TAREFA 1

Nome: Roberto Alejandro Hidalgo Castro
RA: 164787

1. Code description

The code was developed according to the instructions specified in the homework description. This code was developed in Python 2.7.11 and using some functions in sklearn. Following, the code will be shown divided according to the questions.

Lecture of the input

First, we make the lecture of the file `'data1.csv'`. Then, we transfer the data to a matrix called 'matrix' and the classes to each data (0 or 1) to 'values'.

```
# Reading the input from data1.csv
my_data = genfromtxt('data1.csv', delimiter=',')
#Adjusting the input into a matrix.
matrix = my_data[1:477,0:166]
values = my_data[1:477,166:167]
```

PCA of the data, keeping 80% of the variance

First, we create a Principal Component Analysis (PCA) model, giving as parameter the value 0.8 to keep the 80% of variance. Then, we fit the model with the data ('matrix') and apply the dimensionality reduction on 'matrix'. Finally, we print the number of dimensions that we need to keep.

```
# PCA model, keeping 80% of variance
pca = PCA(0.8)
matrix_PCA = pca.fit_transform(matrix)

#Number of dimensions
print "Número de dimensões:"
print matrix_PCA.shape[1]
```

Logistic regression for the training of the original data and the transformed data

First, we separate the data: the first 200 lines will be the training data and the last 276 lines will be the data for the prediction. The training data will be stored in 'training_regres' and its classes in 'values_regres'. The data for prediction will be stored in 'data_test' and its classes in 'data_values'.

```
# Considering the last 276 lines from the data as the data for
prediction
data_test = matrix[200:476]
data_values = values[200:476].reshape(276,)

# Considering the first 200 lines from the data as the data for training
training_regres = matrix[0:200]
values_regres = values[0:200].reshape(200,)
```

Then, we are going to calculate the transformed data for training and prediction with the PCA function, as we have done before. Being the transformed data for training 'training_regres_PCA' and the transformed data for prediction 'data_test_PCA'.

```
# Creating the PCA model, transforming the training data and data for
prediction
pca = PCA(0.8)
training_regres_PCA = pca.fit_transform(training_regres)
data_test_PCA = pca.fit_transform(data_test)
```

Now, we will create a logistic regression model. Then, make the training with the original data (data for training) 'training_regres' and its classes 'values_regres'. Once this have been done, we will make the prediction for 'data_test' and store it in 'predicted_values_regres'. Finally, we print the hit rate (“taxa de acerto”) for the original data.

```
# Logistic regression, training and prediction of the original data
logreg = LogisticRegression() # Logistic regression
logreg.fit(training_regres, values_regres) # Training
predicted_values_regres = logreg.predict(data_test) # Prediction

# Taxa de acerto of logistic regression with the original data
print "Regressão logística dos dados originais (no PCA)"
print accuracy_score(data_values, predicted_values_regres)
```

Then, we will make the same process but for the transformed data. We create a logistic regression model. Then, make the training with the transformed data 'training_regres_PCA' and its classes 'values_regres'. Once this have finished, we will make the prediction for 'data_test_PCA' and store it in 'predicted_values_regres_PCA'. Finally, we print the hit rate (“taxa de acerto”) for the transformed data.

```
# Logistic regression, training and prediction of the transformed data
with the PCA model
logreg = LogisticRegression() # Logistic regression
logreg.fit(training_regres_PCA, values_regres) # Training
```

```
predicted_values_regres_PCA = logreg.predict(data_test_PCA) # Prediction

# Taxa de acerto of the logistic regression with the transformed data
(with the PCA)
print "Regressão logística dos dados transformados com PCA"
print accuracy_score(data_values, predicted_values_regres_PCA)
```

LDA for the training of the original data and the transformed data

As we separated the data before: the first 200 lines will be the training data and the last 276 lines will be the data for the prediction. In this case, the training data will stored in 'training_LDA' and its classes in 'values_LDA'. The data for prediction will be stored in 'data_test' and its classes in 'data_values'.

```
# Considering the first 200 lines from the data as the data for training
training_LDA = matrix[0:200]
values_LDA = values[0:200].reshape(200,)
```

Now, we are going to calculate the transformed data for training and prediction with the PCA function, as we have done in the previous point. Being the transformed data for training 'training_LDA_PCA' and the transformed data for prediction 'data_test_PCA'.

```
# Creating the PCA model, transforming the training data and data for
prediction
pca = PCA(0.8)
training_LDA_PCA = pca.fit_transform(training_LDA)
data_test_PCA = pca.fit_transform(data_test)
```

Then, we will create a LDA model. Make the training with the original data for training 'training_LDA' and its classes 'values_LDA'. Once this have been done, we will make the prediction for 'data_test' and store it in 'predicted_values_LDA'. Finally, we print the accuracy for the original data.

```
# LDA, training and prediction of the original data
clf = LDA()#LDA
clf.fit(training_LDA, values_LDA) # training
predicted_values_LDA = clf.predict(data_test) # prediction

# Accuracy of LDA with the original data
print "LDA dos dados originais (no PCA)"
print accuracy_score(data_values, predicted_values_LDA)
```

Finally, we will make the same process but for the transformed data. We create a LDA model. Then,

make the training with the transformed data 'training_LDA_PCA' and the classes 'values_LDA'. Once this have finished, we will make the prediction for 'data_test_PCA' and store it in 'predicted_values_LDA_PCA'. Finally, we print the accuracy for the transformed data.

```
# LDA, training and prediction of the transformed data with the PCA
model
clf = LDA() # LDA
predicted_values_LDA_PCA = clf.predict(data_test_PCA) # Prediction

# Accuracy of LDA with the transformed data (with PCA)
print "LDA dos dados transformados com PCA"
print accuracy_score(data_values, predicted_values_LDA_PCA)
```

2. Answers

Now, we will show the answers obtained from the implementation shown before:

2.1. The number of dimensions is 13.
2.2. The hit rate (“taxa de acerto”) in the logistic regression model for the original data is 0.797101449275 and for the transformed data with the PCA model is 0.518115942029.
2.3. The accuracy for the LDA model for the original data is 0.677536231884 and for the transformed data with the PCA model is 0.485507246377.
2.4. The best combination is the logistic regression model with the original data (without the PCA model), because it has the highest accuracy.