CS498HW1

Xinyi Gu xinyigu2

Three accuracies for 1A, 1B, 1D

1A:0.7477124

1B:0.7601307

1D:0.7352941

Screenshot of your code

```
1 setwd('/Users/Sunny/Desktop/CS498/HW1')
                                                                                                         44 setwd('/Users/Sunny/Desktop/CS498/HW1')
 2 raw_data<-read.csv('pima-indians-diabetes.csv', header=FALSE)</pre>
                                                                                                         45 wdat<-read.csv('pima-indians-diabetes.csv', header=FALSE)
 3 library(klaR)
                                                                                                         46 library(klaR)
                                                                                                         47 library(caret)
 4 library(caret)
                                                                                                         48 bigx<-wdat[,-c(9)]
5 bigx<-wdat[,-c(9)]</pre>
                                                                                                         49 bigy<-wdat[,9]
6 bigy<-wdat[,9]</pre>
                                                                                                         50 nbx<-bigx
7 trscore<-array(dim=10)</pre>
                                                                                                         51 for (i in c(3, 4, 6, 8))
8 tescore<-array(dim=10)</pre>
                                                                                                         52 {vw<-bigx[, i]==0
9 for (wi in 1:10)
                                                                                                                nbx[vw, i]=NA
10 {wtd<-createDataPartition(y=bigy, p=.8, list=FALSE)
       nbx<-bigx
                                                                                                         55 trscore<-array(dim=10)
12
       ntrbx<-nbx[wtd, ]
                                                                                                         56 tescore <- array(dim=10)
13
       ntrby<-bigy[wtd]
                                                                                                         57 for (wi in 1:10)
       trposflag<-ntrby>0
                                                                                                         58 {wtd<-createDataPartition(y=bigy, p=.8, list=FALSE)
15
       ptregs<-ntrbx[trposflag, ]</pre>
                                                                                                                ntrbx<-nbx[wtd, ]
16
       ntregs<-ntrbx[!trposflag,]
                                                                                                                ntrbv<-biav[wtd]
                                                                                                         60
       ntebx<-nbx[-wtd, ]
                                                                                                                trposflag<-ntrby>0
       nteby<-bigy[-wtd]
18
                                                                                                                ptreas<-ntrbx[trposflag, ]
19
       ptrmean<-sapply(ptregs, mean, na.rm=TRUE)
                                                                                                                ntregs<-ntrbx[!trposflag,]
20
       ntrmean<-sapply(ntregs, mean, na.rm=TRUE)
                                                                                                                ntebx<-nbx[-wtd, ]
21
       ptrsd<-sapply(ptregs, sd, na.rm=TRUE)
                                                                                                                nteby<-bigy[-wtd]
22
       ntrsd<-sapply(ntregs, sd, na.rm=TRUE)
                                                                                                                ptrmean<-sapply(ptregs, mean, na.rm=TRUE)
                                                                                                                ntrmean<-sapply(ntregs, mean, na.rm=TRUE)
       ptroffsets<-t(t(ntrbx)-ptrmean)
                                                                                                                ptrsd<-sapply(ptregs, sd, na.rm=TRUE)
       ptrscales<-t(t(ptroffsets)/ptrsd)
                                                                                                                ntrsd<-sapply(ntregs, sd, na.rm=TRUE)
       ptrlogs \leftarrow -(1/2)*rowSums(apply(ptrscales, c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ptrsd))
                                                                                                                ptroffsets<-t(t(ntrbx)-ptrmean)
       ntroffsets<-t(t(ntrbx)-ntrmean)
                                                                                                                ptrscales<-t(t(ptroffsets)/ptrsd)
       ntrscales<-t(t(ntroffsets)/ntrsd)</pre>
27
                                                                                                                ptrlogs \leftarrow -(1/2) \times rowSums(apply(ptrscales, c(1, 2), function(x)x^2), na.rm=TRUE) - sum(log(ptrsd))
28
       ntrlogs \leftarrow (1/2)*rowSums(apply(ntrscales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ntrsd))
                                                                                                                ntroffsets<-t(t(ntrbx)-ntrmean)
29
       lvwtr<-ptrlogs>ntrlogs
                                                                                                                ntrscales<-t(t(ntroffsets)/ntrsd)
30
       gotrighttr<-lvwtr==ntrby
                                                                                                         75
                                                                                                                \label{eq:ntrlogs} $$\operatorname{TRUE}-(1/2)*rowSums(apply(ntrscales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ntrsd)) $$
31
       trscore[wi]<-sum(gotrighttr)/(sum(gotrighttr)+sum(!gotrighttr))</pre>
                                                                                                         76
                                                                                                                lvwtr<-ptrlogs>ntrlogs
       pteoffsets<-t(t(ntebx)-ptrmean)</pre>
                                                                                                                gotrighttr<-lvwtr==ntrby
33
       ptescales<-t(t(pteoffsets)/ptrsd)</pre>
                                                                                                                trscore[wi]<-sum(gotrighttr)/(sum(gotrighttr)+sum(!gotrighttr))</pre>
       ptelogs < -(1/2)*rowSums(apply(ptescales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ptrsd))
34
                                                                                                                pteoffsets<-t(t(ntebx)-ptrmean)
35
       nteoffsets<-t(t(ntebx)-ntrmean)</pre>
                                                                                                                ptescales<-t(t(pteoffsets)/ptrsd)
36
       ntescales<-t(t(nteoffsets)/ntrsd)
                                                                                                                ptelogs < -(1/2) * rowSums(apply(ptescales,c(1, 2), function(x)x^2), na.rm = TRUE) - sum(log(ptrsd))
37
       ntelogs < -(1/2)*rowSums(apply(ntescales,c(1, 2), function(x)x^2), na.rm=TRUE)-sum(log(ntrsd))
                                                                                                                nteoffsets<-t(t(ntebx)-ntrmean)
38
       lvwte<-ptelogs>ntelogs
                                                                                                         83
                                                                                                                ntescales<-t(t(nteoffsets)/ntrsd)
39
       gotright<-lvwte==nteby
                                                                                                         84
                                                                                                                ntelogs < -(1/2) * rowSums(apply(ntescales, c(1, 2), function(x)x^2), na.rm=TRUE) - sum(log(ntrsd))
40
        tescore[wi]<-sum(gotright)/(sum(gotright)+sum(!gotright))</pre>
                                                                                                         85
                                                                                                                lvwte<-ptelogs>ntelogs
41 }
                                                                                                                gotright<-lvwte==nteby
42
                                                                                                                tescore[wi]<-sum(gotright)/(sum(gotright)+sum(!gotright))</pre>
        sum(tescore)/length(tescore)
                                                                                                         87
                                                                                                         88 }
       setwd('/Users/Sunny/Desktop/CS498/HW1')
       rm(list=ls())
       wdat<-read.csv('pima-indians-diabetes.csv', header=FALSE)
       library(klaR)
       library(caret)
       bigx<-wdat[,-c(9)]
       bigx2 < -apply(bigx, c(1, 2), function(x)x^2)
       bigx<-cbind(bigx, bigx2)</pre>
       errs<-array(dim=10)
       cvs<-c(0.005, 0.01, 0.1)
       for (wi in c(1:10))
       {bigy<-as.factor(wdat[,9])
            wtd<-createDataPartition(y=bigy, p=.8, list=FALSE)
            wstring<-paste("-c", sprintf('%f', cvs[wi]), sep=" ")
            svm<-svmlight(bigx[wtd,], bigy[wtd], pathsvm='/Users/Sunny/Desktop/CS498/HW1/svm_light_OS10.8.4_i7/')
            labels<-predict(svm, bigx[-wtd,])</pre>
            foo<-labels$class
            errs[wi]<-sum(foo==bigy[-wtd])/(sum(foo==bigy[-wtd])+sum(!(foo==bigy[-wtd])))
```

