Classify News Articles

Overview

Statistical classification is used to identify a subset of categories based on a particular data point. The data points can include text, images, and videos. A model is trained on a known dataset then used to predict the category of a separate dataset.

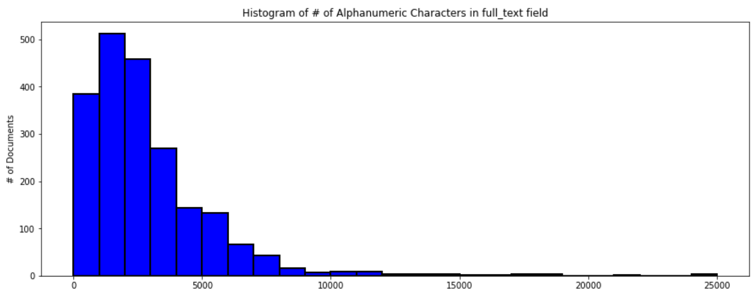
Getting familiar with the dataset

In this first section we are exploring the dataset.

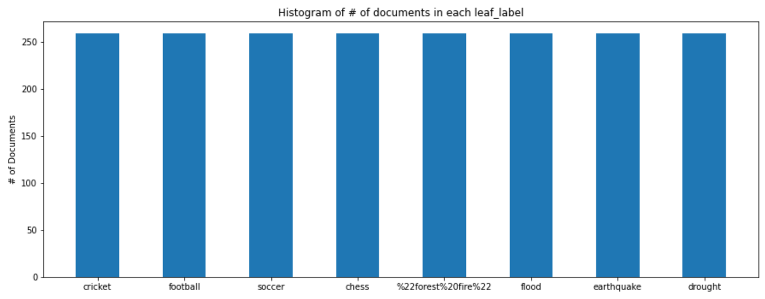
The main columns we are concerned about in the dataset are:

- text_field. The main text of the article
- leaf_label. The secondary category of each article
- root_label. The primary category of each article

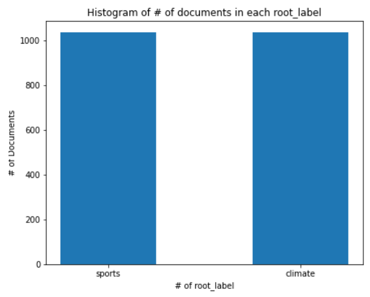
The below histogram shows how many documents have a certain text_field length. This length is based on the number of alphanumeric characters. As you can see the histogram shows that most articles contain between 0 and 3000 alphanumeric characters.



The below histogram shows how many documents are assigned each leaf_label. As you can see the histogram shows that every leaf_label contains the same number of documents.



The below histogram shows how many documents are assigned to each root_label. As you can see the histogram shows that every root_label contains the same number of documents.



Feature Extraction

When completing feature extraction of a text field there are different methods of looking at the individual words within the text:

- Extracting every word in the text.
- Eliminating "stop" words
- Stemming words
- Lemmatizing words

The main problem with using every word is that you will have words contained in every document many times (for example the word "the.") This will cause the corpus to be too big and slow. However, removing them is a new complication.

Before you remove the stop words you want to either stem or lemmatize the words, therefore reducing the size of the corpus and increasing its performance.

Stemming

Stemming is the process of removing the last letters in attempt to reduce the word to its base. For example, the word compared would be stemmed to compare. Below are the pros and cons of stemming.

Stemming	
Pros: Allows for more word matching Reduces model overfitting	Cons: Can change meaning. For example, Organize will stem to Organ

Lemmatizing

Another method of reducing the corpus is Lemmatizing. Lemmatization resembles stemming. They both reduce the words to their bases. The main difference is that lemmatization also considers the part of speech (pos) of each word (i.e., Noun, Verb, Adverb, etc.) Below are the pros and cons of lemmatizing.

Lemmatizing	
Pros: Uses the pos of the word to define the correct base word The meaning of the word is not changed when a base is determined	Cons: The pos needs to be determined Linguistic fundamentals are required resulting in more complicated code

Stemming vs. Lemmatizing

Since lemmatizing attempts to map multiple words to a single root word correctly and efficiently, the dictionary size may be smaller as compared to stemming, which bluntly removes the suffix and could change the meaning of many words.

Order of Operations

Removing different words like stop words, punctuation, and numbers, should be conducted after lemmatizing as their removal could change the sentence structure causing the pos or the words to be inaccurate. If the pos is inaccurate the base word could also be inaccurate causing change of meaning.

Vocabulary

Once the words have been extracted a dictionary (corpus) is built. There are 2 main methods for building this dictionary:

- Bag of Words (BOW)
- Term Frequency-Inverse Document Frequency (TF-IDF)

Bag of Words (BOW)

This approach determines the word frequency in each document. No other words are eliminated using this technique.

Term Frequency-Inverse Document Frequency (TF-IDF)

This approach also determines the word frequency in each document. The main difference is the ability to specify and minimum and maximum document frequency.

