In [1]:
```
#Name: Jose Luis Vargas
#Jan 18, 2017
#CS 178 HW1

import numpy as np
import matplotlib.pyplot as plt


iris = np.genfromtxt("data/iris.txt", delimiter = None) #reads the text file
Y = iris[:,-1] # These are the target values. (5th column)
X = iris[:,0:-1] # 4 Features. (1st-4th columns)
```

Problem 1A

In [2]:
```
X.shape[1] #Number of features
```

Out[2]: 4

In [3]:
```
X.shape[0] #Number of data points
```

Out[3]: 148

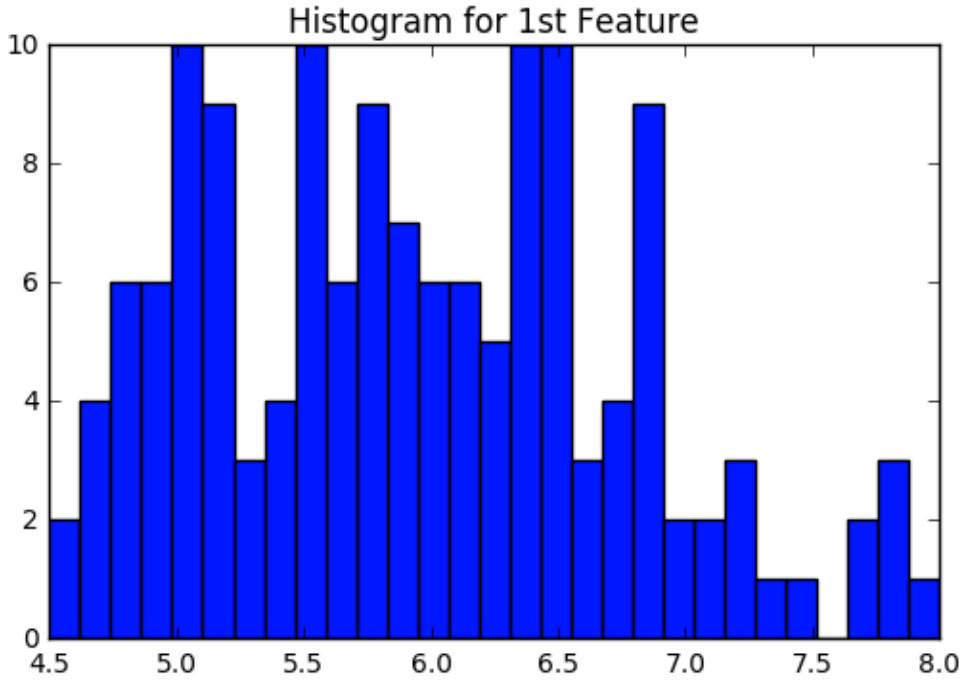
Problem 1B

In [4]:
```
print X[:,0] #Here are all the values of the 1st feature.(1st column)
#This is to analyze the data and to check its min and max values.
```

```
[ 4.9400593  4.7738176  4.620137   5.0774442  5.415624   4.6451451
  5.011375   4.4814082  4.917678   5.4582963  4.8244597  4.846128
  4.3265579  5.8682189  5.7420976  5.4832527  5.1810909  5.7146538
  5.1421791  5.4779664  5.1612603  4.6495985  5.1535719  4.8142352
  5.0066373  5.032968   5.2777809  5.2672442  4.716552   4.8091428
  5.4857799  5.2612812  5.5798048  4.9216602  5.0538592  5.5584716
  4.9465036  4.424549   5.1574214  5.0077006  4.5224878  4.4784896
  5.0944268  5.1931252  4.8349743  5.1889721  4.6811691  5.3654886
  5.0122011  7.0702042  6.4737797  6.9029101  5.559616   6.5500998
  5.739643   6.3992772  4.9283717  6.6055467  5.2152     5.0467768
  5.974119   6.0649203  6.1726794  5.6516173  6.7767935  5.6956664
  5.8284416  6.2411357  5.6720155  5.9181775  6.1460971  6.3699328
  6.1054686  6.4912877  6.6501879  6.8689603  6.7079651  6.0445045
  5.736273   5.5670107  5.508528   5.8723741  6.0548241  5.4349585
  6.0013442  6.7977225  6.332859   5.6520445  5.542998   5.5724534
  6.1793583  5.8764628  5.0243674  5.625148   5.7431951  5.7674293
  6.2679348  5.1187597  5.7311962  6.3644543  5.8648435  7.1474052
  6.3204357  6.5800335  7.6814694  4.9162941  7.3648843  6.759175
  7.2097041  6.5462131  6.407938   6.8588441  5.724238   5.8190088
  6.4996298  6.5281543  7.777258   7.7884463  6.082629   6.9338369
  5.6655144  7.7150636  6.3479656  6.795869   7.2004658  6.2933944
  6.1699019  6.4982575  7.2263808  7.4353972  7.9528335  6.4795806
  6.3434387  6.1938172  7.7829518  6.3857928  6.4916516  6.0621873
  6.9462665  6.7812699  6.9112125  5.8831389  6.8283671  6.7942622
  6.7997386  6.3603635  6.5514822  6.2603127]
```

In [5]:

