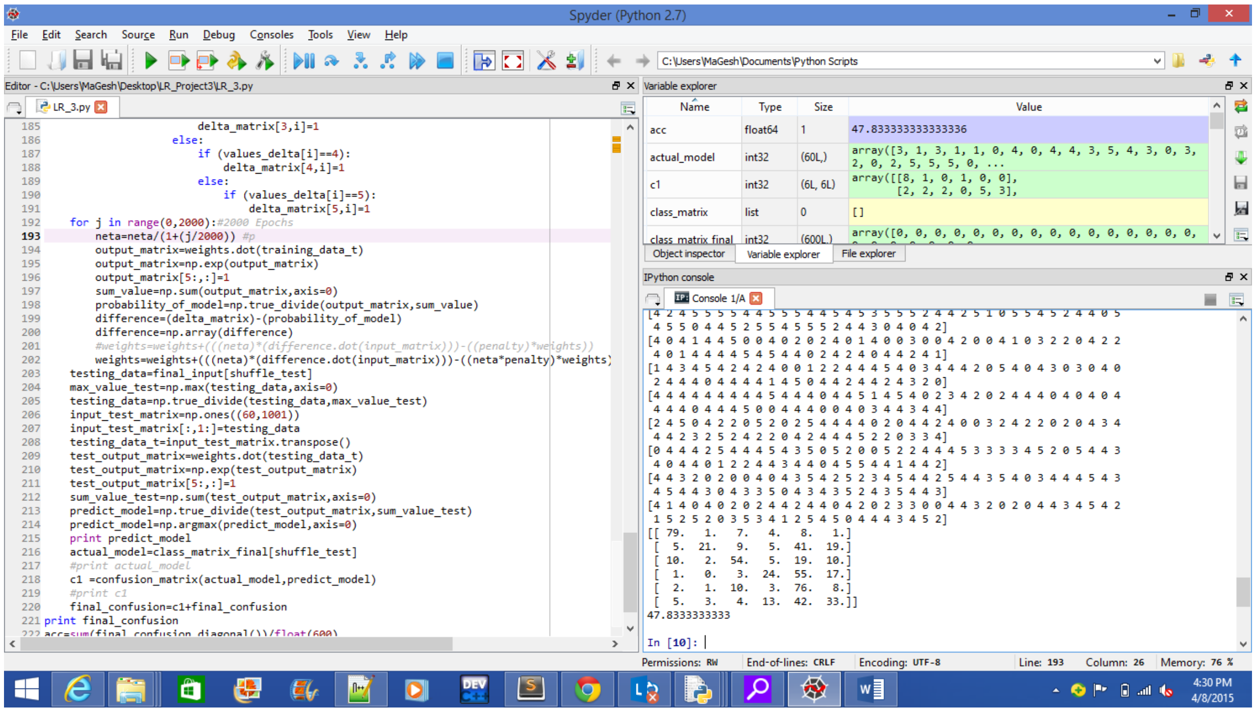
<u>FFT OUTPUT:</u>

1) Implemented Logistic Regression and Gradient Descent in which I have used the FFT components and taken the first 1000 features of it.
2) In order to train and test the data I have used K Fold cross validation in which I have had 10 Folds.
3) Please find the accuracy below

ACCURACY Percentage is 47.833%

You can see the matrix below:



Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Tools View Help
C:\Users\MaGesh\Documents\Python Scripts
Editor - C:\Users\MaGesh\Desktop\LR_Project3\LR_3.py
LR_3.py
Variable explorer
Object inspector
Variable explorer
File explorer
IPython console
Console 1/A

```python
                        delta_matrix[3,i]=1
                    else:
                        if (values_delta[i]==4):
                            delta_matrix[4,i]=1
                        else:
                            if (values_delta[i]==5):
                                delta_matrix[5,i]=1
    for j in range(0,2000):#2000 Epochs
        neta=neta/(1+(j/2000)) #p
        output_matrix=weights.dot(training_data_t)
        output_matrix=np.exp(output_matrix)
        output_matrix[5:,:]=1
        sum_value=np.sum(output_matrix,axis=0)
        probability_of_model=np.true_divide(output_matrix,sum_value)
        difference=(delta_matrix)-(probability_of_model)
        difference=np.array(difference)
        #weights=weights+(((neta)*(difference.dot(input_matrix)))-((penalty)*weights))
        weights=weights+(((neta)*(difference.dot(input_matrix)))-((neta*penalty)*weights)
    testing_data=final_input[shuffle_test]
    max_value_test=np.max(testing_data,axis=0)
    testing_data=np.true_divide(testing_data,max_value_test)
    input_test_matrix=np.ones((60,1001))
    input_test_matrix[:,1:]=testing_data
    testing_data_t=input_test_matrix.transpose()
    test_output_matrix=weights.dot(testing_data_t)
    test_output_matrix=np.exp(test_output_matrix)
    test_output_matrix[5:,:]=1
    sum_value_test=np.sum(test_output_matrix,axis=0)
    predict_model=np.true_divide(test_output_matrix,sum_value_test)
    predict_model=np.argmax(predict_model,axis=0)
    print predict_model
    actual_model=class_matrix_final[shuffle_test]
    #print actual_model
    c1 =confusion_matrix(actual_model,predict_model)
    #print c1
    final_confusion=c1+final_confusion
print final_confusion
acc=sum(final_confusion.diagonal())/float(600)
```

Name	Type	Size	Value
acc	float64	1	47.833333333333336
actual_model	int32	(60L,)	array([3, 1, 3, 1, 1, 0, 4, 0, 4, 4, 3, 5, 4, 3, 0, 3, 2, 0, 2, 5, 5, 5, 0, ...
c1	int32	(6L, 6L)	array([[8, 1, 0, 1, 0, 0], [2, 2, 2, 0, 5, 3],
class_matrix	list	0	[]
class_matrix_final	int32	(600L,)	array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
[4 2 4 5 5 5 5 4 4 5 5 5 4 4 5 4 5 3 5 5 5 2 4 4 2 5 1 0 5 5 4 5 2 4 4 0 5
 4 5 5 0 4 4 5 2 5 5 4 5 5 5 2 4 4 3 0 4 0 4 2]
[4 0 4 1 4 4 5 0 0 4 0 2 0 2 4 0 1 4 0 0 3 0 0 4 2 0 0 4 1 0 3 2 2 0 4 2 2
 4 0 1 4 4 4 4 5 4 5 4 4 0 2 4 2 4 0 4 4 2 4 1]
[1 4 3 4 5 4 2 4 2 4 0 0 1 2 2 4 4 4 5 4 0 3 4 4 4 2 0 5 4 0 4 3 0 3 0 4 0
 2 4 4 4 0 4 4 4 4 1 4 5 0 4 4 2 4 4 2 4 3 2 0]
[4 4 4 4 4 4 4 4 4 5 4 4 4 0 4 4 5 1 4 5 4 0 2 3 4 2 0 2 4 4 4 0 4 0 4 0 4
 4 4 4 0 4 4 4 5 0 0 4 4 4 0 0 4 0 3 4 4 3 4 4]
[2 4 5 0 4 2 2 0 5 2 0 2 5 4 4 4 4 0 2 0 4 4 2 4 0 0 3 2 4 2 2 0 2 0 4 3 4
 4 4 2 3 2 5 2 4 2 2 0 4 2 4 4 4 5 2 2 0 3 3 4]
[0 4 4 4 2 5 4 4 4 5 4 3 5 0 5 2 0 0 5 2 2 4 4 4 5 3 3 3 3 4 5 2 0 5 4 4 3
 4 0 4 4 0 1 2 2 4 4 3 4 4 0 4 5 5 4 4 1 4 4 2]
[4 4 3 2 0 2 0 0 4 0 4 3 5 4 2 5 2 3 4 5 4 4 2 5 4 4 3 5 4 0 3 4 4 4 5 4 3
 4 5 4 4 3 0 4 3 3 5 0 4 3 4 3 5 2 4 3 5 4 4 3]
[4 1 4 0 4 0 2 0 2 4 4 2 4 4 0 4 2 0 2 3 3 0 0 4 4 3 2 0 2 0 4 4 3 4 5 4 2
 1 5 2 5 2 0 3 5 3 4 1 2 5 4 5 0 4 4 4 3 4 5 2]
[[ 79.   1.   7.   4.   8.   1.]
 [  5.  21.   9.   5.  41.  19.]
 [ 10.   2.  54.   5.  19.  10.]
 [  1.   0.   3.  24.  55.  17.]
 [  2.   1.  10.   3.  76.   8.]
 [  5.   3.   4.  13.  42.  33.]]
47.8333333333

