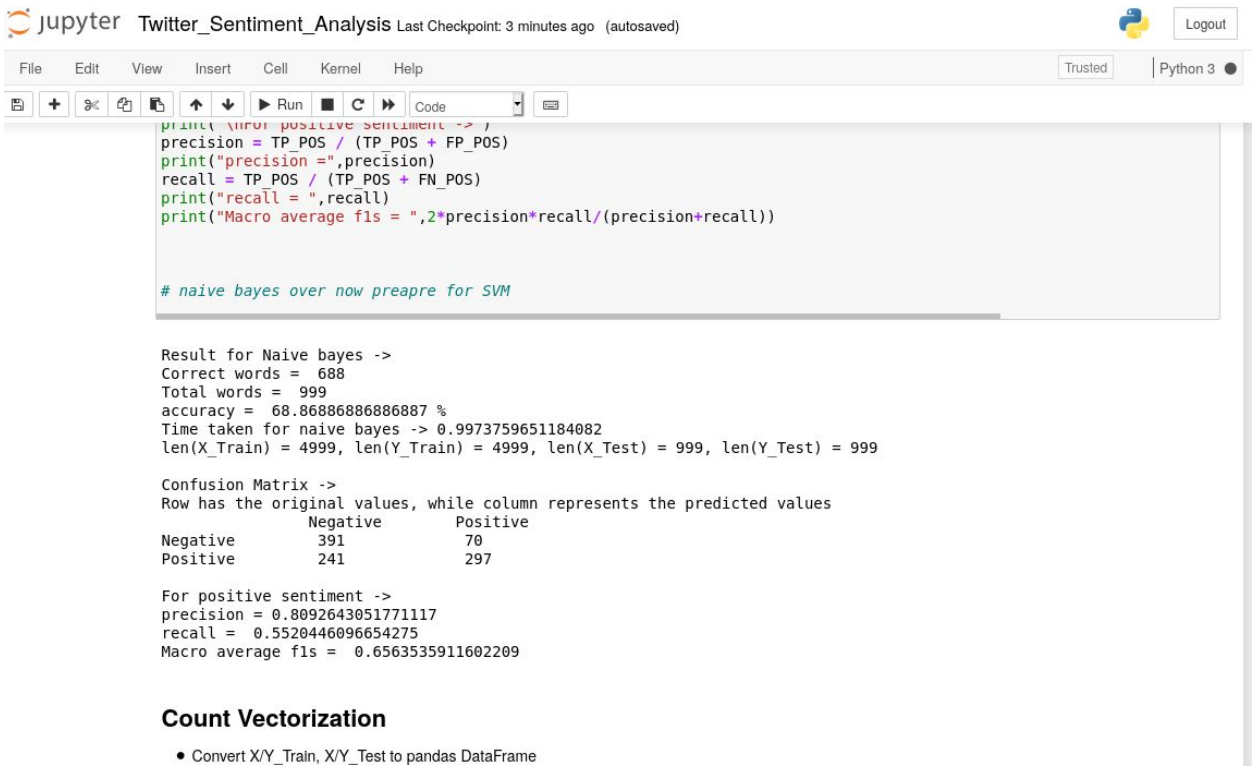
NLP : ASSIGNMENT - 3

OUTPUT

3a : Sentiment Analysis in Twitter

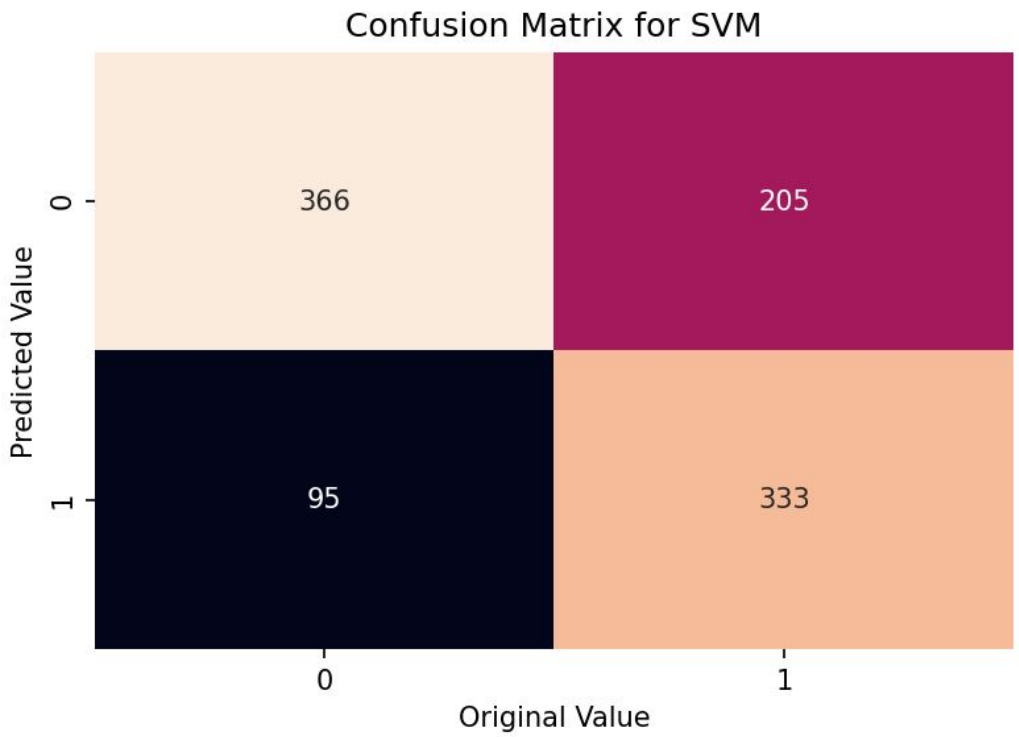
OUTPUT 1

The below screenshots are from the Twitter_Sentiment_Analysis.ipynb, in the conditions that the number of training & testing tweets were 5k & 1k respectively