reference: http://luthuli.cs.uiuc.edu/~daf/courses/AML-18/RCodeClassification/pimanbholdout.R
cs.uiuc.edu/~daf/courses/AML-18/RCodeClassification/pimanbholdout.R

Table of accuracies for all 12 cases.

Method	Gaussian	Bernoulli	DF_10_4	DF_10_16	DF_30_4	DF_30_16
Untouched	0.5535	0.833	0.7545	0.968	0.793	0.9775
Stretched	0.3705	0.6465	0.6205	0.902	0.651	0.917

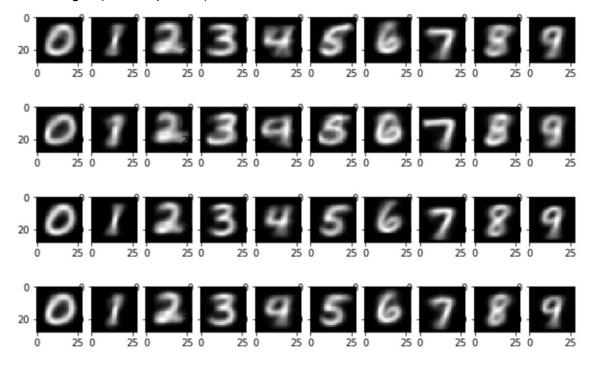
Screenshot of Kaggle (READABLE SCREENSHOT)

