```
#Step 1: Import the required python packages
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
df = pd.read_csv(r"C:\Users\abhis\OneDrive\Desktop\ML Lab\Machine-
Learning\Salary_Data.csv")

df.head()
```

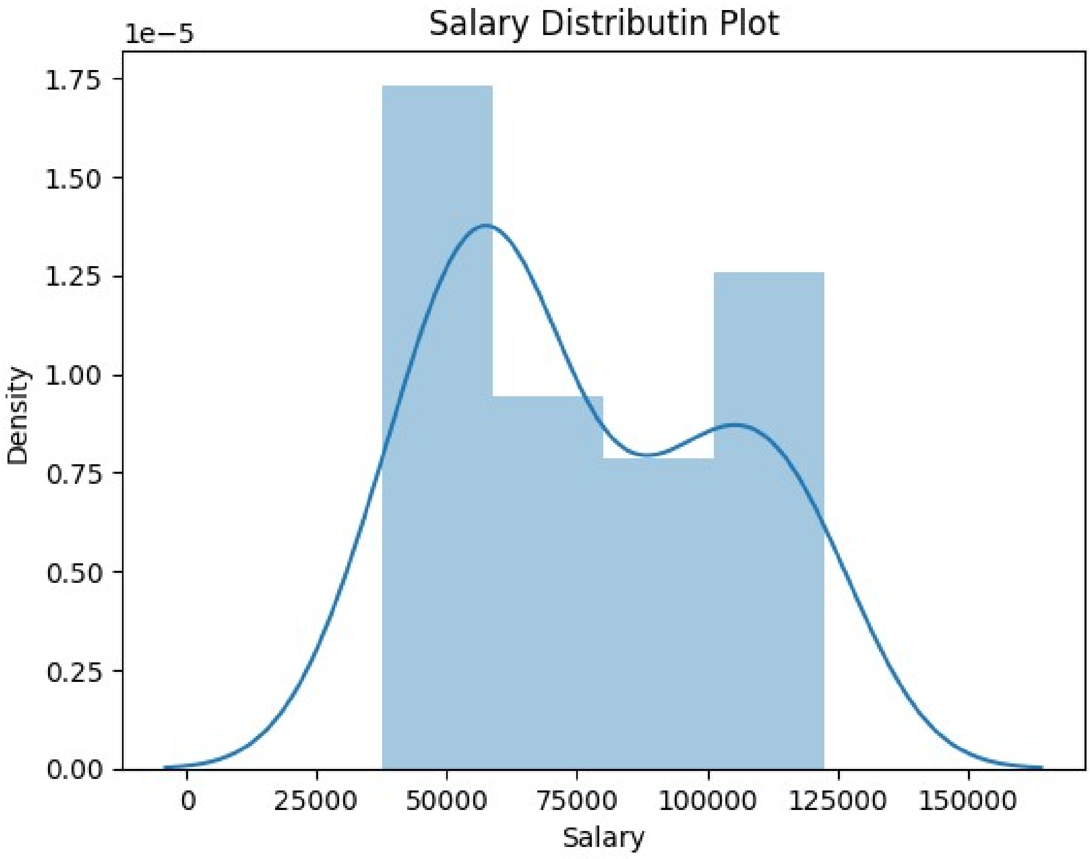
```
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
```

```
#Step 3: Data analysis
#Describe Data
df.describe()
```

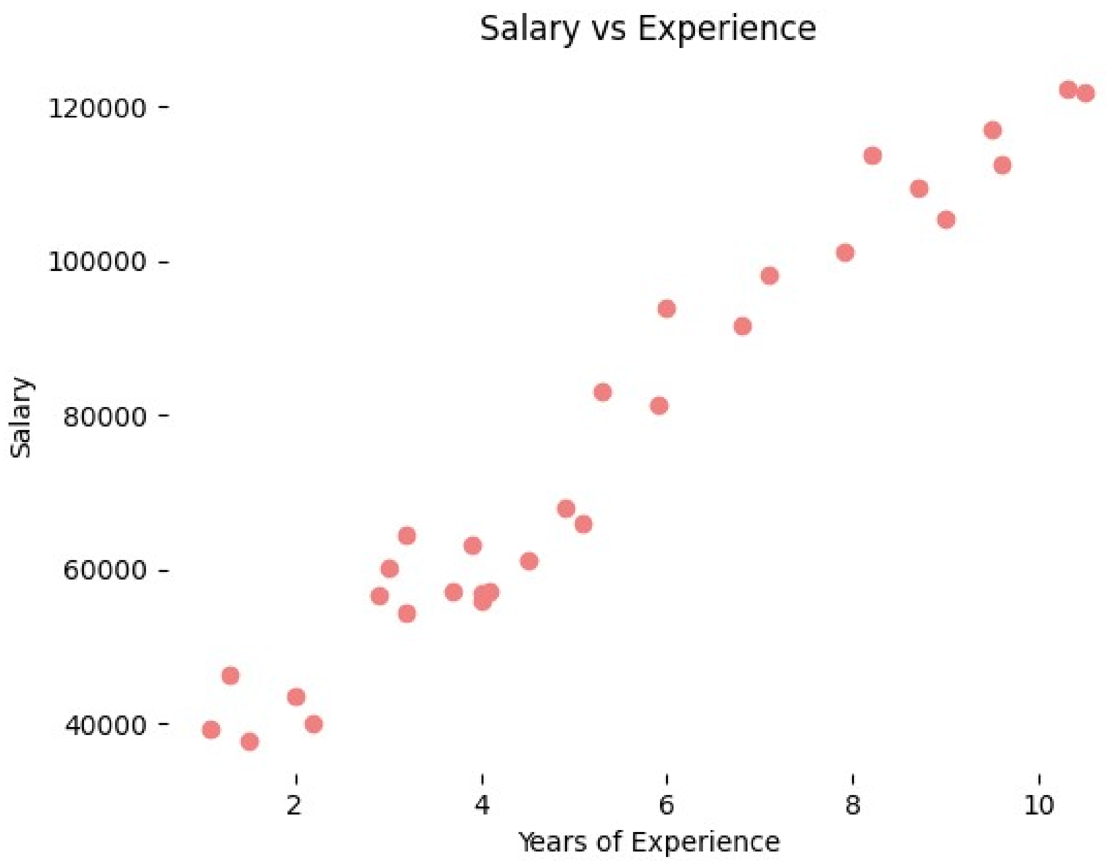
```
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000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

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

```
C:\Users\abhis\AppData\Local\Temp\ipykernel_37644\1268520552.py: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['Salary'])
```



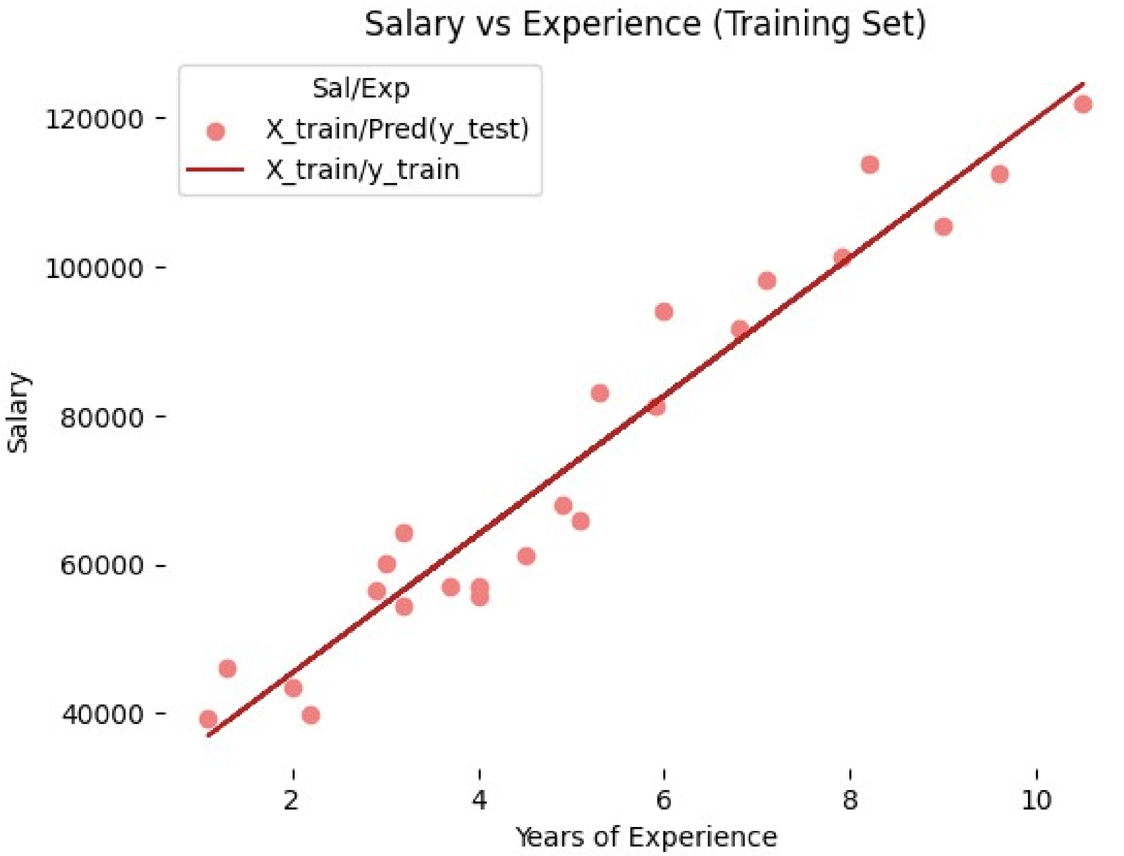
```
# Relationship between Salary And Experience
plt.scatter(df['YearsExperience'], df['Salary'], color = 'lightcoral')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.box(False)
plt.show()
```



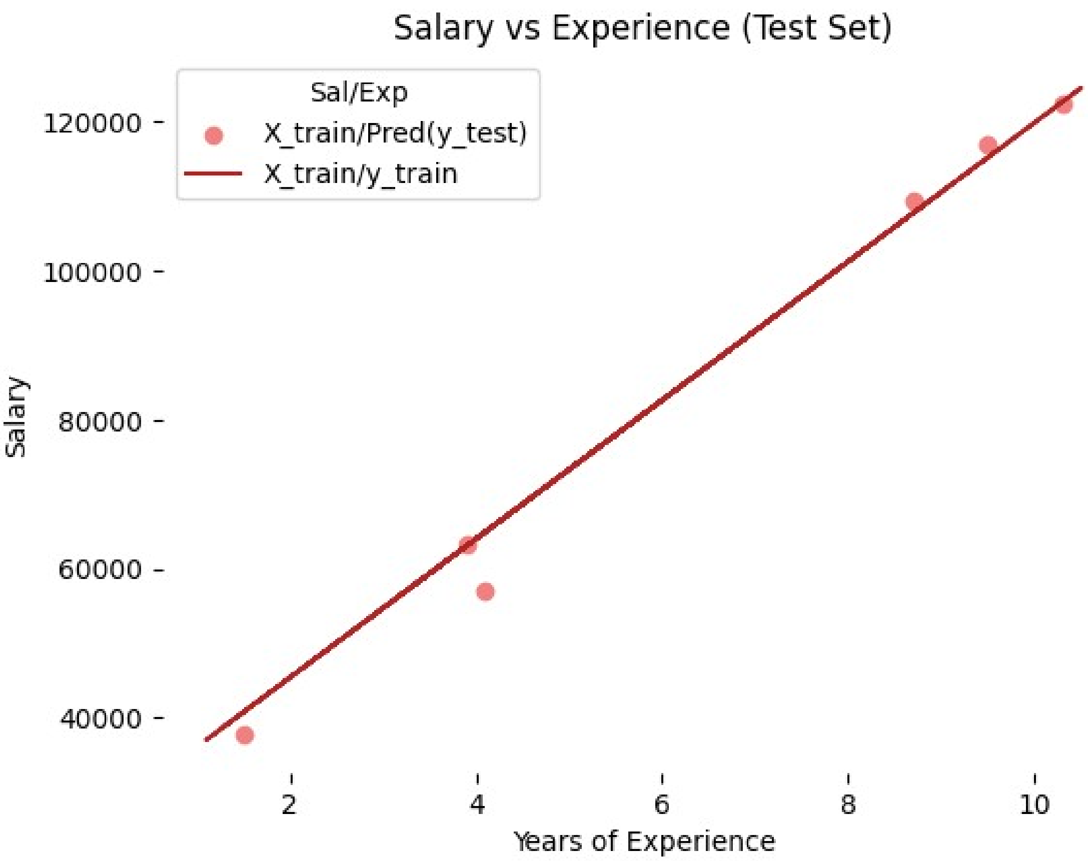
```
#Step 4: Split the dataset into dependent/independent variables