Minimum document frequency (min_df) removes any words that are not contained in the specified minimum number of documents. The higher the value of min_df, the smaller the size of the dictionary.

Maximum document frequency (max_df) removes any words that are contained in more documents than the specified maximum percentage. When a max_df value of 0.8 is applied all words that are contained in over 80% of the documents are removed. The lower the values of max_df, the smaller the size of the dictionary

Dimensionality Reduction

Even using TD-IDF the model could still perform poorly due to the high dimensionality and sparseness of the data. In this project 2 different dimensionality methods will be discussed:

- Latent Semantic Indexing (LSI)
- Non-negative Matrix Factorization (NMF)

Latent Semantic Indexing (LSI)

This process looks at the number of documents associated with each word in the corpus and determines the words with the highest values. It then keeps only the number specified (top k)

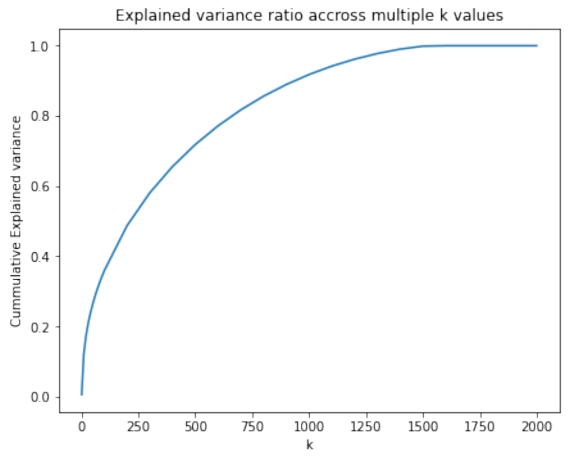
One way to determine the optimal value for k is to calculate the explained variance ratio (the difference between the prediction and actual) for different values of k. The optimal value is 1 but a balance needs to be determined as the higher the explained variance the larger the corpus and the slower the model. 80% is a good ratio to look for in your model. Based on the below graph a k value of around 700 would be optimal.

Non-negative Matrix Factorization (NMF)

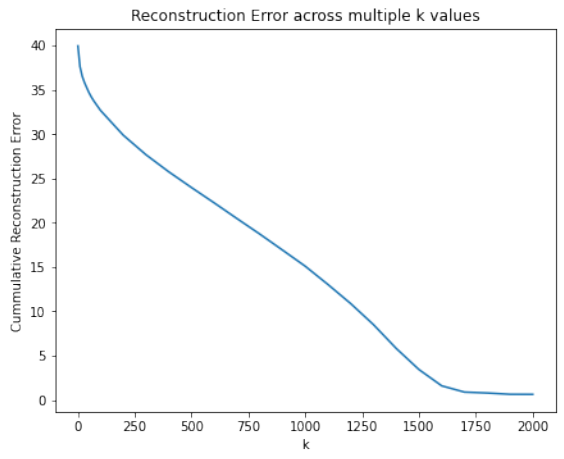
This process looks at things a little differently. Instead of looking at each word separately it looks at combinations of words to try to derive topics. The instance count of each topic is then determined and only the words associated with the top specified number are kept.

One way to determine the optimal value for k is to calculate the reconstruction error (the difference between the prediction and actual) for different values of k. The optimal value is 0 but a balance needs to be determined as the lower the reconstruction error the larger the corpus and the slower the model. Based on the below graph a k value of around 1250 would be optimal.

Latent Semantic Indexing (LSI)



Non-negative Matrix Factorization (NMF)



Data Split Processes

Once the dimensions have been reduced the next step is the train the model. Before the training can begin the dataset needs to be split into different sections. There are different methods for doing this including:

- Train-Test
- Cross Validation (K-fold)

Train-Test

Divide the dataset into two separate objects (train dataset and test dataset) using an 80/20 split. The train dataset is used for training and the validation is completed using the test dataset.

Cross Validation (K-fold)

Divide the dataset into k number of evenly sized datasets. The process then iterates through k times using a different sub dataset for testing and the rest for training. The scores are then averaged over all the iterations.

Here is an example of how a 5-fold cross validation would work.

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Classification Algorithms

In this project I will discuss the following:

