EC3357:Machine Learning

Lecture 3: Linear Regression

Supervised Learning – Linear Regression

- The focus of supervised learning revolves around the input and output variables using an algorithm to predict the outcome, if a new input variable comes into the picture.
- The linear regression algorithm in machine learning is a supervised learning technique to approximate the mapping function to get the best predictions.

What is Regression?

- The main goal of regression is the construction of an efficient model to predict the dependent attributes from a bunch of attribute variables.
- A regression problem is when the output variable is either real or a continuous value i.e salary, weight, area, etc.
- It is used to predict the relationship between a dependent variable and a bunch of independent variables.

Definitions:

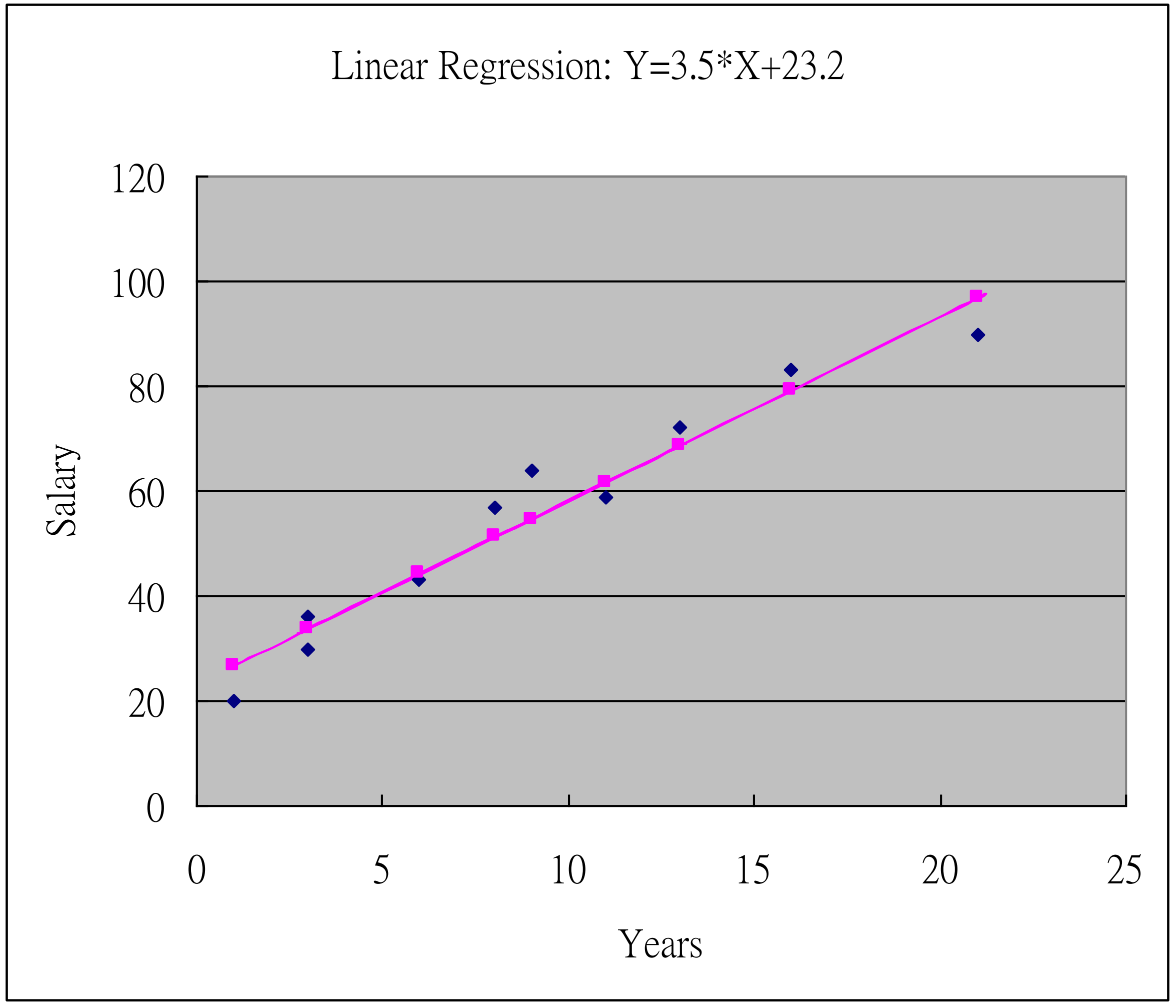
- The variable that researchers are trying to explain or predict is called the response variable. It is also sometimes called the dependent variable because it depends on another variable.
- The variable that is used to explain or predict the response variable is called the explanatory variable. It is also sometimes called the independent variable because it is independent of the other variable.
- In regression, the order of the variables is very important.
- The explanatory variable (or the independent variable) always belongs on the x-axis.
- The response variable (or the dependent variable) always belongs on the y-axis.

