```python
#import python 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 sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

#import inline plotting
%matplotlib inline

#load the dataset
df = pd.read_csv('salary_prediction_dataset.csv')

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"YearsExperience\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
5.5506098770022305,\n        \"min\": 1.0880084370874543,\n
\"max\": 19.99463579243648,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          8.236926275989408,\n
16.515389349959964,\n          9.790915927722525\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
27966.62194385531,\n        \"min\": -4035.8771311213504,\n
\"max\": 107828.78711278818,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          46144.84312762965,\n
85731.00517257371,\n          55059.74811623938\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```python
#checking for missing values
print(df.isnull().sum())
```

```
YearsExperience    0
Salary             0
dtype: int64
```

```python
#plotting data distribution
plt.figure(figsize=(8,5))
sns.scatterplot(x = df['YearsExperience'], y = df['Salary'])
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Years of Experience vs Salary')
plt.show()
```

The scatter plot shows the variation of salary for different years of work experience. More is the experience, more will be the salary

```
#splitting the dataset into features (x) and target variable (y)
x = df[['YearsExperience']] #independent variable
y = df['Salary'] #dependent variable

#splitting into training (80%) and testing(20%)
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)

#checking dataset split sizes
x_train.shape, x_test.shape
```

```
((800, 1), (200, 1))
```

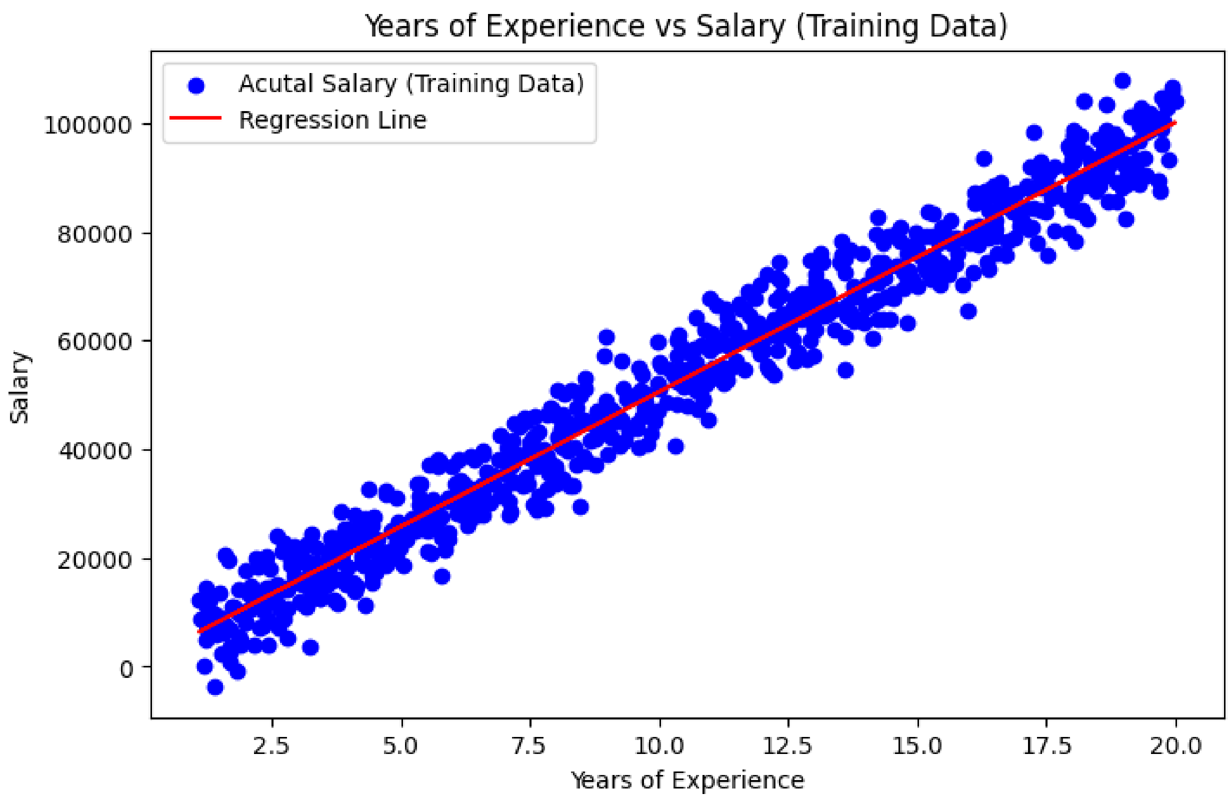
```
#creating and training a simple linear regression model
model = LinearRegression()
model.fit(x_train, y_train)

#display model parameters
print(f'Model intercept: {model.intercept_}')
print(f'Model coefficient: {model.coef_[0]}')
```

