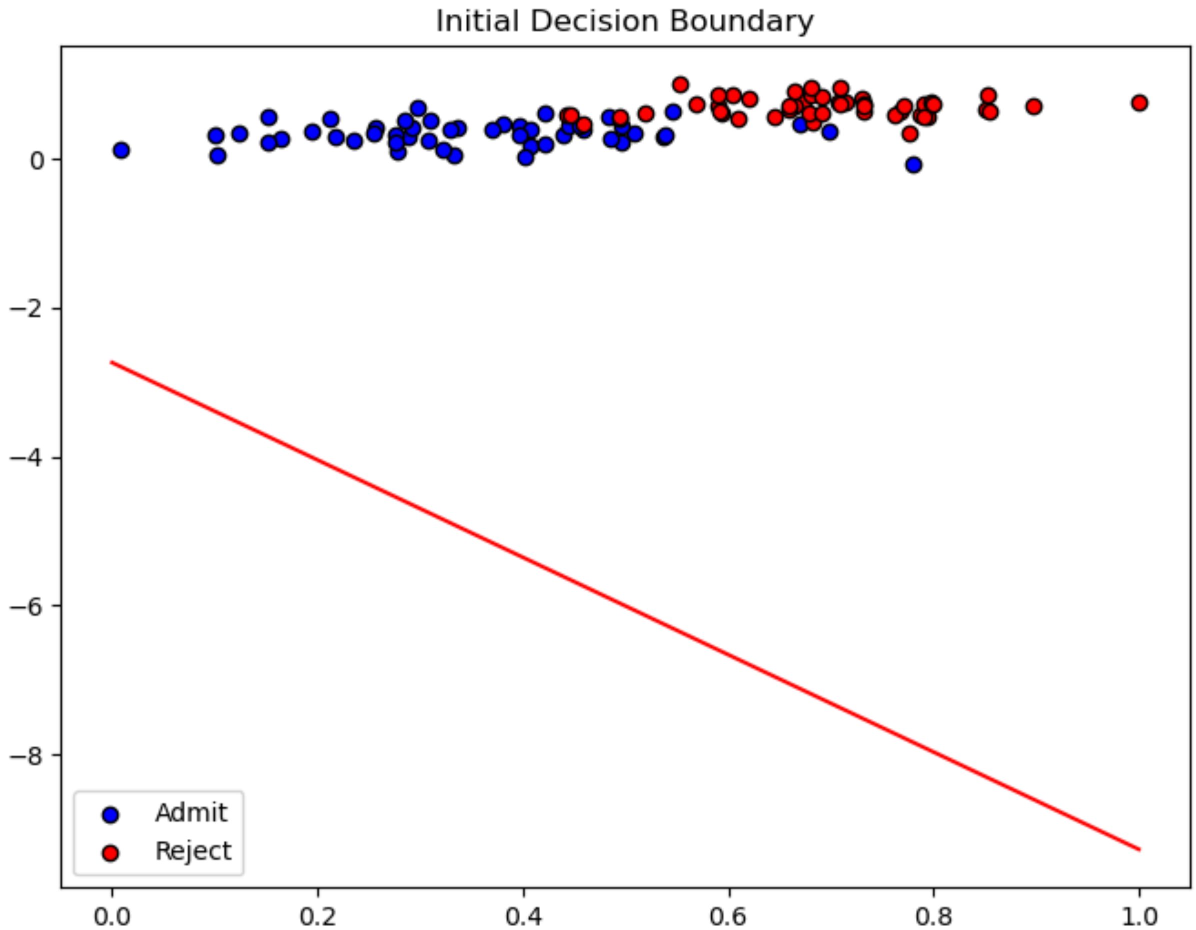
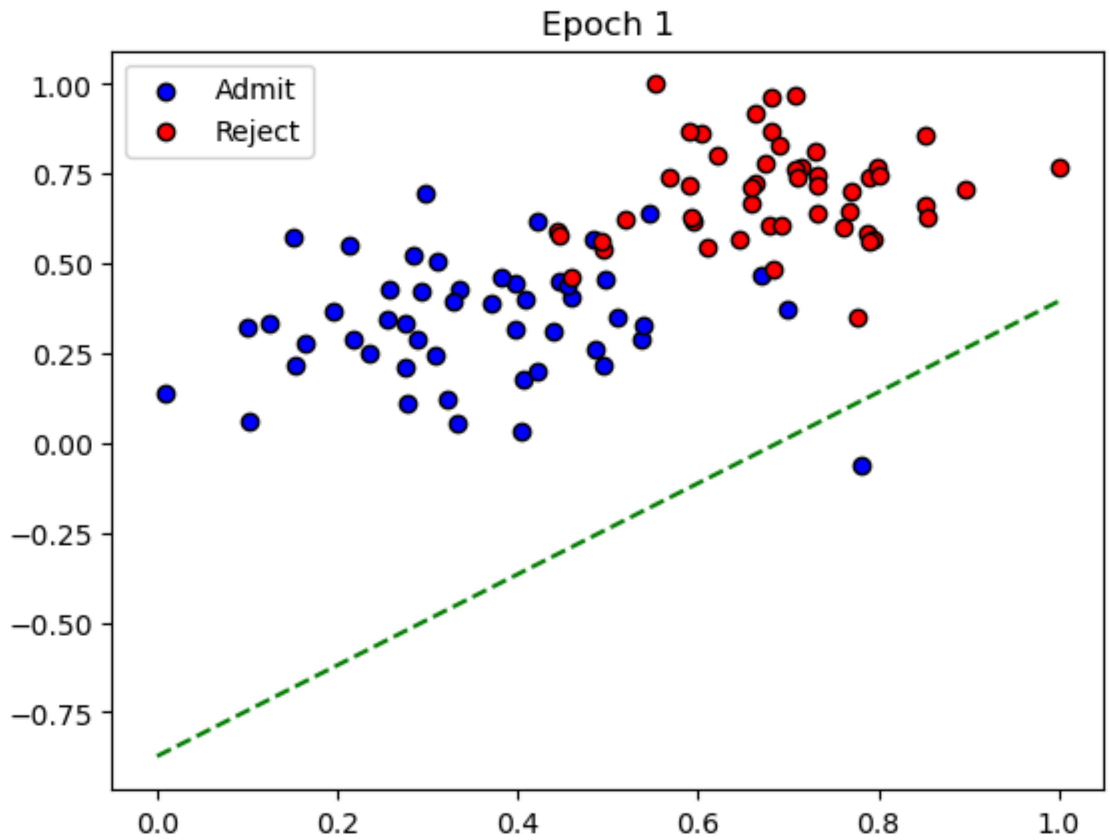
In [31]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('data.csv', header=None, names=['x1', 'x2', 'label'])
X = np.array(data[['x1', 'x2']])
Y = np.array(data['label'])
```

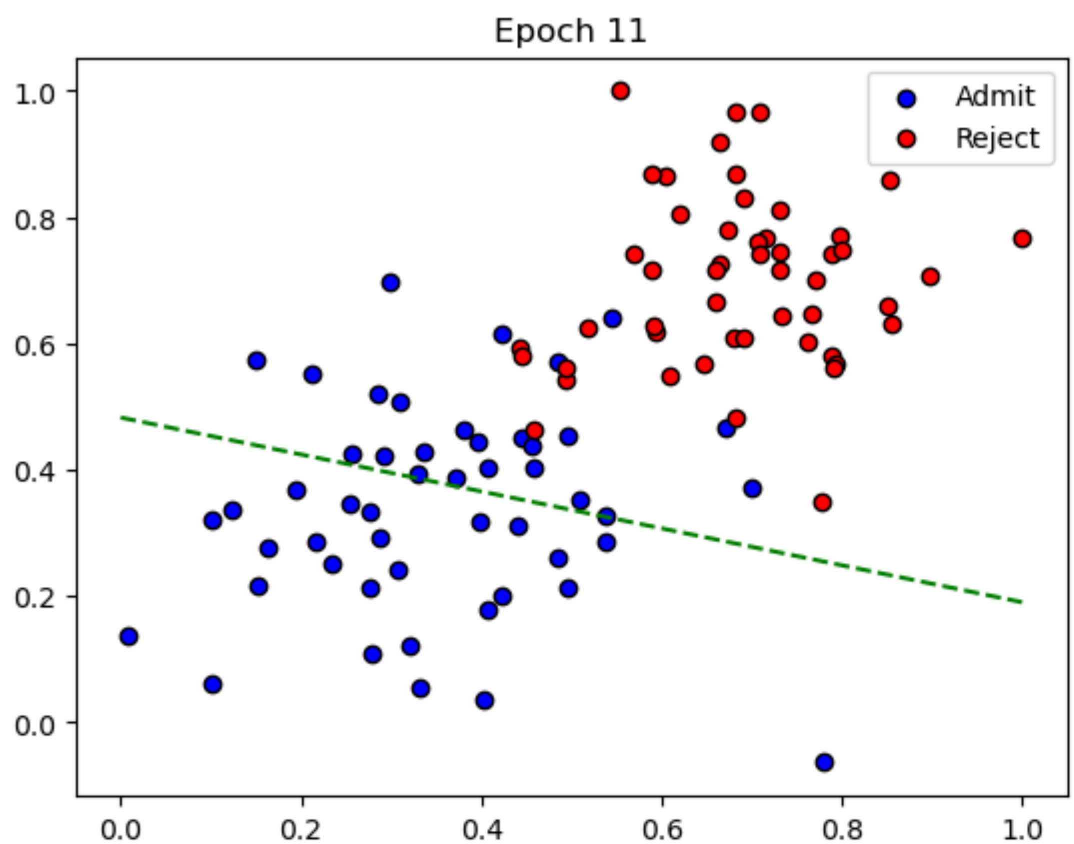
In [33]:
```
def step_function(t):
    return 1 if t >= 0 else 0

