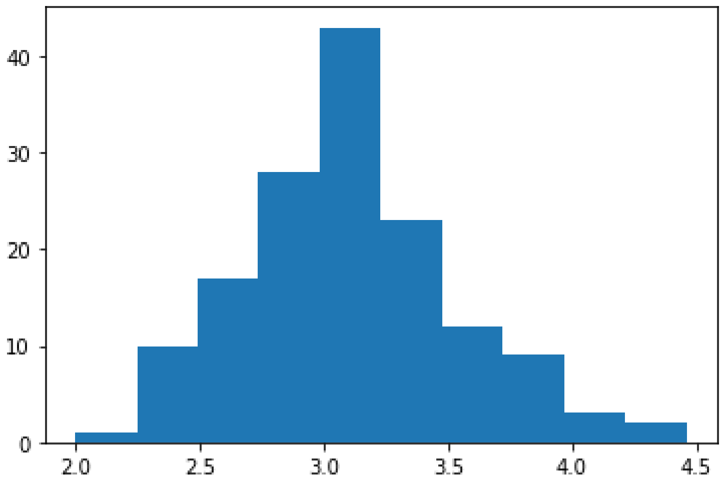
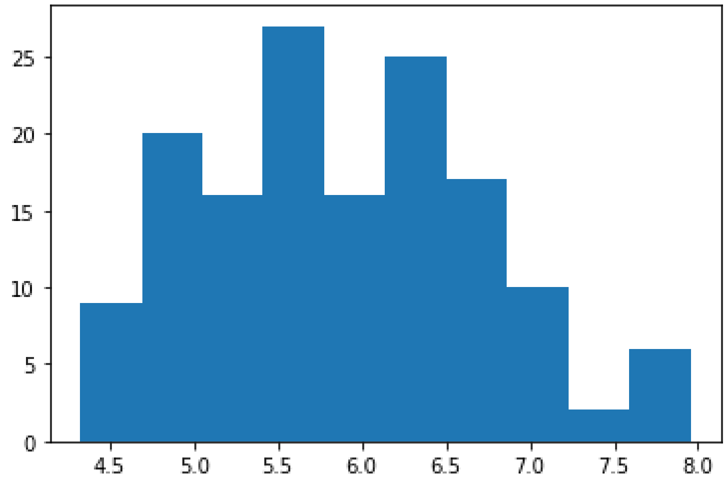
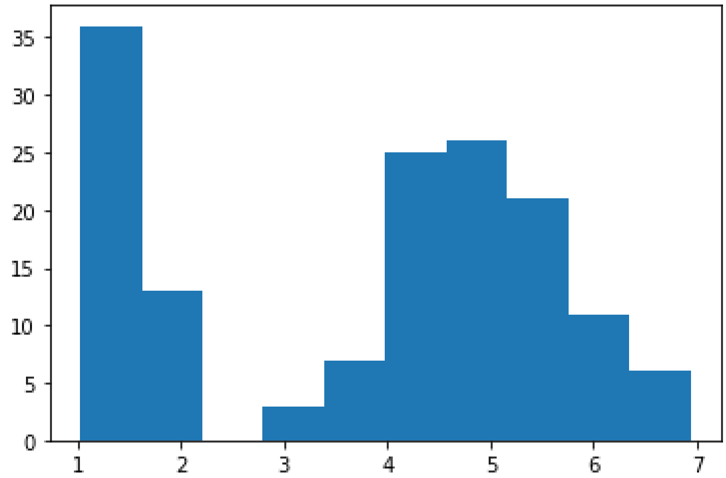
In [45]:
```
import numpy as np

import mltools as ml
import matplotlib.pyplot as plt
iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
```

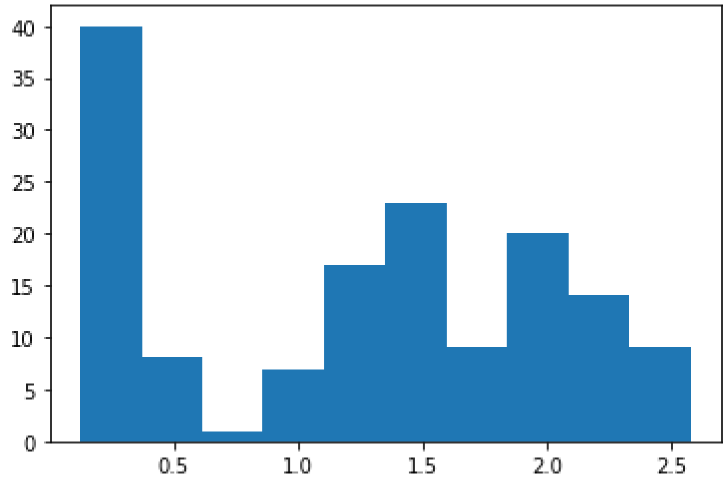
In [60]:
```
#Probleml 1
Y = iris[:,-1] # target value (iris species) is the last column
X = iris[:,0:-1]
print(iris.shape)


m, n = X.shape
print(n,"features")
print(m,"number of data points")




#Problem 1 2

plt.hist(X[:,0])
plt.show()

plt.hist(X[:,1])
plt.show()

plt.hist(X[:,2])
plt.show()

plt.hist(X[:,3])
plt.show()



plt.show()
```

```
(148, 5)
4 features
148 number of data points
```









In [64]:

```
#Problem 1 3
print(np.mean(X,axis=0),"mean")
print(np.var(X,axis=0),"var")
print(np.std(X,axis=0),"std")
```

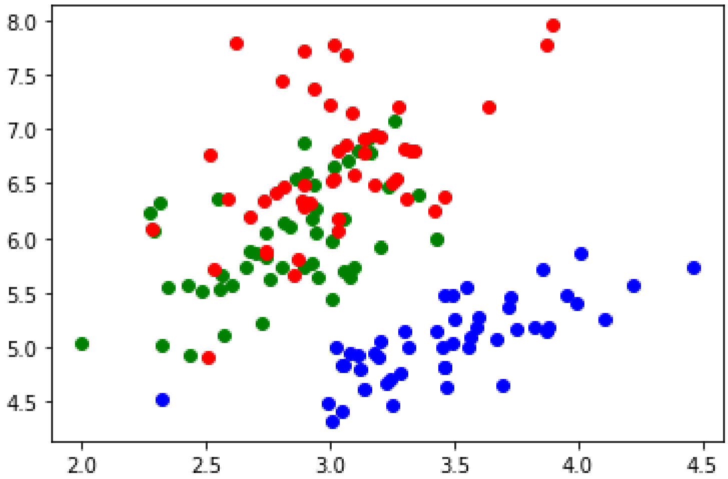
```
[3.09893092 5.90010376 3.81955484 1.25255548] mean
[0.19035057 0.694559   3.07671634 0.57573564] var
[0.43629184 0.83340207 1.75405711 0.75877246] std
```

