Implementing Perceptron from scratch with Python in OOP

In [4]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets
import matplotlib.pyplot as plt
```

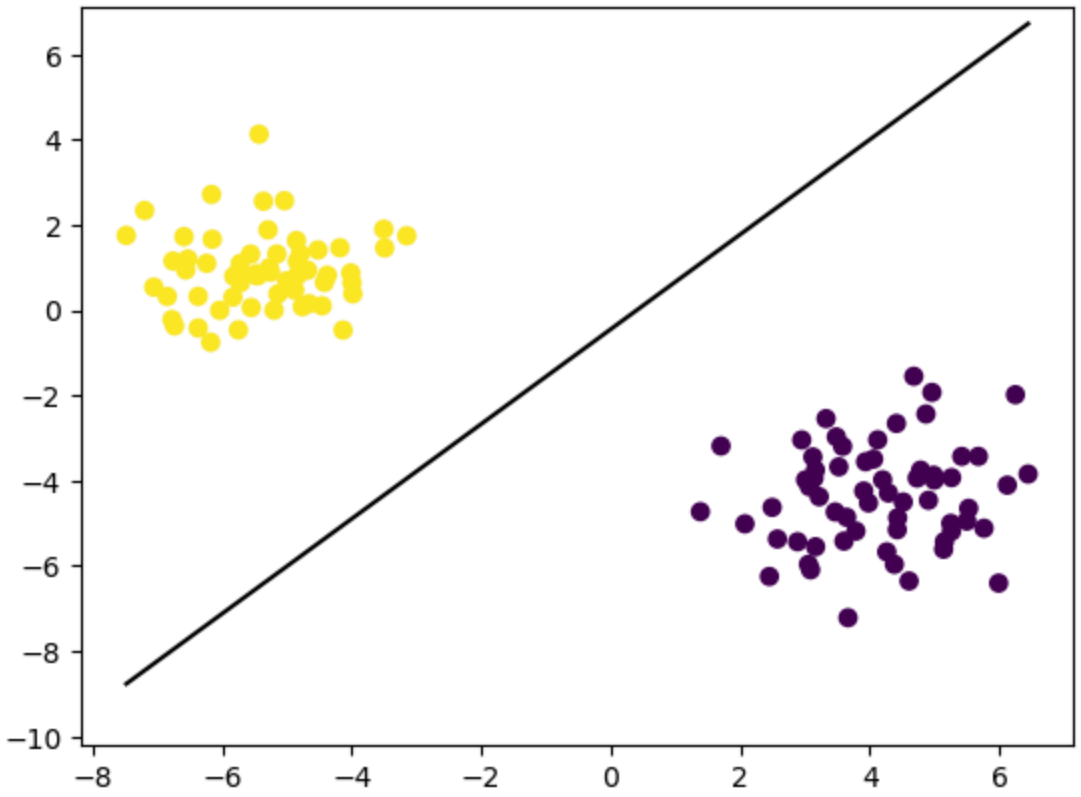
In [5]:
```
def unit_step_func(x):
    return np.where(x > 0, 1, 0)
```

In [6]:
```
class Perceptron:
    def __init__(self, learning_rate=0.01, n_iters=1000):
        self.lr = learning_rate
        self.n_iters = n_iters
        self.activation_func = unit_step_func
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape

        # init parameters
        # self.weights = np.zeros(n_features)
        # random init instead
        self.weights = np.random.rand(n_features)
        self.bias = 0

        # ensure correct class labels
        y_ = np.where(y > 0, 1, 0)

        # optimization: learn weights
        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X): # both index and the ssample
                linear_output = np.dot(x_i, self.weights) + self.bias
                y_pred = self.activation_func(linear_output)

            # update weights and bias parameters
            update = self.lr * (y_[idx] - y_pred)
            self.weights += update * x_i
            self.bias += update


    def predict(self, X):
        linear_output = np.dot(X, self.weights) + self.bias
        y_pred = self.activation_func(linear_output)
        return y_pred
```

In [12]:
```
# Testing
if __name__ == "__main__":
    def accuracy(y_true, y_pred):
```

```
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy

    # Create dataset
    X, y = datasets.make_blobs(
        n_samples=150, n_features=2, centers=2, random_state=123, cluster_st
    )

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

    # Initialize, fit model and make predictions
    p = Perceptron(learning_rate=0.01, n_iters=1000)
    p.fit(X_train, y_train)
    predictions = p.predict(X_test)
    print(f"Perceptron classification accuracy: {accuracy(y_test, predictior

    # Plot the decision boundary
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    plt.scatter(X_train[:, 0], X_train[:, 1], marker="o", c=y_train)

    x0_1 = np.amin(X_train[:, 0])
    x0_2 = np.amax(X_train[:, 0])

    x1_1 = (-p.weights[0] * x0_1 - p.bias) / p.weights[1]
    x1_2 = (-p.weights[0] * x0_2 - p.bias) / p.weights[1]

    ax.plot([x0_1, x0_2], [x1_1, x1_2], "k")

    ymin = np.amin(X_train[:, 1])
    ymax = np.amax(X_train[:, 1])
    ax.set_ylim([ymin - 3, ymax + 3])

    plt.show()
```

```
Perceptron classification accuracy: 1.0
```


6
4
2
0
−2
−4
−6
−8
−10
−8
−6
−4
−2
0
2
4
6