In [10]:
```

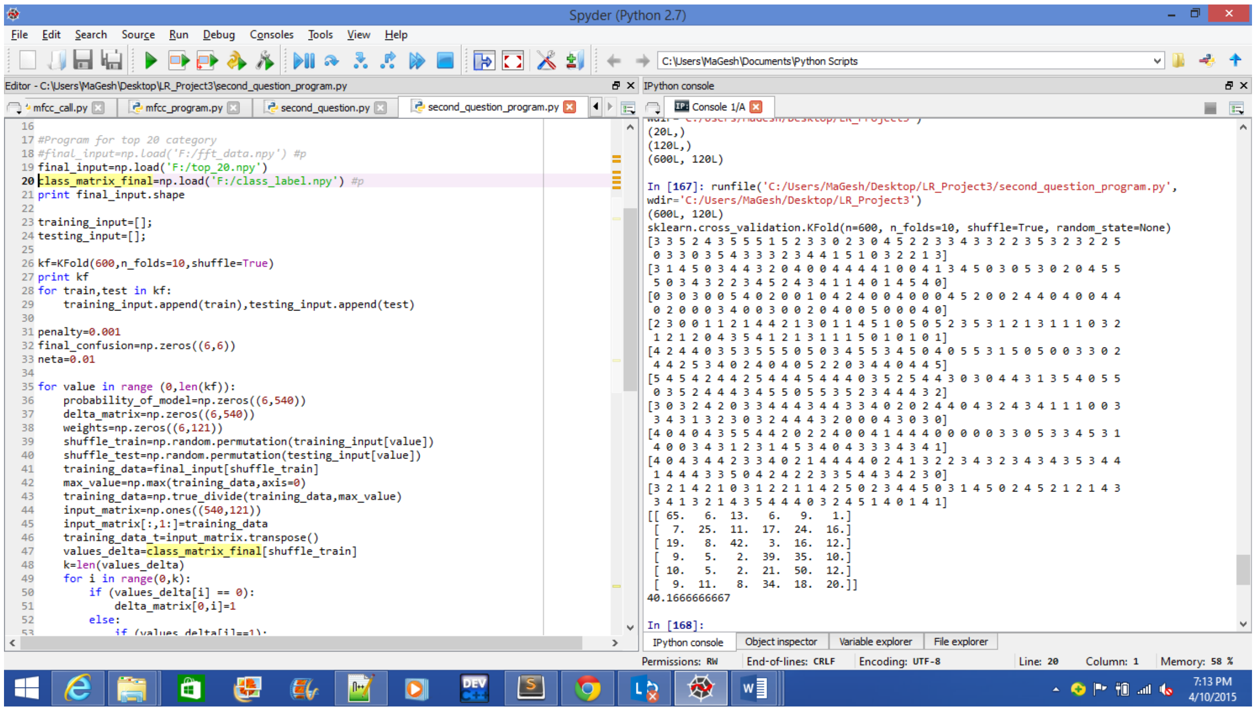
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 193 Column: 26 Memory: 76 %

4:30 PM 4/8/2015

4) I have got accuracy as 47.833% when classifying the training and testing data, I think per your comments, we have taken only 1000 features of