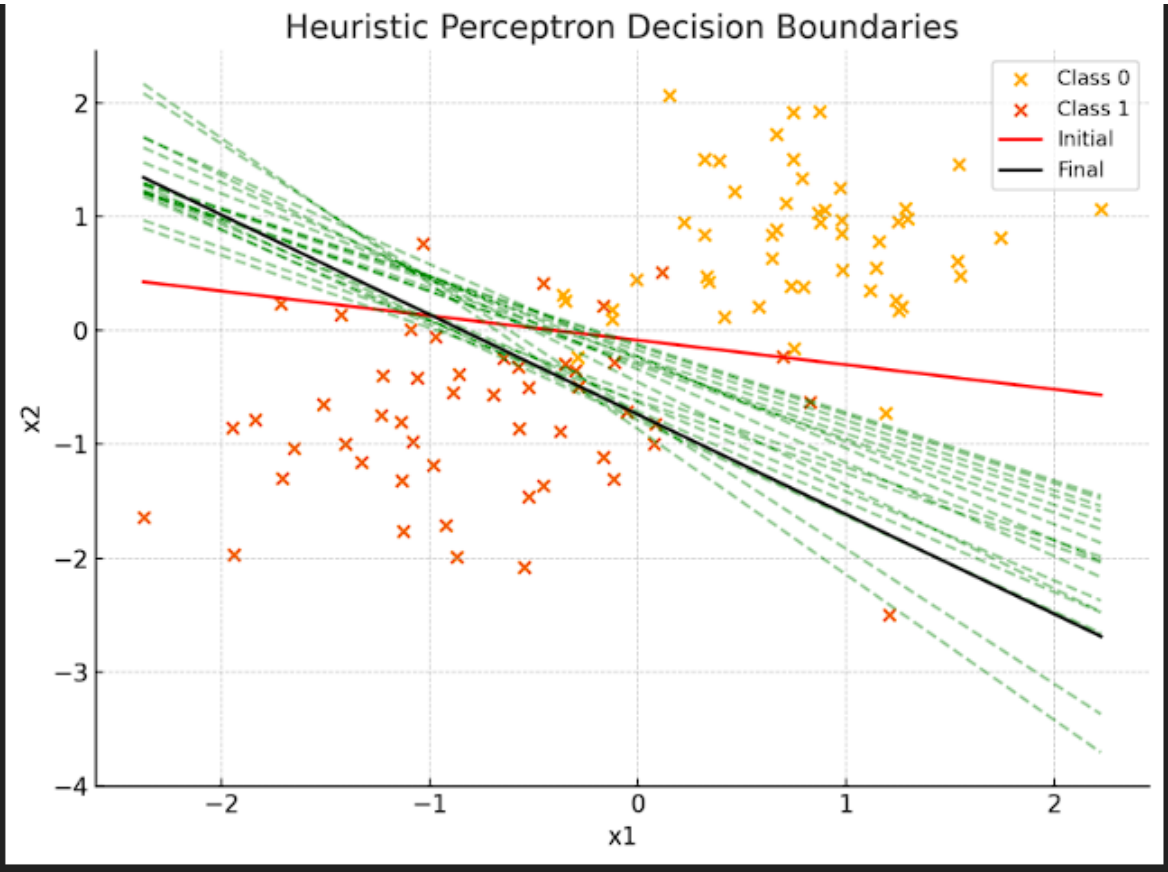
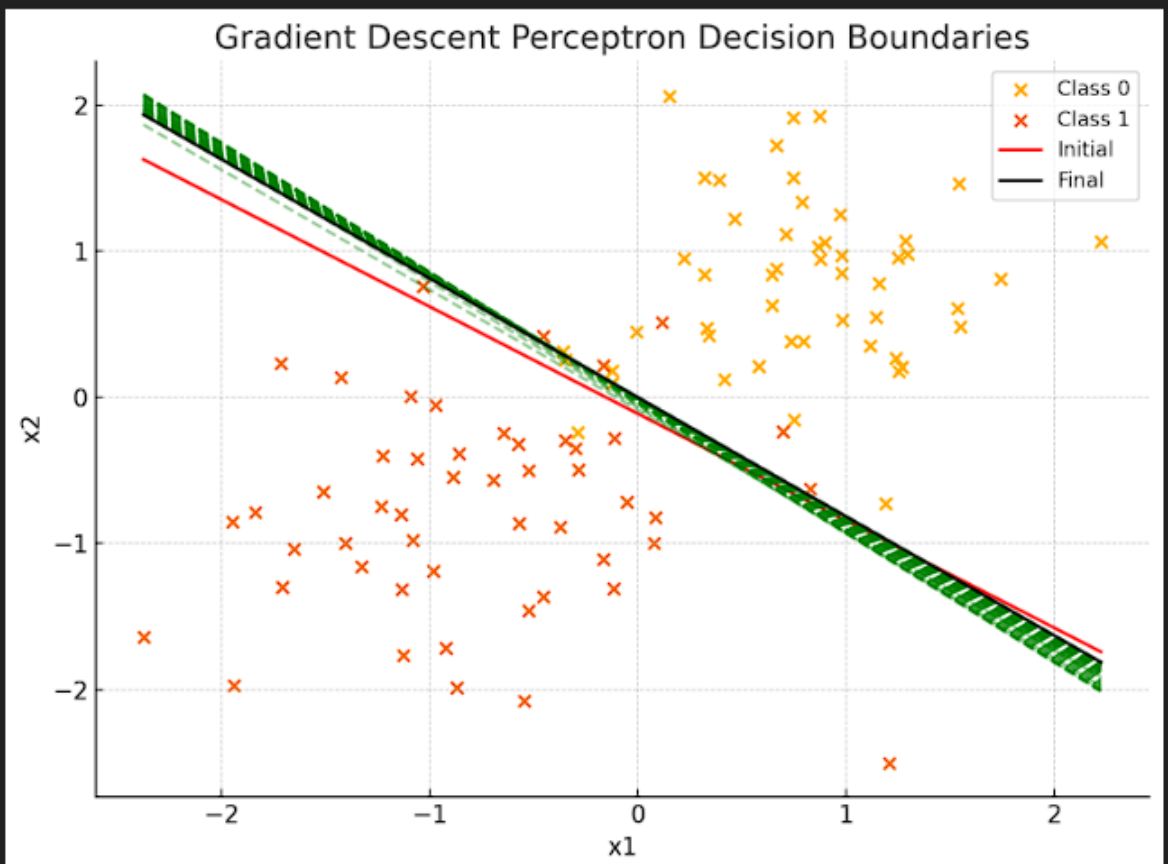
Perceptron Assignment Report

