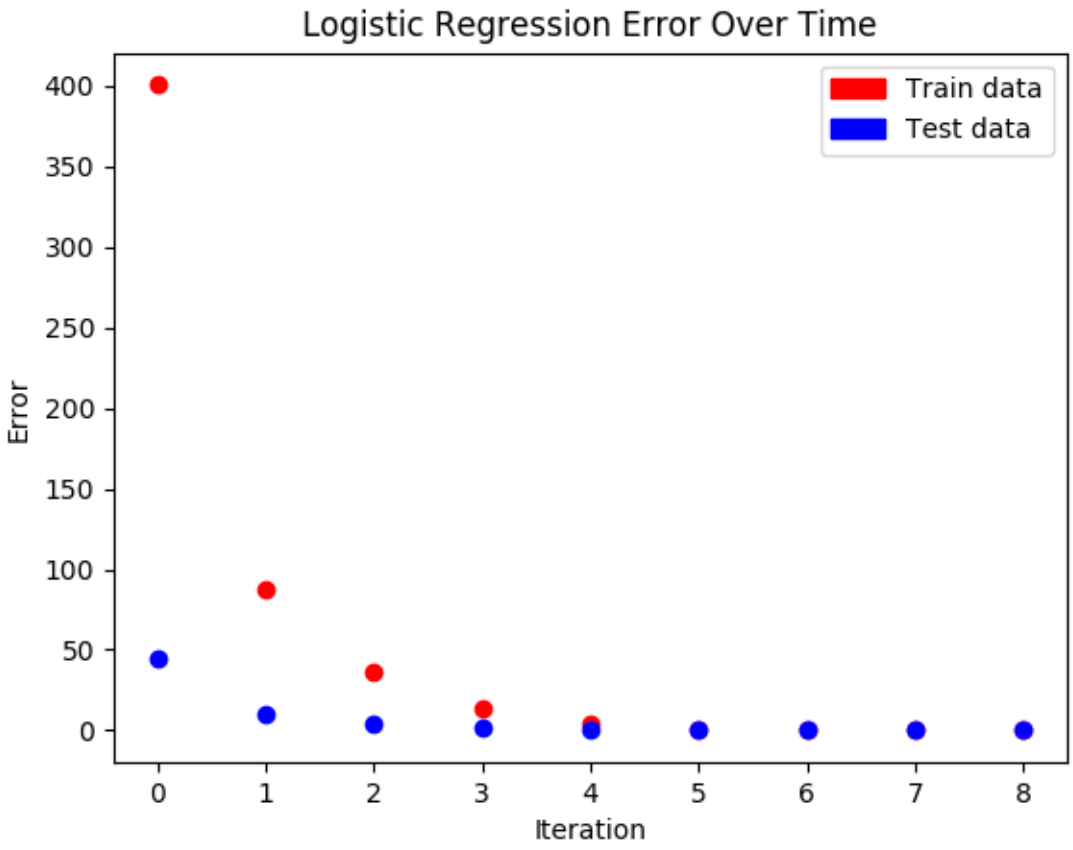
With the aid of Newton's Method, we perform logistic regression on a labeled dataset and plot our classifer error as a function of iterations. The algorithm converges after eight steps. Our code and plot is below:

```
import numpy
from numpy import *
from scipy.special import expit as expit
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

err = 10**8
epsilon=10**-8
w = array([0.0,0.0,0.0])

train_features_load = array(loadtxt('train_features.dat',unpack=True))
train_labels = array(loadtxt('train_labels.dat',unpack=True))
test_features_load = array(loadtxt('test_features.dat',unpack=True))
test_labels = array(loadtxt('test_labels.dat',unpack=True))

train_features = numpy.concatenate((numpy.ones((1,900),dtype=numpy.float64), train_features_load),
axis = 0)
test_features = numpy.concatenate((numpy.ones((1,100),dtype=numpy.float64), test_features_load),
axis = 0)

def sigmoid(xval,wval):
 z= xval @ wval
 return expit(-z)

def getlvalTrain(wval):
 lval = 0.00000000
 for i in range(train_features.shape[1]):
  lsum = -train_labels[i]*numpy.log(sigmoid((train_features[:,i]),wval))-(1-
train_labels[i])*numpy.log(1-sigmoid((train_features[:,i]),wval))
  lval = lval + lsum
 return lval

def getlvalTest(wval):
 lval = 0.00000000
 for i in range(test_features.shape[1]):
  lsum = -test_labels[i]*numpy.log(sigmoid((test_features[:,i]),wval))-(1-test_labels[i])*numpy.log(1-
sigmoid((test_features[:,i]),wval))
  lval = lval + lsum
 return lval

def getHval(wval):
 Hval = numpy.zeros((3,3),dtype=numpy.float64)
 for i in range(train_features.shape[1]):
```

```
  Hsum = numpy.outer((train_features[:,i]),(train_features[:,i]))*sigmoid((train_features[:,i]),wval)*(1-
sigmoid((train_features[:,i]),wval))
  Hval = Hval + Hsum
 return Hval

def getGradval(wval):
 Gradval = numpy.zeros((3),dtype=numpy.float64)
 for i in range(train_features.shape[1]):
  Gradsum = (train_features[:,i])*(sigmoid((train_features[:,i]),wval)-(train_labels[i]))
  Gradval = Gradval + Gradsum
 return Gradval

trainErrs = []
testErrs = []

steps = 0
while abs(err) > epsilon:
 steps = steps + 1
 wOld = w
 lvalOldTrain = getlvalTrain(wOld)
 lvalOldTest = getlvalTest(wOld)
 GradvalOld = getGradval(wOld)
 HvalOld = getHval(wOld)
 Hessinv = numpy.linalg.inv(HvalOld)
 w = w + Hessinv @ GradvalOld
 print(w)
 err = abs(lvalOldTrain - getlvalTrain(w))
 errTest = abs(lvalOldTest - getlvalTest(w))
 print('errTrain is')
 print(err)
 trainErrs.append(err)
 print('errTest is')
 print(errTest)
 testErrs.append(errTest)

plt.plot(range(steps),trainErrs,'ro', range(steps),testErrs,'bo')
plt.title('Logistic Regression Error Over Time')
red_patch = mpatches.Patch(color='red', label='Train data')
blue_patch = mpatches.Patch(color='blue', label='Test data')
plt.legend(handles=[red_patch,blue_patch])

plt.xlabel('Iteration')
plt.ylabel('Error')
plt.show()
```


Logistic Regression Error Over Time
Train data
Test data
Error
Iteration
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8