MCA Homework-2

Jay Rawal
2017240

Part1

Spectrogram results were first flattened out and an SVM model was trained for the same. The following were the results.

```
Loading ./model/model_spc.sav
Length  10000
Getting report!
[0 1 2 3 4 5 6 7 8 9]
              precision    recall  f1-score   support

           0       0.76      0.62      0.68      1000
           1       0.52      0.60      0.55      1000
           2       0.51      0.53      0.52      1000
           3       0.66      0.49      0.56      1000
           4       0.64      0.64      0.64      1000
           5       0.59      0.60      0.60      1000
           6       0.64      0.80      0.71      1000
           7       0.63      0.58      0.60      1000
           8       0.61      0.64      0.63      1000
           9       0.54      0.57      0.55      1000

    accuracy                           0.60     10000
   macro avg       0.61      0.60      0.60     10000
weighted avg       0.61      0.60      0.60     10000

[[620  36  83  19  72  27  27  54  23  39]
 [ 14 596  35   8  86  71  14  25  34 117]
 [ 50  76 525  52  53  23  71  47  75  28]
 [ 22  44 110 487  18  23  44  39 138  75]
 [ 39 122  41   5 635  59  27  42  15  15]
 [  7  82  20  15  45 596  48  46  25 116]
 [  6  17  31   8  11  27 797  44  51   8]
 [ 44  36  67  28  43  48  67 584  18  65]
 [  9  35  64  76   8  13 120  13 639  23]
 [  9 109  56  38  18 115  26  40  23 566]]

C:\Users\Jay\Documents\College\sem6\mca\HW-2\assignment2>
```

We can observe that mean accuracy is 60%. The most correctly predicted class was “6” and the least correctly predicted was “3”.

Next, the same model was tested on validation set. Following were the results.

```
Loading ./model/model_spc.sav
Length  2494
Getting report!
[0 1 2 3 4 5 6 7 8 9]
              precision    recall  f1-score   support

           0       0.80      0.62      0.70       260
           1       0.45      0.61      0.52       230
           2       0.42      0.49      0.45       236
           3       0.58      0.52      0.55       248
           4       0.64      0.62      0.63       280
           5       0.61      0.55      0.58       242
           6       0.69      0.76      0.72       262
           7       0.67      0.57      0.62       263
           8       0.60      0.60      0.60       243
           9       0.47      0.49      0.48       230

    accuracy                           0.59      2494
   macro avg       0.59      0.58      0.59      2494
weighted avg       0.60      0.59      0.59      2494

[[162   7  33   4  21   6   6   9   1  11]
 [  4 140   7   5  21   7   9   6   5  26]
 [ 10  27 115  17  15   5   9   7  19  12]
 [  2   6  33 130   4   5   8   8  39  13]
 [  8  42  18   2 175  16   6  10   0   3]
 [  2  29   6   4  12 133   9   9   2  36]
 [  2   7  11   0   3   3 198  15  19   4]
 [  7  16  21  10  14  10  16 151   5  13]
 [  1   5  16  37   4   3  19   4 145   9]
 [  4  30  14  16   6  30   5   6   6 113]]
```

We get a mean accuracy of around 59% which was expected given that test had 60% for the same. Although, here the number of samples were not the same we can still see that “6” had the highest f1 score. Lowest this time was attained by class “2”.

We observed that our model works best with class “6”, without any noise, on both training set and validation set.

Next, we added noise to the files to observe the results. We trained a new model on this noisy data and reported the result.

