homework1

October 7, 2018

1 Problem 1: Python & Data Exploration

1.1 part 1

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt

        iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
        Y = iris[:,-1] # target value is the last column
        X = iris[:,0:-1] # features are the other columns
        X.shape
```

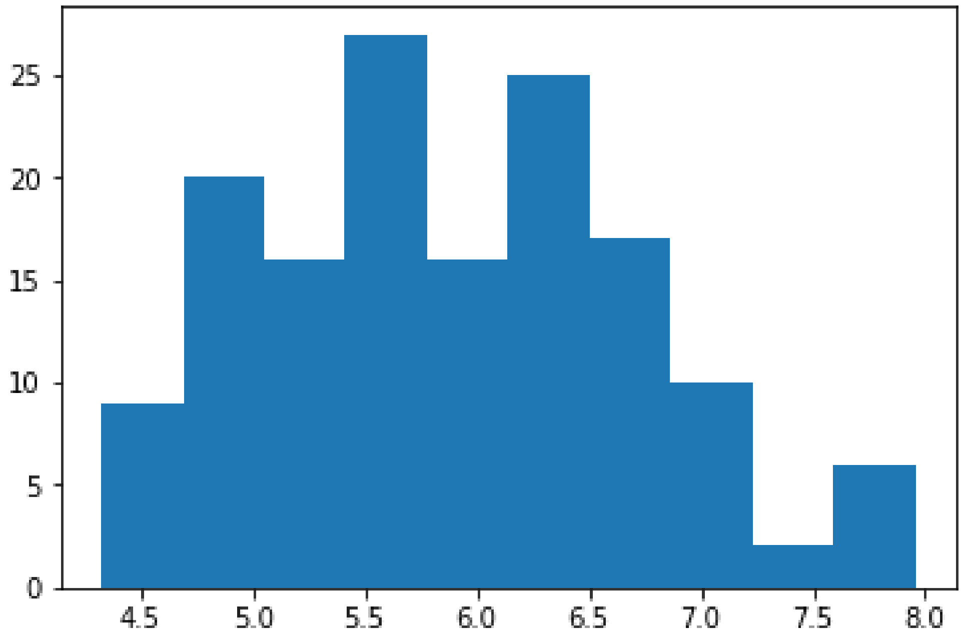
```
Out[2]: (148, 4)
```

4 is the number of features , 148 is the number of data points.

1.2 part 2

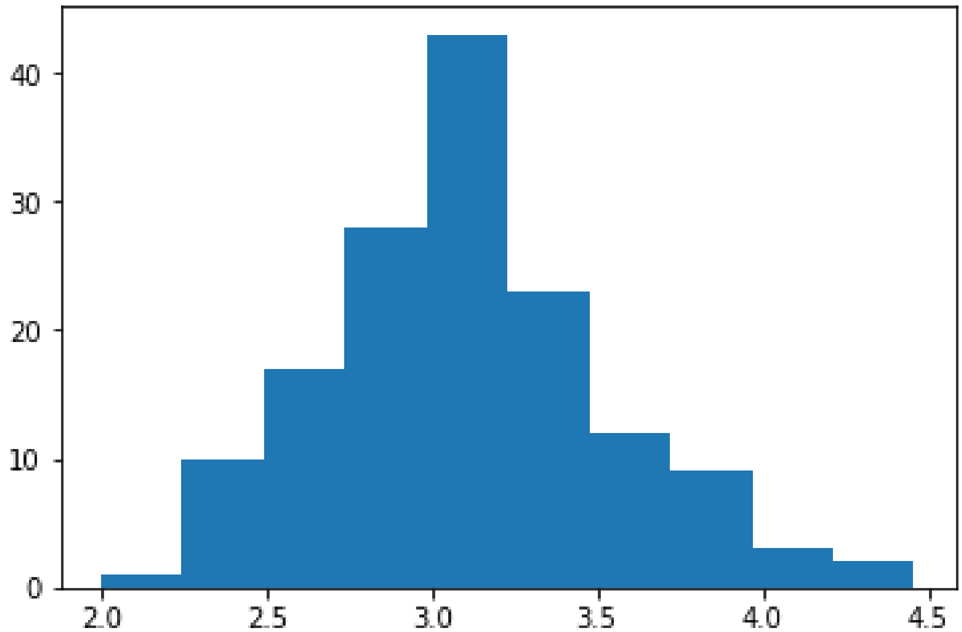
Feature 1 :

