Bias in a Linear Regression Model (Gradient Descent)

1. Importing necessary libraries

First, we need to import the libraries required for data manipulation, modeling, and visualisation.

```
import numpy as np
import matplotlib.pyplot as plt
```

2. Defining the dataset and the parameters

Then, we create a small sample dataset, with `X` representing the input feature values (independent variable) and `Y` representing the corresponding target values (dependent variable). We also initialise the model parameters: the bias (`b`) and weight (`w`), both set to zero. We also define the learning rate and the number of iterations for the gradient descent process.

```
X = np.array([1, 2, 3, 4, 5])
Y = np.array([1, 2, 1.3, 3.75, 2.25])

learning_rate = 0.01
iterations = 1000
m = len(X)  # Number of data points
b = 0
w = 0
```

3. Defining the cost function

We define the cost function as the Mean Squared Error (MSE), which will measure the difference between the predicted values and the actual target values, in order to guide the optimisation process.

```
def compute_cost(X, Y, w, b):
    total_cost = 0
    for i in range(m):
        total_cost += (Y[i] - (w * X[i] + b))**2
    return total_cost / m
```

4. Implementing gradient descent to update bias and weight

We now implement the gradient descent technique, which iteratively updates the model parameters (bias and weight) based on the gradients of the cost function. The bias values are stored during each iteration.

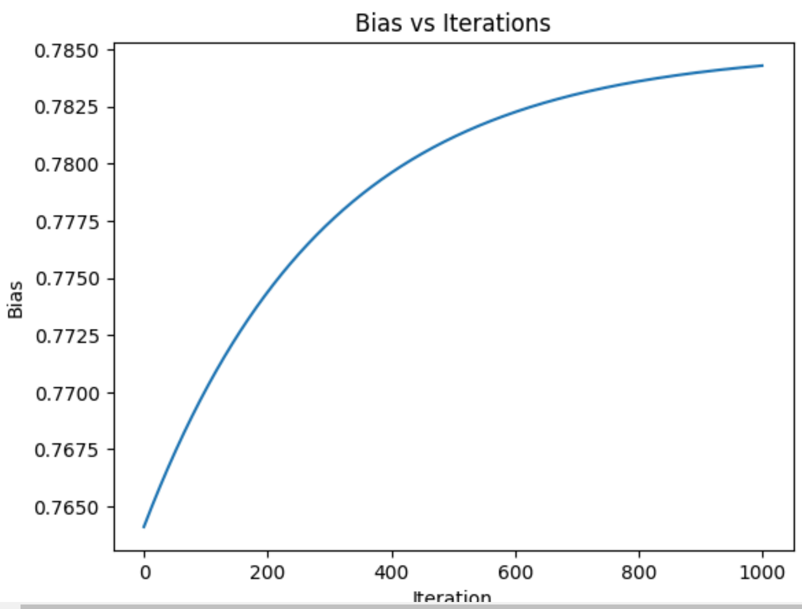
```
biases = []
for i in range(iterations):
    Y_pred = w * X + b

    # Computing the gradients
    dw = -2 * np.dot(X, (Y - Y_pred)) / m
    db = -2 * np.sum(Y - Y_pred) / m

    # Updating the parameters
    w -= learning_rate * dw
    b -= learning_rate * db

    biases.append(b)

    if i % 100 == 0:  # For printing every 100 iterations
        cost = compute_cost(X, Y, w, b)
        print(f"Iteration {i}: Cost = {cost}, Bias = {b}")
```

```
Iteration 0: Cost = 0.558229422298238, Bias = 0.7641111439140331
Iteration 100: Cost = 0.5581903439348103, Bias = 0.7701121302883474
Iteration 200: Cost = 0.5581704934018791, Bias = 0.7743891430129556
Iteration 300: Cost = 0.5581604099791593, Bias = 0.7774374481923778
Iteration 400: Cost = 0.5581552879295851, Bias = 0.7796100310358718
Iteration 500: Cost = 0.5581526860956076, Bias = 0.7811584705568585
Iteration 600: Cost = 0.5581513644488827, Bias = 0.7822620717186435
Iteration 700: Cost = 0.5581506930954909, Bias = 0.7830486284473918
Iteration 800: Cost = 0.5581503520698836, Bias = 0.7836092218111564
Iteration 900: Cost = 0.5581501788400078, Bias = 0.7840087669526706
```

5. Visualising the bias

Finally, after training, we plot how the bias value evolves through the iterations of gradient descent, in order to understand how the model parameters are adjusted during training.

```
plt.plot(biases)
plt.title('Bias vs Iterations')
plt.xlabel('Iteration')
plt.ylabel('Bias')
plt.show()
```



```
Start coding or generate with AI.
```