Large-Scale Data Mining: Models and Algorithms
ECE 219 Winter 2023
Prof. Vwani Roychowdhury
UCLA, Department of ECE

Project 1: End-to-End Pipeline to Classify News Articles

Due Jan 27, 2023 by 11:59 PM

Yaman Yucel - 605704529
Ozgur Bora Gevrek - 505846360

Introduction:

First project of the ECE 219 course consists of building an end-to-end pipeline structure for binary and multiclass classification on documents. Following steps are implemented to classify samples of news articles.

1) Feature Extraction: Using TF-IDF representations, features are extracted from raw text data.
2) Dimensionality Reduction: In order to reduce computation burden and improve performance dimensionality reductions such as Principal Component Analysis(PCA) and non-Negative Matrix Factorization are used on extracted features.
3) Application of Simple Classification Models: After dimensionality reduction, we have applied simple classifier algorithms to classify documents' topics. These algorithms are Support Vector Machines, Logistic/Linear Classification/ Naive Bayes.
4) Evaluation of Pipeline: In order to find the best performing pipeline, Grid-Search and Cross-Validation are used.
5) Replace corpus-level features with pretrained features: Lastly, rather than using features extracted from TF-IDF, GloVE embeddings are used to generate features.

Getting Familiar with the dataset:

Firstly, "Project1-Classification.csv" file is loaded using the pd.read_csv() function.

Question 1:

- Dataset consists of **3150 samples(rows)** and **8 features(columns)**.

- Histograms: a) The total number of alpha-numeric characters per data point in the feature full_text plotted as a histogram. Bins = 100 is used.

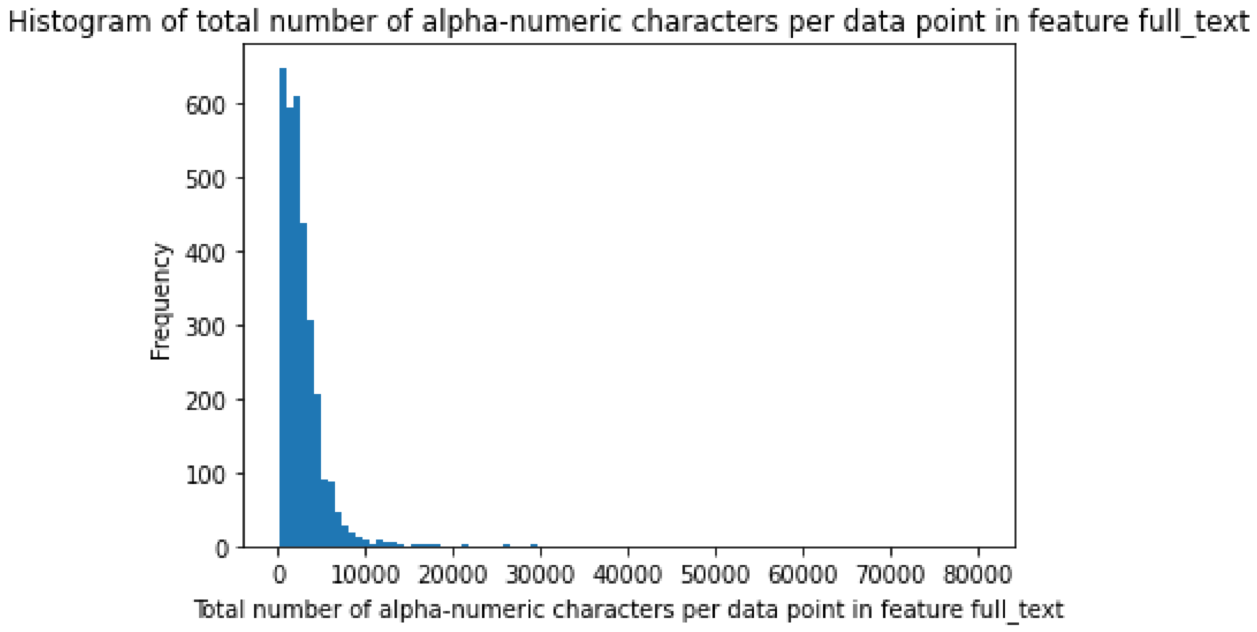


Figure 1: Histogram of alpha-numeric characters per data point in the feature full_text

Interpretation of Figure 1: The documents are mainly distributed between 0 to 10000 characters, however, there are some documents consisting 20000 to 80000 alpha-numeric characters. The document that has the most alpha-numeric characters consists of 80326 alpha-numeric characters. In addition to that, the document that has the least alpha-numeric characters consists of 39 alpha-numeric characters.

