```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the CSV file into a Pandas DataFrame
df = pd.read_csv('Salary_dataset.csv')

# Display the first 5 rows of the DataFrame
print("First 5 rows of the dataset:")
print(df.head())

# Display basic dataset information
print("\nBasic information about the dataset:")
df.info()
```

```
First 5 rows of the dataset:
   Unnamed: 0  YearsExperience   Salary
0           0              1.2  39344.0
1           1              1.4  46206.0
2           2              1.6  37732.0
3           3              2.1  43526.0
4           4              2.3  39892.0

Basic information about the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Unnamed: 0       30 non-null     int64
 1   YearsExperience  30 non-null     float64
 2   Salary           30 non-null     float64
dtypes: float64(2), int64(1)
memory usage: 852.0 bytes
```

```python
# Display the shape of the dataset
print("\nDataset Shape:")
print(df.shape)

# Display column names
print("\nColumn Names:")
print(df.columns.tolist())

# Display data types
print("\nData Types:")
print(df.dtypes)

# Check for missing values
```

```
print("\nMissing Values:")
print(df.isnull().sum())
```

```
Dataset Shape:
(30, 3)

Column Names:
['Unnamed: 0', 'YearsExperience', 'Salary']

Data Types:
Unnamed: 0           int64
YearsExperience    float64
Salary             float64
dtype: object

Missing Values:
Unnamed: 0         0
YearsExperience    0
Salary             0
dtype: int64
```

```
# Separate independent variable (X) and dependent variable (y)
X = df['YearsExperience'].values.reshape(-1, 1)
y = df['Salary'].values

print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
print("\nFirst 5 rows of X:\n", X[:5])
print("\nFirst 5 rows of y:\n", y[:5])
```

```
Shape of X: (30, 1)
Shape of y: (30,)

First 5 rows of X:
 [[1.2]
 [1.4]
 [1.6]
 [2.1]
