*****

RAMAKRISHANA MISSION VIVEKANANDA EDUCATIONAL AND RESEARCH INSTITUTE

BELUR MATH, HOWRAH, 711202 WB

Institute Name :

SCHOOL OF MATHEMATICAL SCIENCE

DEPARTMEN NAME : COMPUTER SCIENCE

Assignment : 1

Submited By/Name : JEETU KUMAR
Registration No./Roll No. : B18732
Program Name : M.Sc. Big Data Analytics
Semester : 2nd, January - May
Paper : Machine Learning
Instructor/Paper-Instructor : Dr. Tanmay Basu

April 1, 2019

Assignment Release Date: March 11, 2018
Due Date Of Submission: March 31, 2018

1
TEXT DOCUMENT CLASSIFICATION

1 Introduction

Lets make a look on abstract of what we have given with and what are we wish to do.
We have *data set* containing 499 files having number of line of texts. These 499 text files are from 10 different classes. we call this data part of data as *training data*[1][2] where classes of these files will be the *class label* [1][2] and we have a other data set containing 185 text files(*test data* [1][2]) all are from same 10 classes. We wish to learn(infer) a function from training data which should infer(predict) class label of test data. Here our objective is to learn 'best function' or better we will say 'most appropriate function'(which takes only reasonable time to find) to predict the class label of test data; And also we wish to extract(identify) 'property of data' and some natural information with in data which either supports to our classifier(*function which we have inferred to predict the class label of test data*)[2][1] or resists.
Here after in section 2 we briefly make look on some basics of classifier which we are going try.Then in section 3 we will describe the evaluation metric which we will use to evaluate the performance of our classifier. And in section 4 we discuss the different tune of parameter of classifier with their results and adopted setup(*value of hyper parameters*[1]) for classifier and reason them them in terms of performance.And in section 5 we will discuss the results of classifier and with their good and bad we will reason their performance and the natural information and hindrance with in data. And at last in subsection 5.2 we make conclusion and further scope.

2 Methods And Classifiers

Here we will briefly discuss about the classifier which we are going to use for classification as one method.

2.1 Naive Bayes

One highly practical 'Bayesian learning method'[1][2] is the 'Naive Bayes Learner'[1][2], often called the naive Bayes classifier. In some domains its performance has been shown to be comparable to that of 'Neural Network'[1][2] and 'decision tree learning'[1][2] e.g. the practical problem of learning to classify natural language text documents[1].
The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set $\mathbb{V}$ [1].
The most probable classification of the new instance is obtained by combining the predictions of all 'hypotheses'[4][2][1], weighted by their 'posterior probabilities'[1][2].

2.2 K-Neighbors Classifier

The most basic instance-based method is the' k-NEAREST NEIGHBOR algorithm'[1][2]. This algorithm assumes all instances correspond to points in the n-dimensional space $\mathbb{R}^n$. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance[1].
K-Nearest Neighbor algorithm takes decision based on majority voting from K nearest data point (instance) from 'training-set'[1].

2.3 Logistic Regression

'Logistic regression'[2] is a widely used binary classification algorithm, a specific instance of conditional maximum entropy models. For the purpose of this chapter, we first give a very brief description of the algorithm [2].This is a hyperplane based classifier. It learns a hyperplane from training data which classifies training data almost correctly, data from one class to one side of plane and from other class to other class of plane and does same for 'test data point'[1].

2.4 Support Vector Machine(SVM)

Support Vector Machine classifier is also based on geometric concept hyperplane. It works similar to 'linear discriminant function'[1][2].The solution returned by the SVM algorithm is the hyperplane with the maximum margin, or distance to the closest points, and is thus known as the maximum-margin hyperplane[2].These closest point are known as 'support vector'[2].
It learns such hyperplane from training data point such that all the classes are almost segregated by the hyperplane and uses same learned strategy to classify the test data point.

3 Evaluation Criteria

There are different types of evaluation criteria to evaluate the performance of a classifier using the ground truths of the given data set. We shall discuss some popular methods here.Let us first look at evaluation terms for two classes cases.

3.1 Confusion Matrix

Let's there two classes cases in the data set, say, positive and negative. A classifier makes error when a positive sample is predicted as negative and a negative sample is predicted as positive. The same may be observed from table below:

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Here TP stands for true positive and it counts the number of data points correctly predicted to the positive class. FP stands for false positive and it counts

the number of data points that actually belong to the negative class, but predicted as positive (i.e., falsely predicted as positive). FN stands for false negative and it counts the number of data points that actually belong to the positive class, but predicted as negative (i.e., falsely predicted as negative). TN stands for true negative and it counts the number of data points correctly predicted to the negative class. The matrix in above figure is termed as 'Confusion Matrix'[3].

3.2 Accuracy

Accuracy of a classifier is defined as the number of samples from the data set correctly predicted to the desired classes divided by the number of total samples in the data set.The accuracy can be defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

3.3 Precision, Recall And F-Measure

The effectiveness of a classifier in class assignment can be measured by the standard precision, recall and 'f-measure'[3]. The precision and recall can be computed as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Lets generalize the concepts for m class classification with the notation of *macro average*[3] and *micro average*[3]

$$Macro\ Averaged\ Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{TP_i}{TP_i + FP_i}$$

$$Macro\ Averaged\ Recall = \frac{1}{m}\sum_{i=1}^{m}\frac{TP_i}{TP_i + FN_i}$$

$$Micro\ Averaged\ Precision = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + \sum_{i=1}^{m} FP_i}$$

$$Micro\ Averaged\ Recall = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m} TP_i + \sum_{i=1}^{m} FN_i}$$

In the above expression all the terms have same description and meaning as described in subsection 3.1 (for case of two class classification) for the i^{th} class. The f-measure combines recall and precision with an equal weight in the form of harmonic mean as follows:

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precission}$$

4 Adopted Setups And Analysis Of Results

Here we will start with parameter tuning, performance analysis and inference about data for classifier *KNN*, *SVM*, *Logistic Regression*, and *Naive Bayes* respectively.Due their exhaustive search nature $GridSearchCV$[6] takes too long time to run, so unfortunately I am unable to use it. I will use $RandomizedSearchCV$[6] which works with greedy approach. To overcome local minima problem of random search we will run number of time, followed by random search I will do manual tuning in order to adopted setup.

4.1 Parameter Tuning And Experimental Results

(Through out this section unless nothing is specified then by the caption '*Performance for first random search 0*' we will meant that $TFIDF$ normalization have not used to get that performance and by the caption '*Performance for first random search 1*' we meant that $TFIDF$ normalization have used. *This notion of used or not used only concerned with 1 and 0 respectively not with other words of caption i.e. '0 means not used and 1 means used'.*)

4.1.1 : for KNN algorithm

Here we will start with default setup(*by default I mean default parameter value pass by scikit-learn*).Lets see what is performance for default setup we are getting.

	max df	min df	ngram	idf	no. token	algorithm	leafsize	metric	K	p
0	0.3	1	1 - 1	No	5694	auto	30	minkowski	5	2
1	0.3	1	1 - 1	Yes	5694	auto	30	minkowski	5	2

Table 1: Default Configuration

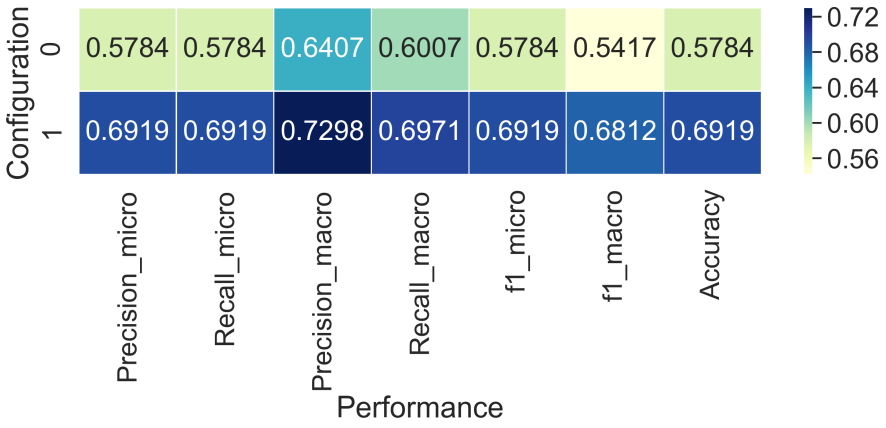

Figure 1: Performance On Default Configuration

Here we can see that normalizing term document matrix using $TFIDF$(*Term Frequency Inverse Document Frequency*) have improved the performance.Here we can mark one thing that

$$f1\_macro < Accuracy < Precision\_macro \tag{1}$$

$$Precision\_micro \approx f1\_micro \approx Accuracy \tag{2}$$

$$Recall\_micro \approx Recall\_macro \tag{3}$$

Let's see what other configurations telling for the above equations 1, 2 and 3 so that we can infer something based on it.
Let's have a look on confusion matrix of configuration given in tabel 1.

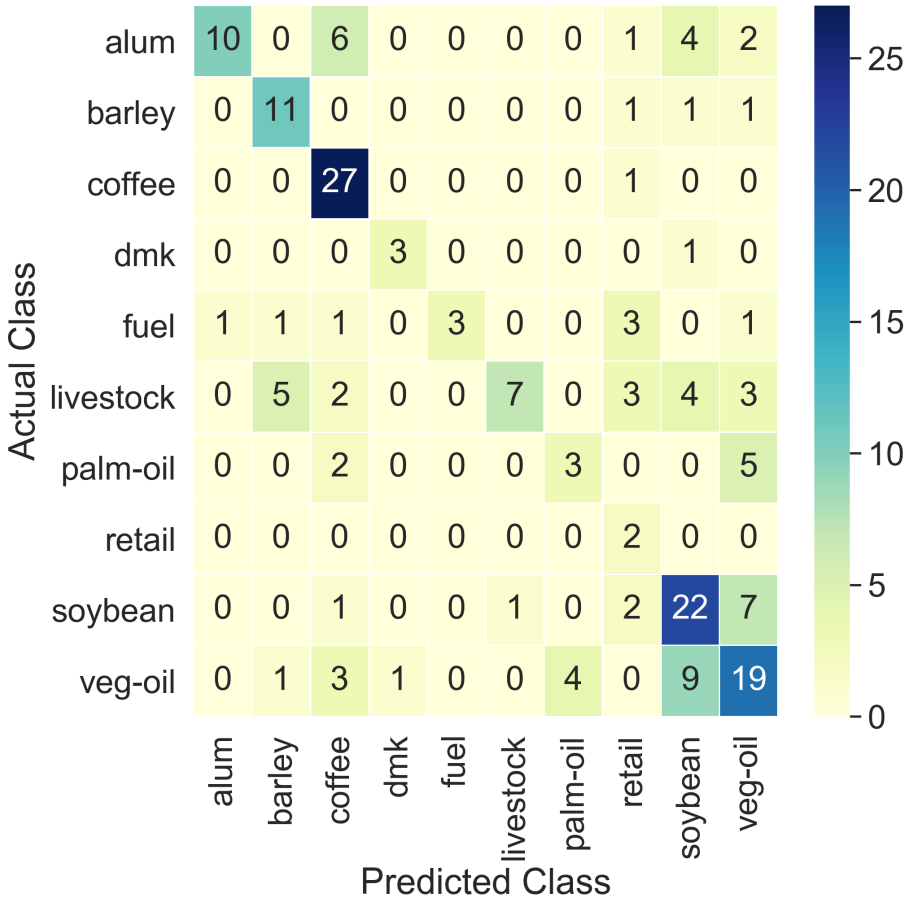

Figure 2: Confusion Matrix for configuration 0

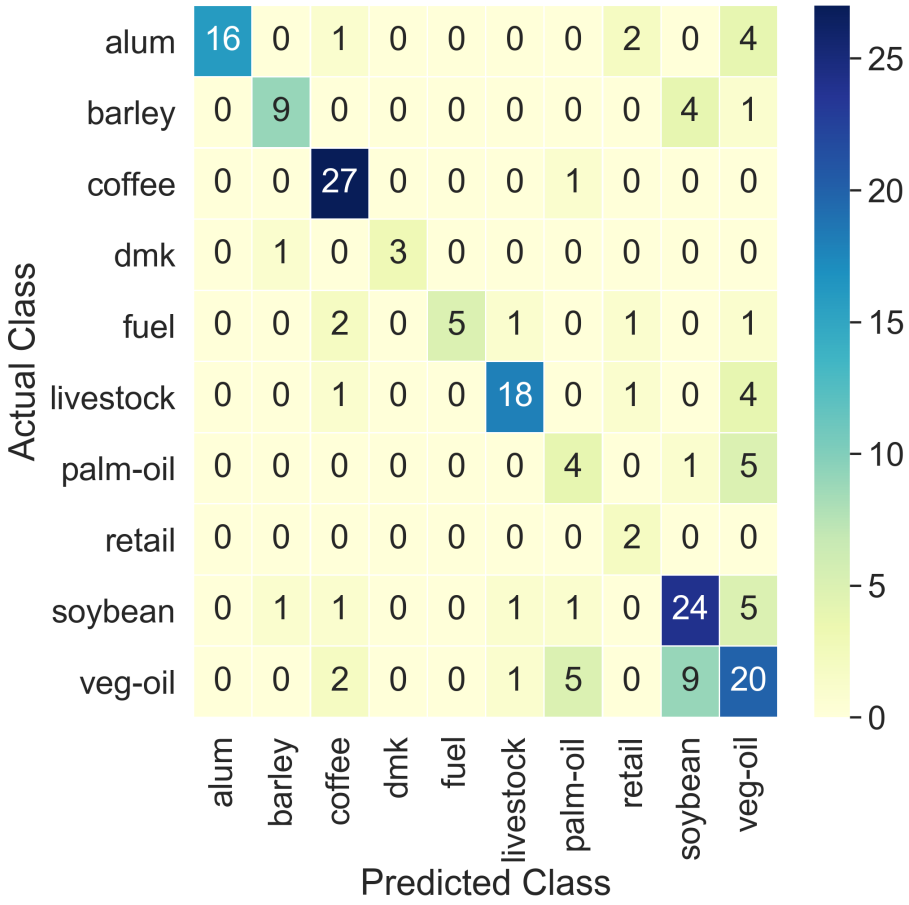

Figure 3: Confusion Matrix for configuration 1

Here we can see that there are the number at diagonal which are less than the off-diagonal number of the same classes and total in that class are also low, which tells that classes are high unbalanced and very few samples are in some classes. We can see that in figure 2 occurrence of non-zero number at off-diagonal is high and those numbers are also large then same in figure 3 this happened because we have not used $TFIDF$, which tell that randomness that a classifier do wrong have increased. Now In order to tune the hyper-parameter we use $RandomizedSearchCV$ and as it's a greedy and search approach so just we will analyses some performance result from them to validate what it infers.