This is the output for Naive Bayes - Bag Of Words approach

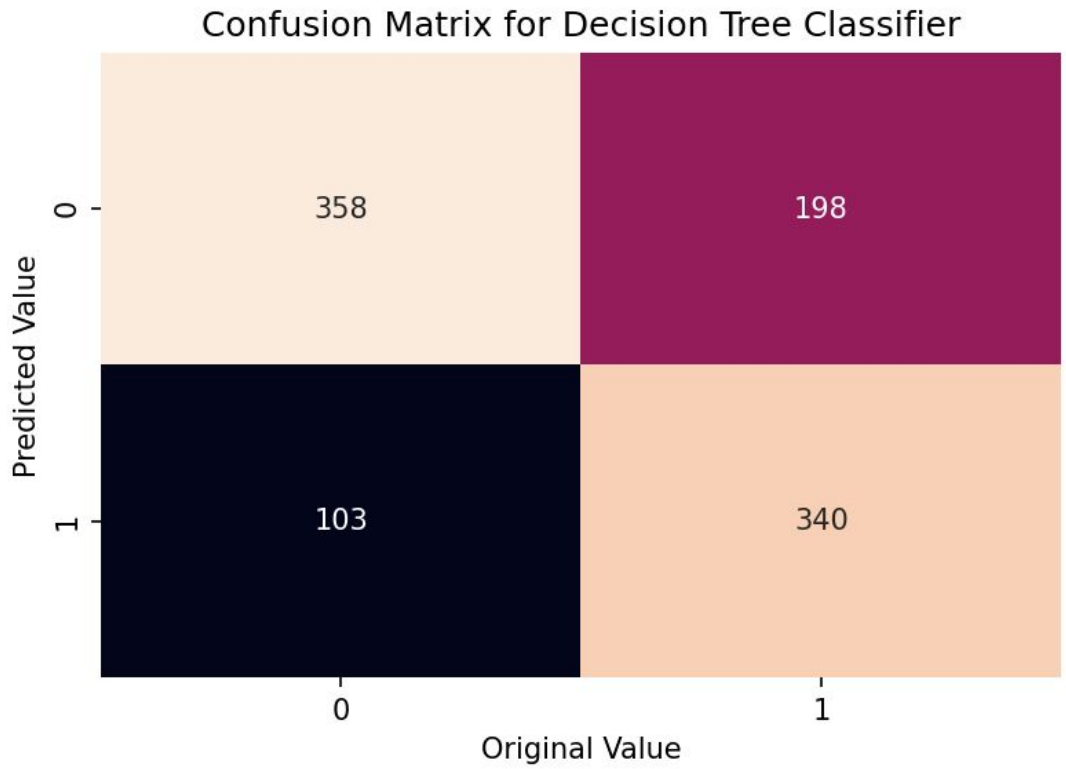
```
Now SVM model.....
No saved model... Creating one!
Time to train & save model -> 843.4613010883331
Accuracy by SVM -> 69.96996996996997 %
Time taken to predict -> 1167.4428622722626
```