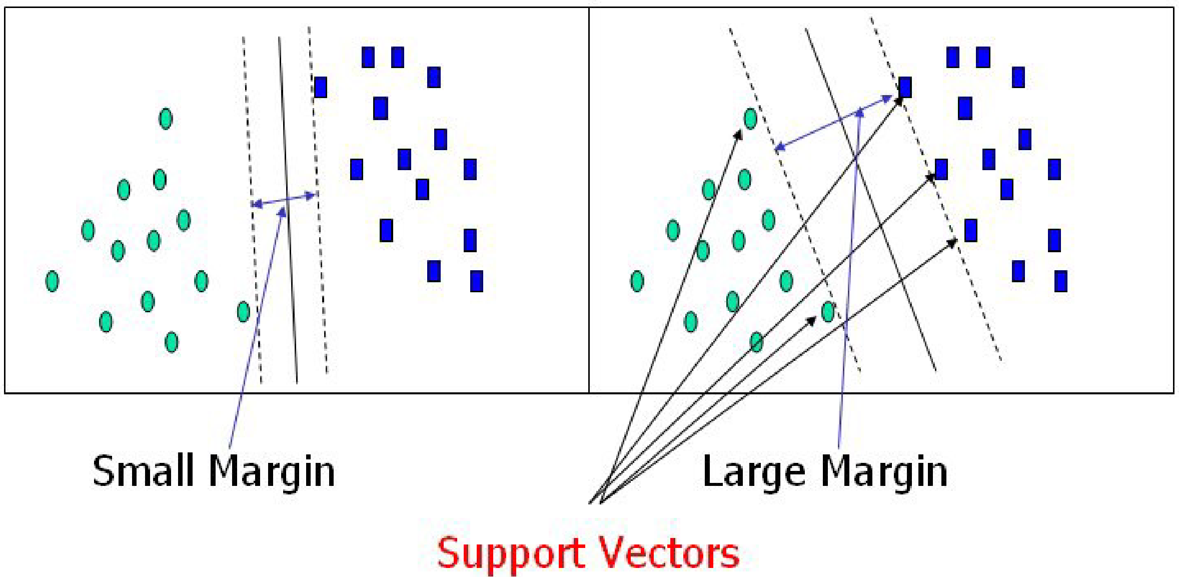
- Support Vector Machine (SVM)
- Logistical Regression
- Naïve Bayes Model

Support Vector Machine (SVM)

The objective of the SVM model is to determine a linear (straight line) division between categories. There are distinct types of support vectors including small (soft) margin and large (hard) margin. Small margin tries to minimize the distance between the support vector and the data elements whereas large margin tries to maximize that distance.

SVM tries to find the maximum margin that separates the classes.

Here is a representation of the large and small margin support vectors. (Image provided from the following link)



When completing multiclass classifications there are 2 options: One vs One and One vs The Rest.

One vs One (based on 4 classes)

- Binary classification Problem 1: class1 vs class2
- Binary classification Problem 1: class1 vs class3
- Binary classification Problem 1: class1 vs class4
- Binary classification Problem 1: class2 vs class3
- Binary classification Problem 1: class2 vs class4
- Binary classification Problem 1: class3 vs class4

One vs The Rest (based on 4 classes)

- Binary classification Problem 1: class1 vs (class2 & class3 & class4)
- Binary classification Problem 2: class2 vs (class1 & class3 & class4)
- Binary classification Problem 3: class3 vs (class1 & class2 & class4)
- Binary classification Problem 3: class4 vs (class1 & class2 & class3)

One issue with the One vs The Rest model is the imbalance that occurs when comparing one class to many.

To deal with the class imbalance problem the SMOTEENN process is used. This process combines SMOTE (Synthetic Minority Oversampling Technique) with ENN (Edited Nearest Neighbors.) These processes are used to add samples (oversample) to the minority class while removing samples (undersample) from the majority class thus balancing the dataset.

Logistical Regression

The objective of the Logistical Regression model is to determine a linear (straight line) division between categories. Logistic Regression does this by minimizing errors.

There are 3 different logistic regression models we will use in this project:

- Without Regularization
- With L1 Regularization
- With L2 Regularization

Naïve Bayes Model

The Naïve Bayes classifier tries to separate data into different classes according to the Bayes' Theorem, which takes the test results correcting for "skew" due to false positives. It's naïve because it is assuming that all the words in the corpus are independent of each other and equally likely to appear.

There are several types of Naïve Bayes models but in this report, we will just look at the Gaussian, which assumes that the distribution of words in the corpus follows a gaussian (normal) distribution.

Performance Measures

Classification quality can be evaluated using different methods:

- Classification Measures (accuracy, precision, recall, and F-score)
- Confusion matrix.
- ROC Curve

Classification measures

Accuracy

Accuracy shows how close the measurements are to the actual values

Precision

Precision shows the degree to which repeated measurements, under the same conditions, show the same result. This is often measured using the standard deviation.

Recall

Recall, or True Positive Rate, shows the percentage of measurements that were correctly predicted by the classifier.

F-Score

F-score combines the precision and recall measures into a single value. The value is based on the harmonic mean of the precision and recall values. The harmonic mean is close to the minimum of the 2 values. A high F-score ensures that both precision and recall are reasonably high.

Confusion Matrix

The confusion matrix shows how well the model works by showing how the followings situations play out:

- True Positive (TP)
- False Positive (FP)
- True Negative (TN)
- False Negative (FN)

The more classifications there are the more complex the matrix is.

2x2 Confusion Matrix

Below is a representation of a confusion matrix for a two-category classification. As you can see it's fairly simple to determine a good model. The more the number of True Positives and True Negatives the more accurate the model is.