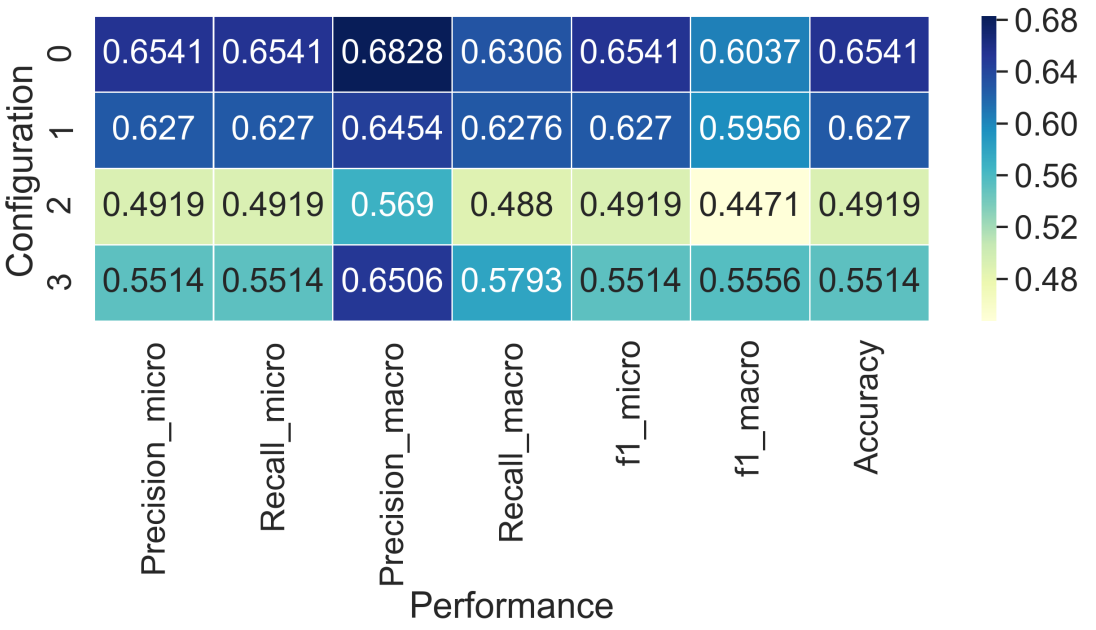

Figure 4: Performance for first random search 0

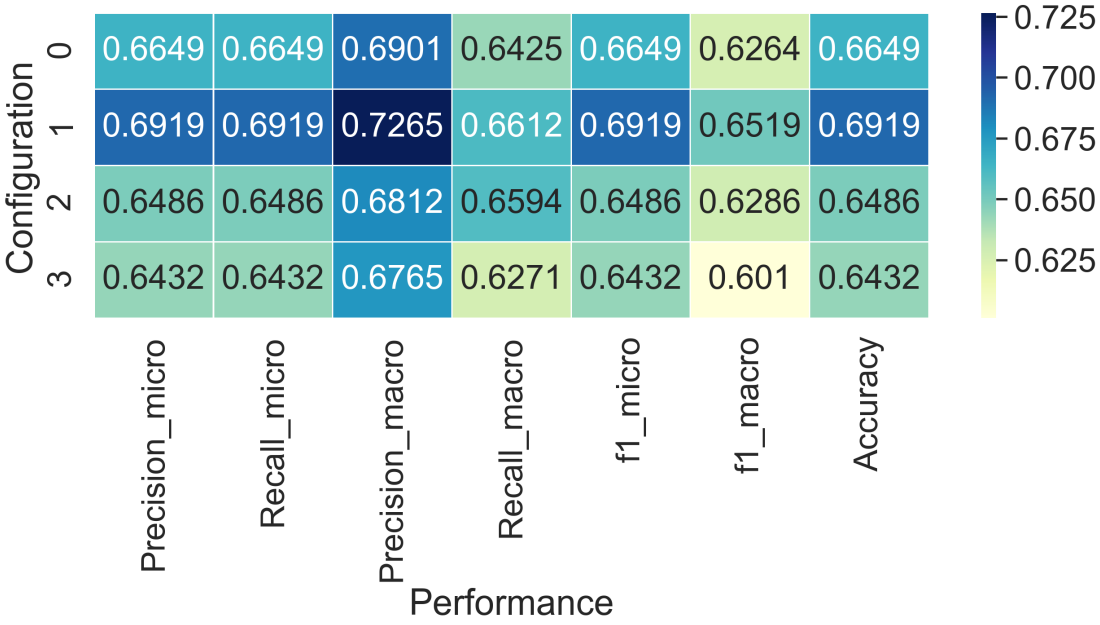

Figure 5: Performance for first random search 1

From both figure 4 and 5 we can see that for best cases the equation 1 and 2 holds good with equation 4

$$Recall\_micro \leq Recall\_macro \tag{4}$$

so we can see that 4 have a slight deviation from equation 3.Which tell that recall with in the class can be improved. But here also from validation of equation 1 and 2 we can say that classes are unbalanced. Now we will look for manual tuning of parameter.

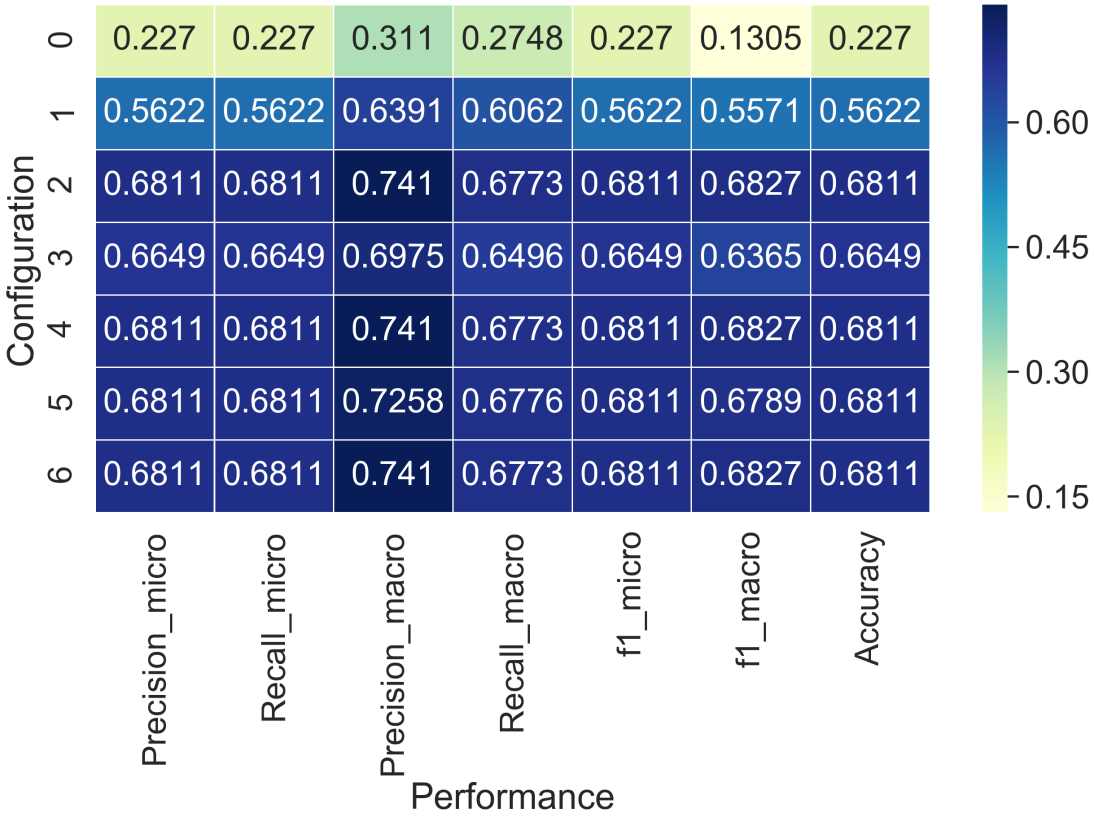

Figure 6: Performance on tuned configuration

Here we can see that increasing number of token dose not helped us and we have the best result around $K = 5$ and cosine similarity metric have worked well. In particular with given data euclidean similarity metric is in run-up. And here also we can see that equation 1, 2, 3 and 4 are highly close toward equality. Based on this we will wish to adopt the setup for KNN-algorithm as given below in table. begintable[h!] Performance of this setup over cross validation are as given below in table Here also we can see that unbalanced class impact of data set.

	max df	min df	ngram	idf	no. token	algorithm	leafsize	metric	K	p
0	0.35	1	1 - 12	No	498538	auto	30	minkowski	5	2
1	0.35	0.005	1 - 1	No	2179	auto	30	minkowski	5	2
2	0.35	0.005	1 - 1	Yes	2179	auto	30	minkowski	5	2
3	0.35	0.005	1 - 3	Yes	7128	auto	30	minkowski	5	2
4	0.35	0.005	1 - 1	Yes	2179	kd tree	30	minkowski	5	2
5	0.35	0.005	1 - 1	Yes	2179	auto	30	cosine	5	2
6	0.35	0.005	1 - 1	Yes	2179	auto	30	euclidean	5	3

Table 2: Tuned Configuration for KNN

	max df	min df	ngram	idf	no. token	algorithm	leafsize	metric	K	p
0	0.35	0.005	1 - 1	Yes	2179	auto	30	cosine	7	2
1	0.35	0.005	1 - 1	Yes	2179	kd tree	30	euclidean	7	3

Table 3: Adopted configuration for KNN

	1	2	3	4	5	6	7	8	9	10	Average	metric
0	0.521	0.608	0.409	0.5	0.647	0.882	0.588	0.812	0.6	0.466	0.603	f1 micro
1	0.459	0.486	0.237	0.536	0.558	0.794	0.492	0.656	0.533	0.435	0.518	f1 macro

Table 4: Performance for 10 fold cross validation

4.1.2 : for SVM algorithm

Here we will start with default setup. Lets see what is performance for default setup we are getting.

	max df	min df	ngram	idf	no. token	C	degree	gamma	kernel	max iter
0	0.3	1	1 - 1	No	5694	1	3	auto	rbf	-1
1	0.3	1	1 - 1	Yes	5694	1	3	auto	rbf	-1

Table 5: Default Configuration

Although we have got very very bad result here, which we will reason latter on. But we can see that normalizing term document matrix using $TFIDF$(*Term Frequency Inverse Document Frequency*) have improved the performance. Let's have a look on confusion matrix of configuration given in tabe 5.