the data out of 600,000 features. So, the attributes are very small when classifying, therefore even I train the system by having 2000 epochs I didn’t get more than 50% of the data correctly classified.

Top 20 Features Output:

Accuracy: 40.1%



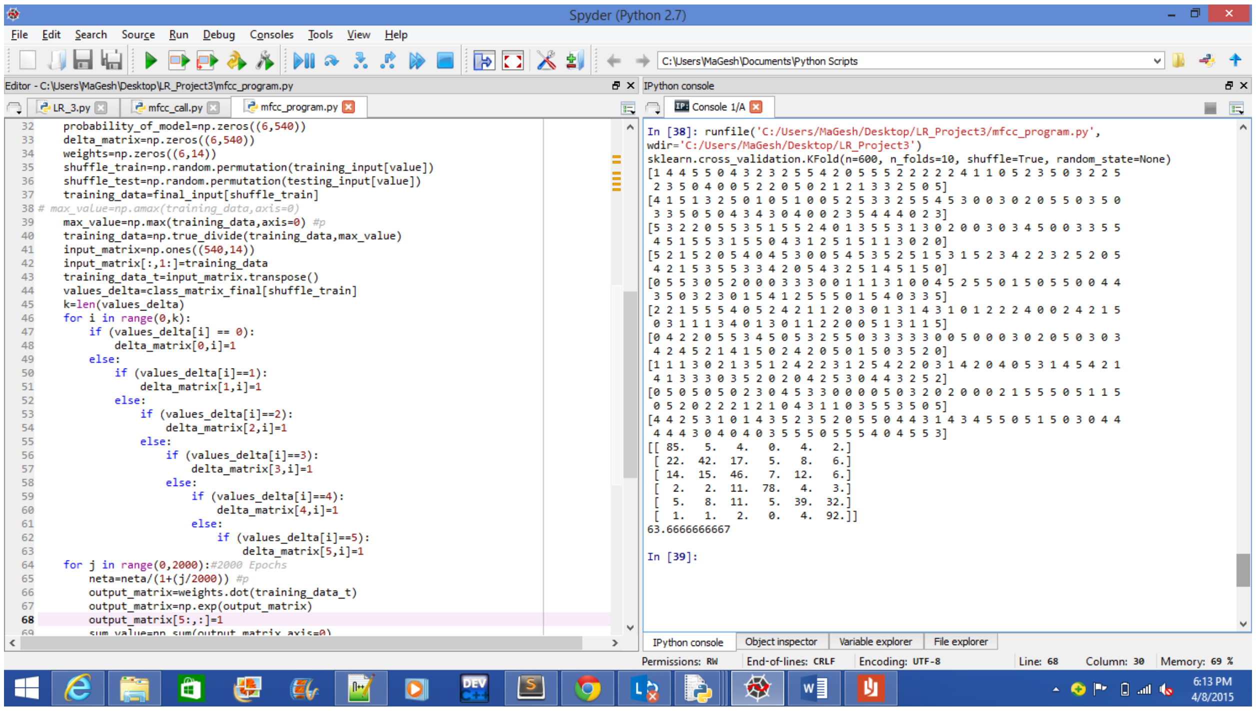
```python