		Prediction	
		1	0
Actual	1	True Positive	False Negative
	0	False Positive	True Negative

True Positive and Negative are the number of times where the predicted value matches the actual.
False Positive and Negative are the number of times where the predicted value is the opposite of the actual

3x3 Confusion Matrix

Below is a representation of a confusion matrix for a three-category classification. As you can see it's still fairly simple to determine a good model though it is getting more complicated. The more categories you have the more difficult it gets.

			Prediction	
		0	1	2
	0	True 0	False 1	False 2
Actual	1	False 0	True 1	False 2
	2	False 0	False 1	True 2

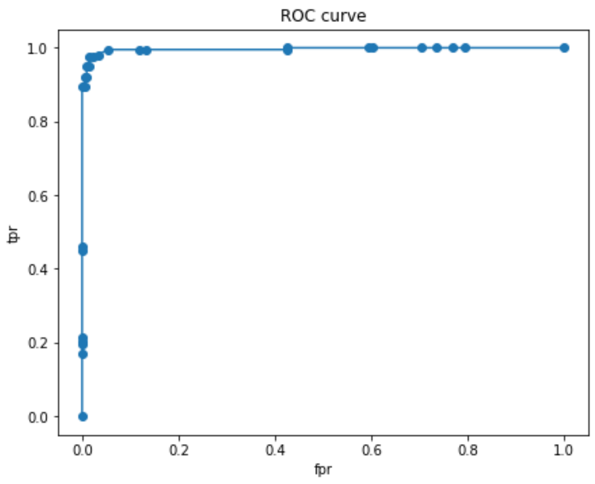
The True Positive, True Negative, False Positive, and False Negative are now based on the Actual value.
For Actual Value 0:

- True Positive includes the values in the True 0 cell
- False Negative includes the values in the False 1 and False 2 cells in the 0 row
- False Positive includes the values in the False 0 cells in the 0 column

ROC Curve

The ROC Curve is a visual representation of the Confusion Matrix. The ROC Curve plots the True Positive Rate versus the False Positive Rate. The True Positive Rate is the proportion of observations that were correctly predicted to be positive divided by all positive observations (TP/(TP + FN)). The False Positive Rate is the proportion of observations that are incorrectly predicted to be positive divided by all negative observations (FP/(TN + FP))

Here is an example of an ROC curve.



GridSearch

As we have seen there are a of number combinations available of the following:

- Feature Extraction
- Dimensionality Reduction
- Classifiers

Also, each section has different parameters that need to be tuned for optimal training.

Due to this complexity, it can be very time consuming to write the code to run and validate all the different combinations and parameters.

One method to manage this is a GridSearch. You can build a process that uses parameters to define the what feature extractors, dimensionality reduction methods, classifiers, and parameters will be used. The process will run every combination and return the best parameters for each combination. This can be a very time-consuming process to run but will simplify the process of determining which combinations should be used.

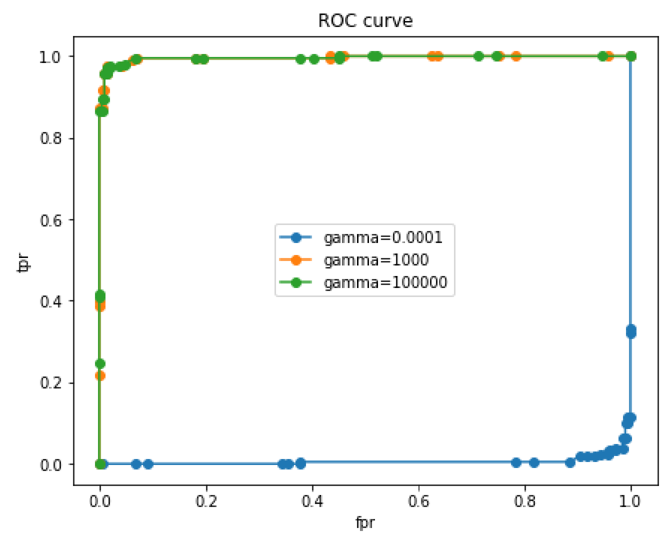
Data Classifications

For this section we tried to classify the articles to the two different root_tabels (climate & sports)

Binary Classification

SVM Model

The below are the results of a SMV model using different gamma values (0.0001, 1000, and 100000). The gamma value of 0.0001 is considered the small margin where the gamma value of 1000 is the large margin



γ = 0.0001

```
confusion matrix:
 [[209   0]
 [206   0]]
precision:  0.0
recall:  0.0
f1:  0.0
accuracy:  0.5036144578313253
```

γ = 1000

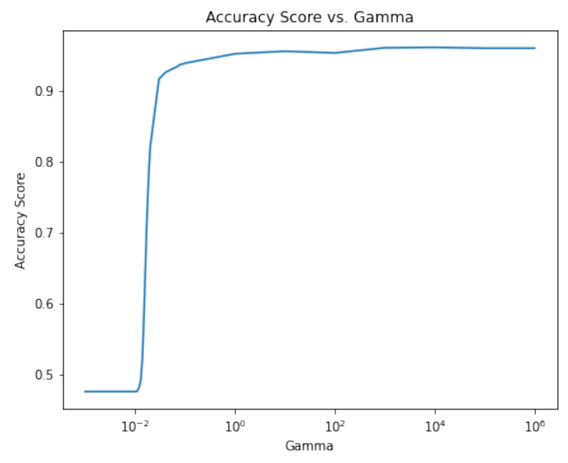
```
confusion matrix:
 [[206   3]
 [  8 198]]
precision:  0.9850746268656716
recall:  0.9611650485436893
f1:  0.972972972972973
accuracy:  0.9734939759036144
```

γ = 100000

```
confusion matrix:
 [[207   2]
 [  9 197]]
precision:  0.9899497487437185
recall:  0.9563106796116505
f1:  0.9728395061728394
accuracy:  0.9734939759036144
```

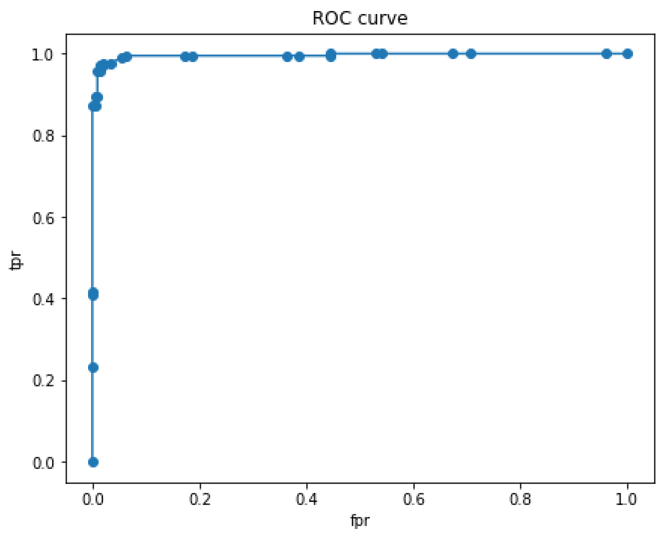
As you can see the large margin support vector is more accurate than the small margin support vector. There are 2 ways to show this. First the confusion matrix from the small margin support vector shows no negative results which we know is not correct, also precision, recall, and f1 are all 0. One the other hand, the confusion matrix for the large margin support vector shows most of the items are either True Positive or True Negative, and the precision, recall, and F1 are all remarkably close to one.

One way to determine the best value for gamma is to run the model for many different values and find out which has the best accuracy score. Below is a graph of such a test along with some of the accuracy scores.



0.4755705601863648, gamma:0.0010
0.8208022422014342, gamma:0.0200
0.9173424817093145, gamma:0.0300
0.9300131037746151, gamma:0.0500
0.9384632184326429, gamma:0.0900
0.9396698576784479, gamma:0.1000
0.9529356095075165, gamma:1.0000
0.9565555272449314, gamma:10.0000
0.9541404287846250, gamma:100.0000
0.9613820842281513, gamma:1000.0000
0.9619917737414918, gamma:10000.0000
0.9607851344956867, gamma:100000.0000
0.9607851344956867, gamma:1000000.0000

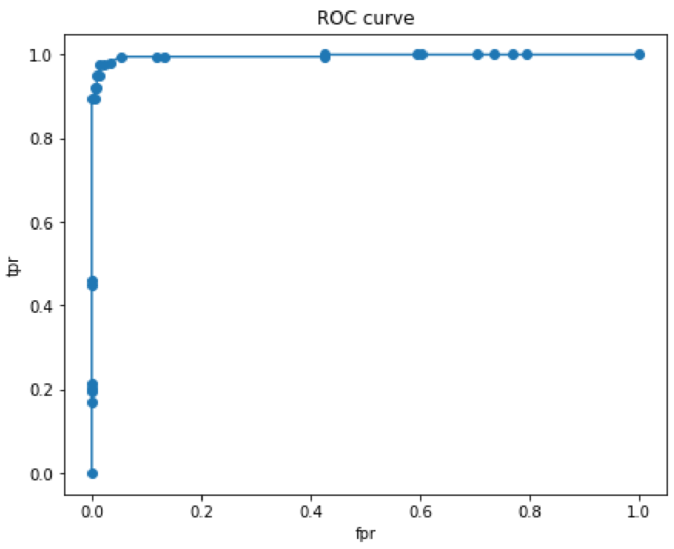
We can see from the graph that the best value for gamma is 10000. Below is the ROC curve, confusion matrix and classification measures when gamma is 10000.



```
confusion matrix:
 [[206   3]
 [  9 197]]
precision:  0.985
recall:  0.9563106796116505
f1:  0.9704433497536945
accuracy:  0.9710843373493976
```

Logistic Regression Model

Here is the ROC curve, confusion matrix, and classification measures for a logistical model without regularization.



```
confusion matrix:
 [[206   3]
 [  7 199]]
precision:  0.9851485148514851
recall:  0.9660194174757282
f1:  0.9754901960784313
accuracy:  0.9759036144578314
```

For L1 and L2 regularization a similar process used for the SVM model is also used here. Below are the gamma values for different gamma values. As you can see from the data the best L1 gamma is 0.1000 and the best L2 gamma is 0.0100.

