```
In [34]: from __future__ import division
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
%matplotlib inline
```

Problem 1

```
In [35]: iris = np.genfromtxt("data/iris.txt", delimiter = None) #load the text file
Y = iris[:, -1]  # target value (iris species) is the last column
X = iris[:,0:-1] # features are the other columns
X.shape
```

```
Out[35]: (148, 4)
```

Problem 1 - Part 1

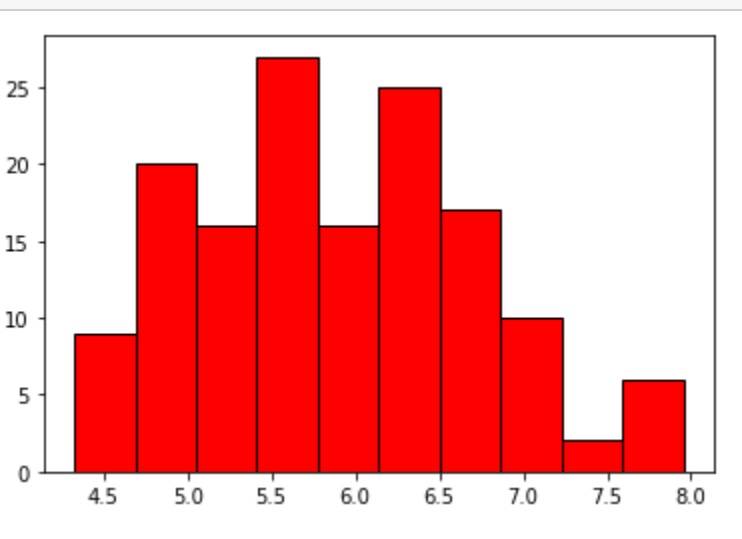
```
In [36]: # X.shape prints the size of the matrix
# here we have 148 rows which are the data points
# and 4 columns which are the features
print("Number of the data points: ", X.shape[0], "\nNumber of the features: ", X.shape[1])
```

```
Number of the data points:  148
Number of the features:  4
```

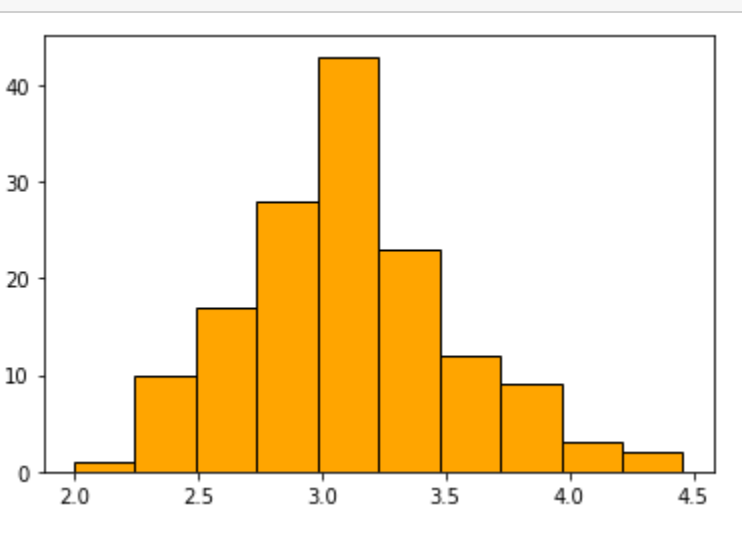
Problem 1 - Part 2

```
In [37]: x1 = X[:,0]  # first columns for all the rows of the features
x2 = X[:,1]  # second columns for all the rows of the features
x3 = X[:,2]  # third columns for all the rows of the features
x4 = X[:,3]  # fourth columns for all the rows of the features
```

