Homework 2
Hetsvi Navnitlal

Outputs

1. The accuracy for question one is 0.96634774002
   The ber for question one is 0.481074983766
2. The accuracy is 0.7829099307159353
   The ber is 0.2069501067753834
3. The training accuracy is 0.782838283828
   The test accuracy is 0.775725593668
   The validation accuracy is 0.759894459103

   The training ber is 0.23562960171604153
   The test ber is 0.2655976538479732
   The validation ber is 0.27269861286254726
4.

	Training	Test	Validation
10^-4	0.23743183367416498	0.2756868131868132	0.21344232515894634
10^-3	0.22167084949425808	0.2502710027100271	0.24417808219178083
10^-2	0.20025260756192953	0.23179532682638282	0.3056910569105691
10^-1	0.27453924914675765	0.22402087449967067	0.24777626193724422
10^0	0.2172729156184331	0.25627289377289375	0.18064312736443888
10^1	0.23413261750509773	0.3032289628180038	0.32351230858693547
10^2	0.22366405434806746	0.2901084010840109	0.4171409214092141
10^3	0.21342428757682996	0.2534405892614848	0.25895503952569165
10^4	0.150743992849256	0.23313982213438744	0.2799902152641879

The classifier that I would select is 10^0. I would select this because the validation error is the lowest out of all the classifiers.

5.

Beta	Test
1	0.152542372881
0.1	0.126840159073
10	0.660130718954

6. Question 7

```
[ 2.69375795e-18  1.99462091e-07 -9.16065286e-07  1.00707260e-06
  4.47685176e-06  1.85776360e-03  6.53915446e-07  2.22531140e-07
  5.81102093e-06 -5.45981895e-07  8.59874241e-07  1.82969703e-07
  1.22985272e-06  4.38342849e-07  2.22531140e-07 -7.48959325e-04
  1.04744395e-06  6.28786785e-06  2.22531140e-07  4.75984924e-07
  4.51171370e-05 -1.27359959e-05  1.64901802e-07  4.45954189e-07
  6.13104677e-07  1.06092949e-06  9.29260913e-07  4.31157545e-05
```

```
 2.40648310e-05  3.84884436e-06 -1.62703453e-06  4.67271963e-07
-2.77183738e-04  3.70637320e-06 -1.74704454e-06  1.42836161e-07
-1.09426321e-06  9.21557527e-04  7.94876410e-07  1.89091849e-07
 2.15872860e-06 -3.98326451e-06  4.17694411e-07  4.08361082e-05
-4.28264606e-06 -3.04098094e-06  3.22953454e-06  7.11783144e-05
 2.13526045e-07  4.55156204e-07  3.96598476e-06 -8.08134792e-07
-7.23066574e-07  9.15819864e-07  2.40590833e-05  9.99997487e-01
 2.12870572e-07  7.48895118e-07 -2.87901895e-07 -5.42799090e-07
-1.00399336e-04 -1.17071683e-05 -2.54988847e-04  5.11795495e-06
-3.70148719e-06]
```

7. Question 8

N	Test	Validation
5	0.329143223417	0.2828459866
10	0.325293315143	0.353498505345
15	0.254825158946	0.329657895905
20	0.24496996997	0.262784090909
25	0.312926551836	0.358176649221
30	0.233219178082	0.241855821774

Code

```python
#from scipy.io import arff
#from io import StringIO
#import pandas as pd

#data = arff.loadarff('5year.arff')
#Anwers to questions
# the answers are on the document the code is this
#imports
from sklearn import linear_model
import random
from sklearn.model_selection import train_test_split
```

one -parse the bankruptcy data   Train a logistic regressor

```python
file = open('5year.arff', 'r')
while not '@data' in file.readline():
    pass

dataset = []
for i in file:
    if '?' in i:
        continue
    i = i.split(',')
    values = [1] + [float(x) for x in i]
    values[-1] = values[-1] > 0
    dataset.append(values)

X = [values[:-1] for values in dataset]
Y = [values[-1] for values in dataset]
```

```
model = linear_model.LogisticRegression()
model.fit(X,Y)
predictions = model.predict(X)
correct = predictions == Y
accuracy = float(sum(correct))/float(len(correct))
print(accuracy)

labeled_false = 0
for j in Y:
    if j == False:
        labeled_false +=1
labeled_true = 0
for k in Y:
    if k == True:
        labeled_true +=1
false_x = 0
true_x = 0
for n, o in zip(Y, predictions):
    if o == False and n == False:
        false_x +=1
    if o == True and n == True:
        true_x +=1

TPR = float(true_x) / float(labeled_true)
TNR = float(false_x)/ float(labeled_false)
BER = 1- (0.5*(TPR + TNR))
print(BER)
```