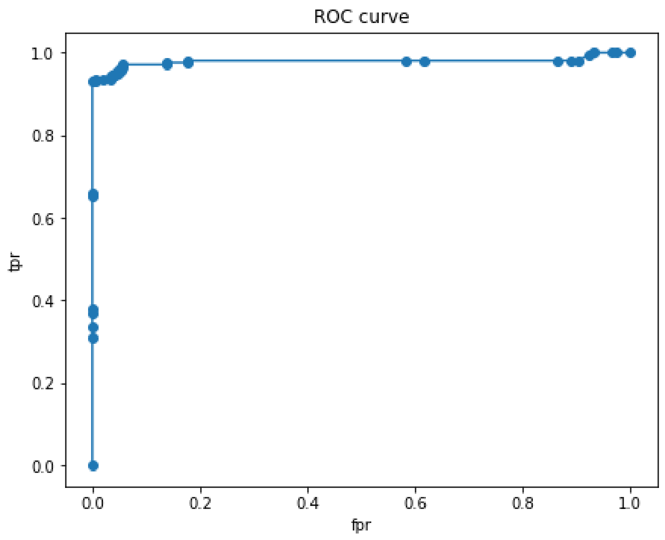
L1	L2
0.9541313289411422, reg: 0.0001	0.9553379681869473, reg: 0.0001
0.9547337385796965, reg: 0.0010	0.9571506570086994, reg: 0.0010
0.9571470170713063, reg: 0.0100	0.9589597058930586, reg: 0.0100
0.9577494267098604, reg: 0.1000	0.9571397371965201, reg: 0.1000
0.9517107705747462, reg: 1.0000	0.9577403268663780, reg: 1.0000
0.9070469187929968, reg: 10.0000	0.9462727041094894, reg: 10.0000
0.4991173151821789, reg: 100.0000	0.9239398682342663, reg: 100.0000
0.4991173151821789, reg: 1000.0000	0.8617533578422452, reg: 1000.0000
0.4991173151821789 ,reg: 10000.0003	0.8309613074655117 ,reg: 10000.0003

Below are the confusion matrix and classification measures for Liner Regression with no regularization, L1 regularization with gamma of 0.1000, and L2 regularization with gamma of 0.0100

No Regulization	L1	L2
confusion matrix:	confusion matrix:	confusion matrix:
[[206 3]	[[207 2]	[[192 17]
[7 199]]	[15 191]]	[12 194]]
precision: 0.9851485148514851	precision: 0.9896373056994818	precision: 0.919431279620853
recall: 0.9660194174757282	recall: 0.9271844660194175	recall: 0.941747572815534
f1: 0.9754901960784313	f1: 0.9573934837092731	f1: 0.9304556354916067
accuracy: 0.9759036144578314	accuracy: 0.9590361445783132	accuracy: 0.9301204819277108

Naïve Bayes Model

Below are the confusion matrix and classification measures for Gaussian Naïve Bayes model. As you can see it’s pretty accurate.



```
confusion matrix:
 [[197  12]
 [  6 200]]
precision:  0.9433962264150944
recall:  0.970873786407767
f1:  0.9569377990430622
accuracy:  0.9566265060240964
```

GridSearch

For this project I used the following options

Module	Options
Loading Data	Cleaned
	Not-cleaned
Feature Extraction	TF-IDF (min_df = 3,5)
Compression Module	None
	Stemming
	Lemmatization
Dimensionality Reduction	LSI (k=[5,50,200])
	NMF (k=[5,50,200])
Classifiers	SVM (use best gamma found above)
	L1 Logistic Regression (use best gamma found above)
	L2 Logistic Regression (use best gamma found above)
	GausieanNB

This process resulted in the 5 best combinations as follows:

Rating	clean	lem_stem	dim_red	model	CV Score	n	min_df	test_perf
1	clean	stem	NMF	gaussian	0.972	200	5	0.975903614
2	notclean	stem	NMF	gaussian	0.972	200	5	0.975903614
3	clean	lem	NMF	gaussian	0.970	200	5	0.975903614
4	notclean	lem	NMF	gaussian	0.970	200	5	0.975903614
5	clean	nolemstem	NMF	gaussian	0.970	200	5	0.973493976