xinyigu2_4.csv 6 minutes ago by xinyigu2 Bernoulli_stretched		0.65510	
xinyigu2_3.csv 6 minutes ago by xinyigu2 Bernoulli_untouched		0.83410	
xinyigu2_2.csv 9 minutes ago by xinyigu2 Gaussian_stretched		0.37180	
xinyigu2_1.csv 10 minutes ago by xinyigu2 Gaussian_untouched		0.55560	
xinyigu2_5.csv 10 minutes ago by xinyigu2 Decision_Forest_untouched 10trees 4 depth		0.73990	
xinyigu2_6.csv 3 minutes ago by xinyigu2 Decision_Forest_stretched 10trees 4 depth		0.62915	
xinyigu2_7.csv 9 minutes ago by xinyigu2 Decision_Forest_untouched 10trees 16 depth		0.95840	
xinyigu2_8.csv 3 minutes ago by xinyigu2 Decision_Forest_stretched 10 trees 4 depth		0.89635	
xinyigu2_9.csv 9 minutes ago by xinyigu2 Decision_Forest_untouched 30trees 4 depth		0.80125	
xinyigu2_10.csv 2 minutes ago by xinyigu2 Decision_Forest_stretched 30 trees 4 depth	0.64745		
xinyigu2_11.csv 8 minutes ago by xinyigu2 Decision_Forest_untouched 30trees 16 depth	0.97000		
xinyigu2_12.csv a minute ago by xinyigu2 Decision_Forest_stretched 30 trees 16 depth	0.91235		

A brief explanation of which model is better and why

I think decision forest classifier is better if proper number of trees and depth are chosen. It is easy to interpret and we can apply different numbers to see under what trees and depth this model perform the best. The untouched data set works,in most cases, better than the stretched data set and this is probably because the boundary condition.

40 mean images (4x10 of part 2A)