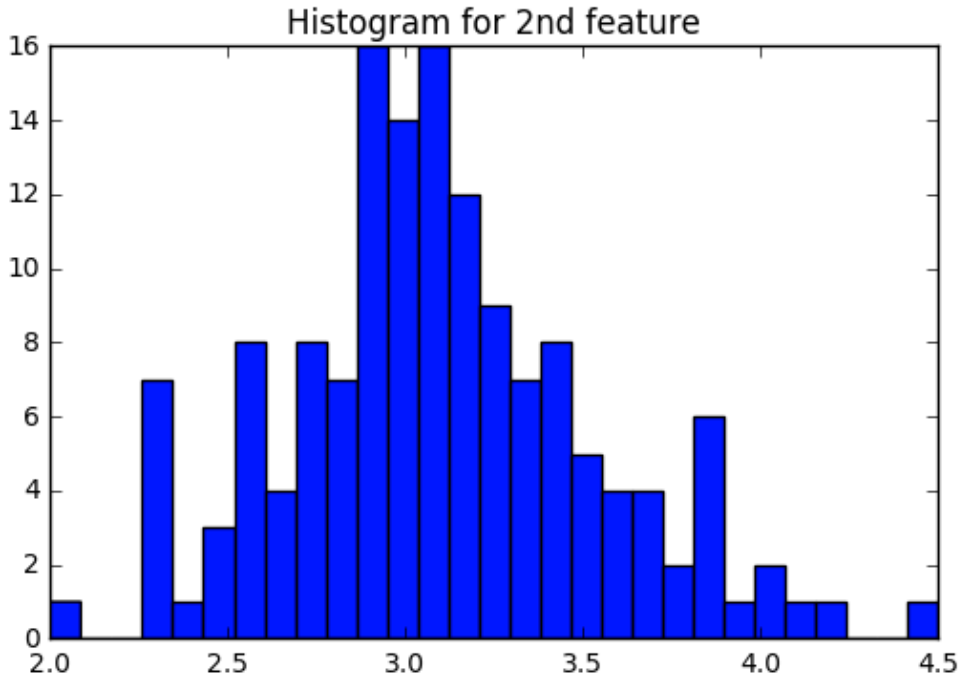
```
f1 = X[:,0] #Gets the values of the first column(1st feature)
Bins = np.linspace(4.5,8,30) #The Best way to graph the data is to start at 4
                             #because there is no value smaller than 4 and greater than 8
plt.hist(f1, bins = Bins)
plt.title("Histogram for 1st Feature")
plt.show()
```



In [6]:
```
print X[:,1] #values for the 2nd feature. The values are not less than 1 and not greater than 5.
```

```
[ 3.081924   3.2812362  3.1387625  3.6684541  3.9942807  3.4706607
  3.4491627  2.9916573  3.1954035  3.7263543  3.462181   3.0482423
  3.0098618  4.0119708  4.458073   3.9533324  3.5847085  3.8543564
  3.870928   3.4583399  3.752496   3.6959479  3.3008335  3.4620354
  3.0245998  3.4930898  3.5954967  3.4978359  3.2434848  3.1185676
  3.4897792  4.1047388  4.2166009  3.1111247  3.2026691  3.5510408
  3.1762573  3.0514512  3.4231124  3.5585824  2.3244504  3.2489006
  3.5621951  3.8242269  3.0568048  3.8760472  3.2279905  3.7153595
  3.3158748  3.2568599  3.2295192  3.1513333  2.3432164  2.862918
  2.8052828  3.3512431  2.4325255  2.9007846  2.7252409  2.0020454
  3.0042396  2.2880185  2.9259247  2.9548243  3.1591117  3.0588435
  2.7398879  2.2737921  2.5604946  3.204523   2.8128232  2.5476395
  2.834675   2.9333136  3.01558    2.8955501  3.07274    2.9424033
  2.6637372  2.4244231  2.4833235  2.7002929  2.7431544  3.0096323
  3.4257753  3.1113063  2.3169067  3.0778244  2.5553458  2.6012641
  3.0596351  2.6798051  2.3207136  2.7560392  3.0968838  2.9247291
  2.9460072  2.5699889  2.8936035  3.3089424  2.7386875  3.0886295
  2.9185437  3.0977692  3.0660154  2.5107312  2.9309605  2.518707
  3.6379432  3.2661319  2.7800405  3.059977   2.5272322  2.8667918
  3.2525588  3.0100596  3.8687924  2.6172729  2.2777502  3.2000016
  2.8548709  2.8910809  2.7361103  3.3222764  3.2761173  2.8924806
  3.033324   2.89205    3.002125   2.8024043  3.8910375  2.8164015
  2.8831155  2.6800039  3.0143588  3.4555813  3.174443   3.0340987
  3.1728895  3.1332403  3.1404237  2.7396583  3.294404   3.3402902
  3.028202   2.5873023  3.0142411  3.4184766]
```

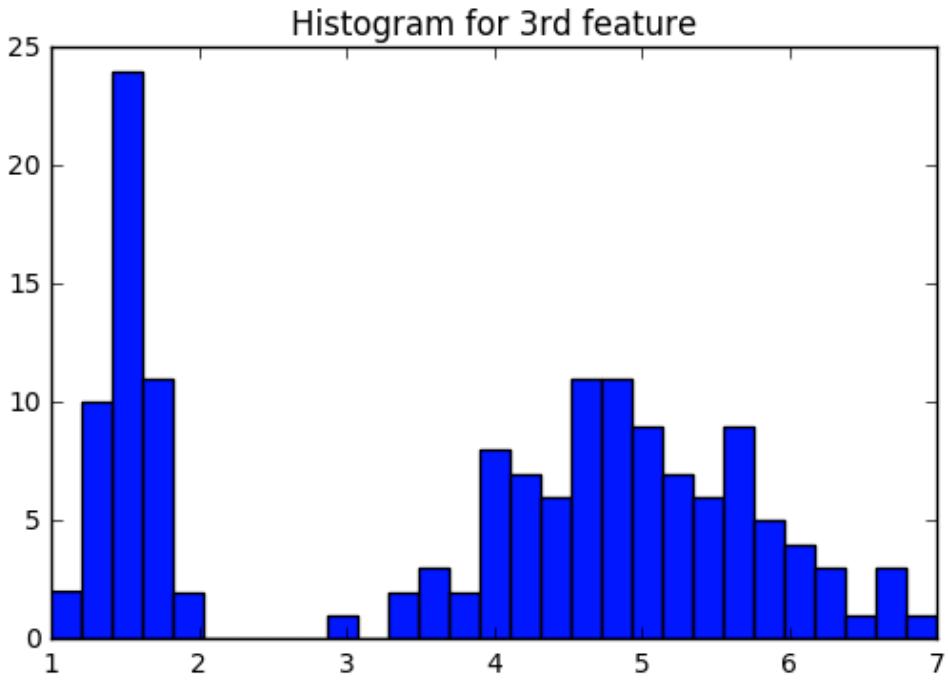
In [7]:
```
f2 = X[:,1] #
Bins = np.linspace(2,4.5, 30) # 2 > values > 5
plt.hist(f2,bins = Bins)
plt.title("Histogram for 2nd feature")
plt.show()
```