Multiclass Classification

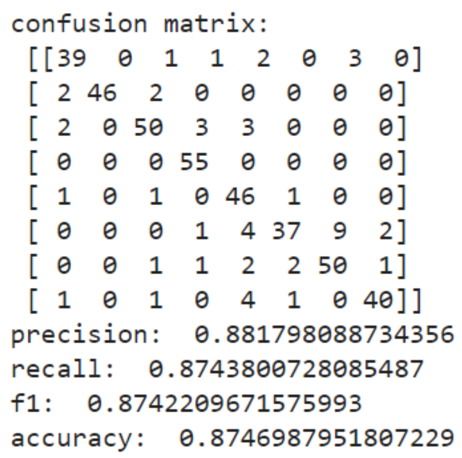
For this section of the report, we looked at more than 2 classifiers. We used the leaf_label which contains the following values: Chess, Cricket, Soccer, Football, Forest Fire, Flood, Earthquake, Drought

I completed three classification models:

- GaussianNB
- SVM – One vs One
- SMV – One vs The Rest (with balancing and without)

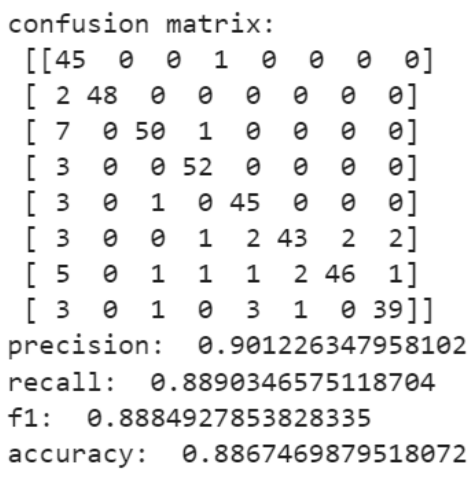
Here are the results

GaussianNB



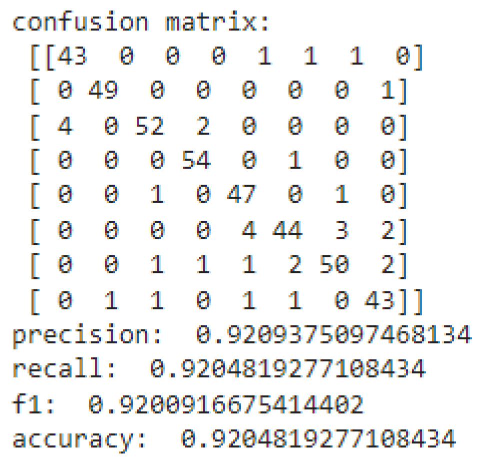
```
confusion matrix:
 [[39  0  1  1  2  0  3  0]
 [ 2 46  2  0  0  0  0  0]
 [ 2  0 50  3  3  0  0  0]
 [ 0  0  0 55  0  0  0  0]
 [ 1  0  1  0 46  1  0  0]
 [ 0  0  0  1  4 37  9  2]
 [ 0  0  1  1  2  2 50  1]
 [ 1  0  1  0  4  1  0 40]]
precision:  0.881798088734356
recall:  0.8743800728085487
f1:  0.8742209671575993
accuracy:  0.8746987951807229
```

SVM – One vs One



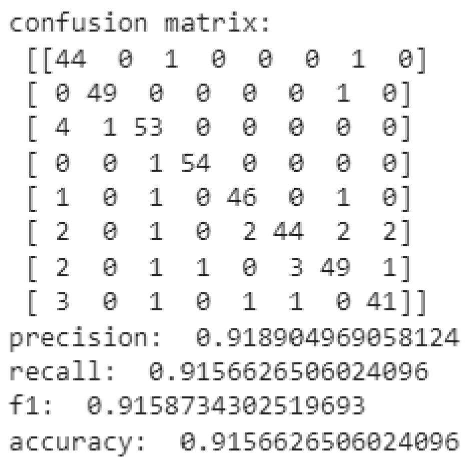
```
confusion matrix:
 [[45  0  0  1  0  0  0  0]
 [ 2 48  0  0  0  0  0  0]
 [ 7  0 50  1  0  0  0  0]
 [ 3  0  0 52  0  0  0  0]
 [ 3  0  1  0 45  0  0  0]
 [ 3  0  0  1  2 43  2  2]
 [ 5  0  1  1  1  2 46  1]
 [ 3  0  1  0  3  1  0 39]]
precision:  0.901226347958102
recall:  0.8890346575118704
f1:  0.8884927853828335
accuracy:  0.8867469879518072
```

SVM - One VS Rest (no balancing)



```
confusion matrix:
 [[43  0  0  0  1  1  1  0]
 [ 0 49  0  0  0  0  0  1]
 [ 4  0 52  2  0  0  0  0]
 [ 0  0  0 54  0  1  0  0]
 [ 0  0  1  0 47  0  1  0]
 [ 0  0  0  0  4 44  3  2]
 [ 0  0  1  1  1  2 50  2]
 [ 0  1  1  0  1  1  0 43]]
precision:  0.9209375097468134
recall:  0.9204819277108434
f1:  0.9200916675414402
accuracy:  0.9204819277108434
```