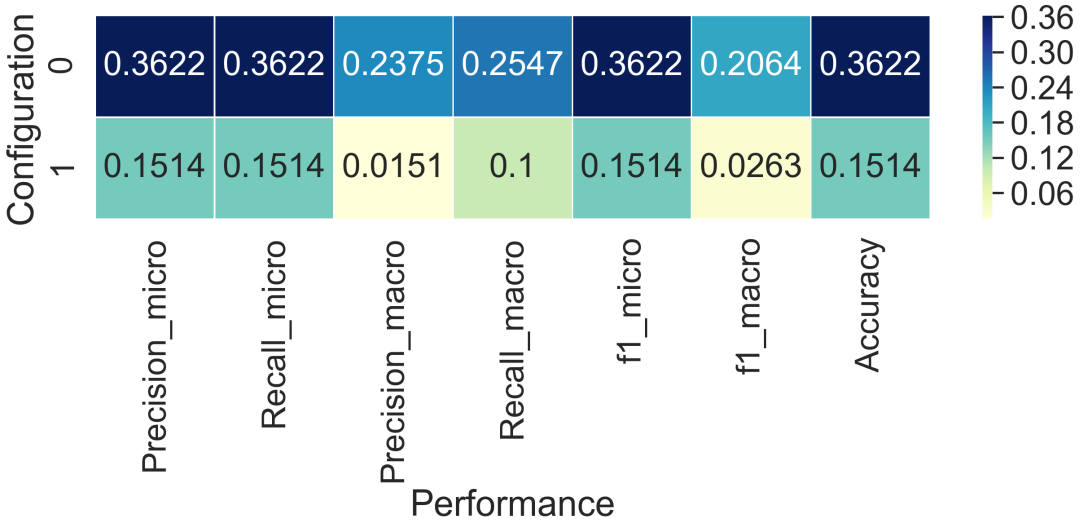

Figure 7: Performance On Default Configuration

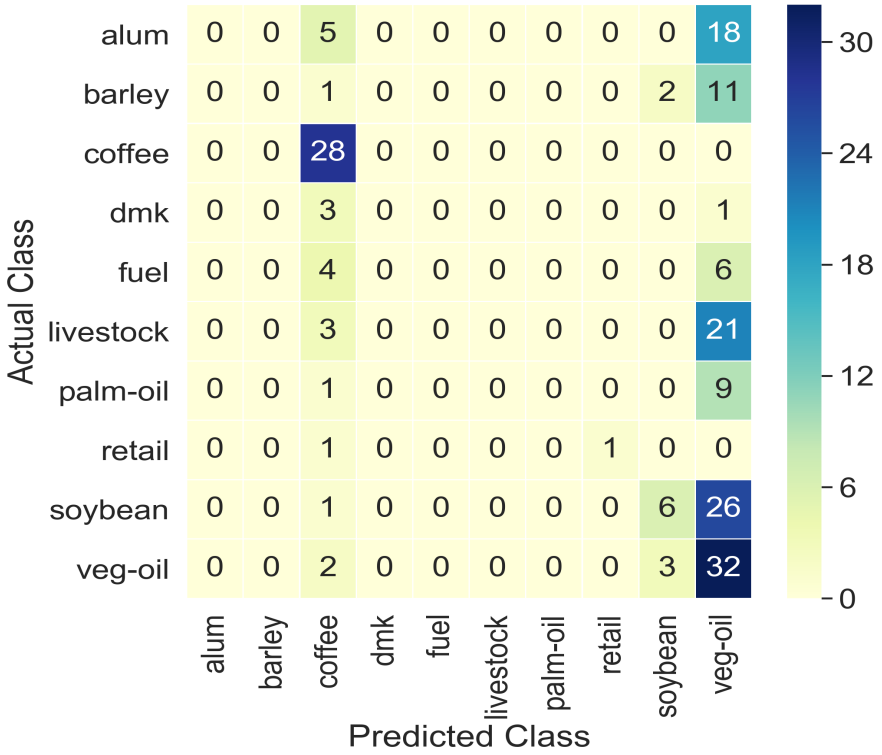

Figure 8: Confusion Matrix for configuration 0

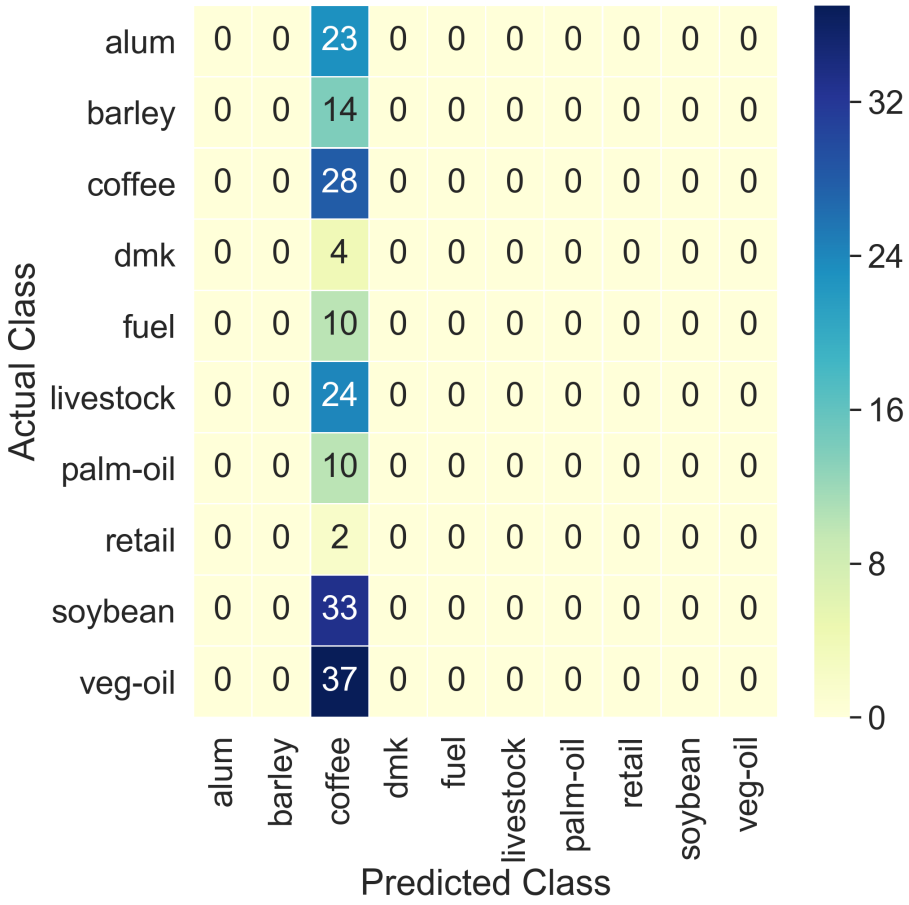

Figure 9: Confusion Matrix for configuration 1

Here we can see that due improper parameter value classifier have not learned all the classes.It just learned one classes when data is normalized with $TFIDF$ and 3 classes in other first case.
Now In order to tune the hyper-parameter we use $RandomizedSearchCV$ and as it's a greedy and search approach so just we will analyses some performance result from them to validate what it infers.

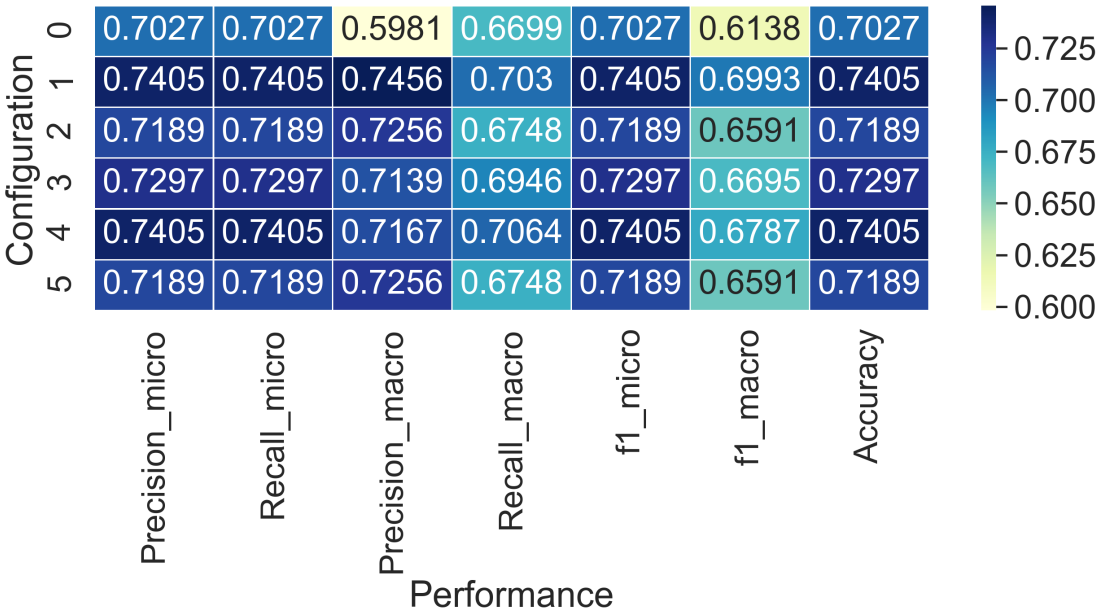

Figure 10: Performance for first random search 0

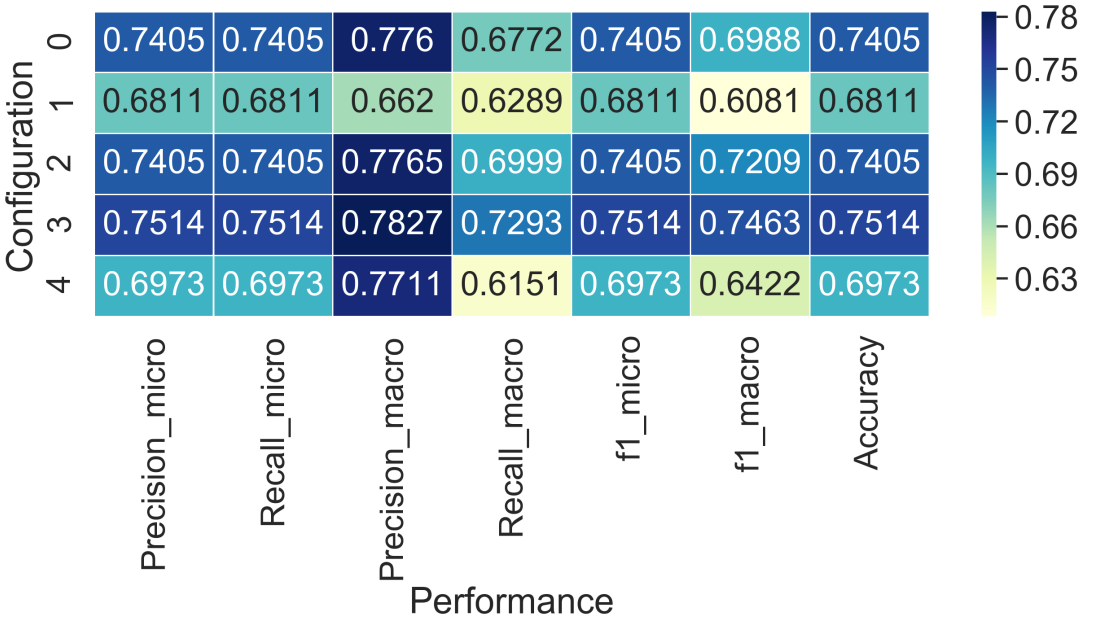

Figure 11: Performance for first random search 1

Here from figure 10 and 11 we can see that for best cases equation 1, 2, 4 more close towards equality. Most of these best performance I have got for linear kernel. So I will prefer to use $RandomizedSearchCV$ for $Linear - SVM$ and the performance results are :

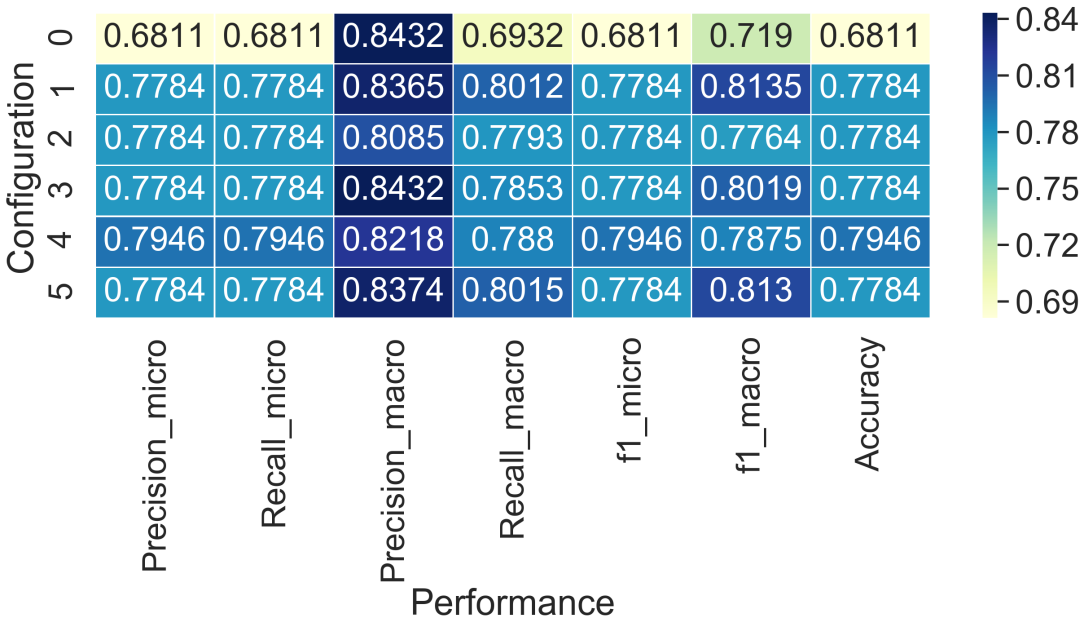

Figure 12: Performance for second random search 1

Here from figure 12 we can see that for best cases equation 2, 4 are holds with inequality. Here we have a complete hike in precision_macro, f1_macro, and accuracy.Here we can see that a clear relation

$$Accuracy < f1\_macro < Precision\_macro \tag{5}$$

which completely differ from equation 1 and 1 is challenged to hold.Now we will look for manual tuning of parameter. Here I will just present the performance of classifier on tuned parameter.