 [2.3]]

First 5 rows of y:
 [39344. 46206. 37732. 43526. 39892.]
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print(f"Shape of X_train: {X_train.shape}")
print(f"Shape of X_test: {X_test.shape}")
```

```python
print(f"Shape of y_train: {y_train.shape}")
print(f"Shape of y_test: {y_test.shape}")
```

```
Shape of X_train: (24, 1)
Shape of X_test: (6, 1)
Shape of y_train: (24,)
Shape of y_test: (6,)
```

```python
from sklearn.linear_model import LinearRegression

# Create a Linear Regression model object
linear_regressor = LinearRegression()

print("Linear Regression model object created.")
```

```
Linear Regression model object created.
```

```python
# Train the Linear Regression model
linear_regressor.fit(X_train, y_train)

# Display the learned coefficient (slope) and intercept
print(f"Coefficient (Slope): {linear_regressor.coef_[0]:.2f}")
print(f"Intercept: {linear_regressor.intercept_:.2f}")
```

```
Coefficient (Slope): 9423.82
Intercept: 24380.20
```

```python
# Predict salary values for the test dataset
y_pred = linear_regressor.predict(X_test)

# Create a DataFrame to display actual vs. predicted values in tabular
format
results_df = pd.DataFrame({'YearsExperience': X_test.flatten(),
'Actual Salary': y_test, 'Predicted Salary': y_pred})

print("Actual vs. Predicted Salary Values:")
print(results_df.round(2))
```

```
Actual vs. Predicted Salary Values:
   YearsExperience  Actual Salary  Predicted Salary
0              9.7       112636.0         115791.21
1              5.0        67939.0          71499.28
2              8.3       113813.0         102597.87
3              5.4        83089.0          75268.80
4              3.3        64446.0          55478.79
5              3.8        57190.0          60190.70
```

```python
from sklearn.metrics import mean_squared_error

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