#Program for top 20 category
#final_input=np.load('F:/fft_data.npy') #p
final_input=np.load('F:/top_20.npy')
class_matrix_final=np.load('F:/class_label.npy') #p
print final_input.shape

training_input=[];
testing_input=[];

kf=KFold(600,n_folds=10,shuffle=True)
print kf
for train,test in kf:
    training_input.append(train),testing_input.append(test)

penalty=0.001
final_confusion=np.zeros((6,6))
neta=0.01

for value in range (0,len(kf)):
    probability_of_model=np.zeros((6,540))
    delta_matrix=np.zeros((6,540))
    weights=np.zeros((6,121))
    shuffle_train=np.random.permutation(training_input[value])
    shuffle_test=np.random.permutation(testing_input[value])
    training_data=final_input[shuffle_train]
    max_value=np.max(training_data,axis=0)
    training_data=np.true_divide(training_data,max_value)
    input_matrix=np.ones((540,121))
    input_matrix[:,1:]=training_data
    training_data_t=input_matrix.transpose()
    values_delta=class_matrix_final[shuffle_train]
    k=len(values_delta)
    for i in range(0,k):
        if (values_delta[i] == 0):
            delta_matrix[0,i]=1
        else:
```

```
(20L,)
(120L,)
(600L, 120L)

In [167]: runfile('C:/Users/MaGesh/Desktop/LR_Project3/second_question_program.py',
wdir='C:/Users/MaGesh/Desktop/LR_Project3')
(600L, 120L)
sklearn.cross_validation.KFold(n=600, n_folds=10, shuffle=True, random_state=None)
[3 3 5 2 4 3 5 5 5 1 5 2 3 3 0 2 3 0 4 5 2 2 3 3 4 3 3 2 2 3 5 3 2 3 2 2 5
 0 3 3 0 3 5 4 3 3 3 2 3 4 4 1 5 1 0 3 2 2 1 3]
[3 1 4 5 0 3 4 4 3 2 0 4 0 0 4 4 4 4 1 0 0 4 1 3 4 5 0 3 0 5 3 0 2 0 4 5 5
 5 0 3 4 3 2 2 3 4 5 2 4 3 4 1 1 4 0 1 4 5 4 0]
[0 3 0 3 0 0 5 4 0 2 0 0 1 0 4 2 4 0 0 4 0 0 0 4 5 2 0 0 2 4 4 0 4 0 0 4 4
 0 2 0 0 0 3 4 0 0 3 0 0 2 0 4 0 0 5 0 0 0 4 0]
[2 3 0 0 1 1 2 1 4 4 2 1 3 0 1 1 4 5 1 0 5 0 5 2 3 5 3 1 2 1 3 1 1 1 0 3 2
 1 2 1 2 0 4 3 5 4 1 2 1 3 1 1 1 5 0 1 0 1 0 1]
[4 2 4 4 0 3 5 3 5 5 5 0 5 0 3 4 5 5 3 4 5 0 4 0 5 5 3 1 5 0 5 0 0 3 3 0 2
 4 4 2 5 3 4 0 2 4 0 4 0 5 2 2 0 3 4 4 0 4 4 5]
[5 4 5 4 2 4 4 2 5 4 4 4 5 4 4 4 0 3 5 2 5 4 4 3 0 3 0 4 4 3 1 3 5 4 0 5 5
 0 3 5 2 4 4 4 3 4 5 5 0 5 5 3 5 2 3 4 4 4 3 2]
[3 0 3 2 4 2 0 3 3 4 4 4 3 4 4 3 3 4 0 2 0 2 4 4 0 4 3 2 4 3 4 1 1 1 0 0 3
 3 4 3 1 3 2 3 0 3 2 4 4 4 3 2 0 0 0 4 3 0 3 0]
[4 0 4 0 4 3 5 5 4 4 2 0 2 2 4 0 0 4 1 4 4 4 0 0 0 0 0 3 3 0 5 3 3 4 5 3 1
 4 0 0 3 4 3 1 2 3 1 4 5 3 4 0 4 3 3 3 4 3 4 1]
[4 0 4 3 4 4 2 3 3 4 0 2 1 4 4 4 4 0 2 4 1 3 2 2 3 4 3 2 3 4 3 4 3 5 3 4 4
 1 4 4 4 3 3 5 0 4 2 4 2 2 3 3 5 4 4 3 4 2 3 0]
[3 2 1 4 2 1 0 3 1 2 2 1 1 4 2 5 0 2 3 4 4 5 0 3 1 4 5 0 2 4 5 2 1 2 1 4 3
 3 4 1 3 2 1 4 3 5 4 4 4 0 3 2 4 5 1 4 0 1 4 1]
[[ 65.   6.  13.   6.   9.   1.]
 [  7.  25.  11.  17.  24.  16.]
 [ 19.   8.  42.   3.  16.  12.]
 [  9.   5.   2.  39.  35.  10.]
 [ 10.   5.   2.  21.  50.  12.]
 [  9.  11.   8.  34.  18.  20.]]
40.1666666667