Linear Regression – Simple example

- Given one variable (X)
- Goal: Predict Y
- Example:
  - Given Years of Experience
  - Predict Salary
- Questions:
  - When X=10, what is Y?
  - When X=25, what is Y?
  - This is known as regression

X (years)	Y (salary, $1,000)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20

Linear Regression Example



Basic Idea

- Linear equation

$$Y = \alpha + \beta X$$

$$\beta = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$\alpha = \bar{y} - \beta \overline{x}$$

For the example data

$$\alpha = 23.2,$$

$$\beta = 3.5$$

$$y = 23.2 + 3.5x$$

Thus, when x=10 years, prediction of y (salary) is: 23.2+35=58.2 K dollars/year.

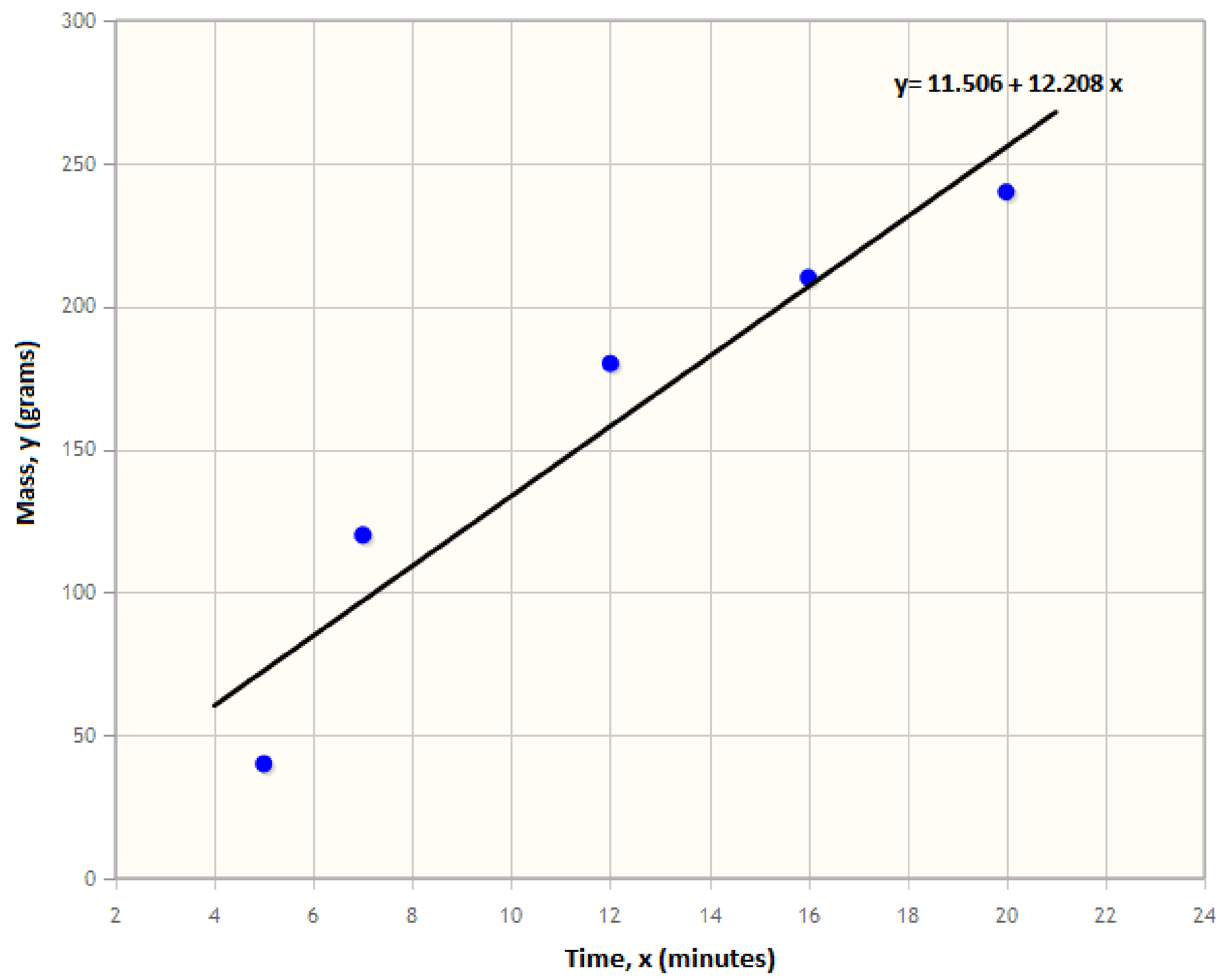
Example 2

- Consider the example below where the mass, y (grams), of a chemical is related to the time, x (seconds), for which the chemical reaction has been taking place according to the table below.

- Find the equation of the regression line.

Time, x (seconds)	55	77	1212	1616	2020
Mass, y (grams)	4040	120120	180180	210210	240

Example (Continued)