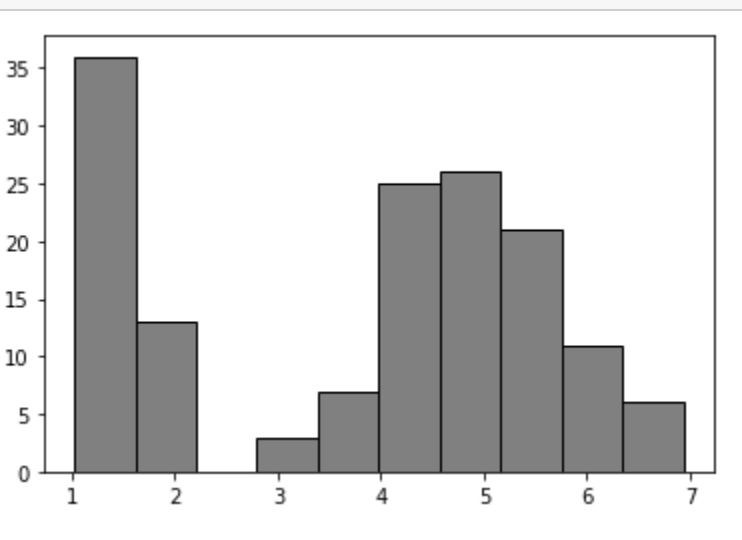
```
In [38]: plt.hist(x1, color = "r", edgecolor = "black")
plt.show()
```



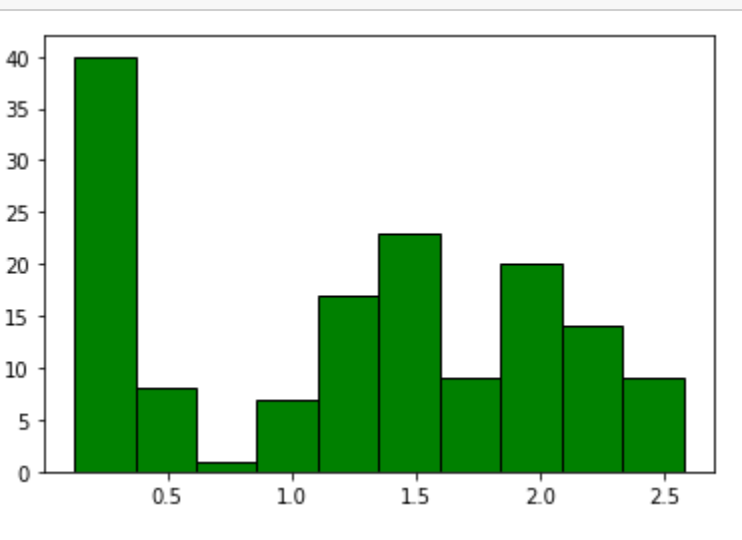

```
In [39]: plt.hist(x2, color = "orange", edgecolor = "black")
plt.show()
```




```
In [57]: plt.hist(x3, color = "grey", edgecolor = "black")
plt.show()
```




```
In [41]: plt.hist(x4, color = "green", edgecolor = "black")
plt.show()
```



Problem 1 - Part 3

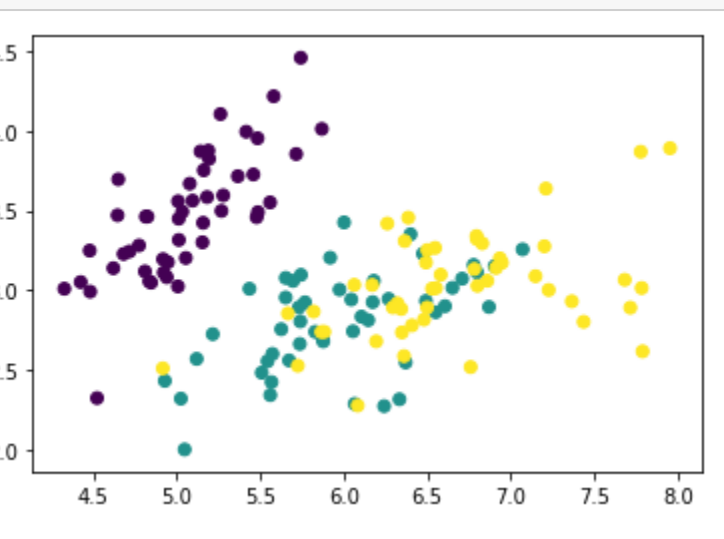
```
In [42]: print("Feature one - Mean: ", x1.mean())
print("Feature one  - Standared diviation: ", x1.std())
print("Feature Two - Mean: ", x2.mean())
print("Feature Two  - Standared diviation: ", x2.std())
```

```
Feature one - Mean:  5.900103764189188
Feature one  - Standared diviation:  0.833402066774894
Feature Two - Mean:  3.098930916891892
Feature Two  - Standared diviation:  0.43629183800107685
```