def predict(x, weights, bias):
    return step_function(np.dot(weights, x) + bias)

def plot_points(data):
    admitted = data[data['label'] == 1]
    rejected = data[data['label'] == 0]
    plt.scatter(admitted['x1'], admitted['x2'], color='blue', edgecolor='k',
    plt.scatter(rejected['x1'], rejected['x2'], color='red', edgecolor='k',
    plt.legend()

def plot_decision_boundary(weights, bias, color='g--'):
    x_vals = np.array([0, 1])
    y_vals = -(weights[0] * x_vals + bias) / weights[1]
    plt.plot(x_vals, y_vals, color)
```

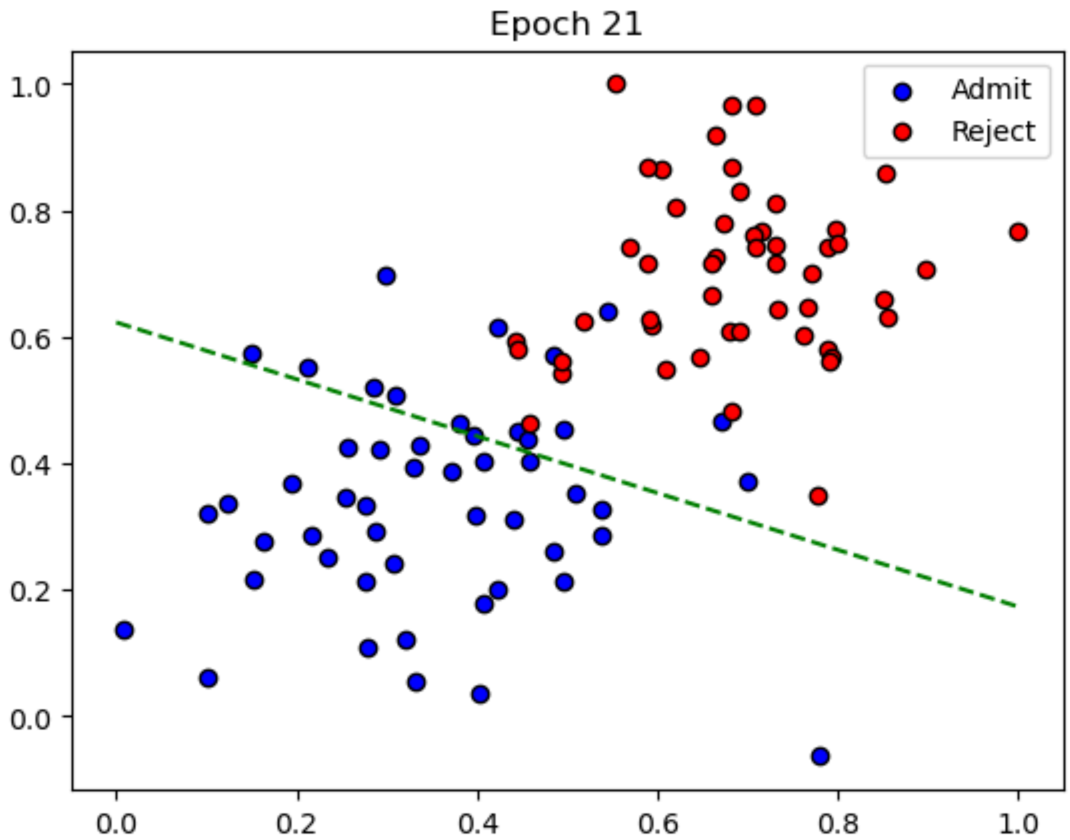
In [35]:
```
def train_perceptron(X, Y, learning_rate=0.1, num_epochs=65):
    weights = np.random.rand(2)
    bias = np.random.rand()

    plt.figure(figsize=(8, 6))
    plot_points(data)
    plot_decision_boundary(weights, bias, 'r-')
    plt.title("Initial Decision Boundary")
    plt.show()

    for epoch in range(num_epochs):
        for i in range(len(X)):
            x = X[i]
            y = Y[i]
            y_hat = predict(x, weights, bias)
            error = y - y_hat
            weights[0] += learning_rate * error * x[0]
            weights[1] += learning_rate * error * x[1]
            bias += learning_rate * error

        if epoch % 10 == 0:
            plot_points(data)
            plot_decision_boundary(weights, bias, 'g--')
            plt.title(f"Epoch {epoch + 1}")
            plt.show()

    plot_points(data)
```

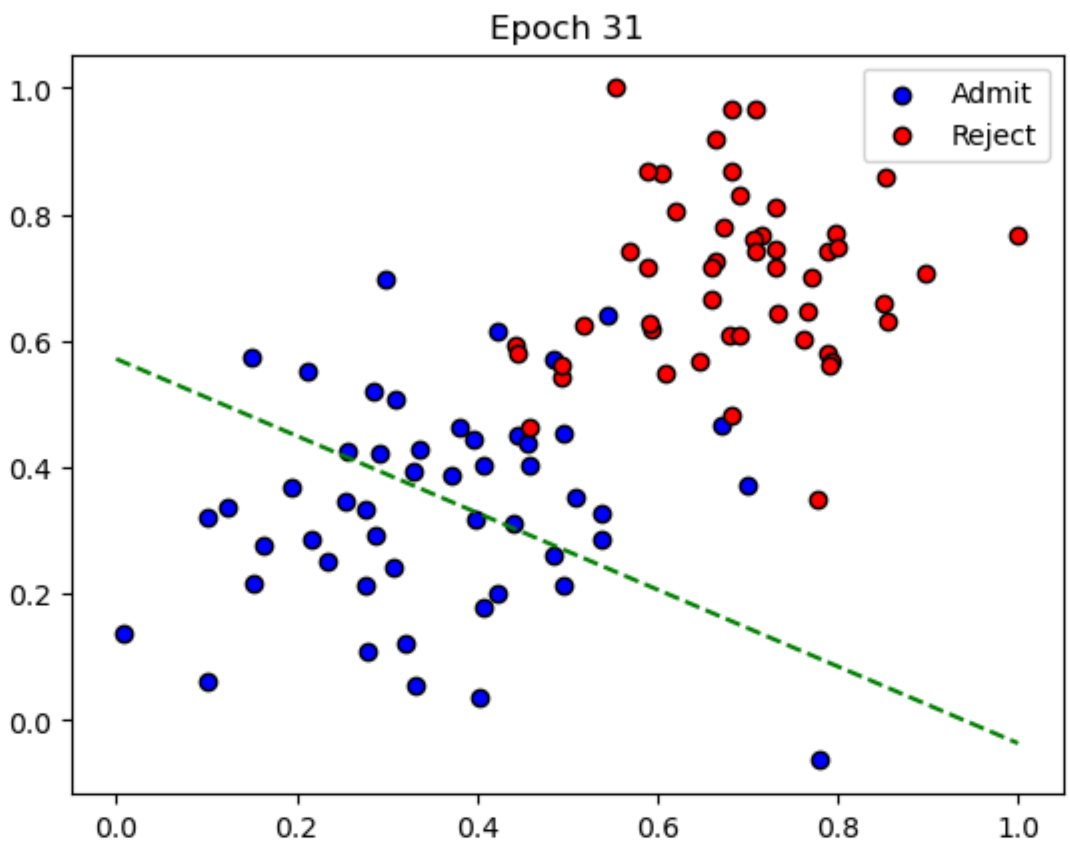
```
    plot_decision_boundary(weights, bias, 'k-')
    plt.title("Final Decision Boundary")
    plt.show()

