In [1]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import metrics
from sklearn.preprocessing import normalize,scale
from sklearn.decomposition import PCA

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.decomposition import PCA as sklearnPCA
from pandas.plotting import parallel_coordinates
```

In [2]:
```
#importing dataset and converting to datasframe
header = ['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle', 'sacr
al_slope', 'pelvic_radius', 'grade_of_spondylolisthesis', 'label']
df = pd.read_csv('./vertebral_column_data/column_3C.dat', sep=' ', header=N
one, names=header)
print(df.dtypes)
```

```
pelvic_incidence              float64
pelvic_tilt                   float64
lumbar_lordosis_angle         float64
sacral_slope                  float64
pelvic_radius                 float64
grade_of_spondylolisthesis    float64
label                          object
dtype: object
```

In [3]:
```
# extracting features and lables
x = df.iloc[:,0:6]
y = df.iloc[:,6]

# split into training and test subsets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4)
print(x_test.shape)
```

```
(124, 6)
```

In [4]:
```
model = KNeighborsClassifier(n_neighbors=5, weights='distance')
```

In [5]:
```
#10-fold cross validation
scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
# print scores
print ("10-Fold Accuracy : ", scores.mean()*100)
```

```
10-Fold Accuracy :  83.5483870967742
```

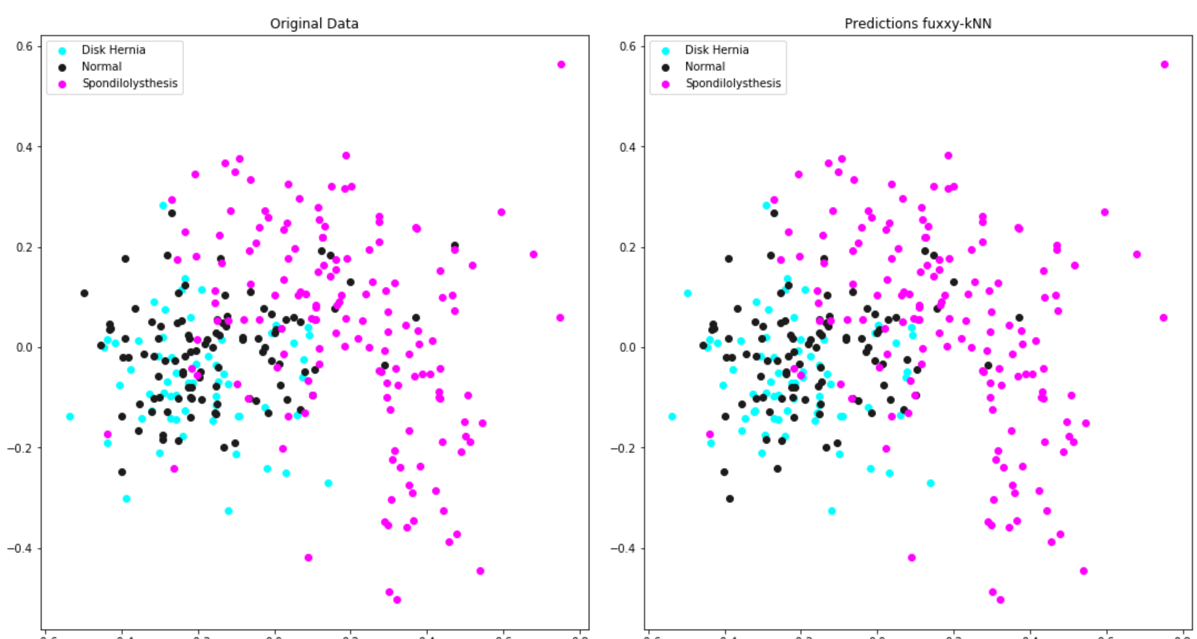
In [6]:
```
#creation of the confusion matrix
model.fit(x_train,y_train)
print ("Testing Accuracy : ",model.score(x_test, y_test)*100)

color_dict = {'DH' : 'cyan', 'SL' : 'magenta', 'NO' : '#1B1B1B'}
predicted_test = model.predict(x_test)
predicted = model.predict(x)
predicted_test_proba = model.predict_proba(x_test)
```