Interpreting the Regression Line

- The simple linear regression line, ˆ=$a+bx$, can be interpreted as follows:
  - x is the value of the independent variable
  - ˆ (“y-hat”) is the predicted value of the response variable for a given value of x
  - b is the slope, the amount by which y changes for every one-unit increase in x
  - a is the intercept, the value of y when x = 0

- We can also use the equation of the regression line for finding approximate values for missing data.

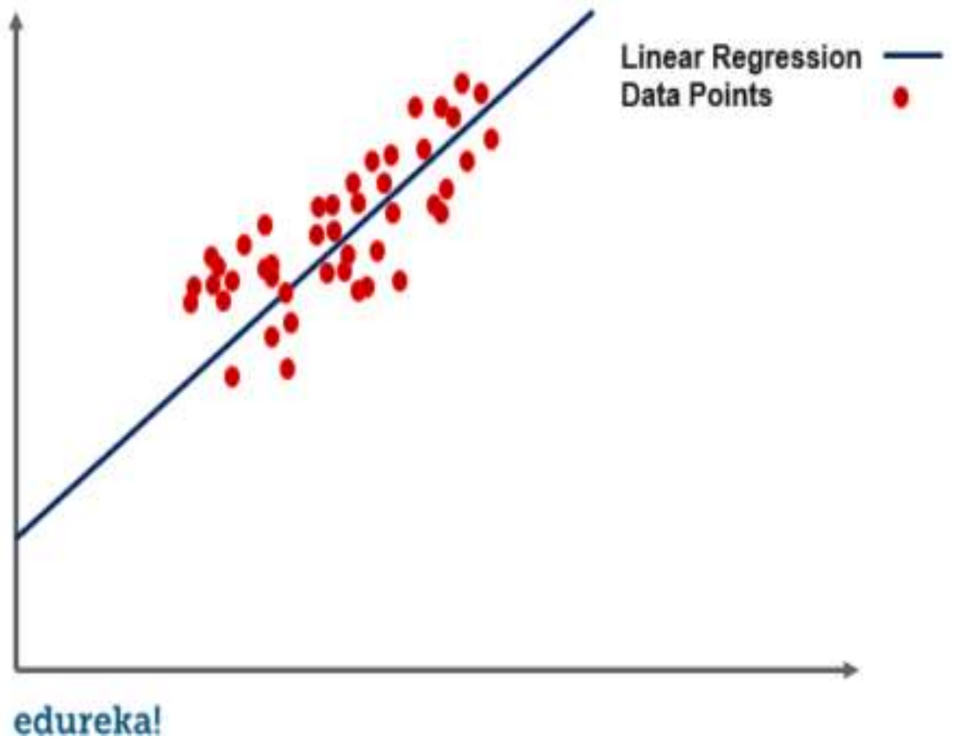
Linear Regression

- Linear regression is a **linear model**, e.g. a model that assumes a linear relationship between the input variables (x) – independent variables and the single output variable (y) – dependent variable
- More specifically, that y can be calculated from a linear combination of the input variables (x).
- When there is a single input variable (x), the method is referred to as **simple linear regression**.
- When there are **multiple input variables**, literature from statistics often refers to the method as **multiple linear regression**