	max df	min df	ngram	idf	no. token	C	gamma	kernel	max iter
0	0.35	1	1 - 3	YES	83216	1	1	Linear	1000
1	0.35	1	1 - 3	No	83216	1	1	Linear	1000
2	0.35	0.005	1 - 3	Yes	7128	1	1	Linear	1000
3	0.35	0.005	1 - 7	Yes	11883	1	1	Linear	1000

Table 6: Tuned Configuration for SVM

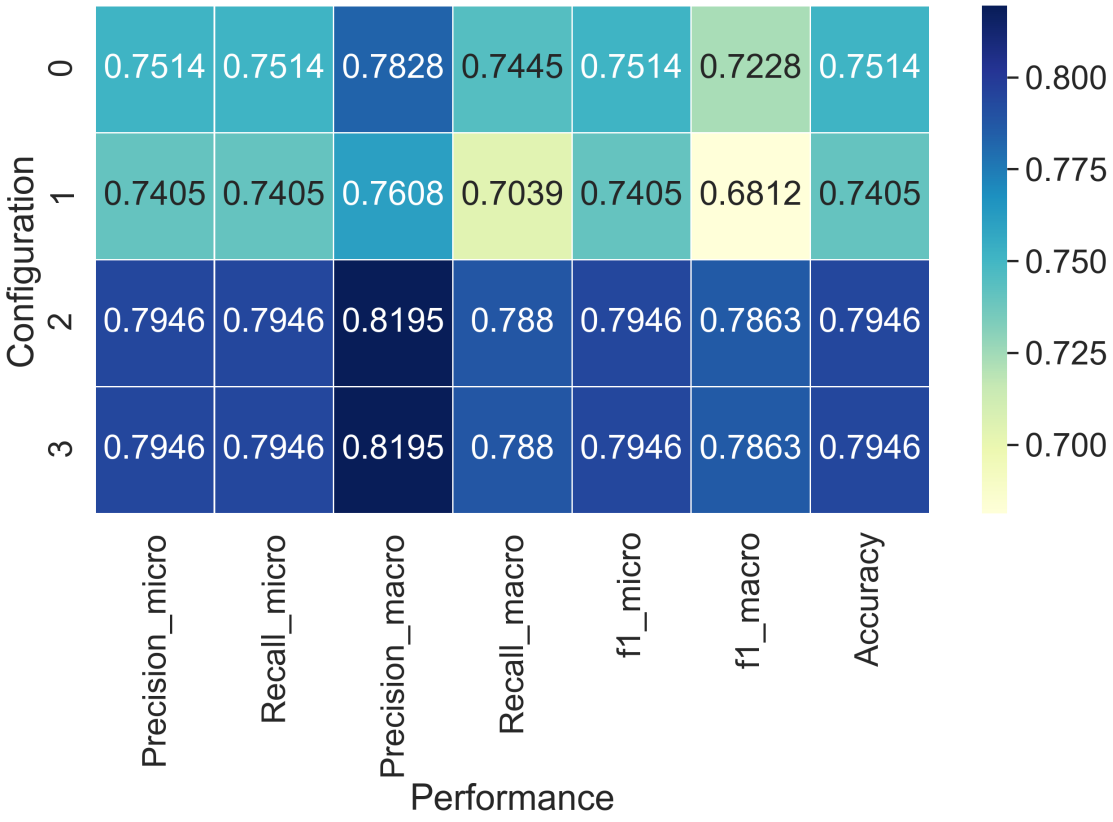

Figure 13: Performance for tuned configuration

Here from the table 6 and figure 13 we can see that higher number of token does not have helped us and neither we have got any improvement due to tracking more words in a sentence. we have got best performance by tracking only 3 words in sentence which tells that most of content in the document are likely to be highly similar irrespective of their class. we may try to validate this from our equation 5.
Based on performance shown in figure 13 I have adopted the configuration for $'Linear\ SVM'$ as given below in table and performance in figure and cross validation score in table

	max df	min df	ngram	idf	no. token	C	gamma	kernel	max iter
0	0.35	0.005	1 - 3	Yes	7511	7	1	Linear	2000

Table 7: Adopted Configuration for SVM

Let's have a look on confusion matrix of classifier.

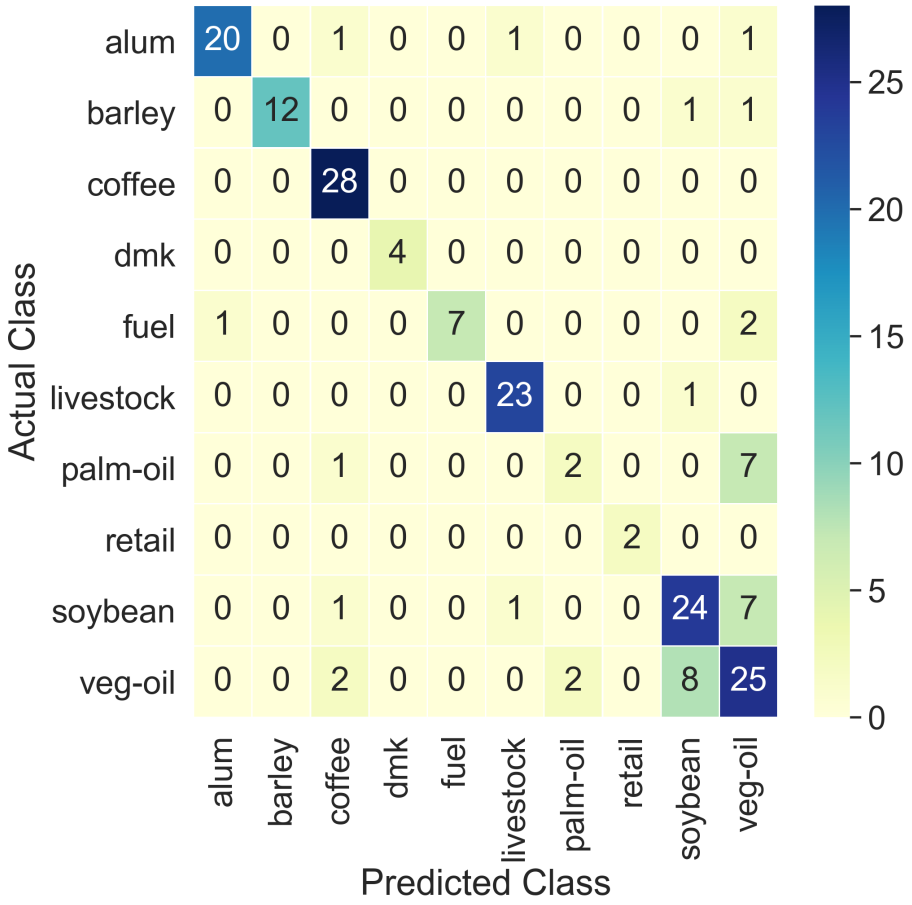

Figure 14: Confusion matrix for adopted configuration

	1	2	3	4	5	6	7	8	9	10	Average	metric
0	0.759	0.792	0.773	0.803	0.803	0.816	0.816	0.851	0.847	0.782	0.804	f1 micro
1	0.778	0.794	0.773	0.815	0.801	0.82	0.776	0.822	0.824	0.802	0.801	f1 macro

Table 8: Performance of Adopted Configuration on 10 fold cross validation

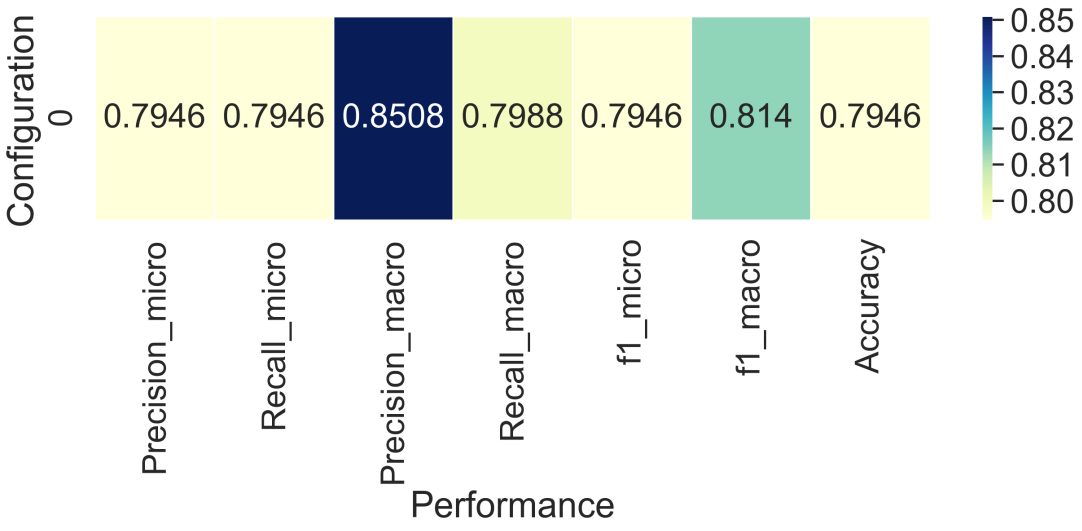

Figure 15: Performance of adopted configuration

Here in table 8 we can see that although performance of classifier in terms of f1_micro and f1_macro have very very low difference which tells that learning of classifier is a kind saturated, from this we can also conclude that classifier is what ever doing bad or good with in the class same it is doing through out the classes(means for complete document). which tends to tell classes are overlapped.

4.1.3 : for Logistic Regression algorithm

Here we will start with default setup. Lets see what is performance for default setup we are getting.

	max df	min df	ngram	idf	no. token	C	gamma	solver	max iter
0	0.3	1	1 - 1	No	5694	1	1	lbfgs	100
1	0.3	1	1 - 1	Yes	5694	1	1	lbfgs	100

Table 9: Default Configuration

Here we can see that default setup of logistic regression have have good performance, But one thing here we can see that performance have reduced by normalizing the data with $TFIDF$.
Let's have a look on confusion matrix of configuration given in table 9.