```
Testing Accuracy :  85.48387096774194
```

In [7]:
```
# confusion matrix
cm = metrics.confusion_matrix(y_test, predicted_test, labels=['DH', 'NO',
'SL'])
print (cm)
print()
print (metrics.classification_report(y_test, predicted_test))
```

```
[[18  4  0]
 [10 28  2]
 [ 1  1 60]]

             precision    recall  f1-score   support

         DH       0.62      0.82      0.71        22
         NO       0.85      0.70      0.77        40
         SL       0.97      0.97      0.97        62

avg / total       0.87      0.85      0.86       124
```

In [8]:
```
# mean normalization and feature scaling
x_norm = (x - x.mean())/(x.max() - x.min())

# principle component analysis
pca = PCA(n_components=2) #2-dimensional PCA
# print(pca.fit_transform(x_norm))
transformed = pd.DataFrame(pca.fit_transform(x_norm))
# print(transformed.iloc[[0,1,2],:])
# plot
fig = plt.figure(figsize=(15,8))
ax1 = fig.add_subplot(121)
ax1.scatter(transformed[y=='DH'][0], transformed[y=='DH'][1], label='Disk H
ernia', c='cyan')
ax1.scatter(transformed[y=='NO'][0], transformed[y=='NO'][1], label='Norma
l', c='#1B1B1B')
ax1.scatter(transformed[y=='SL'][0], transformed[y=='SL'][1], label='Spondi
lolysthesis', c='magenta')
ax1.legend()
ax1.set_title('Original Data')

ax2 = fig.add_subplot(122)
ax2.scatter(transformed[predicted=='DH'][0], transformed[predicted=='DH'][1
], label='Disk Hernia', c='cyan')
ax2.scatter(transformed[predicted=='NO'][0], transformed[predicted=='NO'][1
], label='Normal', c='#1B1B1B')
ax2.scatter(transformed[predicted=='SL'][0], transformed[predicted=='SL'][1
], label='Spondilolysthesis', c='magenta')
ax2.legend()
ax2.set_title('Predictions fuxxy-kNN')

plt.tight_layout(pad=0.4, w_pad=1.5, h_pad=2.0)
plt.show()
```



In [1]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import metrics
from sklearn.decomposition import PCA

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
header = ['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle', 'sacr
al_slope', 'pelvic_radius', 'grade_of_spondylolisthesis', 'label']
df = pd.read_csv(filepath_or_buffer='./vertebral_column_data/column_3C.dat'
, header=None, sep=' ', names=header)
print(df.dtypes)
```

```
pelvic_incidence              float64
pelvic_tilt                   float64
lumbar_lordosis_angle         float64
sacral_slope                  float64
pelvic_radius                 float64
grade_of_spondylolisthesis    float64
label                          object
dtype: object
```

In [3]:
```
x = df.iloc[:, 0:6]
y = df.iloc[:, 6]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4)
```

In [4]:
```
# inv of regularization constant
c = 3 #(lambda = 1/3)

model = LogisticRegression(C=c, multi_class='ovr')
```

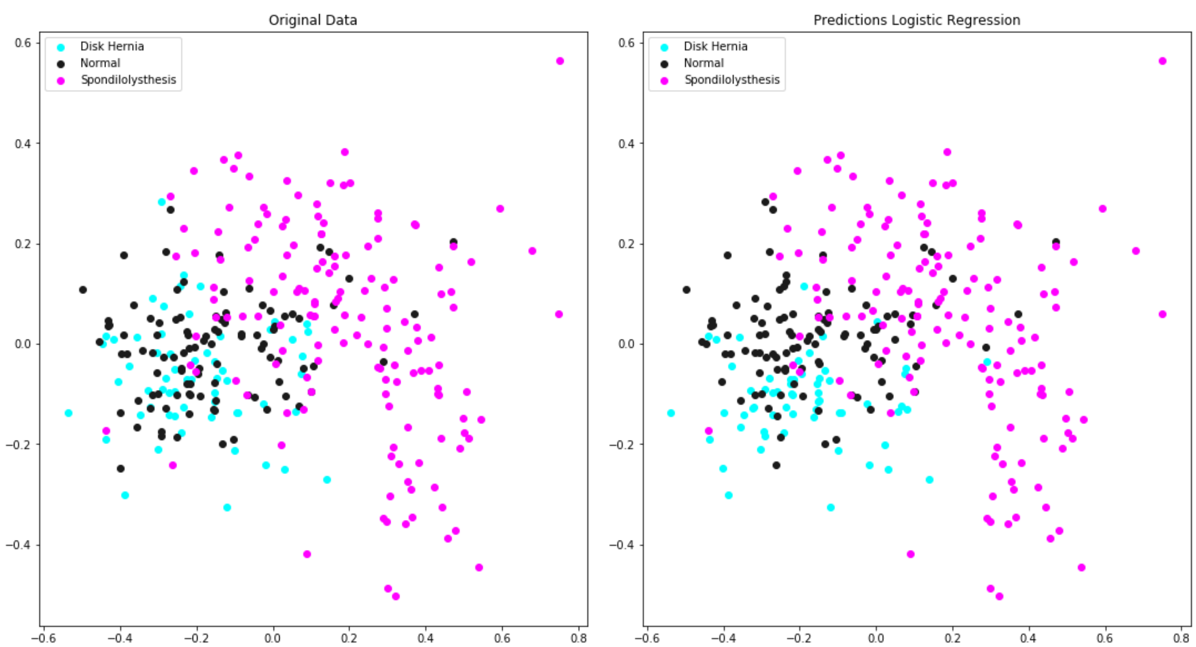
In [5]:
```
# 10 fold cross-validation
scores = cross_val_score(model, x, y, scoring='accuracy', cv=10)
print('10 fold accuracy: {}'.format(scores.mean()*100))
```

```
10 fold accuracy: 83.87096774193547
```

In [6]:
```
# creation of confusion matrix
model = model.fit(x_train, y_train)
print('Testing accuracy: {}'.format(model.score(x_test, y_test)*100))

predicted = model.predict(x)

cm = metrics.confusion_matrix(y, predicted, labels=['DH', 'NO', 'SL'])
print(cm)

print(metrics.classification_report(y, predicted))
```

```
Testing accuracy: 84.67741935483872
[[ 41  17   2]
 [ 17  81   2]
 [  2   3 145]]
             precision    recall  f1-score   support

         DH       0.68      0.68      0.68        60
         NO       0.80      0.81      0.81       100
         SL       0.97      0.97      0.97       150

avg / total       0.86      0.86      0.86       310
```

In [11]:
```
# mean normalization and feature scaling
x_norm = (x - x.mean())/(x.max() - x.min())

# principle component analysis
pca = PCA(n_components=2) #2-dimensional PCA
# print(pca.fit_transform(x_norm))
transformed = pd.DataFrame(pca.fit_transform(x_norm))
predicted = pd.DataFrame(predicted[:]).iloc[:, 0]

# print(transformed[y=='DH'])

# plot
fig = plt.figure(figsize=(15,8))
ax1 = fig.add_subplot(121)
ax1.scatter(transformed[y=='DH'][0], transformed[y=='DH'][1], label='Disk H
ernia', c='cyan')
ax1.scatter(transformed[y=='NO'][0], transformed[y=='NO'][1], label='Norma
l', c='#1B1B1B')
ax1.scatter(transformed[y=='SL'][0], transformed[y=='SL'][1], label='Spondi
lolysthesis', c='magenta')
ax1.legend()
ax1.set_title('Original Data')

ax2 = fig.add_subplot(122)
ax2.scatter(transformed[predicted=='DH'][0], transformed[predicted=='DH'][1
], label='Disk Hernia', c='cyan')
ax2.scatter(transformed[predicted=='NO'][0], transformed[predicted=='NO'][1
], label='Normal', c='#1B1B1B')
ax2.scatter(transformed[predicted=='SL'][0], transformed[predicted=='SL'][1
], label='Spondilolysthesis', c='magenta')
ax2.legend()
ax2.set_title('Predictions Logistic Regression')

plt.tight_layout(pad=0.4, w_pad=1.5, h_pad=2.0)
plt.show()
```