Problem 1 - Part 4

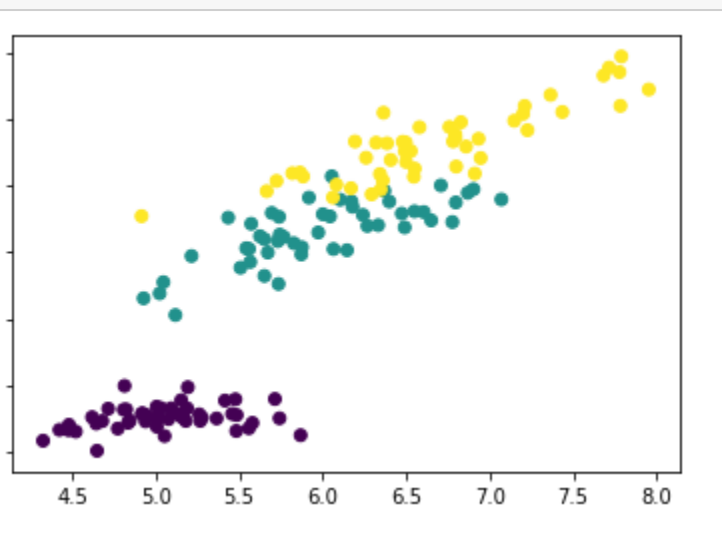
Feature (1, 2)

```
In [43]: plt.scatter(X[:,0], X[:,1], c = Y)
plt.show()
```



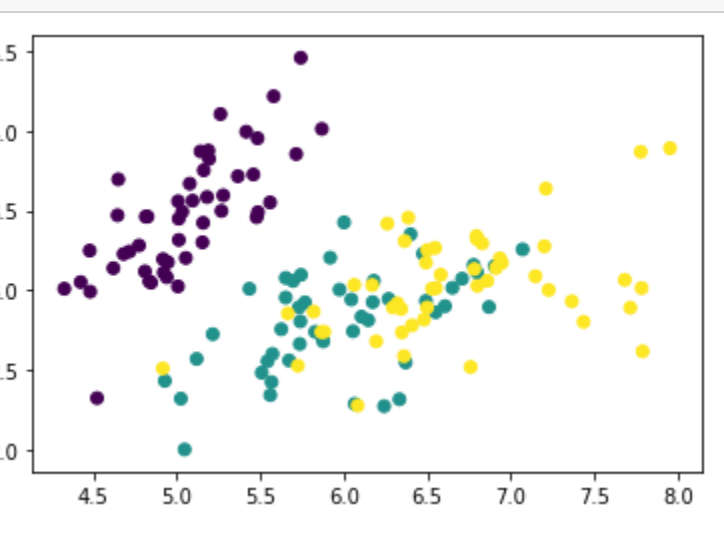
Feature (1, 3)

```
In [44]: plt.scatter(X[:,0], X[:,2], c = Y)
plt.show()
```