Simple Linear Regression

- One of the most interesting and common regression technique is simple linear regression.
- In this, we predict the outcome of a dependent variable based on the independent variables, the relationship between the variables is linear. Hence, the word linear regression.
- Simple linear regression is a regression technique in which the independent variable has a linear relationship with the dependent variable. The straight line in the diagram is the best fit line.
- The main goal of the simple linear regression is to consider the given data points and plot the best fit line to fit the model in the best way possible.

Simple Linear Regression



Simple Linear Regression

- Simple regression problem (a single x and a single y), the form of the model would be:

$$\underset{\text{Dependent variable (DV)}}{y} = \overset{\text{Constant}}{b0} + \overset{\text{Coefficient}}{b1} * \underset{\text{Independent variable (IV)}}{x1}$$

- **b0** (y-intercept) and **b1** (slope) are the coefficients whose values represent the accuracy of predicted values with the actual values.

Example-2

- Let’s make this concrete with another example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

$$\mathbf{y = B0 + B1 * x1}$$

or

$$\mathbf{weight = B0 + B1 * height}$$

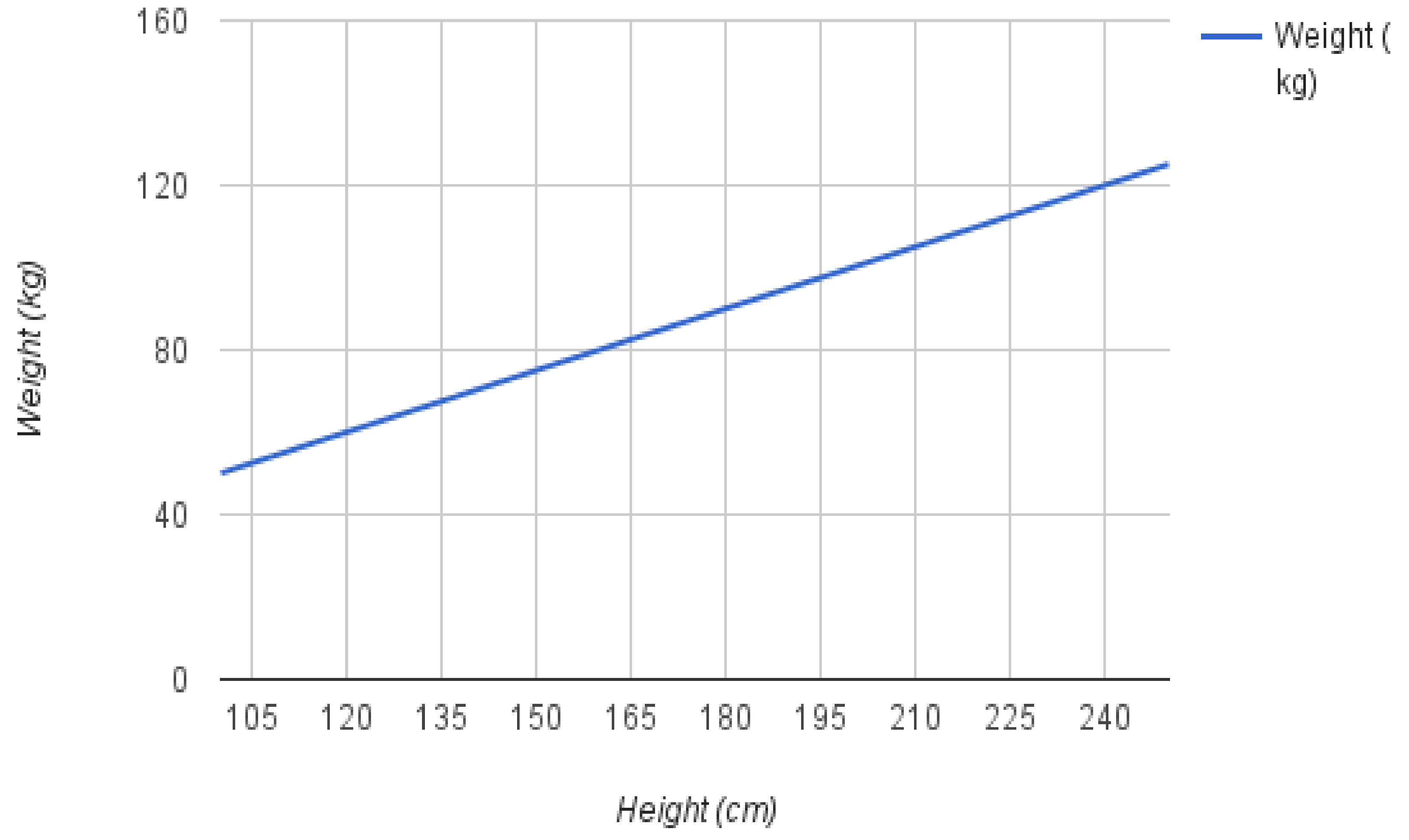
- Where **B0 is the bias coefficient** and **B1 is the coefficient for the height column.** We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.
- For example, lets use **B0 = 0.1 and B1 = 0.5.** Let's plug them in and calculate the weight (in kilograms) for a person with the **height of 182** centimeters.