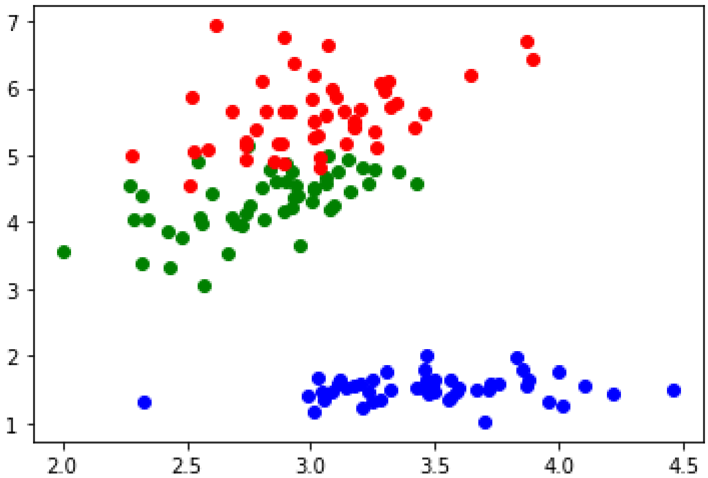
Problem 1 4

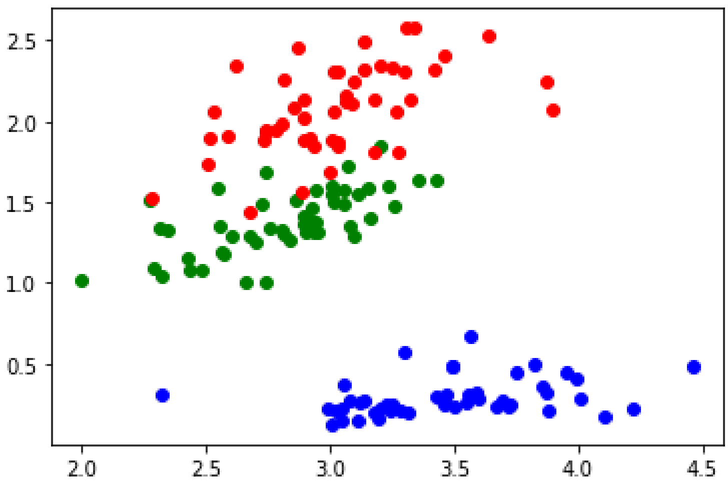
In [63]:

```
plt.scatter(X[np.where(Y==0), 0],X[np.where(Y==0), 1],c=['b'])
plt.scatter(X[np.where(Y==1), 0],X[np.where(Y==1), 1],c=['g'])
plt.scatter(X[np.where(Y==2), 0],X[np.where(Y==2), 1],c=['r'])
```

```
plt.show()
plt.scatter(X[np.where(Y==0), 0],X[np.where(Y==0), 2],c=['b'])
plt.scatter(X[np.where(Y==1), 0],X[np.where(Y==1), 2],c=['g'])
plt.scatter(X[np.where(Y==2), 0],X[np.where(Y==2), 2],c=['r'])


plt.show()

plt.scatter(X[np.where(Y==0), 0],X[np.where(Y==0), 3],c=['b'])
plt.scatter(X[np.where(Y==1), 0],X[np.where(Y==1), 3],c=['g'])
plt.scatter(X[np.where(Y==2), 0],X[np.where(Y==2), 3],c=['r'])
plt.show()
```





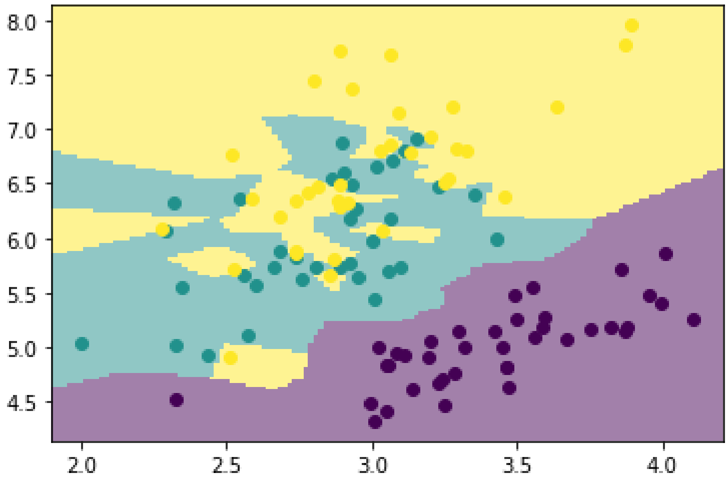


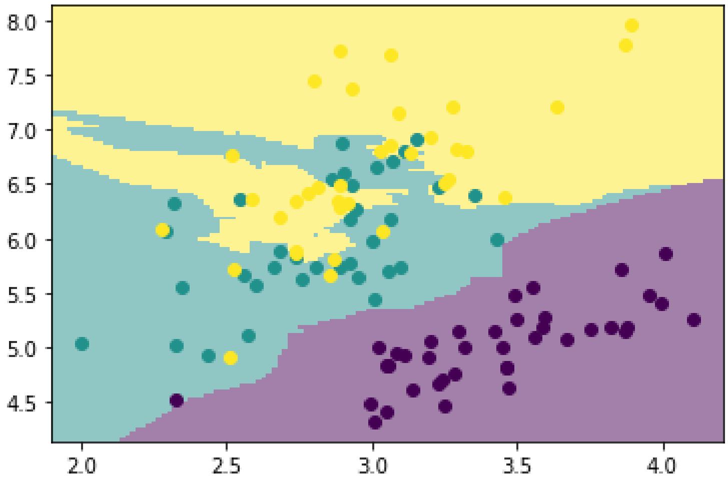
In [6]:
```
iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the data
Y = iris[:,-1]
X = iris[:,0:-1]
```

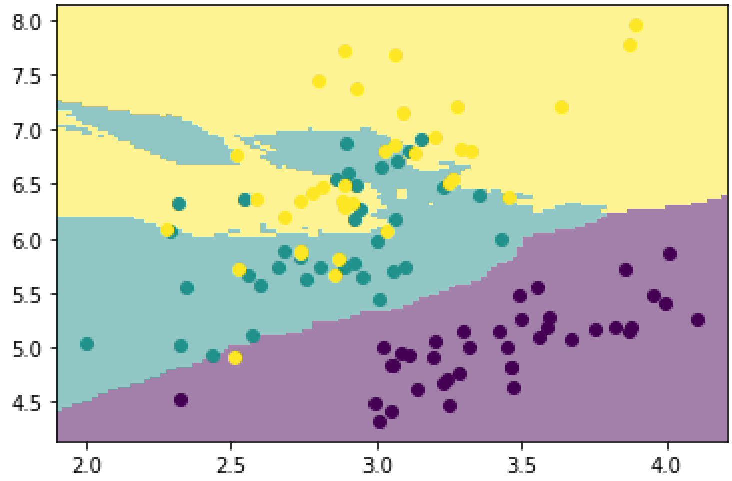
```
np.random.seed(0) # set the random number seed
X,Y = ml.shuffleData(X,Y); # shuffle data randomly
 # (This is a good idea in case your data are ordered in some systematic way.)

Xtr,Xva,Ytr,Yva = ml.splitData(X,Y, 0.75); # split data into 75/25 train/validation
```

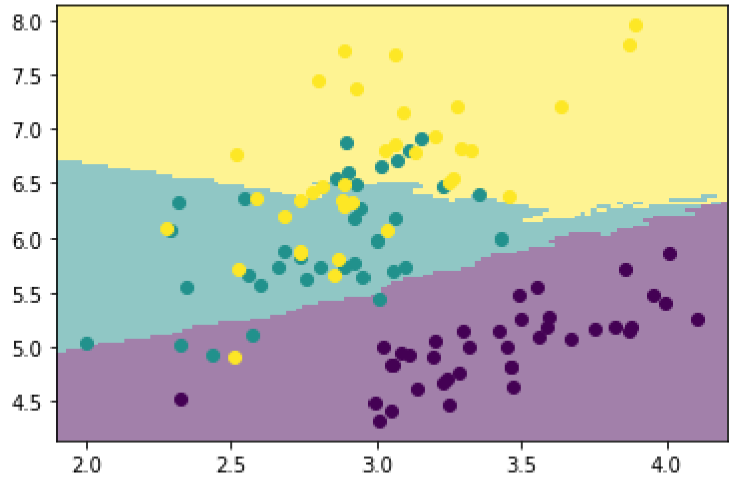
In [54]:

```
#Problem 2 1
knn=ml.knn.knnClassify()

knn.train(Xtr[:,:2],Ytr,K=1)

ml.plotClassify2D(knn,Xtr[:,:2],Ytr)
plt.show()

knn=ml.knn.knnClassify()
knn.train(Xtr[:,:2],Ytr,K=5)

ml.plotClassify2D(knn,Xtr[:,:2],Ytr)
plt.show()

knn=ml.knn.knnClassify()
knn.train(Xtr[:,:2],Ytr,K=10)

ml.plotClassify2D(knn,Xtr[:,:2],Ytr)
plt.show()

knn=ml.knn.knnClassify()
knn.train(Xtr[:,:2],Ytr,K=50)


ml.plotClassify2D(knn,Xtr[:,:2],Ytr)
plt.show()
```



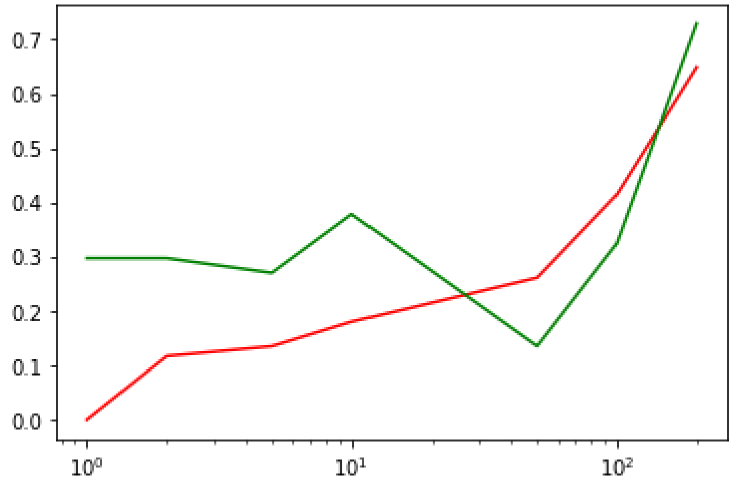






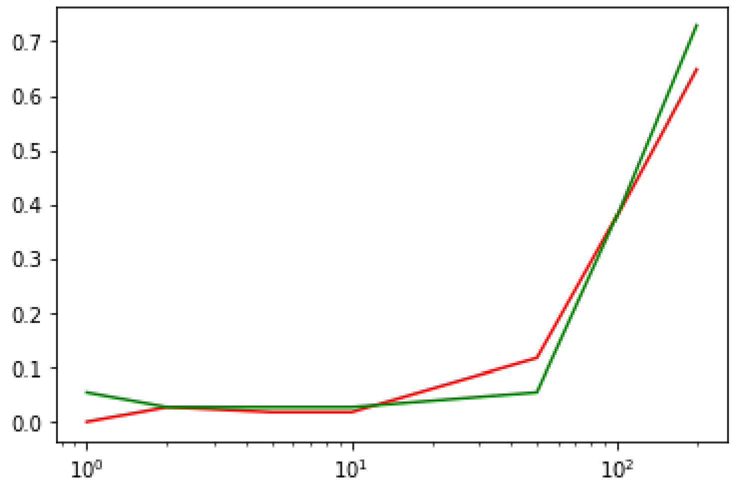
In [56]:

```
##Problem 2 2
K=[1,2,5,10,50,100,200];
errTrain = np.zeros((len(K),))
errValid = np.zeros((len(K),))


for i,k in enumerate(K):
    knn = ml.knn.knnClassify();
    knn.train(Xtr[:,:2], Ytr, k)



    errTrain[i] = knn.err(Xtr[:,:2], Ytr)
    errValid[i] = knn.err(Xva[:,:2], Yva)


# print()
plt.semilogx(K, errTrain,'r')
plt.semilogx(K, errValid,'g')
plt.show()
```



K=50 green line is lowest, and has the lowest validation error, so I would choose 50

In [59]:
```
##Problem 2 3
K=[1,2,5,10,50,100,200];
errTrain = np.zeros((len(K),))
errValid = np.zeros((len(K),))
plt.show()

for i,k in enumerate(K):
    learner = ml.knn.knnClassify()
    knn.train(Xtr, Ytr, K=k)

    errTrain[i] = knn.err(Xtr, Ytr)
    errValid[i] = knn.err(Xva, Yva)


plt.semilogx(K, errTrain,'r')
plt.semilogx(K, errValid,'g')

plt.show()
```



K=1 is not has the lower validation error than 2 and then the following K values is not better the efficienct at k=2

Problem 3 1

p(x1| y0)=1/2

p(x2| y0)=5/6

p(x3| y0)=2/3

p(x4| y0)=5/6

p(x5| y0)=1/3

p(x1| y1)=3/4

p(x2| y1)=0

p(x3| y1)=3/4

p(x4| y1)=1/2

p(x5| y1)=1/4

In [51]:

```
#Problem 3 2
px1y0=1/2
px2y0=5/6
px3y0=2/3
px4y0=5/6
px5y0=1/3


px1y1=3/4
px2y1=0
px3y1=3/4
px4y1=1/2
px5y1=1/4
#x = (0 0 0 0 0)
a=4/10*(1-px1y1)*(1-px2y1)*(1-px3y1)*(1-px4y1)*(1-px5y1)
b=6/10*(1-px1y0)*(1-px2y0)*(1-px3y0)*(1-px4y0)*(1-px5y0)
print("p y1x00000",a/(a+b))




#x = (1 1 0 1 0)
print()
c=4/10*(px1y1)*(px2y1)*(1-px3y1)*(px4y1)*(1-px5y1)

d=6/10*(px1y0)*(px2y0)*(1-px3y0)*(px4y0)*(1-px5y0)

print("p y1x11010",c/(c+d))
```

```
p y1x00000 0.8350515463917526

p y1x11010 0.0
```

In [47]:

```
#Problem 3 3

#x = (0 0 0 0 0)
a=4/10*(1-px1y1)*(1-px2y1)*(1-px3y1)*(1-px4y1)*(1-px5y1)
b=6/10*(1-px1y0)*(1-px2y0)*(1-px3y0)*(1-px4y0)*(1-px5y0)
print(a>b)

print(" class is 1")
```

```
#x = (1 1 0 1 0)
print()
c=4/10*(px1y1)*(px2y1)*(1-px3y1)*(px4y1)*(1-px5y1)
print()
d=6/10*(px1y0)*(px2y0)*(1-px3y0)*(px4y0)*(1-px5y0)

print(c>d)
print(" class is -1")
```

```
True
 class is 1


False
 class is -1
```

3.4 We should not use joint bayes because the joint probability would create more spots for data, however, we only have 10 data for each feature totest, it would cause problems for those mismatched data.

3.5 No, we dont retrain our model since we lose the x1 and we know the independence of features, we would only calculate the rest features using x2, x3,x4,x5

4 I have done this homework by myself
Minzhechen