In [1]:
```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics

import numpy as np
import pandas as pd
```

In [2]:
```
dataset = pd.read_csv('./vertebral_column_data/column_3C.dat', delimiter =
' ', header = None)
X = dataset.iloc[:,0:-1].values
Y = dataset.iloc[:,-1].values
```

In [3]:
```
le = LabelEncoder()
Y = le.fit_transform(Y)
#Y = np.reshape(Y,(310,1))
#onehotencoder = OneHotEncoder(categorical_features=[0])
#Y = onehotencoder.fit_transform(Y).toarray()

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2,random
_state = 0)
```

In [4]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = RandomForestClassifier(n_estimators=10, criterion = 'entropy',
random_state = 0)
classsifier = classifier.fit(X_train,Y_train)
```

In [5]:
```
Y_pred = classifier.predict(X_test)

reverse = dict(zip(range(3),['DH','NO','SL']))
Y_test = np.vectorize(reverse.get)(Y_test)
Y_pred = np.vectorize(reverse.get)(Y_pred)

cm = confusion_matrix(Y_test, Y_pred)

print(cm)
print(metrics.classification_report(Y_test, Y_pred))
```

```
[[ 3  9  1]
 [ 1 16  2]
 [ 0  1 29]]
             precision    recall  f1-score   support

         DH       0.75      0.23      0.35        13
         NO       0.62      0.84      0.71        19
         SL       0.91      0.97      0.94        30

avg / total       0.78      0.77      0.74        62
```

In [1]:
```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder, StandardScale
r
from sklearn.model_selection import train_test_split

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import keras
from keras.layers import Dense
from keras.models import Sequential
```

```
Using TensorFlow backend.
```

In [2]:
```
dataset = pd.read_csv('./vertebral_column_data/column_3C.dat', delimiter =
' ', header = None)
X = dataset.iloc[:,0:-1].values
Y = dataset.iloc[:,-1].values

le = LabelEncoder()
Y = le.fit_transform(Y)
Y = np.reshape(Y,(310,1))
onehotencoder = OneHotEncoder(categorical_features=[0])
Y = onehotencoder.fit_transform(Y).toarray()
```

In [3]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2,random
_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = Sequential()

classifier.add(Dense(units=5, input_dim = 6, kernel_initializer='uniform',
activation='relu'))

classifier.add(Dense(units=4, kernel_initializer='uniform', activation='rel
u'))

classifier.add(Dense(units=3,  kernel_initializer='uniform', activation='so
ftmax'))

classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy',me
trics = ['accuracy'] )
```

In [4]:
```
classifier.fit(X_train,Y_train,epochs = 100, batch_size = 10)
```

```
Epoch 1/100
248/248 [==============================] - 0s 951us/step - loss: 1.0961 - a
cc: 0.4718
Epoch 2/100
248/248 [==============================] - 0s 118us/step - loss: 1.0910 - a
cc: 0.4839
Epoch 3/100
248/248 [==============================] - 0s 144us/step - loss: 1.0857 - a
cc: 0.4839
Epoch 4/100
248/248 [==============================] - 0s 165us/step - loss: 1.0784 - a
cc: 0.4839
Epoch 5/100
248/248 [==============================] - 0s 128us/step - loss: 1.0664 - a
cc: 0.4839
Epoch 6/100
248/248 [==============================] - 0s 117us/step - loss: 1.0460 - a
cc: 0.4839
Epoch 7/100
248/248 [==============================] - 0s 127us/step - loss: 1.0136 - a
cc: 0.5081
Epoch 8/100
248/248 [==============================] - 0s 128us/step - loss: 0.9663 - a
cc: 0.6210
Epoch 9/100
248/248 [==============================] - 0s 126us/step - loss: 0.9105 - a
cc: 0.6774
Epoch 10/100
248/248 [==============================] - 0s 150us/step - loss: 0.8507 - a
cc: 0.7016
Epoch 11/100
248/248 [==============================] - 0s 129us/step - loss: 0.7976 - a
cc: 0.7177
Epoch 12/100
248/248 [==============================] - 0s 117us/step - loss: 0.7513 - a
cc: 0.7097
Epoch 13/100
248/248 [==============================] - 0s 137us/step - loss: 0.7165 - a
cc: 0.6895
Epoch 14/100
248/248 [==============================] - 0s 121us/step - loss: 0.6892 - a
cc: 0.6935
Epoch 15/100
248/248 [==============================] - ETA: 0s - loss: 0.5978 - acc: 0.
700 - 0s 135us/step - loss: 0.6687 - acc: 0.6895
Epoch 16/100
248/248 [==============================] - 0s 130us/step - loss: 0.6530 - a
cc: 0.6935
Epoch 17/100
248/248 [==============================] - 0s 121us/step - loss: 0.6395 - a
cc: 0.6976
Epoch 18/100
248/248 [==============================] - 0s 131us/step - loss: 0.6279 - a
cc: 0.7016
Epoch 19/100
248/248 [==============================] - 0s 118us/step - loss: 0.6186 - a
cc: 0.7016
```

```
Epoch 20/100
248/248 [==============================] - 0s 127us/step - loss: 0.6091 - a
cc: 0.7056
Epoch 21/100
248/248 [==============================] - 0s 131us/step - loss: 0.6008 - a
cc: 0.7056
Epoch 22/100
248/248 [==============================] - 0s 159us/step - loss: 0.5931 - a
cc: 0.7097
Epoch 23/100
248/248 [==============================] - 0s 123us/step - loss: 0.5855 - a
cc: 0.7097
Epoch 24/100
248/248 [==============================] - 0s 120us/step - loss: 0.5770 - a
cc: 0.7137
Epoch 25/100
248/248 [==============================] - 0s 116us/step - loss: 0.5689 - a
cc: 0.7218
Epoch 26/100
248/248 [==============================] - 0s 154us/step - loss: 0.5601 - a
cc: 0.7218
Epoch 27/100
248/248 [==============================] - 0s 116us/step - loss: 0.5505 - a
cc: 0.7258
Epoch 28/100
248/248 [==============================] - 0s 132us/step - loss: 0.5399 - a
cc: 0.7258
Epoch 29/100
248/248 [==============================] - 0s 127us/step - loss: 0.5308 - a
cc: 0.7339
Epoch 30/100
248/248 [==============================] - 0s 121us/step - loss: 0.5212 - a
cc: 0.7419
Epoch 31/100
248/248 [==============================] - 0s 122us/step - loss: 0.5118 - a
cc: 0.7379
Epoch 32/100
248/248 [==============================] - 0s 130us/step - loss: 0.5030 - a
cc: 0.7419
Epoch 33/100
248/248 [==============================] - 0s 115us/step - loss: 0.4946 - a
cc: 0.7460
Epoch 34/100
248/248 [==============================] - 0s 122us/step - loss: 0.4870 - a
cc: 0.7460
Epoch 35/100
248/248 [==============================] - 0s 115us/step - loss: 0.4800 - a
cc: 0.7460
Epoch 36/100
248/248 [==============================] - 0s 132us/step - loss: 0.4729 - a
cc: 0.7581
Epoch 37/100
248/248 [==============================] - 0s 135us/step - loss: 0.4662 - a
cc: 0.7621
Epoch 38/100
248/248 [==============================] - 0s 138us/step - loss: 0.4600 - a
cc: 0.7661
```

```
Epoch 39/100
248/248 [==============================] - 0s 119us/step - loss: 0.4542 - a
cc: 0.7702
Epoch 40/100
248/248 [==============================] - 0s 136us/step - loss: 0.4490 - a
cc: 0.7742
Epoch 41/100
248/248 [==============================] - 0s 112us/step - loss: 0.4439 - a
cc: 0.7782
Epoch 42/100
248/248 [==============================] - 0s 128us/step - loss: 0.4392 - a
cc: 0.7823
Epoch 43/100
248/248 [==============================] - 0s 120us/step - loss: 0.4357 - a
cc: 0.7823
Epoch 44/100
248/248 [==============================] - 0s 116us/step - loss: 0.4318 - a
cc: 0.7823
Epoch 45/100
248/248 [==============================] - 0s 122us/step - loss: 0.4286 - a
cc: 0.7823
Epoch 46/100
248/248 [==============================] - 0s 112us/step - loss: 0.4253 - a
cc: 0.7823
Epoch 47/100
248/248 [==============================] - 0s 127us/step - loss: 0.4231 - a
cc: 0.7863
Epoch 48/100
248/248 [==============================] - 0s 118us/step - loss: 0.4203 - a
cc: 0.7863
Epoch 49/100
248/248 [==============================] - 0s 116us/step - loss: 0.4181 - a
cc: 0.7863
Epoch 50/100
248/248 [==============================] - 0s 128us/step - loss: 0.4151 - a
cc: 0.7863
Epoch 51/100
248/248 [==============================] - 0s 124us/step - loss: 0.4127 - a
cc: 0.7863
Epoch 52/100
248/248 [==============================] - 0s 103us/step - loss: 0.4111 - a
cc: 0.7863
Epoch 53/100
248/248 [==============================] - 0s 113us/step - loss: 0.4091 - a
cc: 0.7863
Epoch 54/100
248/248 [==============================] - 0s 127us/step - loss: 0.4074 - a
cc: 0.7863
Epoch 55/100
248/248 [==============================] - 0s 95us/step - loss: 0.4051 - ac
c: 0.7863
Epoch 56/100
248/248 [==============================] - 0s 141us/step - loss: 0.4030 - a
cc: 0.7863
Epoch 57/100
248/248 [==============================] - 0s 116us/step - loss: 0.4008 - a
cc: 0.7863
```

```
Epoch 58/100
248/248 [==============================] - 0s 109us/step - loss: 0.4000 - a
cc: 0.7863
Epoch 59/100
248/248 [==============================] - 0s 130us/step - loss: 0.3976 - a
cc: 0.7903
Epoch 60/100
248/248 [==============================] - 0s 116us/step - loss: 0.3953 - a
cc: 0.7863
Epoch 61/100
248/248 [==============================] - 0s 119us/step - loss: 0.3928 - a
cc: 0.8024
Epoch 62/100
248/248 [==============================] - 0s 116us/step - loss: 0.3911 - a
cc: 0.8065
Epoch 63/100
248/248 [==============================] - 0s 126us/step - loss: 0.3894 - a
cc: 0.8065
Epoch 64/100
248/248 [==============================] - 0s 109us/step - loss: 0.3874 - a
cc: 0.8024
Epoch 65/100
248/248 [==============================] - 0s 114us/step - loss: 0.3860 - a
cc: 0.8065
Epoch 66/100
248/248 [==============================] - 0s 110us/step - loss: 0.3839 - a
cc: 0.8145
Epoch 67/100
248/248 [==============================] - 0s 110us/step - loss: 0.3830 - a
cc: 0.8185
Epoch 68/100
248/248 [==============================] - 0s 109us/step - loss: 0.3806 - a
cc: 0.8185
Epoch 69/100