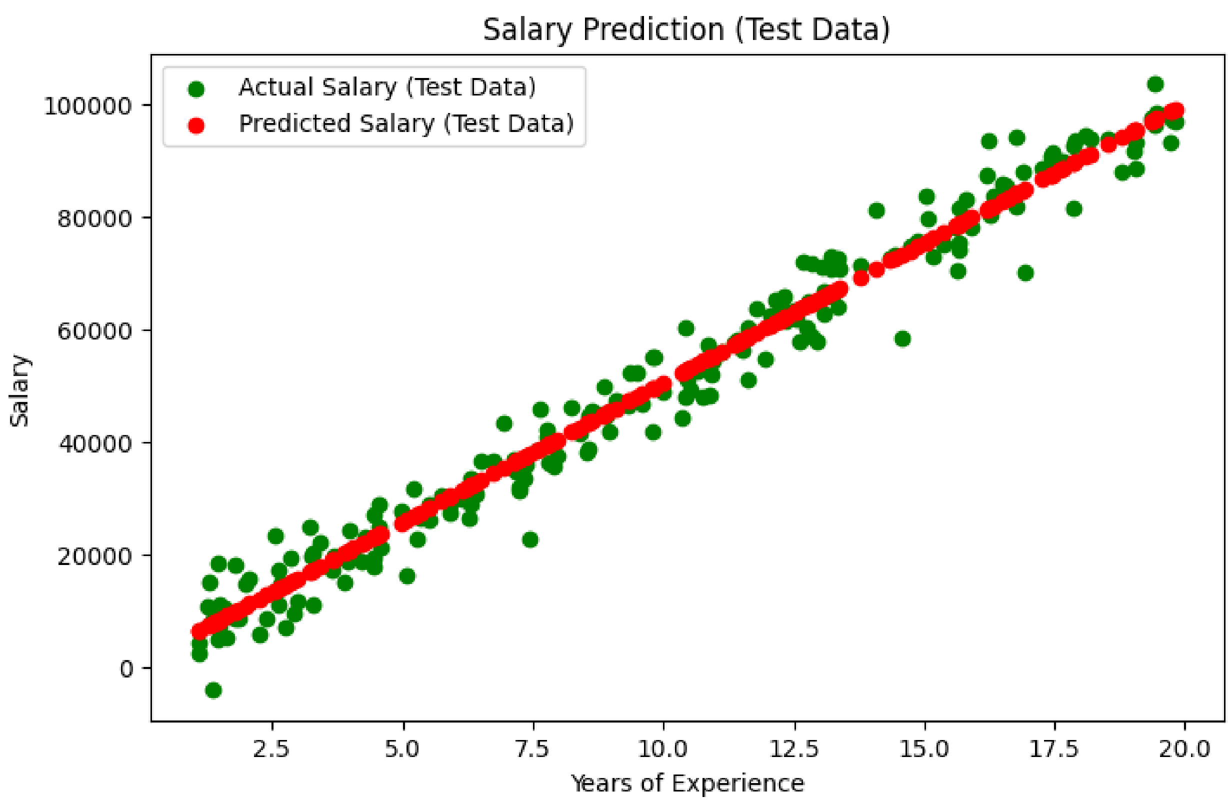
```
Model intercept: 995.5770464802481
Model coefficient: 4953.709417260945
```

```
#predicting salary for test data
y_pred = model.predict(x_test)

#creating a Dataframe to compare actual vs predicted values
results = pd.DataFrame({'Actual Salary': y_test, 'Predicted Salary':
y_pred})
results.head()
```


```
{"summary":"{\n  \"name\": \"results\",\n  \"rows\": 200,\n
\"fields\": [\n    {\n      \"column\": \"Actual Salary\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
27774.710877905603,\n        \"min\": -4035.8771311213504,\n
\"max\": 103548.45493157378,\n        \"num_unique_values\": 200,\n
\"samples\": [\n          8545.03248337904,\n
10965.208867584226,\n          66847.07415862133\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Predicted Salary\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
27236.785721073156,\n        \"min\": 6425.6851594813115,\n
\"max\": 99133.62900002314,\n        \"num_unique_values\": 200,\n
\"samples\": [\n          12797.77547097714,\n
13997.390340189799,\n          65724.60504750915\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"results"}
```


```
#plot training data with regression line
plt.figure(figsize=(8,5))
plt.scatter(x_train, y_train, color = 'blue', label = 'Acutal Salary
(Training Data)')
plt.plot(x_train, model.predict(x_train), color = 'red', label =
'Regression Line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Years of Experience vs Salary (Training Data)')
plt.legend()
plt.show()

#plot test data predictions
plt.figure(figsize=(8,5))
plt.scatter(x_test, y_test, color = 'green', label = 'Actual Salary
(Test Data)')
plt.scatter(x_test, y_pred, color = 'red', label = 'Predicted Salary
(Test Data)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary Prediction (Test Data)')
plt.legend()
plt.show()
```