$$\textbf{weight} = 0.1 + 0.05 * 182$$

$$\textbf{weight} = 91.1$$

- You can see that the above equation could be plotted as a line in two-dimensions. **The B0 is our starting point** regardless of what height we have.
- We can run through a bunch of heights from 100 to 250 centimeters and plug them to the equation and get weight values, creating our line.

Height vs Weight



Simple Linear Regression-Let’s code

- **Step 1: Import the required python packages**
- **Step 2: Load the dataset**
- **Step 3: Data analysis**
- **Step 4: Split data into Train/Test sets**
- **Step 5: Train the regression model**
- **Step 6: Predict the result**

Example

- 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 concerning the experience of employees. In this dataset, we have two columns *YearsExperience* and *Salary*

Step 1: Import the required python packages

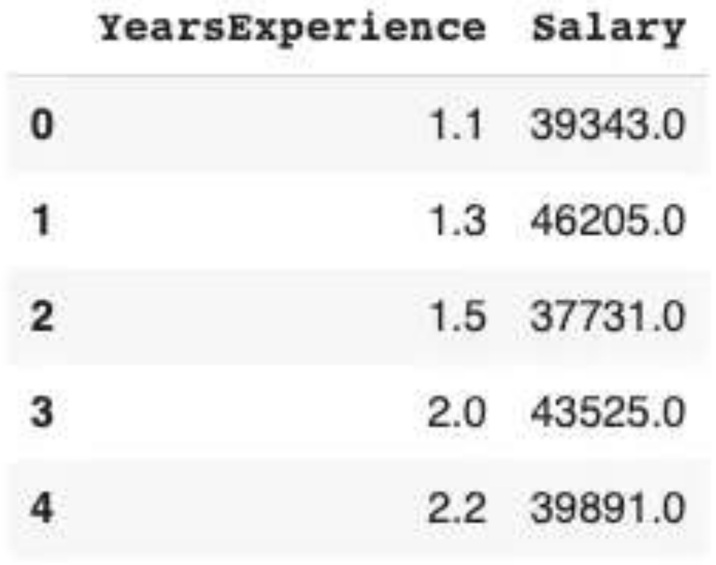
- We need Pandas for data manipulation, NumPy for mathematical calculations, and MatplotLib, and Seaborn for visualizations. Sklearn libraries are used for machine learning operations.

```
# 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 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()
```