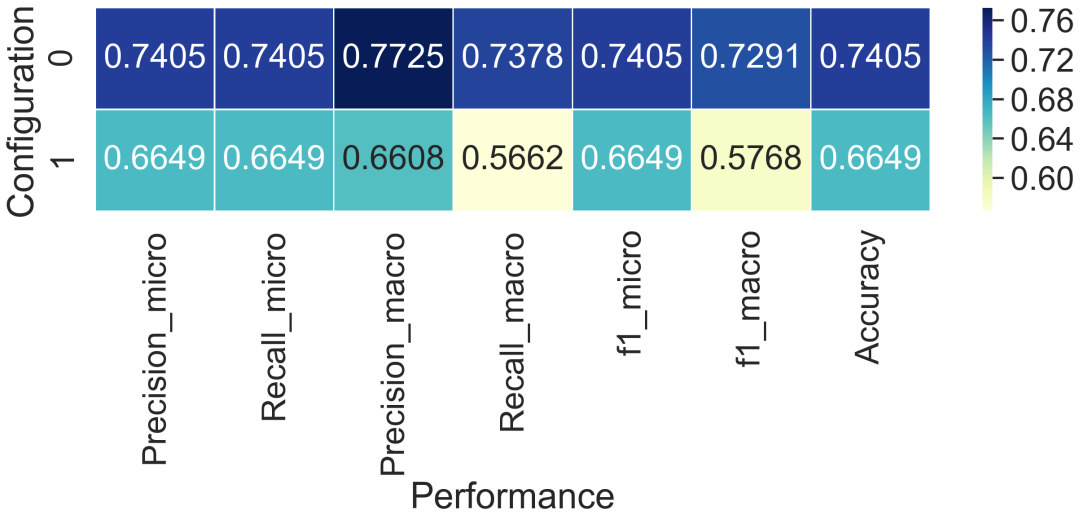

Figure 16: Performance On Default Configuration

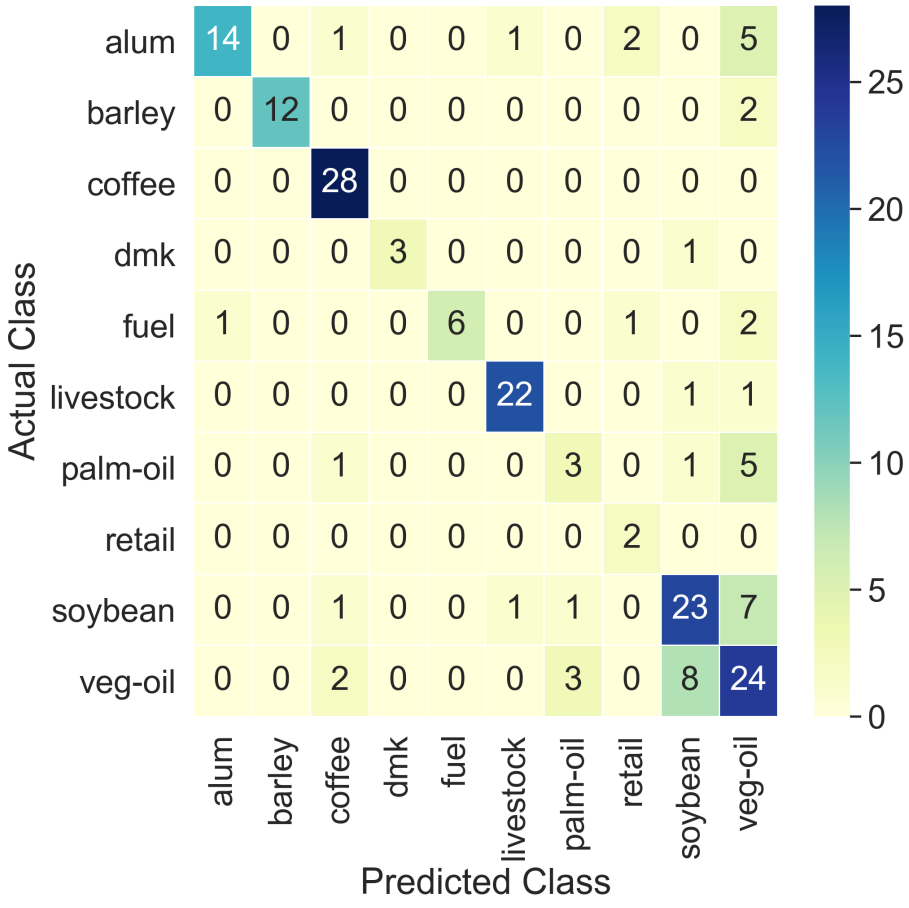

Figure 17: Confusion Matrix for configuration 0

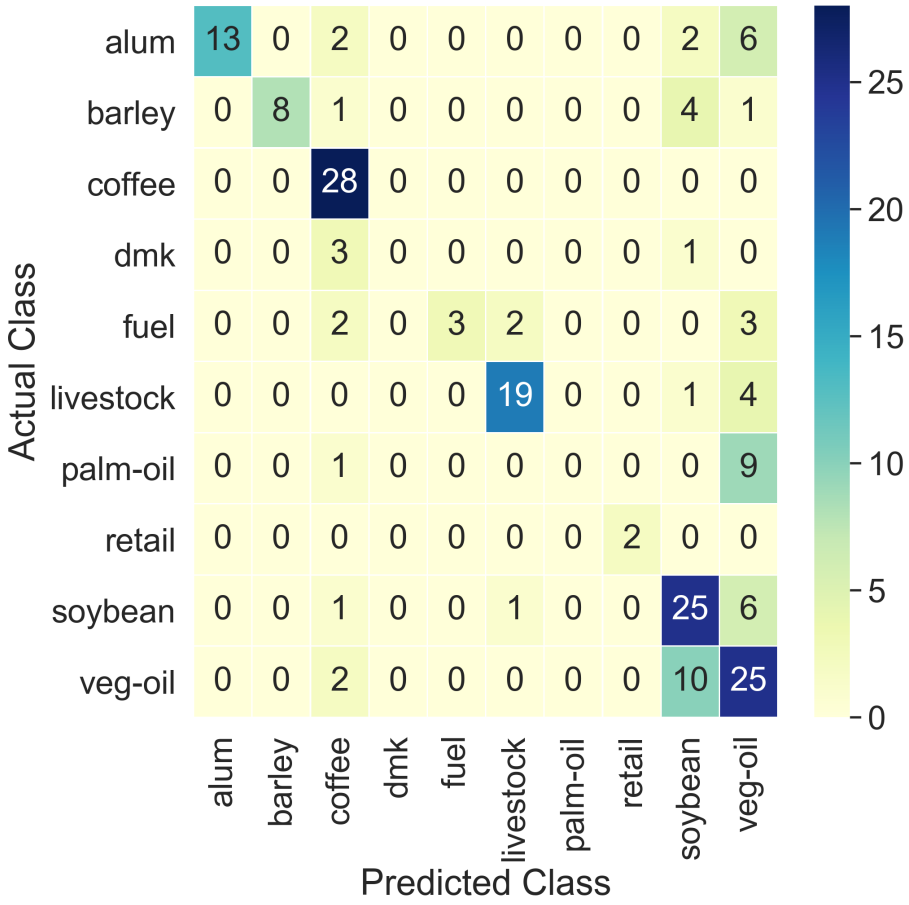

Figure 18: Confusion Matrix for configuration 1

Here also we can validate the same that we have seen in case of other classifier but with one opposite train of use of $TFIDF$.
Now In order to tune the parameters we will take help of $RandomizedSearchCV$ and as it's a greedy and search approach so just we will analyses some performance result from them to validate what it infers.

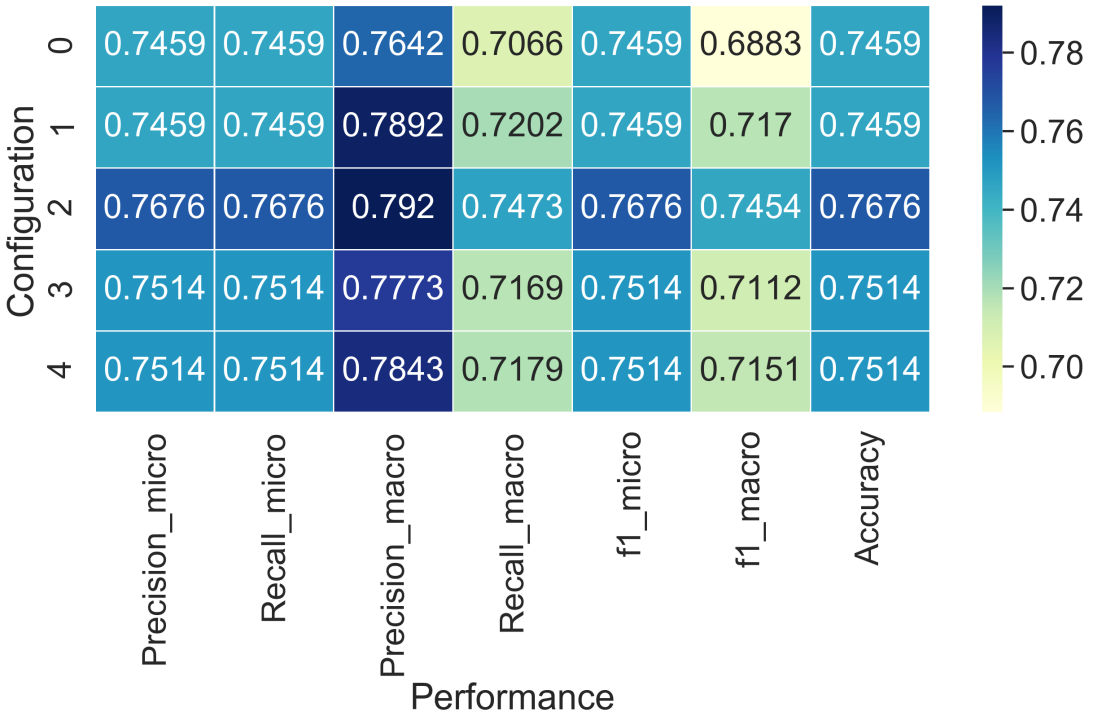

Figure 19: Performance for first random search 0

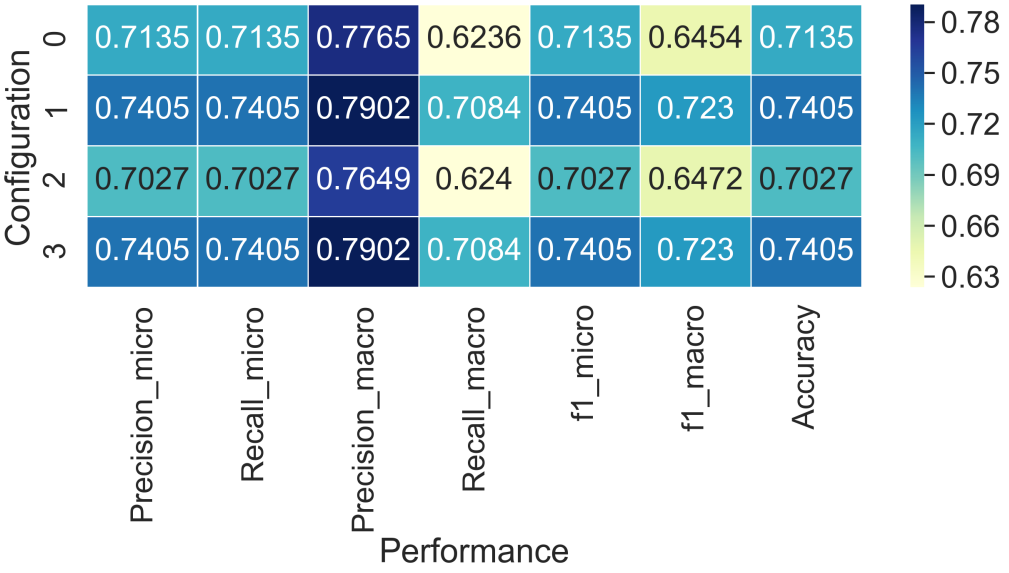

Figure 20: Performance for first random search 1

Here we can see that train of the performance metric have changed. By looking on figure 16, 19 and 20 we can see that the equation 1 and 2 holds good with relation

$$Recall\_macro < Recall\_micro \tag{6}$$

which is completely opposite to equation 4, and 4 is challenged to hold. Based on the above performance analysis we adopted the setup and performance of that setup is as given below in table 10 and figure 21.

	max df	min df	ngram	idf	no. token	C	gamma	solver	max iter
0	0.3	0.005	1 - 3	No	7128	6	0.3	liblinear	1500
1	0.3	0.005	1 - 3	Yes	7128	6	0.3	liblinear	1500

Table 10: Adopted Configuration

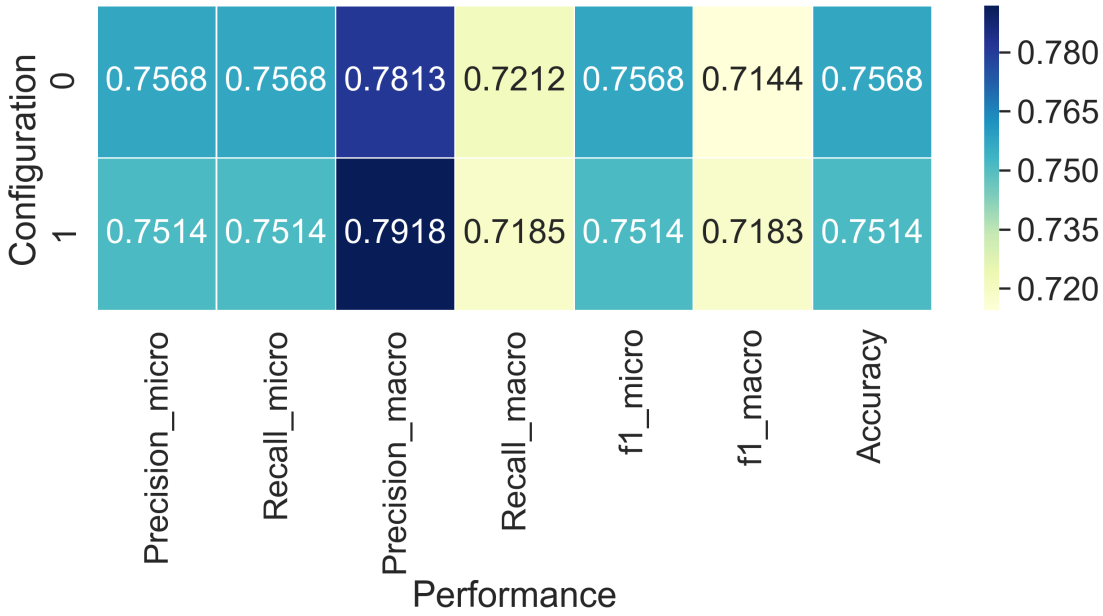

Figure 21: Performance On Adopted Configuration 1

	1	2	3	4	5	6	7	8	9	10	Average	metric
0	0.759	0.773	0.754	0.823	0.764	0.775	0.816	0.829	0.782	0.739	0.781	f1 micro
1	0.778	0.758	0.704	0.834	0.750	0.706	0.776	0.81	0.784	0.735	0.763	f1 macro