This report compares two ways to train a perceptron: the heuristic approach and the gradient descent method. Both were tested using the same dataset to classify two groups. We also tracked the decision boundaries and calculated how the model's error changed over time.





Heuristic Perceptron Code

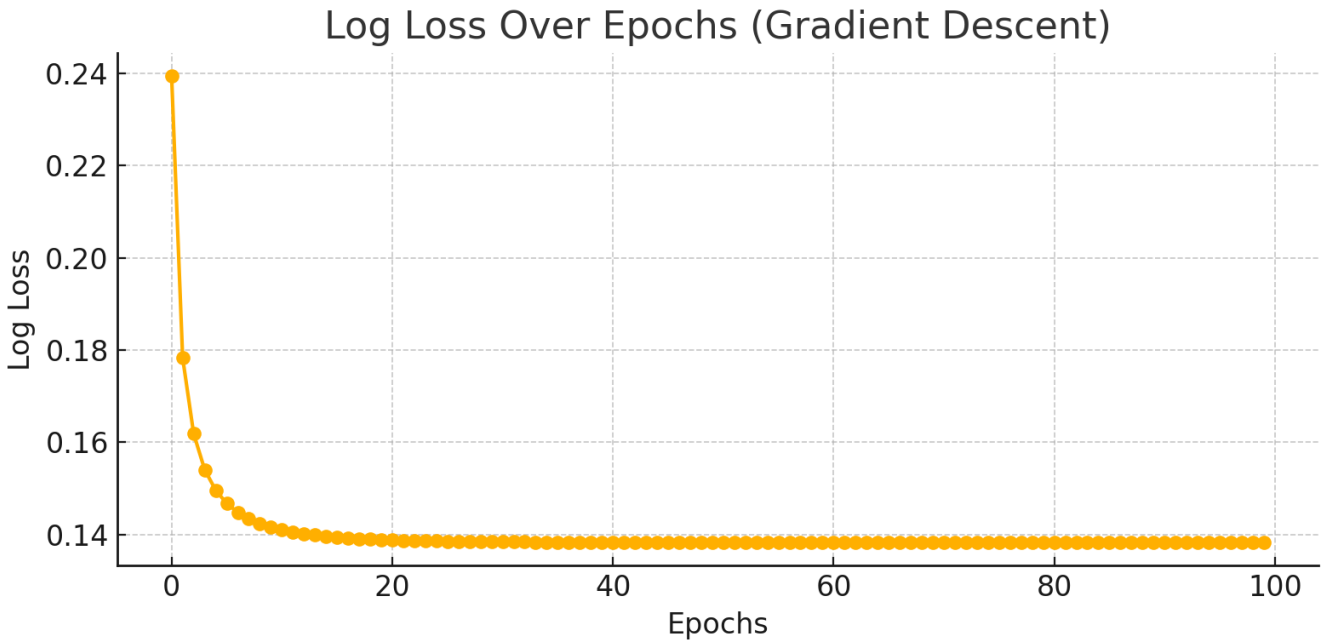
```
def heuristic_perceptron(X, y, learning_rate=0.1, max_epochs=20):
    for epoch in range(max_epochs):
        for i in range(len(X)):
            y_pred = 1 if np.dot(w, X[i]) + b >= 0 else 0
            error = y[i] - y_pred
```

```
            w += learning_rate * error * X[i]
            b += learning_rate * error
```

Gradient Descent Perceptron Code

```
def gradient_descent_perceptron(X, y, learning_rate=0.1, epochs=100):
    for epoch in range(epochs):
        for i in range(len(X)):
            y_hat = sigmoid(np.dot(w, X[i]) + b)
            error = y[i] - y_hat
            w += learning_rate * error * X[i]
            b += learning_rate * error
```

Log Loss Over Epochs



Conclusion

The heuristic method worked quickly but had more jumps between steps. Gradient descent was smoother and reduced the error steadily. Tracking the log loss helped see how well the model learned. Learning rate played a big role in how fast or slow the model improved.