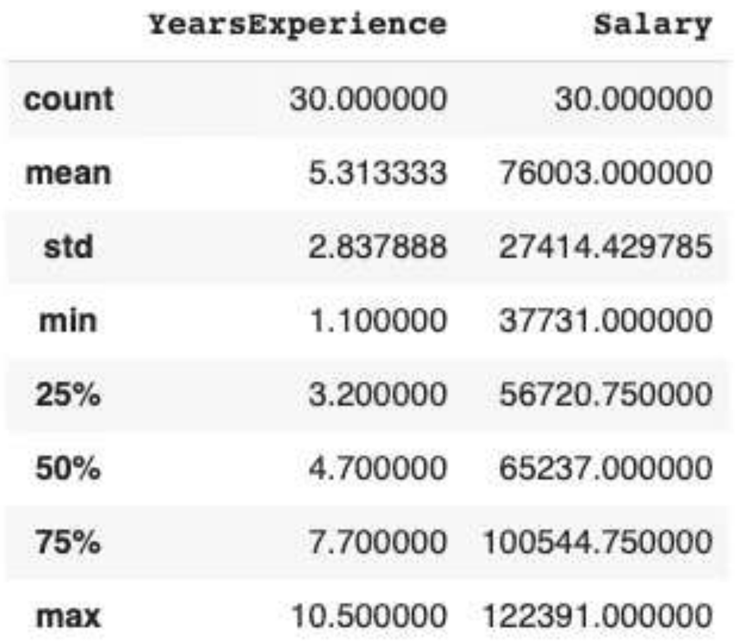


```
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
```

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 –

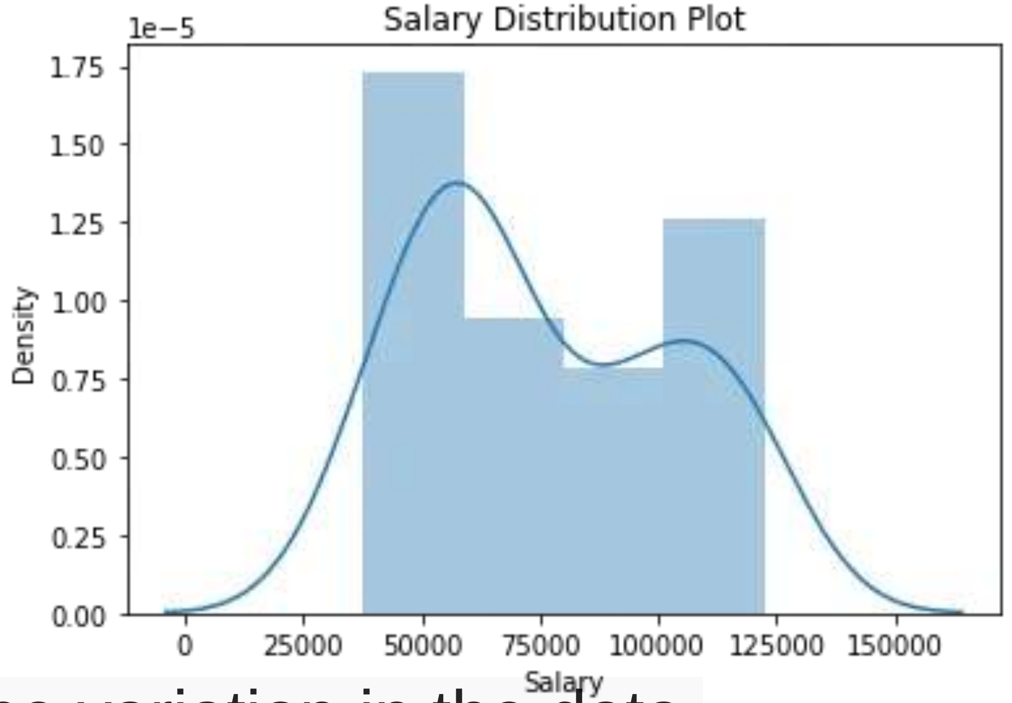
```
# Describe data
df_sal.describe()
```



	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

- 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.