248/248 [==============================] - 0s 115us/step - loss: 0.3801 - a
cc: 0.8145
Epoch 70/100
248/248 [==============================] - 0s 110us/step - loss: 0.3772 - a
cc: 0.8226
Epoch 71/100
248/248 [==============================] - 0s 112us/step - loss: 0.3752 - a
cc: 0.8185
Epoch 72/100
248/248 [==============================] - 0s 117us/step - loss: 0.3742 - a
cc: 0.8226
Epoch 73/100
248/248 [==============================] - 0s 127us/step - loss: 0.3710 - a
cc: 0.8266
Epoch 74/100
248/248 [==============================] - 0s 119us/step - loss: 0.3699 - a
cc: 0.8226
Epoch 75/100
248/248 [==============================] - 0s 109us/step - loss: 0.3677 - a
cc: 0.8266
Epoch 76/100
248/248 [==============================] - 0s 114us/step - loss: 0.3668 - a
cc: 0.8266
```

```
Epoch 77/100
248/248 [==============================] - 0s 118us/step - loss: 0.3641 - a
cc: 0.8266
Epoch 78/100
248/248 [==============================] - 0s 113us/step - loss: 0.3625 - a
cc: 0.8266
Epoch 79/100
248/248 [==============================] - 0s 107us/step - loss: 0.3595 - a
cc: 0.8226
Epoch 80/100
248/248 [==============================] - 0s 117us/step - loss: 0.3574 - a
cc: 0.8226
Epoch 81/100
248/248 [==============================] - 0s 112us/step - loss: 0.3560 - a
cc: 0.8266
Epoch 82/100
248/248 [==============================] - 0s 119us/step - loss: 0.3558 - a
cc: 0.8387
Epoch 83/100
248/248 [==============================] - 0s 114us/step - loss: 0.3511 - a
cc: 0.8306
Epoch 84/100
248/248 [==============================] - 0s 112us/step - loss: 0.3498 - a
cc: 0.8266
Epoch 85/100
248/248 [==============================] - 0s 119us/step - loss: 0.3480 - a
cc: 0.8306
Epoch 86/100
248/248 [==============================] - 0s 110us/step - loss: 0.3459 - a
cc: 0.8347
Epoch 87/100
248/248 [==============================] - 0s 110us/step - loss: 0.3430 - a
cc: 0.8347
Epoch 88/100
248/248 [==============================] - 0s 106us/step - loss: 0.3413 - a
cc: 0.8387
Epoch 89/100
248/248 [==============================] - 0s 122us/step - loss: 0.3385 - a
cc: 0.8387
Epoch 90/100
248/248 [==============================] - 0s 102us/step - loss: 0.3374 - a
cc: 0.8427
Epoch 91/100
248/248 [==============================] - 0s 120us/step - loss: 0.3342 - a
cc: 0.8387
Epoch 92/100
248/248 [==============================] - 0s 107us/step - loss: 0.3338 - a
cc: 0.8468
Epoch 93/100
248/248 [==============================] - 0s 121us/step - loss: 0.3308 - a
cc: 0.8508
Epoch 94/100
248/248 [==============================] - 0s 121us/step - loss: 0.3285 - a
cc: 0.8589
Epoch 95/100
248/248 [==============================] - 0s 109us/step - loss: 0.3272 - a
cc: 0.8710
```

```
Epoch 96/100
248/248 [==============================] - 0s 109us/step - loss: 0.3245 - a
cc: 0.8831
Epoch 97/100
248/248 [==============================] - 0s 109us/step - loss: 0.3230 - a
cc: 0.8710
Epoch 98/100
248/248 [==============================] - 0s 120us/step - loss: 0.3209 - a
cc: 0.8871
Epoch 99/100
248/248 [==============================] - 0s 98us/step - loss: 0.3194 - ac
c: 0.8790
Epoch 100/100
248/248 [==============================] - 0s 114us/step - loss: 0.3171 - a
cc: 0.8750
```

```
Out[4]: <keras.callbacks.History at 0x7fc05b9ded68>
```

In [5]:
```
Y_pred = classifier.predict(X_test)
Y_pred = (Y_pred>0.5).astype(float)
print(Y_pred)
```

```
[[0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
```

```
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]]
```