Praticle 2 : Implementating Simple Linear Regression

In [11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
```

In [12]:
```
# Load the dataset
data = pd.read_csv('salary_data.csv')
```

In [13]:
```
# Handling missing values in the feature variable
imputer = SimpleImputer(strategy='mean')
data['Years of Experience'] = imputer.fit_transform(data[['Years of Exper
```

In [14]:
```
# Handling missing values in the target variable
data['Salary'] = imputer.fit_transform(data[['Salary']])
```

In [15]:
```
# Splitting the data into features (X) and the target variable (y)
X = data[['Years of Experience']]
y = data['Salary']
```

In [16]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

In [17]:
```
# Train the Simple Linear Regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[17]:
```
▾ LinearRegression

LinearRegression()
```

In [18]:
```
# Predict on the test set
y_pred = regressor.predict(X_test)
```

In [22]:
```
# Print the test and training data along with predicted data
print("Test Data:")
print(X_test)
print("Actual Salary:")
print(y_test)
```

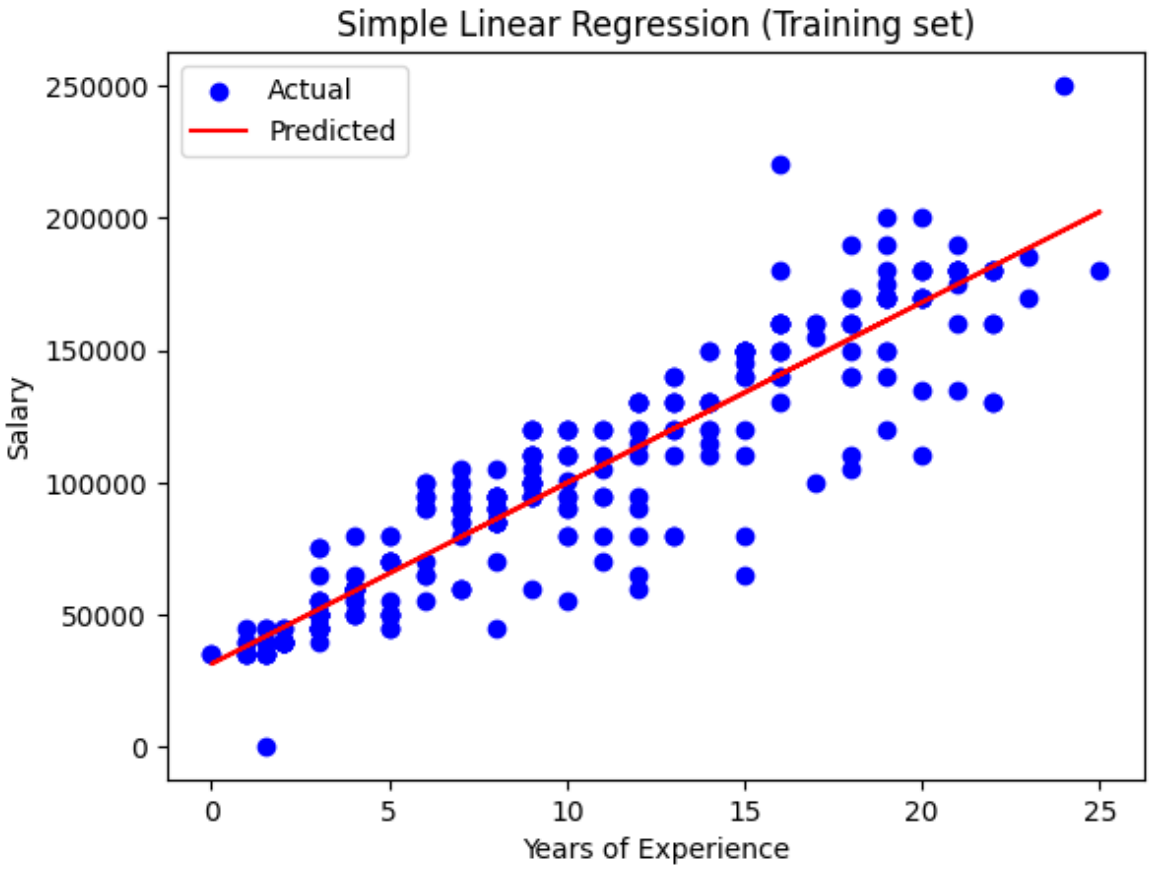
```
Test Data:
     Years of Experience
167                 18.0
33                  10.0
15                  16.0
316                  6.0
57                  17.0
..                   ...
94                   7.0
196                 11.0
350                 16.0
312                 15.0
349                  8.0

[75 rows x 1 columns]
Actual Salary:
167    150000.0
33      65000.0
15     125000.0
316     80000.0
57     140000.0
         ...
94      75000.0
196     90000.0
350    160000.0
312    150000.0
349    110000.0
Name: Salary, Length: 75, dtype: float64
```