In [168]:
```

1) Logic to take the top twenty features:
    - I have taken the standard deviation of the FFT data for the whole matrix for each class and subtracted the value with the matrix to find the least matching values of it.
    - I have sorted the least matching values and taken the indices of the first 20 features. And so when I do it for six classes I got 120 features which is then passed onto my Logistic Regression and Gradient descent to classify test and train data and find the accuracy of it.
2) Passed the 600*120 data to the cross validation of 10 Folds and then passed it to LR
3) Got accuracy of 40.1 %
4) I used the standard deviation extraction which is one of the worst techniques to take and classify the data. Because of that, I have got the lower accuracy level of classification.

<u>MFCC OUTPUT:</u>

ACCURACY PERCENTAGE: 63.66%



```python
    probability_of_model=np.zeros((6,540))
    delta_matrix=np.zeros((6,540))
    weights=np.zeros((6,14))
    shuffle_train=np.random.permutation(training_input[value])
    shuffle_test=np.random.permutation(testing_input[value])
    training_data=final_input[shuffle_train]
# max_value=np.amax(training_data,axis=0)
    max_value=np.max(training_data,axis=0) #p
    training_data=np.true_divide(training_data,max_value)
    input_matrix=np.ones((540,14))
    input_matrix[:,1:]=training_data
    training_data_t=input_matrix.transpose()
    values_delta=class_matrix_final[shuffle_train]
    k=len(values_delta)
    for i in range(0,k):
        if (values_delta[i] == 0):
            delta_matrix[0,i]=1
        else:
            if (values_delta[i]==1):
                delta_matrix[1,i]=1
            else:
                if (values_delta[i]==2):
                    delta_matrix[2,i]=1
                else:
                    if (values_delta[i]==3):
                        delta_matrix[3,i]=1
                    else:
                        if (values_delta[i]==4):
                            delta_matrix[4,i]=1
                        else:
                            if (values_delta[i]==5):
                                delta_matrix[5,i]=1
    for j in range(0,2000):#2000 Epochs
        neta=neta/(1+(j/2000)) #p
        output_matrix=weights.dot(training_data_t)
        output_matrix=np.exp(output_matrix)
        output_matrix[5:,:]=1
        sum_value=np.sum(output_matrix,axis=0)
```

```
In [38]: runfile('C:/Users/MaGesh/Desktop/LR_Project3/mfcc_program.py',
wdir='C:/Users/MaGesh/Desktop/LR_Project3')
sklearn.cross_validation.KFold(n=600, n_folds=10, shuffle=True, random_state=None)
[1 4 4 5 5 0 4 3 2 3 2 5 5 4 2 0 5 5 5 2 2 2 2 2 4 1 1 0 5 2 3 5 0 3 2 2 5
 2 3 5 0 4 0 0 5 2 2 0 5 0 2 1 2 1 3 3 2 5 0 5]
[4 1 5 1 3 2 5 0 1 0 5 1 0 0 5 2 5 3 3 2 5 5 4 5 3 0 0 3 0 2 0 5 5 0 3 5 0
 3 3 5 0 5 0 4 3 4 3 0 4 0 0 2 3 5 4 4 4 0 2 3]
[5 3 2 2 0 5 5 3 5 1 5 5 2 4 0 1 3 5 5 3 1 3 0 2 0 0 3 0 3 4 5 0 0 3 3 5 5
 4 5 1 5 5 3 1 5 5 0 4 3 1 2 5 1 5 1 1 3 0 2 0]
[5 2 1 5 2 0 5 4 0 4 5 3 0 0 5 4 5 3 5 2 5 1 5 3 1 5 2 3 4 2 2 3 2 5 2 0 5
 4 2 1 5 3 5 5 3 3 4 2 0 5 4 3 2 5 1 4 5 1 5 0]
[0 5 5 3 0 5 2 0 0 0 3 3 3 0 0 1 1 1 3 1 0 0 4 5 2 5 5 0 1 5 0 5 5 0 0 4 4
 3 5 0 3 2 3 0 1 5 4 1 2 5 5 5 0 1 5 4 0 3 3 5]
[2 2 1 5 5 5 4 0 5 2 4 2 1 1 2 0 3 0 1 3 1 4 3 1 0 1 2 2 2 4 0 0 2 4 2 1 5
 0 3 1 1 1 3 4 0 1 3 0 1 1 2 2 0 0 5 1 3 1 1 5]
[0 4 2 2 0 5 5 3 4 5 0 5 3 2 5 5 0 3 3 3 3 3 0 0 5 0 0 0 3 0 2 0 5 0 3 0 3
 4 2 4 5 2 1 4 1 5 0 2 4 2 0 5 0 1 5 0 3 5 2 0]
[1 1 1 3 0 2 1 3 5 1 2 4 2 2 3 1 2 5 4 2 2 0 3 1 4 2 0 4 0 5 3 1 4 5 4 2 1
 4 1 3 3 3 0 3 5 2 0 2 0 4 2 5 3 0 4 4 3 2 5 2]
[0 5 0 5 0 5 0 2 3 0 4 5 3 3 0 0 0 0 5 0 3 2 0 2 0 0 0 2 1 5 5 5 0 5 1 1 5
 0 5 2 0 2 2 2 1 2 1 0 4 3 1 1 0 3 5 5 3 5 0 5]
[4 4 2 5 3 1 0 1 4 3 5 2 3 5 2 0 5 5 0 4 4 3 1 4 3 4 5 5 0 5 1 5 0 3 0 4 4
 4 4 4 3 0 4 0 4 0 3 5 5 5 0 5 5 5 4 0 4 5 5 3]
[[ 85.   5.   4.   0.   4.   2.]
 [ 22.  42.  17.   5.   8.   6.]
 [ 14.  15.  46.   7.  12.   6.]
 [  2.   2.  11.  78.   4.   3.]
 [  5.   8.  11.   5.  39.  32.]
 [  1.   1.   2.   0.   4.  92.]]
63.6666666667

In [39]:
```

1) Used MFCC libraries to read the features of the data.
2) Again like the same for the previous two cases, I have used the K Fold cross validation method with 10 folds.
3) Accuracy level is 63.6% while with MFCC implementation.
4) MFCC is the one of the best functions to be used for audio compression. This Mel Spectrum is equally spaced on the Mel Scale hence this will be more approximate than the normal spectrum. Therefore the classification of the data using Mel Spectrum has given the higher accuracy level than the FFT. Though we have the Mel frequency, there should be high noise level in the data and that’s why I am getting the accuracy in the range of 60-70%.

<u>How to improve the classification task?</u>

1) Increasing in taking number of features may increase our accuracy level but still we need to be quiet sure in choosing the number of features. Taking more number of features may tend to overfitting problem.
2) Using the techniques such as FFT or MFCC is not the efficient way to read the audio files and classify it. Hence, we should think about some other ways to read the data efficiently and the data should also be noiseless, if the data is noisy then there should be high probability that the classifier wouldn’t classify it correctly.