Table 11: Performance of Adopted Configuration 1 on 10 fold cross validation

Lets have a look on cross validation score and confusion matrix for this setup

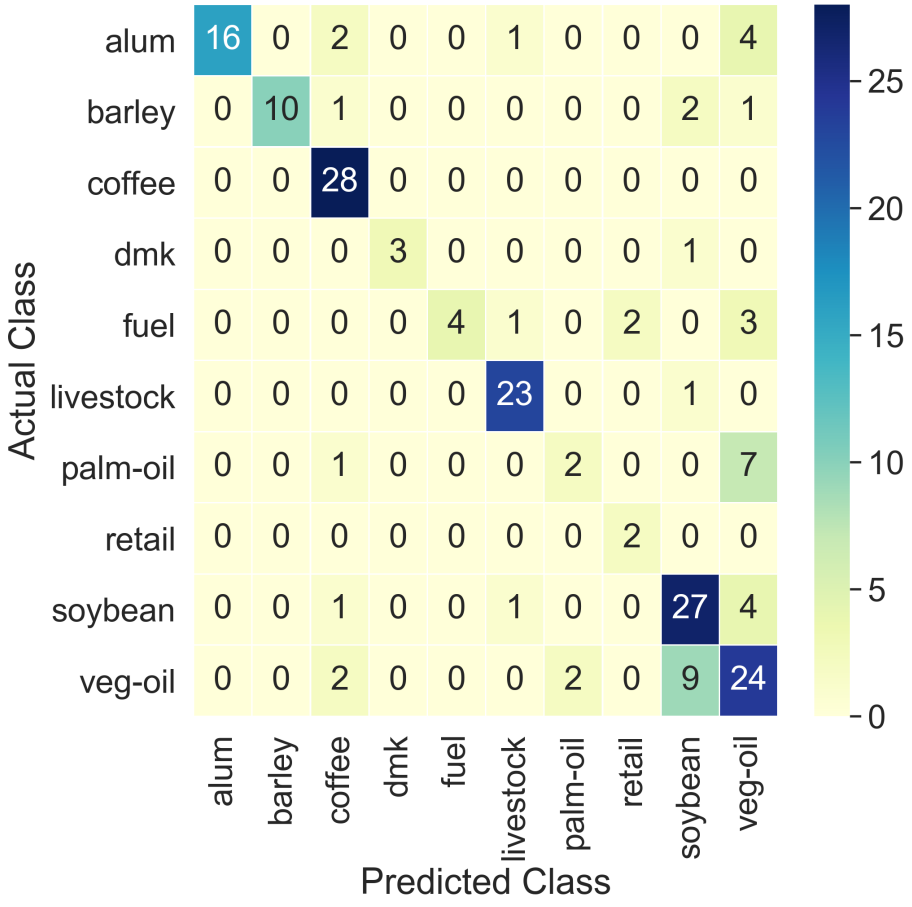

Figure 22: Confusion Matrix for adopted configuration 1

Here from table 12 we can see that cross validation performance of logistic

	1	2	3	4	5	6	7	8	9	10	Avg.	metric
0	0.759	0.792	0.773	0.823	0.803	0.795	0.816	0.851	0.869	0.804	0.808	f1 micro
1	0.778	0.771	0.773	0.834	0.806	0.812	0.773	0.806	0.846	0.818	0.802	f1 macro

Table 12: Performance of Adopted Configuration 0 on 10 fold cross validation

regression classifier is high and very very close to cross validation performance of SVM see table 8. We can see that of Logistic Regression With using $TFIDF$ (see table 11) and without using $TFIDF$ (see table 12) both case it is better but when we have not used $TFIDF$ it is higher.

In case of Logistic Regression also I have got some other setup whose performance is very-very close to one I have adopted; so due to lower computational cost of $'liblinear'$ solver I have adopted the mentioned setup.

4.1.4 : for Bernoulli Naive Bayes

Lets start with default setup. Lets see what is performance for default setup we are getting.

	max df	min df	ngram	idf	no. token	alpha
0	0.3	1	1 - 1	No	5694	1
1	0.3	1	1 - 1	Yes	5694	1

Table 13: Default Configuration

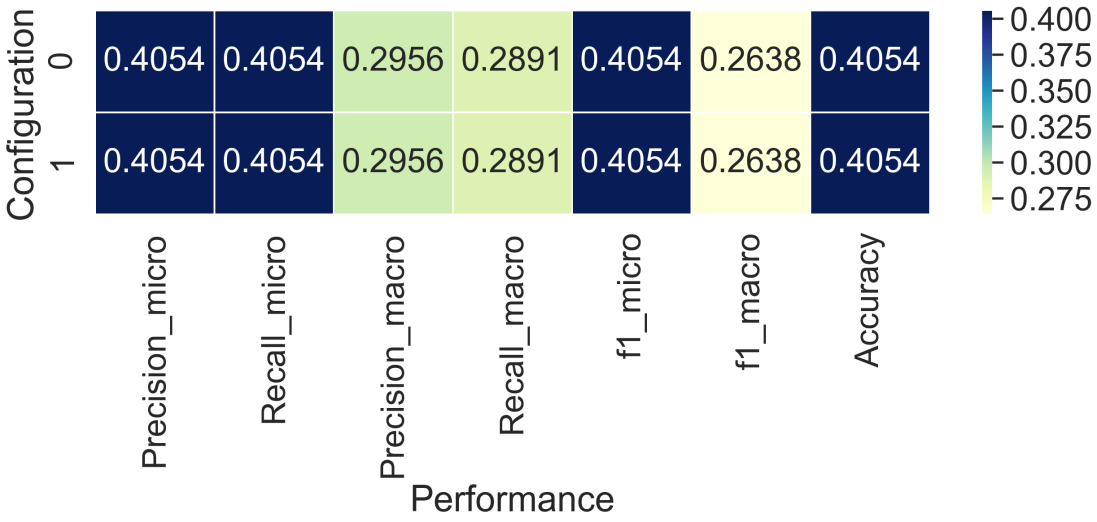

Figure 23: Performance On Default Configuration

Here from table 13 and 23 we can see performance is worst then coin toss. And we can see that there is a no effect of $TFIDF$ normalization.

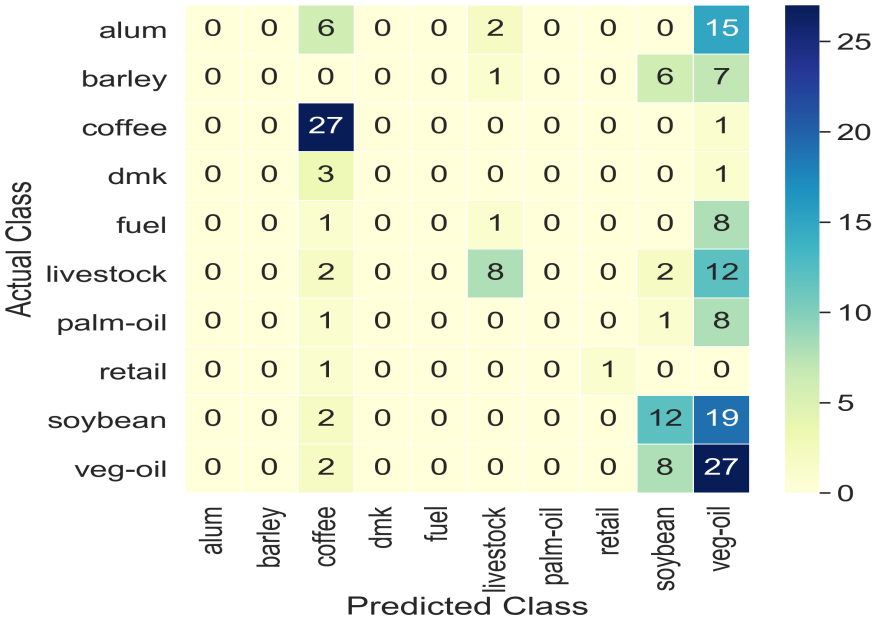

Figure 24: Confusion Matrix for configuration 0

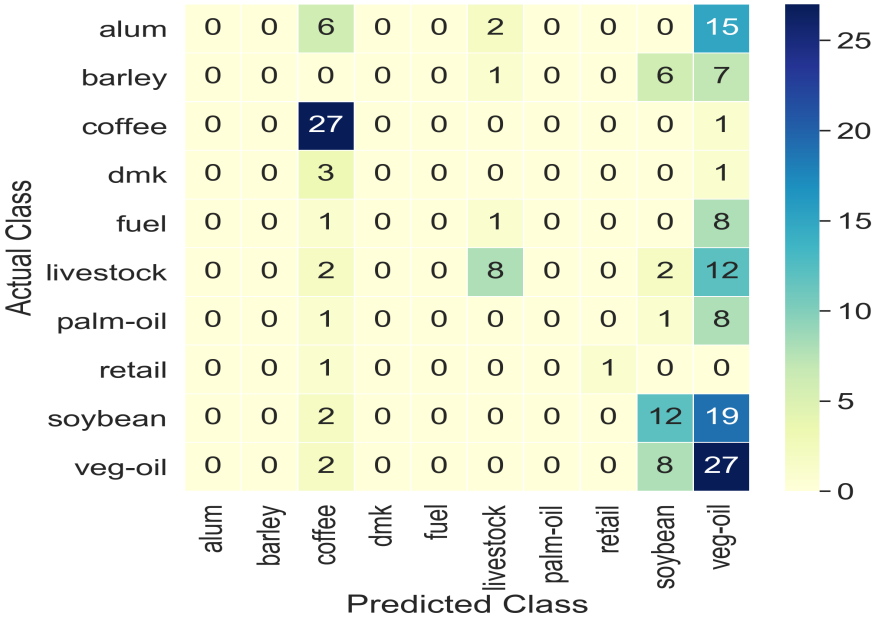

Figure 25: Confusion Matrix for configuration 1

We may look on confusion matrix to see how worst it have got. From the figure 24 and 25We can see that there are higher number on off diagonal, and most of them are predicted to one class.
Now In order to tune the parameters we will take help of *RandomizedSearchCV* and as it's a greedy and search approach so just we will analyses some performance result from them to validate what it infers.

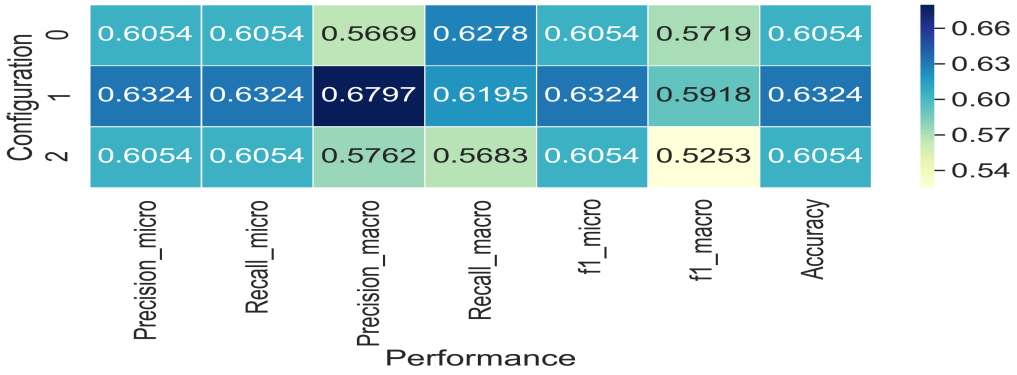

Figure 26: Performance for first random search 0

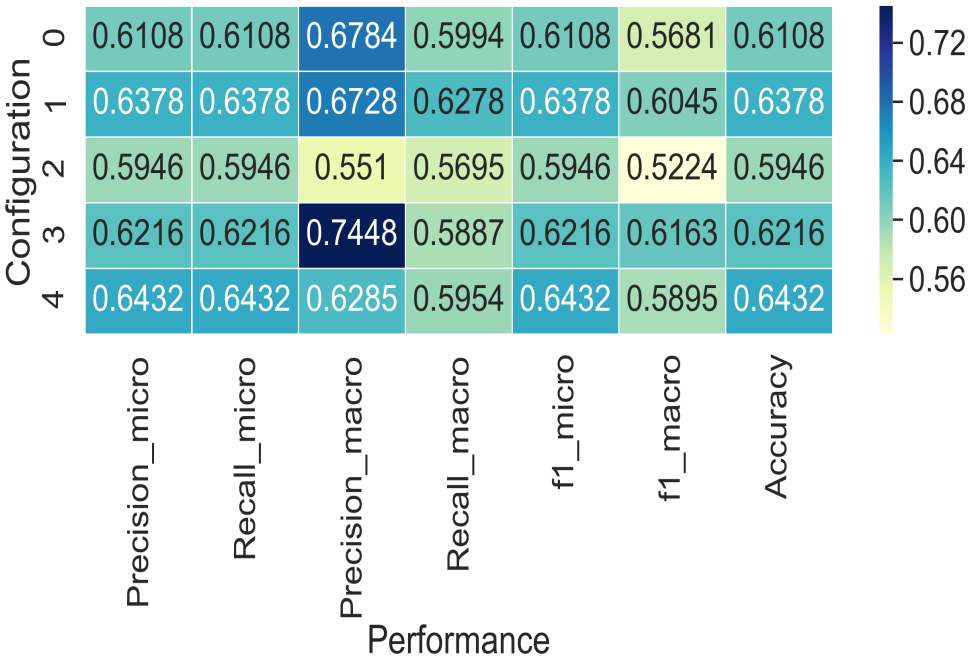

Figure 27: Performance for first random search 1

From the performance shown in figure 26 and 27 we can see that our equation 1, 2 and 6 holds good.Based on the above performance we will adopt our setup as follows

	max df	min df	ngram	idf	no. token	alpha
0	0.4	0.01	1 - 2	No	2439	0.3
1	0.4	0.01	1 - 2	Yes	2734	0.3

Table 14: Adopted Configuration

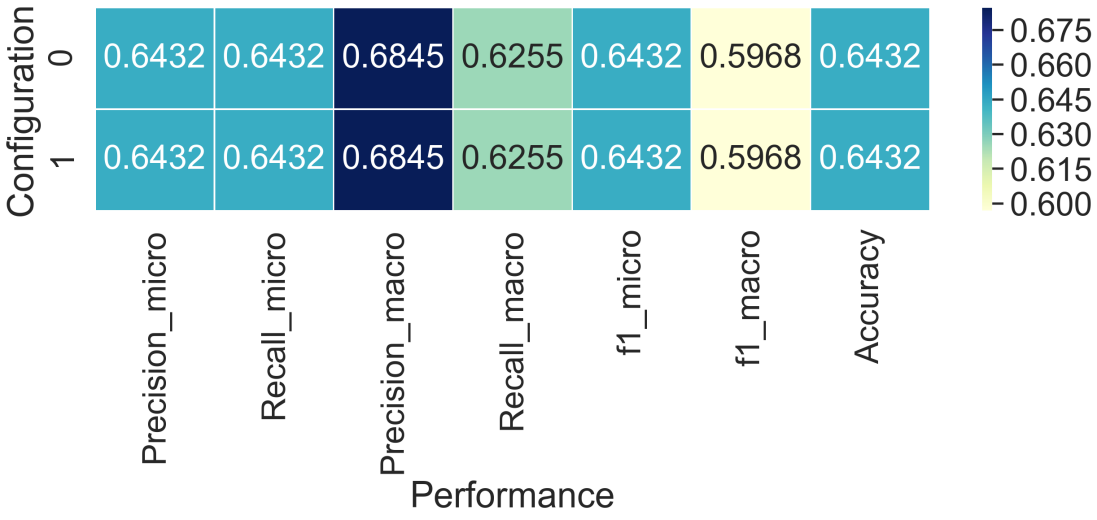

Figure 28: Performance On Adopted Configuration

	1	2	3	4	5	6	7	8	9	10	Avg.	metric
0	0.685	0.698	0.735	0.647	0.686	0.734	0.673	0.659	0.760	0.565	0.684	f1 micro
1	0.719	0.657	0.665	0.717	0.702	0.647	0.697	0.612	0.772	0.586	0.677	f1 macro

Table 15: 10 fold Cross validation performance Adopted Configuration 0

	1	2	3	4	5	6	7	8	9	10	Avg.	metric
0	0.685	0.698	0.735	0.647	0.686	0.734	0.673	0.659	0.760	0.565	0.684	f1 micro
1	0.719	0.657	0.665	0.717	0.702	0.647	0.697	0.612	0.772	0.586	0.677	f1 macro

Table 16: 10 fold Cross validation performance Adopted Configuration 1

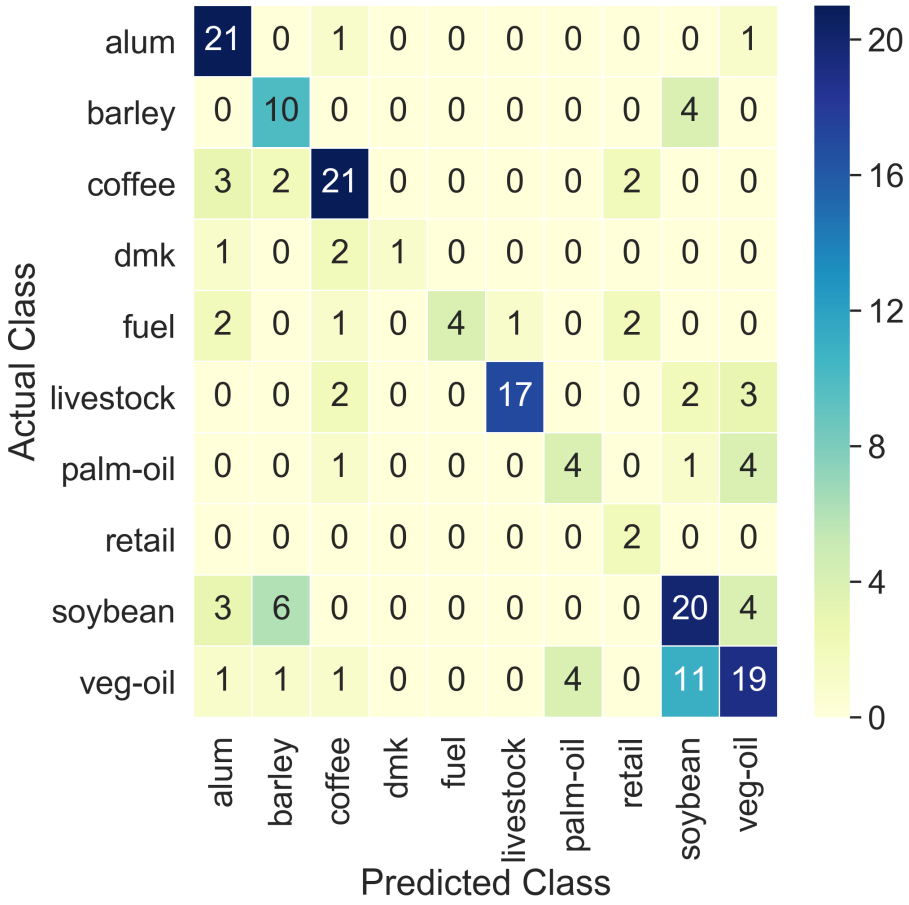

Figure 29: Confusion Matrix of Adopted Configuration 1

We can see that normalization of term document matrix does not have played any crucial role. More over be can say that algorithm have not learned significantly. But we can see that cross validation score have a slight increase which tell that there are some problem with data set which playing crucial role.

4.1.5 : for Multinomial Naive Bayes

Let's see what performance we are getting with default setup.

	max df	min df	ngram	idf	no. token	alpha
0	0.3	1	1 - 1	No	5694	1
1	0.3	1	1 - 1	Yes	5694	1

Table 17: Default Configuration

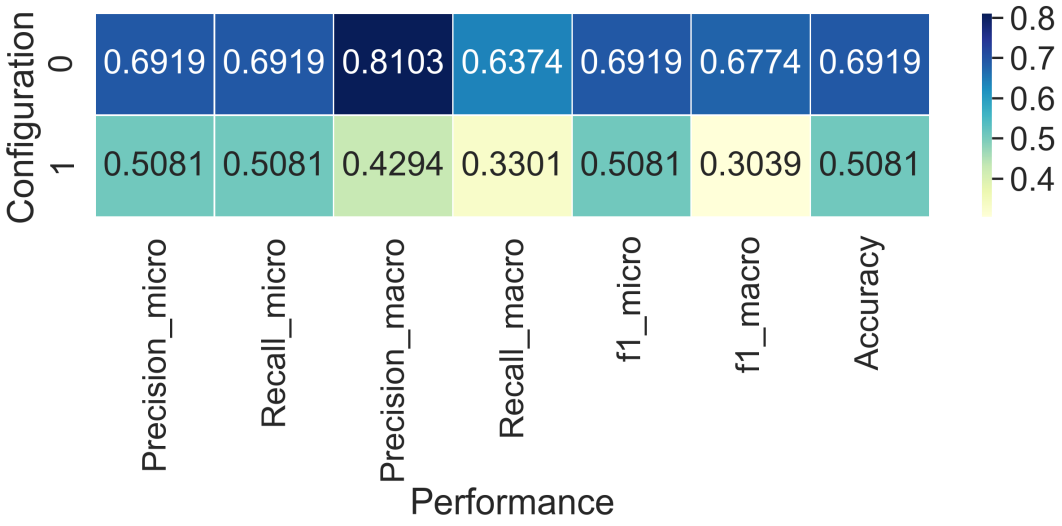

Figure 30: Performance of configuration

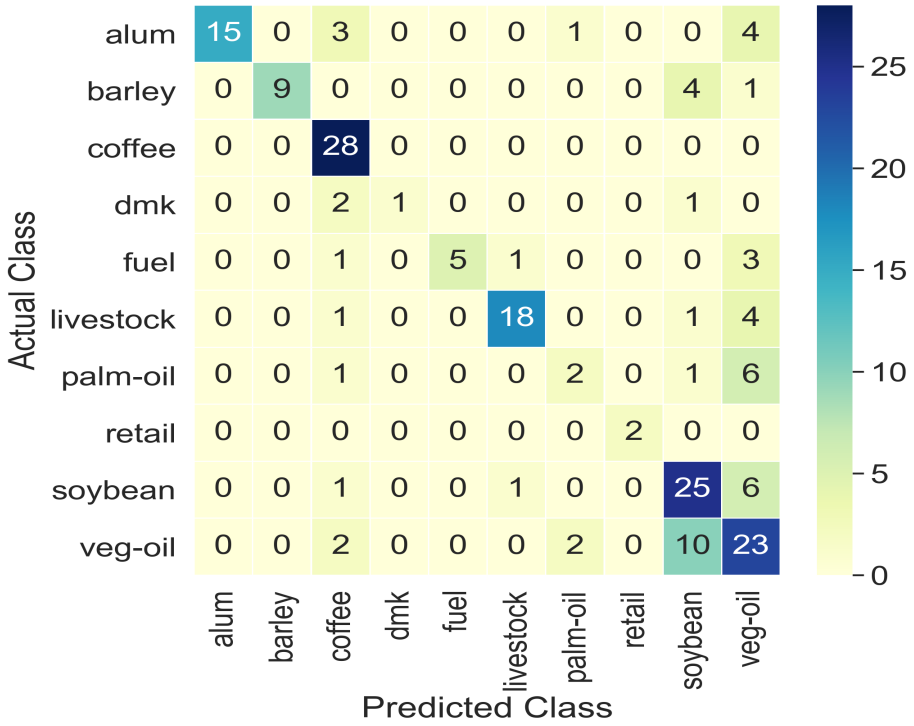

Figure 31: Confusion Matrix of configuration 0

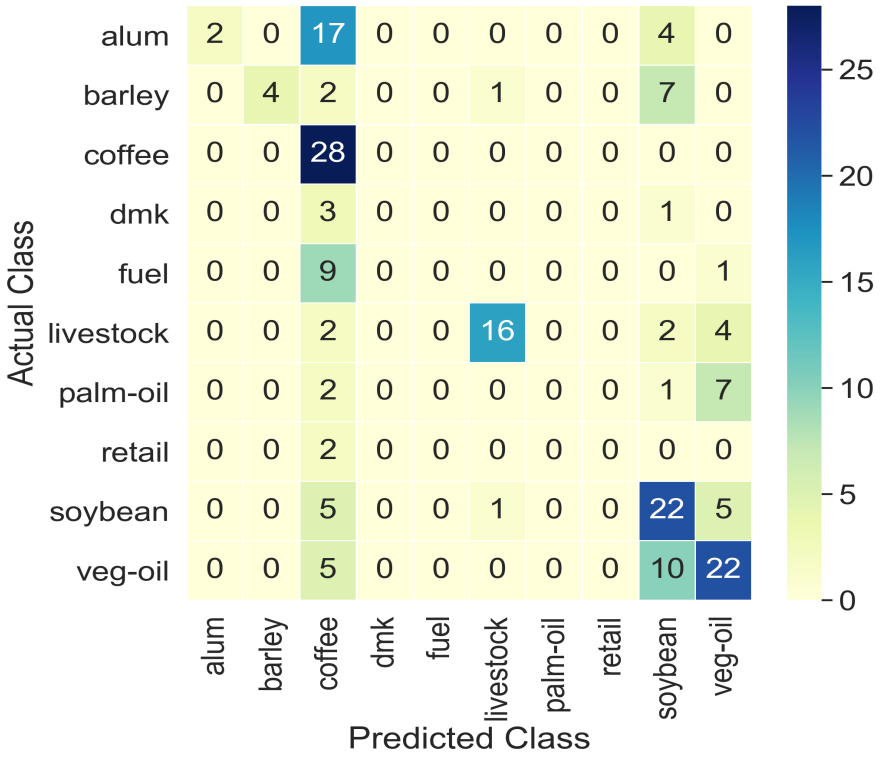

Figure 32: Confusion Matrix of configuration 0

We have got not good but a reasonable performance. Lets see what performance we are getting for this classifier on *RandomizedSearchCV*.

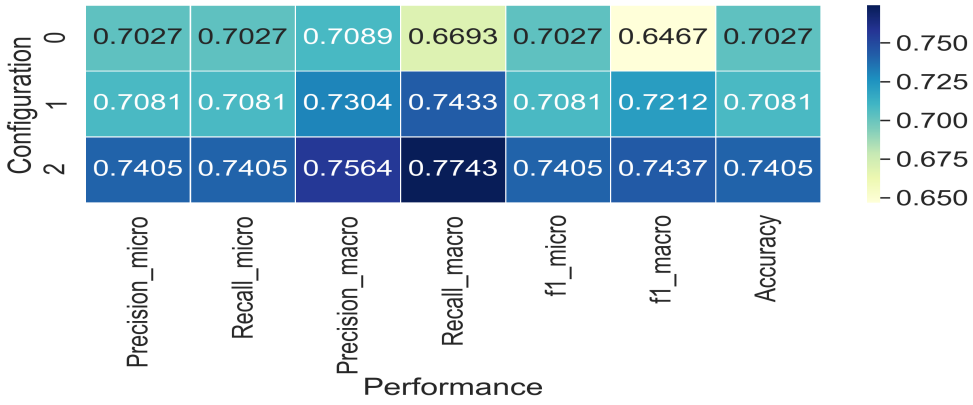

Figure 33: Performance of random search 0

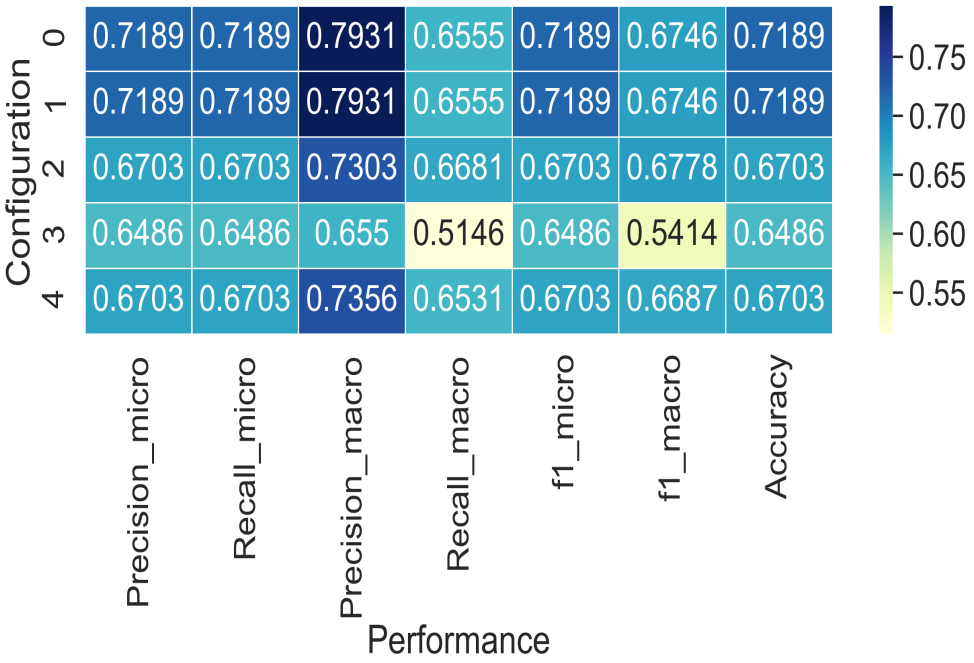

Figure 34: Performance of random search 1

Based on above performance adopted the setup is

	max df	min df	ngram	idf	no. token	alpha
0	0.4	0.01	1 - 2	No	2439	0.3
1	0.4	0.01	1 - 2	Yes	2439	0.3

Table 18: Adopted Configuration

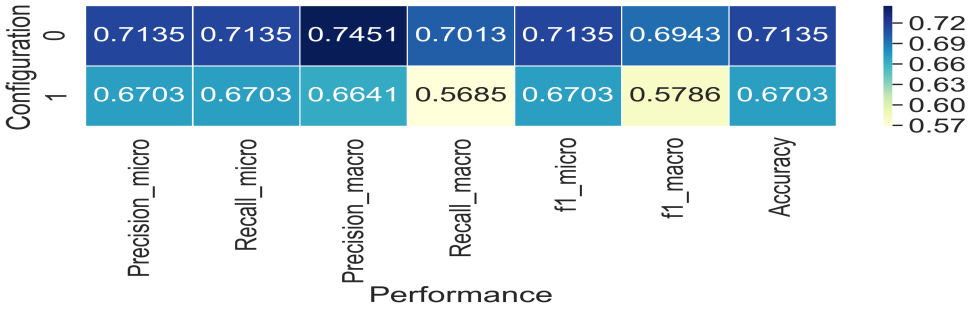

Figure 35: Performance of Adopted of configuration

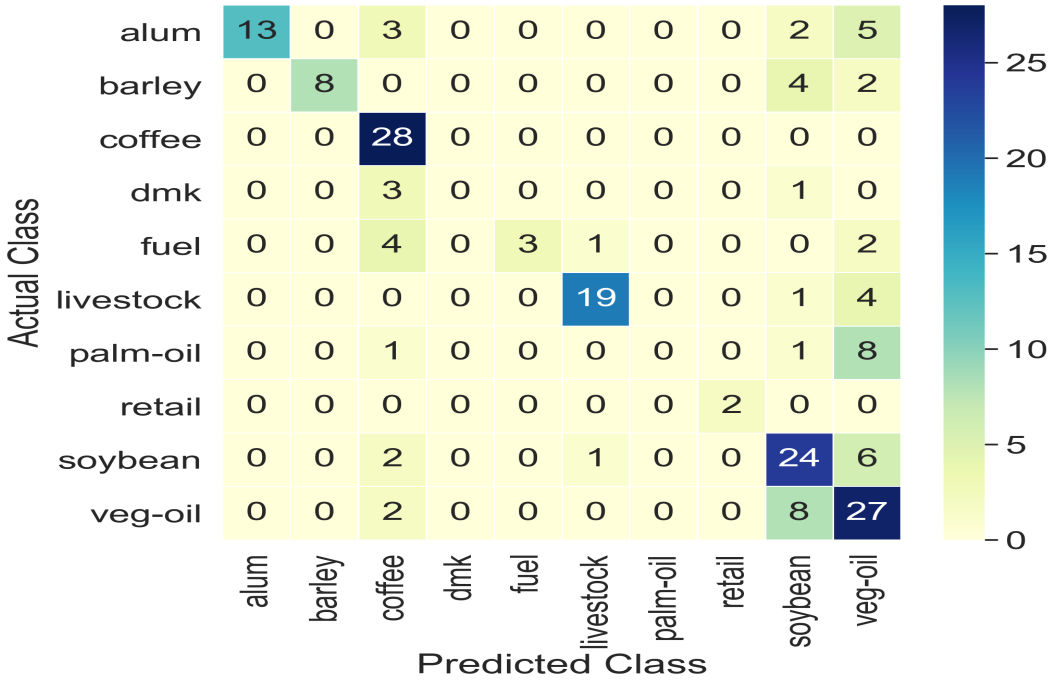

Figure 36: Confusion Matrix of adopted configuration 1

	1	2	3	4	5	6	7	8	9	10	Avg.	metric
0	0.740	0.792	0.792	0.725	0.843	0.795	0.693	0.766	0.739	0.739	0.762	f1 micro
1	0.773	0.814	0.799	0.779	0.855	0.823	0.711	0.809	0.760	0.795	0.792	f1 macro

Table 19: 10 fold Cross validation performance for Configuration 0

Here we can see that although test case performance of classifier low but cross validation performance is quite high. Also we can see that larger count of term does not have supported. And one opposite train can also countered that performance without $TFIDF$ normalization is quite high then with $TFIDF$ normalization. And also we can see that there is a gap between f1_micro and f1_macro, which tell that classes are overlapped.Here from figure 23, 26, 27 and 28 we can see that the 3,, 4 and 6 holds with equality asymptotically and but 1 and 5 have got distorted and 2 is a kind little bit deviated.

5 Discussions And Conclusion

5.1 Discussions

Here first I would like to present two example of document then we will use some conclusion from them.

- **D1_l1** He goes to school.
- **D1_l2** He goes to Market.
- **D2_l1** He goes to school,
- **D2_l2** Where he get involved with academic activity.

In the first document $D1\_l1$ the term 'goes' doesn't comes with any independent and direct information but it created dependency for occurrence of term 'school' and in the same way it has got dependency form occurrence of term 'market' in document $D1\_l2$. So occurrence of term 'goes' can be expressed in terms of occurrence of term term 'school' and 'market'. If these two document are in the class then it will create within the class dependency between the features and if they are in different class then it will create dependency with the class; And same will hold for the $D2\_l1$ and $D2\_l2$ because content of $D2\_l2$ are completely dependent on content of $D2\_l1$ and $D2\_l1$ doesn't comes with any purposeful extra information. Lets present some fact from given data set. In the table 20

	max df	min df	ngram	idf	no. token	max var	min var	phvrc	plvrc
0	0.35	0.005	1 - 2	No	5289	11.218	0	0.0009	0.990
1	0.35	0.005	1 - 2	Yes	5289	0.011	0	0.027	0.073
2	0.35	0.005	1 - 2	No	2439	11.218	0	0.0024	0.978
3	0.4	0.01	1 - 2	Yes	2439	0.013	0	0.080	0.253

Table 20: Variance analysis of terms

	max df	min df	ngram	idf	no. token	phrc
0	0.35	0.005	1 - 2	No	5289	0.744
1	0.35	0.005	1 - 2	Yes	5289	0.77
2	0.35	0.005	1 - 2	No	2439	0.508
3	0.4	0.01	1 - 2	Yes	2439	0.475

Table 21: Correlation analysis of terms

the term 'max var' and 'min var' meant for maximum and minimum variance [4] respectively and the term '*phvrc*'and the term '*plvrc*' meant for proportion of token have variance 'higher than 5 for case 0 and 2' and 'higher than 10^{-3} for case 1 and 3' and the term '*plvrc*' meant for proportion of token have variance 'less than 1 for case 0 and 2', 'less than 10^{-5} for case 1' and 'less than 10^{-4} for case 3'.
And in table 21 the term $'phrc'$ meant for the proportion of token have correlation coefficient [4] higher than 0.65 but not 1.
Now from the table we can see that 20 we can see that there are high proportion of tokens which have lower variability we have not used $TFIDF$ normalization and when we have used $TFIDF$ normalization large portion of data densely packed central variation(mean in between lower bound and upper bound that

we have choosen above).
And from table 21 we can see that there are large proportion of tokens which have got at least once higher correlation this proportion have reduced while using $TFIDF$ normalization but not more.
Our data is full of documents whose content is sufficient enough to create dependency like dependency created by $D1\_l1$, $D1\_l2$, $D2\_l1$, and $D2\_l2$ with in the class.
As we have equations above from which equation 2 have hold through out which tells that precision micro is low as classifiers doesn't doing well with in the class by making more false positive instance as by this f1 micro becomes low and so does accuracy. And 1 and 5 have slight different with one good commonality i.e. precision macro is upper bound in both cases which tell that although within the class proportion of false positive is high but through out the document it is less this implies that there are documents and classes which gets completely learned.
And opposing behaviour of equation 3, 4 and 6 tells classes are highly unbalanced and some classes are highly similar content or kind of overlapped, so when the classifier which is insensitive with overlapped information this comes with negative information that more false negative are made.

5.2 Conclusion

SVM and *Logistic Regression* have worked closely and KNN is in run-up. Naive Bayes classifiers have not worked that much. As assumption of Naive Bayes that 'features are not dependent' is not satisfied.
Within the sentence Only tracking reasonable number of words can help us not many number of words same may hold for sentence tracking.
Generating large number of token is not important. Important is generating the set of tokens which captures almost complete information from data and are highly independent and having good variability. As we have seen that our models have suffered due to insufficiency of such feature(token) set so making a model to get such feature set having high information accuracy and works efficiently could be a good future work from here.

The data have been provided by Instructor :
$https://drive.google.com/open?id = 1jCFvyhku0NjeKJAqEvsrzEO7K17ddoiE$

Technology Used ::

Programming Language : PYTHON3
Environment/IDE : JUPYTER NOTEBOOK
Package/Library : PANDAS(pandas), SCIKIT-LEANRN(sklearn)

Thanks!

References

[1] Tom M. Mitchell, *Machine Learning*, McGraw-Hill Science/Engineering/Math; March 1, 1997

[2] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar *Foundations of Machine Learning* , The MIT Press Cambridge, Massachusetts London, England; 2012

[3] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*, Cambridge University Press, New York, 2008

[4] RICHARD A. JOHNSON, DEAN W. WICHERN, *Applied Multivariate Statistical Analysis*, PEARSON Prentice Hall; 2007

[5] Python3, *PYTHON*, $https://www.python.org$;

[6] Scikit-Learn, *SKLEARN*, $https://scikit-learn.org$;

[7] Pandas, *PANDAS*, $https://pandas.pydata.org$;