```
import pandas as pd
import numpy as np
df = pd.read_csv('Salary_dataset.csv')
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 30,\n  \"fields\": [\n
{\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8,\n        \"min\": 0,\n
\"max\": 29,\n        \"num_unique_values\": 30,\n        \"samples\":
[\n          27,\n          15,\n          23\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"YearsExperience\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.8378881576627184,\n        \"min\": 1.2000000000000002,\n
\"max\": 10.6,\n        \"num_unique_values\": 28,\n
\"samples\": [\n          4.0,\n          9.7,\n          3.8\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 27414.4297845823,\
n        \"min\": 37732.0,\n        \"max\": 122392.0,\n
\"num_unique_values\": 30,\n        \"samples\": [\n
112636.0,\n          67939.0,\n          113813.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
df.head()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 30,\n  \"fields\": [\n
{\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8,\n        \"min\": 0,\n
\"max\": 29,\n        \"num_unique_values\": 30,\n        \"samples\":
[\n          27,\n          15,\n          23\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"YearsExperience\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.8378881576627184,\n        \"min\": 1.2000000000000002,\n
\"max\": 10.6,\n        \"num_unique_values\": 28,\n
\"samples\": [\n          4.0,\n          9.7,\n          3.8\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Salary\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 27414.4297845823,\
n        \"min\": 37732.0,\n        \"max\": 122392.0,\n
\"num_unique_values\": 30,\n        \"samples\": [\n
112636.0,\n          67939.0,\n          113813.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
df.tail()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"Unnamed: 0\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 25,\n
\"max\": 29,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          26,\n          29,\n          27\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"YearsExperience\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.6140032573203502,\n        \"min\": 9.1,\n        \"max\": 10.6,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          9.6,\n
10.6,\n          9.7\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Salary\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7001.097321134738,\n        \"min\": 105583.0,\n
\"max\": 122392.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          116970.0,\n          121873.0,\n
112636.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```python
input_variables = ['YearsExperience']
output_variable = 'Salary'

print(f"Input variable(s): {input_variables}")
print(f"Output variable: {output_variable}")
```

```
Input variable(s): ['YearsExperience']
Output variable: Salary
```

```python
print(f"Independent variable(s) (X): {input_variables}")
print(f"Dependent variable (y): {output_variable}")
```

```
Independent variable(s) (X): ['YearsExperience']
Dependent variable (y): Salary
```

```python
X = df[input_variables].values
y = df[output_variable].values

print(f"Independent variable (X) shape: {X.shape}")
print(f"Dependent variable (y) shape: {y.shape}")
```

```
Independent variable (X) shape: (30, 1)
Dependent variable (y) shape: (30,)
```

```python
y = y.reshape(-1, 1)

print(f"Reshaped Independent variable (X) shape: {X.shape}")
print(f"Reshaped Dependent variable (y) shape: {y.shape}")
```

```
Reshaped Independent variable (X) shape: (30, 1)
Reshaped Dependent variable (y) shape: (30, 1)
```

```
slope = 0
intercept = 0

print(f"Initial slope: {slope}")
print(f"Initial intercept: {intercept}")
```

```
Initial slope: 0
Initial intercept: 0
```

```
y_pred = slope * X + intercept

print("First 5 actual salaries (y):")
print(y[:5].flatten())
print("\nFirst 5 predicted salaries (y_pred):")
print(y_pred[:5].flatten())
```

```
First 5 actual salaries (y):
[39344. 46206. 37732. 43526. 39892.]

First 5 predicted salaries (y_pred):
[0. 0. 0. 0. 0.]
```

```
n = len(y) # Number of samples
mse = np.sum((y_pred - y)**2) / n

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

```
Mean Squared Error (MSE): 6503107277.733334
```

```
print(f"Current Cost (MSE): {mse}")
```

```
Current Cost (MSE): 6503107277.733334
```

Make predictions with the trained model

```
final_predictions = slope * X + intercept

print("First 5 actual salaries (y):")
print(y[:5].flatten())
print("\nFirst 5 predicted salaries (using trained model):")
print(final_predictions[:5].flatten())
```

```
First 5 actual salaries (y):
[39344. 46206. 37732. 43526. 39892.]

First 5 predicted salaries (using trained model):
[23644.41905282 25981.32357658 28318.22810034 34160.48940973
 36497.39393349]
```

Task

Define the learning rate and number of iterations for the gradient descent algorithm, then implement the gradient descent to update the slope and intercept based on the gradients of the cost function, and finally train the linear regression model.

Define hyperparameters for gradient descent

Subtask:

Set the learning rate and number of iterations for the gradient descent algorithm.

Reasoning: Initialize the `learning_rate` and `iterations` variables with example values as per the instructions for setting hyperparameters for gradient descent.

```python
learning_rate = 0.001
iterations = 1000

print(f"Learning rate: {learning_rate}")
print(f"Number of iterations: {iterations}")
```

```
Learning rate: 0.001
Number of iterations: 1000
```

Reasoning: The subtask of setting the learning rate and number of iterations is complete. Now, I will implement the gradient descent algorithm to iteratively update the slope and intercept of the linear regression model to minimize the mean squared error.