SMV - One VS Rest (with balancing)



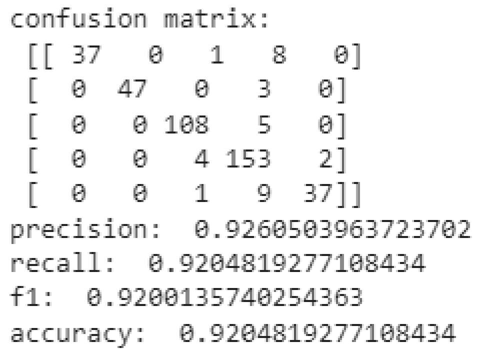
```
confusion matrix:
 [[44  0  1  0  0  0  1  0]
 [ 0 49  0  0  0  0  1  0]
 [ 4  1 53  0  0  0  0  0]
 [ 0  0  1 54  0  0  0  0]
 [ 1  0  1  0 46  0  1  0]
 [ 2  0  1  0  2 44  2  2]
 [ 2  0  1  1  0  3 49  1]
 [ 3  0  1  0  1  1  0 41]]
precision:  0.918904969058124
recall:  0.9156626506024096
f1:  0.9158734302519693
accuracy:  0.9156626506024096
```

Based on the results you can see that the element in the major diagonal that is more visible is “cricket” which means that it’s more easily distinguishable using the text data.
The show that 2 groupings are found:

i. soccer, football
ii. forest fire, earthquake, flood

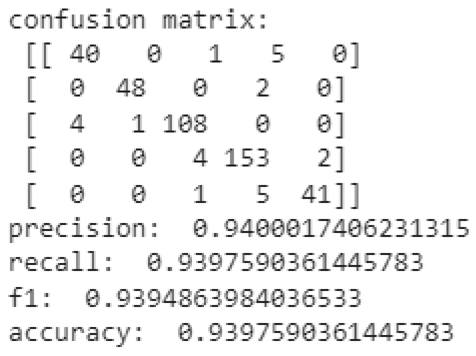
After modifying the classes base on the above groupings, the accuracy of both the One VS One and One VS Rest improved.

One VS One



```
confusion matrix:
 [[ 37   0   1   8   0]
 [  0  47   0   3   0]
 [  0   0 108   5   0]
 [  0   0   4 153   2]
 [  0   0   1   9  37]]
precision:  0.9260503963723702
recall:  0.9204819277108434
f1:  0.9200135740254363
accuracy:  0.9204819277108434
```

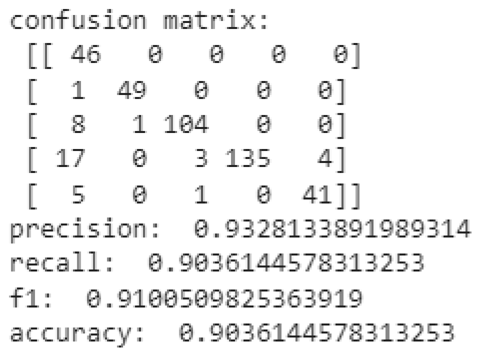
One VS Rest



```
confusion matrix:
 [[ 40   0   1   5   0]
 [  0  48   0   2   0]
 [  4   1 108   0   0]
 [  0   0   4 153   2]
 [  0   0   1   5  41]]
precision:  0.9400017406231315
recall:  0.9397590361445783
f1:  0.9394863984036533
accuracy:  0.9397590361445783
```

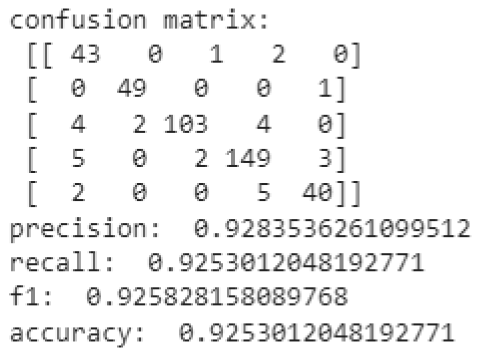
Though accuracy decreased a bit when the data was balanced

One VS One



```
confusion matrix:
 [[ 46   0   0   0   0]
 [  1  49   0   0   0]
 [  8   1 104   0   0]
 [ 17   0   3 135   4]
 [  5   0   1   0  41]]
precision:  0.9328133891989314
recall:  0.9036144578313253
f1:  0.9100509825363919
accuracy:  0.9036144578313253
```

One VS Rest



```
confusion matrix:
 [[ 43   0   1   2   0]
 [  0  49   0   0   1]
 [  4   2 103   4   0]
 [  5   0   2 149   3]
 [  2   0   0   5  40]]
precision:  0.9283536261099512
recall:  0.9253012048192771
f1:  0.925828158089768
accuracy:  0.9253012048192771
```