	precision	recall	f1-score	support
0	0.64	0.79	0.71	461
4	0.78	0.62	0.69	538
accuracy			0.70	999
macro avg	0.71	0.71	0.70	999
weighted avg	0.71	0.70	0.70	999

This is the output for SVM

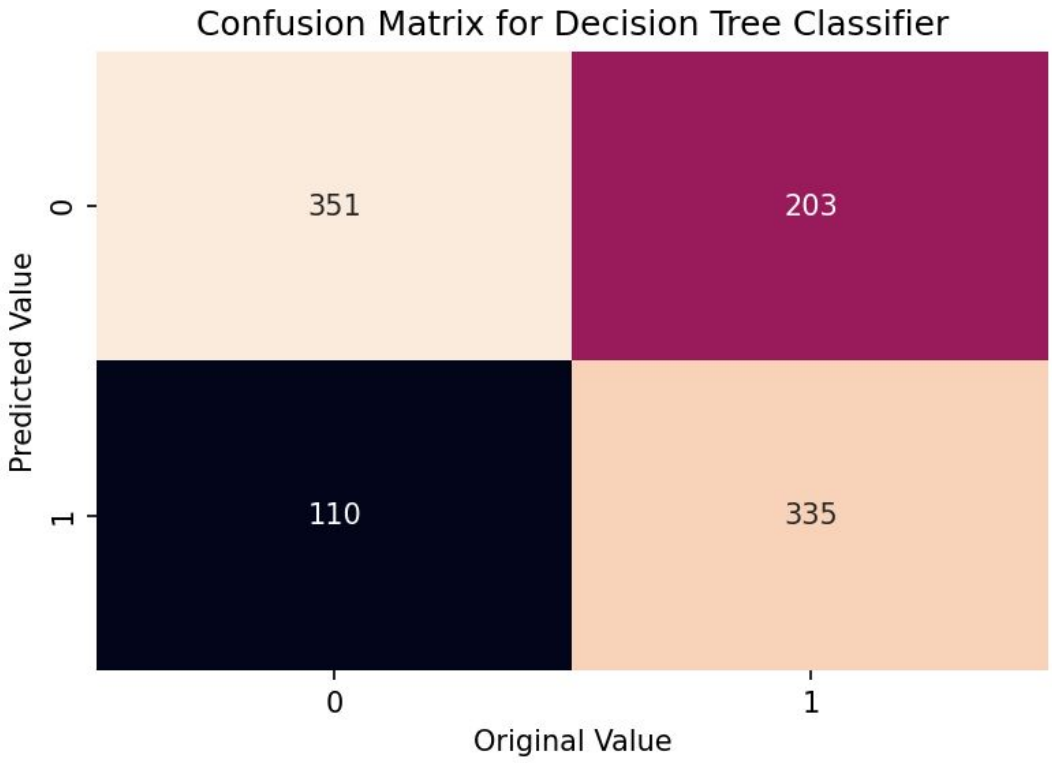
```
Now Decision Tree Classifier...
Accuracy by decision tree -> 69.86986986986987 %
Time taken for .fit & .predict -> 1172.3926916122437
```



	precision	recall	f1-score	support
0	0.64	0.78	0.70	461
4	0.77	0.63	0.69	538
accuracy			0.70	999
macro avg	0.71	0.70	0.70	999
weighted avg	0.71	0.70	0.70	999

This is the output for Decision Tree Classifier

```
MLP Classifier now.....
Accuracy by MLP Classifier -> 68.66866866866866 %
Time taken for .fit & .predict -> 1205.3631665706635
```



