HW1 Problem 2

Part A

Read in dataset MNIST

In [ ]:

```
from sklearn.datasets import fetch_mldata

# read in data
mnist = fetch_mldata('MNIST original',
                     data_home='D:\MCS-DS\AML\HW1-b')
```

Create untouched, bounding data

In [24]:

```
import pandas as pd
import numpy as np
from PIL import Image
from resizeimage import resizeimage
from sklearn.model_selection import train_test_split

# untouched

X_train_utch,X_test_utch,y_train_utch,y_test_utch=train_test_split(mnist.data,
                                                                   mnist.target,test_size=1/7.0)
# bounding box
data = pd.DataFrame(mnist.data)
df = np.where(data<128,0,1)
df = pd.DataFrame(df)
rec=[]
#img=Image.new('1',(28,28))

for i in range(df.shape[0]):
    img=Image.new('1',(28,28))
    img.putdata(df.iloc[i,])
    img.save('my.png')
    im=Image.open('my.png')
    im = resizeimage.resize_crop(im,[20,20])
    pixel=[i/255 for i in list(im.getdata())]
    rec.append(pixel)

data_bb=pd.DataFrame(rec)
```

create stretched bounding data

In [39]:

```
#streched bounding
rac=[]
for i in range(df.shape[0]):
    #generate a new image
    img=Image.new('1',(28,28))
    # put the data in
    img.putdata(df.iloc[i,])
    img.save('my.png')
    im=Image.open('my.png')
    #crop into the getbox() coordinates
    im=im.crop(im.getbbox())
    # enlarge to size 20*20
    im = im.resize((20,20))
    # get the data
    pixel=[i/255 for i in list(im.getdata())]
    rac.append(pixel)

data_sb = pd.DataFrame(rac)
```

split dataset

In [42]:

```
#split the data 60000 training vs 10000 test
X_train_bb,X_test_bb,y_train_bb,y_test_bb=train_test_split(data_bb,
                                                          mnist.target,test_size=1/7.0)
X_train_sb,X_test_sb,y_train_sb,y_test_sb=train_test_split(data_sb,
                                                          mnist.target,test_size=1/7.0)
```

In [61]:

```
#Gaussian, untouch
from sklearn.naive_bayes import GaussianNB, BernoulliNB
gnb=GaussianNB()
y_pred=gnb.fit(X_train_utch,y_train_utch).\
predict(X_test_utch)
a=((y_pred==y_test_utch).sum()/len(y_test_utch))
print(a)
```

```
0.5564
```

In [62]:

```
#Gaussian, sb
y_pred=gnb.fit(X_train_sb,y_train_sb).\
predict(X_test_sb)
b=((y_pred==y_test_sb).sum()/len(y_test_sb))
print(b)
```

```
0.8179
```

In [63]:

```
#Bernoulli,untouched
bnb=BernoulliNB()
y_pred=bnb.fit(X_train_utch,y_train_utch).\
predict(X_test_utch)
c=((y_pred==y_test_utch).sum()/len(y_test_utch))
print(c)
```

```
0.8244
```

In [64]:

```
#Bernoulli, sb
y_pred=bnb.fit(X_train_sb,y_train_sb).\
predict(X_test_sb)
d=((y_pred==y_test_sb).sum()/len(y_test_sb))
print(d)
```

```
0.8166
```

In [65]:

```
pd.DataFrame(np.reshape([a,b,c,d],(2,2)),
             columns=['Gaussian','Bernoulli'],
             index=['untouched','stretched bounding'])
```

Out[65]:

	Gaussian	Bernoulli
untouched	0.5564	0.8179
stretched bounding	0.8244	0.8166

For untouched dataset, Bernoulli is better. For stretched bounding, Gaussian performs slightly better.Classifier are ranked by prediction accuracy on test data

Part b

Random forest with untouched data

In [57]:

```
#random forest, untouched
res=[]
for i in [10,20,30]:
    for j in [4,8,16]:
        rf = RandomForestClassifier(n_estimators=i,max_depth=j)
        rf.fit(X_train_utch,y_train_utch)
        y_pred = rf.predict(X_test_utch)
        res.append(((y_pred==y_test_utch).sum()/len(y_test_utch)))

pd.DataFrame(np.reshape(res,(3,3)), index=['No. trees 10','No. trees 20','No. trees 30'],
             columns=['depth = 4','depth = 8','depth = 16'])
```

Out[57]:

	depth = 4	depth = 8	depth = 16
No. trees 10	0.7479	0.8969	0.9429
No. trees 20	0.7787	0.9041	0.9535
No. trees 30	0.7831	0.9114	0.9573

Random forest with stretched bounding data

In [60]:

```
#random forest, stretched bounding
from sklearn.ensemble import RandomForestClassifier
res1=[]
for i in [10,20,30]:
    for j in [4,8,16]:
        rf = RandomForestClassifier(n_estimators=i,max_depth=j)
        rf.fit(X_train_sb,y_train_sb)
        y_pred = rf.predict(X_test_sb)
        res1.append(((y_pred==y_test_sb).sum()/len(y_test_sb)))
pd.DataFrame(np.reshape(res1,(3,3)), index=['No. trees 10','No. trees 20','No. trees 30'],
             columns=['depth = 4','depth = 8','depth = 16'])
```

Out[60]:

	depth = 4	depth = 8	depth = 16
No. trees 10	0.7216	0.8849	0.9444
No. trees 20	0.7255	0.8966	0.9547
No. trees 30	0.7638	0.9019	0.9568

for both datasets, depth with 16 and 30 total decision trees perform better because more trees with higher depth will get higher accuracy

In [ ]:

```

```