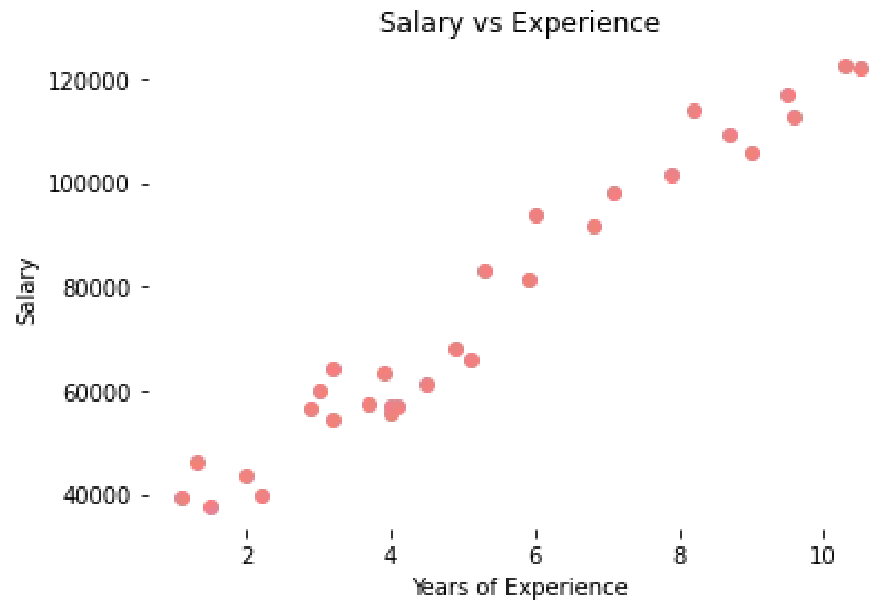


-
  ```
  # Data distribution
  plt.title('Salary Distribution Plot')
  sns.distplot(df_sal['Salary'])
  plt.show()
  ```