two- above model using the class weight=’balanced’

```
file = open('5year.arff', 'r')
while not '@data' in file.readline():
    pass
dataset = []
for i in file:
    if '?' in i:
        continue
    i = i.split(',')
    values = [1] + [float(x) for x in i]
    values[-1] = values[-1] > 0
    dataset.append(values)

X = [values[:-1] for values in dataset]
Y = [values[-1] for values in dataset]

model = linear_model.LogisticRegression(class_weight='balanced')
model.fit(X,Y)
predictions = model.predict(X)
correct = predictions == Y
accuracy = float(sum(correct))/float(len(correct))
print("balanced question two",accuracy)

labeled_false = 0
for j in Y:
    if j == False:
        labeled_false +=1
labeled_true = 0
for k in Y:
    if k == True:
        labeled_true +=1
false_x = 0
```

```
true_x = 0
for n, o in zip(Y, predictions):
    if o == False and n == False:
        false_x +=1
    if o == True and n == True:
        true_x +=1
TPR = float(true_x) / float(labeled_true)
TNR = float(false_x)/ float(labeled_false)
BER = 1- (0.5*(TPR + TNR))
print("balanced question two BER",BER)
```

three- the training/validation/test accuracy and BER

```
file = open('5year.arff', 'r')
while not '@data' in file.readline():
    pass
dataset = []
for i in file:
    if '?' in i:
        continue
    i = i.split(',')
    values = [1] + [float(x) for x in i]
    values[-1] = values[-1] > 0
    dataset.append(values)

random.shuffle(dataset)
X = [values[:-1] for values in dataset]
Y = [values[-1] for values in dataset]

X_train, X_split, y_train, y_split = train_test_split(X, Y, test_size =0.5)
x_vali, x_test, y_vali, y_test = train_test_split(X_split, y_split, test_size = 0.5)

model = linear_model.LogisticRegression(class_weight='balanced')
model.fit(X_train, y_train)
predictionsTrain = model.predict(X_train)
predictionsTest = model.predict(x_test)
predictionsVali = model.predict(x_vali)
correctPredictionsTrain = predictionsTrain == y_train
correctPredictionsTest = predictionsTest == y_test
correctPredictionsVali = predictionsVali == y_vali
train_accuracy = float(sum(correctPredictionsTrain)) /
float(len(correctPredictionsTrain))
test_accuracy = float(sum(correctPredictionsTest)) /
float(len(correctPredictionsTest))
vali_accuracy = float(sum(correctPredictionsVali)) /
float(len(correctPredictionsVali))
print(train_accuracy)
print(test_accuracy)
print(vali_accuracy)
labeled_false_train = 0
for j in y_train:
    if j == False:
        labeled_false_train +=1
labeled_true_train = 0
for k in y_train:
    if k == True:
        labeled_true_train +=1
false_x_train = 0
true_x_train = 0
for n, o in zip(y_train, predictionsTrain):
    if o == False and n == False:
        false_x_train +=1
    if o == True and n == True:
        true_x_train +=1
TPR_train = float(true_x_train) / float(labeled_true_train)
TNR_train = float(false_x_train)/ float(labeled_false_train)
BER_train = 1- (0.5*(TPR_train + TNR_train))
print("question three train",BER_train)

labeled_false_test = 0
for j in y_test:
    if j == False:
        labeled_false_test +=1
labeled_true_test = 0
for k in y_test:
    if k == True:
        labeled_true_test +=1
false_x_test = 0
true_x_test = 0
for n, o in zip(y_test, predictionsTest):
    if o == False and n == False:
        false_x_test +=1
    if o == True and n == True:
        true_x_test +=1
TPR_test = float(true_x_test) / float(labeled_true_test)
TNR_test = float(false_x_test)/ float(labeled_false_test)
BER_test = 1- (0.5*(TPR_test + TNR_test))
print("question three test ber", BER_test)

labeled_false_vali = 0
for j in y_vali:
    if j == False:
        labeled_false_vali +=1
labeled_true_vali = 0
for k in y_vali:
    if k == True:
        labeled_true_vali +=1
false_x_vali = 0
true_x_vali = 0
for n, o in zip(y_vali, predictionsVali):
    if o == False and n == False:
        false_x_vali +=1
    if o == True and n == True:
        true_x_vali +=1
TPR_vali = float(true_x_vali) / float(labeled_true_vali)
TNR_vali = float(false_x_vali)/ float(labeled_false_vali)
BER_vali = 1- (0.5*(TPR_vali + TNR_vali))
print("question three validation ber",BER_vali)
```