```

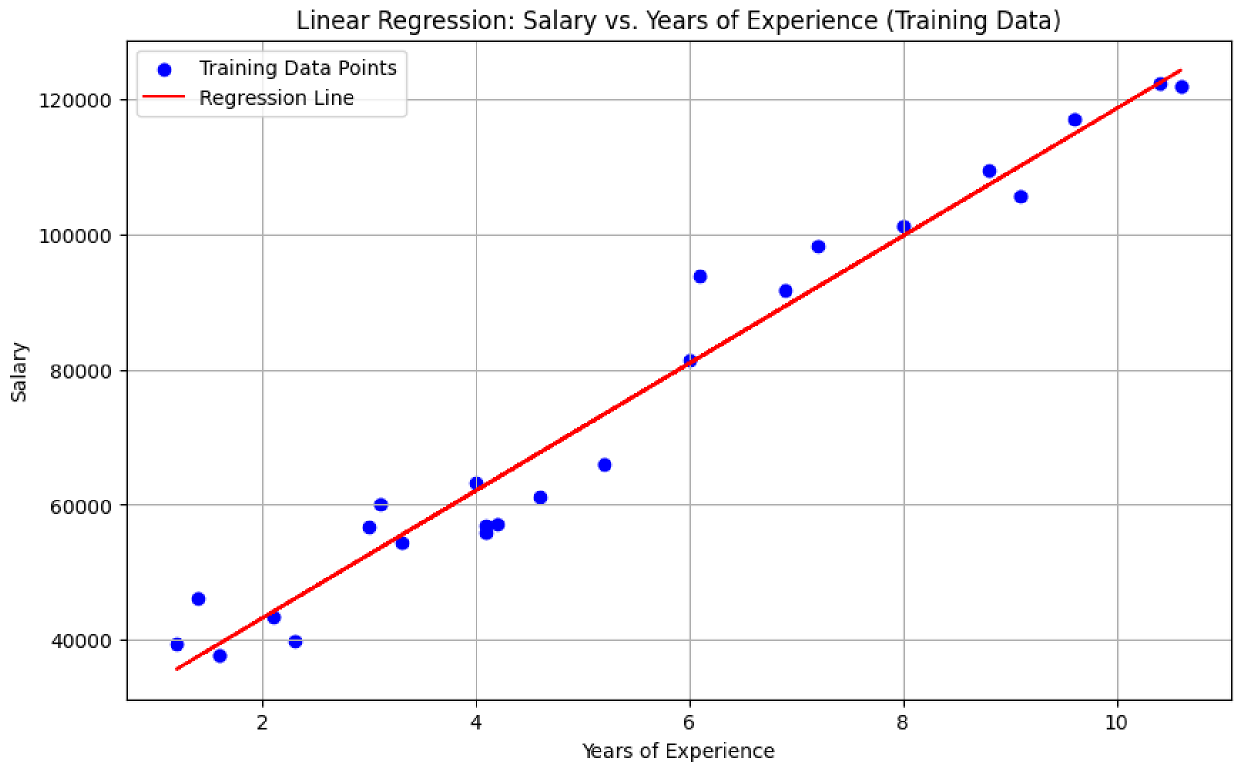
```
print(f"Mean Squared Error (MSE): {mse:.2f}")
```

```
Mean Squared Error (MSE): 49830096.86
```

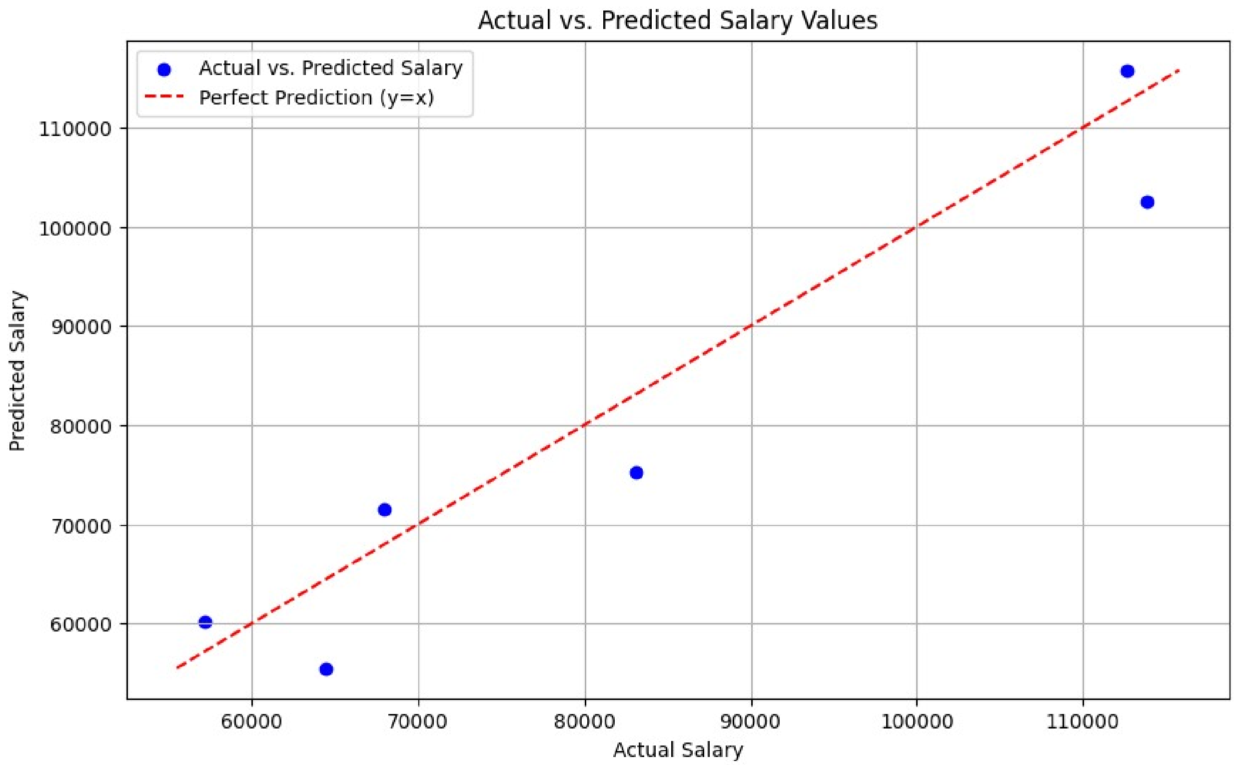
```
from sklearn.metrics import r2_score

# Calculate R-squared score
r2 = r2_score(y_test, y_pred)

print(f"R-squared (R²): {r2:.2f}")
```

```
R-squared (R²): 0.90
```

```
plt.figure(figsize=(10, 6))

# Plotting the training data points
plt.scatter(X_train, y_train, color='blue', label='Training Data
Points')

# Plotting the regression line
# Use the trained model to predict y values for the training X values
plt.plot(X_train, linear_regressor.predict(X_train), color='red',
label='Regression Line')

# Labeling axes and adding title
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Linear Regression: Salary vs. Years of Experience (Training
Data)')

# Adding a legend
plt.legend()

# Display the plot
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(10, 6))

# Create a scatter plot of actual vs. predicted salary values
plt.scatter(y_test, y_pred, color='blue', label='Actual vs. Predicted
Salary')

# Add a reference line (y = x) for perfect predictions
# Use min and max of both actual and predicted values to set the line
range
min_val = min(min(y_test), min(y_pred))
max_val = max(max(y_test), max(y_pred))
plt.plot([min_val, max_val], [min_val, max_val], color='red',
linestyle='--', label='Perfect Prediction (y=x)')

# Labeling axes and adding title
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('Actual vs. Predicted Salary Values')

# Adding a legend
plt.legend()

# Display the plot
plt.grid(True)
plt.show()
```



Actual vs. Predicted Salary Values
Actual vs. Predicted Salary
Perfect Prediction (y=x)
Predicted Salary
110000
100000
90000
80000
70000
60000
60000
70000
80000
90000
100000
110000
Actual Salary