_test)). It signifies the relationship or comparison between the input features (X\_train) and the model's predictions on the test set (Pred(y\_test)).
2. 'X\_train/y\_train':
   * This label likely represents another series of data points or a line plot derived from the training data (X\_train) and the actual target values (y\_train). It signifies the relationship between the input features (X\_train) and the true output values (y\_train) from the training set.



**Plot test set data vs predictions**  
Secondly, we plot the result of test sets *(X\_test, y\_test)* with *X\_train* and predicted value of *y\_train (regressor.predict(X\_train))*

# Prediction on test set

plt.scatter(X\_test, y\_test, color = 'yellow')

plt.plot(X\_train, y\_pred\_train, color = 'blue')

plt.title('Salary vs Experience (Test Set)')

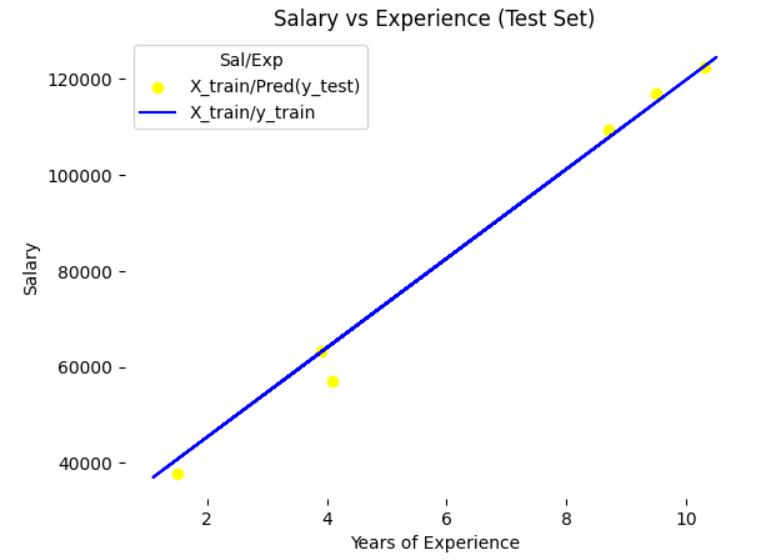
plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.legend(['X\_train/Pred(y\_test)', 'X\_train/y\_train'], title = 'Sal/Exp', loc='best', facecolor='white')

plt.box(False)

plt.show()



**We can see, in both plots, the regressor line covers train and test data.**

**In the linear equation *y = mx + c*, we can also get the *c* *(y intercept)* and *m* *(slope/coefficient)*from the regressor model.**

# Regressor coefficients and intercept

print(f'Coefficient: {regressor.coef\_}')

print(f'Intercept: {regressor.intercept\_}')

Coeff**icien**t: [[9312.57512673]]

Intercept: [26780.09915063]

**Testing Accuracy of the Model**

from sklearn.metrics import r2\_score

score = r2\_score(y\_train,y\_pred\_train )

print("The accuracy of our model is {}%".format(round(score, 2) \*100))

The accuracy of our model is 94.0%

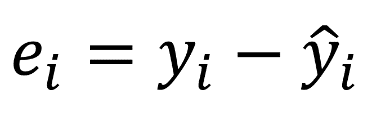
**The value of R Square is 94.0, which indicates that 94.0% of the data fit the regression model.**

## **What are Residuals?**

It is not ideal or possible for a model to accurately predict the value of a continuous variable in a regression problem. A regression model can only predict values that are lower or higher than the actual value. As a result, the only way to determine the model’s accuracy is through residuals.

Residuals are the difference between the actual and predicted values. You can think of residuals as being a distance. So, the closer the residual is to zero, the better our model performs in making its predictions.

Here's the formula for calculating residuals:



In the above formula:

ei -- stands for the residual value.

yi -- stands for the actual value.

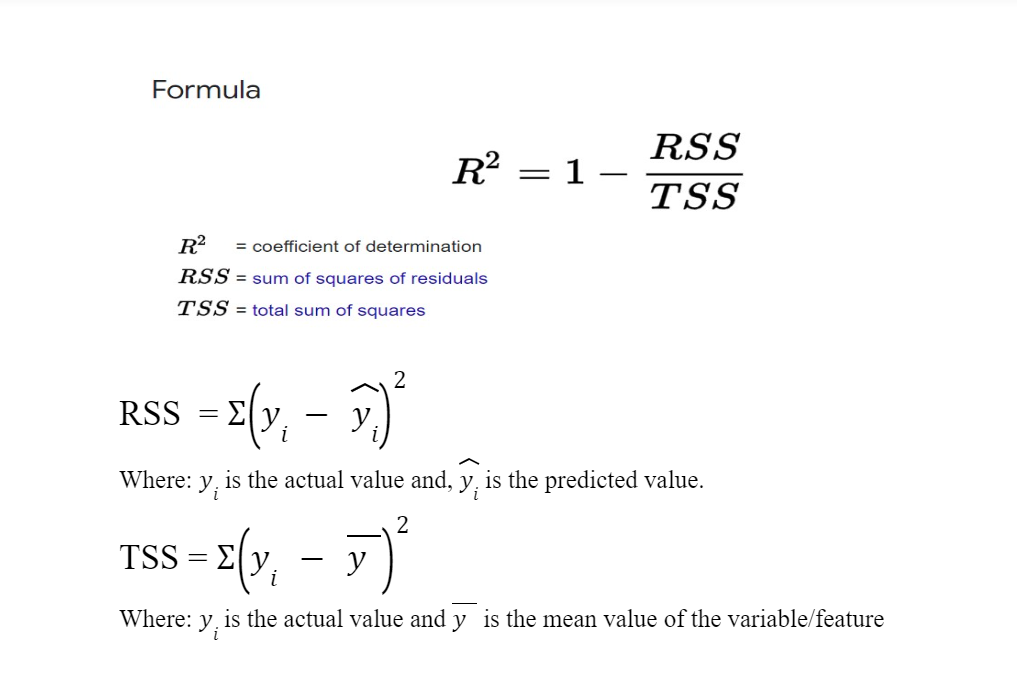
y^i -- stands for the predicted value.

So say, for instance, that the actual value in the dataset is 5 and the predicted value is 8. The residual value will be -3.

### **R2 Score**

The R2 score (pronounced R-Squared Score) is a statistical measure that tells us how well our model is making all its predictions on a scale of zero to one.

As mentioned above, it's not ideal for a model to predict the actual values in a regression problem But we can use the R2 score to determine the accuracy of our model in terms of distance or residual. You can calculate the R2 score using the formula below:



#### When to Use the R2 Score

You can use the R2 score to get the accuracy of your model on a percentage scale, that is 0–100.

**In simple terms, linear regression is a potent supervised machine learning approach that enables us to predict linear correlations between two variables.**