



## ✓ Import python libraries

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.model_selection import train_test_split
6 from sklearn.linear_model import LinearRegression
```

## ✓ Load the dataset

```
1 data = pd.read_csv('/content/Salary_Data.csv')
2 data.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	

Next steps: [View recommended plots](#)

## ✓ Data analysis

```
1 # Describe Data
2 data.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	

```
1 # Data distribution
```

```
2 plt.title('Salary Distribution Plot')
3 sns.distplot(data['Salary'])
4 plt.show()
```

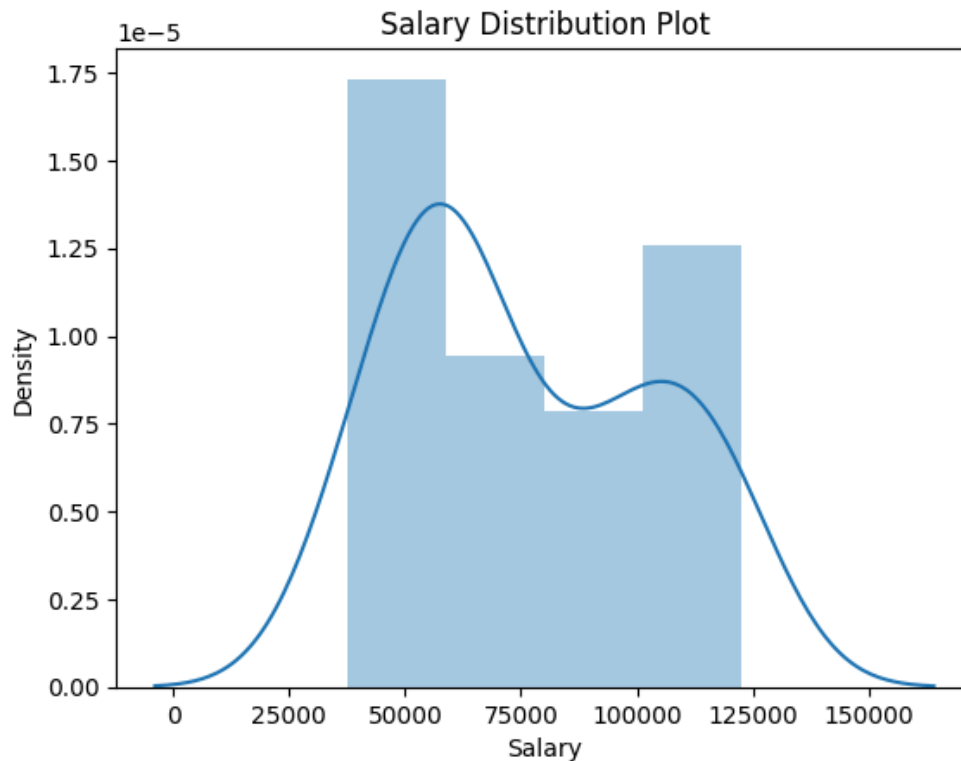
<ipython-input-56-f0d4714b203c>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data['Salary'])
```



```

1 # Relationship between Salary and Experience
2 plt.scatter(data['YearsExperience'], data['Salary'], color = 'orange')
3 plt.title('Salary vs Experience')
4 plt.xlabel('Years of Experience')
5 plt.ylabel('Salary')
6 plt.box(False)
7 plt.show()

```



## ✓ Split data Dependent and Independent variable

```

1 # Splitting variables
2 X = data.iloc[:, :1] # independent
3 y = data.iloc[:, 1:] # dependent

```

## ✓ split data into train/test sets

```

1 # Splitting dataset into test/train
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

```

## ✓ Train the Regression model

```

1 regressor = LinearRegression()
2 regressor.fit(X_train, y_train)

```

```
LinearRegression
LinearRegression()
```

## ✓ predict the result

```
1 # Prediction result
2 y_pred_test = regressor.predict(X_test)    # predicted value of y_test
3 y_pred_train = regressor.predict(X_train)
```

## ✓ plot training and test results

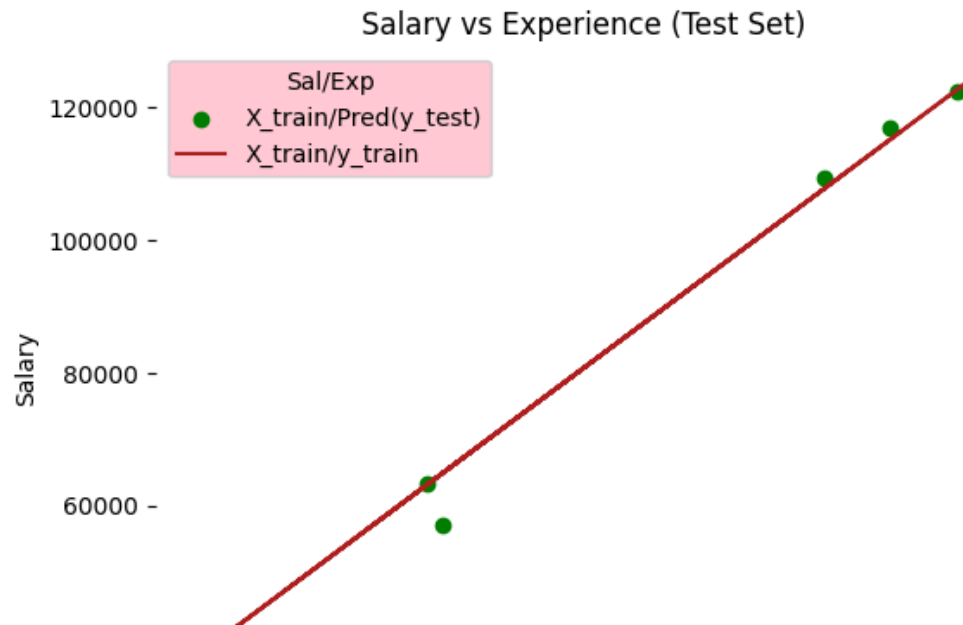
```
1 # Prediction on training set
2 plt.scatter(X_train, y_train, color = 'green')
3 plt.plot(X_train, y_pred_train, color = 'orange')
4 plt.title('Salary vs Experience (Training Set)')
5 plt.xlabel('Years of Experience')
6 plt.ylabel('Salary')
7 plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title = 'Sal/Exp', loc='best', facecolor
8 plt.box(False)
9 plt.show()
```



```

1 # Prediction on test set
2 plt.scatter(X_test, y_test, color = 'green')
3 plt.plot(X_train, y_pred_train, color = 'firebrick')
4 plt.title('Salary vs Experience (Test Set)')
5 plt.xlabel('Years of Experience')
6 plt.ylabel('Salary')
7 plt.legend(['X_train/Pred(y_test)', 'X_train/y_train'], title = 'Sal/Exp', loc='best', facecolor
8 plt.box(False)
9 plt.show()

```



```

1 # Regressor coefficients and intercept
2 print(f'Coefficient: {regressor.coef_}')
3 print(f'Intercept: {regressor.intercept_}')

```

```

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

```