Feature (1, 4)

```
In [45]: plt.scatter(X[:,0], X[:,1], c = Y)
plt.show()
```

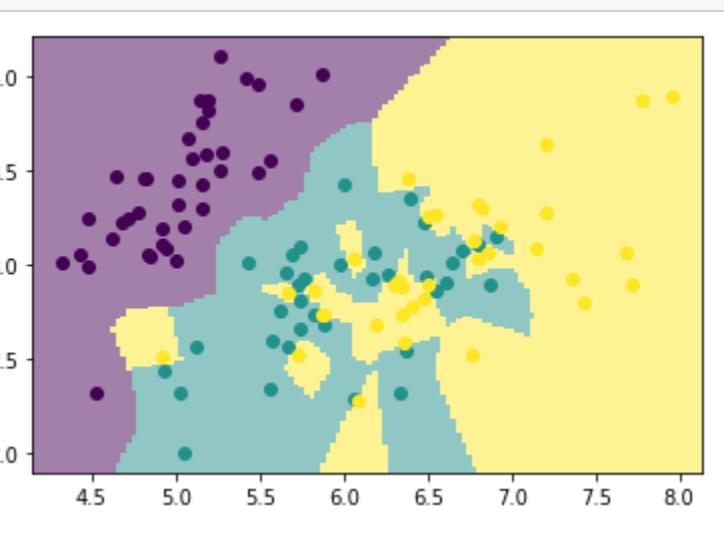


Problem 2

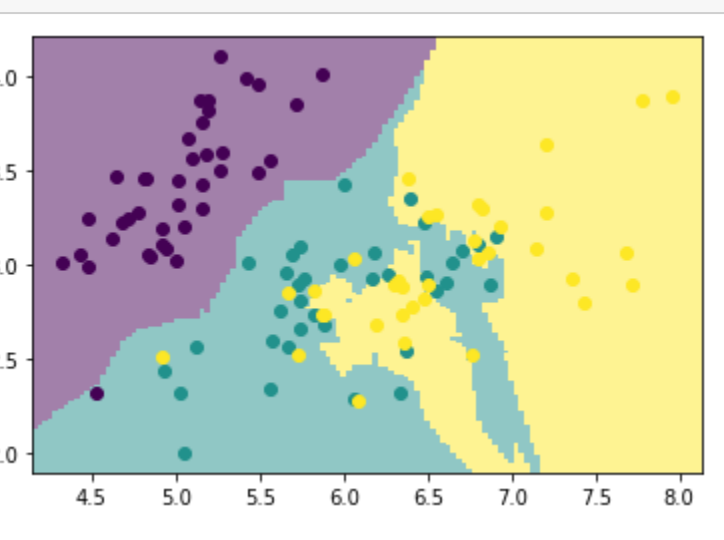
```
In [46]: np.random.seed(0)
iris = np.genfromtxt("data/iris.txt",delimiter=None)
np.random.seed(0)
Y = iris[:, -1]
X = iris[:, 0:-1]
X, Y = ml.shuffleData(X, Y)    # Shuffle the data
Xtr, Xva, Ytr, Yva = ml.splitData(X, Y, 0.75) #split the data into 75/25 train validation
```