train_perceptron(X, Y)
```

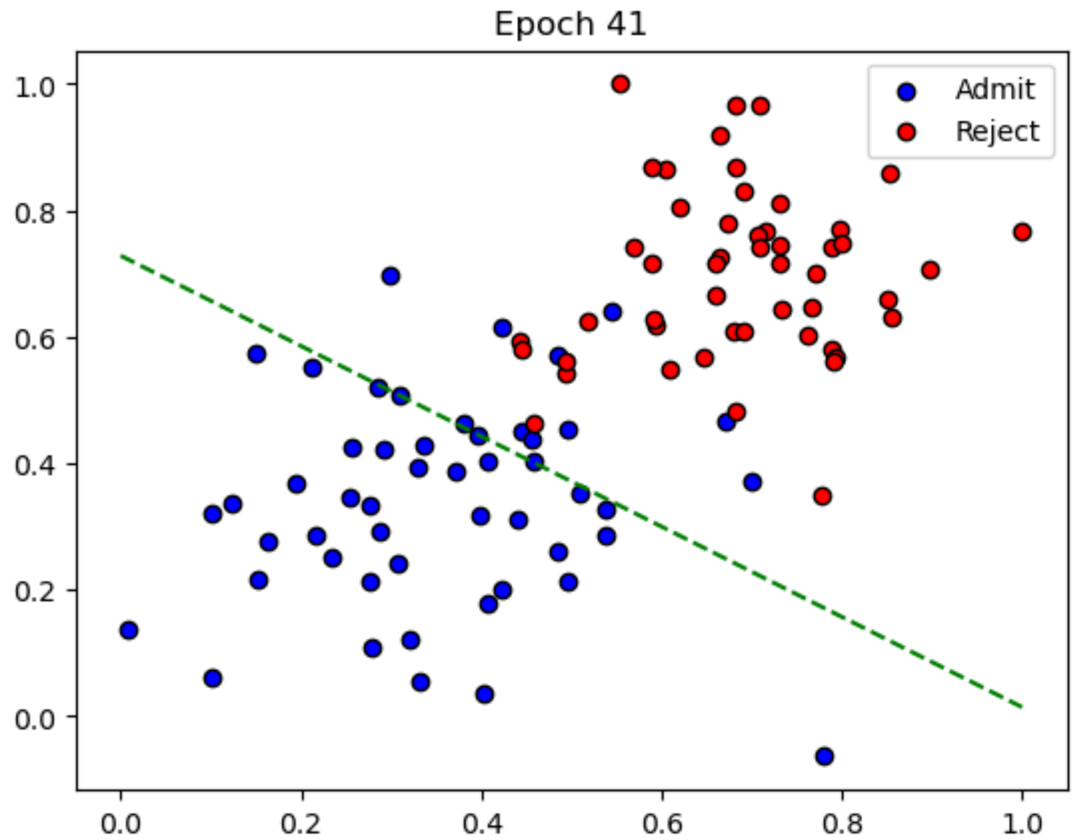


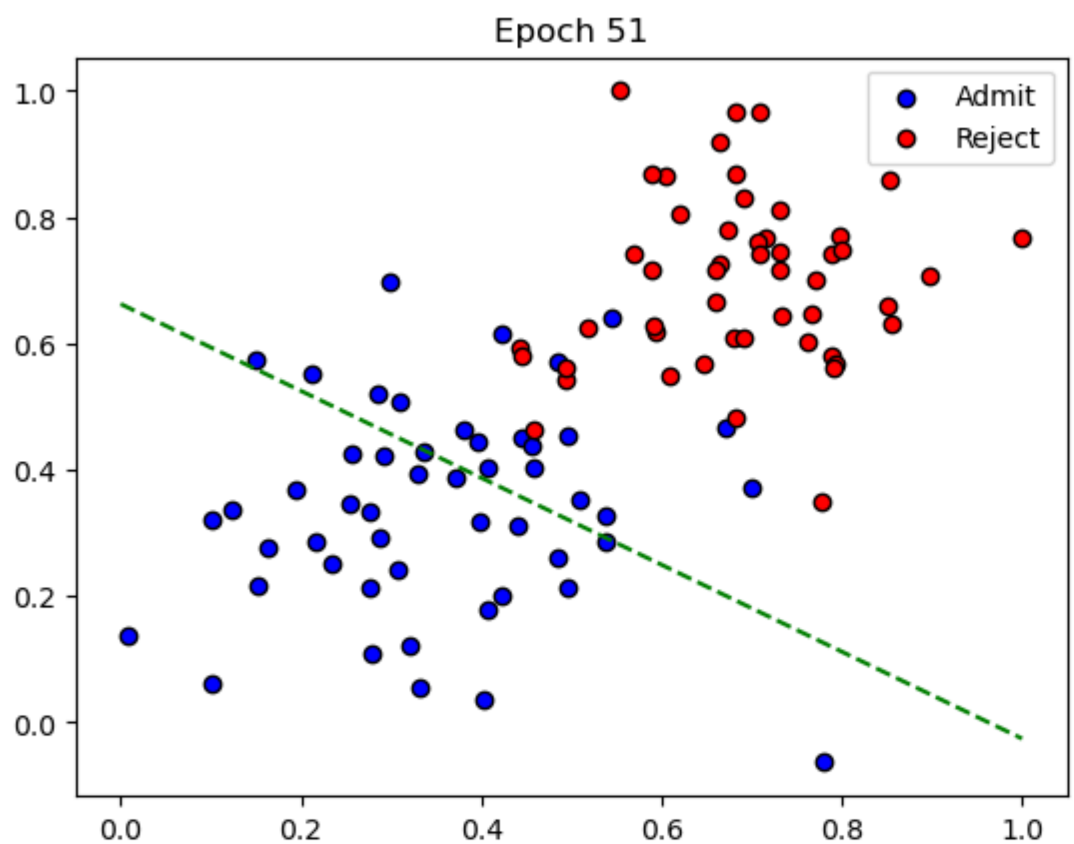

Epoch 1
Admit
Reject
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0.0
0.2
0.4
0.6
0.8
1.0

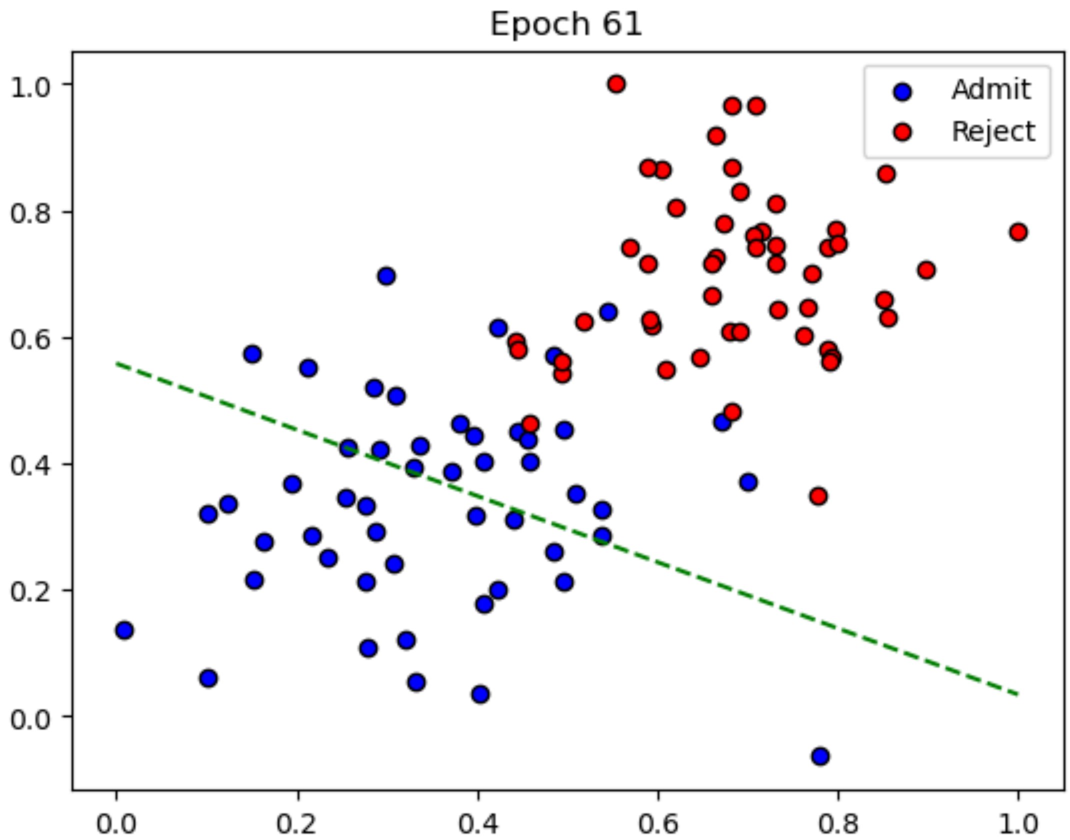

Epoch 11
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

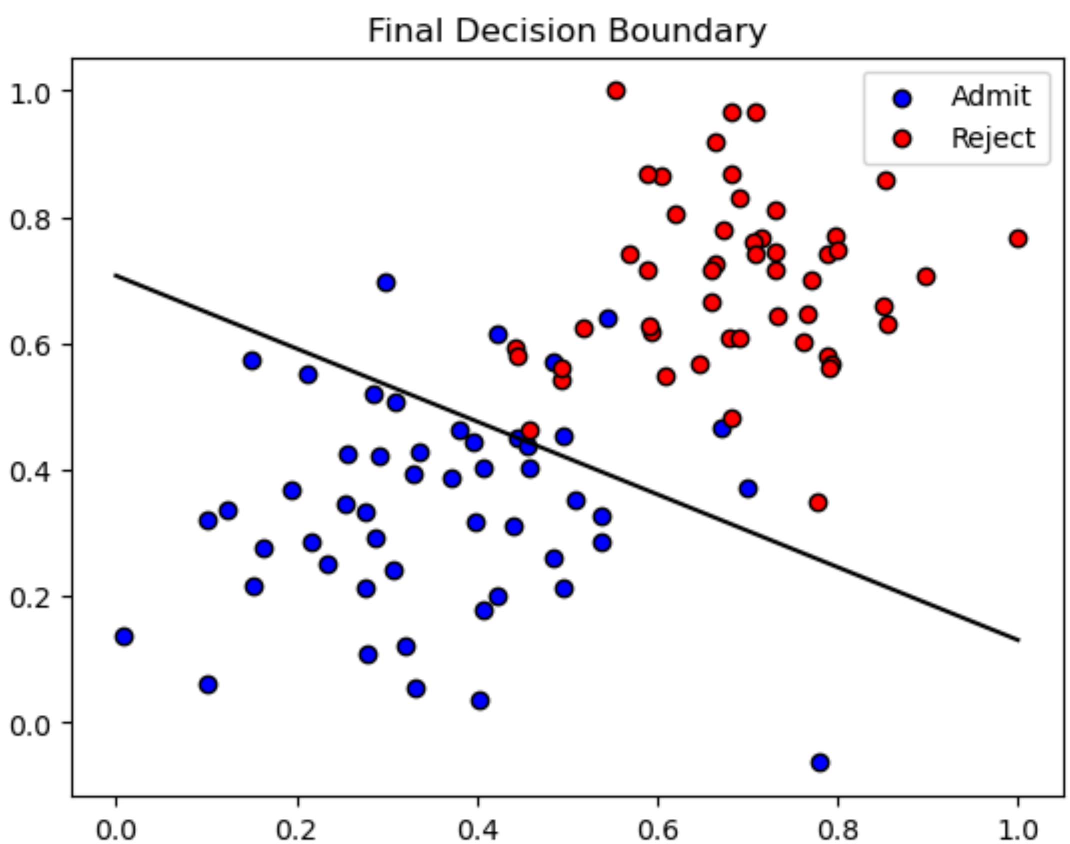

Epoch 21
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Epoch 31
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Epoch 41
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Epoch 51
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Epoch 61
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Final Decision Boundary
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]:

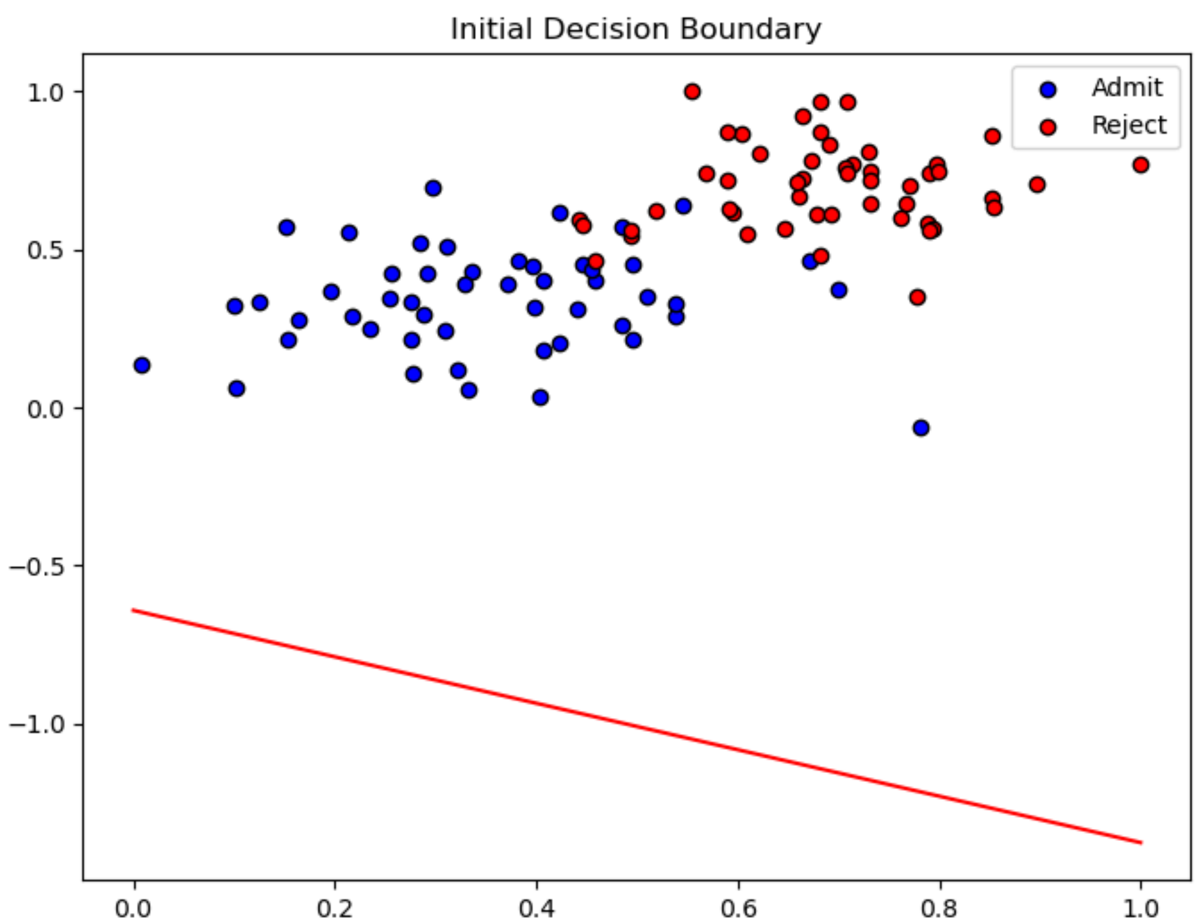
In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('data.csv', header=None, names=['x1', 'x2', 'label'])
X = np.array(data[['x1', 'x2']])
Y = np.array(data['label'])
```

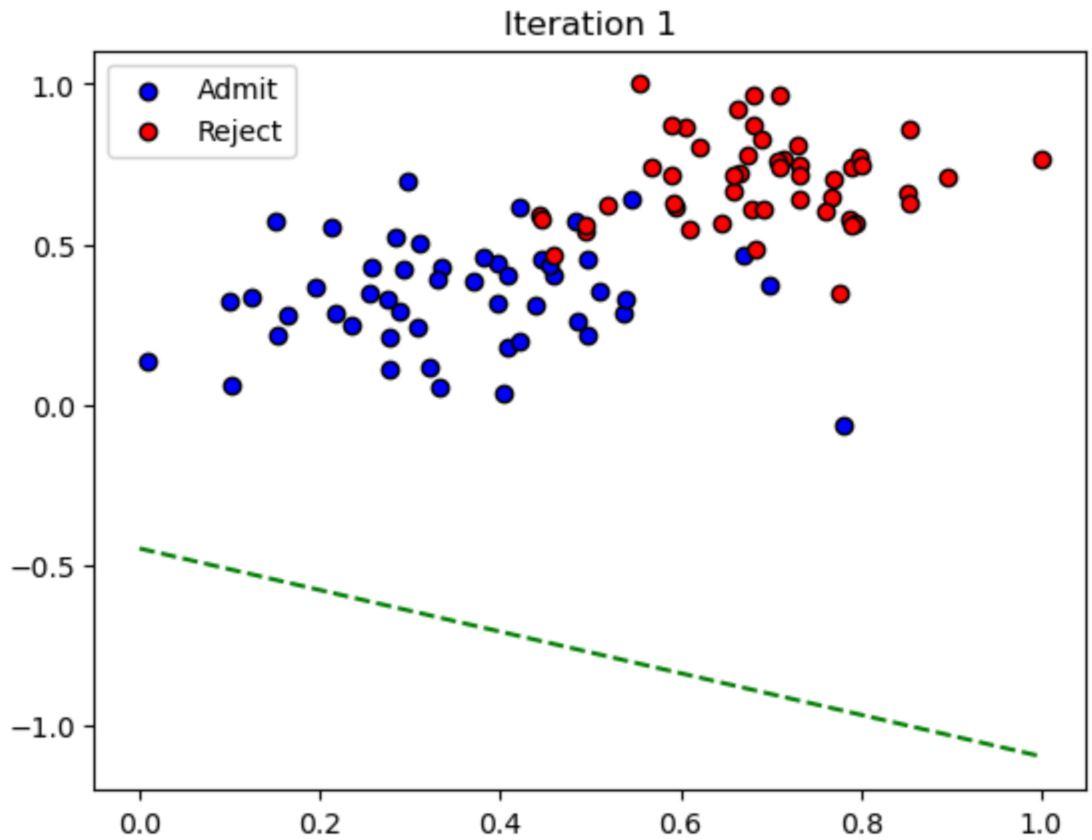
In [13]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def log_loss(y, y_hat):
    epsilon = 1e-9  # To prevent log(0)
    return -y * np.log(y_hat + epsilon) - (1 - y) * np.log(1 - y_hat + epsil

def predict_prob(x, weights, bias):
    return sigmoid(np.dot(weights, x) + bias)

def predict(x, weights, bias):
    return 1 if predict_prob(x, weights, bias) >= 0.5 else 0

def plot_points(data):
    admitted = data[data['label'] == 1]
    rejected = data[data['label'] == 0]
    plt.scatter(admitted['x1'], admitted['x2'], color='blue', edgecolor='k',
    plt.scatter(rejected['x1'], rejected['x2'], color='red', edgecolor='k',
    plt.legend()

def plot_decision_boundary(weights, bias, color='g--'):
    x_vals = np.array([0, 1])
    y_vals = -(weights[0] * x_vals + bias) / weights[1]
    plt.plot(x_vals, y_vals, color)
```

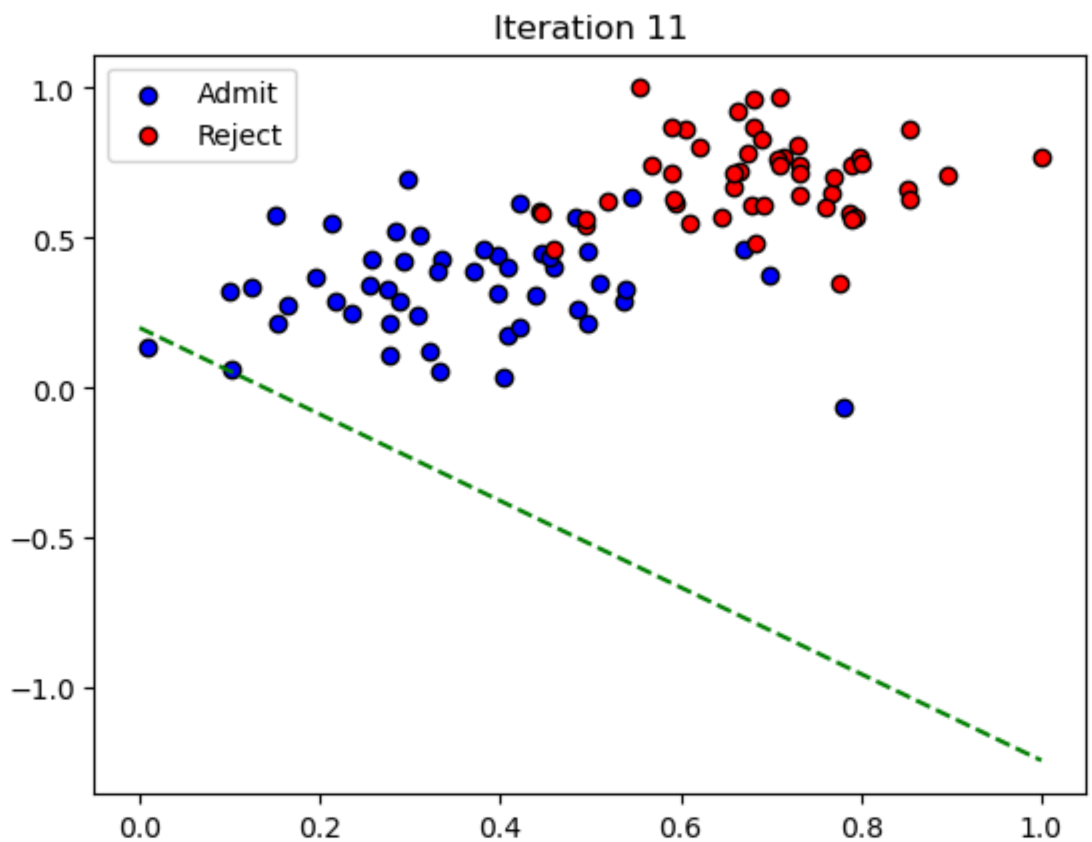
In [15]:
```
def train_gradient_descent(X, Y, learning_rate=0.01, num_iterations=100):
    weights = np.random.rand(2)
    bias = np.random.rand()
    losses = []

    plt.figure(figsize=(8, 6))
    plot_points(data)
    plot_decision_boundary(weights, bias, 'r-')
    plt.title("Initial Decision Boundary")
    plt.show()

    for iteration in range(num_iterations):
        total_loss = 0
        for i in range(len(X)):
            x = X[i]
            y = Y[i]
            y_hat = predict_prob(x, weights, bias)

            error = y - y_hat
            weights[0] += learning_rate * error * x[0]
            weights[1] += learning_rate * error * x[1]
```

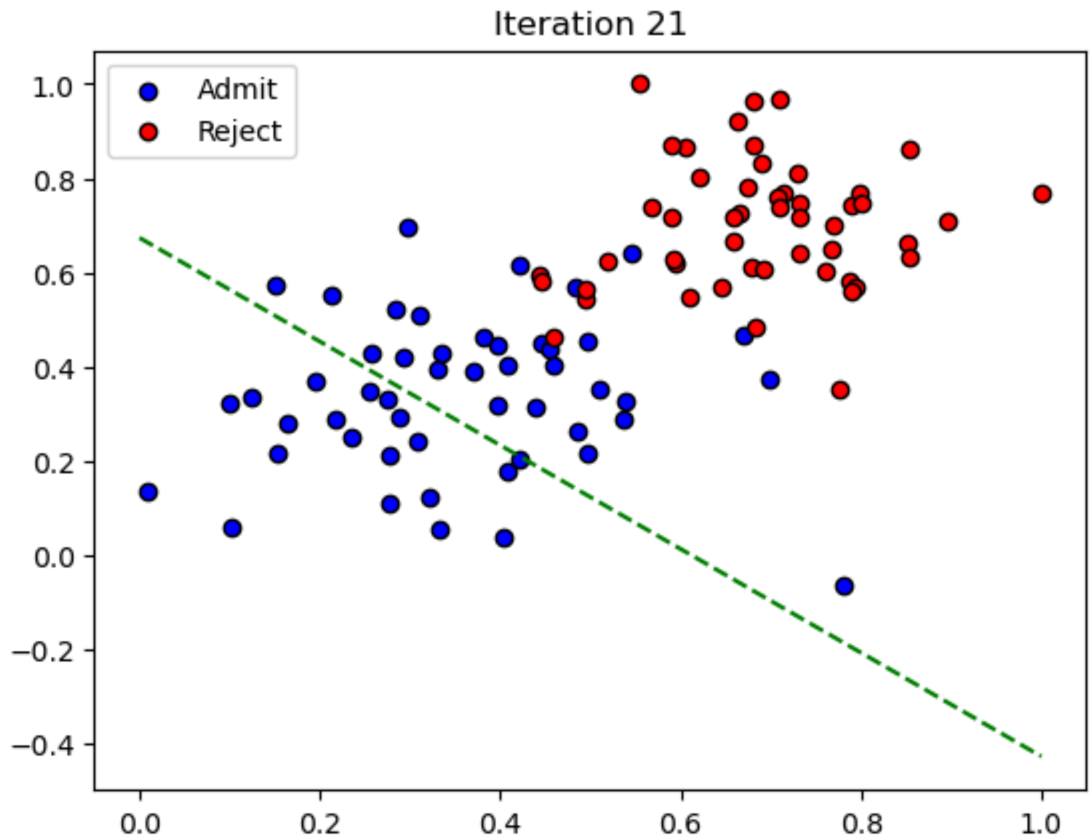
```
            bias += learning_rate * error

            total_loss += log_loss(y, y_hat)

        losses.append(total_loss / len(X))

        if iteration % 10 == 0:
            plot_points(data)
            plot_decision_boundary(weights, bias, 'g--')
            plt.title(f"Iteration {iteration + 1}")
            plt.show()

    plot_points(data)
    plot_decision_boundary(weights, bias, 'k-')
    plt.title("Final Decision Boundary After All Iterations")
    plt.show()

    plt.plot(range(num_iterations), losses, marker='o')
    plt.title("Log Loss Over Iterations")
    plt.xlabel("Iteration")
    plt.ylabel("Log Loss")
    plt.show()
```

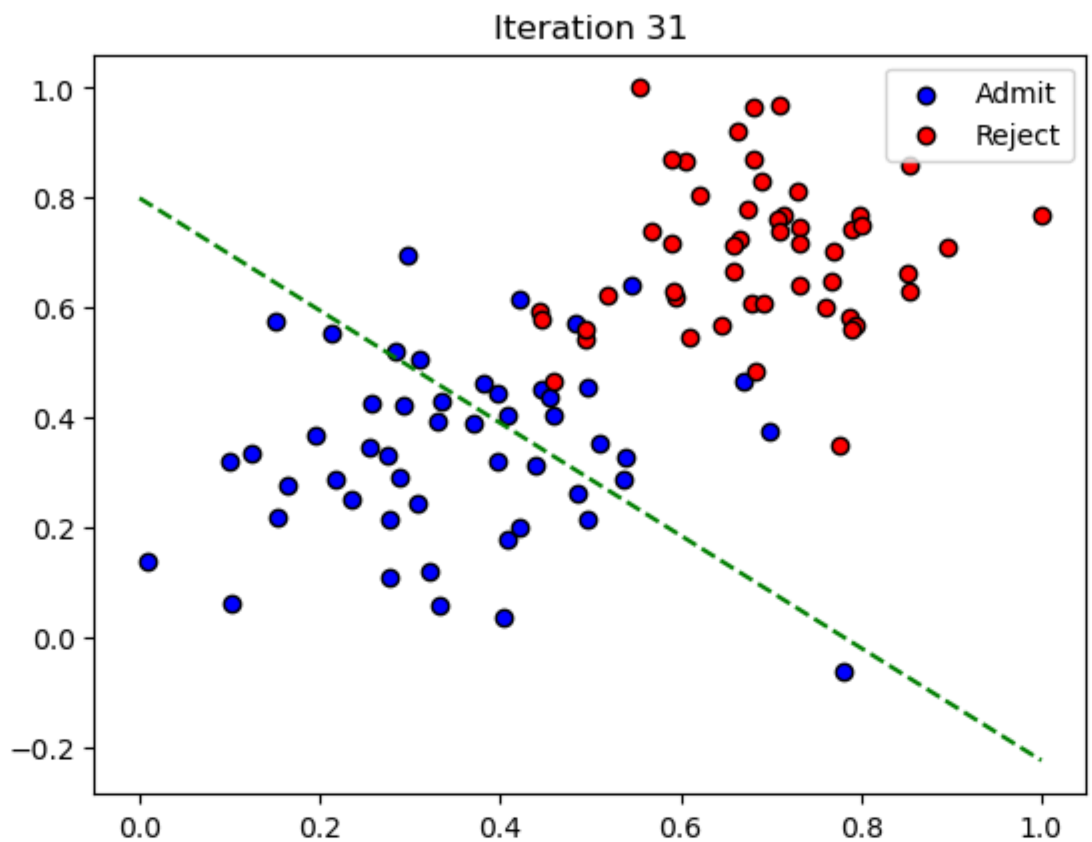
In [17]:
```
train_gradient_descent(X, Y, learning_rate=0.01, num_iterations=100)
```

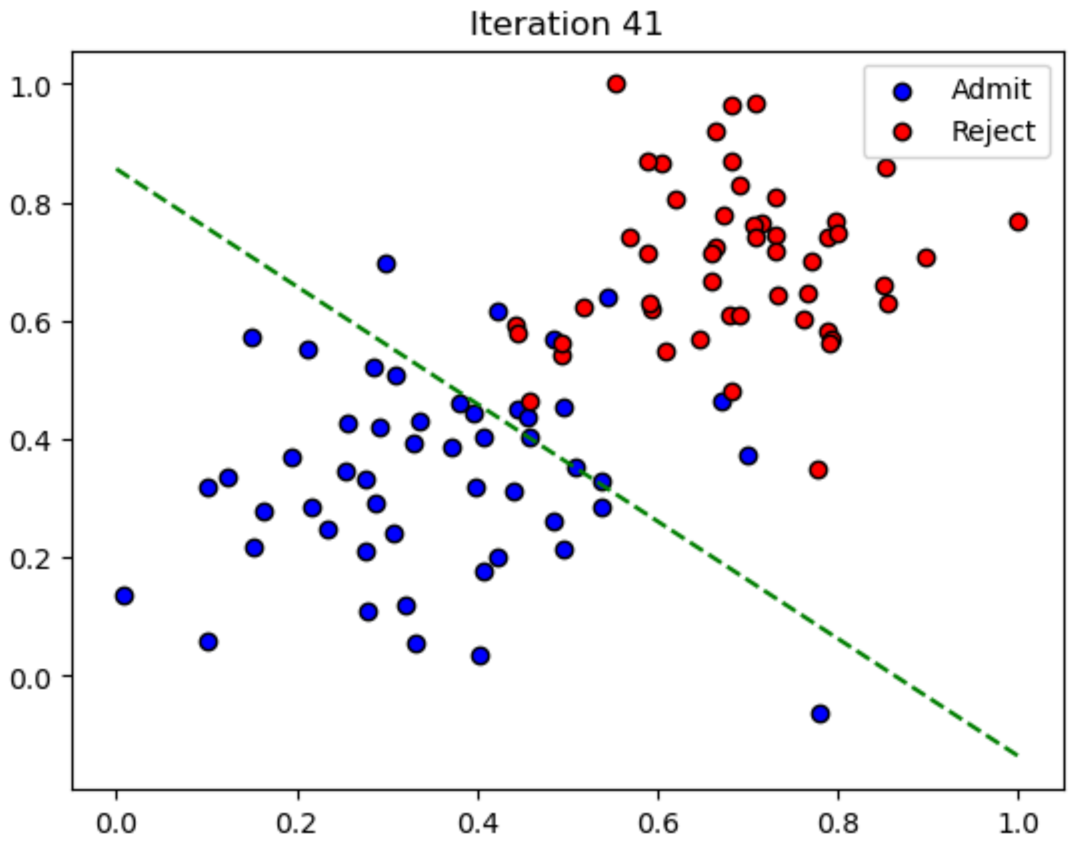


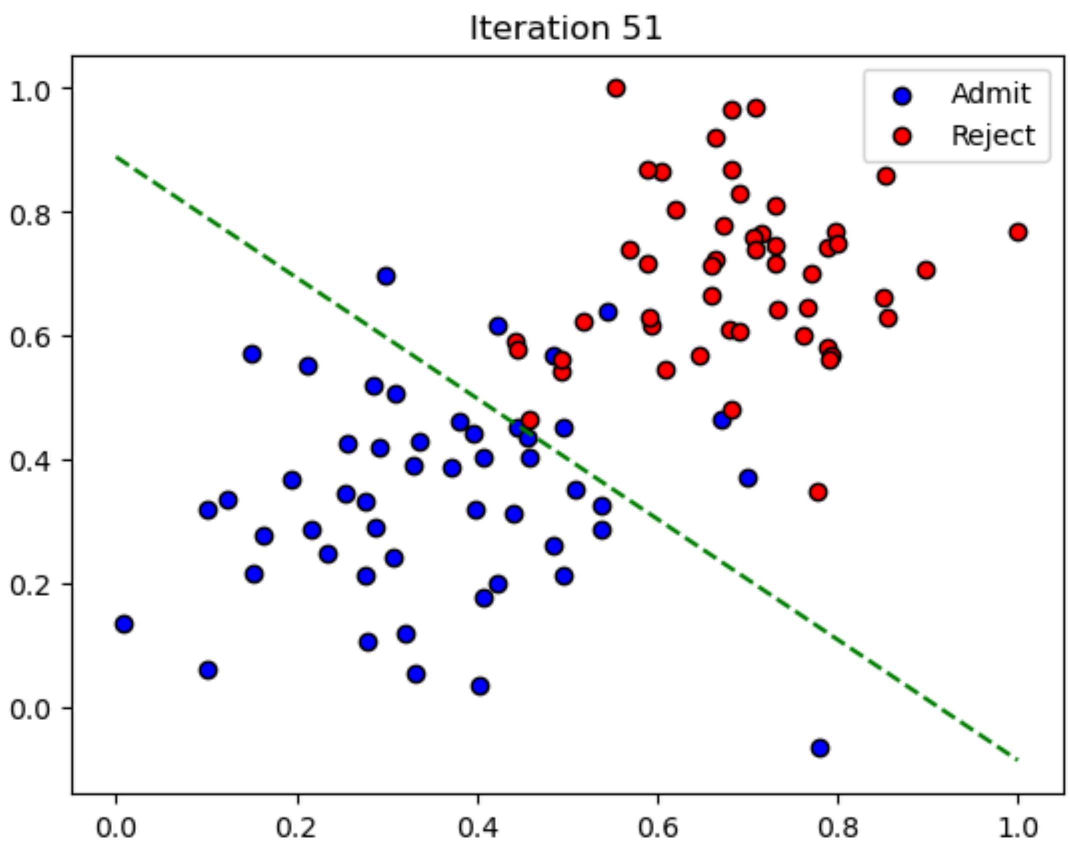

Iteration 1
Admit
Reject
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


Iteration 11
Admit