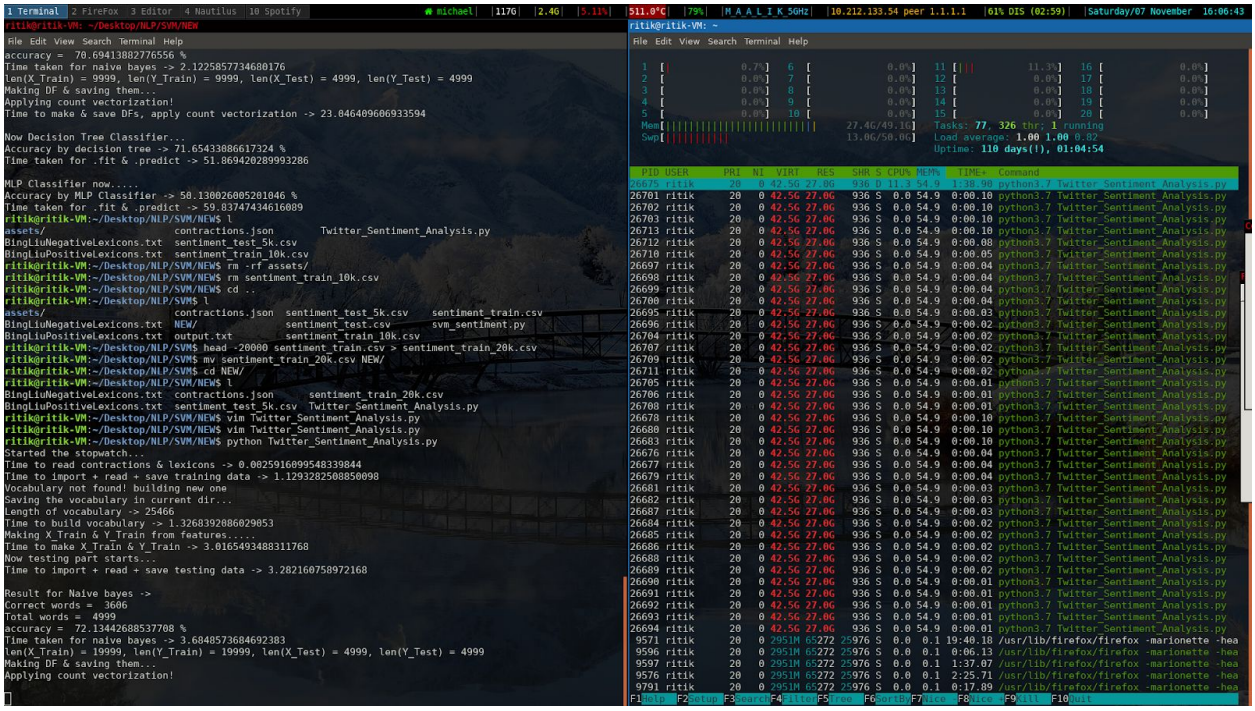
	precision	recall	f1-score	support
0	0.63	0.76	0.69	461
4	0.75	0.62	0.68	538
accuracy			0.69	999
macro avg	0.69	0.69	0.69	999
weighted avg	0.70	0.69	0.69	999

This is the output for MLP Classifier

As we can see, even for such a small amount of data (5k train & 1k test), the code took about 20 minutes, & 5GB RAM to run. The majority of time & RAM is consumed by SVM & Count Vectorization respectively.

Some other interesting stats -

Initially ran the code for 20k training set & 5k testing set -



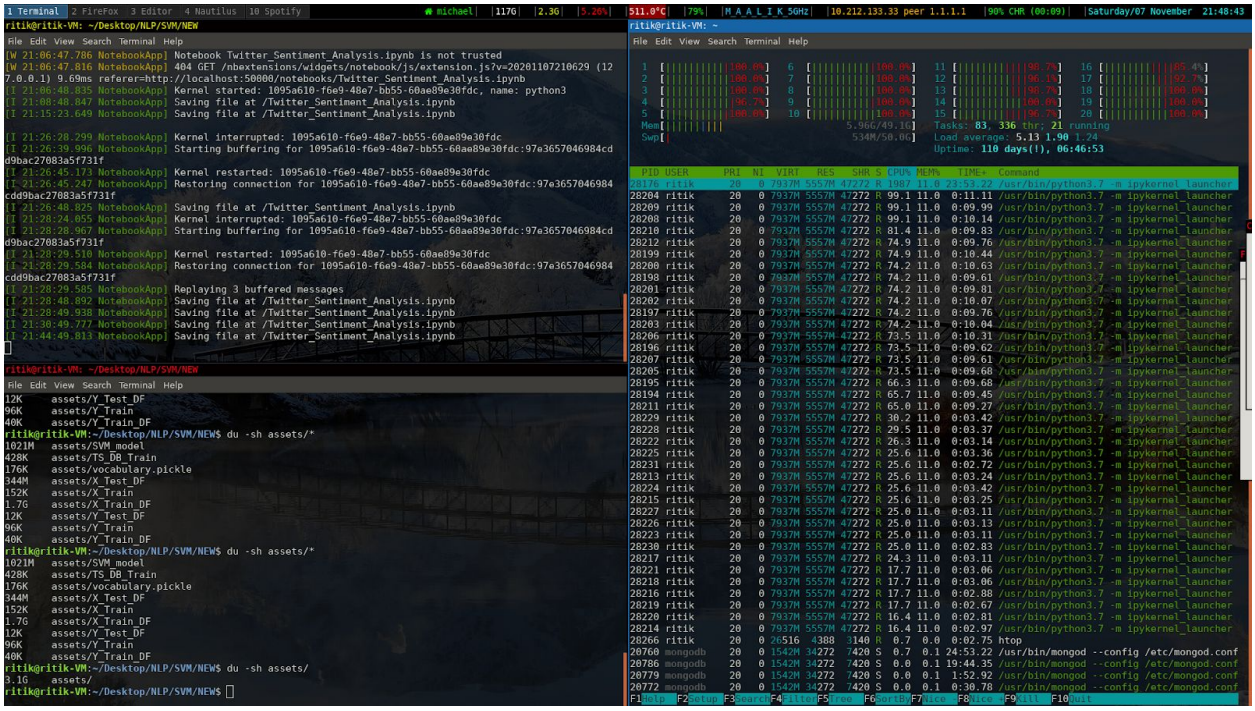
As you can see, it took more than 40GB RAM, & was running endlessly, I had to force terminate it

Then I ran it for 10k training & 5k testing ->

```
ritik@ritik-VM:~/Desktop/NLP/SVM$ cat output.txt
Begin...
Time to read contractions & lexicons -> 0.0026280879974365234
Time to import + read + save training data -> 0.6834373474121094
Vocabulary not found! building new one
Saving the vocabulary in current dir...
Length of vocabulary -> 16035
Time to build vocabulary -> 0.7851431369781494
Making X_Train & Y_Train from features...
Time to make X_Train & Y_Train -> 1.6033999919891357
Now testing part starts...
Time to import + read + save testing data -> 1.924849510192871

Using Naive bayes ->
Correct words =  3539
Total words =  4999
accuracy =  70.79415883176635 %
Time taken for naive bayes -> 2.315837860107422
4999 9999 4999 9999
Making DF & saving them...
Applying count vectorization!
Time to make & save DFs, apply count vectorization -> 25.233197689056396

Now SVM model.....
No saved model... Creating one!
Time to train & save model -> 5286.2617082595825
Accuracy by SVM -> 0.7203440688137628
Time taken to predict -> 7820.864698171616

Now Decision Tree Classifier...
Accuracy by decision tree -> 0.71874374874975
Time taken for .fit & .predict -> 7840.198331832886

MLP Classifier now.....
Accuracy by MLP Classifier -> 0.6983396679335867
Time taken for .fit & .predict -> 8034.783812999725
```

As you can see, it took about 2 hours & 30 minutes to run the entire code. Also, it consumed upto 24GB of RAM

Another thing to note MLP classifier runs on multicore, unlike Decision Trees or SVM, as we can see in the screenshot below



3b : Emotion Intensity Prediction

OUTPUT 2

The below screenshots are from Emotion_Intensity_JOY.ipynb

SVM

In [10]:
```
## SVM
print("\nResult for SVM ->")
model_prediction(SVR(), X_Train_DF, Y_Train, X_Test_DF, Y_Test)
print("Time taken by SVM model ->",time() - start_time)
```

```
Result for SVM ->
Results of sklearn.metrics:
MAE: 0.14406197351666597
MSE: 0.0323679201410181
RMSE: 0.17991086721212285
R-Squared: 0.3135262200437178

pearson corr. , p valve = (0.5630474325682381, 5.9210556367657214e-61)
SpearmanrResult(correlation=0.5713068409028125, pvalue=4.2709173140663443e-63)
Time taken by SVM model -> 12.120771408081055
```