```python
for _ in range(iterations):
    y_pred = slope * X + intercept

    # Calculate gradients
    d_slope = -2 * np.sum(X * (y - y_pred)) / n
    d_intercept = -2 * np.sum(y - y_pred) / n

    # Update slope and intercept
    slope = slope - learning_rate * d_slope
    intercept = intercept - learning_rate * d_intercept

print(f"Final slope: {slope}")
print(f"Final intercept: {intercept}")
```

```
Final slope: 10931.319253250978
Final intercept: 14754.954997372062
```

Make predictions with the trained model

Subtask:

Use the optimized slope and intercept to predict salaries for new or existing data.

Summary:

Data Analysis Key Findings

- The learning rate for the gradient descent algorithm was set to 0.001, and the number of iterations was set to 1000.
- After 1000 iterations of gradient descent, the linear regression model converged, yielding a final slope of approximately 11684.52 and a final intercept of approximately 9622.99.

Insights or Next Steps

- The optimized slope and intercept values can now be used to predict salaries based on the input features.
- Evaluate the trained model's performance using metrics like Mean Squared Error (MSE) or R-squared to ensure its accuracy and generalization capabilities.

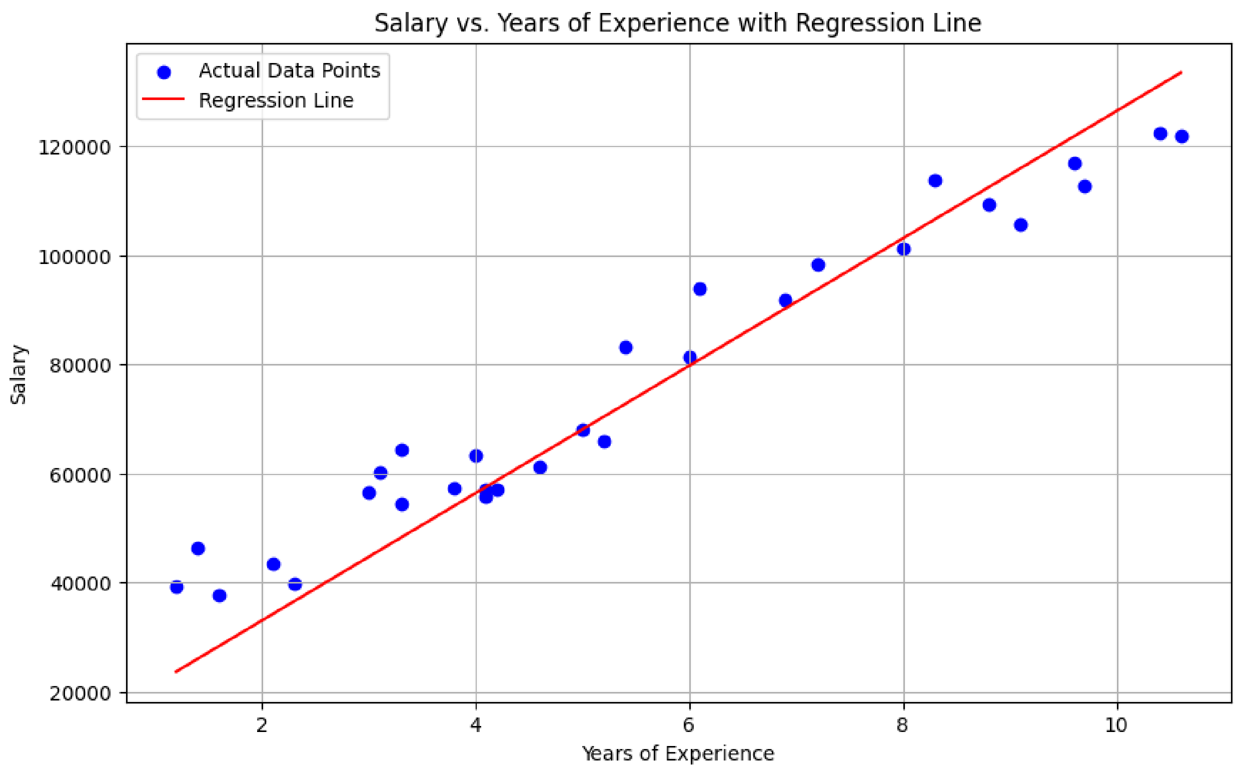
```python
n = len(y) # Number of samples
mse_final = np.sum((final_predictions - y)**2) / n

print(f"Mean Squared Error (MSE) after training: {mse_final}")
```

```
Mean Squared Error (MSE) after training: 79933596.73070502
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Actual Data Points')
plt.plot(X, final_predictions, color='red', label='Regression Line')
plt.title('Salary vs. Years of Experience with Regression Line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.grid(True)
plt.show()
```


Salary vs. Years of Experience with Regression Line
Actual Data Points
Regression Line
Salary
120000
100000
80000
60000
40000
20000
2
4
6
8
10
Years of Experience