Problem 2 - Part 1

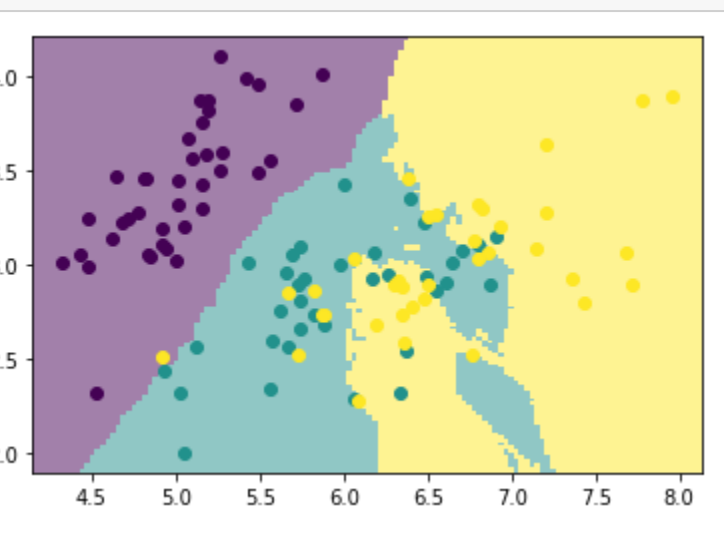
```
In [47]: knn = ml.knn.knnClassify() # create the object and train it
K = [1, 5, 10, 50]
knn.train(Xtr[:, 0:2], Ytr, K[0])
YvaHat = knn.predict(Xva[:, :2])
ml.plotClassify2D(knn, Xtr[:, :2], Ytr)
```



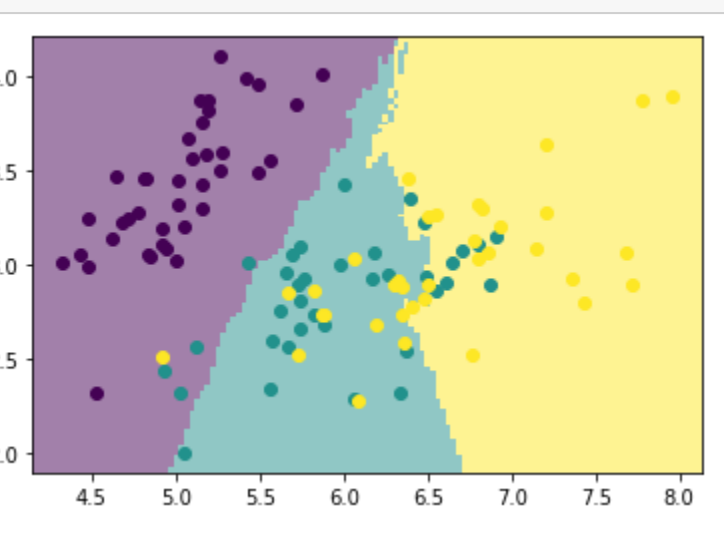

```
In [48]: knn = ml.knn.knnClassify() # create the object and train it
knn.train(Xtr[:, :2], Ytr, K[1])
YvaHat = knn.predict(Xva[:, :2])
ml.plotClassify2D(knn, Xtr[:, :2], Ytr)
```




```
In [49]: knn = ml.knn.knnClassify() # create the object and train it
knn.train(Xtr[:, :2], Ytr, K[2])
YvaHat = knn.predict(Xva[:, :2])
ml.plotClassify2D(knn, Xtr[:, :2], Ytr)
```

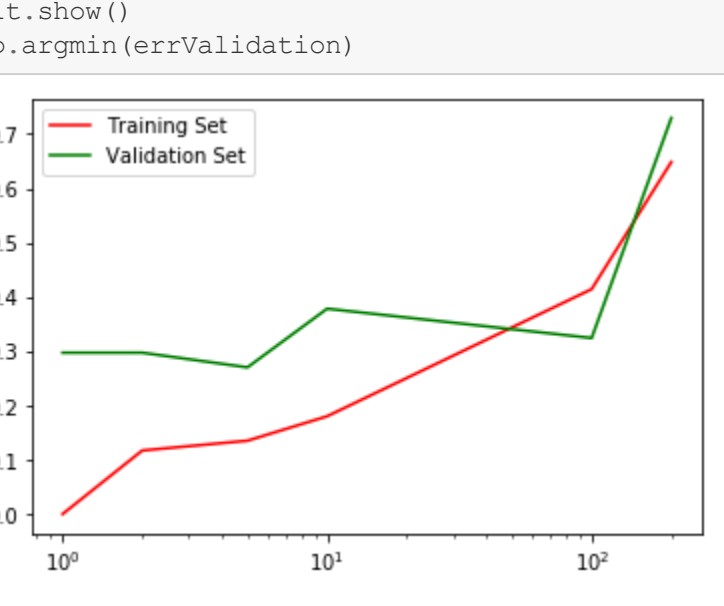



```
In [50]: knn = ml.knn.knnClassify() # create the object and train it
knn.train(Xtr[:, :2], Ytr, K[3])
YvaHat = knn.predict(Xva[:, :2])
ml.plotClassify2D(knn, Xtr[:, :2], Ytr)
```



