ALL CODES IN Python

```
def naive(total):
	f = open('../data/train','r')
	line = f.readline()

	# Hash of nonspam words
	Ham ={}
	#number of nonspam emails
	nham =0
	hamWords =0

	#hash of spam words
	Spam ={}
	#number of spam emails
	nspam =0
	spamWords=0

	#nu
#	total = 3000;

	while(line):

		email = line.split(' ')
		if(email[1]=='ham'):
			i=2
			while(i<len(email)-1):
				word = email[i]
				count= int(email[i+1])
				if(Ham.has_key(word)==True):
					Ham[word] = Ham[word]+count
				else:
					Ham[word]=count
				i+=2
				hamWords+=count
			nham+=1

		else:
			i=2
			while(i<len(email)-1):
				word = email[i]
				count= int(email[i+1])
				if(Spam.has_key(word)==True):
					Spam[word] = Spam[word]+count
				else:
					Spam[word]=count
				i+=2
				spamWords+=count
			nspam+=1
		line = f.readline()
		total-=1
		if(total==0):
```

```
                break
f.close()

dict = len(Spam)+len(Ham)+763



ftest = open('../data/test','r')
line = ftest.readline()
val =1
error =0

while(line):
        email = line.split(' ')
        i=2
        pham =0.0
        pspam=0.0

        while(i<len(email)-1):

                        word = email[i]
                        count= int(email[i+1])
                        if(Ham.has_key(word)==True):
                                val = float(Ham[word]+1)/(dict+hamWords)
                        else:
                                val = float(1)/(dict+hamWords)

                        pham = pham+ count*(math.log(val))
                        i+=2
        #pham = pham+math.log(float(nham)/(nham+nspam))
        i=2
        while(i<len(email)-1):
                        word = email[i]
                        count= int(email[i+1])
                        if(Spam.has_key(word)==True):
                                val = float(Spam[word]+1)/(dict+spamWords)
                        else:
                                val = float(1)/(dict+spamWords)
                        pspam = pspam+ count*(math.log(val))
                        i+=2

        if(email[1]=='ham' and pspam > pham):
                error+=1
        if(email[1]=='spam' and pspam < pham):
                error+=1
        line = ftest.readline()

arr = Ham.items()
brr = Spam.items()
i =0
top ={}
for i in range(len(arr)):
```

```
                if(Spam.has_key(arr[i][0])):
                        ratio = Spam[arr[i][0]]/Ham[arr[i][0]]
                        top[arr[i][0]] = ratio
        arr= top.items()
        arr.sort(key=lambda tup: tup[1])
        #print arr
        #print Spam['Mier']
        return (1000-error)/10

train =1000

while(train<10000):
        print train,'\t',naive(train)
        train+=1000
```

perceptron

```
import math

def mult(a,b):
        mul=0
        for i in range(len(a)):
                mul=mul+a[i]*b[i]
        return mul

def add(a,b):
        for i in range(len(a)):
                a[i]=a[i]+b[i]
        return a

def norm(a):
        sum=0
        for i in range(len(a)):
                sum=sum+a[i]*a[i]
        return math.sqrt(sum)

label ={}
x ={}
f = open('../data/train','r')
line = f.readline()
itr =0
total = 1000;
Total = total
max_len =0
while(line):
        itr=itr+1
        email = line.split(' ')
        if(email[1]=='ham'):
                label[itr] =1
        else:
                label[itr] =0
```

```
        i=2
        arr= []
        while(i<len(email)-1):
                word = email[i]
                count= int(email[i+1])
                arr.append(count)
                i+=2
        x[itr] = arr
        if(len(arr)>max_len):
                max_len=len(arr)

        line = f.readline()
        total-=1
        if(total==0):
                break

f.close()

w=[0]*max_len
converge = False
while(converge!=True):
        delta_w =[0]*max_len
        while(itr>0):
                out = mult(x[itr],w)
                o=0
                if(out>0):
                        o=1
                for j in range(len(x[itr])):
                        delta_w[j] = delta_w[j]+0.1*(label[itr]-o)*(x[itr][j])
                w= add(w,delta_w)
                if(norm(delta_w) <0.01):
                        converge = True

                itr-=1
        itr = Total
#print len(w),max_len


label_test ={}
x_test ={}
ftest = open('../data/test','r')
line = ftest.readline()
itr =0
max_len =0
while(line):
        itr=itr+1
        email = line.split(' ')
        if(email[1]=='ham'):
                label_test[itr] =1
        else:
                label_test[itr] =0
        i=2
```

```
        arr= []
        while(i<len(email)-1):
                word = email[i]
                count= int(email[i+1])
                arr.append(count)
                i+=2
        x_test[itr] = arr

        line = ftest.readline()

ftest.close()
error =0

while(itr>0):
        out = mult(x_test[itr],w)
        o=0
        if(out>0):
                o=1
        if(o!=label_test[itr]):
                error+=1
        itr-=1
print error
```

svm

```
import math
import svm

def mult(a,b):
        mul=0
        for i in range(len(a)):
                mul=mul+a[i]*b[i]
        return mul

def add(a,b):
        for i in range(len(a)):
                a[i]=a[i]+b[i]
        return a

def norm(a):
        sum=0
        for i in range(len(a)):
                sum=sum+a[i]*a[i]
        return math.sqrt(sum)

label ={}
x ={}
f = open('../data/train','r')
line = f.readline()
itr =0
total = 1000;
Total = total
```

```
max_len =0


while(line):
	itr=itr+1
	email = line.split(' ')
	if(email[1]=='ham'):
		label[itr] =-1
	else:
		label[itr] =1
	i=2
	arr= []
	while(i<len(email)-1):
		word = email[i]
		count= int(email[i+1])
		arr.append(count)
		i+=2
	x[itr] = arr
	if(len(arr)>max_len):
		max_len=len(arr)

	line = f.readline()
	total-=1
	if(total==0):
		break

f.close()

trained = svm.train(lebel,x,'-t 0')



label_test ={}
x_test ={}
ftest = open('../data/test','r')

line = ftest.readline()
itr =0
max_len =0
while(line):
	itr=itr+1
	email = line.split(' ')
	if(email[1]=='ham'):
		label_test[itr] =-1
	else:
		label_test[itr] =1
	i=2
	arr= []
	while(i<len(email)-1):
		word = email[i]
		count= int(email[i+1])
		arr.append(count)
```

```
            i+=2
        x_test[itr] = arr

        line = ftest.readline()

ftest.close()

[prediction,accuracy,values]= svm.predict(label_test,x_test,trained)
```