four- Implement a complete regularization pipeline with the balanced classifier

```
def questionFour(c):
    file = open('5year.arff', 'r')
    while not '@data' in file.readline():
      pass
    dataset = []
    for i in file:
        if '?' in i:
            continue
        i = i.split(',')
        values = [1] + [float(x) for x in i]
        values[-1] = values[-1] > 0
        dataset.append(values)
    random.shuffle(dataset)
    X = [values[:-1] for values in dataset]
    Y = [values[-1] for values in dataset]

    X_train, X_split, y_train, y_split = train_test_split(X, Y, test_size =0.5)
    x_vali, x_test, y_vali, y_test = train_test_split(X_split, y_split, test_size =
0.5)

    model = linear_model.LogisticRegression(class_weight='balanced', C=c)
    model.fit(X_train, y_train)
    predictionsTrain = model.predict(X_train)
    predictionsTest = model.predict(x_test)
    predictionsVali = model.predict(x_vali)

    labeled_false_train = 0
    for j in y_train:
        if j == False:
            labeled_false_train +=1
    labeled_true_train = 0
    for k in y_train:
        if k == True:
            labeled_true_train +=1
    false_x_train = 0
    true_x_train = 0
    for n, o in zip(y_train, predictionsTrain):
        if o == False and n == False:
            false_x_train +=1
        if o == True and n == True:
            true_x_train +=1
    TPR_train = float(true_x_train) / float(labeled_true_train)
    TNR_train = float(false_x_train)/ float(labeled_false_train)
    BER_train = 1- (0.5*(TPR_train + TNR_train))
    print("question four train",BER_train)

    labeled_false_test = 0
    for j in y_test:
        if j == False:
            labeled_false_test +=1
    labeled_true_test = 0
    for k in y_test:
        if k == True:
            labeled_true_test +=1
    false_x_test = 0
    true_x_test = 0
    for n, o in zip(y_test, predictionsTest):
        if o == False and n == False:
            false_x_test +=1
        if o == True and n == True:
            true_x_test +=1
    TPR_test = float(true_x_test) / float(labeled_true_test)
    TNR_test = float(false_x_test)/ float(labeled_false_test)
    BER_test = 1- (0.5*(TPR_test + TNR_test))
    print("question four test ber", BER_test)

    labeled_false_vali = 0
    for j in y_vali:
        if j == False:
            labeled_false_vali +=1
    labeled_true_vali = 0
    for k in y_vali:
        if k == True:
            labeled_true_vali +=1
    false_x_vali = 0
    true_x_vali = 0
    for n, o in zip(y_vali, predictionsVali):
        if o == False and n == False:
            false_x_vali +=1
        if o == True and n == True:
            true_x_vali +=1
    TPR_vali = float(true_x_vali) / float(labeled_true_vali)
    TNR_vali = float(false_x_vali)/ float(labeled_false_vali)
    BER_vali = 1- (0.5*(TPR_vali + TNR_vali))
    print("question four validate ber",BER_vali)

cs = [10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1, 10**2, 10**3, 10**4]
for i in cs:
    questionFour(i)
```

five- Compute the Fβ scores

```
def questionFive(beta):
    file = open('5year.arff', 'r')
    while not '@data' in file.readline():
      pass
    dataset = []
    for i in file:
        if '?' in i:
            continue
        i = i.split(',')
        values = [1] + [float(x) for x in i]
        values[-1] = values[-1] > 0
        dataset.append(values)

    random.shuffle(dataset)
    X = [values[:-1] for values in dataset]
    Y = [values[-1] for values in dataset]

    X_train, X_split, y_train, y_split = train_test_split(X, Y, test_size =0.5)
    x_vali, x_test, y_vali, y_test = train_test_split(X_split, y_split, test_size =
0.5)

    model = linear_model.LogisticRegression(class_weight='balanced', C=1)
    model.fit(X_train, y_train)
    predictionsTest = model.predict(x_test)

    false_x_test = 0
    true_x_test = 0
    false_false = 0
    false_true = 0
    for n, o in zip(y_test, predictionsTest):
        if o == False and n == False:
            false_x_test +=1
        if o == True and n == True:
            true_x_test +=1
        if o == True and n == False:
            false_true +=1
        if o == False and n == True:
            false_false += 1

    precision = float(true_x_test) / float((true_x_test + false_true))
    #print(precision)
    recall = float(true_x_test )/ float((true_x_test + false_false))
    fBeta = (float(1+ beta**2)) *
```

```
(float((precision*recall))/float((beta**2*precision+recall)))
    print(fBeta)

for i in [1, 0.1, 10]:
    questionFive(i)
```