Validation with noises on NOISY MODEL SPC gave the following results:

```
Loading ./model/model_spc_noisy.sav
Length  2494
Getting report!
[0 1 2 3 4 5 6 7 8 9]
              precision    recall  f1-score   support

           0       0.77      0.43      0.55       260
           1       0.36      0.61      0.45       230
           2       0.25      0.35      0.29       236
           3       0.46      0.38      0.42       248
           4       0.64      0.45      0.53       280
           5       0.58      0.39      0.47       242
           6       0.45      0.69      0.55       262
           7       0.42      0.50      0.46       263
           8       0.55      0.44      0.49       243
           9       0.50      0.34      0.40       230

    accuracy                           0.46      2494
   macro avg       0.50      0.46      0.46      2494
weighted avg       0.50      0.46      0.46      2494

[[112  18  44   8  14   8  21  27   1   7]
 [  3 140  14  10  11  10   9  19   3  11]
 [ 10  24  83  34   8   3  31  22  15   6]
 [  5   5  48  94   2   2  33  21  29   9]
 [  3  68  23   4 126  14  15  24   2   1]
 [  2  47  10   6  16  95  14  23   4  25]
 [  1   5  29   1   0   3 182  15  24   2]
 [  7  26  22   8  10   8  38 131   3  10]
 [  0   1  44  29   1   2  48   3 108   7]
 [  3  56  10  11   8  19  11  25   9  78]]

C:\Users\Jay\Documents\College\sem6\mca\HW-2\assignment2>
```

We see that our accuracy dropped from 59% to 46% when we added the noise. Although the accuracy dropped, we can claim that the model is more robust now due to the introduction of noise.

From the confusion matrix, we can observe that the most predicted class was “6” overall.

Part2

Next, mfcc was implemented an SVM model was trained on mfcc features.

Train without noise, results:

```
Loading ./model/model_mfcc_.sav
Length  10000
Getting report!
[0 1 2 3 4 5 6 7 8 9]
              precision    recall  f1-score   support

           0       0.78      0.63      0.70      1000
           1       0.51      0.62      0.56      1000
           2       0.46      0.58      0.51      1000
           3       0.63      0.54      0.58      1000
           4       0.62      0.63      0.63      1000
           5       0.65      0.56      0.60      1000
           6       0.70      0.77      0.73      1000
           7       0.71      0.59      0.64      1000
           8       0.61      0.65      0.63      1000
           9       0.58      0.57      0.57      1000

    accuracy                           0.61     10000
   macro avg       0.62      0.61      0.62     10000
weighted avg       0.62      0.61      0.62     10000

[[629  31 129   9  72  15  21  47  13  34]
 [  9 619  49  13  97  50  21   7  43  92]
 [ 36  72 578  63  56  10  46  62  55  22]
 [ 11  36 124 543  13  20  32  28 128  65]
 [ 29 154  65   1 632  62  14  16  12  15]
 [ 16  96  42  14  55 563  39  37  30 108]
 [ 10  26  38  11  15  16 771  21  77  15]
 [ 32  37 106  36  58  41  47 588  14  41]
 [ 13  26  72 112   5   3  98   2 653  16]
 [ 21 115  54  61  17  91  18  17  40 566]]

C:\Users\Jay\Documents\College\sem6\mca\HW-2\assignment2>
```

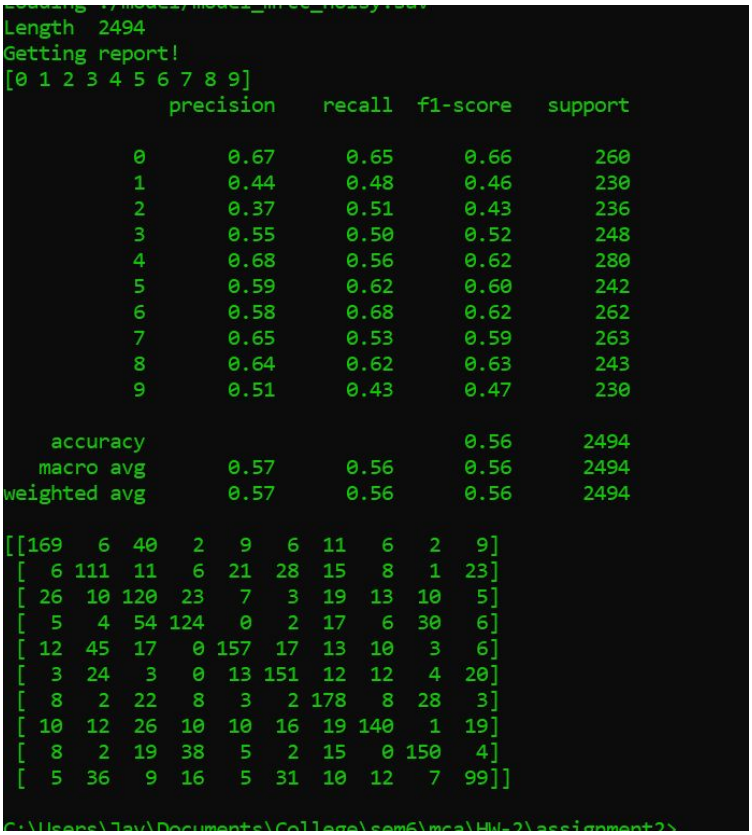
Similar to the spectrogram, we obtain 61% accuracy in this case as well with “6” being the best-predicted class. We can say that this is happening due to the dataset as a completely different model still is able to give the best result for the same.

