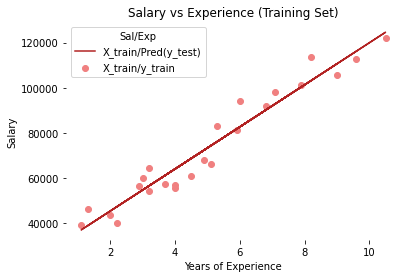
**What is Simple Linear Regression?**

In statistics, simple linear regression is a linear regression model with a single explanatory variable. In simple linear regression, we predict scores on one variable based on results on another. The criteria variable Y is the variable we are predicting. Predictor variable X is the variable using which we are making our predictions. The prediction approach is known as simple regression as there is only one predictor variable,

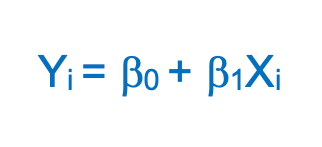
As a result, a linear function that predicts the values of the dependent variable as a function of the independent variable is discovered for two-dimensional sample points with one independent variable and one dependent variable.

The below graph explains the relation between Salary and Years of Experience



**Equation : y = mx + c**

This is the simple linear regression equation where ***c*** is the *constant* and ***m*** is the *slope* and describes the relationship between ***x****(independent variable)* and ***y****(dependent variable)*. The coefficient can be positive or negative and is the degree of change in the dependent variable for every 1 unit of change in the independent variable.



**β0**(y-intercept) and **β1**(slope)are the coefficients whose values represent the accuracy of predicted values with the actual values.

**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

**Step 2: Load the dataset**

Download the dataset from [here](https://www.kaggle.com/datasets/karthickveerakumar/salary-data-simple-linear-regression) and upload it to your notebook and read it into the pandas dataframe.

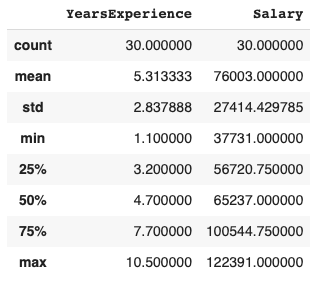
# Get dataset  
df\_sal = pd.read\_csv('/content/Salary\_Data.csv')  
df\_sal.head()



**Step 3: Data analysis**

Now that we have our data ready, let's analyze and understand its trend in detail. To do that we can first describe the data below -

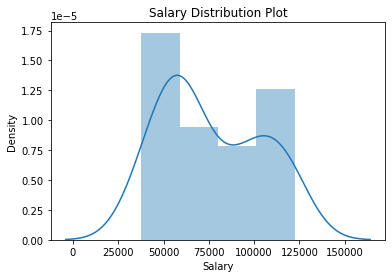
# Describe data  
df\_sal.describe()



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

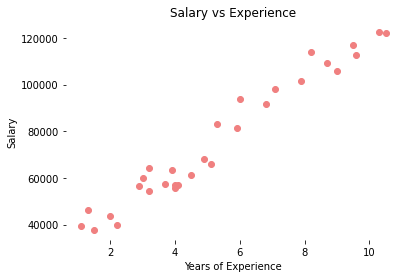
# Data distribution  
plt.title('Salary Distribution Plot')  
sns.distplot(df\_sal['Salary'])  
plt.show()



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

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

# Relationship between Salary and Experience  
plt.scatter(df\_sal['YearsExperience'], df\_sal['Salary'], color = 'lightcoral')  
plt.title('Salary vs Experience')  
plt.xlabel('Years of Experience')  
plt.ylabel('Salary')  
plt.box(False)  
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  
y = df\_sal.iloc[:, 1:] # dependent

**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)

**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***

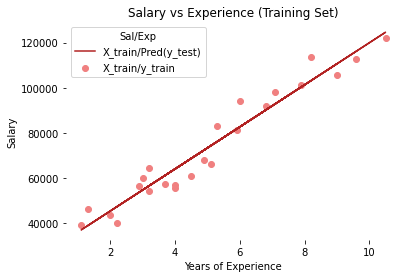
# Prediction result  
y\_pred\_test = regressor.predict(X\_test) # predicted value of y\_test  
y\_pred\_train = regressor.predict(X\_train) # predicted value of y\_train

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

Its time to test our predicted results by plotting graphs

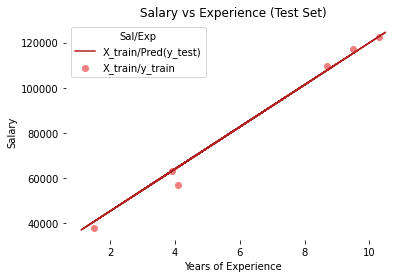
* **Plot training set data vs predictions**First we plot the result of training sets ***(X\_train, y\_train)*** with ***X\_train*** and predicted value of ***y\_train*** ***(regressor.predict(X\_train))***

# Prediction on training set  
plt.scatter(X\_train, y\_train, color = 'lightcoral')  
plt.plot(X\_train, y\_pred\_train, color = 'firebrick')  
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()



* **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 = 'lightcoral')  
plt.plot(X\_train, y\_pred\_train, color = 'firebrick')  
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.

Also, you can plot results with the predicted value of y\_test (***regressor.predict(X\_test))***but the regression line would remain the same at it is generated from the unique equation of linear regression with the same training data.

If you remember from the beginning of this article, we discussed 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\_}')

https://miro.medium.com/v2/resize:fit:700/1*2CzMgU0XCJdZMgsoRNayUA.png