This is the output for JOY by SVM

Decision Tree Classifier

In [11]:
```
## Using Decision Tree
print("\nResult Decision Tree ->")
model_prediction(DecisionTreeRegressor(max_depth = 5), X_Train_DF, Y_Train, X_Test_DF, Y_Test)
print("Time taken by Decision Tree ->",time() - start_time)
```

```
Result Decision Tree ->
Results of sklearn.metrics:
MAE: 0.1538318236993331
MSE: 0.037988457554869993
RMSE: 0.19490627890057824
R-Squared: 0.19432327011481154

pearson corr. , p valve = (0.4784480070703477, 4.00490638315593e-42)
SpearmanrResult(correlation=0.4766555956767596, pvalue=8.852764699823259e-42)
Time taken by Decision Tree -> 12.420202016830444
```

This is the output for JOY by Decision Tree Classifier

MLP Regressor

```
In [12]: ## using MLP
         print("\nResult for MLP ->")
         clf = MLPRegressor(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1,max_iter=1000)
         model_prediction(clf, X_Train_DF, Y_Train, X_Test_DF, Y_Test)
         print("Time taken by MLP ->",time() - start_time)
```

```
Result for MLP ->
Results of sklearn.metrics:
MAE: 0.18284210257614578
MSE: 0.047417972355903115
RMSE: 0.21775668154135505
R-Squared: -0.005662229120769302

pearson corr. , p valve = (nan, nan)
SpearmanrResult(correlation=nan, pvalue=nan)
Time taken by MLP -> 13.100698947906494
```

This is the output for JOY by MLP Classifier

As you can see, the code takes no more than 13 seconds to run, so no need to save the models

OUTPUT 3

The below screenshots are from Emotion_Intensity_ANGER.ipynb

SVM

```
In [10]: ## SVM
         print("\nResult for SVM ->")
         model_prediction(SVR(), X_Train_DF, Y_Train, X_Test_DF, Y_Test)
         print("Time taken by SVM model ->",time() - start_time)
```

```
Result for SVM ->
Results of sklearn.metrics:
MAE: 0.11319175362244595
MSE: 0.019879849385711378
RMSE: 0.14099591974845008
R-Squared: 0.3262405365895137

pearson corr. , p valve = (0.5790818107120931, 2.9132508996171585e-69)
SpearmanrResult(correlation=0.5689982163886088, pvalue=2.06129836924723e-66)
Time taken by SVM model -> 11.63190221786499
```

This is the output for ANGER by SVM

Decision Tree Classifier

In [11]:
```
## Using Decision Tree
print("\nResult Decision Tree ->")
model_prediction(DecisionTreeRegressor(max_depth = 5), X_Train_DF, Y_Train, X_Test_DF, Y_Test)
print("Time taken by Decision Tree ->",time() - start_time)
```

```
Result Decision Tree ->
Results of sklearn.metrics:
MAE: 0.1154059323323111
MSE: 0.020980108851885355
RMSE: 0.14484512022117058
R-Squared: 0.2889510072194118

pearson corr. , p valve = (0.547594469013182, 1.1151218338857375e-60)
SpearmanrResult(correlation=0.5507191215289514, pvalue=1.7222153245789713e-61)
Time taken by Decision Tree -> 11.959733486175537
```

This is the output for ANGER by Decision Tree Classifier

MLP Regressor

In [12]:
```
## using MLP
print("\nResult for MLP ->")
clf = MLPRegressor(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1,max_iter=1000)
model_prediction(clf, X_Train_DF, Y_Train, X_Test_DF, Y_Test)
print("Time taken by MLP ->",time() - start_time)
```

```
Result for MLP ->
Results of sklearn.metrics:
MAE: 0.11389595058964681
MSE: 0.020303883056515853
RMSE: 0.14249169469311485
R-Squared: 0.311869366417845

pearson corr. , p valve = (0.5831974578708614, 1.872611882396523e-70)
SpearmanrResult(correlation=0.5732742289854856, pvalue=1.3114177982316639e-67)
Time taken by MLP -> 18.60454535484314
```

In [ ]:

This is the output for ANGER by MLP Classifier