# Splitting Variables
X = df.iloc[:, :1] # independent
y = df.iloc [:, 1:] # dependent

# Step 4: Split data into Train/Test sets

#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

# Regressor Model

regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression()
```

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

#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]
```

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import (mean_absolute_error, mean_squared_error,
median_absolute_error, mean_absolute_percentage_error, r2_score,
explained_variance_score, max_error)
import numpy as np

df = pd.read_csv("Salary_Data.csv")

# Split into features and target
X = df[['YearsExperience']]
y = df['Salary']

# Train the Linear Regression Model
model = LinearRegression()
model.fit(X, y)
```

```
LinearRegression()
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, max_error, r2_score, explained_variance_score

y_pred = model.predict(X)

# Calculate error metrics
mae = mean_absolute_error(y, y_pred)
mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y, y_pred)
medae = median_absolute_error(y, y_pred)
mape = mean_absolute_percentage_error(y, y_pred)
max_err = max_error(y, y_pred)
evs = explained_variance_score(y, y_pred)

# Display the results
print(f"□ Linear Regression Equation:")
print(f"Salary = {model.coef_[0]:.2f} * YearsExperience +
{model.intercept_:.2f}\n")

print("□ Error Metrics:")
print(f"1. Mean Absolute Error (MAE): {mae:.2f}")
print(f"2. Mean Squared Error (MSE): {mse:.2f}")
print(f"3. Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"4. Median Absolute Error: {medae:.2f}")
print(f"5. Mean Absolute Percentage Error (MAPE): {mape*100:.2f}%")
print(f"6. Max Error: {max_err:.2f}")
print(f"7. R² Score: {r2:.4f}")
print(f"8. Explained Variance Score: {evs:.4f}")
```

```
□ Linear Regression Equation:
Salary = 9449.96 * YearsExperience + 25792.20

□ Error Metrics:
1. Mean Absolute Error (MAE): 4644.20
2. Mean Squared Error (MSE): 31270951.72
```

```
3. Root Mean Squared Error (RMSE): 5592.04
4. Median Absolute Error: 4017.93
5. Mean Absolute Percentage Error (MAPE): 7.05%
6. Max Error: 11448.03
7. R² Score: 0.9570
8. Explained Variance Score: 0.9570
```