Shahjalal University of science & technology,sylhet

Assignment on Text Classification

ML

Submitted To :

Ms Sayma Sultana Chowdhury
Assistant Professor
Dept. of Software Engineering(IICT),SUST

Submitted By :

Dipankar Bala
Reg.No. : 2017831047
Dept.of Software Engineering(IICT),SUST

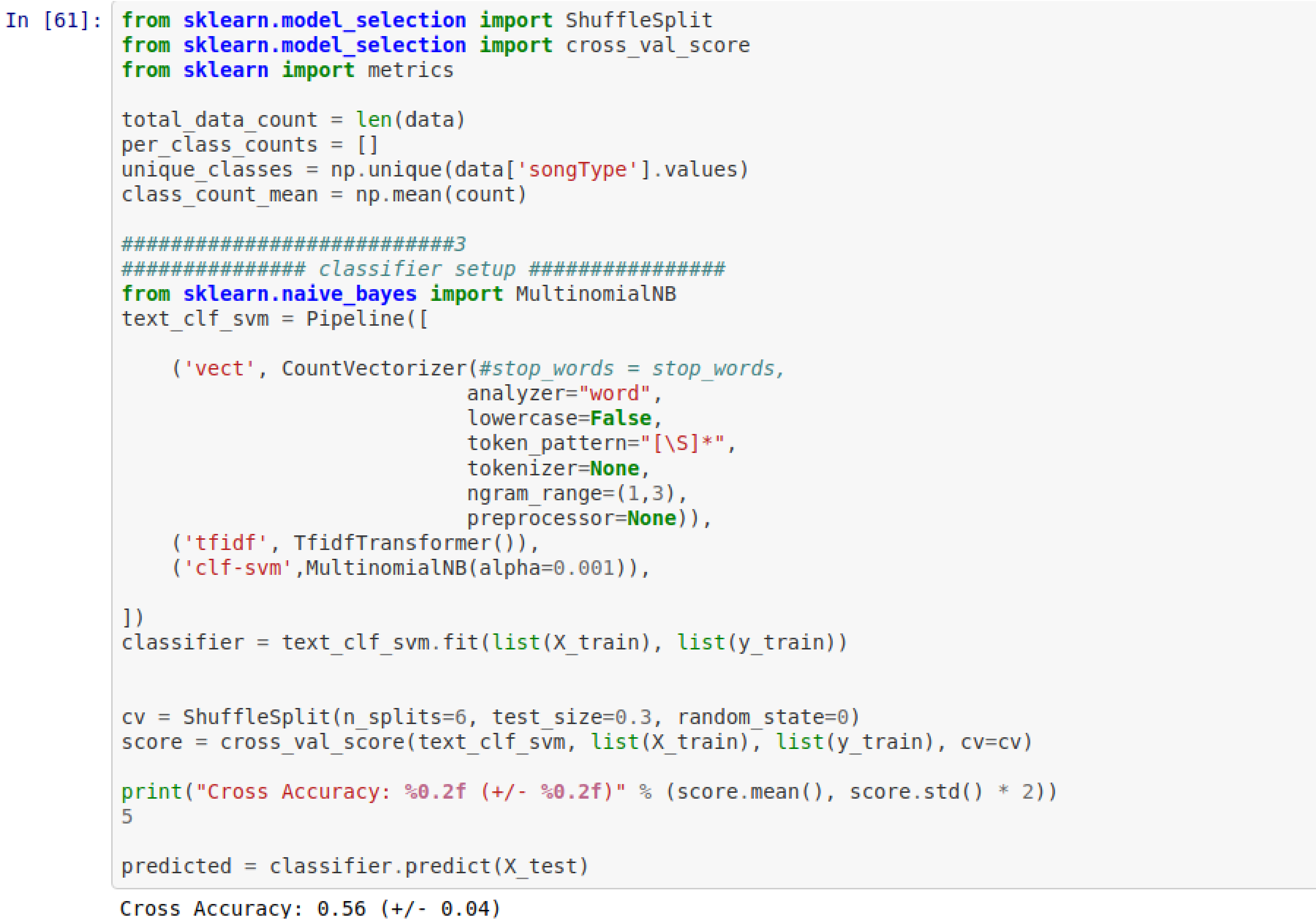
11 february, 2021

1.

Classification Report

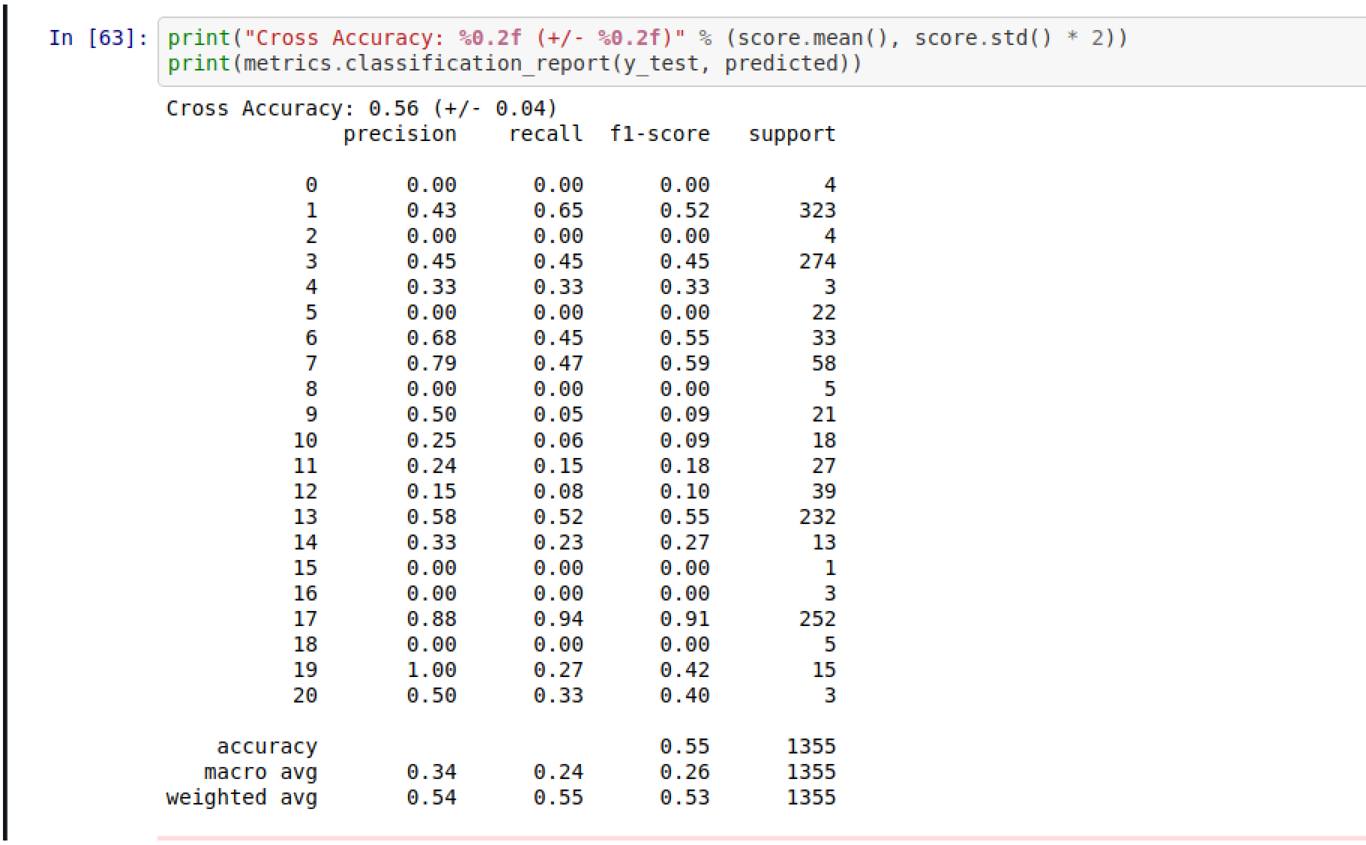
1 Naive Bayes Classification:

The cross accuracy of Multinomial Native Bayes Model is shown in the below picture.



```
In [61]: from sklearn.model_selection import ShuffleSplit
         from sklearn.model_selection import cross_val_score
         from sklearn import metrics

         total_data_count = len(data)
         per_class_counts = []
         unique_classes = np.unique(data['songType'].values)
         class_count_mean = np.mean(count)

         ############################3
         ############### classifier setup ################
         from sklearn.naive_bayes import MultinomialNB
         text_clf_svm = Pipeline([

             ('vect', CountVectorizer(#stop_words = stop_words,
                                     analyzer="word",
                                     lowercase=False,
                                     token_pattern="[\S]*",
                                     tokenizer=None,
                                     ngram_range=(1,3),
                                     preprocessor=None)),
             ('tfidf', TfidfTransformer()),
             ('clf-svm',MultinomialNB(alpha=0.001)),

         ])
         classifier = text_clf_svm.fit(list(X_train), list(y_train))


         cv = ShuffleSplit(n_splits=6, test_size=0.3, random_state=0)
         score = cross_val_score(text_clf_svm, list(X_train), list(y_train), cv=cv)

         print("Cross Accuracy: %0.2f (+/- %0.2f)" % (score.mean(), score.std() * 2))
         5

         predicted = classifier.predict(X_test)

         Cross Accuracy: 0.56 (+/- 0.04)
```

Here is the accuracy,macro average & weighted average are given.

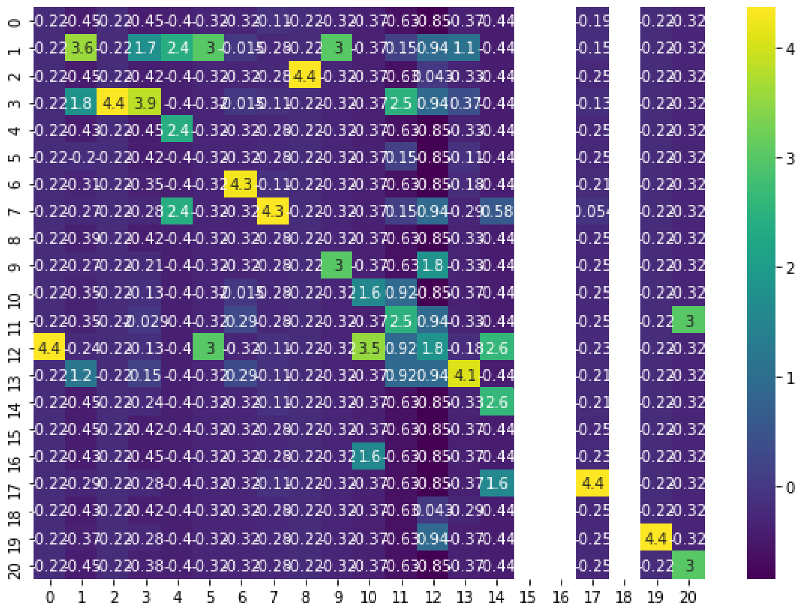


```
In [63]: print("Cross Accuracy: %0.2f (+/- %0.2f)" % (score.mean(), score.std() * 2))
         print(metrics.classification_report(y_test, predicted))
```

```
Cross Accuracy: 0.56 (+/- 0.04)
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       0.43      0.65      0.52       323
           2       0.00      0.00      0.00         4
           3       0.45      0.45      0.45       274
           4       0.33      0.33      0.33         3
           5       0.00      0.00      0.00        22
           6       0.68      0.45      0.55        33
           7       0.79      0.47      0.59        58
           8       0.00      0.00      0.00         5
           9       0.50      0.05      0.09        21
          10       0.25      0.06      0.09        18
          11       0.24      0.15      0.18        27
          12       0.15      0.08      0.10        39
          13       0.58      0.52      0.55       232
          14       0.33      0.23      0.27        13
          15       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         3
          17       0.88      0.94      0.91       252
          18       0.00      0.00      0.00         5
          19       1.00      0.27      0.42        15
          20       0.50      0.33      0.40         3

    accuracy                           0.55      1355
   macro avg       0.34      0.24      0.26      1355
weighted avg       0.54      0.55      0.53      1355
```