Years of Experience vs Salary (Training Data)
Acutal Salary (Training Data)
Regression Line
Salary
Years of Experience
0
20000
40000
60000
80000
100000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Salary Prediction (Test Data)
Actual Salary (Test Data)
Predicted Salary (Test Data)
Salary
Years of Experience
0
20000
40000
60000
80000
100000
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

```
#model evaluation
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'R² Score: {r2}')
```

```
Mean Absolute Error (MAE): 3602.3955438952066
Mean Squared Error (MSE): 21434311.66454385
R² Score: 0.9720753738957405
```

```
#predict salary based on input
experience = float(input('Enter years of experience : '))
predicted_salary = model.predict([[experience]])
print(f'Predicted Salary for {experience} years of experience:
{predicted_salary[0]:.2f}')
```

```
Enter years of experience : 11
Predicted Salary for 11.0 years of experience: 55486.38

/usr/local/lib/python3.11/dist-packages/sklearn/utils/
validation.py:2739: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

```
# prompt: make a conclusion on this model. how this model can help HR
professionals

This linear regression model predicts salary based on years of
experience.  For HR professionals, this model offers several potential
applications:

* **Salary Benchmarking:**  HR can use the model to benchmark salaries
against market rates for specific years of experience, ensuring
competitive compensation packages.  This helps attract and retain
talent.

* **Salary Planning and Budgeting:**  The model aids in forecasting
salary expenses for future hires or promotions, improving budget
accuracy and resource allocation.

* **Identifying Salary Discrepancies:**  By comparing actual salaries
against predicted values, HR can pinpoint potential salary inequities
or anomalies within the organization, helping to ensure fair
compensation practices.

* **Performance-Based Salary Adjustments:** The model can help in
determining salary adjustments based on experience growth and
performance reviews.
```

```
* **Negotiation Support:** Provides data-driven insights during salary
negotiations with potential candidates or current employees, leading
to more objective and fair outcomes.


**Limitations:**  The model's accuracy depends on the quality and
representativeness of the input data.  It's crucial to remember that
salary is influenced by factors beyond just experience (e.g.,
location, education, skills, company size and performance, industry,
job title).  The model should be used as a tool to inform decisions,
not as the sole determinant of salary.  Regularly retraining the model
with updated data is necessary to maintain accuracy and relevance.
```