In [8]:
```
print (X[:,2]) #Again i Analyze the data in order to check the min values and the max values.
```

```
[ 1.4644697  1.3564281  1.5276568  1.4986909  1.7762706  1.4318731
  1.5274824  1.4125519  1.5758385  1.5765389  1.6430542  1.4755897
  1.1749194  1.2567601  1.5107326  1.3209401  1.4718295  1.7998278
  1.5676795  1.7954862  1.5773352  1.0238316  1.7821504  1.9969141
  1.6849479  1.6629449  1.5206862  1.4721063  1.6540981  1.6392436
  1.5593606  1.5701631  1.4409139  1.586193   1.2448985  1.3614918
  1.5572413  1.3373238  1.5288726  1.3842511  1.3121212  1.3327339
  1.6613321  1.9765757  1.4423184  1.6602356  1.4693496  1.5083478
  1.4989376  4.7918252  4.5801284  4.9446523  4.0491001  4.6133837
  4.5307328  4.7598275  3.3093526  4.6023079  3.9433124  3.5510767
  4.2959689  4.0479976  4.7617325  3.64511    4.453085   4.5890595
  4.134822   4.5589807  3.9956406  4.8190289  4.0304635  4.9199845
  4.7878583  4.37242    4.477464   4.8901952  5.001692   4.5374986
  3.5241584  3.8540178  3.7679702  3.9676624  5.138123   4.518779
  4.5698176  4.7465921  4.4052982  4.194808   4.0599264  4.4266212
  4.6815209  4.0744851  3.3823362  4.2404153  4.2641813  4.2284719
  4.3943772  3.0586676  4.1665081  6.0915733  5.1948135  5.9713065
  5.6423445  5.8739755  6.6489904  4.5407182  6.3592575  5.8774727
  6.1926853  5.1322237  5.3838356  5.5847755  5.0687842  5.1847108
  5.3536864  5.5171382  6.700331   6.9347277  5.0130118  5.700611   4.915451
  6.7680334  4.9392178  5.7216688  6.0751238  4.8678084  4.959205
  5.6439034  5.829466   6.1022063  6.4402054  5.6561366  5.1719669
  5.6596668  6.19409    5.6326364  5.5152685  4.8211268  5.4123429
  5.6607708  5.1796523  5.1357765  5.9498213  5.7722221  5.2847303
  5.0778827  5.2532499  5.4165338]
```

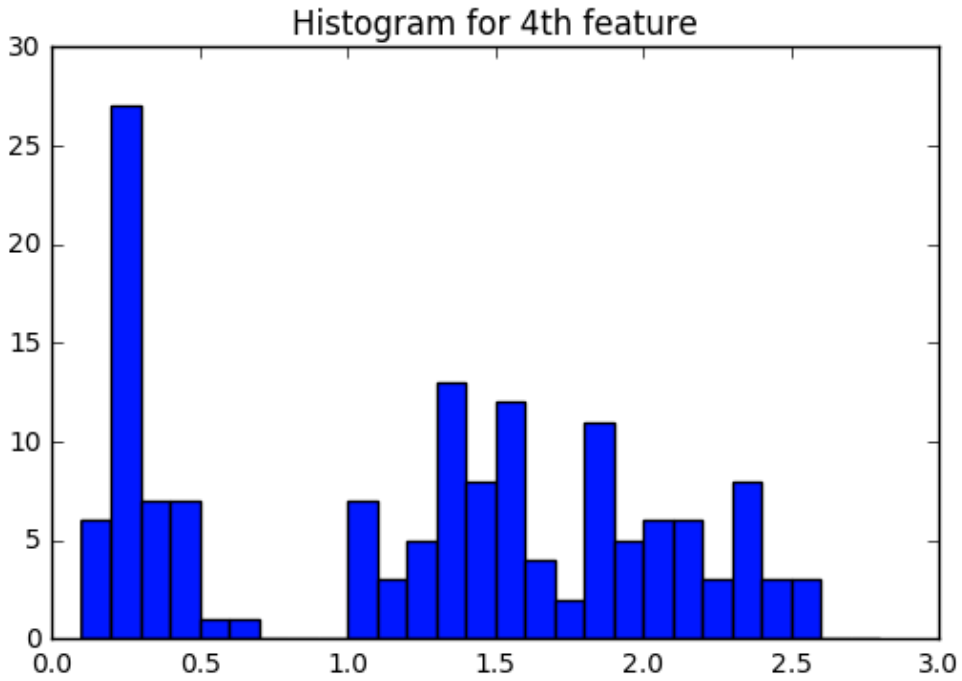
In [9]:
```
f3 = X[:,2] #
Bins = np.linspace(1,7, 30) # 2 > values > 5
plt.hist(f3,bins = Bins)
plt.title("Histogram for 3rd feature")
plt.show()
```