```
In [9]: import numpy as np
        import matplotlib.pyplot as plt

        iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
        Y = iris[:,-1] # target value is the last column
        X = iris[:,0:-1] # features are the other columns
        m = X[:,0]
        plt.hist(m  )
        plt.show()
```



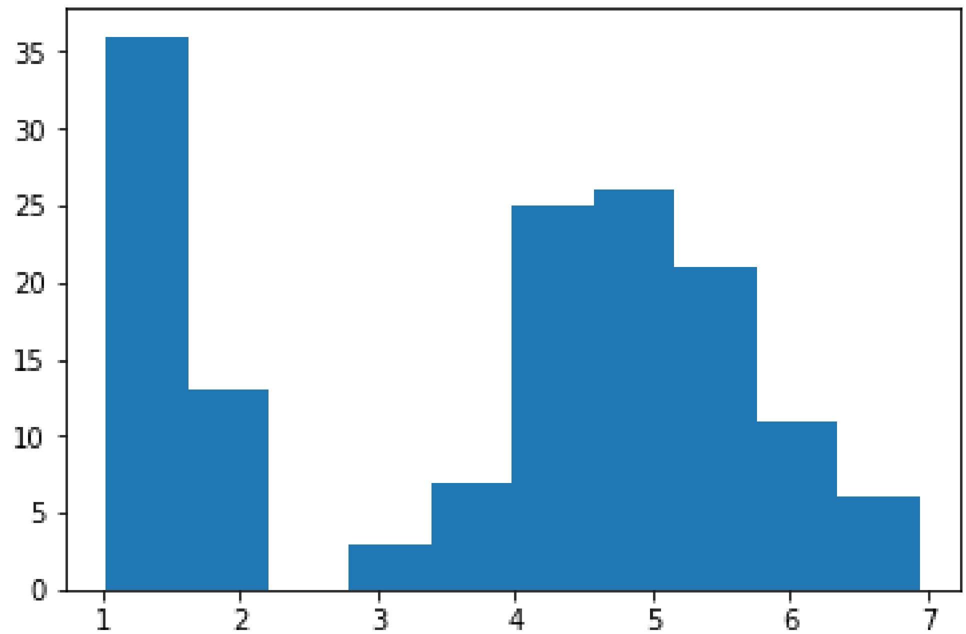
after fixing the parameter bin and X[;,0], i have the result. The rest is the same.
Feature 2 :

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt

iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
Y = iris[:,-1] # target value is the last column
X = iris[:,0:-1] # features are the other columns
m = X[:,1]
plt.hist(m  )
plt.show()
```



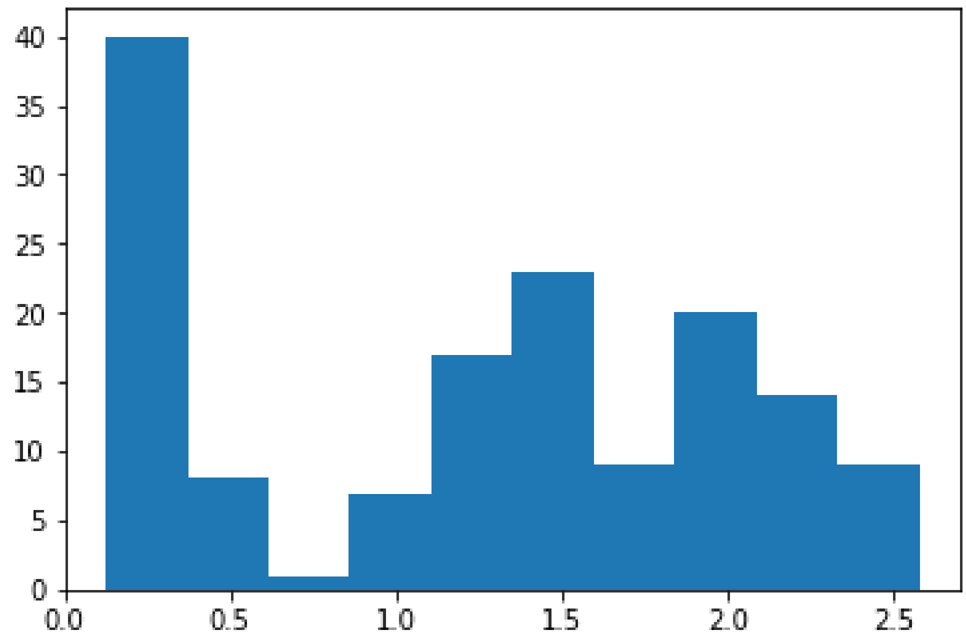
Fearture 3 :

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

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         m = X[:,2]
         plt.hist(m  )
         plt.show()
```



Feature 4:

```
In [12]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         m = X[:,3]
         plt.hist(m  )
         plt.show()
```



1.3 part 3

```
In [20]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         m = X[:,0]
         print(np.mean(m))
         print(np.std(m))
```

```
5.900103764189188
0.833402066774894
```

For feature 1 , mean is 5.9001 while standard deviation is 0.8334 .

```
In [21]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         m = X[:,1]
         print(np.mean(m))
         print(np.std(m))
```

```
3.098930916891892
0.43629183800107685
```

For feature 2 , mean is 3.0989 while standard deviation is 0.43629 .

```
In [22]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         m = X[:,2]
         print(np.mean(m))
         print(np.std(m))
```

```
3.8195548405405404
1.7540571093439352
```

For feature 2 , mean is 3.8196 while standard deviation is 1.754 .

```
In [23]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         m = X[:,3]
         print(np.mean(m))
         print(np.std(m))
```

```
1.2525554845945945
0.7587724570263247
```

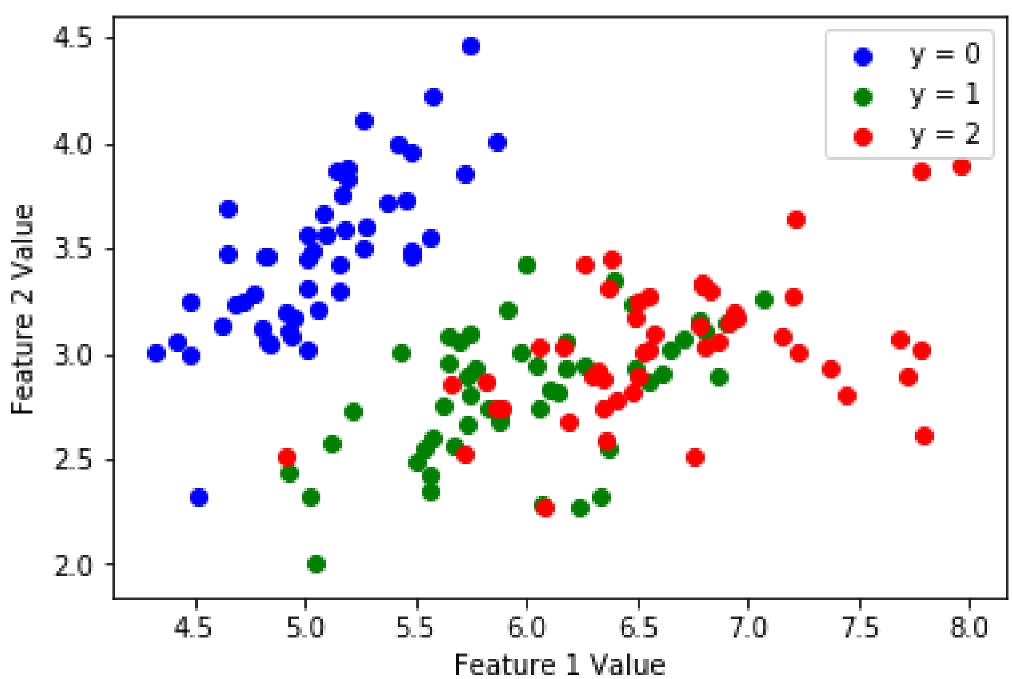
For feature 2 , mean is 1.2526 while standard deviation is 0.7588 .

1.4 part 4

Features (1 , 2) :

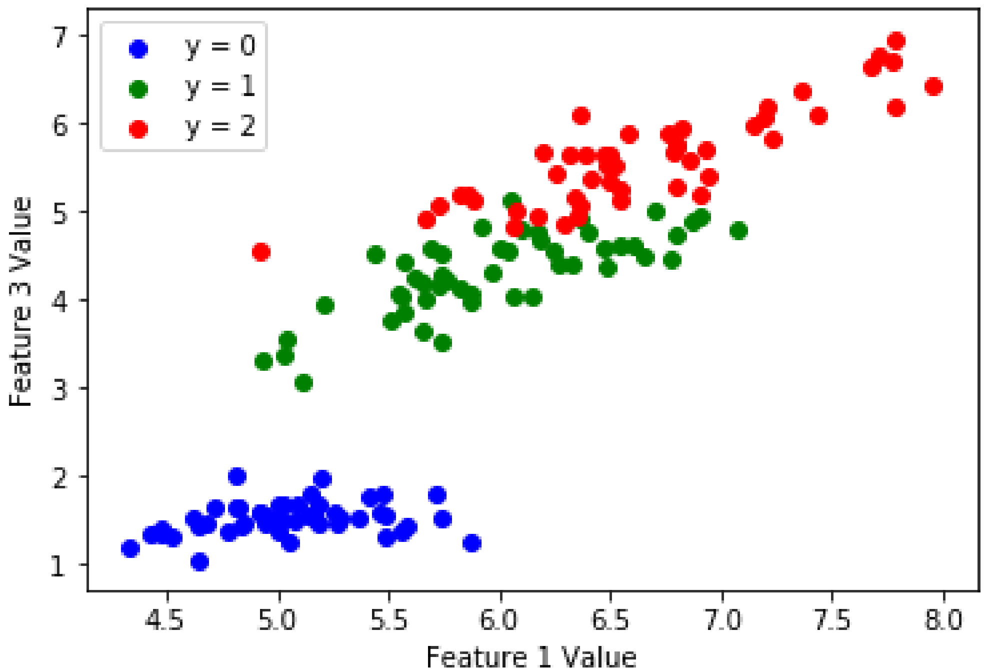
```
In [24]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         plt.xlabel("Feature 1 Value")
```

```
plt.ylabel("Feature 2 Value")
class0x = []
class0y = []
class1x = []
class1y = []
class2x = []
class2y = []
for i in range(148):
    if Y[i] == 0:  # y = 0
        class0x.append(X[i,0])
        class0y.append(X[i,1])

    if Y[i] == 1:  # y = 1
        class1x.append(X[i,0])
        class1y.append(X[i,1])

    if Y[i] == 2:  # y = 2
        class2x.append(X[i,0])
        class2y.append(X[i,1]) #class all the data points by y
plt.scatter(class0x,class0y,c = 'b', label = 'y = 0')
plt.legend()
plt.scatter(class1x,class1y,c = 'g', label = 'y = 1')
plt.legend()
plt.scatter(class2x,class2y,c = 'r', label = 'y = 2')
plt.legend()
plt.show()
```



Feature (1 , 3) :

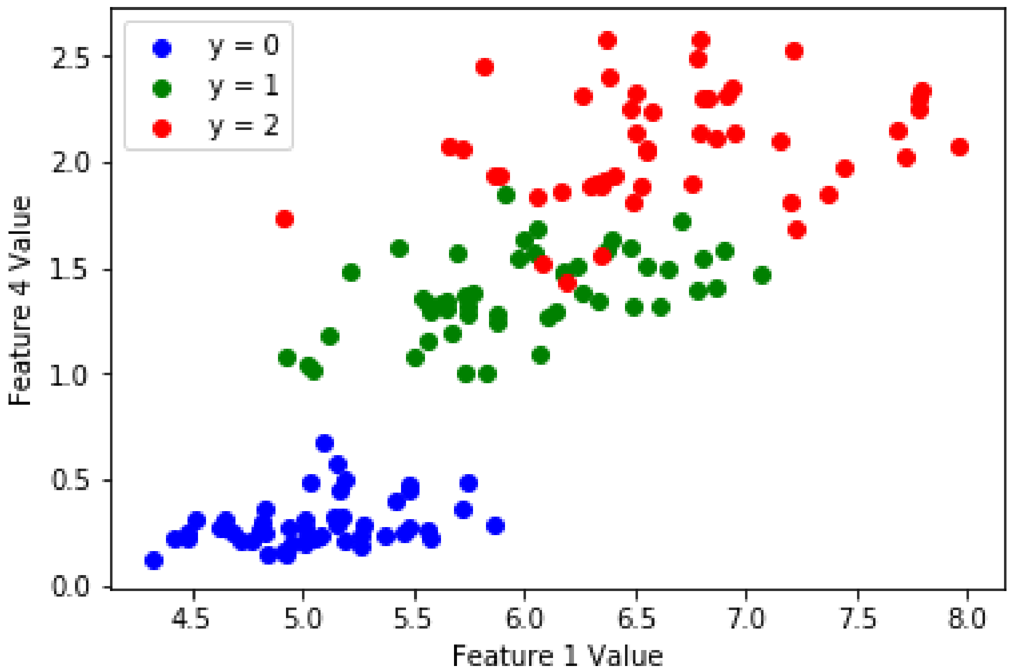
In [27]:
```
import numpy as np
import matplotlib.pyplot as plt

iris = np.genfromtxt("",delimiter=None) # load the text file
Y = iris[:,-1] # target value is the last column
X = iris[:,0:-1] # features are the other columns
plt.xlabel("Feature 1 Value")
plt.ylabel("Feature 3 Value")
class0x = []
class0y = []
class1x = []
class1y = []
class2x = []
class2y = []
for i in range(148):
    if Y[i] == 0:  # y = 0
        class0x.append(X[i,0])
        class0y.append(X[i,2])

    if Y[i] == 1:  # y = 1
        class1x.append(X[i,0])
        class1y.append(X[i,2])

    if Y[i] == 2:  # y = 2
        class2x.append(X[i,0])
        class2y.append(X[i,2]) #class all the data points by y
plt.scatter(class0x,class0y,c = 'b', label = 'y = 0')
plt.legend()
plt.scatter(class1x,class1y,c = 'g', label = 'y = 1')
plt.legend()
plt.scatter(class2x,class2y,c = 'r', label = 'y = 2')
plt.legend()
plt.show()
```



Feature (1 , 4) :

```
In [28]: import numpy as np
         import matplotlib.pyplot as plt

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the text file
         Y = iris[:,-1] # target value is the last column
         X = iris[:,0:-1] # features are the other columns
         plt.xlabel("Feature 1 Value")
         plt.ylabel("Feature 4 Value")
         class0x = []
         class0y = []
         class1x = []
         class1y = []
         class2x = []
         class2y = []
         for i in range(148):
             if Y[i] == 0:  # y = 0
                 class0x.append(X[i,0])
                 class0y.append(X[i,3])

             if Y[i] == 1:  # y = 1
                 class1x.append(X[i,0])
                 class1y.append(X[i,3])
```

```
    if Y[i] == 2:  # y = 2
        class2x.append(X[i,0])
        class2y.append(X[i,3]) #class all the data points by y
plt.scatter(class0x,class0y,c = 'b', label = 'y = 0')
plt.legend()
plt.scatter(class1x,class1y,c = 'g', label = 'y = 1')
plt.legend()
plt.scatter(class2x,class2y,c = 'r', label = 'y = 2')
plt.legend()
plt.show()
```

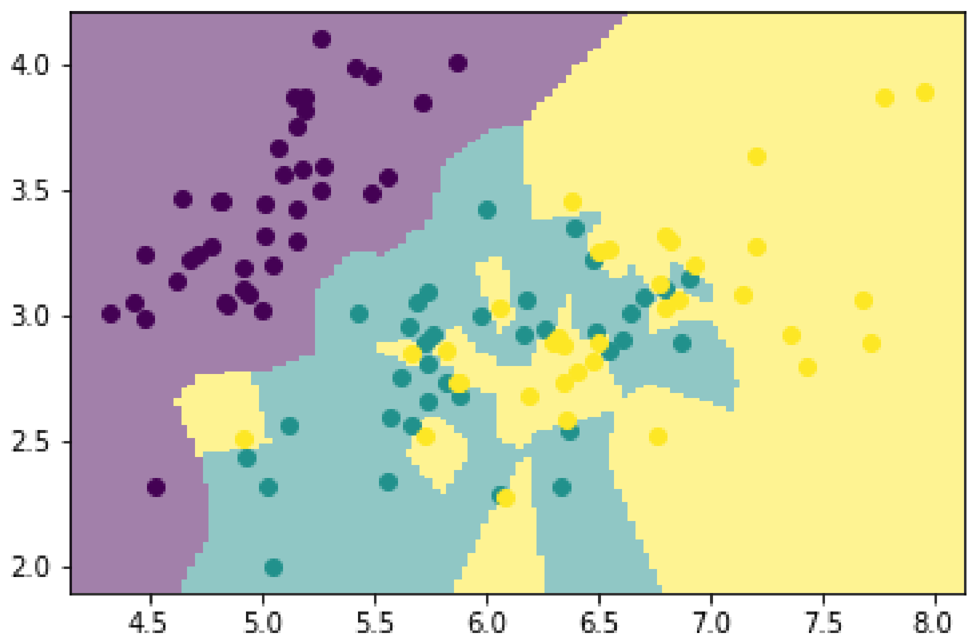


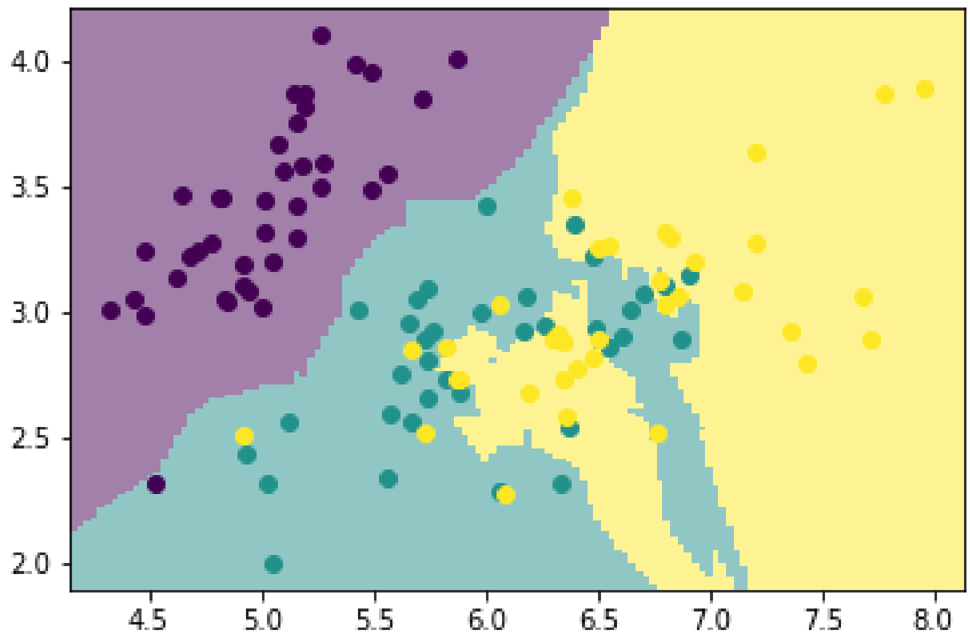
2 Problem 2: kNN predictions

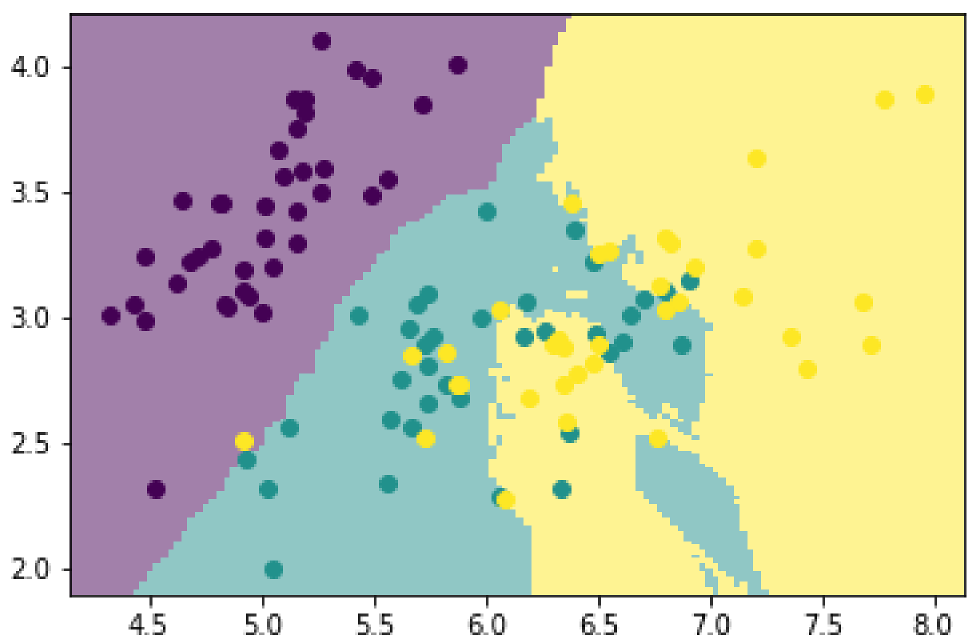
2.1 part 1

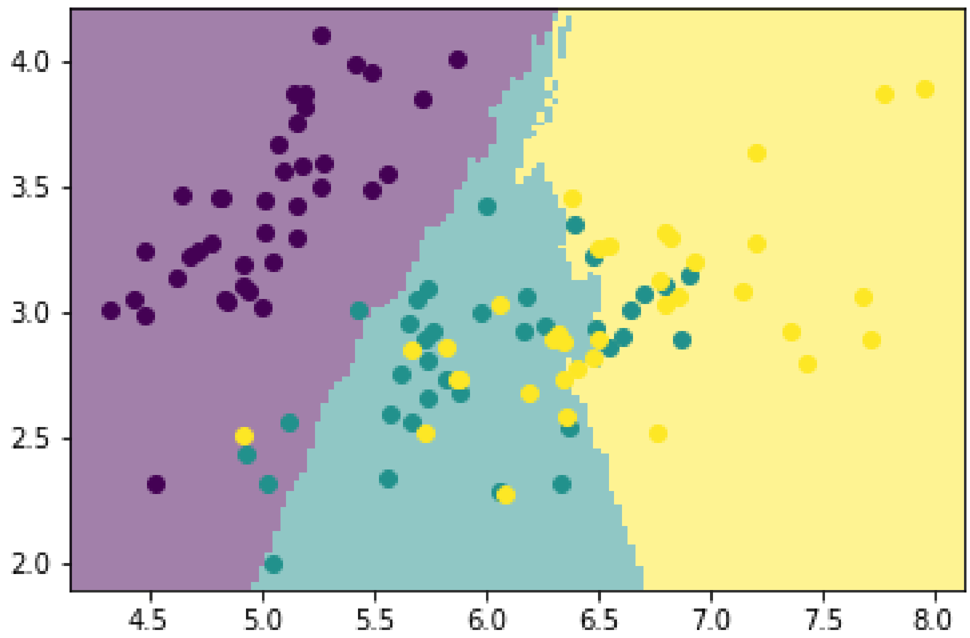
```
In [29]: import numpy as np
         import matplotlib.pyplot as plt
         import sys
         sys.path.append('/path/to/parent/dir/')

         iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the data
         Y = iris[:,-1]
         X = iris[:,0:2]#first two features
```

```python
import mltools as ml
#We'll use some data manipulation routines in the provided class code
#Make sure the "mltools" directory is in a directory on your Python path, e.g.,
#export PYTHONPATH=$\$${PYTHONPATH}:/path/to/parent/dir
# or add it to your path inside Python:
np.random.seed(0)
X,Y = ml.shuffleData(X,Y); # shuffle data randomly
# (This is a good idea in case your data are ordered in some pathological way,
# as the Iris data are)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y, 0.75); # split data into 75/25 train/validation
K=[1,5,10,50]
for i in range(len(K)):
    knn = ml.knn.knnClassify() # create the object and train it
    knn.train(Xtr, Ytr, K[i]) # where K is an integer, e.g. 1 for nearest neighbor pr
    YvaHat = knn.predict(Xva) # get estimates of y for each data point in Xva
    ml.plotClassify2D( knn, Xtr, Ytr ); # make 2D classification plot with data (Xtr,
```




4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

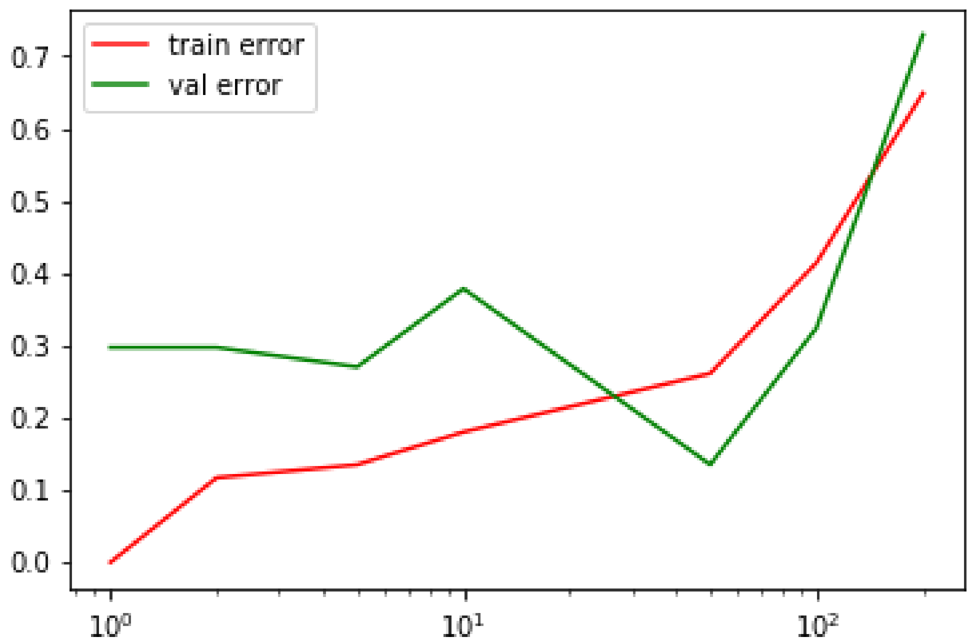


From top to bottom, the pictures are the results of K = 1 , 5 , 10 ,50

2.2 part 2

In [32]:
```
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.append('/path/to/parent/dir/')

iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the data
Y = iris[:,-1]
X = iris[:,0:2]#first two features

import mltools as ml
#We'll use some data manipulation routines in the provided class code
np.random.seed(0)
X,Y = ml.shuffleData(X,Y); # shuffle data randomly
# (This is a good idea in case your data are ordered in some pathological way,
# as the Iris data are)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y, 0.75); # split data into 75/25 train/validation

