

HW1-Kai Ye

October 9, 2017

1 Problem 1

1.1 1.

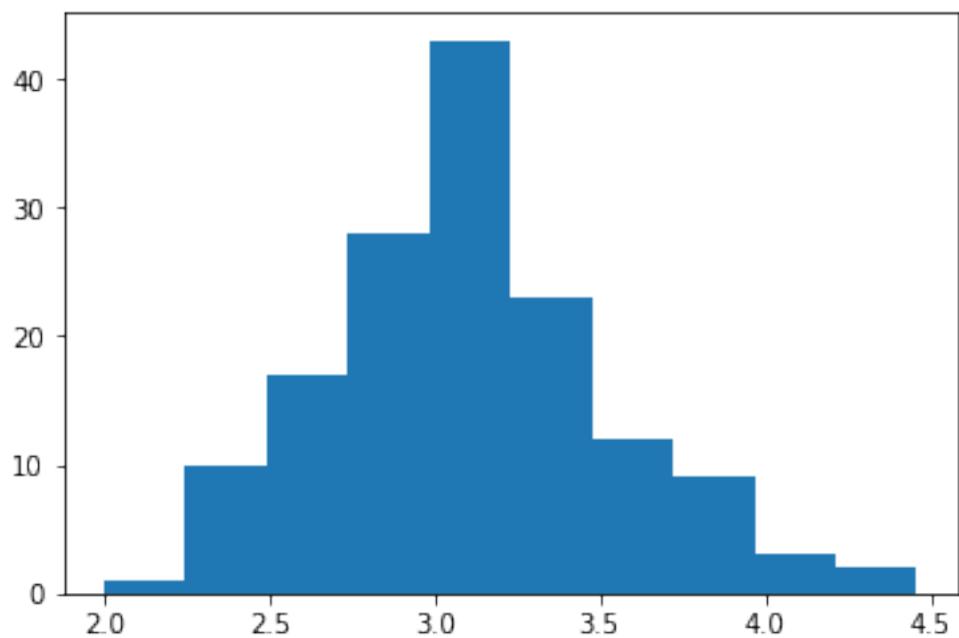
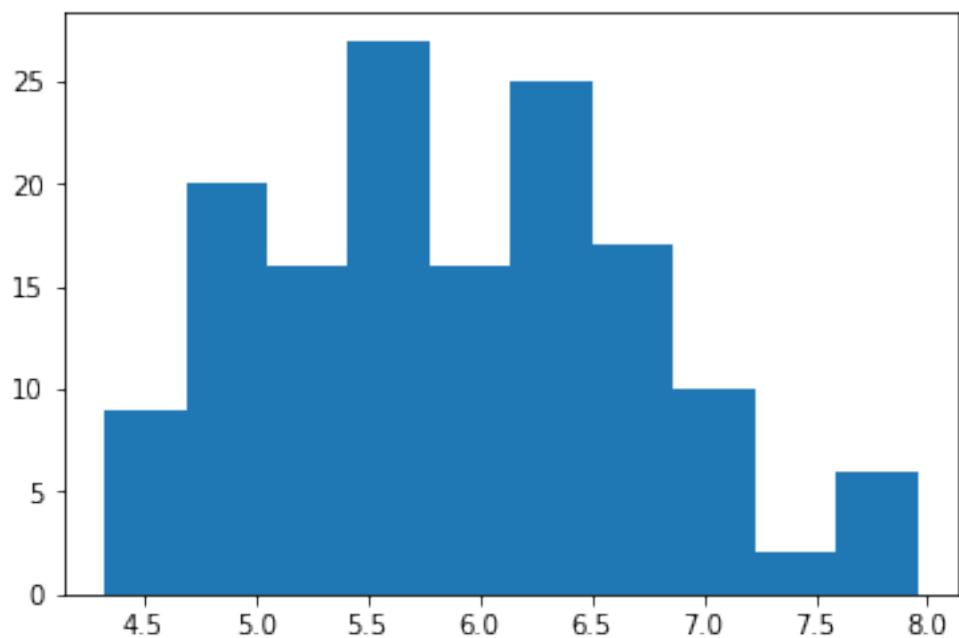
```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:-1]
        print(X.shape)
```

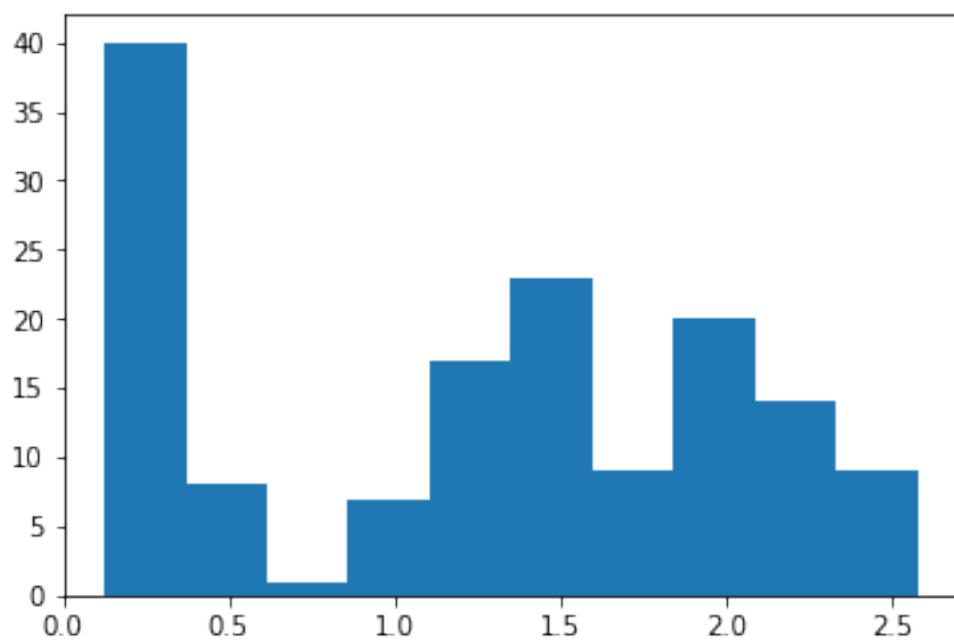
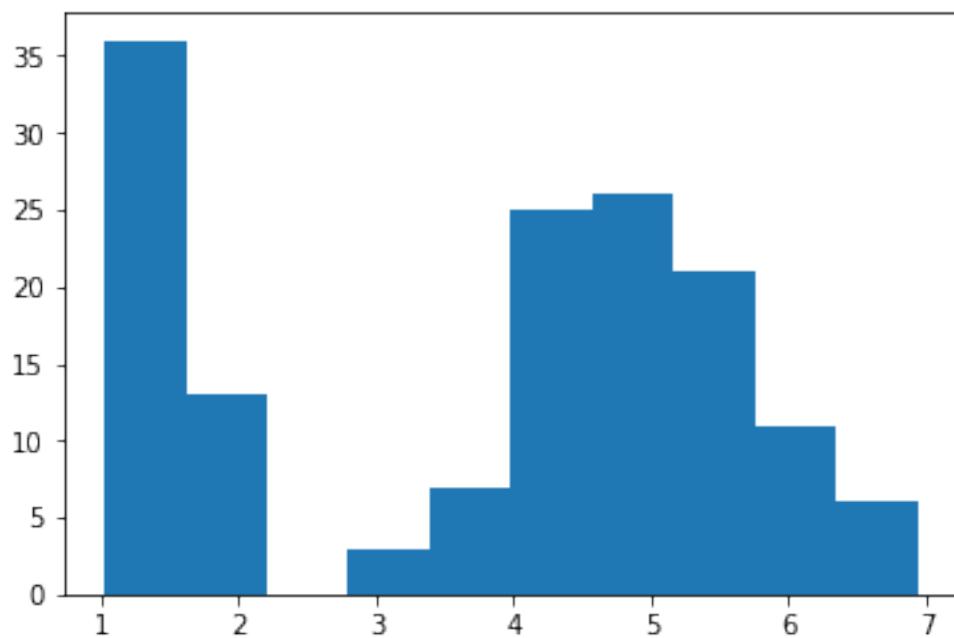
(148, 4)

From the result we know there are 148 data points and 4 features. We can also know that from the data file, the first four columns are features and the last one is target.

1.2 2.

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:-1]
        num=[0,1,2,3]
        for i in num:
            plt.hist(iris[:,i:i+1])
            plt.show()
```





using the codes above, I get 4 histograms. We can also know the feature values and frequencies from the x and y axes.

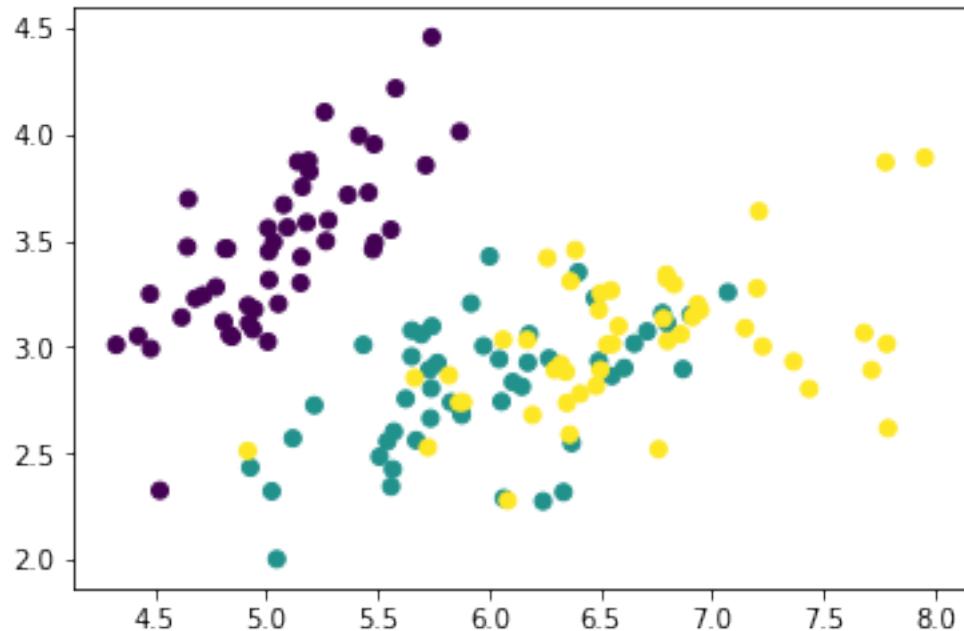
1.3 3.

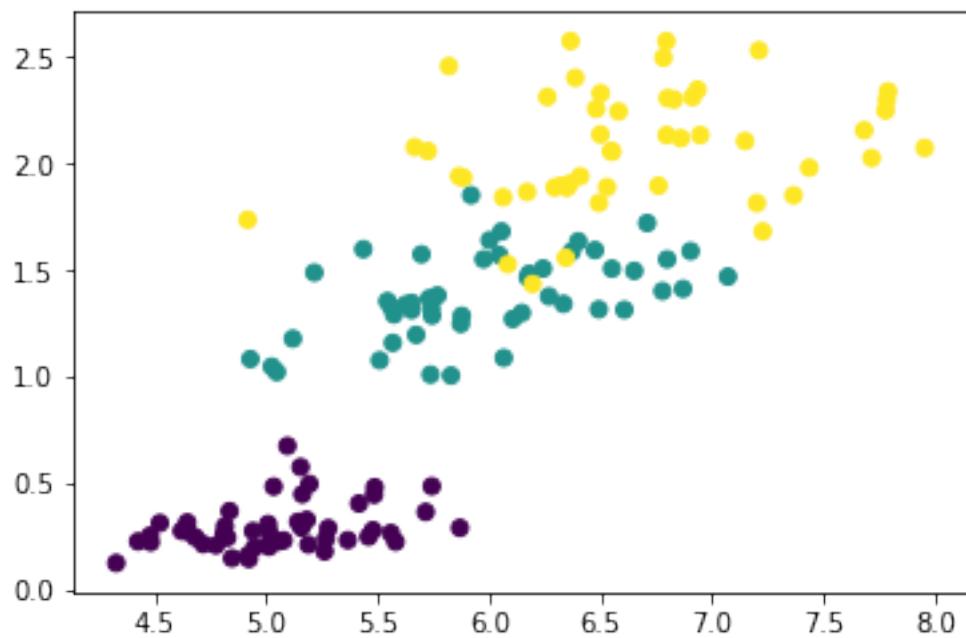
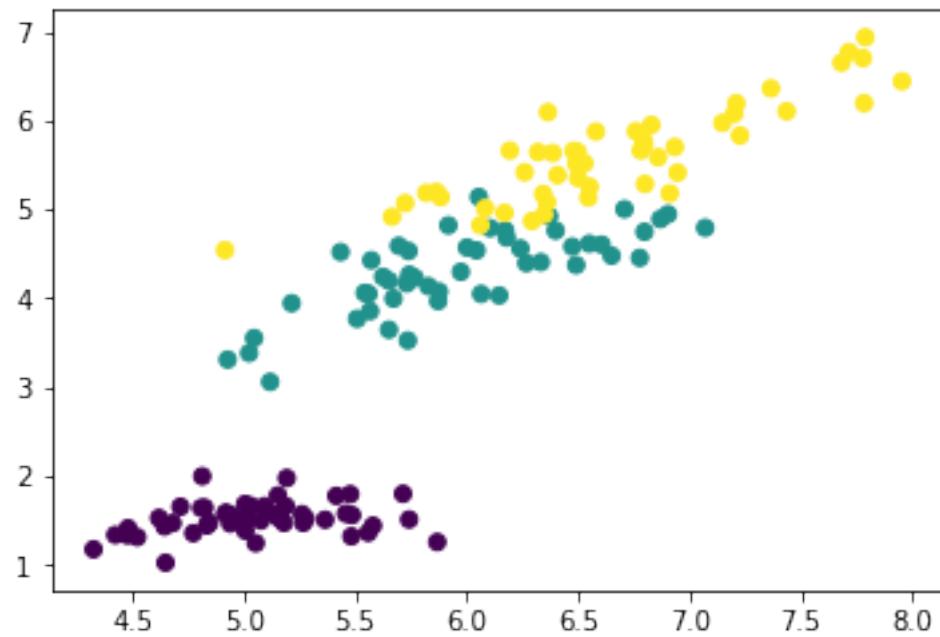
```
In [5]: import numpy as np
        import matplotlib.pyplot as plt
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:-1]
        a=np.array(X)
        print(np.mean(a, axis=0))
        print(np.std(a, axis=0))
```

```
[ 5.90010376  3.09893092  3.81955484  1.25255548]
[ 0.83340207  0.43629184  1.75405711  0.75877246]
```

1.4 4.

```
In [6]: import numpy as np
        import matplotlib.pyplot as plt
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:-1]
        plt.scatter(iris[:, 0],iris[:, 1],c=Y)
        plt.show()
        plt.scatter(iris[:, 0],iris[:, 2],c=Y)
        plt.show()
        plt.scatter(iris[:, 0],iris[:, 3],c=Y)
        plt.show()
```



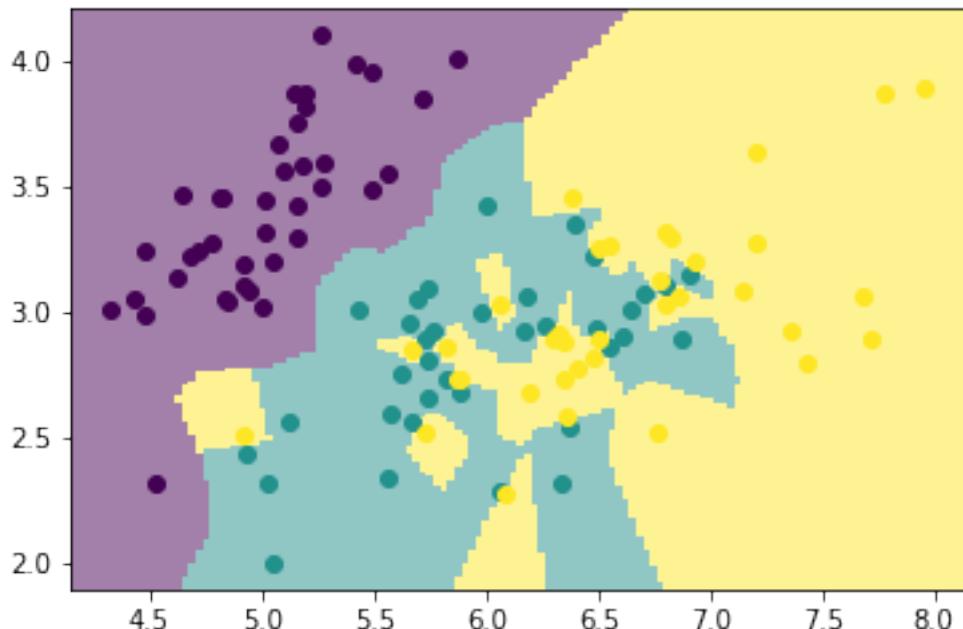


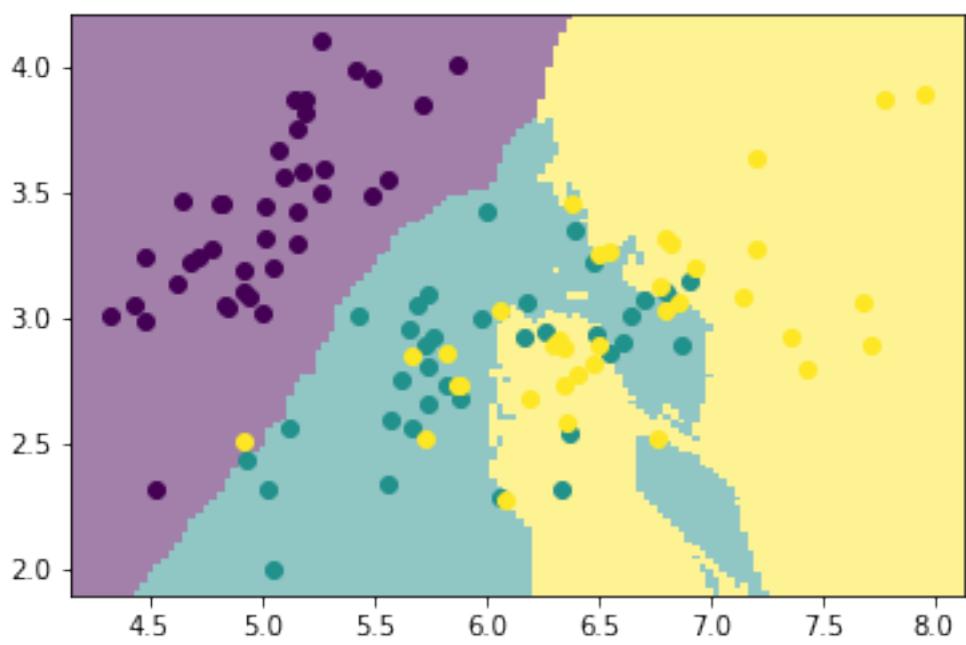
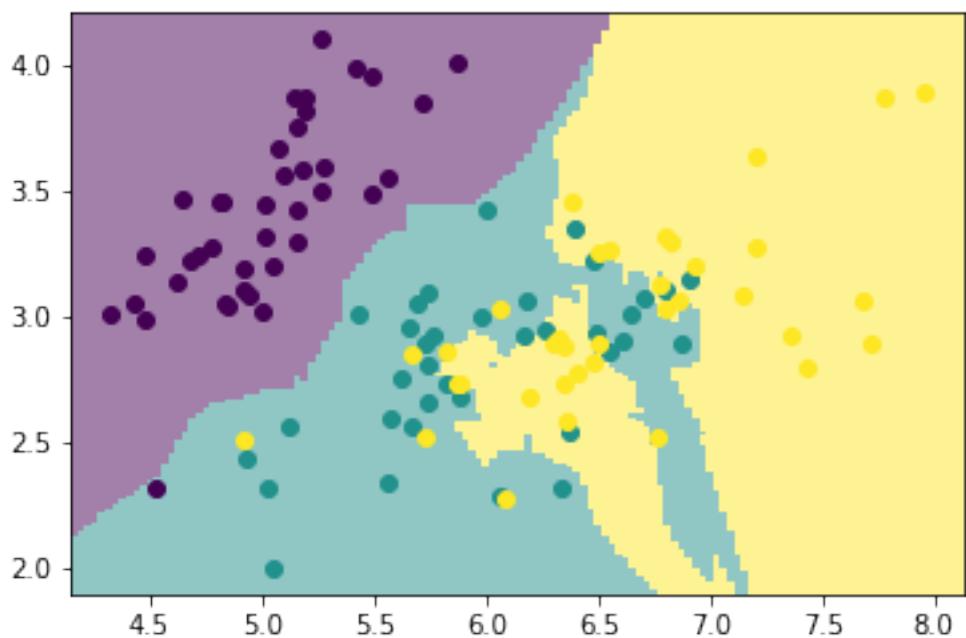
In []: In the scatterplots above, dark red means $y=0$, green means $y=1$ and yellow means $y=2$.

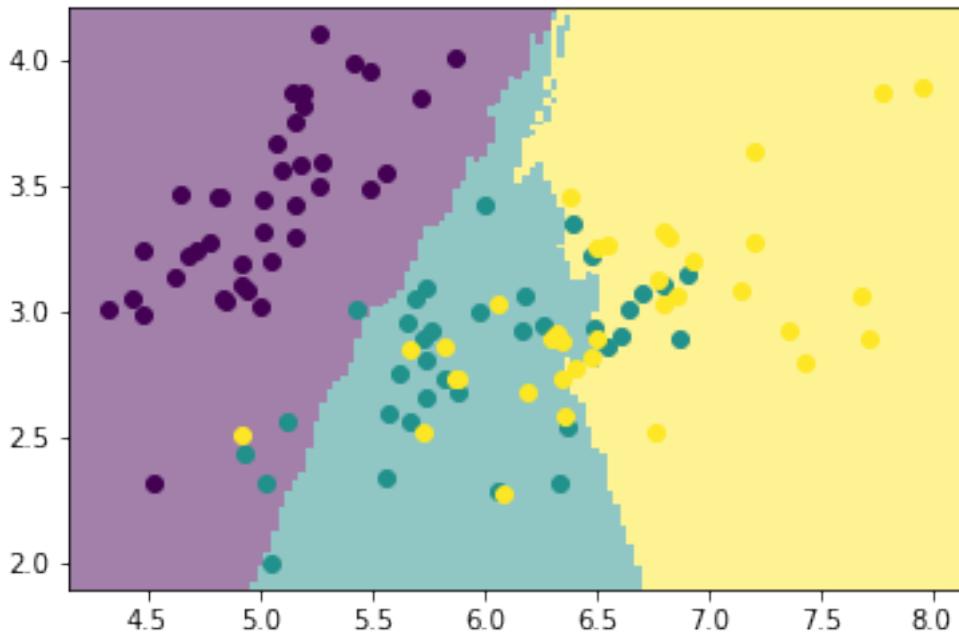
2 Problem 2

2.1 1.

```
In [8]: import numpy as np
        import matplotlib.pyplot as plt
        import mltools as ml
        np.random.seed(0)
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:2]
        X,Y=ml.shuffleData(X,Y)
        Xtr,Xva,Ytr,Yva=ml.splitData(X,Y,0.75)
        K=[1,5,10,50]
        knn=ml.knn.knnClassify()
        for i in K:
            knn.train(Xtr,Ytr,i)
            YvaHat=knn.predict(Xva)
            ml.plotClassify2D(knn,Xtr,Ytr)
            plt.show()
```



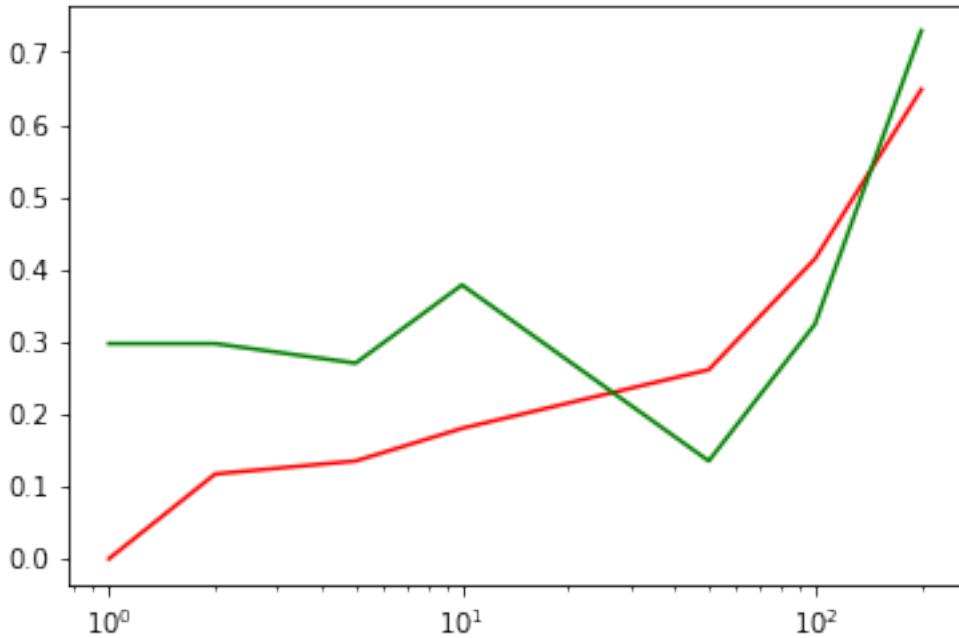




In []: From the scatterplots above, we can see that as K grows bigger, the decision boundary becomes more complex.

2.2 2.

```
In [10]: import numpy as np
        import matplotlib.pyplot as plt
        import mltools as ml
        np.random.seed(0)
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:2]
        X,Y=ml.shuffleData(X,Y)
        Xtr,Xva,Ytr,Yva=ml.splitData(X,Y,0.75)
        K=[1,2,5,10,50,100,200]
        errTrain=[0,0,0,0,0,0,0]
        errVal=[0,0,0,0,0,0,0]
        for i,k in enumerate(K):
            learner=ml.knn.knnClassify()
            learner.train(Xtr,Ytr,k)
            Yhat=learner.predict(Xtr)
            errTrain[i]=np.mean(Yhat!=Ytr)
            Yhatt = learner.predict(Xva)
            errVal[i]=np.mean(Yhatt!=Yva)
        plt.semilogx(K,errTrain,'r')
        plt.semilogx(K,errVal,'g')
        plt.show()
```

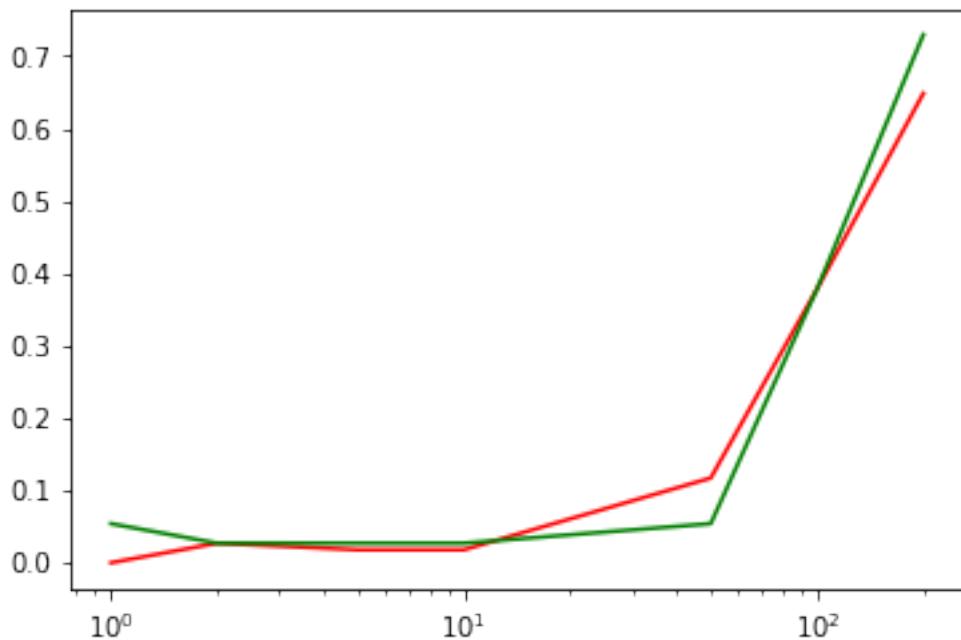


In []: From the plots above, we know that green line(error rate of validation data) has the lowest error rate **is** the smallest at that point, that means we get the model which can make precise prediction.

2.3 3.

```
In [12]: import numpy as np
        import matplotlib.pyplot as plt
        import mltools as ml
        np.random.seed(0)
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:, -1]
        X=iris[:, 0:-1]
        X,Y=ml.shuffleData(X,Y)
        Xtr,Xva,Ytr,Yva=ml.splitData(X,Y,0.75)
        K=[1,2,5,10,50,100,200]
        errTrain=[0,0,0,0,0,0,0]
        errVal=[0,0,0,0,0,0,0]
        for i,k in enumerate(K):
            learner=ml.knn.knnClassify()
            learner.train(Xtr,Ytr,k)
            Yhat=learner.predict(Xtr)
            errTrain[i]=np.mean(Yhat!=Ytr)
            Yhatt = learner.predict(Xva)
            errVal[i]=np.mean(Yhatt!=Yva)
        plt.semilogx(K,errTrain,'r')
```

```
plt.semilogx(K,errVal, 'g')
plt.show()
print(errVal[0],errVal[1],errVal[2],errVal[3],errVal[4],errVal[5],errVal[6])
```



```
0.0540540540541 0.027027027027 0.027027027027 0.027027027027 0.0540540540541 0.378378378378 0.75
```

In []: From the plots above, we can see that the red line changed little, but the green line changed a lot. The value of the points which K=1,2,5,10...etc, so I print the error rate of them. From the figure, we can see that the error rate of K=10 is the same as K=20. Because we have 148 data points, so I think K=2 or K=5 is too small for this problem. I recommand K=10.

3 Problem 3

In [1]: `from IPython.display import Image
Image("1.JPG")`

Out[1] :

$$\begin{aligned}
 1. \quad P(y = -1) &= \frac{6}{10} & P(y = 1) &= \frac{4}{10} \\
 P(X_1 = 0 | y = -1) &= \frac{3}{6} & P(X_1 = 1 | y = -1) &= \frac{3}{6} \\
 P(X_2 = 0 | y = -1) &= \frac{1}{6} & P(X_2 = 1 | y = -1) &= \frac{5}{6} \\
 P(X_3 = 0 | y = -1) &= \frac{2}{6} & P(X_3 = 1 | y = -1) &= \frac{4}{6} \\
 P(X_4 = 0 | y = -1) &= \frac{1}{6} & P(X_4 = 1 | y = -1) &= \frac{5}{6} \\
 P(X_5 = 0 | y = -1) &= \frac{4}{6} & P(X_5 = 1 | y = -1) &= \frac{2}{6} \\
 P(X_1 = 0 | y = 1) &= \frac{1}{4} & P(X_1 = 1 | y = 1) &= \frac{3}{4} \\
 P(X_2 = 0 | y = 1) &= 1 & P(X_2 = 1 | y = 1) &= 0 \\
 P(X_3 = 0 | y = 1) &= \frac{1}{4} & P(X_3 = 1 | y = 1) &= \frac{3}{4} \\
 P(X_4 = 0 | y = 1) &= \frac{2}{4} & P(X_4 = 1 | y = 1) &= \frac{3}{4} \\
 P(X_5 = 0 | y = 1) &= \frac{3}{4} & P(X_5 = 1 | y = 1) &= \frac{1}{4}
 \end{aligned}$$

```
In [2]: from IPython.display import Image
Image("2.JPG")
```

Out [2] :

$$\begin{aligned}
 2. P(y=1 | X=00000) &= \frac{P(y=1, X=00000)}{P(X=00000)} \\
 &\stackrel{\text{mutually independent given } X}{=} P(y=1) \cdot P(X=00000 | y=1) \\
 &= \frac{\frac{6}{10} \times \frac{3}{6} \times \frac{1}{6} \times \frac{2}{6} \times \frac{1}{6} \times \frac{4}{6}}{\frac{6}{10} \times \frac{3}{6} \times \frac{1}{6} \times \frac{2}{6} \times \frac{1}{6} \times \frac{4}{6} + \frac{4}{10} \times \frac{1}{4} \times \frac{1}{4} \times \frac{3}{4} \times \frac{3}{4}} \\
 &= \frac{\frac{1}{15} \times \frac{1}{36}}{\frac{1}{15} \times \frac{1}{36} + \frac{1}{10} \times \frac{1}{4} \times \frac{1}{2} \times \frac{3}{4}} \\
 &= \frac{\frac{1}{540}}{\frac{1}{540} + \frac{3}{320}} \\
 &= \frac{16}{97}
 \end{aligned}$$

In [6]: `from IPython.display import Image
Image("22.JPG")`

Out [6] :

$$P(y=1 | X=000000) = 1 - \frac{16}{97} = \frac{81}{97}$$

So $y=1$ should be predicted for $x=(000000)$

$$P(y=-1 | X=11010) = \frac{P(y=-1, X=11010)}{P(X=11010)}$$

$$= \frac{P(y=-1) \cdot P(X=11010 | y=-1)}{P(y=-1) \cdot P(X=11010 | y=-1) + P(y=1) \cdot P(X=11010 | y=1)}$$

$$= \frac{\frac{6}{10} \times \frac{3}{6} \times \frac{5}{6} \times \frac{2}{6} \times \frac{5}{6} \times \frac{4}{6}}{\frac{6}{10} \times \frac{3}{6} \times \frac{5}{6} \times \frac{2}{6} \times \frac{5}{6} \times \frac{4}{6} + \frac{4}{10} \times \frac{3}{4} \times 0 \times \frac{1}{4} \times \frac{3}{4} \times \frac{2}{4}}$$

$$= \frac{1}{10} = 0.1$$

$P(y=1 | X=11010) = 1 - 0.1 = 0.9$.

So $y=-1$ should be predicted for $X=(11010)$

```
In [3]: from IPython.display import Image
Image("3.JPG")
```

Out [3] :

3. As we mentioned in question 2, the probability that $y=+1$ given $X=(11010)$ is 0.

```
In [4]: from IPython.display import Image  
Image("4.JPG")
```

Out[4] :

4. Because in "joint" Bayes classifier, the features may be related. If there are n features, there will be 2^n parameters, which is a huge number. In this problem, we have 5 features, so we may need about $2^5 = 32$ parameters, naive Bayes classifier reduce the model complexity, so in this case we just need 11 parameters, as the features are independent. So the model is simpler and we can use a smaller amount of data instead of a huge number.

```
In [5]: from IPython.display import Image  
Image("5.JPG")
```

Out[5] :

5. We should re-train the model with $x_2 \dots x_5$, 4 features. The new model will be like:

$$P(y|X_2=x_2, X_3=x_3, X_4=x_4, X_5=x_5) = \frac{P(y, X_2=x_2, X_3=x_3, X_4=x_4, X_5=x_5)}{P(X_2=x_2, X_3=x_3, X_4=x_4, X_5=x_5)}$$

We will need 9 parameters :

$$\begin{aligned} & P(y=1), P(X_2=1, y=1), P(X_2=1, y=-1), P(X_3=1, y=1), \\ & P(X_3=1, y=-1), P(X_4=1, y=1), P(X_4=1, y=-1), P(X_5=1, y=1), \\ & P(X_5=1, y=-1). \end{aligned}$$

4 Statement of Collaboration

In []: No discussion **with** others.