Validation without noise:

```
oading ./model/model_mfcc_.sav
ength  2494
etting report!
0 1 2 3 4 5 6 7 8 9]
              precision    recall  f1-score   support

           0       0.82      0.67      0.74       260
           1       0.48      0.58      0.53       230
           2       0.37      0.53      0.44       236
           3       0.57      0.56      0.57       248
           4       0.65      0.65      0.65       280
           5       0.64      0.62      0.63       242
           6       0.72      0.78      0.75       262
           7       0.76      0.61      0.68       263
           8       0.66      0.63      0.64       243
           9       0.54      0.46      0.50       230

    accuracy                           0.61      2494
   macro avg       0.62      0.61      0.61      2494
eighted avg       0.63      0.61      0.62      2494

[174   4  38   2  16   4   3   8   1  10]
[  2 133  13   5  25  19  11   2   3  17]
[ 12  23 124  20  19   3   5  13  10   7]
[  1  10  34 139   1   1   8   5  35  14]
[  3  43  31   0 182  12   4   3   1   1]
[  3  22  11   3  10 149   9   9   3  23]
[  0   6  11   3   3   7 204   9  16   3]
[  5  10  34  10  13   7  11 160   1  12]
[  0   1  18  38   6   1  22   0 152   5]
[ 12  23  17  23   3  29   5   2  10 106]]
```

Results were on expected lines here as well, getting a 61% accuracy for validation set without noise.

We observe a less accuracy on the validation with noises on MFCC Noisy Model. Data with noise bring accuracy down to 56%. Here are the results:



```
Length  2494
Getting report!
[0 1 2 3 4 5 6 7 8 9]
              precision    recall  f1-score   support

           0       0.67      0.65      0.66       260
           1       0.44      0.48      0.46       230
           2       0.37      0.51      0.43       236
           3       0.55      0.50      0.52       248
           4       0.68      0.56      0.62       280
           5       0.59      0.62      0.60       242
           6       0.58      0.68      0.62       262
           7       0.65      0.53      0.59       263
           8       0.64      0.62      0.63       243
           9       0.51      0.43      0.47       230

    accuracy                           0.56      2494
   macro avg       0.57      0.56      0.56      2494
weighted avg       0.57      0.56      0.56      2494

[[169   6  40   2   9   6  11   6   2   9]
 [  6 111  11   6  21  28  15   8   1  23]
 [ 26  10 120  23   7   3  19  13  10   5]
 [  5   4  54 124   0   2  17   6  30   6]
 [ 12  45  17   0 157  17  13  10   3   6]
 [  3  24   3   0  13 151  12  12   4  20]
 [  8   2  22   8   3   2 178   8  28   3]
 [ 10  12  26  10  10  16  19 140   1  19]
 [  8   2  19  38   5   2  15   0 150   4]
 [  5  36   9  16   5  31  10  12   7  99]]
```

We observe that although, the results without noise were comparable between spectrogram and mfcc features (both around 60%) when encountered with noise both techniques behaved differently after training new models.
Where spectrogram accuracy got reduced to around 46%, mfcc predicted with better accuracy of 56%. Our mfcc noisy model can be considered a robust and adequate model as with noise too, accuracy didn't suffer much.

We also made an observation, that with the given dataset and trained models, class "6" was very prominent.

References:

- https://kevinsprojects.wordpress.com/2014/12/13/short-time-fourier-transform-using-python-and-numpy/
- https://www.kaggle.com/ilyamich/mfcc-implementation-and-tutorial
- https://towardsdatascience.com/fast-fourier-transform-937926e591cb

---