- Histogram: b) The column leaf_label - class on the x-axis

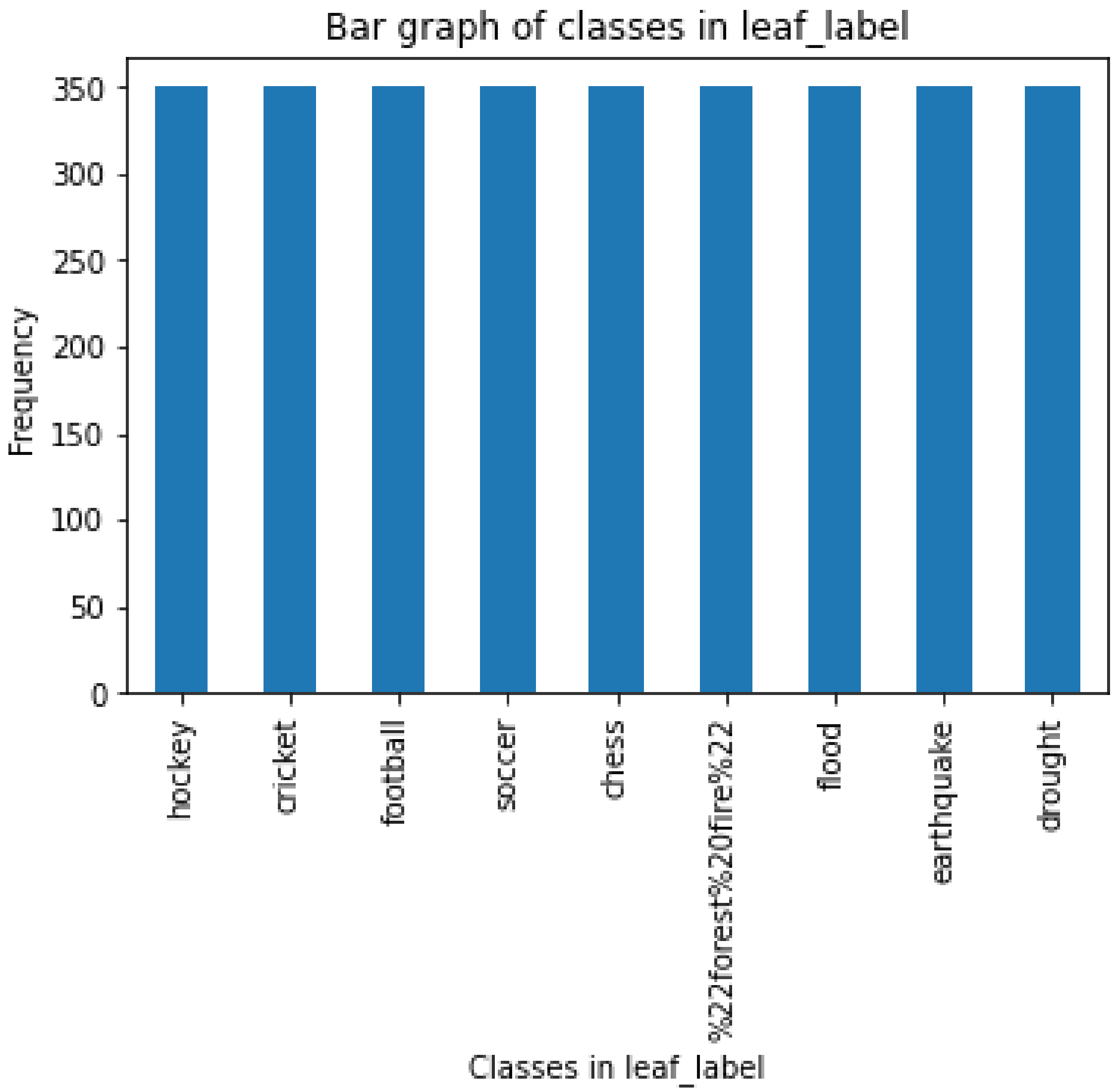


Figure 2: Histogram of leaf label classes

Interpretation of Figure 2: There are exactly 350 samples for each leaf label which indicates that we have a perfectly balanced dataset. Perfectly balanced dataset avoids class imbalance problems.

- Histogram: c) The column root_label- class on the x-axis

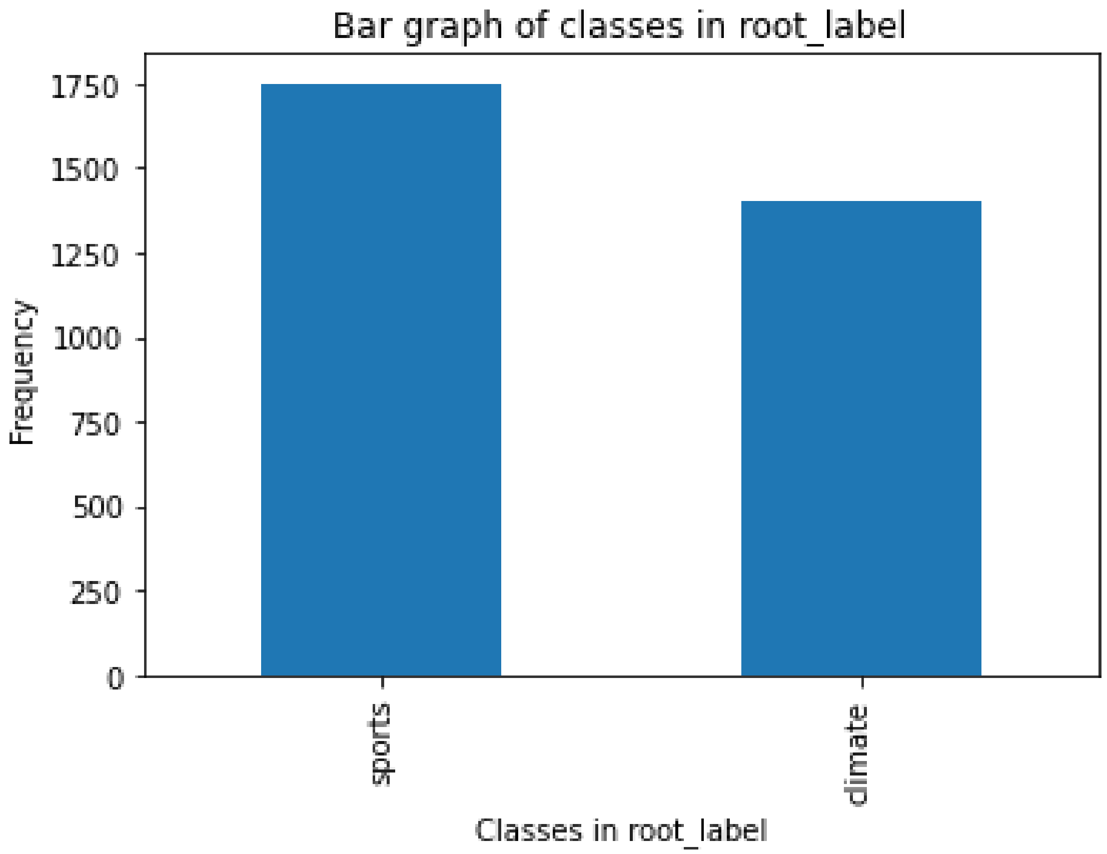


Figure 3: Histogram of root label classes

Interpretation of Figure 3: There are exactly 1750 samples for sports label and 1400 samples for climate label. Although two numbers are not exactly the same, they are close to each other. Therefore, the class imbalance issue will not be significant.

Binary Classification:

In this part, features will be extracted. Then, dimensionality of the dataset will be reduced and at last binary classification will be performed using the root labels.

Splitting the entire dataset into training and testing data:

Data should be splitted into training and testing datasets with the ratio of 80 to 20 to measure the performance of our binary classification model.

Question 2:

After using the train_test_split method, data is splitted. Training dataset consists of **2520 samples** and the testing dataset consists of **630 samples**.

Feature extraction:

First thing to do when dealing with a dataset consisting of a corpus of text is finding a good representation of each sample. Feature extraction module is required to transform raw text features into processed computationally usable and useful features.

Main attributes of a good feature extraction module is that it should preserve the class-discriminating information and avoid the computational burden that is caused by large amounts of data.

One model: The Bag-of-Words (BOW): Using this model, you can use the frequencies of words inside each sample as features using a fixed vocabulary of words.

