Salary Dataset

Import the required python packages

```
# 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
```

Load the dataset(Salary Dataset)

```
# 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

Next steps: Generate code with df_sal View recommended plots

Data analysis

```
# 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

Data Distribution Plot

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

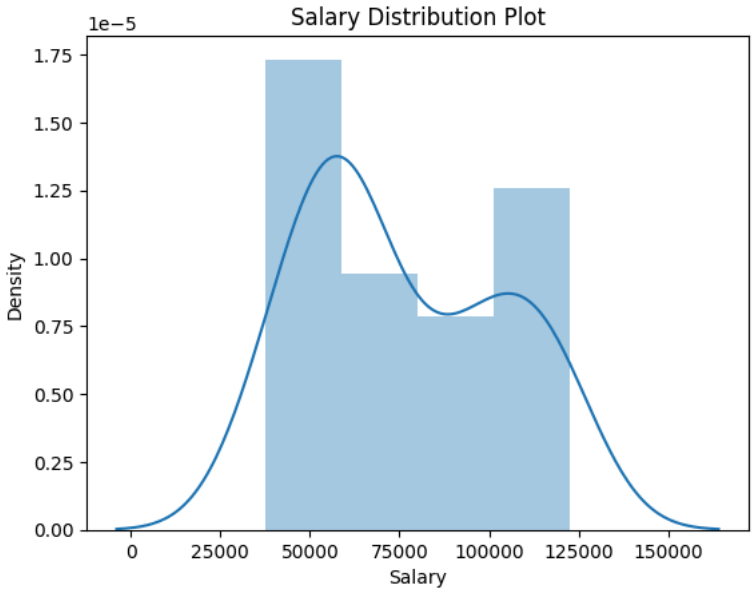
```
<ipython-input-17-d6ace42bc912>:3: UserWarning:

`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(df_sal['Salary'])
```



To check the realtionship between experience and salary

```
# 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 that data varies linearly. That means, that an individual receives more Salary as they gain Experience

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
```

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)
```

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)
```

```
▾ LinearRegression
LinearRegression()
```

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

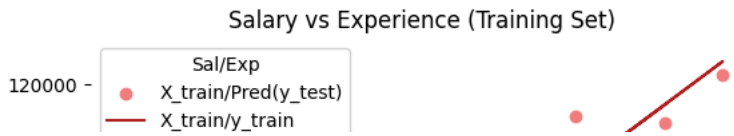
```
# Prediction result
y_pred_test = regressor.predict(X_test)     # predicted value of y_test
y_pred_train = regressor.predict(X_train)   # predicted value of y_train
```

Plot the training and test results

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

```
# 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()
```


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

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

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.