In [23]:
```
print("Predicted Salary:")
print(y_pred)
```

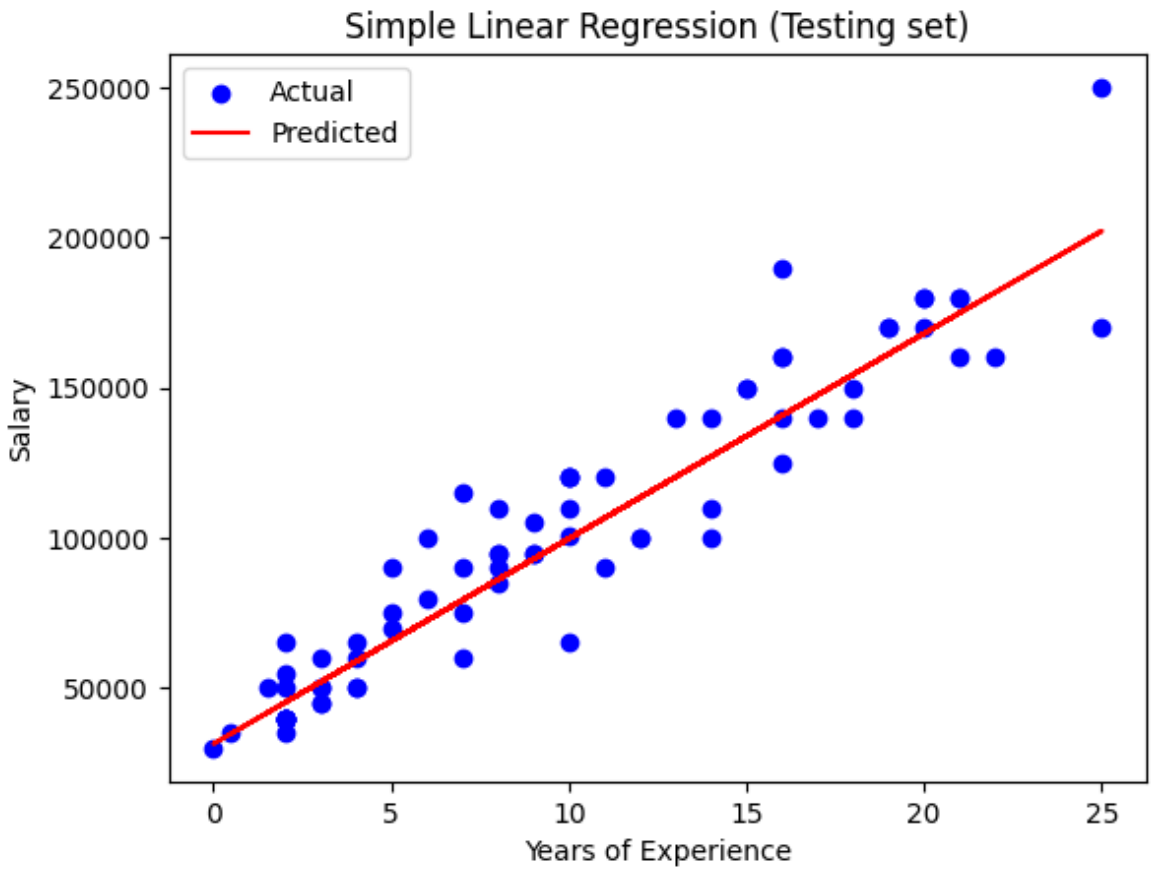
```
Predicted Salary:
[154355.22616873  99726.43447057 140698.02824419  72412.03862149
 147526.62720646 174841.02305554 181669.62201781  99726.43447057
  51926.24173467  92897.8355083  120212.23135738 174841.02305554
  45097.6427724   92897.8355083   58754.84069694 140698.02824419
  65583.43965921  41683.34329127  79240.63758376  86069.23654603
  99726.43447057  92897.8355083  140698.02824419  65583.43965921
  99726.43447057  58754.84069694 168012.42409327  99936.96768254
 154355.22616873 161183.825131    51926.24173467 127040.83031965
  51926.24173467 133869.42928192  79240.63758376  34854.744329
 174841.02305554  45097.6427724   45097.6427724   58754.84069694
  45097.6427724   86069.23654603 202155.41890462 161183.825131
  45097.6427724  140698.02824419 106555.03343284  45097.6427724
  45097.6427724   45097.6427724   99726.43447057  86069.23654603
  86069.23654603 113383.63239511 168012.42409327  58754.84069694
 202155.41890462  79240.63758376  51926.24173467  72412.03862149
 113383.63239511 127040.83031965 168012.42409327  65583.43965921
  45097.6427724   45097.6427724   51926.24173467 127040.83031965
  45097.6427724   31440.44484786  79240.63758376 106555.03343284
 140698.02824419 133869.42928192  86069.23654603]
```

In [20]:
```
# Visualization of training data
plt.scatter(X_train, y_train, color='blue', label='Actual')
plt.plot(X_train, regressor.predict(X_train), color='red', label='Predict
plt.title('Simple Linear Regression (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
```

```
plt.legend()
plt.show()
```



In [21]:
```
# Visualization of testing data
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Predicted')
plt.title('Simple Linear Regression (Testing set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.show()
```



Steps Followed:

1. Data Loading and Preprocessing:
    - Load the dataset.
    - Handle missing values using an imputer transformer, such as SimpleImputer.
2. Splitting the Data:
    - Separate the dataset into the feature variable ('Years of Experience') and the target variable ('Salary').
    - Split the data into training and testing sets using train_test_split from sklearn.model_selection.
3. Model Training
4. Predictions and Evaluation
5. Visualization using matplotlib.

In [ ]: