In [96]:
```
import numpy as np
```

Q7

In [97]:
```
# (7a)
# First Function
def perceptron(w, b, x):
  return np.sign(np.dot(w.flatten(), x.flatten()) + b)

def has_converged(w, b, X, l):
  return np.allclose(np.sign(X @ w + b).flatten(), l.flatten())

MAX_ITERS = 72

# Second Function
def train(X, l):
  # data: n x d, labels: n
  n, d = X.shape
  assert(l.shape == (n,))

  w = np.zeros((d, 1))
  b = 0

  num_updates = 0

  for i in range(MAX_ITERS):
    permutation = np.random.choice(range(n), n, replace=False)
    X_perm = X[permutation]
    l_perm = l[permutation]

    for j in range(n):
      x_j = X_perm[[j],:]
      l_j = l_perm[j]
      if perceptron(w, b, x_j) != l_j:
        num_updates += 1
        w = w + l_j * x_j.reshape((-1, 1))
        b += l_j

    if has_converged(w, b, X_perm, l_perm):
      break

  return w, b, num_updates
```

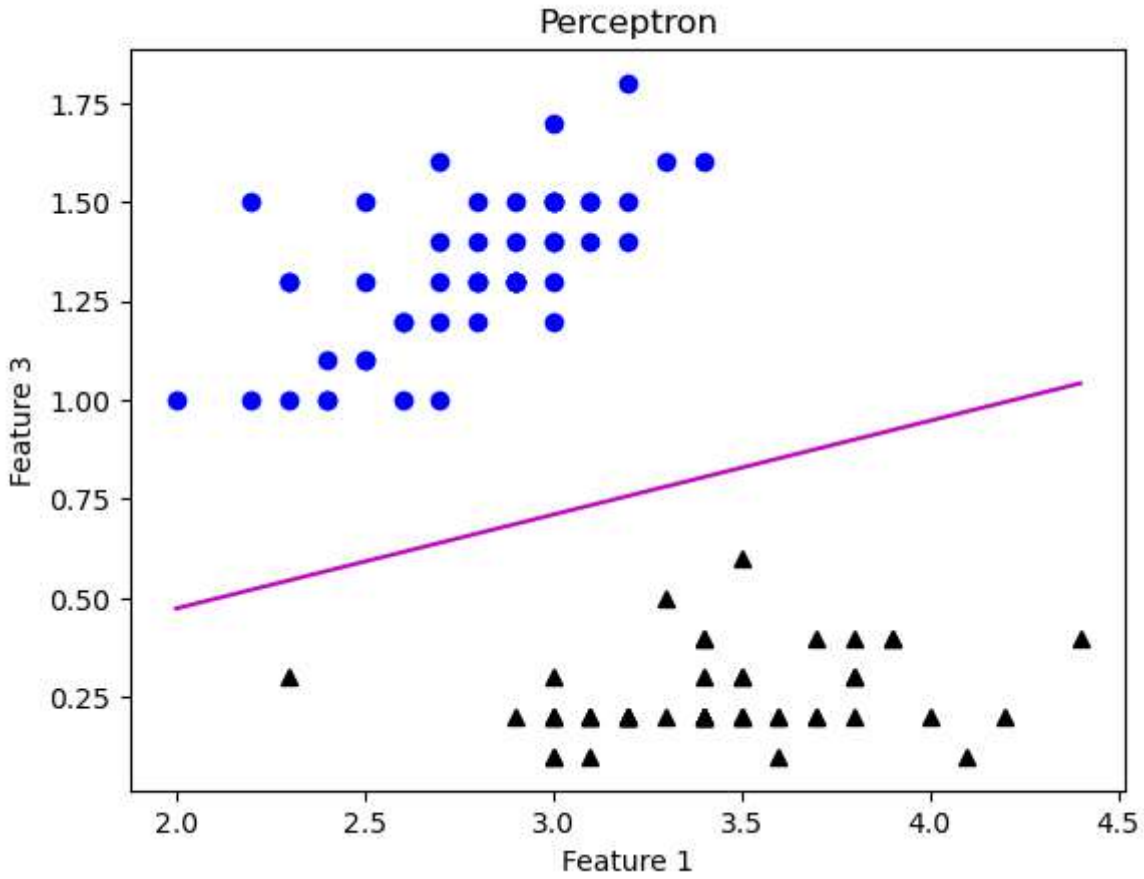
In [98]:
```
# (7b, 7c)

from sklearn import datasets
iris = datasets.load_iris()
X0 = iris.data
y0 = iris.target

print(X0.shape, y0.shape)

rows = np.array(np.nonzero(y0 <= 1)[0])

X_1 = X0[rows.reshape((-1, 1)),(1,3)]
```

```
y_1 = np.where(y0[rows] == 0, -1, 1)

print(X_1.shape, y_1.shape)
```

```
(150, 4) (150,)
(100, 2) (100,)
```

In [99]:

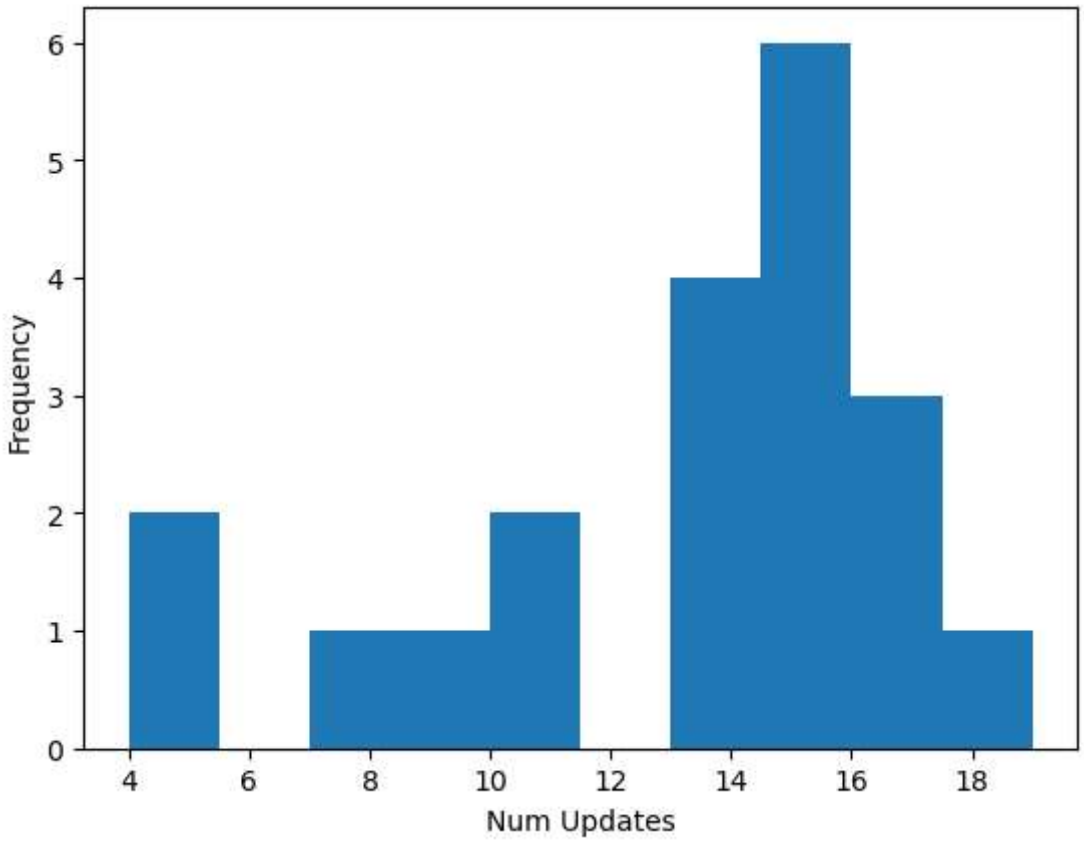
```
import matplotlib.pyplot as plt

# (7c)
np.random.seed(123)
w, b, num_updates = train(X_1, y_1)

def plot(w, b, X, l, draw_line = True, title = "Perceptron"):
  w1, w2 = w.flatten()
  x_min, x_max = np.min(X[:,0]), np.max(X[:,0])
  line_x = np.linspace(x_min, x_max, 100)
  # w1x + w2y + b = 0
  # y = -w1/w2 x - b / w2
  line_y = - w1 / w2 * line_x - b / w2


  plt.plot(X[l==-1,0], X[l==-1,1], "^k")
  plt.plot(X[l==1,0], X[l==1,1], "ob")
  if draw_line:
    plt.plot(line_x, line_y, "-m")
  plt.xlabel("Feature 1")
  plt.ylabel("Feature 3")
  plt.title(title)

plot(w, b, X_1, y_1)
```



In [100…

```
np.random.seed(1337)
n_trials = 20
updates = []
for i in range(n_trials):
  _, _, n_i = train(X_1, y_1)
  updates.append(n_i)
total_updates = sum(updates)
print(f"Average number of updates: {total_updates}/{n_trials}={total_updates / n_trials
plt.hist(updates)
plt.xlabel("Num Updates")
plt.ylabel("Frequency");
```

```
Average number of updates: 258/20=12.9
```

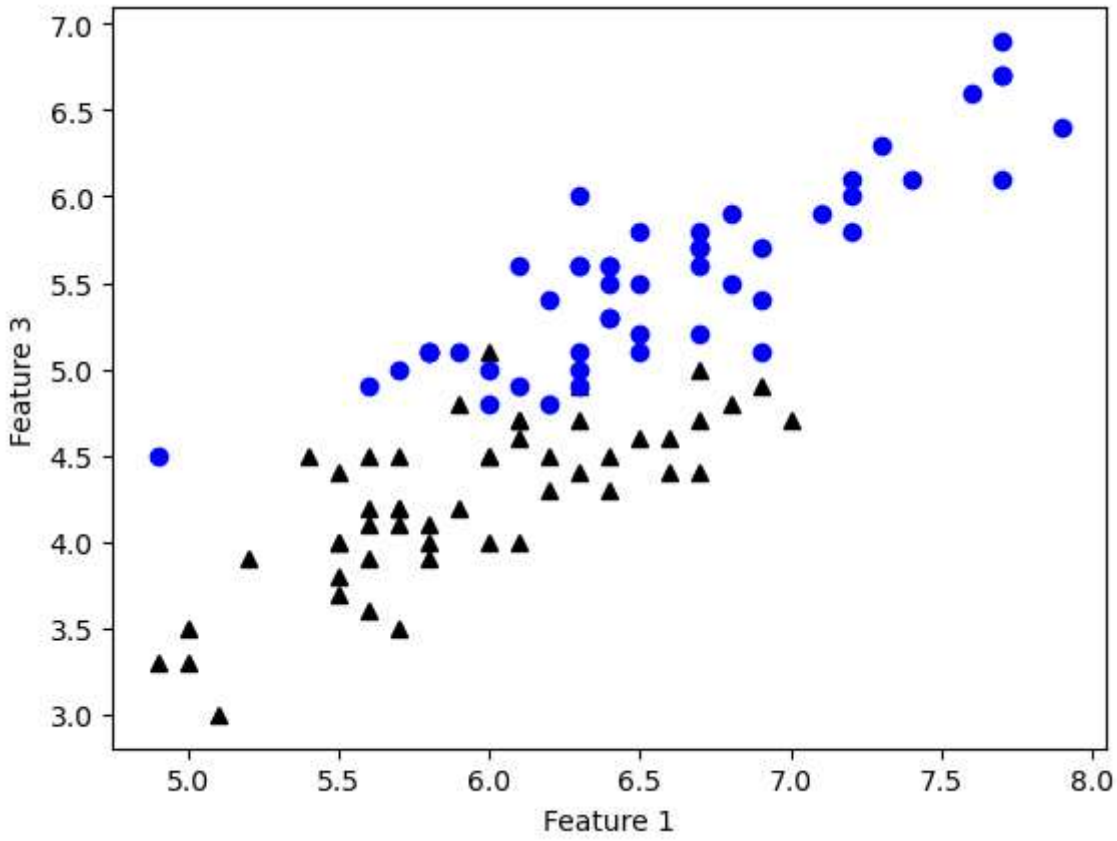


Q8

In [101…

```
rows_2 = np.array(np.nonzero(y0 > 0)[0])
X_2 = X0[rows_2.reshape((-1, 1)),(0, 2)]
y_2 = np.where(y0[rows_2] == 1, -1, 1)

print(X0.shape, y0.shape)
print(X_2.shape, y_2.shape)

plot(np.ones((2, 1)), 1, X_2, y_2, draw_line = False, title = None)

print("(8a) The data is NOT linearly seperable")
```

```
(150, 4) (150,)
(100, 2) (100,)
(8a) The data is NOT linearly seperable
```



In [102…

```
np.c_[np.array([1,2,3]), np.array([4,5,6])]
```

Out[102…

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

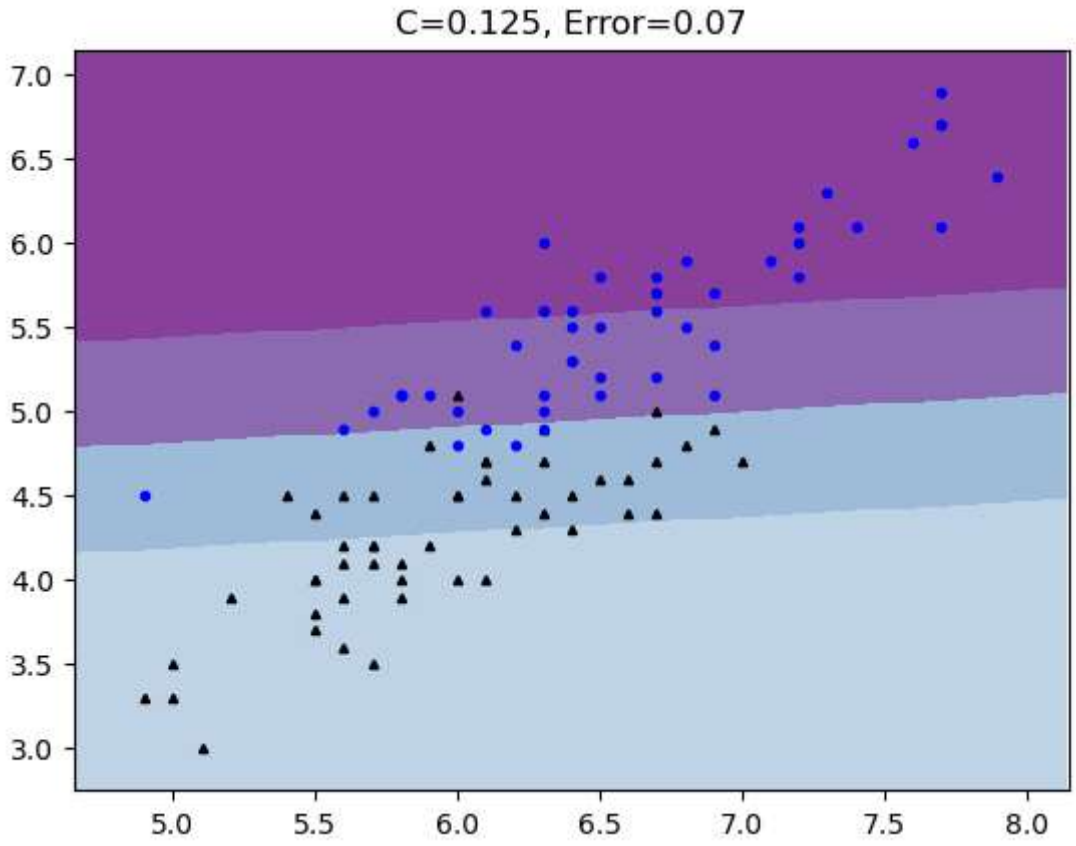
In [103…

```
import sklearn
def full_plot(X, y, svc: sklearn.svm.SVC, C, error):
  plt.plot(X[y==-1,0], X[y==-1,1], "^k", markersize=3)
  plt.plot(X[y==1,0], X[y==1,1], "ob", markersize=3)
  margin = 0.25
  x_min, x_max = np.min(X[:,0]), np.max(X[:,0])
  y_min, y_max = np.min(X[:,1]), np.max(X[:,1])
  delta = 0.01

  xx, yy = np.meshgrid(np.arange(x_min - margin, x_max + margin, delta), np.arange(y_mi
  Z = svc.decision_function(np.c_[xx.flatten(), yy.flatten()])
  for i in range(len(Z)):
    Z[i] = min(Z[i],1.0)
    Z[i] = max(Z[i],-1.0)
    if (Z[i] > 0.0) and (Z[i] < 1.0):
        Z[i] = 0.5
    if (Z[i] < 0.0) and (Z[i] > -1.0):
        Z[i] = -0.5
  Z = Z.reshape(xx.shape)
  plt.pcolormesh(xx, yy, Z, cmap=plt.cm.BuPu, vmin=-2, vmax=2)
  plt.xlim((x_min - margin, x_max + margin))
  plt.ylim((y_min - margin, y_max + margin))
  plt.title(f"C={C}, Error={error}")
  plt.show()
```

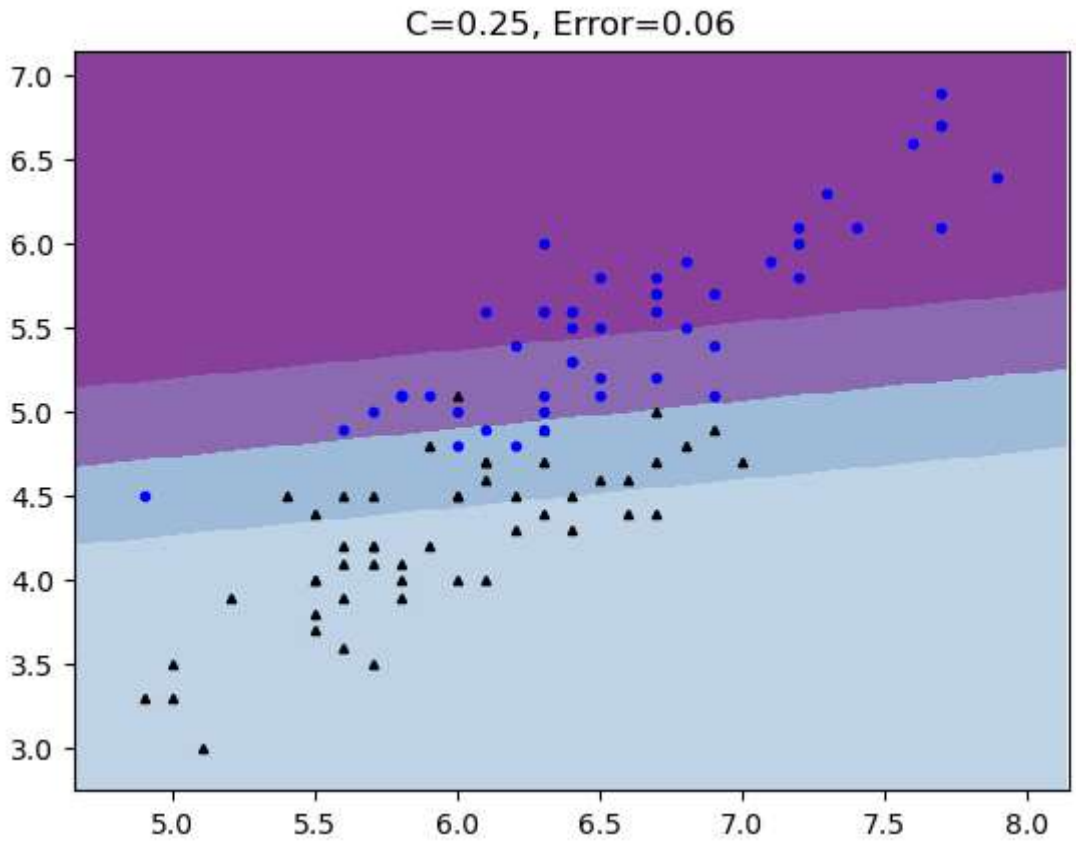
In [104…

```
import sklearn

Cs = np.exp2(np.arange(-3, 10))
data = []
for c in Cs:
  print(f"=== {c} ===")
  svm = sklearn.svm.SVC(kernel="linear", C=c)
  svc = svm.fit(X=X_2, y=y_2)

  wrong = np.mean(np.not_equal(svm.predict(X_2), y_2))
  print(f"\tError rate: {wrong}")
  full_plot(X_2, y_2, svc, c, wrong)
  num_sv = len(svc.support_)

  data.append((c, wrong, num_sv))
```

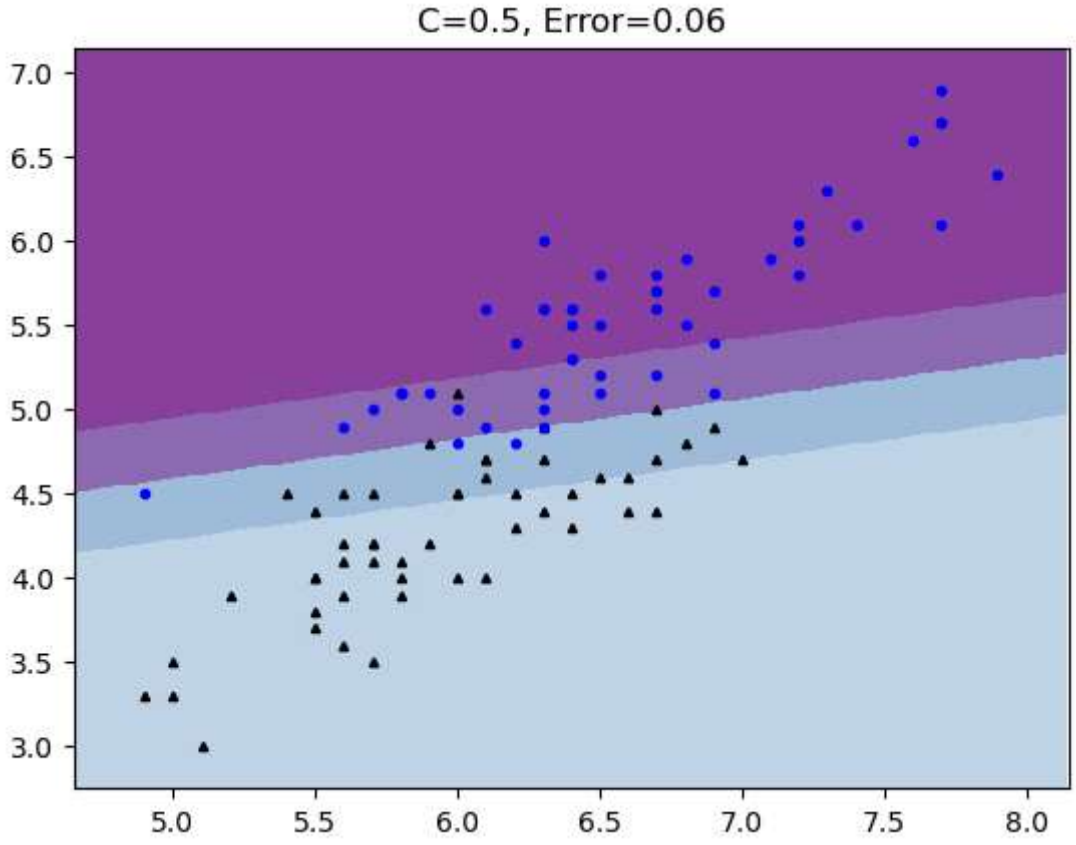
```
=== 0.125 ===
        Error rate: 0.07
```



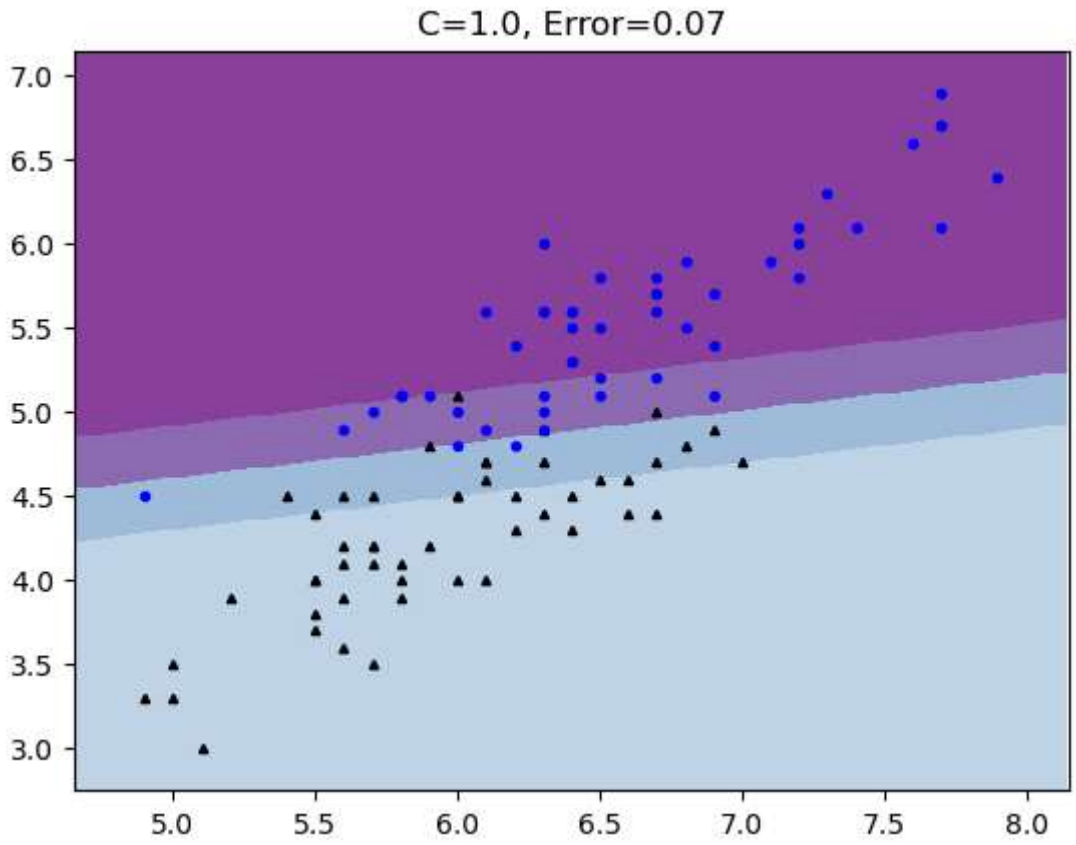

```
=== 0.25 ===
        Error rate: 0.06
```



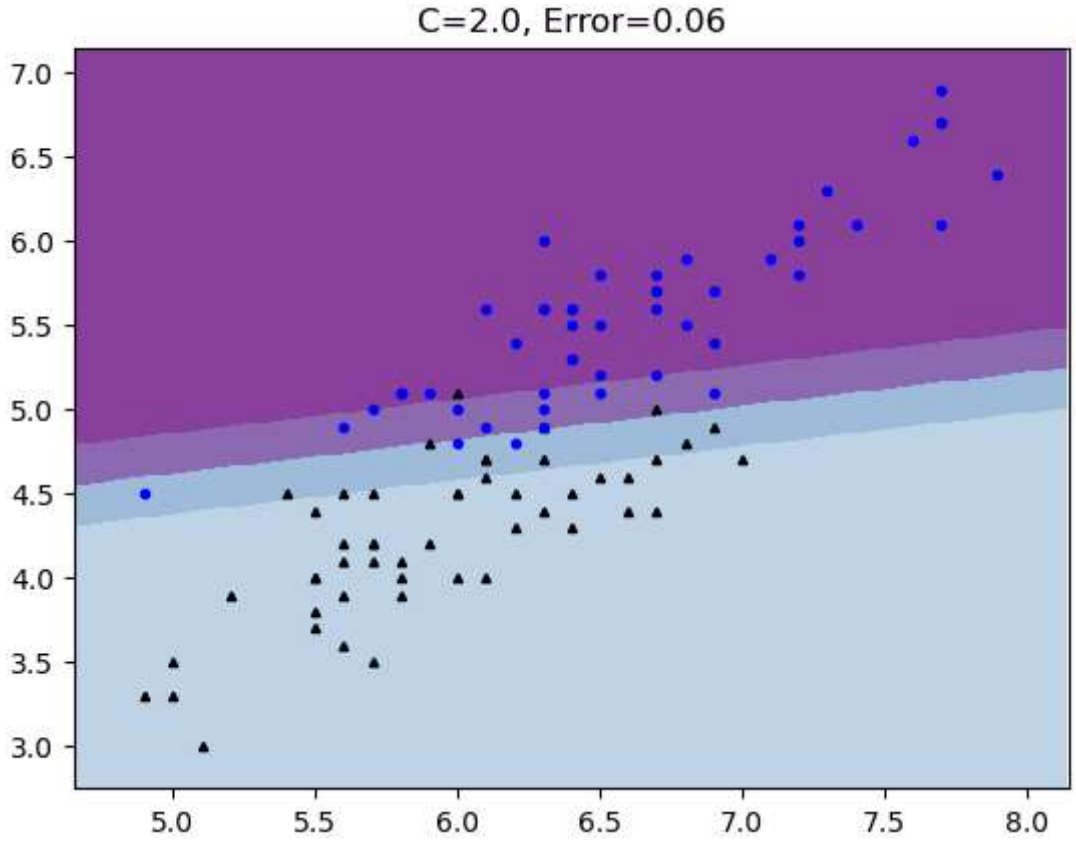

```
=== 0.5 ===
        Error rate: 0.06
```



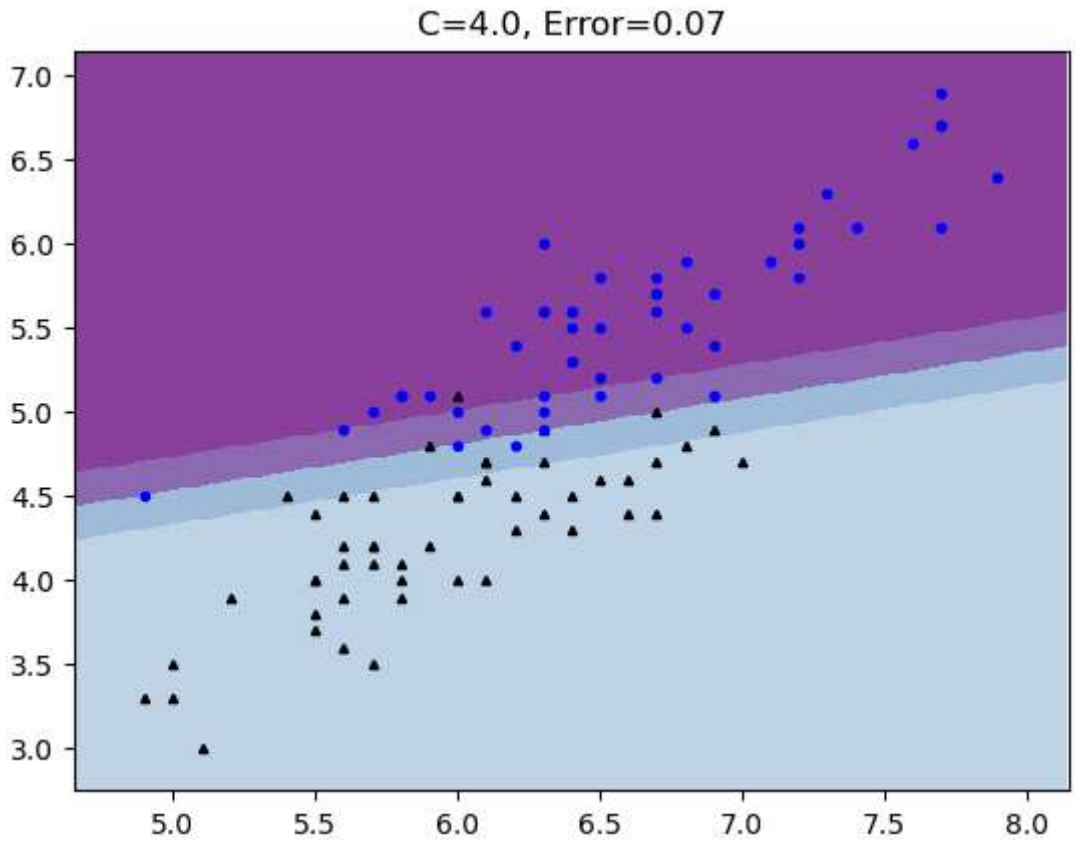

```
=== 1.0 ===
        Error rate: 0.07
```



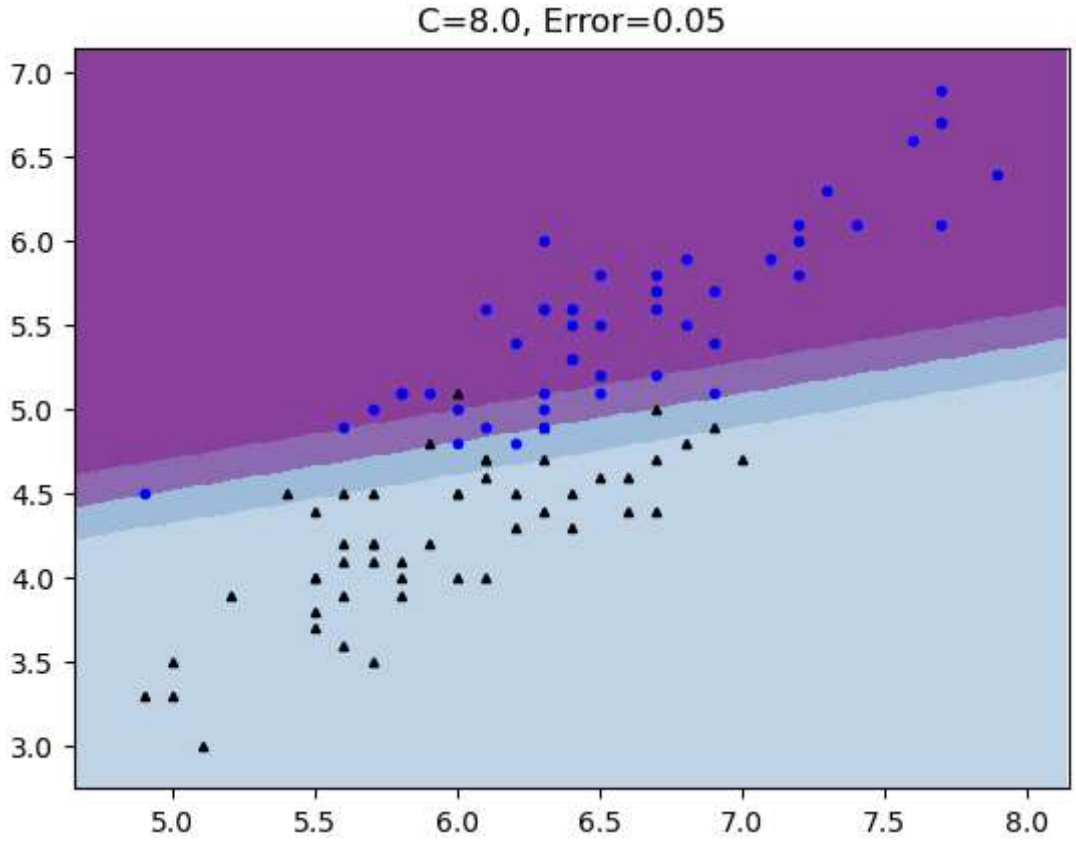

```
=== 2.0 ===
        Error rate: 0.06
```



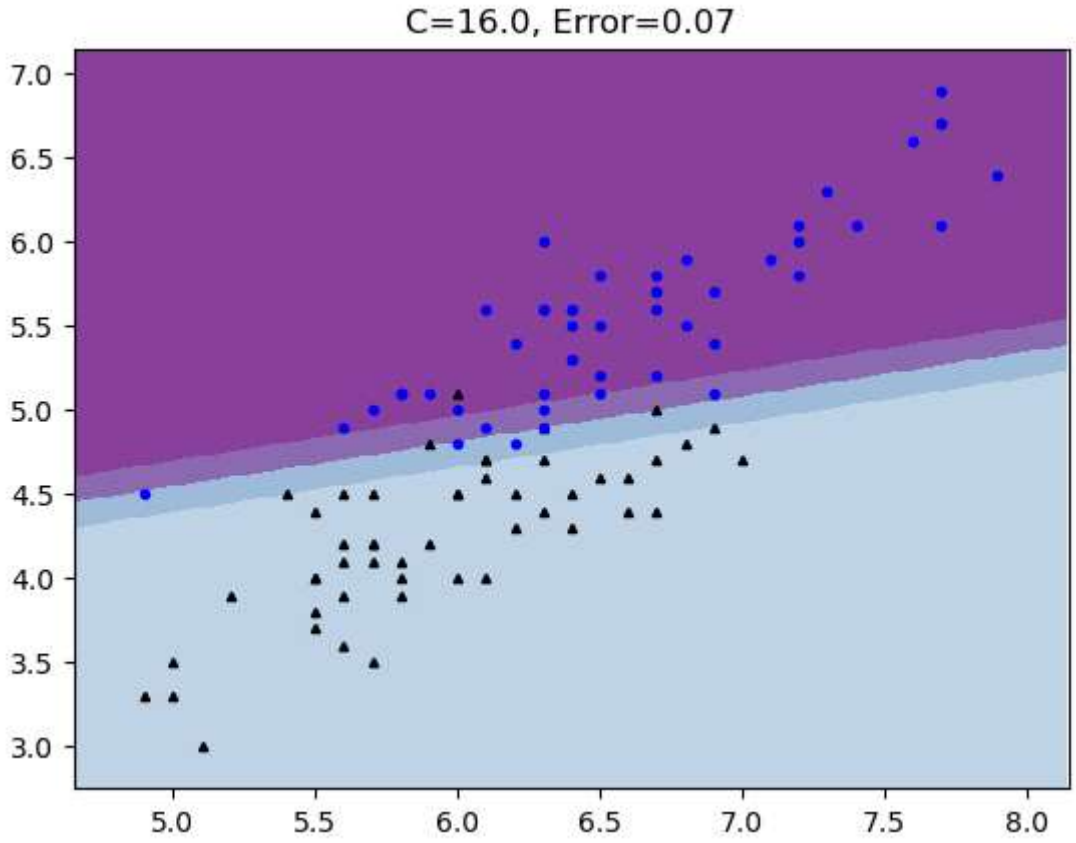

```
=== 4.0 ===
        Error rate: 0.07
```



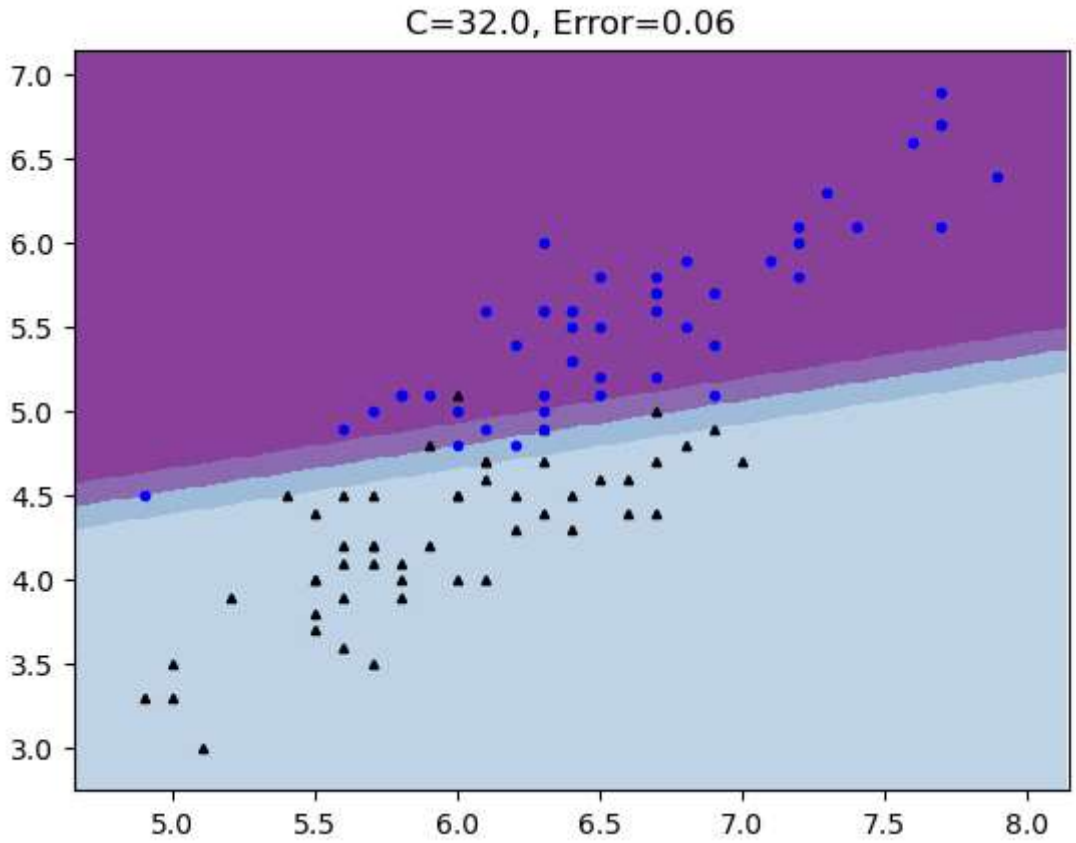

```
=== 8.0 ===
        Error rate: 0.05
```



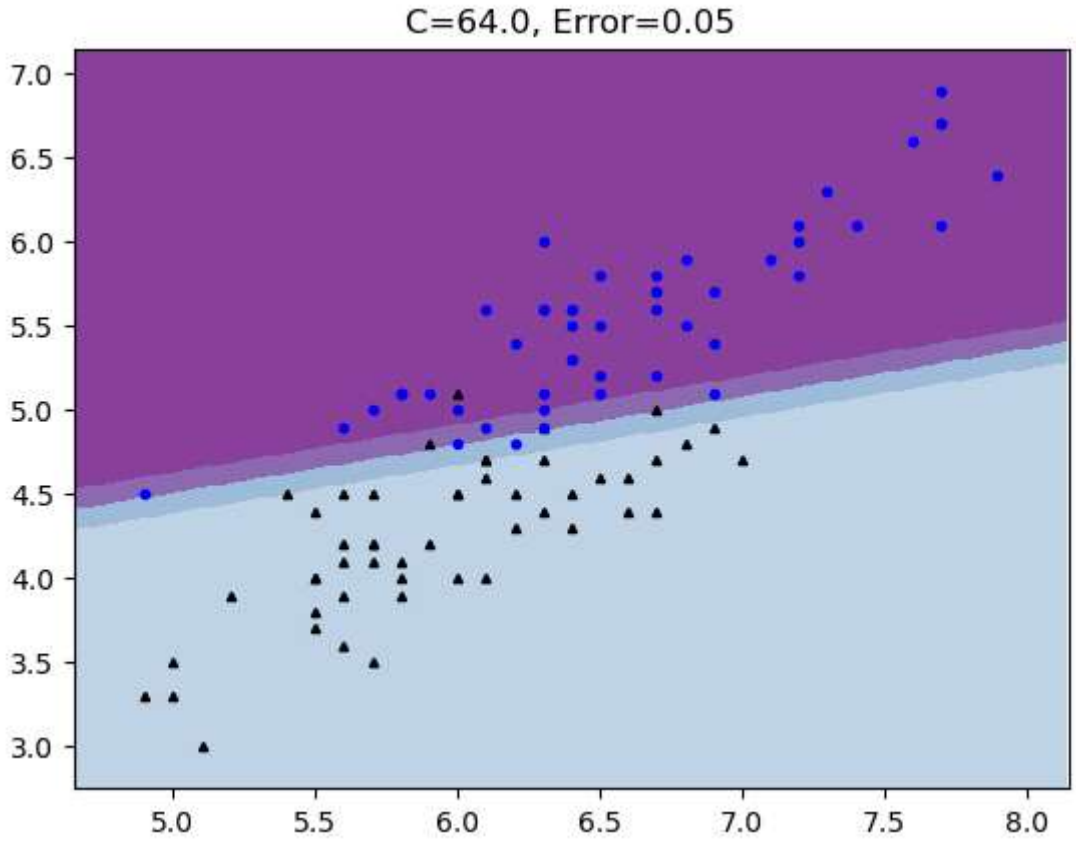

```
=== 16.0 ===
        Error rate: 0.07
```



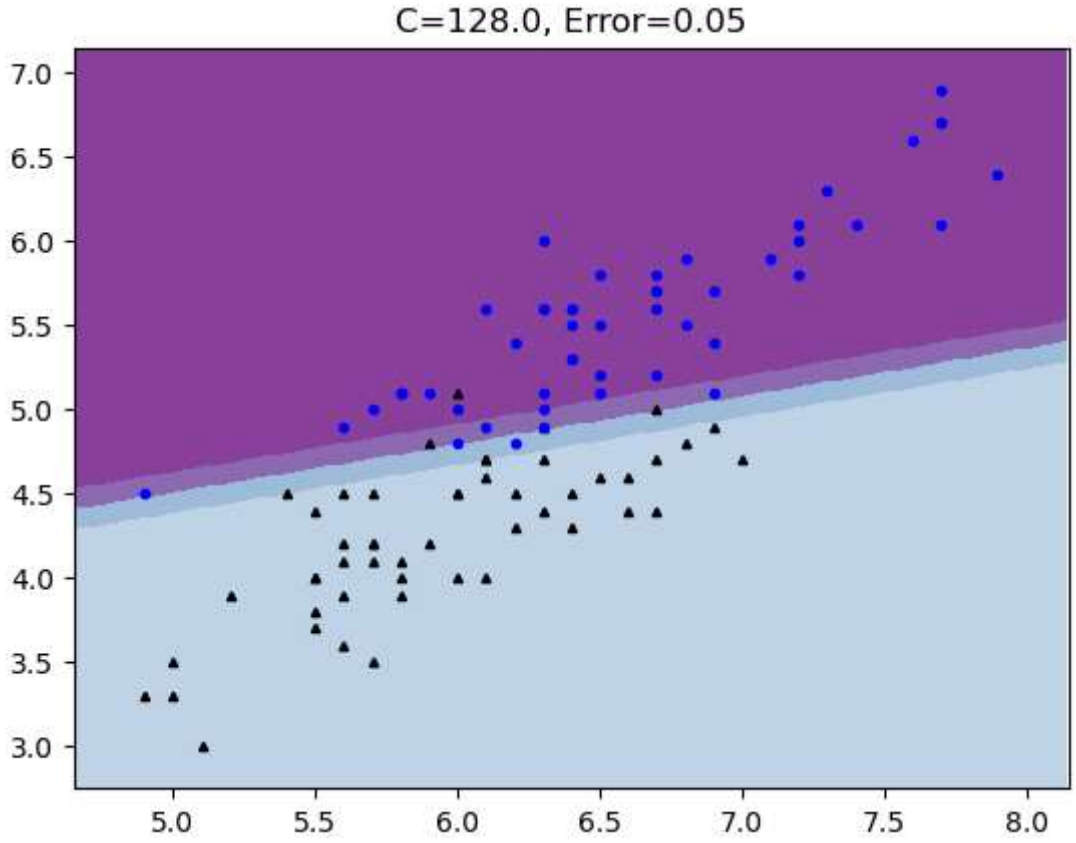

```
=== 32.0 ===
        Error rate: 0.06
```



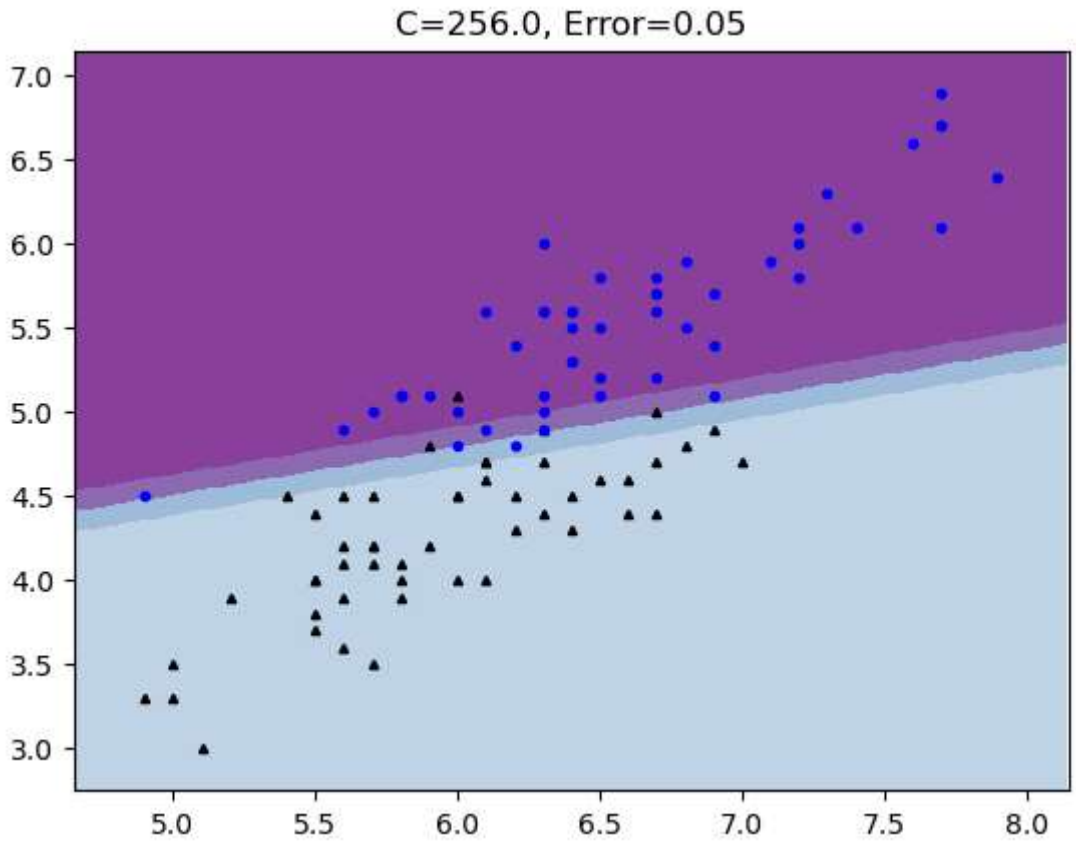

```
=== 64.0 ===
        Error rate: 0.05
```



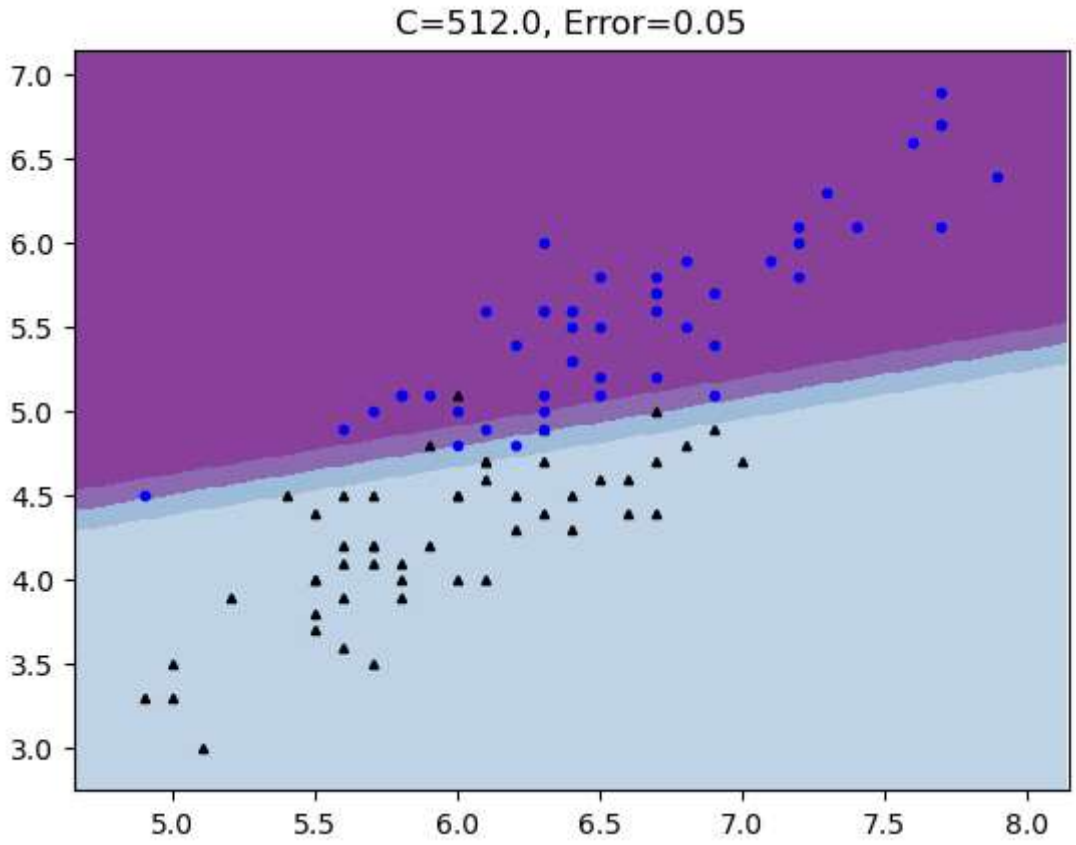

```
=== 128.0 ===
        Error rate: 0.05
```




```
=== 256.0 ===
        Error rate: 0.05
```




```
=== 512.0 ===
        Error rate: 0.05
```



In [105…

```
for v in data:
  print(str(v)[1:-1].replace(',', ' &') + "\\\\\\hline")
```

```
0.125 & 0.07 & 52\\\hline
0.25 & 0.06 & 45\\\hline
0.5 & 0.06 & 38\\\hline
1.0 & 0.07 & 31\\\hline
2.0 & 0.06 & 24\\\hline
4.0 & 0.07 & 21\\\hline
8.0 & 0.05 & 19\\\hline
16.0 & 0.07 & 16\\\hline
32.0 & 0.06 & 15\\\hline
64.0 & 0.05 & 14\\\hline
128.0 & 0.05 & 14\\\hline
256.0 & 0.05 & 14\\\hline
512.0 & 0.05 & 14\\\hline
```