Aim: Implement Naive Bayes classifier for Spambased dataset.

In [171]:
```
import numpy as np
from sklearn.cross_validation import train_test_split
```

In [172]:
```
data=open('C:/Users/cglab/Desktop/spam.txt','r')
```

In [173]:
```
d=[]
for line in data:
    line=[float(element) for element in line.rstrip('\n').split(',')]
    d.append(np.asarray(line))

```

In [174]:
```
num_features=48
x=[d[i][:num_features] for i in range(len(d))]
y=[int(d[i][-1]) for i in range(len(d))]
```

In [175]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_stat
```

In [176]:
```
#calculating likelihood estimations
x_train_class0=[x_train[i] for i in range(len(x_train)) if y_train[i]==0]
x_train_class1=[x_train[i] for i in range(len(x_train)) if y_train[i]==1]
```

In [177]:
```
likelihood_class0=np.mean(x_train_class0,axis=0)/100.0
likelihood_class1=np.mean(x_train_class1,axis=0)/100.0
```

In [178]:
```
def cal_log_likelihood(feature_vector,Class):
    assert len(feature_vector)==num_features
    log_likelihood=0.0
    if Class==0:
        for feature_index in range(len(feature_vector)):
            if feature_vector[feature_index]==1:
                log_likelihood += np.log10(likelihood_class0[feature_index])
            elif feature_vector[feature_index] ==0:
                log_likelihood+=np.log10(1.0-likelihood_class0[feature_index]
    elif Class==1:
        for feature_index in range(len(feature_vector)):
            if feature_vector[feature_index]==1:
                log_likelihood+=np.log10(likelihood_class1[feature_index])
            elif feature_vector[feature_index]==0:
                log_likelihood+=np.log10(1.0-likelihood_class1[feature_index]
    else:
        raise ValueError("Class takes integer values 0 or 1")
    return log_likelihood
```

In [179]:
```
num_class0=float(len(x_train_class0))
num_class1=float(len(x_train_class1))

prior_prob_class0=num_class0/(num_class0+num_class1)
prior_prob_class1=num_class1/(num_class0+num_class1)

log_prior_class0=np.log10(prior_prob_class0)
log_prior_class1=np.log10(prior_prob_class1)
```

In [180]:
```
def cal_class_posterier(feature_vector):
    log_likelihood_class0=cal_log_likelihood(feature_vector,Class=0)
    log_likelihood_class1=cal_log_likelihood(feature_vector,Class=1)

    log_posterier_class0=log_likelihood_class0+log_prior_class0
    log_posterier_class1=log_likelihood_class1+log_prior_class1

    return log_posterier_class0,log_posterier_class1

```

In [181]:
```
def classify_spam(doc_vector):
    feature_vector=[int(element>0.0) for element in doc_vector]
    log_posterier_class0,log_posterier_class1=cal_class_posterier(feature_vec
    if log_posterier_class0>log_posterier_class1:
        return 0
    else:
        return 1

```

In [182]:
```
predictions=[]
for email in x_test:
    predictions.append(classify_spam(email))
```

In [183]:
```
def accuracy(predictions,y):
    correct_count=0.0
    for item in range(len(predictions)):
        if predictions[item]==y[item]:
            correct_count+=1.0
    acc=correct_count/len(predictions)
    return acc
```

In [184]:
```
accuracy(predictions,y_test)*100
```

Out[184]: 89.22675933970461