Reject
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

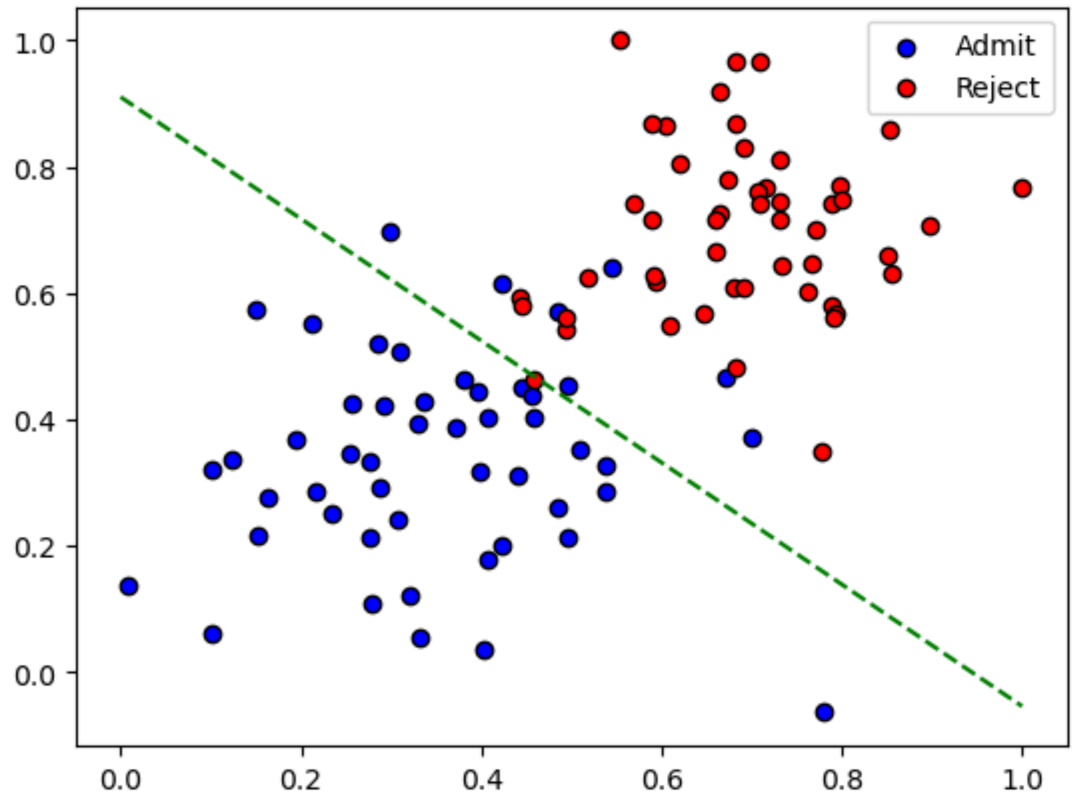

Iteration 21
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0


Iteration 31
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

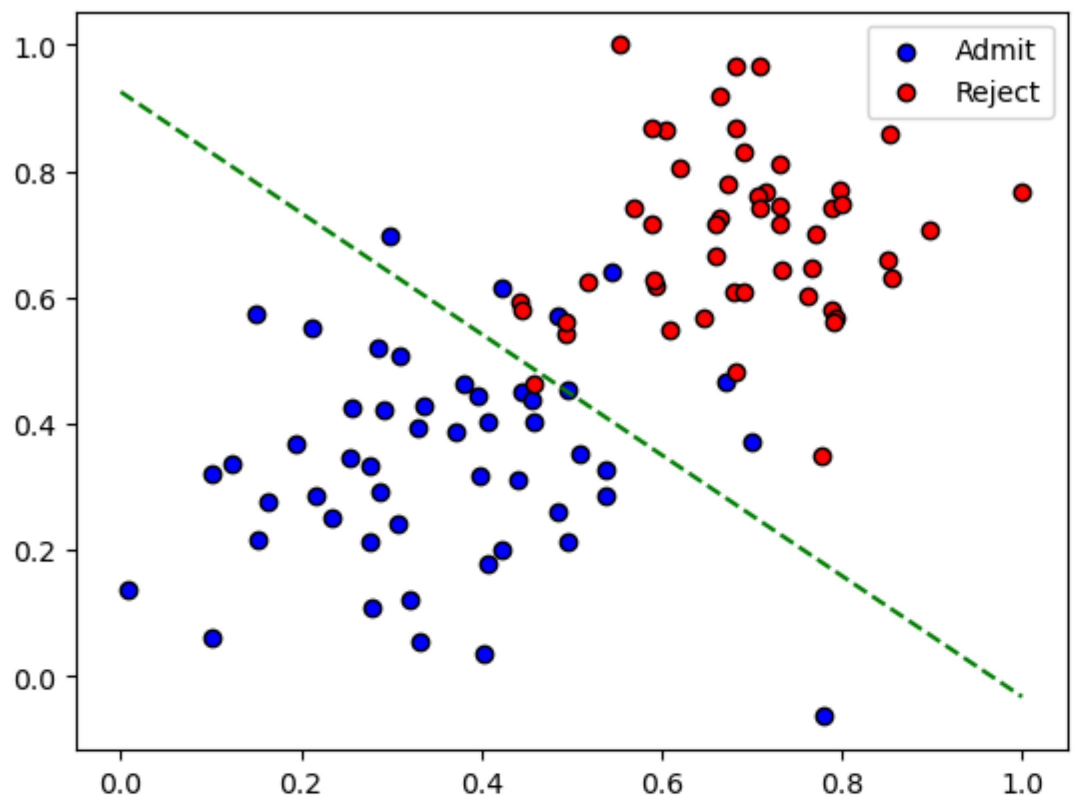

Iteration 41
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

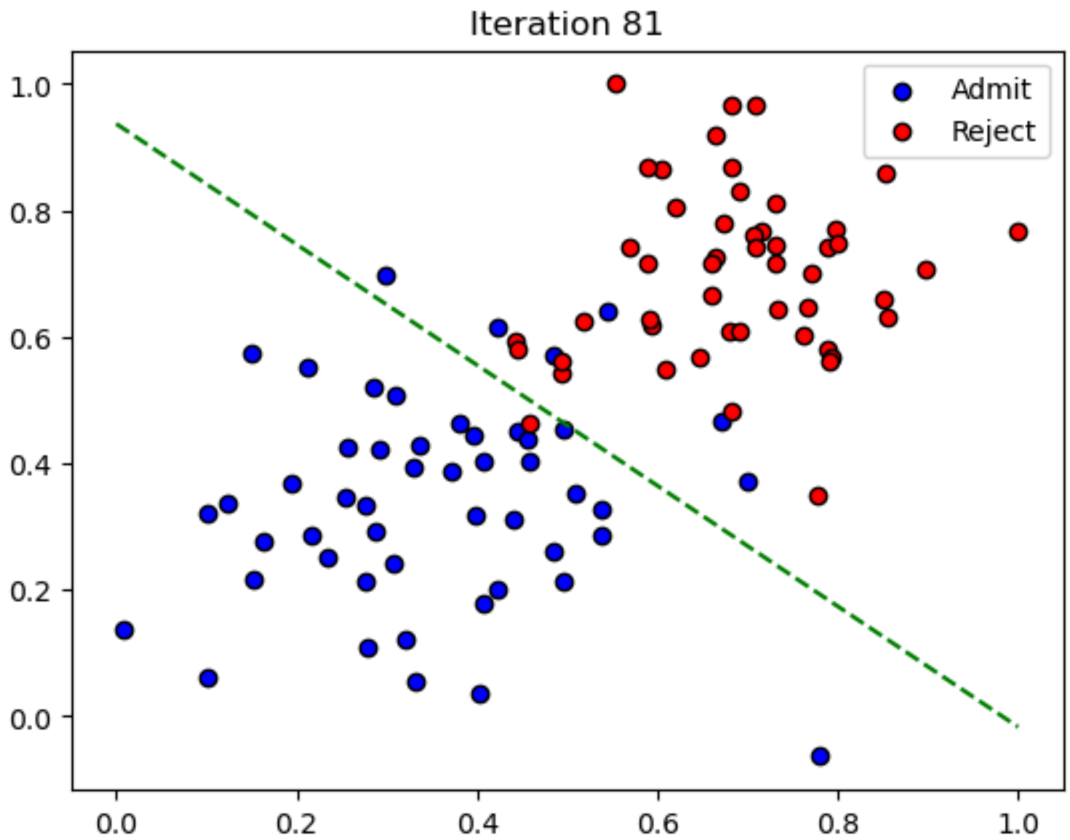

Iteration 51
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

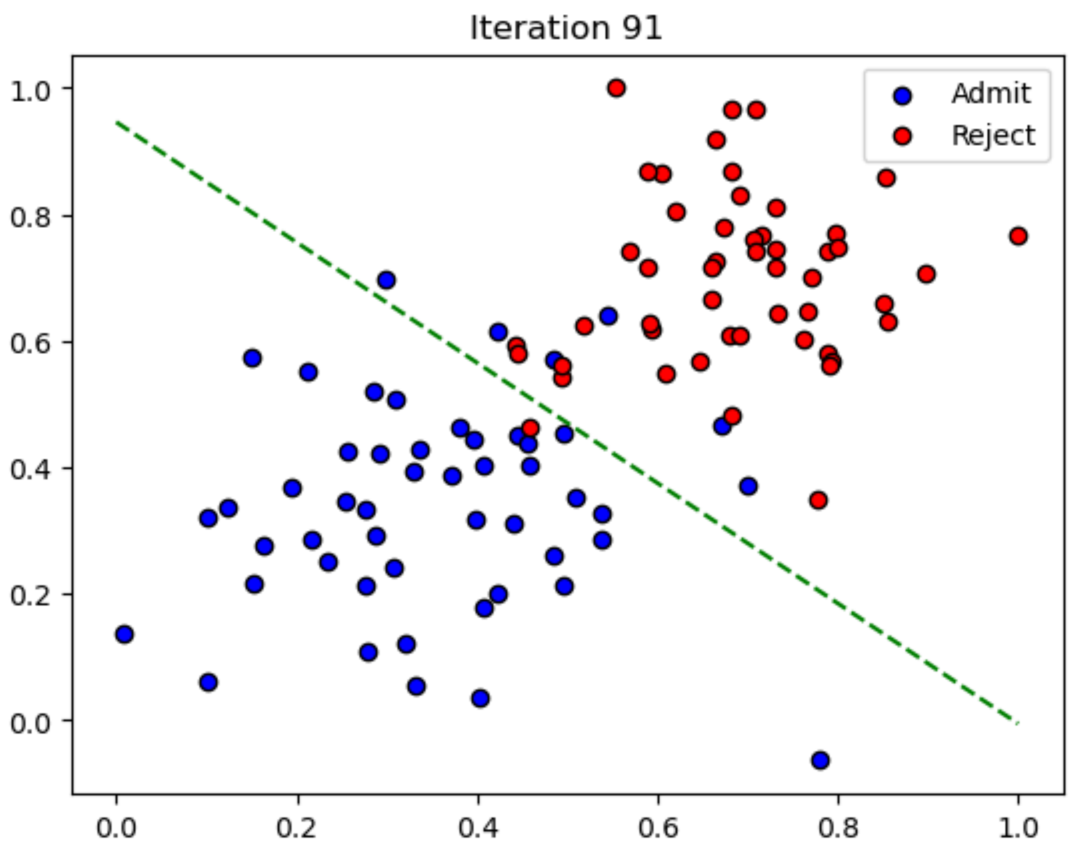