Screenshot of your code

```
classifier = BernoulliNB()
                                                                                                                       classifier.fit(train set,train label)
     import numpy as np
                                                                                                                       accuracy = accuracy_score(val_label, classifier.predict(val_set))
     import pandas as pd
                                                                                                                       print(accuracy)
     import matplotlib.pyplot as plt
                                                                                                                       return classifier.predict(test set)
     from collections import Counter
     import csv
                                                                                                                  def Decision_Forest(train_set,train_label,test_set,trees,depth,val_set,val_label):
     from sklearn.metrics import accuracy_score
                                                                                                                       classifier = RandomForestClassifier(n_estimators =trees, max_depth=depth)
                                                                                                                       classifier.fit(train_set,train_label)
     from sklearn.naive_bayes import GaussianNB
     from sklearn.naive_bayes import BernoulliNB
                                                                                                                       accuracy = accuracy_score(val_label, classifier.predict(val_set))
     from sklearn.ensemble import RandomForestClassifier
                                                                                                                       print(accuracy)
    def bounding_box(u):
                                                                                                                       return classifier.predict(test set)
          minx = 27
                                                                                                                  def drawpic(test_label, test_data):
          minv = 27
                                                                                                                       num = []
          maxx = 0
                                                                                                                       array = np.zeros((10, 28*28))
          maxy = 0
                                                                                                                       count = Counter(test_label)
          u = np.reshape(u, (28, 28))
                                                                                                                       for i in range(10):
          for x in range (28):
                                                                                                                           num.append(count[i])
                 for y in range (28):
                       if u[x][y] > 0:
                                                                                                                       for i in range(20000):
                             if x < minx:</pre>
                                                                                                                           label = test_label[i]
                                  minx = x
                                                                                                                           for i in range(28*28):
                             if y < miny:</pre>
                                                                                                                                if(test_data[i][j] > 0):
                                   miny = y
                                                                                                                                    array[label][j] += int(test_data[i][j])/(num[label]*255)
                             if x > maxx:
                                                                                                                       draw_figure =plt.figure(figsize=(10,10))
                                   maxx = x
                                                                                                                       for i in range(10):
                             if y > maxy:
                                                                                                                           draw_figure.add_subplot(1, 10, i+1)
                                   maxy = y
                                                                                                                           plt.imshow(array[i].reshape((28,28)), cmap="gray")
          v = u[minx:maxx+1][miny:maxy+1]
          v = np.resize(v,(20,20))
                                                                                                                   train_set = pd.read_csv("/Users/Sunny/Desktop/CS498/HW1/train.csv").as_matrix()
          v = np.reshape(v, (400))
                                                                                                                   test_set = pd.read_csv("/Users/Sunny/Desktop/CS498/HW1/test.csv", header=None).as_matrix()
          return v
                                                                                                                   val_set = pd.read_csv("/Users/Sunny/Desktop/CS498/HW1/val.csv").as_matrix()
     def array_to_csv(name,array):
                                                                                                                   train label = train set[:,1]
          with open(name, 'wt') as csvfile:
                                                                                                                   train_untouched =train_set[:,2:]
                 writer = csv.writer(csvfile)
                                                                                                                   train_stretched = [bounding_box(row) for row in train_untouched]
                 writer.writerow(('ImageId','Label'))
                                                                                                                  val_label = val_set[:,0]
                 for i in range(len(array)):
                                                                                                                  val untouched = val set[:.1:]
                       writer.writerow((i , array[i]))
                                                                                                                   val_stretched = [bounding_box(row.reshape(784)) for row in val_untouched]
     def Gaussian_NB(train_set,train_label,test_set,val_set,val_label):
                                                                                                                  Gaussian_untouched = Gaussian_NB(train_untouched,train_label,test_set,val_untouched,val_label)
          classifier = GaussianNB()
                                                                                                                  Gaussian_stretched = Gaussian_NB(train_stretched,train_label,test_stretched,val_stretched,val_label)
          classifier.fit(train_set,train_label)
                                                                                                                  Bernoulli\_untouched = Bernoulli\_NB(train\_untouched, train\_label, test\_set, val\_untouched, val\_label)
          accuracy = accuracy_score(val_label, classifier.predict(val_set))
                                                                                                                   Bernoulli_stretched = Bernoulli_NB(train_stretched,train_label,test_stretched,val_stretched,val_label)
          print(accuracy)
                                                                                                                  Decision_Forest_untouched104 = Decision_Forest(train_untouched,train_label,test_set,
          return classifier.predict(test_set)
                                                                                                                       10.4.val untouched.val label)
     def Bernoulli_NB(train_set,train_label,test_set,val_set,val_label):
                                                                                                                  {\tt Decision\_Forest\_stretched104 = Decision\_Forest(train\_stretched, train\_label, test\_stretched,}
          classifier = BernoulliNB()
                                                                                                                       10,4,val_stretched,val_label)
          classifier.fit(train_set,train_label)
Gaussian_untouched = Gaussian_NB(train_untouched,train_label,test_set,val_untouched,val_label)
Gaussian_stretched = Gaussian_NB(train_stretched,train_label,test_stretched,val_stretched,val_label)
Bernoulli_untouched = Bernoulli_NB(train_untouched,train_label,test_set,val_untouched,val_label)
Bernoulli_stretched = Bernoulli_NB(train_stretched,train_label,test_stretched,val_stretched,val_label)
Decision_Forest_untouched104 = Decision_Forest(train_untouched,train_label,test_set,

10,4,val_untouched,val_label)
Decision Forest untouched1016 = Decision Forest(train untouched.train label.test set.
10,16,val_stretched,val_label)

Decision_Forest_stretched1016 = Decision_Forest(train_stretched,train_label,test_stretched, 10,16,val_stretched,val_label)
Decision_Forest_untouched304 = Decision_Forest(train_untouched,train_label,test_set, 30,4,val_untouched,val_label)
Decision_Forest_stretched304 = Decision_Forest(train_stretched,train_label,test_stretched, 30,4,val_stretched,val_label)
Decision_Forest_untouched3016 = Decision_Forest(train_untouched,train_label,test_set,
30,16,val_untouched,val_label)
Decision_Forest_stretched3016 = Decision_Forest(train_stretched,train_label,test_stretched,
      30,16, val_stretched, val_label)
drawpic(Gaussian_untouched, test_set)
drawpic(Gaussian_antouched, test_set)
drawpic(Gaussian_stretched, test_set)
drawpic(Bernoulli_stretched, test_set)
drawpic(Bernoulli_stretched, test_set)
array_to_csv('xinyigu2_1.csv',Gaussian_untouched)
array_to_csv('xinyigu2_2.csv',Gaussian_stretched)
array_to_csv('xinyigu2_3.csv',Bernoulli_untouched)
array_to_csv('xinyigu2_4.csv',Bernoulli_stretched)
array_to_csv('xinyigu2_5.csv',Decision_Forest_untouched104)
array_to_csv('xinyigu2_6.csv',Decision_Forest_stretched104)
array_to_csv('xinyigu2_7.csv',Decision_Forest_untouched1016)
array_to_csv('xinyigu2_8.csv',Decision_Forest_stretched1016)
array_to_csv('xinyigu2_9.csv',Decision_Forest_untouched304)
array_to_csv('xinyigu2_10.csv',Decision_Forest_stretched304)
array_to_csv('xinyigu2_11.csv',Decision_Forest_untouched3016)
array_to_csv('xinyigu2_12.csv',Decision_Forest_stretched3016)
```

def Bernoulli_NB(train_set,train_label,test_set,val_set,val_label):