Problem 2 - Part 2

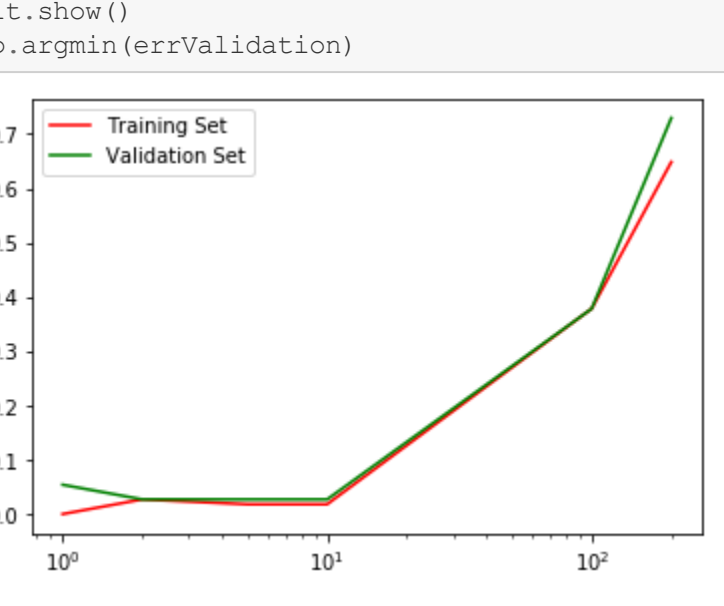
```
In [61]: K = [1, 2, 5, 10, 100, 200]
errTrain      = [None] * len(K)  # A list of size K with None stored in them
errValidation = [None] * len(K)
for i, k in enumerate(K):
    learner = ml.knn.knnClassify(Xtr[:, :2], Ytr, k)
    Yhat = learner.predict(Xva[:, :2])
    errTrain[i]      = learner.err(Xtr[:, :2], Ytr) # Error in training data
    errValidation[i] = learner.err(Xva[:, :2], Yva)    # Error in Validation (Test) data
plt.semilogx(K,errTrain, color = "red", label = "Training Set")
plt.semilogx(K,errValidation, color = "green", label = "Validation Set")
plt.legend()
plt.show()
np.argmin(errValidation)
```




```
Out[61]: 2
```

Problem 2 - Part 3

```
In [60]: K = [1, 2, 5, 10, 100, 200]
errTrain      = [None] * len(K)  # A list of size K with None stored in them
errValidation = [None] * len(K)
for i, k in enumerate(K):
    learner = ml.knn.knnClassify(Xtr, Ytr, k)
    Yhat = learner.predict(Xva)
    errTrain[i]      = learner.err(Xtr, Ytr) # Error in training data
    errValidation[i] = learner.err(Xva, Yva)    # Error in Validation (Test) data
plt.semilogx(K, errTrain, color = "red", label = "Training Set")
plt.semilogx(K, errValidation, color = "green", label = "Validation Set")
plt.legend()
plt.show()
np.argmin(errValidation)
```




```
Out[60]: 1
```

Problem 3:

Problem 3 - Part 1:

P(Y = 1) = 4/10

P(X1 = 1 | Y = 1) = 3/4

P(X2 = 1 | Y = 1) = 0

P(X3 = 1 | Y = 1) = 3/4

P(X4 = 1 | Y = 1) = 2/4 = 1/2

P(X5 = 1 | Y = 1) = 1/4

P(X1 = 1 | Y = 1) = 1/4

P(X2 = 1 | Y = 1) = 1

P(X3 = 1 | Y = 1) = 1/4

P(X4 = 1 | Y = 1) = 2/4 = 1/2

P(X5 = 1 | Y = 1) = 3/4

P(Y = -1) = 6/10

P(X1 = 1 | Y = -1) = 3/6 = 1/2

P(X2 = 1 | Y = -1) = 5/6

P(X3 = 1 | Y = -1) = 4/6 = 2/3

P(X4 = 1 | Y = -1) = 5/6

P(X5 = 1 | Y = -1) = 2/6 = 1/3

P(X1 = 1 | Y = -1) = 3/6 = 1/2

P(X2 = 1 | Y = -1) = 1/6

P(X3 = 1 | Y = -1) = 2/6 = 1/3

P(X4 = 1 | Y = -1) = 1/6

P(X5 = 1 | Y = -1) = 4/6 = 2/3

Problem 3 Part 2:

P(00000,Y = 1) = P(Y = 1) [P(X1 = 0 | Y = 1) P(X2 = 0 | Y = 1) P(X3 = 0 | Y = 1) P(X4 = 0 | Y = 1) P(X5 = 0 | Y = 1)] = (4/10) (1/4) (4/4) (1/4) (2/4) (3/4) = 0.009375

P(00000,Y = -1) = P(Y = -1) P(X1 = 0 | Y = -1) P(X2 = 0 | Y = -1) P(X3 = 0 | Y = -1) P(X4 = 0 | Y = -1) P(X5 = 0 | Y = -1) = (6/10) (3/6) (1/6) (2/6) (1/6) (4/6) = 0.00185

yhat = 1 which means class y = 1 will be pridicted since p(y = 1, X = 00000) = 0.009375 is bigger. This means email must be "read".

P(11010,Y = 1) = P(Y = 1) P(X1 = 1 | Y = 1) P(X2 = 1 | Y = 1) P(X3 = 1 | Y = 1) P(X4 = 1 | Y = 1) P(X5 = 1 | Y = 1) = (4/10) (3/4) (0/4) (3/4) (2/4) (1/4) = 0

P(11010,Y = -1) = P(Y = -1) P(X1 = 1 | Y = -1) P(X2 = 1 | Y = -1) P(X3 = 1 | Y = -1) P(X4 = 1 | Y = -1) P(X5 = 1 | Y = -1) = (6/10) (3/6) (5/6) (2/6) (5/6) (4/6) = 0.0463

yhat = -1 which means class y = -1 will be pridicted since p(y = -1, X = 11010) = 0.0463 is bigger. This means email will be "discarded".

Problem 3 - Part 3:

P(Y=1 | X = 11010) = P(X = 11010 | Y = 1) P(Y = 1) / P(Y = 1) P(X = 11010 | Y = 1) + P(X = 11010 | Y = -1) P(Y = -1) = 0

P(Y=1 | X = 00000) = P(X = 00000 | Y = 1) P(Y = 1) / P(Y = 1) P(X = 00000 | Y = 1) + P(X = 00000 | Y = -1) P(Y = -1) = 0.8351

Problem 3 - Part 4:

Joint Bayes classifier takes more time which in this case we will have 33 parameters. Using joint Bayas classifier makes the computation more complex. So when features are independent, it is better to use naïve Bayes classifier which in this case will be 11 parameters because we have fewer parameters and it is easier to do probabilities.

Problem 3 - Part 5:

In this case, because our parameters are independent we do not need to retain our classifier. In our case, we have nine parameters because we have four features. We just won't use X1 probability which means we don't need to use P(X1 = 1 | Y = 1) and probability of rest will not change so we do not need to retain.

Problem 4:

For this homework the notes from lecture and discussion helped me a lot and I also followed up with piazza whenever I needed help.