Iteration 61

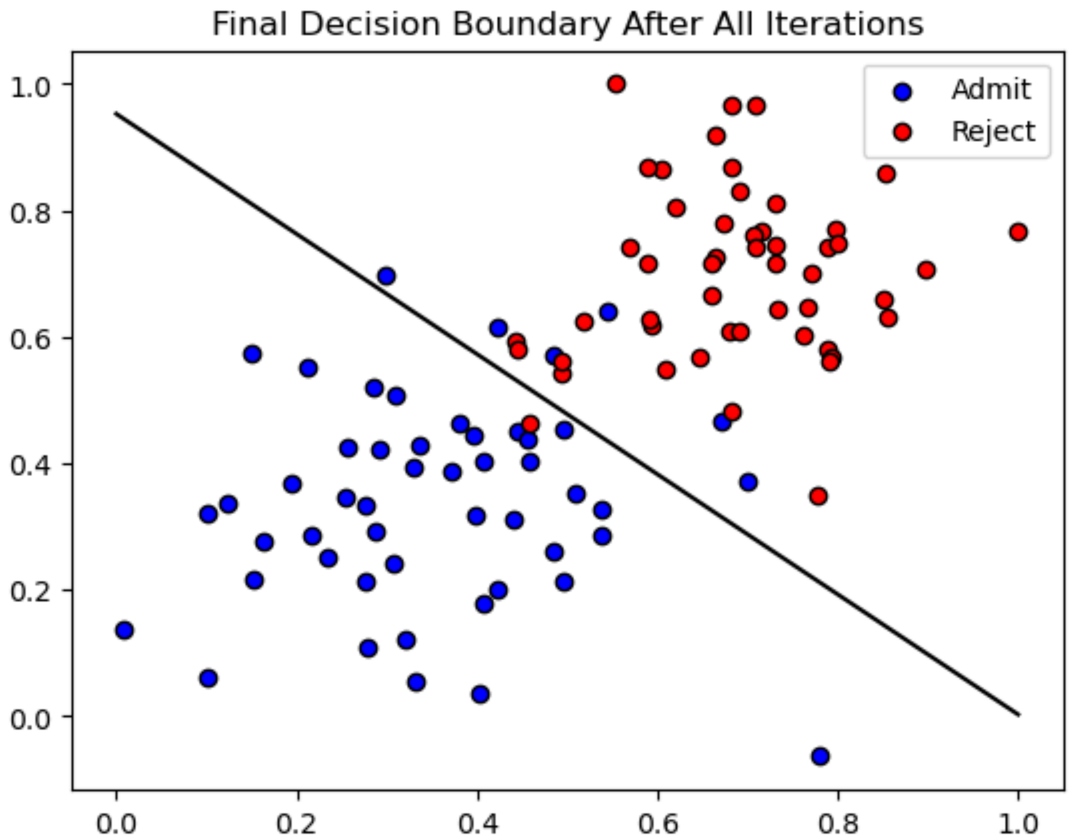


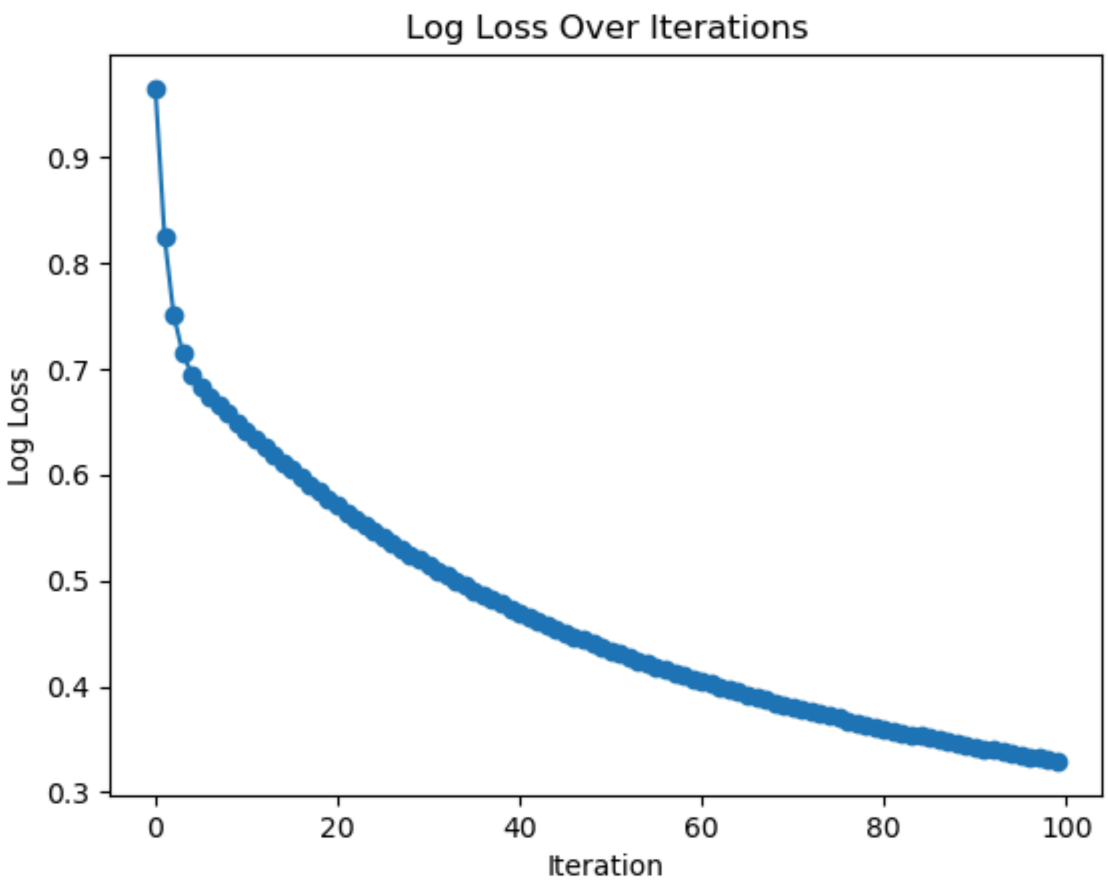
Iteration 71




Iteration 81
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Iteration 91
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Final Decision Boundary After All Iterations
Admit
Reject
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Log Loss Over Iterations
Log Loss
Iteration
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100

In [ ]: