HW1-Kai Ye

October 9, 2017

1 Problem 1

1.1 1.

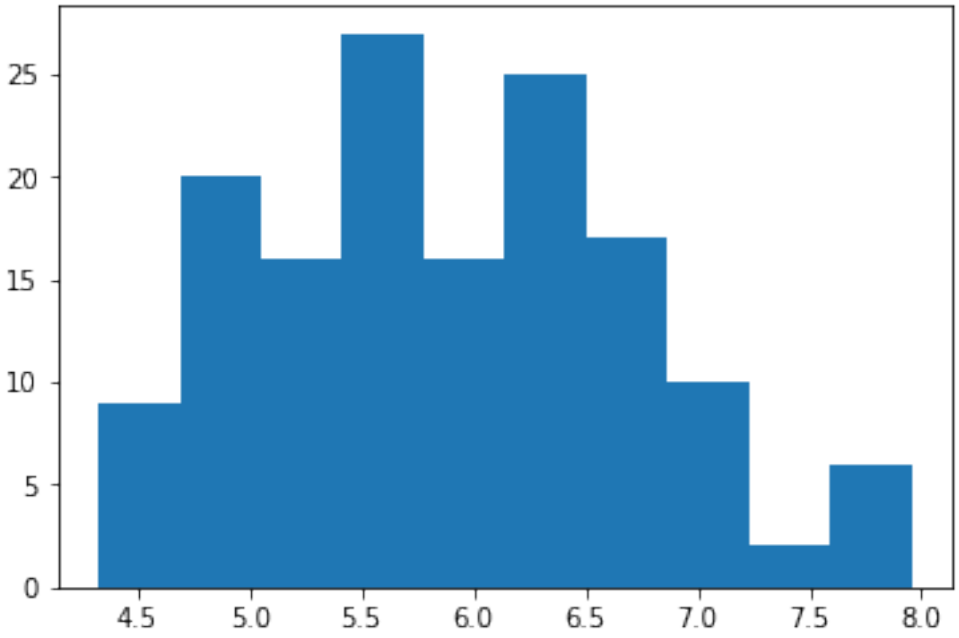
```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:,-1]
        X=iris[:,0:-1]
        print(X.shape)
```

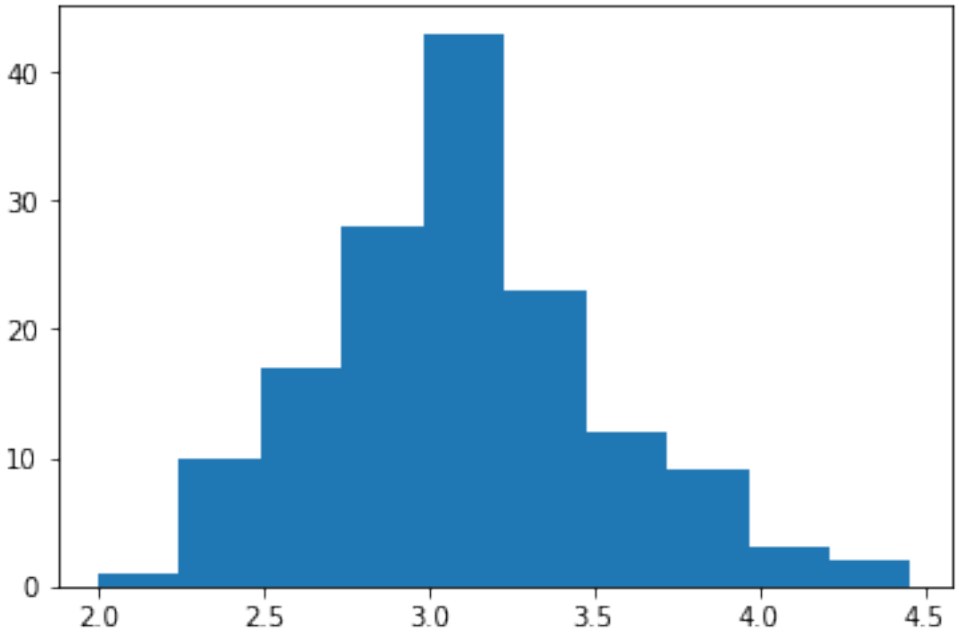
```
(148, 4)
```

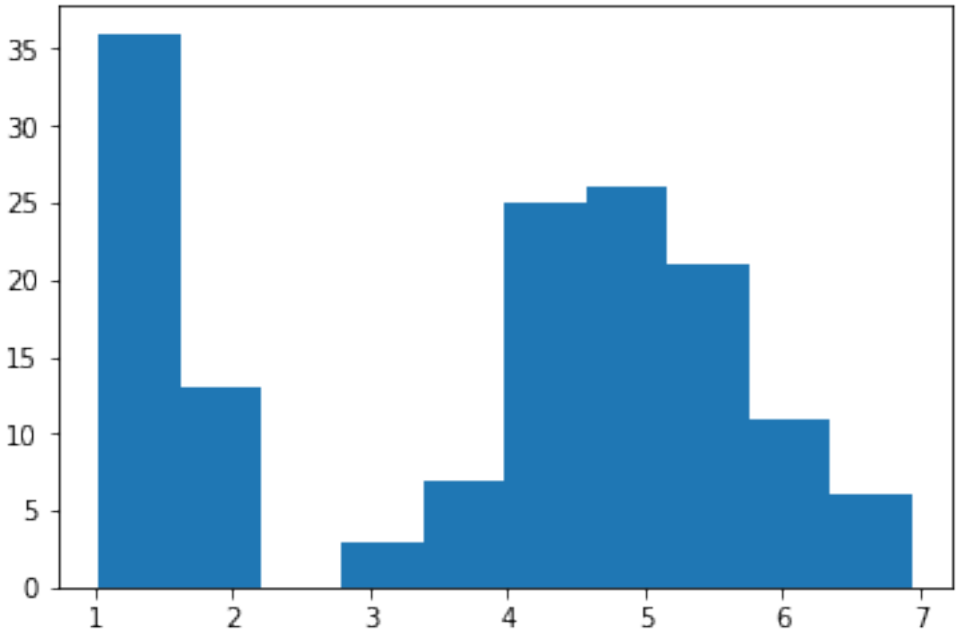
From the result we know there are 148 data points and 4 features. We can also know that from the data file, the first four columns are features and the last one is target.

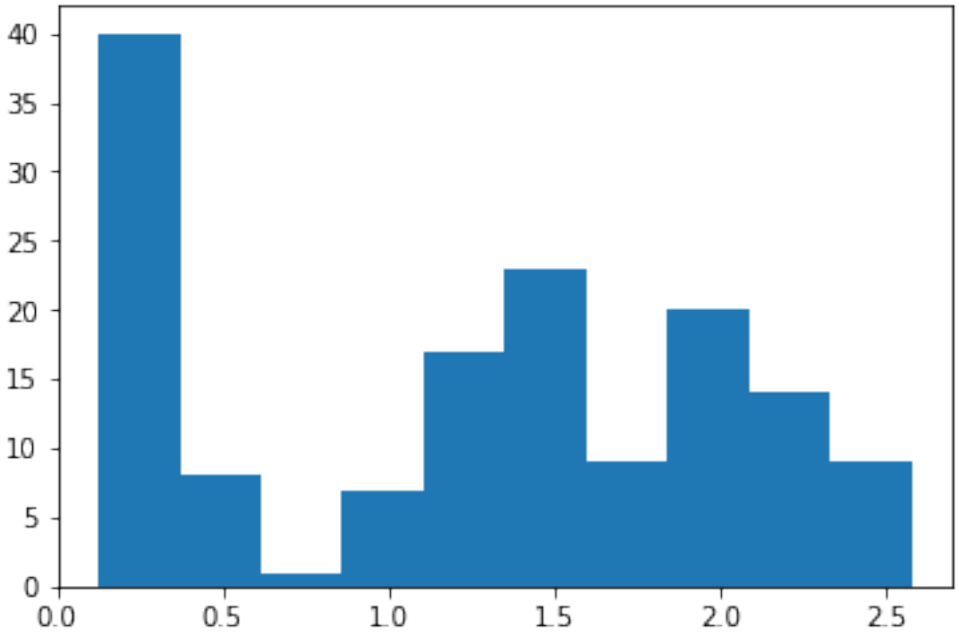
1.2 2.

```
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()
```


25
20
15
10
5
0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


40
30
20
10
0
2.0
2.5
3.0
3.5
4.0
4.5





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

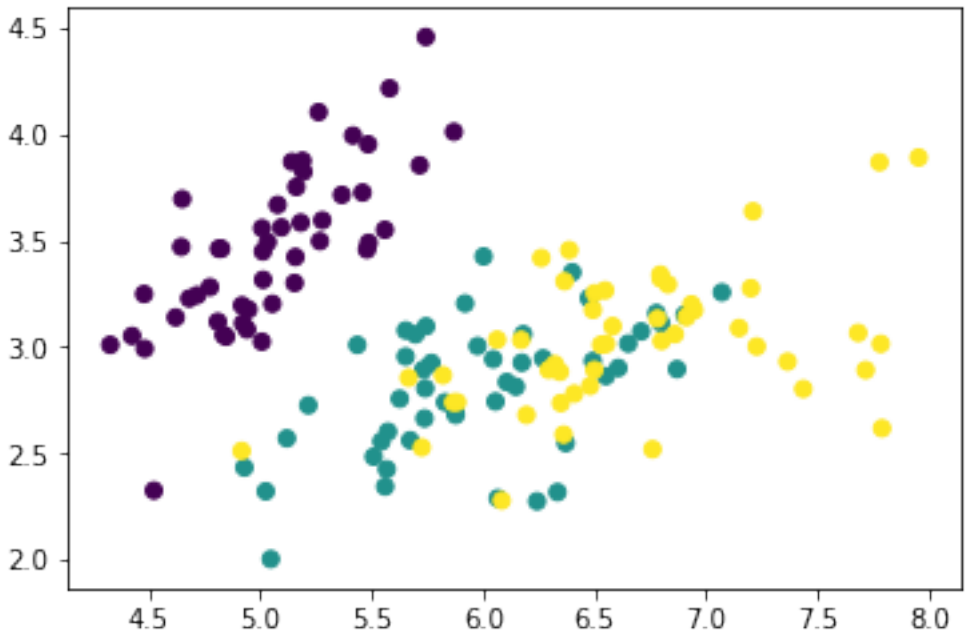
1.3 3.

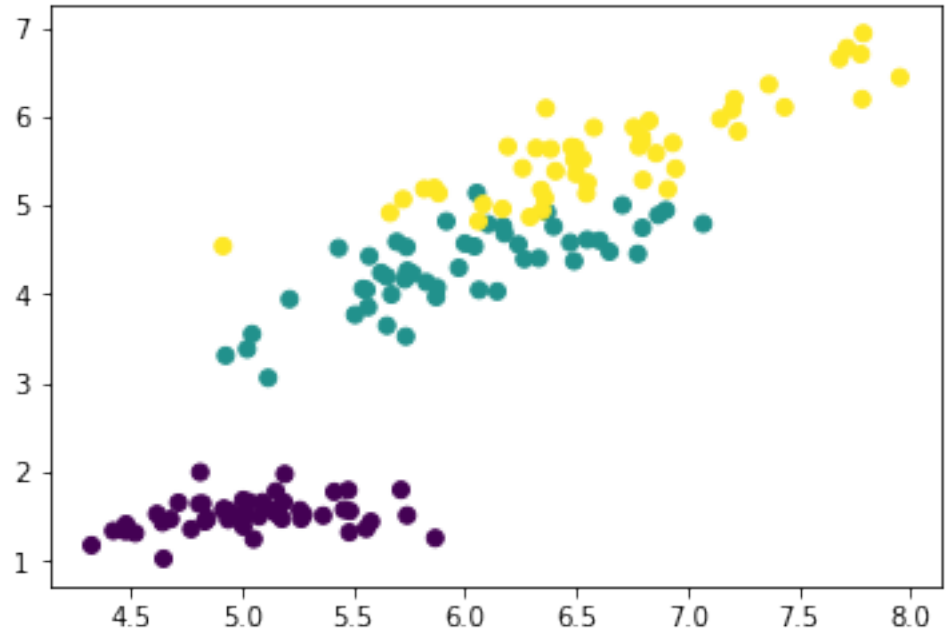
```
In [5]: import numpy as np
        import matplotlib.pyplot as plt
        iris=np.genfromtxt("data/iris.txt",delimiter=None)
        Y=iris[:,-1]
        X=iris[:,0:-1]
        a=np.array(X)
        print(np.mean(a,axis=0))
        print(np.std(a,axis=0))
```

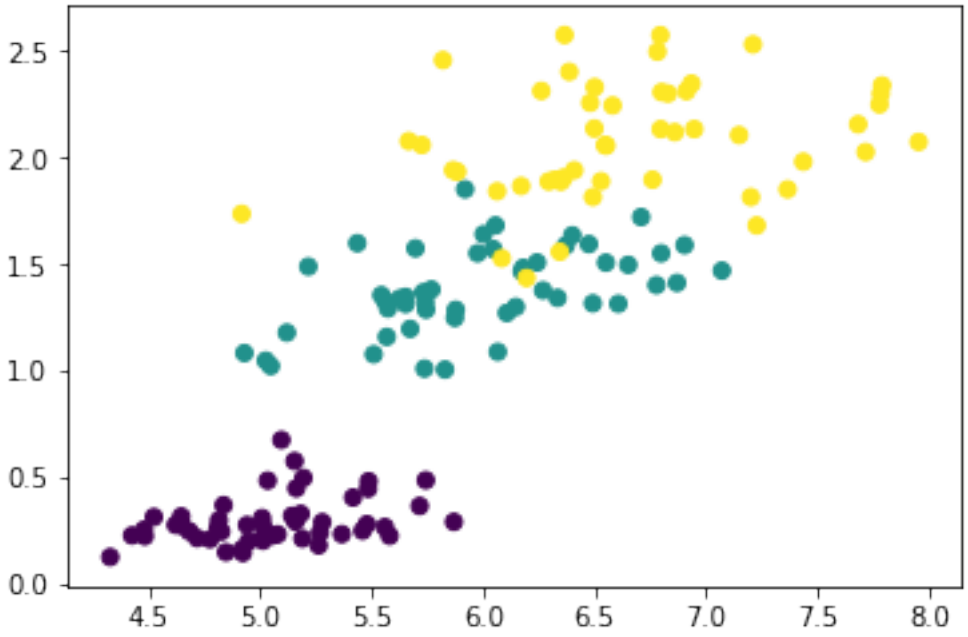
```
[ 5.90010376  3.09893092  3.81955484  1.25255548]
[ 0.83340207  0.43629184  1.75405711  0.75877246]
```

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()
```





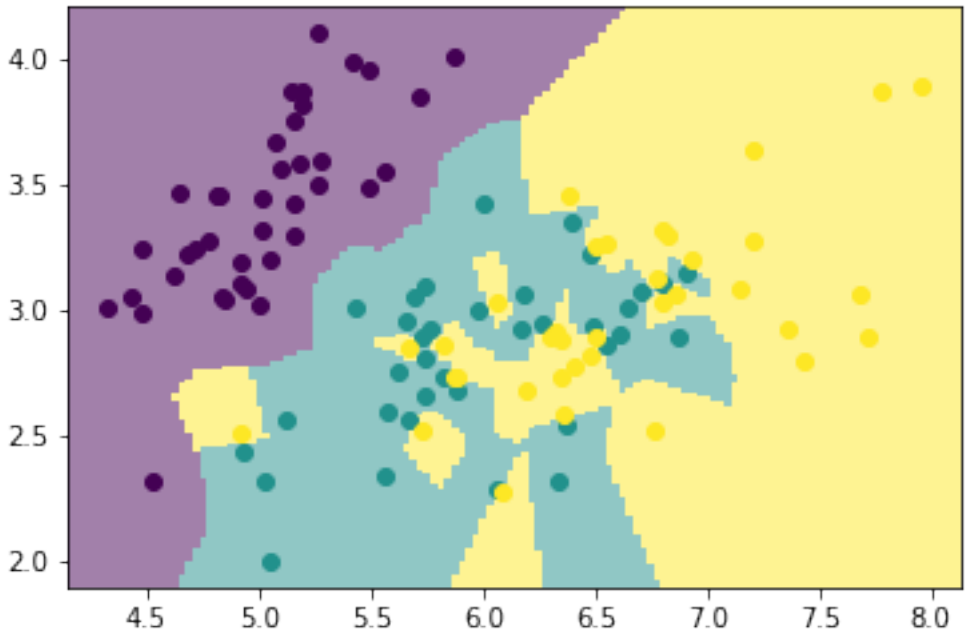


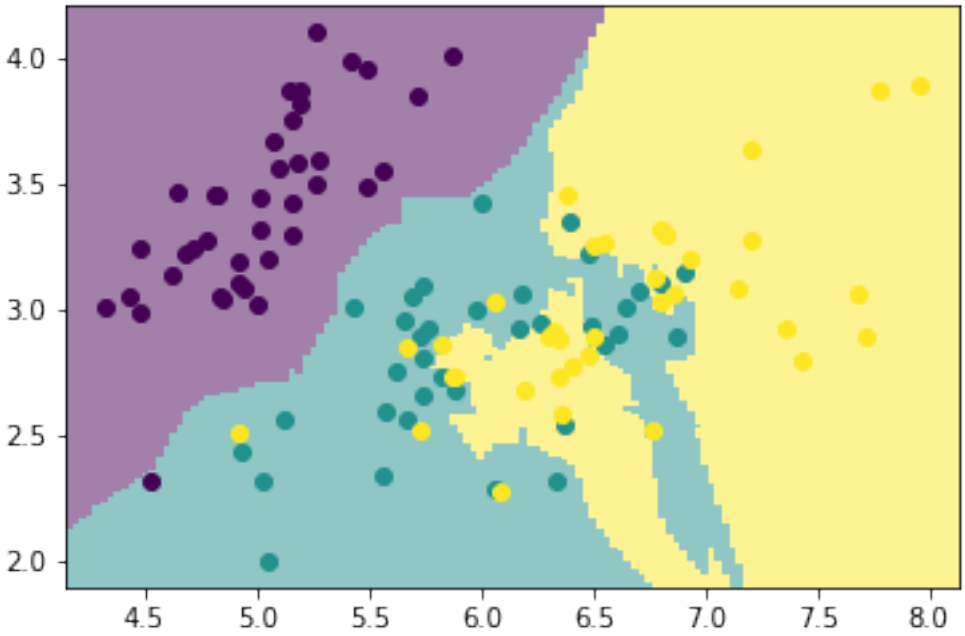
In [ ]: In the scatterplots above, dark red means y=0, green means y=1 and yellow means y=2.

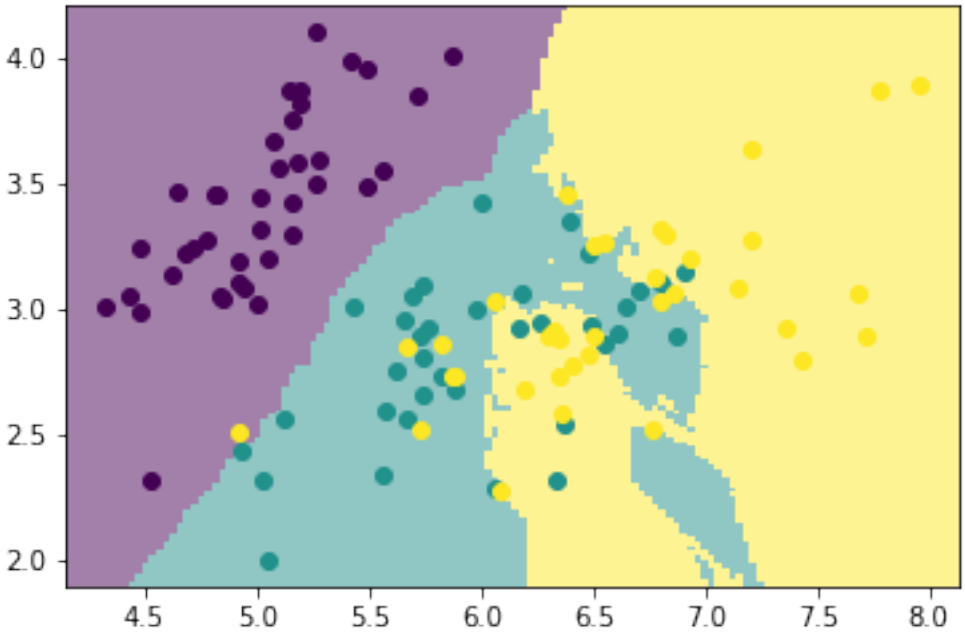
2 Problem 2

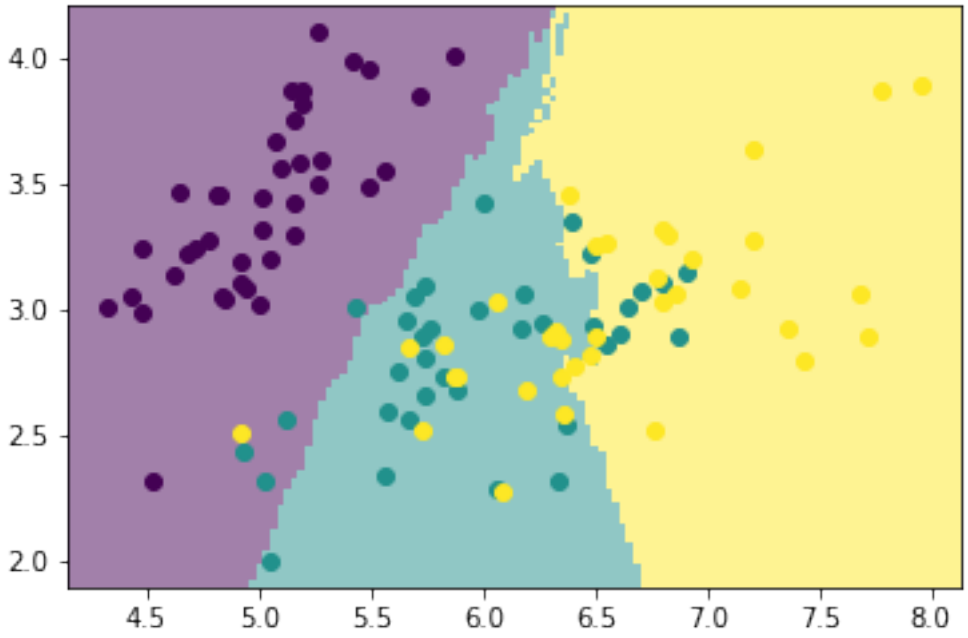
2.1 1.

```
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()
```




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

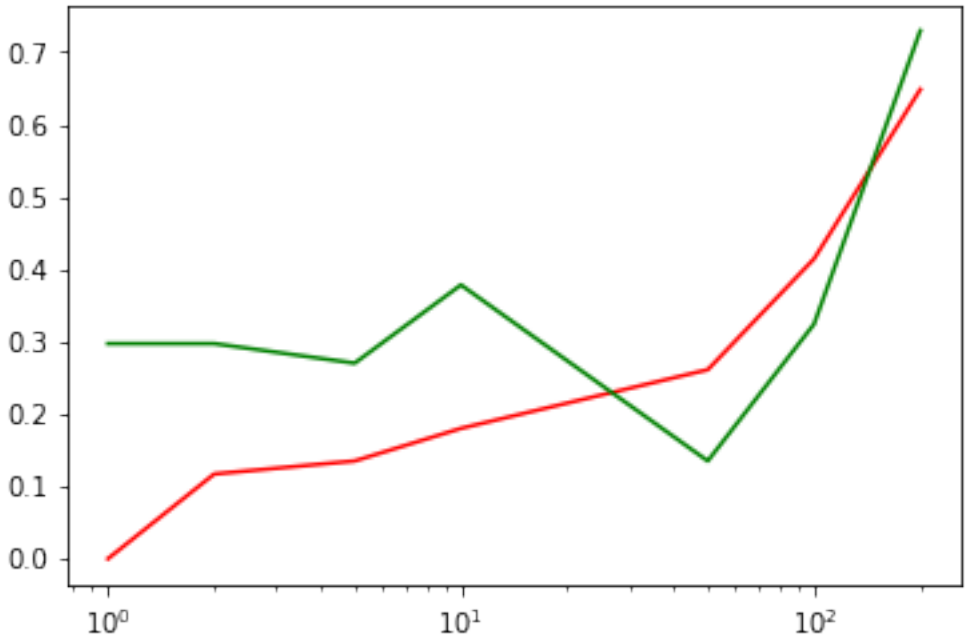

4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



In [ ]: From the scatterplots above, we can see that as K grows bigger, the decision boundary l

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()
```

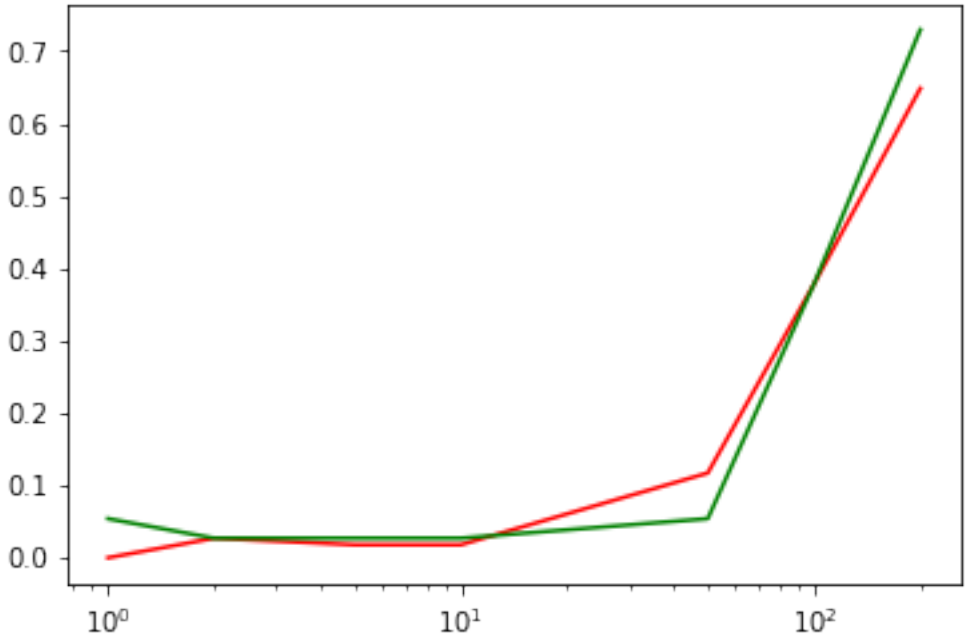


In [ ]: From the plots above, we know that green line(error rate of validation data) has the lo
rate is the smallest at that point, that means we get the model which can make precise

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])
```



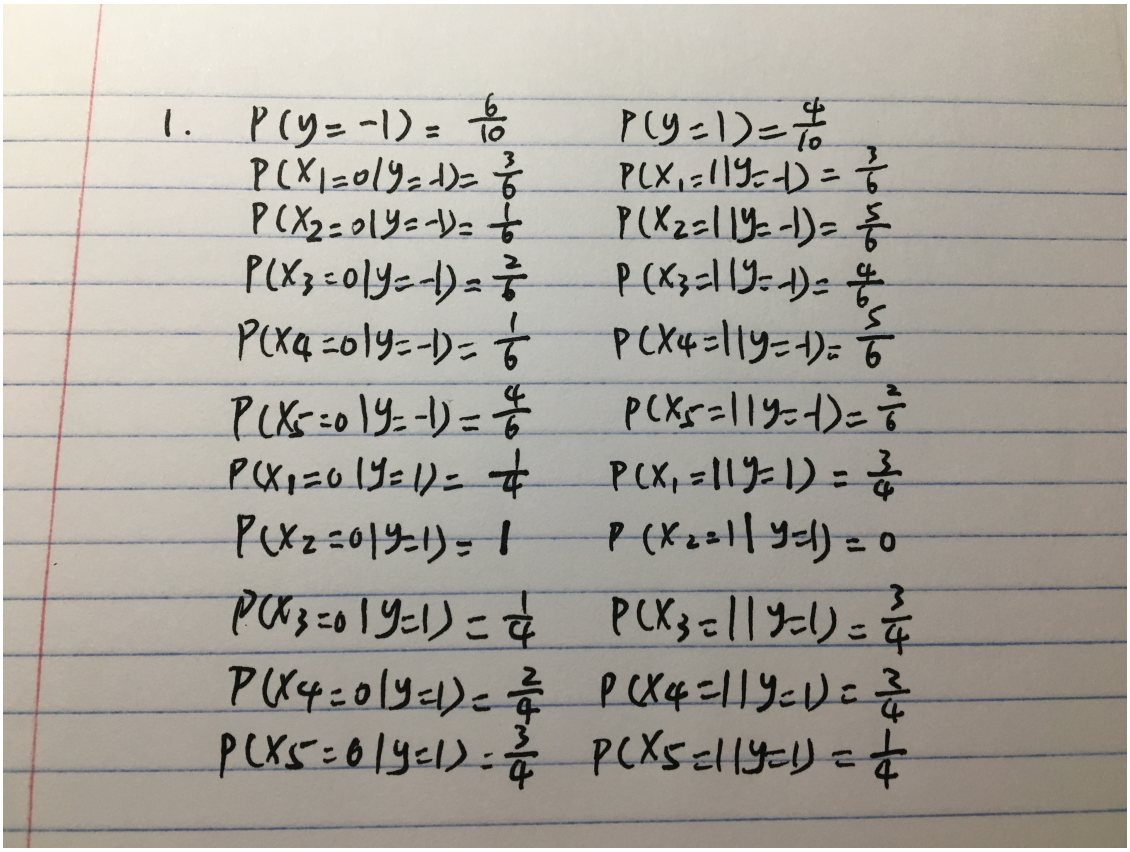

```
0.0540540540541 0.027027027027 0.027027027027 0.027027027027 0.0540540540541 0.378378378378 0.
```

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

3 Problem 3

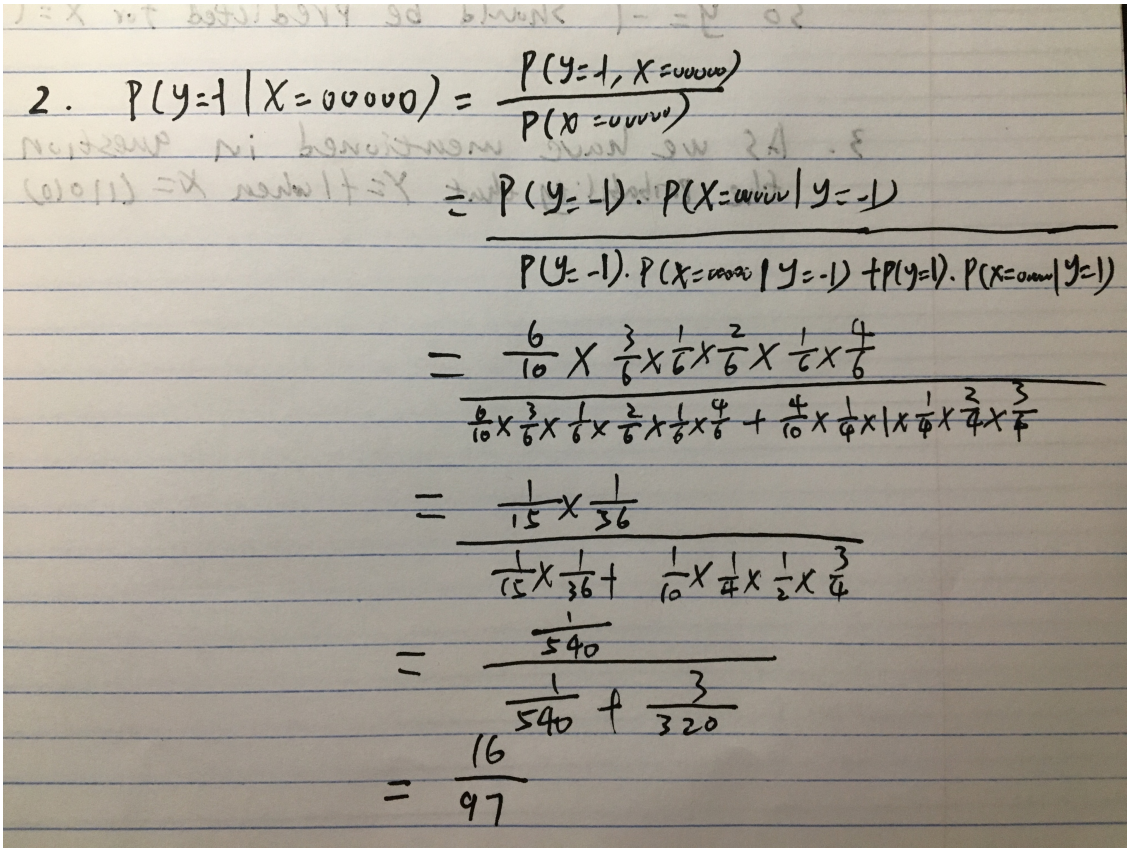
In [1]:
```
from IPython.display import Image
Image("1.JPG")
```

Out[1]:



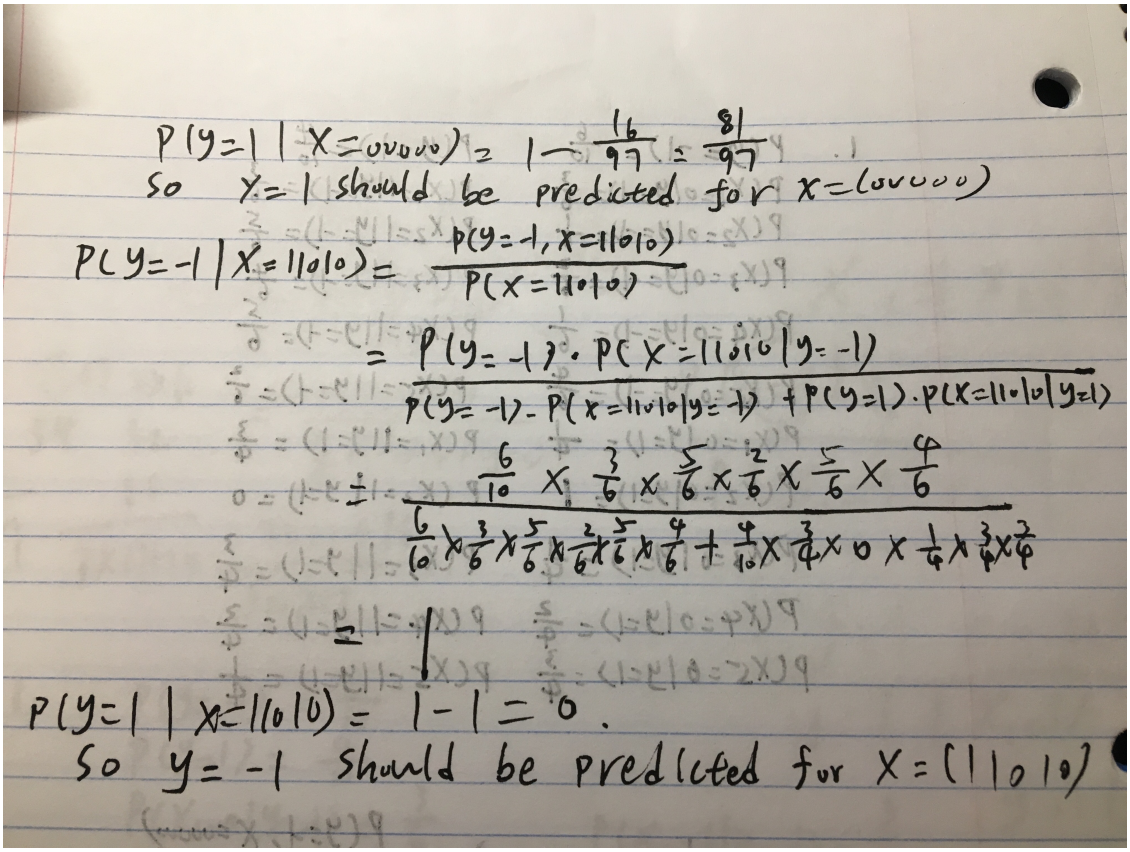
In [2]:
```
from IPython.display import Image
Image("2.JPG")
```

Out[2]:



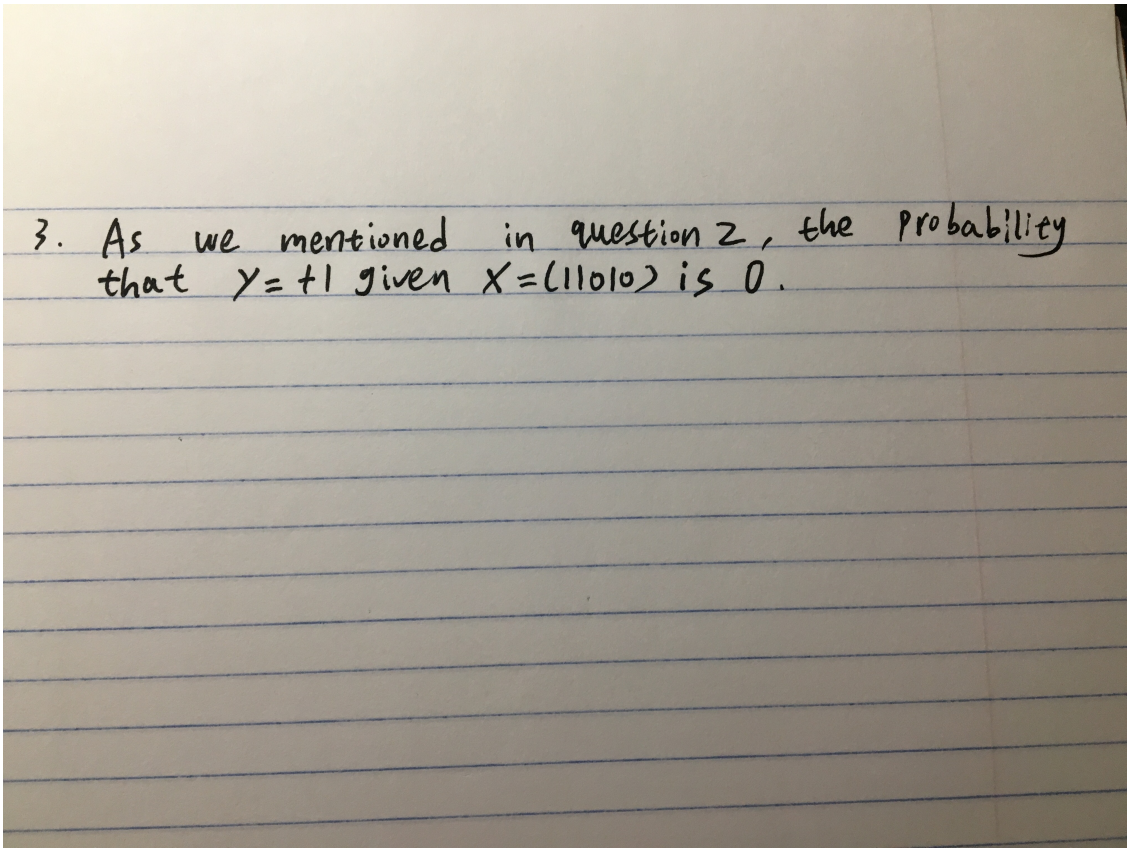
In [6]:
```
from IPython.display import Image
Image("22.JPG")
```

Out[6]:



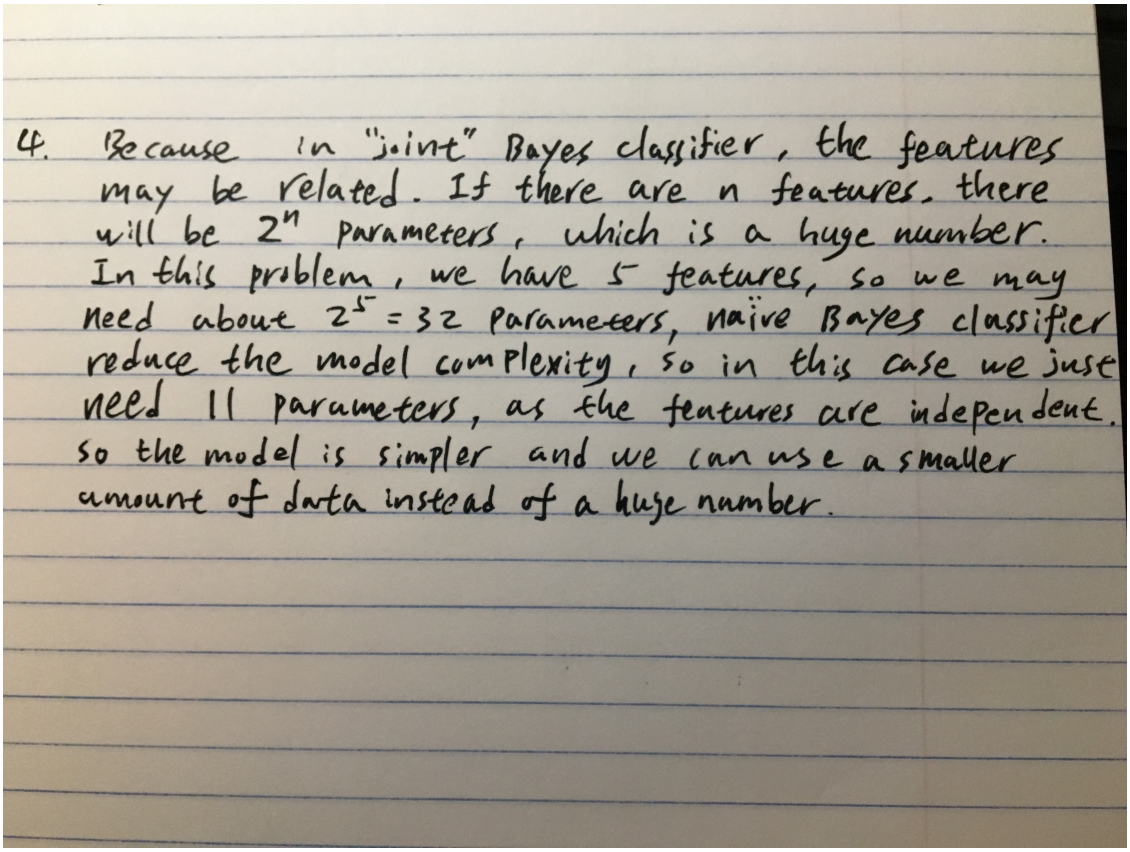
In [3]:
```
from IPython.display import Image
Image("3.JPG")
```

Out[3]:



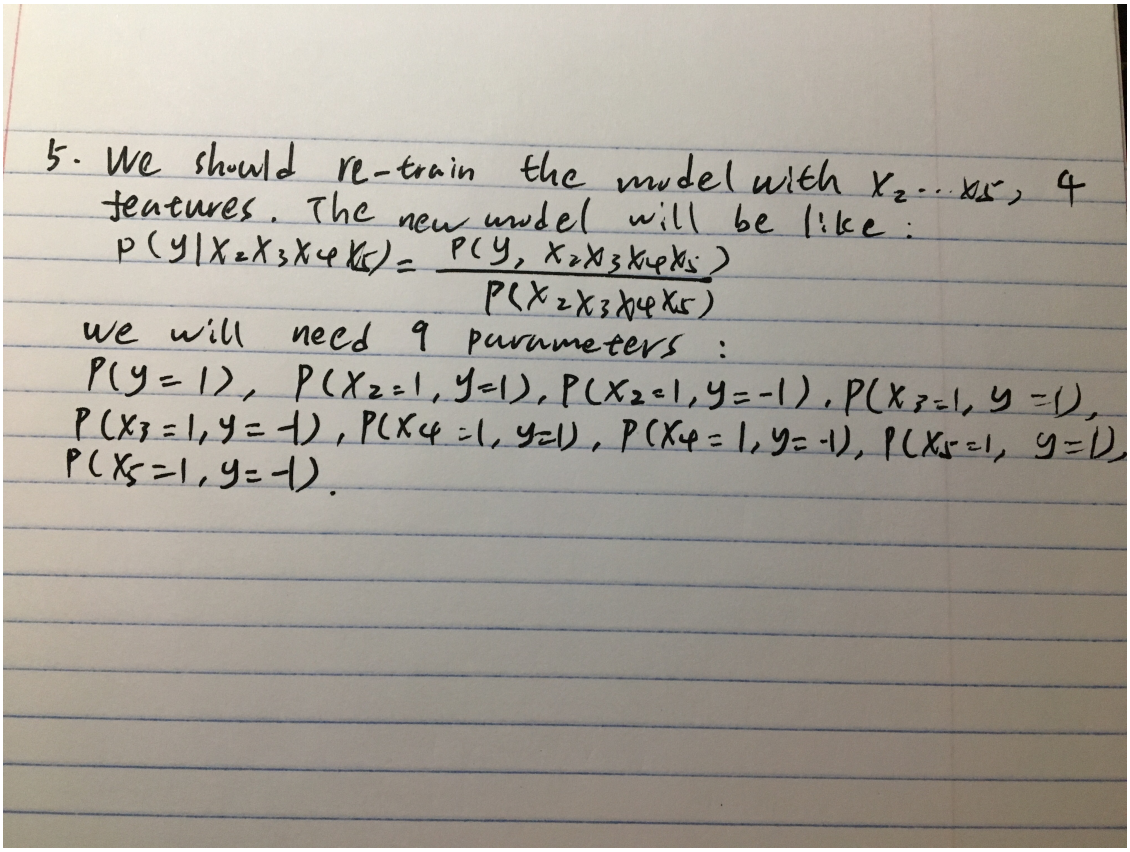

```
In [4]: from IPython.display import Image
        Image("4.JPG")
```

Out[4]:




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

Out[5]:



4 Statement of Collaboration

In [ ]: No discussion with others.