In [10]:
```
print X[:,3]#The values of the 4th feature
```

```
[ 0.27530531  0.2088531   0.27594034  0.23158353  0.40472913  0.31580663
  0.27401837  0.22260937  0.16081406  0.24882518  0.24454937  0.14781055
  0.12509037  0.29031447  0.48582304  0.44588467  0.32790175  0.36497322
  0.31879384  0.27925637  0.44809892  0.27169144  0.57595996  0.29792002
  0.21183207  0.48412659  0.28886228  0.24005617  0.21389081  0.26237446
  0.48092519  0.17986071  0.22469027  0.14427887  0.22588383  0.2659389
  0.19802051  0.22648398  0.29155335  0.30816019  0.31294706  0.25339295
  0.67518508  0.49668953  0.36869881  0.21193737  0.24611012  0.23250016
  0.20170725  1.4704444   1.5949115   1.5902054   1.3280813   1.5077236
  1.3234637   1.6360461   1.0831529   1.314734    1.4893971   1.0220861
  1.5520799   1.0887076   1.4663328   1.3133255   1.402491    1.5752051
  1.0058557   1.5084995   1.1968283   1.8527465   1.3000095   1.5882043
  1.2714598   1.3167463   1.4975133   1.4129098   1.7219667   1.5715136
  1.0102091   1.159344    1.0773659   1.2485955   1.6827298   1.5992032
  1.641381    1.5517088   1.3424061   1.3471648   1.3552137   1.2926645
  1.4835253   1.2864767   1.047818    1.3391932   1.2873475   1.3819564
  1.3775166   1.1790334   1.3683223   2.5771235   1.9422928   2.1071683
  1.8970702   2.2460323   2.157781    1.7372157   1.8522461   1.8974497
  2.5330226   2.0599316   1.941808    2.1206394   2.0602898   2.4592193
  2.3321161   1.890291    2.2510418   2.3403692   1.5286846   2.3486519
  2.078654    2.0274015   1.888334    2.1349981   1.8156282   1.8890369
  1.8679044   2.1379478   1.6824957   1.9818894   2.0749438   2.2589604
  1.558912    1.4360165   2.3013068   2.4042476   1.8161674   1.8432514
  2.135207    2.4988512   2.3136878   1.9347013   2.3024203   2.5763084
  2.3088859   1.9071244   2.0575521   2.3144559 ]
```

In [11]:
```
f4 = X[:,3] #
Bins = np.linspace(0.1,2.8, 28) # 0 > values > 5
plt.hist(f4,bins = Bins)
plt.title("Histogram for 4th feature")
plt.show()
```



Problem 1C

In [12]:
```
mf1,mf2,mf3,mf4 = np.mean(X,axis = 0)
print ("MEAN for Feature 1: {}\nMEAN for Feature 2: {} \nMEAN for Feature 3: {}\nMEAN for Feature 4: {}".format(mf1, mf
```

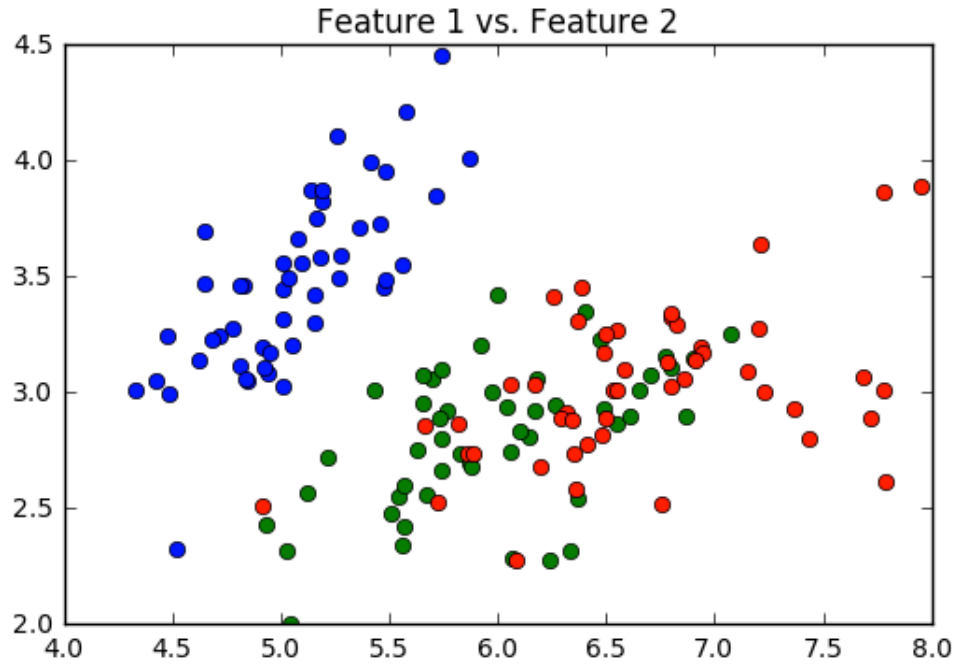
```
MEAN for Feature 1: 5.90010376419
MEAN for Feature 2: 3.09893091689
MEAN for Feature 3: 3.81955484054
MEAN for Feature 4: 1.25255548459
```