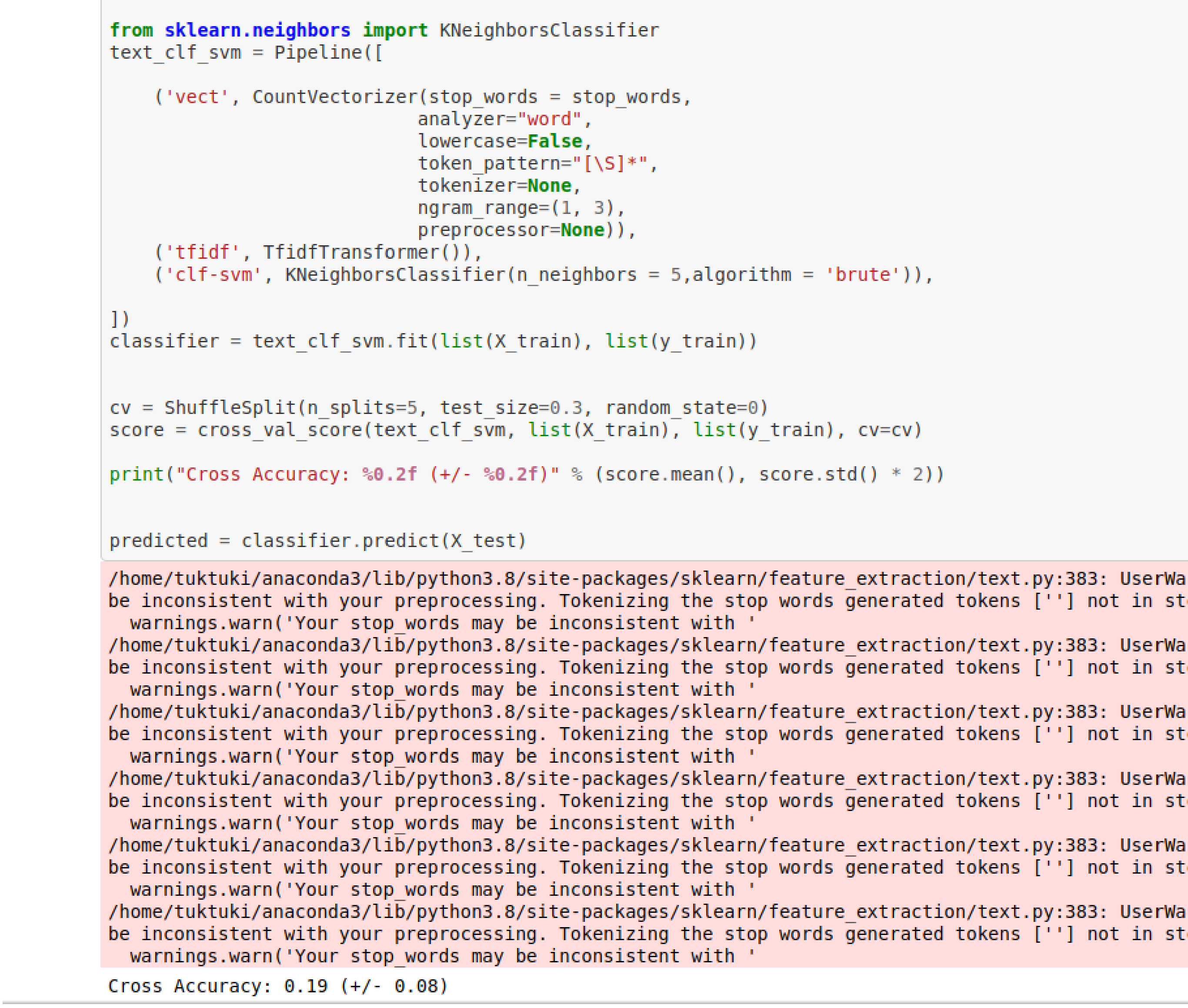
The confution matrix of Multinomial Native Bayes Model.

```
Out[65]: <AxesSubplot:>
```



2 k-Nearest Neighbors :

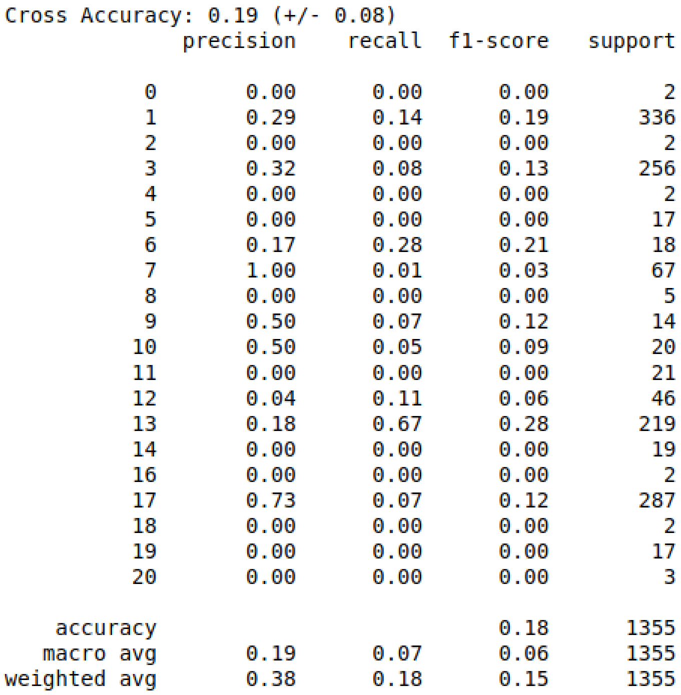
The cross accuracy of KNeighborsClassifier Model is shown in the below picture.



```
from sklearn.neighbors import KNeighborsClassifier
text_clf_svm = Pipeline([

    ('vect', CountVectorizer(stop_words = stop_words,
                            analyzer="word",
                            lowercase=False,
                            token_pattern="[\S]*",
                            tokenizer=None,
                            ngram_range=(1, 3),
                            preprocessor=None)),
    ('tfidf', TfidfTransformer()),
    ('clf-svm', KNeighborsClassifier(n_neighbors = 5,algorithm = 'brute')),

])
classifier = text_clf_svm.fit(list(X_train), list(y_train))


cv = ShuffleSplit(n_splits=5, test_size=0.3, random_state=0)
score = cross_val_score(text_clf_svm, list(X_train), list(y_train), cv=cv)

print("Cross Accuracy: %0.2f (+/- %0.2f)" % (score.mean(), score.std() * 2))


predicted = classifier.predict(X_test)
```

```
/home/tuktuki/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:383: UserWa
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens [''] not in st
  warnings.warn('Your stop_words may be inconsistent with '
/home/tuktuki/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:383: UserWa
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens [''] not in st
  warnings.warn('Your stop_words may be inconsistent with '
/home/tuktuki/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:383: UserWa
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens [''] not in st
  warnings.warn('Your stop_words may be inconsistent with '
/home/tuktuki/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:383: UserWa
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens [''] not in st
  warnings.warn('Your stop_words may be inconsistent with '
/home/tuktuki/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:383: UserWa
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens [''] not in st
  warnings.warn('Your stop_words may be inconsistent with '
/home/tuktuki/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:383: UserWa
be inconsistent with your preprocessing. Tokenizing the stop words generated tokens [''] not in st
  warnings.warn('Your stop_words may be inconsistent with '
Cross Accuracy: 0.19 (+/- 0.08)
```

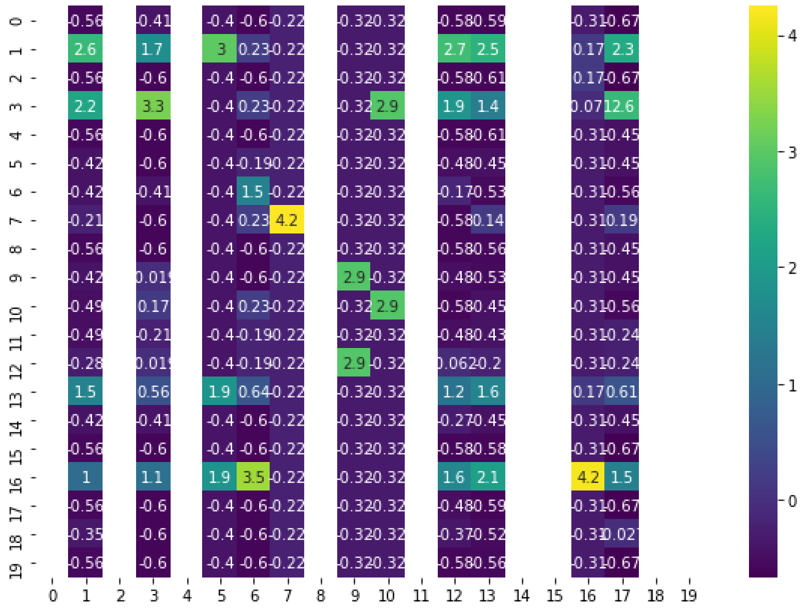
Here is the accuracy,macro average & weighted average are given.



```
Cross Accuracy: 0.19 (+/- 0.08)
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.29      0.14      0.19       336
           2       0.00      0.00      0.00         2
           3       0.32      0.08      0.13       256
           4       0.00      0.00      0.00         2
           5       0.00      0.00      0.00        17
           6       0.17      0.28      0.21        18
           7       1.00      0.01      0.03        67
           8       0.00      0.00      0.00         5
           9       0.50      0.07      0.12        14
          10       0.50      0.05      0.09        20
          11       0.00      0.00      0.00        21
          12       0.04      0.11      0.06        46
          13       0.18      0.67      0.28       219
          14       0.00      0.00      0.00        19
          16       0.00      0.00      0.00         2
          17       0.73      0.07      0.12       287
          18       0.00      0.00      0.00         2
          19       0.00      0.00      0.00        17
          20       0.00      0.00      0.00         3

    accuracy                           0.18      1355
   macro avg       0.19      0.07      0.06      1355
weighted avg       0.38      0.18      0.15      1355
```

The confution matrix of KNeighborsClassifier Model.

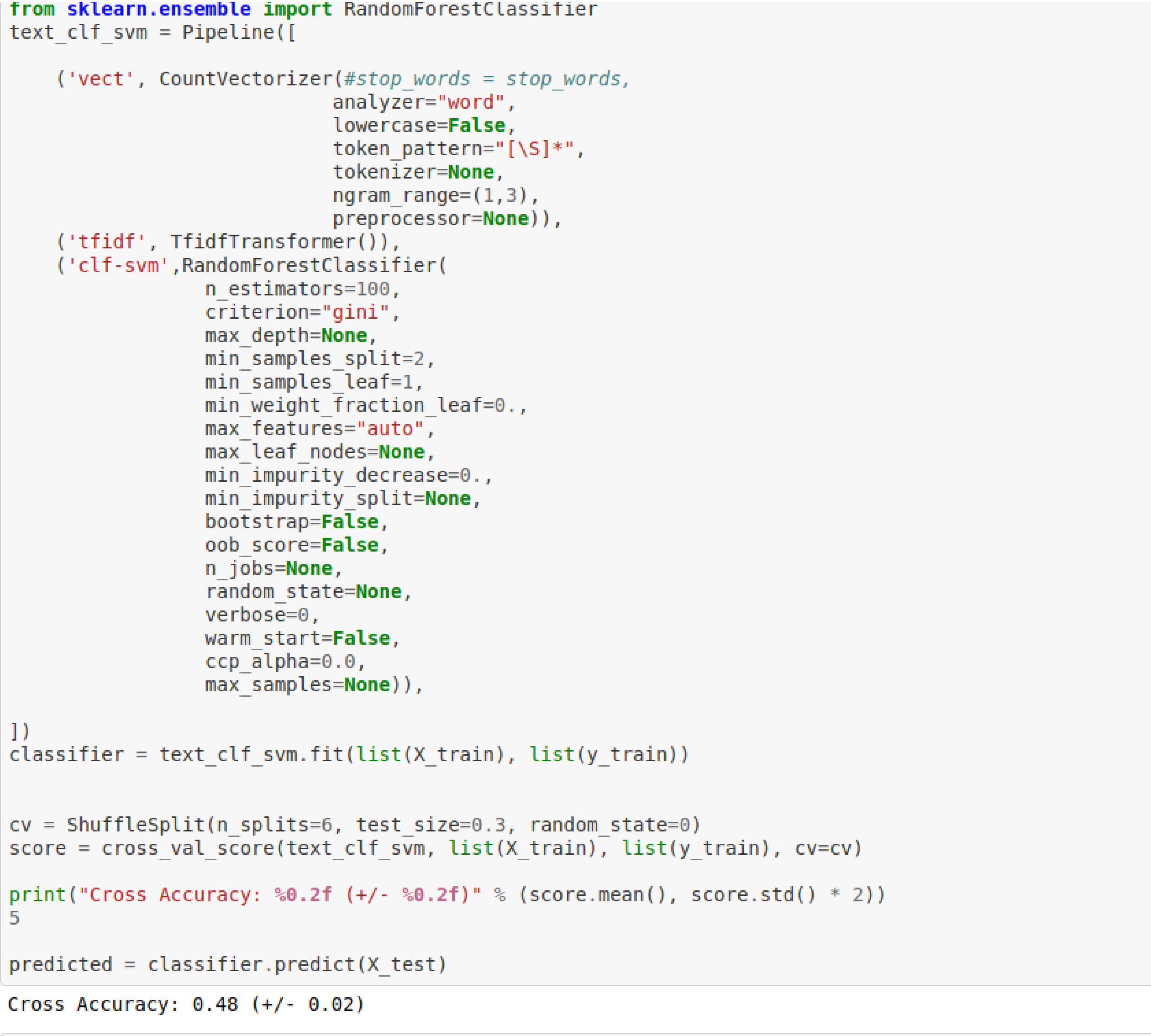
```
Out[124]: <AxesSubplot:>
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		-0.56		-0.41		-0.4	-0.6	-0.22		-0.32	-0.32		-0.58	-0.59			-0.31	-0.67		
1		2.6		1.7		3	0.23	-0.22		-0.32	-0.32		2.7	2.5			0.17	2.3		
2		-0.56		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.58	-0.61			0.17	-0.67		
3		2.2		3.3		-0.4	0.23	-0.22		-0.32	2.9		1.9	1.4			0.071	2.6		
4		-0.56		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.58	-0.61			-0.31	-0.45		
5		-0.42		-0.6		-0.4	-0.19	-0.22		-0.32	-0.32		-0.48	-0.45			-0.31	-0.45		
6		-0.42		-0.41		-0.4	1.5	-0.22		-0.32	-0.32		-0.17	-0.53			-0.31	-0.56		
7		-0.21		-0.6		-0.4	0.23	4.2		-0.32	-0.32		-0.58	0.14			-0.31	0.19		
8		-0.56		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.58	-0.56			-0.31	-0.45		
9		-0.42		0.019		-0.4	-0.6	-0.22		2.9	-0.32		-0.48	-0.53			-0.31	-0.45		
10		-0.49		0.17		-0.4	0.23	-0.22		-0.32	2.9		-0.58	-0.45			-0.31	-0.56		
11		-0.49		-0.21		-0.4	-0.19	-0.22		-0.32	-0.32		-0.48	-0.43			-0.31	-0.24		
12		-0.28		0.019		-0.4	-0.19	-0.22		2.9	-0.32		0.062	-0.2			-0.31	-0.24		
13		1.5		0.56		1.9	0.64	-0.22		-0.32	-0.32		1.2	1.6			0.17	0.61		
14		-0.42		-0.41		-0.4	-0.6	-0.22		-0.32	-0.32		-0.27	-0.45			-0.31	-0.45		
15		-0.56		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.58	-0.58			-0.31	-0.67		
16		1		1.1		1.9	3.5	-0.22		-0.32	-0.32		1.6	2.1			4.2	1.5		
17		-0.56		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.48	-0.59			-0.31	-0.67		
18		-0.35		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.37	-0.52			-0.31	0.027		
19		-0.56		-0.6		-0.4	-0.6	-0.22		-0.32	-0.32		-0.58	-0.56			-0.31	-0.67		

3 Random Forest Classifier :

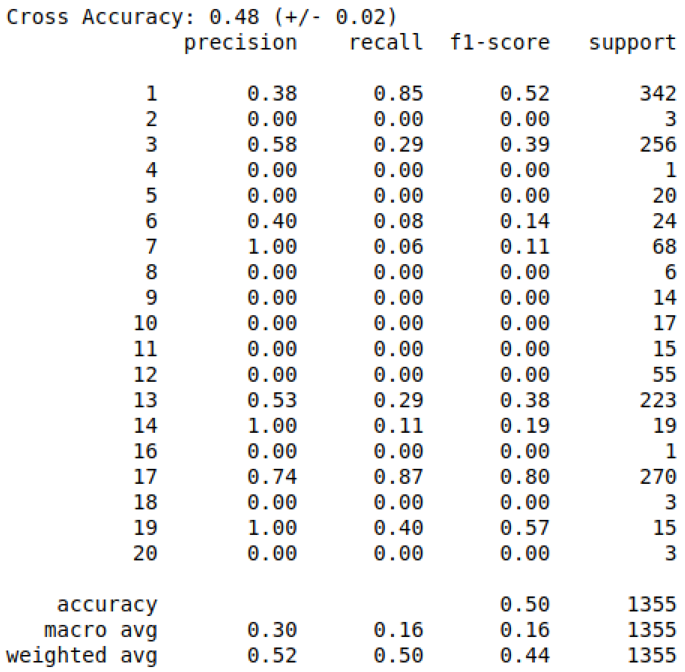
The cross accuracy of RandomForestClassifier Model is shown in the below picture.



```
from sklearn.ensemble import RandomForestClassifier
text_clf_svm = Pipeline([

    ('vect', CountVectorizer(#stop_words = stop_words,
                            analyzer="word",
                            lowercase=False,
                            token_pattern="[\S]*",
                            tokenizer=None,
                            ngram_range=(1,3),
                            preprocessor=None)),
    ('tfidf', TfidfTransformer()),
    ('clf-svm',RandomForestClassifier(
                 n_estimators=100,
                 criterion="gini",
                 max_depth=None,
                 min_samples_split=2,
                 min_samples_leaf=1,
                 min_weight_fraction_leaf=0.,
                 max_features="auto",
                 max_leaf_nodes=None,
                 min_impurity_decrease=0.,
                 min_impurity_split=None,
                 bootstrap=False,
                 oob_score=False,
                 n_jobs=None,
                 random_state=None,
                 verbose=0,
                 warm_start=False,
                 ccp_alpha=0.0,
                 max_samples=None)),

])
classifier = text_clf_svm.fit(list(X_train), list(y_train))


cv = ShuffleSplit(n_splits=6, test_size=0.3, random_state=0)
score = cross_val_score(text_clf_svm, list(X_train), list(y_train), cv=cv)

print("Cross Accuracy: %0.2f (+/- %0.2f)" % (score.mean(), score.std() * 2))
5

predicted = classifier.predict(X_test)
```

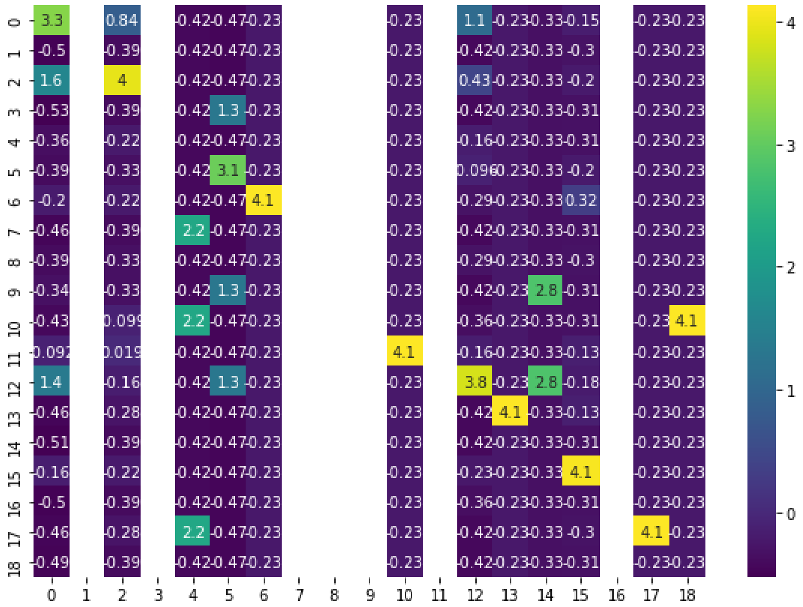
```
Cross Accuracy: 0.48 (+/- 0.02)
```

Here is the accuracy,macro average & weighted average are given.



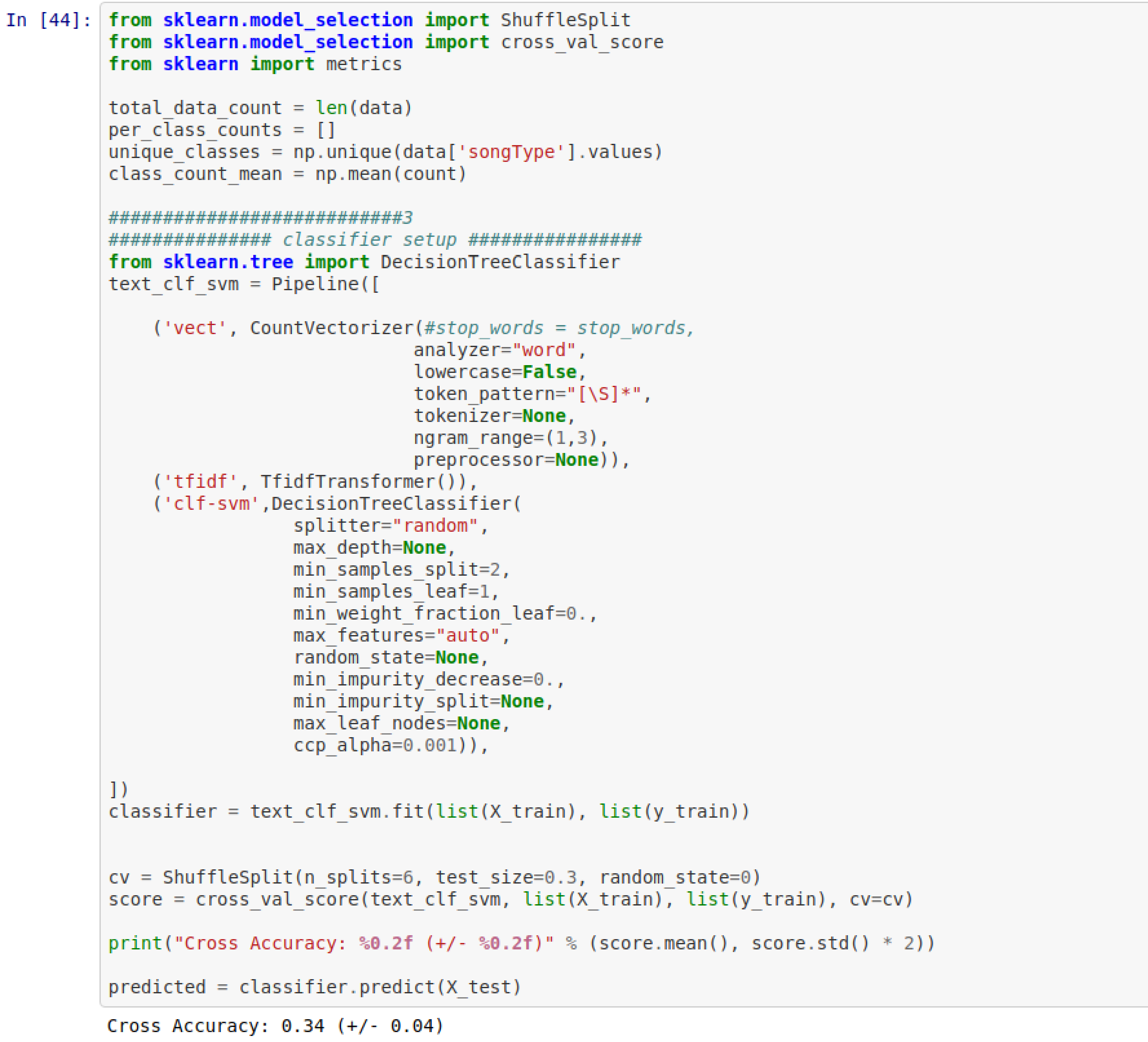
```
Cross Accuracy: 0.48 (+/- 0.02)
              precision    recall  f1-score   support

           1       0.38      0.85      0.52       342
           2       0.00      0.00      0.00         3
           3       0.58      0.29      0.39       256
           4       0.00      0.00      0.00         1
           5       0.00      0.00      0.00        20
           6       0.40      0.08      0.14        24
           7       1.00      0.06      0.11        68
           8       0.00      0.00      0.00         6
           9       0.00      0.00      0.00        14
          10       0.00      0.00      0.00        17
          11       0.00      0.00      0.00        15
          12       0.00      0.00      0.00        55
          13       0.53      0.29      0.38       223
          14       1.00      0.11      0.19        19
          16       0.00      0.00      0.00         1
          17       0.74      0.87      0.80       270
          18       0.00      0.00      0.00         3
          19       1.00      0.40      0.57        15
          20       0.00      0.00      0.00         3

    accuracy                           0.50      1355
   macro avg       0.30      0.16      0.16      1355
weighted avg       0.52      0.50      0.44      1355
```

The confution matrix of RandomForestClassifier Model.



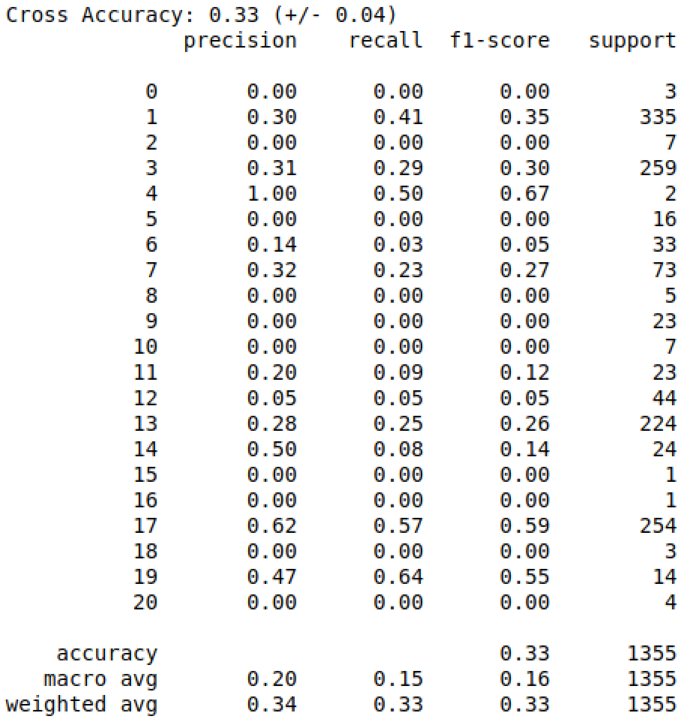
4 Decision Tree Classifier :

The cross accuracy of DecisionTreeClassifier Model is shown in the below picture.



```
In [44]: from sklearn.model_selection import ShuffleSplit
         from sklearn.model_selection import cross_val_score
         from sklearn import metrics

         total_data_count = len(data)
         per_class_counts = []
         unique_classes = np.unique(data['songType'].values)
         class_count_mean = np.mean(count)

         ############################3
         ############### classifier setup ################
         from sklearn.tree import DecisionTreeClassifier
         text_clf_svm = Pipeline([

             ('vect', CountVectorizer(#stop_words = stop_words,
                                     analyzer="word",
                                     lowercase=False,
                                     token_pattern="[\S]*",
                                     tokenizer=None,
                                     ngram_range=(1,3),
                                     preprocessor=None)),
             ('tfidf', TfidfTransformer()),
             ('clf-svm',DecisionTreeClassifier(
                                  splitter="random",
                                  max_depth=None,
                                  min_samples_split=2,
                                  min_samples_leaf=1,
                                  min_weight_fraction_leaf=0.,
                                  max_features="auto",
                                  random_state=None,
                                  min_impurity_decrease=0.,
                                  min_impurity_split=None,
                                  max_leaf_nodes=None,
                                  ccp_alpha=0.001)),

         ])
         classifier = text_clf_svm.fit(list(X_train), list(y_train))


         cv = ShuffleSplit(n_splits=6, test_size=0.3, random_state=0)
         score = cross_val_score(text_clf_svm, list(X_train), list(y_train), cv=cv)

         print("Cross Accuracy: %0.2f (+/- %0.2f)" % (score.mean(), score.std() * 2))

         predicted = classifier.predict(X_test)

         Cross Accuracy: 0.34 (+/- 0.04)
```

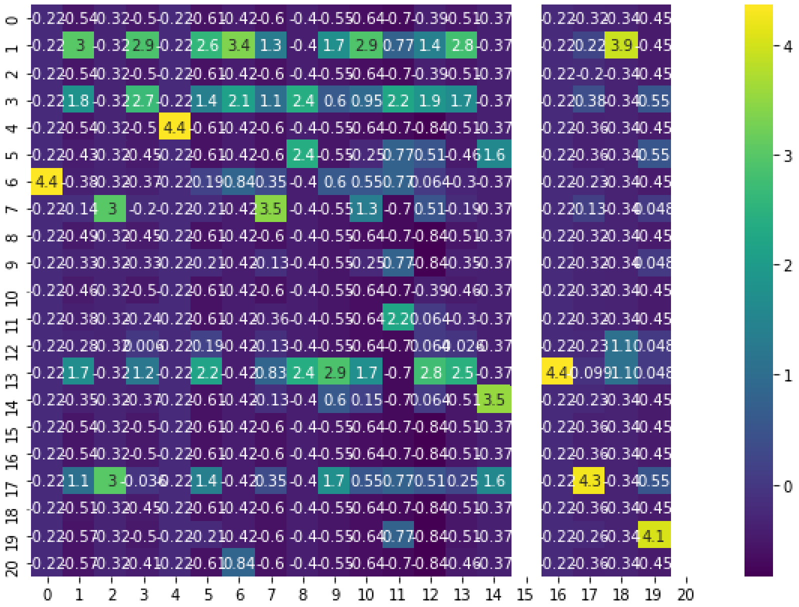
Here is the accuracy,macro average & weighted average are given.