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

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

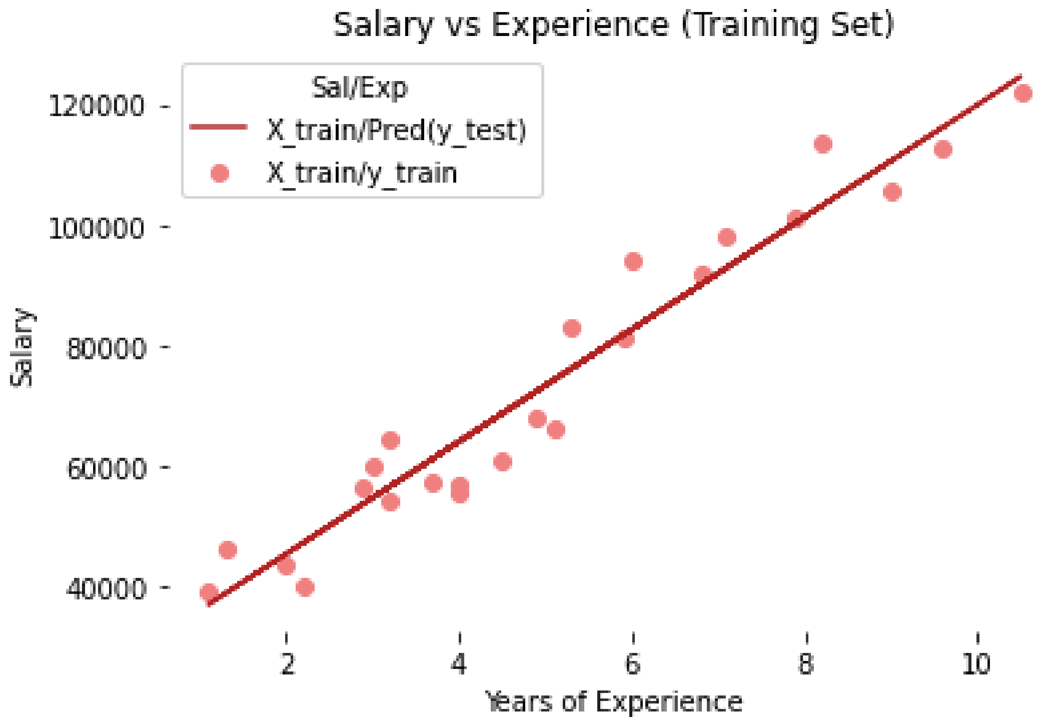
- Here comes the interesting part, when we are all set and ready to predict any value of ***y*** *(Salary)* dependent on ***X*** *(Experience)* with the trained model using ***regressor.predict***
-
```
# Prediction result
y_pred_test = regressor.predict(X_test) # predicted value of y_test
y_pred_train = 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

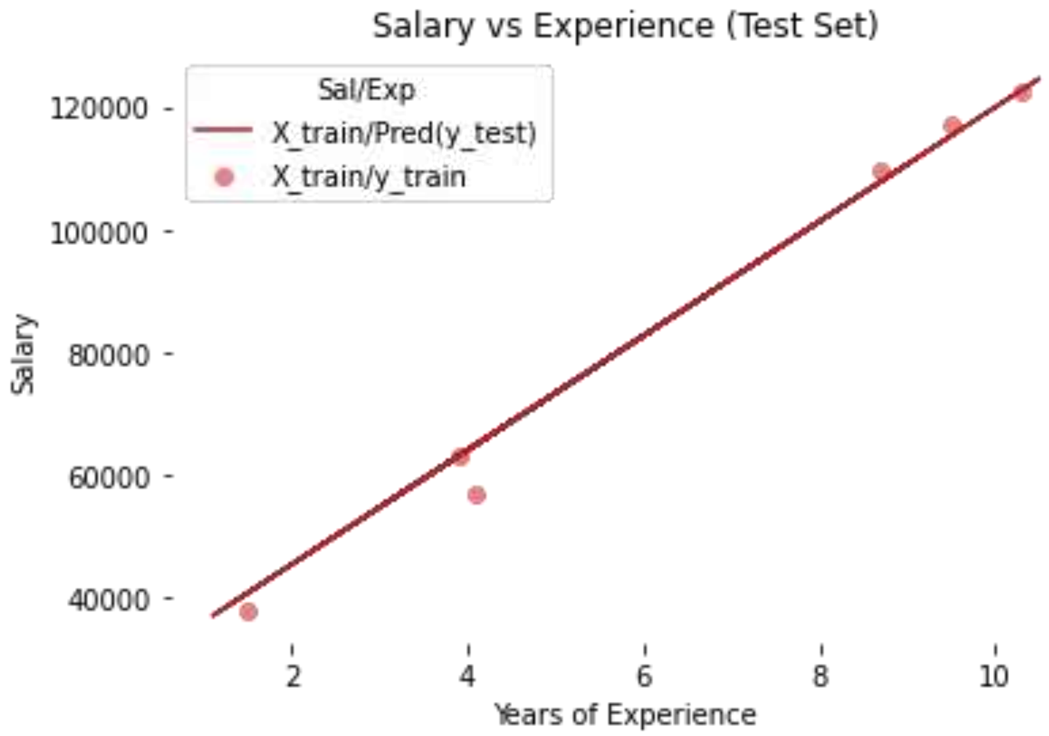
First we plot the result of training sets (X_train, y_train) with X_train and predicted value of y_train (regressor.predict(X_train))

```
# 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()
```


Salary vs Experience (Training Set)
Sal/Exp
X_train/Pred(y_test)
X_train/y_train
120000
100000
80000
60000
40000
Salary
2
4
6
8
10
Years of Experience

- 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))***

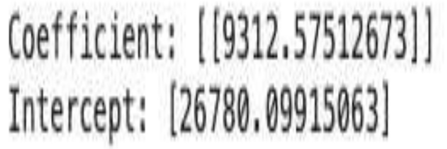
  -
    ```
    # Prediction on test set
    plt.scatter(X_test, y_test, color = 'lightcoral')
    plt.plot(X_train, y_pred_train, color = 'firebrick')
    plt.title('Salary vs Experience (Test Set)')
    plt.xlabel('Years of Experience')
    plt.ylabel('Salary')
    plt.legend(['X_train/Pred(y_test)', '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.

- If you remember from the beginning of this topic, we discussed the linear equation ´$=a+bx$, we can also get the ***a*** *(y-intercept)* and ***a*** *(slope/coefficient)* from the regressor model.

-
  ```
  # Regressor coefficients and intercept
  print(f'Coefficient: {regressor.coef_}')
  print(f'Intercept: {regressor.intercept_}’)
  ```



```
Coefficient: [[9312.57512673]]
Intercept: [26780.09915063]
```