In [13]:
```
sf1,sf2,sf3,sf4 = np.std(X,axis = 0)
print ("STD for Feature 1: {}\nSTD for Feature 2: {} \nSTD for Feature 3: {}\nSTD for Feature 4: {}".format(sf1, sf2, sf
```

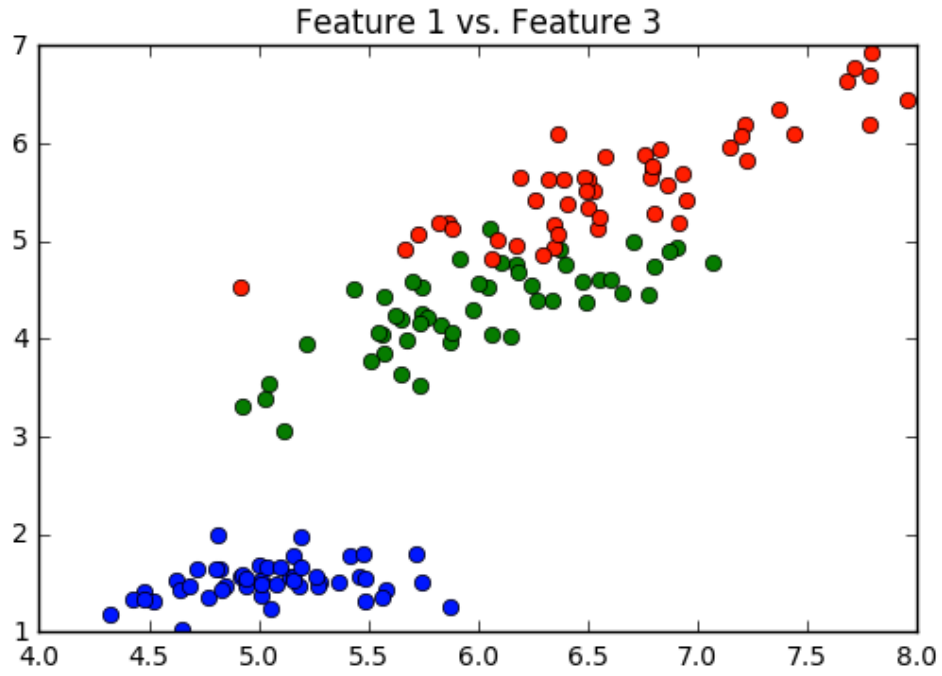
```
STD for Feature 1: 0.833402066775
STD for Feature 2: 0.436291838001
STD for Feature 3: 1.75405710934
STD for Feature 4: 0.758772457026
```

Problem 1D

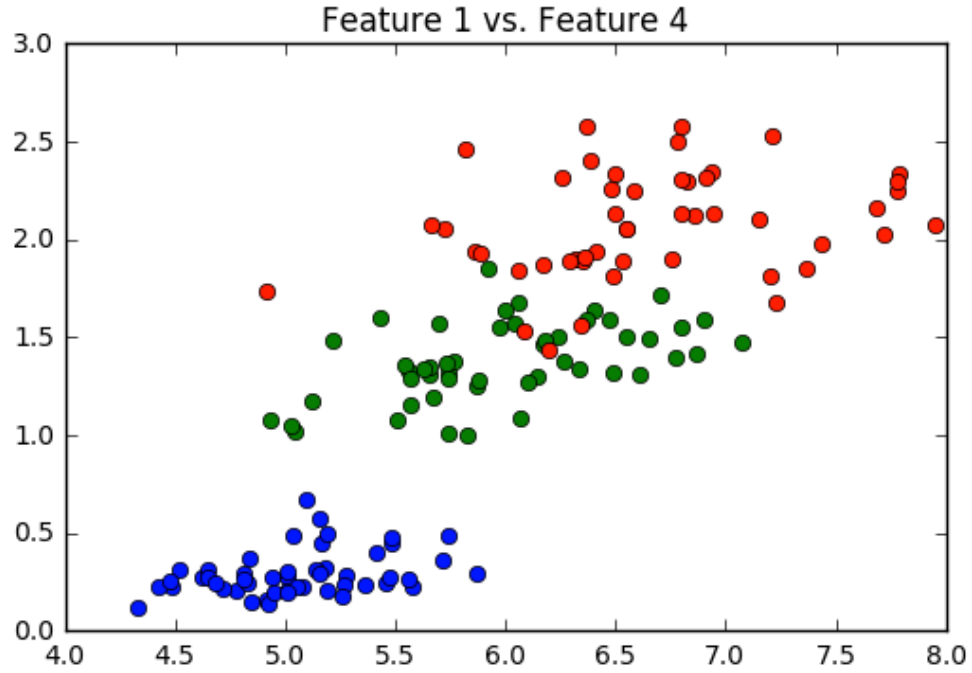
In [14]:
```
colors = ['b','g','r']#I chose three colors because there are 3 Classes for the iris flower.
for i in np.unique(Y):
    plt.plot(X[Y==i,0],X[Y==i,1],'o',color=colors[int(i)]) #This compares Feature1 vs Feature2
plt.title("Feature 1 vs. Feature 2")
plt.show()
```



In [15]:
```
colors = ['b','g','r']
for i in np.unique(Y):
    plt.plot(X[Y==i,0],X[Y==i,2],'o',color=colors[int(i)]) #This compares Feature 1 vs Feature 3
plt.title("Feature 1 vs. Feature 3")
plt.show()
```



In [16]:
```
colors = ['b','g','r']
for i in np.unique(Y):
    plt.plot(X[Y==i,0],X[Y==i,3],'o',color=colors[int(i)]) #This compares Feature 1 vs Feature 4
plt.title("Feature 1 vs. Feature 4")
plt.show()
```