```
Cross Accuracy: 0.33 (+/- 0.04)
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.30      0.41      0.35       335
           2       0.00      0.00      0.00         7
           3       0.31      0.29      0.30       259
           4       1.00      0.50      0.67         2
           5       0.00      0.00      0.00        16
           6       0.14      0.03      0.05        33
           7       0.32      0.23      0.27        73
           8       0.00      0.00      0.00         5
           9       0.00      0.00      0.00        23
          10       0.00      0.00      0.00         7
          11       0.20      0.09      0.12        23
          12       0.05      0.05      0.05        44
          13       0.28      0.25      0.26       224
          14       0.50      0.08      0.14        24
          15       0.00      0.00      0.00         1
          16       0.00      0.00      0.00         1
          17       0.62      0.57      0.59       254
          18       0.00      0.00      0.00         3
          19       0.47      0.64      0.55        14
          20       0.00      0.00      0.00         4

    accuracy                           0.33      1355
   macro avg       0.20      0.15      0.16      1355
weighted avg       0.34      0.33      0.33      1355
```

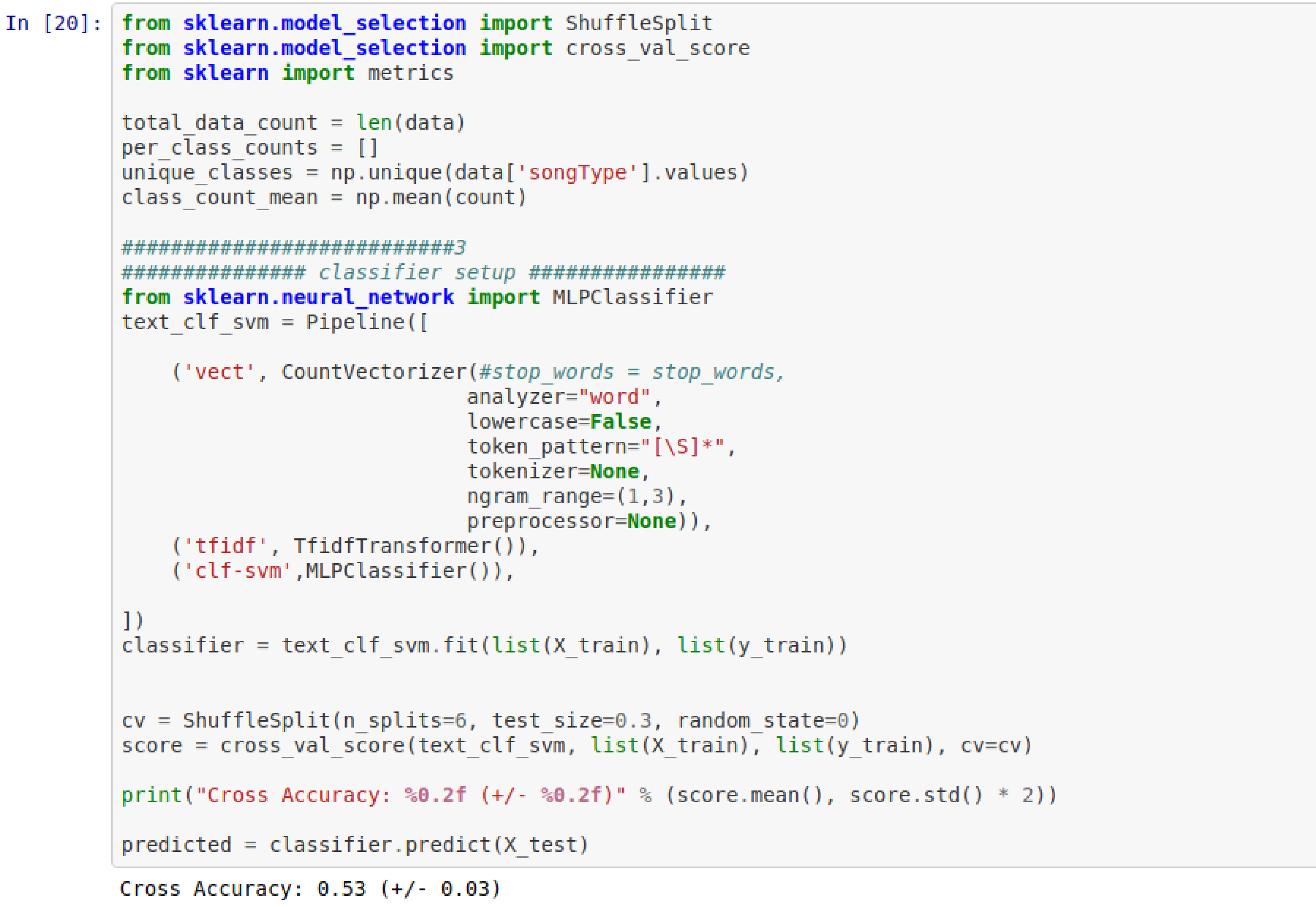
The confution matrix of DecisionTreeClassifier Model.

```
Out[34]: <AxesSubplot:>
```



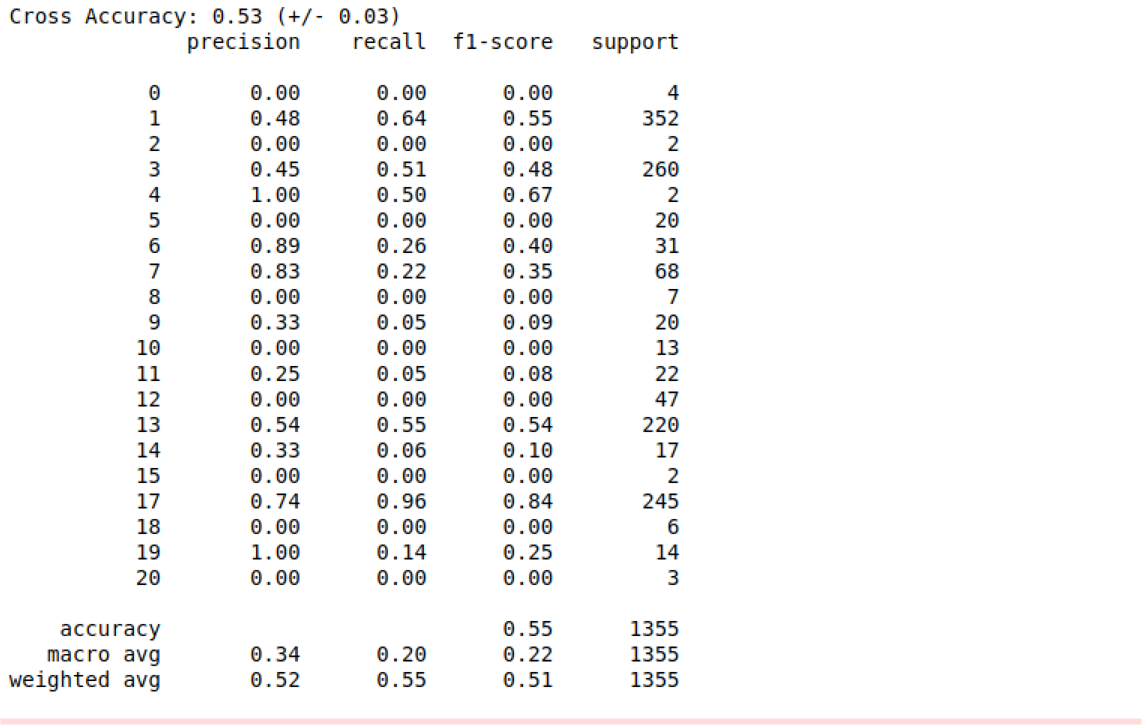
5 Artificial neural networks :

The cross accuracy of TfidfTransformer Model is shown in the below picture.



```
In [20]: from sklearn.model_selection import ShuffleSplit
         from sklearn.model_selection import cross_val_score
         from sklearn import metrics

         total_data_count = len(data)
         per_class_counts = []
         unique_classes = np.unique(data['songType'].values)
         class_count_mean = np.mean(count)

         ###########################3
         ############### classifier setup ################
         from sklearn.neural_network import MLPClassifier
         text_clf_svm = Pipeline([

             ('vect', CountVectorizer(#stop_words = stop_words,
                                     analyzer="word",
                                     lowercase=False,
                                     token_pattern="[\S]*",
                                     tokenizer=None,
                                     ngram_range=(1,3),
                                     preprocessor=None)),
             ('tfidf', TfidfTransformer()),
             ('clf-svm',MLPClassifier()),

         ])
         classifier = text_clf_svm.fit(list(X_train), list(y_train))


         cv = ShuffleSplit(n_splits=6, test_size=0.3, random_state=0)
         score = cross_val_score(text_clf_svm, list(X_train), list(y_train), cv=cv)

         print("Cross Accuracy: %0.2f (+/- %0.2f)" % (score.mean(), score.std() * 2))

         predicted = classifier.predict(X_test)

         Cross Accuracy: 0.53 (+/- 0.03)
```

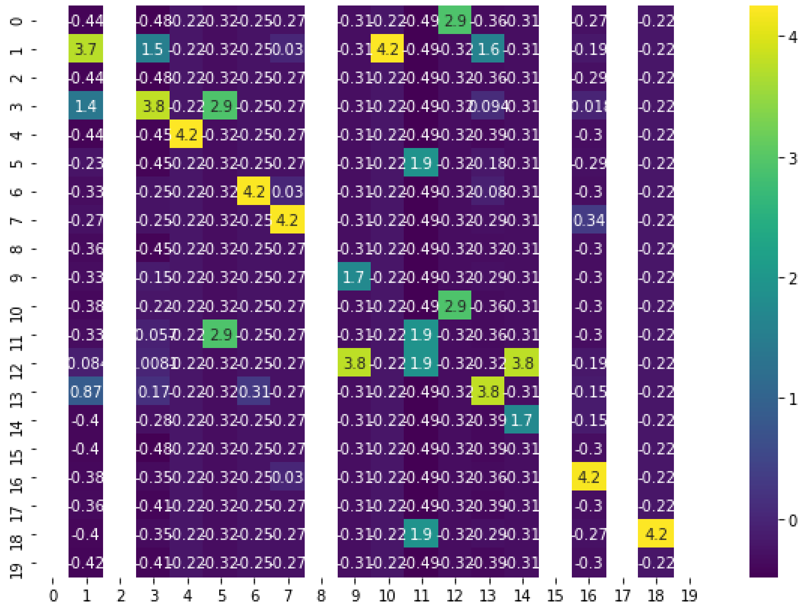
Here is the accuracy,macro average & weighted average are given.



```
Cross Accuracy: 0.53 (+/- 0.03)
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         4
           1       0.48      0.64      0.55       352
           2       0.00      0.00      0.00         2
           3       0.45      0.51      0.48       260
           4       1.00      0.50      0.67         2
           5       0.00      0.00      0.00        20
           6       0.89      0.26      0.40        31
           7       0.83      0.22      0.35        68
           8       0.00      0.00      0.00         7
           9       0.33      0.05      0.09        20
          10       0.00      0.00      0.00        13
          11       0.25      0.05      0.08        22
          12       0.00      0.00      0.00        47
          13       0.54      0.55      0.54       220
          14       0.33      0.06      0.10        17
          15       0.00      0.00      0.00         2
          17       0.74      0.96      0.84       245
          18       0.00      0.00      0.00         6
          19       1.00      0.14      0.25        14
          20       0.00      0.00      0.00         3

    accuracy                           0.55      1355
   macro avg       0.34      0.20      0.22      1355
weighted avg       0.52      0.55      0.51      1355
```

The confution matrix of DTfidfTransformer Model.

```
Out[24]: <AxesSubplot:>
```



2.
The explanation behind ”The model gives very low f1 score for some classes but not the same for others”is given below :

We know that the F1/F Score is a measure of how accurate a model is by using Precision and Recall following the formula of:

F1 Score = 2 * ((Precision * Recall) / (Precision + Recall))

Precision is commonly called positive predictive value. It is also interesting to note that the PPV can be derived using Bayes' theorem as well.
Precision = True Positives / (True Positives + False Positives)

Recall is also known as the True Positive Rate and is defined as the following:
Recall = True Positives / (True Positives + False Negatives)

If the precision is very low and recall value gets very high then the F1 score will become very low.But it should become the average of precision and recall.The alternative situation aslo behave the same. So, In the end, We can say. Some model gives the high precision and high recall value ,which are made the F1 score high.But if one's score gets very low then the F1 score also become very low.

3.
The low f1 score issue is tried to fix in below :

If the F1-score is the figure of merit, I would try to tune the class weights. It should be pretty easy, if we have a binary classification problem. We can feed class weight a dictionary with the weights for each class.
Here's a little example.

```
clf = RandomForestClassifier()
params = {'class_weight':[{0:neg_weight, 1:1} for neg_weight in np.arange(1.0,
5.0, 0.5)]}
gs = GridSearchCV(estimator= clf, param_grid = params, cv = 5)
gs.fitX_train, y_train
```

The End