K=[1,2,5,10,50,100,200]
errTrain = []
errTest = []
for i, k in enumerate(K) :
```

```
        counttr = 0
        countte = 0
        learner = ml.knn.knnClassify(Xtr,Ytr,k) # train model
        Yhattr = learner.predict(Xtr) #  predict results for training Y on training data
        for t in range(len(Ytr)):
            if Yhattr[t]!= Ytr[t] : counttr += 1 # count what fraction of training data p
        errTrain.append(float(counttr)/len(Ytr)) #average
        Yhatte = learner.predict(Xva) #  predict results for validation Y
        for m in range(len(Yva)):
            if Yhatte[m]!=Yva[m] : countte += 1 # count what fraction of validation data f
        errTest.append(float(countte)/len(Yva))#average
    plt.semilogx(K,errTrain,color = 'red',label = 'train error')
    plt.legend()
    plt.semilogx(K,errTest,color = 'green',label = 'val error')
    plt.legend()
    plt.show()
```

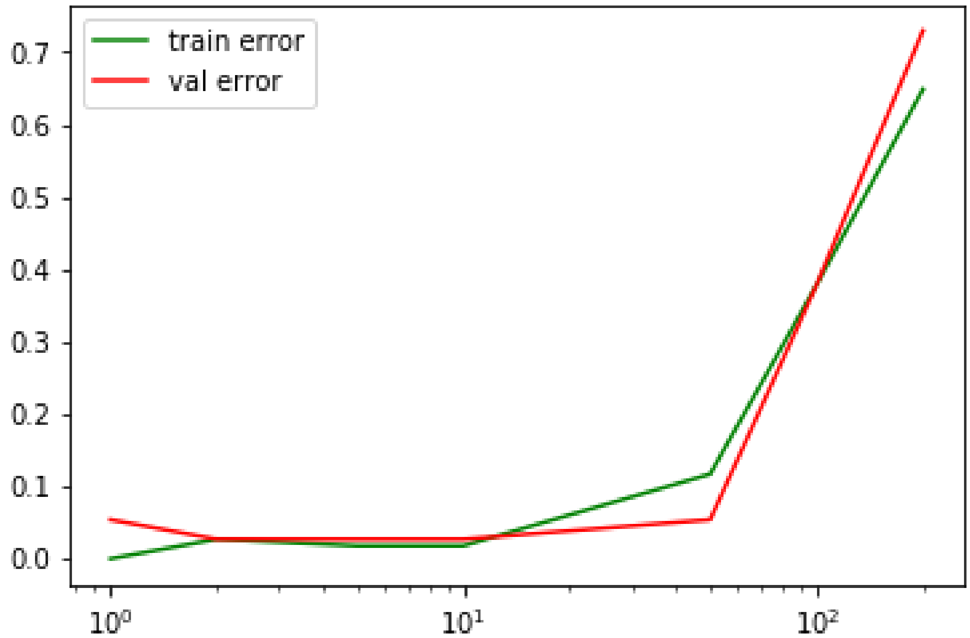


The green one is the error rate for validation data, red one is the error rate for training data. I would recommend K = 50

2.3 part 3

In [35]:
```
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.append('/path/to/parent/dir/')
```

```python
iris = np.genfromtxt("data/iris.txt",delimiter=None) # load the data
Y = iris[:,-1]
X = iris[:,0:-1] #all the features
import mltools as ml
np.random.seed(0)
X,Y = ml.shuffleData(X,Y); # shuffle data randomly
# (This is a good idea in case your data are ordered in some pathological way,
# as the Iris data are)
Xtr,Xva,Ytr,Yva = ml.splitData(X,Y, 0.75); # split data into 75/25 train/validation

K=[1,2,5,10,50,100,200]
errTrain = []
errTest = []
for i, k in enumerate(K) :

    counttr = 0
    countte = 0
    learner = ml.knn.knnClassify(Xtr,Ytr,k) # train model
    Yhattr = learner.predict(Xtr) #  predict results for training Y on training data
    for t in range(len(Ytr)):
        if Yhattr[t]!= Ytr[t] : counttr += 1 # count what fraction of training data p
    errTrain.append(float(counttr)/len(Ytr))
    Yhatte = learner.predict(Xva) #  predict results for validation Y
    for m in range(len(Yva)):
        if Yhatte[m]!=Yva[m] : countte += 1
    errTest.append(float(countte)/len(Yva))
plt.semilogx(K,errTrain,color = 'green',label = 'train error')
plt.legend()
plt.semilogx(K,errTest,color = 'red',label = 'val error')
plt.legend()
plt.show()
```



The plot is different , i would recommand K = 2 / K = 10 this time.

3 Problem 3: Naïve Bayes Classifiers

3.1 part 1

In [36]:
```
import numpy as np
data = np.array([
    [0,0,1,1,0,-1],
    [1,1,0,1,0,-1],
    [0,1,1,1,1,-1],
    [1,1,1,1,0,-1],
    [0,1,0,0,0,-1],
    [1,0,1,1,1,1],
    [0,0,1,0,0,1],
    [1,0,0,0,0,1],
    [1,0,1,1,0,1],
    [1,1,1,1,1,-1],
]
) # the matrix
Y = data[:,-1]
X = data[:,0:5]
count = 0
for i in range(10):
    if Y[i] == 1: count += 1
prob = float(count)/len(Y)
```

```
        print(prob,1-prob) #p(y=1) and p(y=-1)
        result = np.zeros((5,4),dtype = float)
        for j in range(5) :
            for i in range(10):
                if X[i][j] == 0 and Y[i] == -1 : result[j][0] += 1
                if X[i][j] == 1 and Y[i] == -1 : result[j][1] += 1
                if X[i][j] == 0 and Y[i] == 1 : result[j][2] += 1
                if X[i][j] == 1 and Y[i] == 1 : result[j][3] += 1

        for k in range(5) :
            result[k][0] /= ((1-prob))
            result[k][1] /= ((1-prob))
            result[k][2] /= (prob)
            result[k][3] /= (prob)
        print(result/10)
```

```
0.4 0.6
[[0.5        0.5        0.25       0.75      ]
 [0.16666667 0.83333333 1.         0.        ]
 [0.33333333 0.66666667 0.25       0.75      ]
 [0.16666667 0.83333333 0.5        0.5       ]
 [0.66666667 0.33333333 0.75       0.25      ]]
```

class y probabilities : p(y=1) = 0.4 p(y=-1) =0.6
feature probabilities :
p(x1=0|y=-1) = 0.5 p(x1=1|y=-1) = 0.5 p(x1=0|y=1) = 0.25 p(x1=1|y=1) = 0.75
p(x2=0|y=-1) = 0.167 p(x2=1|y=-1) = 0.833 p(x2=0|y=1) = 1 p(x2=1|y=1) = 0
p(x3=0|y=-1) = 0.333 p(x3=1|y=-1) = 0.667 p(x3=0|y=1) = 0.25 p(x1=3|y=1) = 0.75
p(x4=0|y=-1) = 0.167 p(x4=1|y=-1) = 0.833 p(x4=0|y=1) = 0.5 p(x4=1|y=1) = 0.5
p(x5=0|y=-1) = 0.667 p(x5=1|y=-1) = 0.333 p(x5=0|y=1) = 0.75 p(x5=1|y=1) = 0.25

3.2 part 2

```
In [37]: import numpy as np
         feaprob = np.array([
          [ 0.5 ,0.5,0.25,0.75],
          [ 0.16666667,0.83333333,1,0],
          [ 0.33333333,0.66666667,0.25,0.75],
          [ 0.16666667,0.83333333,0.5,0.5],
          [ 0.66666667,0.33333333,0.75,0.25]
          ],dtype = float)
         ypro = 0.4
         input = np.array([
             [0,0,0,0,0],
             [1,1,0,1,0]
             ])
         for i in range(2):
```

```
            f1 = 1.0
            f2 = 1.0
            for j in range(5):
                if input[i][j] == 1 :
                    f1 *= feaprob[j][3]
                    f2 *= feaprob[j][1]
                if input[i][j] == 0 :
                    f1 *= feaprob[j][2]
                    f2 *= feaprob[j][0]
            prb = (0.4*f1)/(0.6*f2+0.4*f1)
            print(prb,1-prb)
```

```
0.8350515415708365 0.1649484584291635
0.0 1.0
```

when x=(0,0,0,0,0) , p(y=1) =0.835 p(y=-1) = 0.165 , so it belongs to class y = 1 ;
when x=(1,1,0,1,0) , p(y=1) =0 p(y=-1) = 1 , so it belongs to class y = -1 ;

3.3 part 3

Since there is no situation x= (1,1,0,1,0) while y = 1 , so p(x=(1 1 0 1 0), y = 1) = 0 ; so the posterior probability for p(x=(1 1 0 1 0) | y = 1) = 0 .

3.4 part 4

There are 5 features in this model . Each feature has two possible values 0 or 1 that represent different meanings. It needs more than 2ˆ5 =32 groups of input x=(x1 x2 x3 x4 x5) to conclude all possible situations, but we are only given 10 different situatons . We don't have enough data to give an accurate probability by "joint" Bayes Classifiers which needs a huge amount data to obtain enough accuracy level.

3.5 part 5

We don't need to re-train the model :

Since there is independency in P(x1 x2 x3 x4 . . . . . . | y = a) means it equals P(x1 | y =a)* P(x2 | y = a)* P(x3| y =a) . . . . . . The probabilities of each xi features are not affected by others. They are independent. If the data of x1 is missing in our model, the probabilities of x2 x3 x4 x5 will still be the same. Just erase the x1 charactors in computing procedure.

4 Statement of Collaboration

I obey all the rules of UCI academic integrity and finish the project only by my own. Ziyang Zhang 7/10/2018