In [248]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)

iris = np.genfromtxt("data/iris.txt",delimiter=None)
Y = iris[:,-1]
X = iris[:,0:2] #since iris[:,0:-1] gives you all the columns. I changed it to iris[:,0:2] to only show two columns
print (X)
```

```
[[ 4.9400593  3.081924 ]
 [ 4.7738176  3.2812362]
 [ 4.620137   3.1387625]
 [ 5.0774442  3.6684541]
 [ 5.415624   3.9942807]
 [ 4.6451451  3.4706607]
 [ 5.011375   3.4491627]
 [ 4.4814082  2.9916573]
 [ 4.917678   3.1954035]
 [ 5.4582963  3.7263543]
 [ 4.8244597  3.462181 ]
 [ 4.846128   3.0482423]
 [ 4.3265579  3.0098618]
 [ 5.8682189  4.0119708]
 [ 5.7420976  4.458073 ]
 [ 5.4832527  3.9533324]
 [ 5.1810909  3.5847085]
 [ 5.7146538  3.8543564]
 [ 5.1421791  3.870928 ]
```

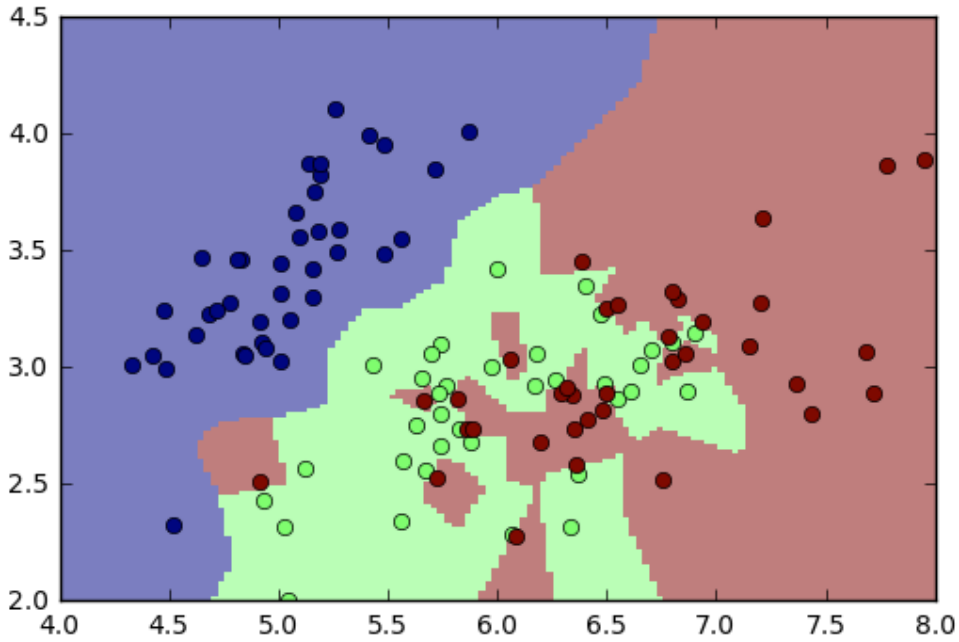
In [249]:
```
import mltools as ml
X,Y = ml.shuffleData(X,Y);
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y, 0.75);
```

Problem 2A

K=1

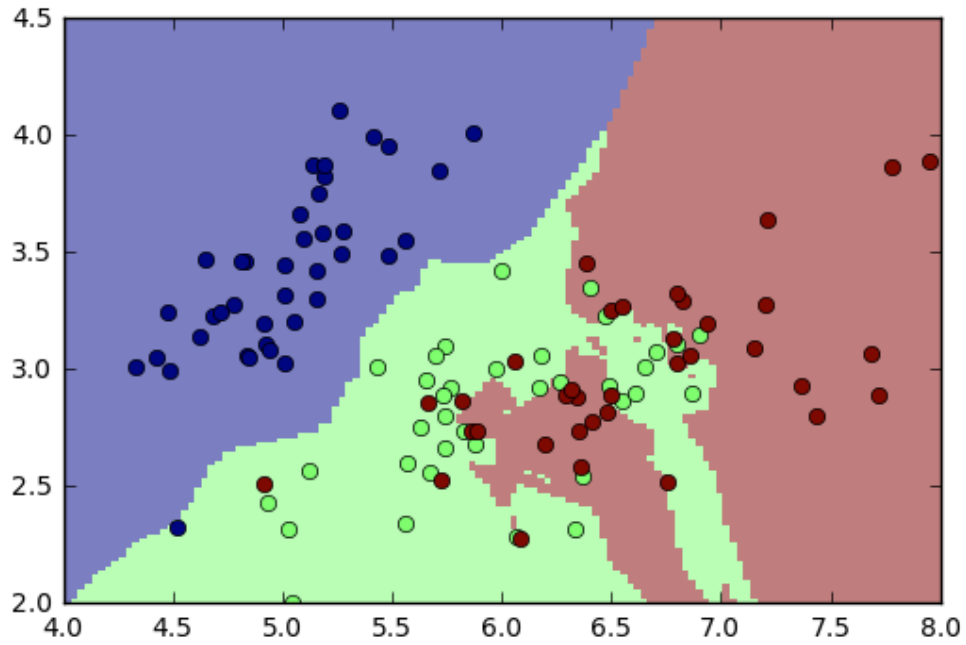
In [250]:
```
knn = ml.knn.knnClassify()
knn.train(Xtr, Ytr, 1) #The graph seems to be overfitting and we see that the boundries are very dependent.
                       #This could cause a higher error rate and less predictive performance.
YvaHat = knn.predict(Xva)
```

In [251]:
```
ml.plotClassify2D(knn, Xtr, Ytr );
plt.show()
```



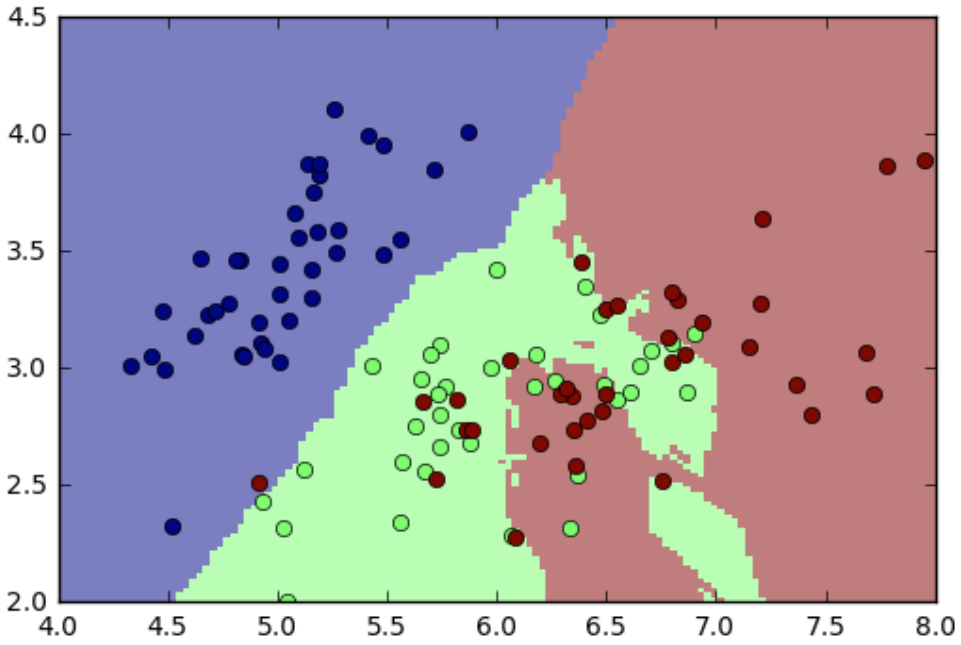
K=5

In [252]:
```
knn = ml.knn.knnClassify()
knn.train(Xtr, Ytr, 5) #when K=5 there seems to be a balance between the boundries.
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn, Xtr, Ytr );
plt.show()
```



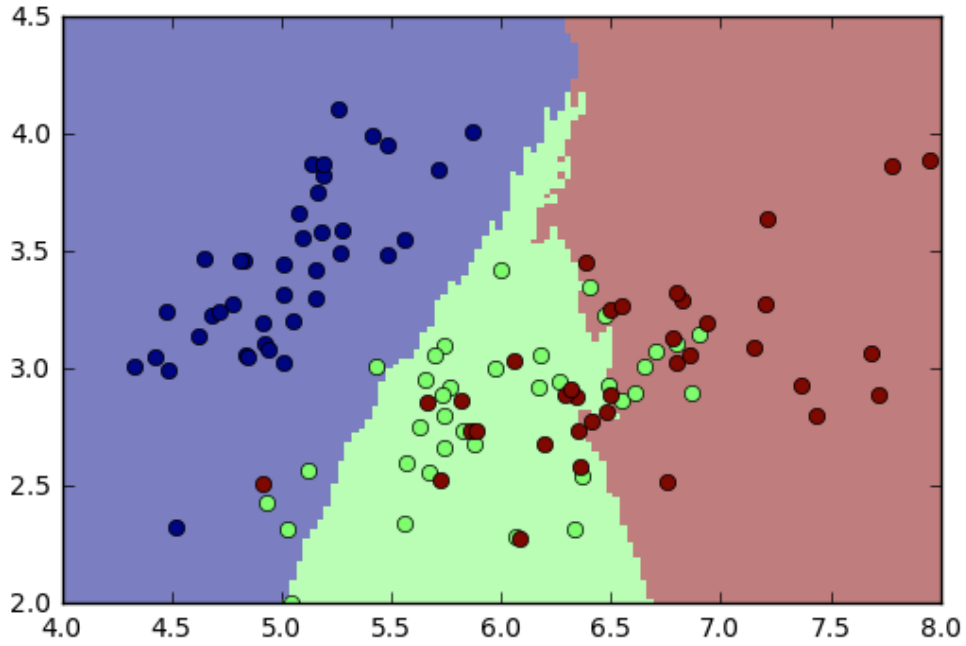
K=10

In [253]:
```
knn = ml.knn.knnClassify() # create the object and train it
knn.train(Xtr, Ytr, 10) # where K is an integer, e.g. 1 for nearest neighbor prediction
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn, Xtr, Ytr );
plt.show()
```



K=50

In [254]:
```
knn = ml.knn.knnClassify()
knn.train(Xtr, Ytr, 50) #when K=50 we see that the boundries for each color are underfitting since it does not fit
                        #the data enough .
YvaHat = knn.predict(Xva)
ml.plotClassify2D(knn, Xtr, Ytr );
plt.show()
```



Problem 2B

In [255]:

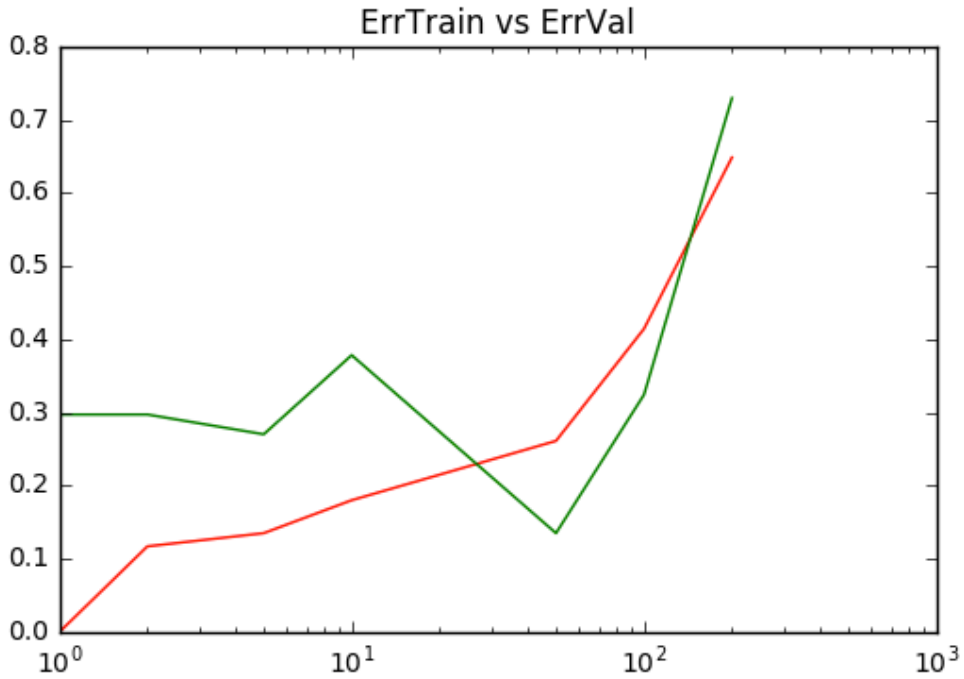
```
K=[1,2,5,10,50,100,200];
#Arrays to input the data given by the err(x,y)

errTrain = [0,0,0,0,0,0,0]
errVal = [0,0,0,0,0,0,0]

#errTrain = np.empty(100)
#errVal = np.empty(100)
#print (Xtr)
#print (Ytr)
#print (Xva)


for i,k in enumerate(K):
    learner = ml.knn.knnClassify(Xtr,Ytr,k);
    Yhat = learner.predict(Xva)
    errTrain[i] = learner.err(Xtr,Ytr)
    errVal[i] = learner.err(Xva,Yva)

plt.title("ErrTrain vs ErrVal")
plt.semilogx(K, errTrain,'r', K, errVal,'g')
plt.show()
```



In [ ]:
```
# K=50 is the ideal integer value i would recommend to avoid overfitting and underfiting
#we can see that at the begining of the graph the distance of the two lines is larger than any other place in the graph
#This means that the data is being overfitting which causes more errors and it has less predictive performance.
```

Problem 3A

$P(Y=1) = \boxed{4/10}$ $\quad P(Y=1) = 1 - P(Y=-1)$

$P(Y=-1) = \boxed{6/10}$ $\quad P(Y=-1) = 1 - P(Y=1)$

$P(X_1=1 \mid Y=1) = 3/4$	$P(X_1=1 \mid Y=+1) = 3/6$
$P(X_2=1 \mid Y=1) = 0/4$	$P(X_2=1 \mid Y=-1) = 5/6$
$P(X_3=1 \mid Y=1) = 3/4$	$P(X_3=1 \mid Y=-1) = 4/6$
$P(X_4=1 \mid Y=1) = 2/4$	$P(X_4=1 \mid Y=+1) = 5/6$
$P(X_5=1 \mid Y=1) = 1/4$	$P(X_5=1 \mid Y=-1) = 2/6$

$P(X_1=0 \mid Y=1) = 1/4$	$P(X_1=0 \mid Y=-1) = 3/6$
$P(X_2=0 \mid Y=1) = 4/4$	$P(X_2=0 \mid Y=-1) = 1/6$
$P(X_3=0 \mid Y=1) = 1/4$	$P(X_3=0 \mid Y=-1) = 2/6$
$P(X_4=0 \mid Y=1) = 2/4$	$P(X_4=0 \mid Y=-1) = 1/6$
$P(X_5=0 \mid Y=1) = 3/4$	$P(X_5=0 \mid Y=-1) = 4/6$

Problem 3B

$\underline{X} = (0,0,0,0,0)$ will predict class $\boxed{y=-1}$

$\underline{X} = (1,1,0,1,0)$ will predict class $\boxed{y=-1}$

Problem 3C

$$P(y=+1 \mid \underline{X}=(1\,1\,0\,1\,0)) = \frac{P(X=(1\,1\,0\,1\,0) \mid y=+1)\ P(y=+1)}{P(X=(1\,1\,0\,1\,0))}$$

$$= \left(\frac{3}{4}\right) \times \left(\frac{0}{4}\right) \times \left(\frac{1}{4}\right) \times \left(\frac{2}{4}\right) \times \left(\frac{3}{4}\right) \times \left(\frac{4}{10}\right)$$

$= \boxed{0}$ because $P(X=1 \mid y=+1) = 0$

Problem 3D

We shouldn't use Joint Bayes classifier for these data because you will not be accounting for the probabilities of Individual Features.. Instead, you will be looking at a probability of a set of features. For instance, finding the likelihood ($P(X_1|Y)$) will not be possible because if we use Joint Bayers Classifier, the likelihoods will look like $P(X_1 X_2 X_3 X_4 X_5 | Y)$.

Problem 3E

Yes we should retrain the model. This could be done through a learner.

Without X_1 the paremeters change for a Naive Bayes model because only $X_2 \cdots X_5$ Features remain. Meaning there are only 2^4 possible paremeters. since there are 4 features left. The 2 is because Y is {read, notread}