seven - compute the PCA basis on the training set

```
import numpy
import urllib
import scipy.optimize
import random
from sklearn.decomposition import PCA # PCA library
from sklearn import linear_model
import ast

file = open('5year.arff', 'r')
while not '@data' in file.readline():
    pass
dataset = []
for i in file:
    if '?' in i:
        continue
    i = i.split(',')
    values = [1] + [float(x) for x in i]
    values[-1] = values[-1] > 0
    dataset.append(values)

random.shuffle(dataset)
X = [values[:-1] for values in dataset]
Y = [values[-1] for values in dataset]

X_train, X_split, y_train, y_split = train_test_split(X, Y, test_size =0.5)
x_vali, x_test, y_vali, y_test = train_test_split(X_split, y_split, test_size = 0.5)
pca = PCA(n_components=65)
pca.fit(X_train)
pca.components_
psi = pca.components_
print(psi[0])
```

eight- Next we'll train a model using a low-dimensional feature vector

```
def questionEight(n):
    file = open('5year.arff', 'r')
    while not '@data' in file.readline():
        pass
    dataset = []
    for i in file:
        if '?' in i:
            continue
        i = i.split(',')
        values = [1] + [float(x) for x in i]
        values[-1] = values[-1] > 0
        dataset.append(values)

    random.shuffle(dataset)
    X = [values[:-1] for values in dataset]
    Y = [values[-1] for values in dataset]

    X_train, X_split, y_train, y_split = train_test_split(X, Y, test_size =0.5)
    x_vali, x_test, y_vali, y_test = train_test_split(X_split, y_split, test_size =
0.5)
    pca = PCA(n_components=65)
    pca.fit(X_train)
    pca.components_
    psi = pca.components_
    Xpca_train = numpy.matmul(X_train, pca.components_.T)
    Xpca_valid = numpy.matmul(x_vali, pca.components_.T)
    Xpca_test = numpy.matmul(x_test, pca.components_.T)
    reduced_train = [x[:n] for x in Xpca_train]
    reduced_valid = [x[:n] for x in Xpca_valid]
    reduced_test = [x[:n] for x in Xpca_test]
    mod = linear_model.LogisticRegression(class_weight = "balanced", C = 1)
    mod.fit(reduced_train, y_train)
    predict_vlid = mod.predict(reduced_valid)
    predict_tst = mod.predict(reduced_test)
    labeled_false_test = 0
    for j in y_test:
        if j == False:
            labeled_false_test +=1
    labeled_true_test = 0
    for k in y_test:
        if k == True:
            labeled_true_test +=1
    false_x_test = 0
    true_x_test = 0
    for n, o in zip(y_test, predict_tst):
        if o == False and n == False:
            false_x_test +=1
        if o == True and n == True:
            true_x_test +=1
    TPR_test_eight = float(true_x_test) / float(labeled_true_test)
    TNR_test_eight = float(false_x_test)/ float(labeled_false_test)
    BER_test = 1- (0.5*(TPR_test_eight + TNR_test_eight))
    print(BER_test)


    labeled_true_validation = 0
    for n in y_vali:
        if n == True:
            labeled_true_validation = labeled_true_validation +1
    labeled_false_validation = 0
    for o in y_vali:
        if o == False:
            labeled_false_validation = labeled_false_validation +1

    false_x_validation = 0
    true_x_validation = 0

    for i, j in zip(y_vali, predict_vlid):
        if i == False and j == False:
            false_x_validation += 1
        if i == True and j == True:
            true_x_validation +=1

    TPR_vali_eight = float(true_x_validation) / float(labeled_true_validation)
    TNR_vali_eight = float(false_x_validation)/ float(labeled_false_validation)
    BER_validation = 1- (0.5*(TPR_vali_eight + TNR_vali_eight))
    print(BER_validation)

N = [5, 10, 15, 20, 25, 30]
for i in N:
    questionEight(i)
```