Before compiling the vocabulary, one should clean some HTML artifacts using the given clean(text) function.

Compilation of the vocabulary:

a) Each sample should be split into sentences.
b) Each sentence should be split into words.
c) The list of words per sentence are passed to a position tagger. Note that WordNetLemmatizer requires Morphy tags. However, the pos_tag function from nltk library returns penn treebank tags. Therefore, mapping is required between two position tagger types.
d) Using those tags each word is lemmatized or stemmed. WordNetLemmatizer() object is capable of lemmatizing words using the Morphy position tags and PorterStemmer() object is capable of stemming words.
e) Word list should be filtered. We have filtered the word list by excluding english stop words and words that contain digits.
f) Remaining word list is added to the set of vocabulary. This process continues until all training dataset is used.

Lastly, using the CountVectorizer, we have obtained the data matrix which consists of the count of each word per data point.

Better model: The Term Frequency-Inverse Document Frequency Model (TF-IDF): In addition to using CountVectorizer, TF-IDF transformation scales the count of the word by the number of data points which the word has appeared. The data matrix is transformed using the TF-IDF transformation.

Question 3:

Pros and cons of lemmatization versus stemming:

Lemmatizer uses the positional tag and context to find the basic form of the word, whereas stemming only removes the rightmost characters of words while trying to map word derivatives to a basic word. However, stemming can create illogical words such as “s”. Therefore, the advantage of lemmatization over stemming is that lemmatization produces more accurate and logical words.

However, there is a downside for producing accurate words. Lemmatization is more computationally expensive since it uses a pre learned vocabulary and tries to simplify the word using context of the word. Stemming is computationally cheap since it uses a heuristic method to remove some characters from the end of the word.

The dictionary size for lemmatization is bigger than the stemming dictionary. Lemmatization dictionary consists of 13160 words and the stemming dictionary consists of 11755 words. Stemming dictionary is smaller since stemming can map two unrelated words into the same word, therefore lemmatization is capable of producing a more varied dictionary.

Min_df and TF-IDF matrix:

When we increase min_df, we have observed words that do not occur frequently are not added to the vocabulary. Therefore, columns of the data matrix decrease as the dictionary size decreases. Conversely, when min_df is decreased, the data matrix gets bigger since there are more words in the dictionary.

Stop words and punctuations before or after lemmatizing:

English stop words should be removed after lemmatizing, since the meaning of the sentence is not preserved after removing stop words. If we discard stop words before lemmatizing, position taggers may falsely tag the word and lemmatization may produce out of context words since the sentence structure is changed. Since numbers and punctuations can not be lemmatized, they should be used while lemmatizing. Lemmatizing requires a full structured sentence, so numbers and punctuations are important to preserve the sentence structure. In brief, we should do lemmatization first, and remove stop words, punctuations and numbers second.

Shape of TF-IDF train and test matrices:

Using lemmatization produces a training data matrix with 2520 samples and 13160 features(words) and a testing data matrix with 630 samples and 13610 features(words).

Using stemming produces a testing data matrix with 2520 samples and 11755 features(words) and a testing data matrix with 630 samples and 11755 features(words).

Dimensionality Reduction:

For the latter, TF-IDF train and test matrices that are obtained through lemmatization are used.

Note that TF-IDF train and test matrices have shapes 2520x13160 and 630x13160. Further computations with these matrices are expensive and may lead to poor performance of classifiers. Therefore, dimensionality reduction techniques such as Latent Semantic Indexing and non-Negative Matrix Factorization are commonly used. Also, the TF-IDF matrix is sparse, therefore it is easier to project features into lower dimensions.

Latent Semantic Indexing(LSI): The LSI is obtained by taking SVD of the TF-IDF matrix. Then singular vectors that correspond to the largest k singular values are used for projecting the TF-IDF into k dimension. One can use TruncatedSVD, randomized_svd or scipy.sparse.linalg.svds to obtain sparse SVD. TruncatedSVD is the most commonly used for dimensionality reduction transformation, therefore the data is transformed using the TruncatedSVD. However, Frobenius norms are calculated using randomized_svd and svds.

Non-negative Matrix factorization(NMF): NMF is obtained using a loss function Frobenius Norm of X-WH where W,H>=0. When the loss function is minimized, we obtain the W dimensionality reduced TF-IDF matrix.

Question 4:

Explained Variance Ratio:

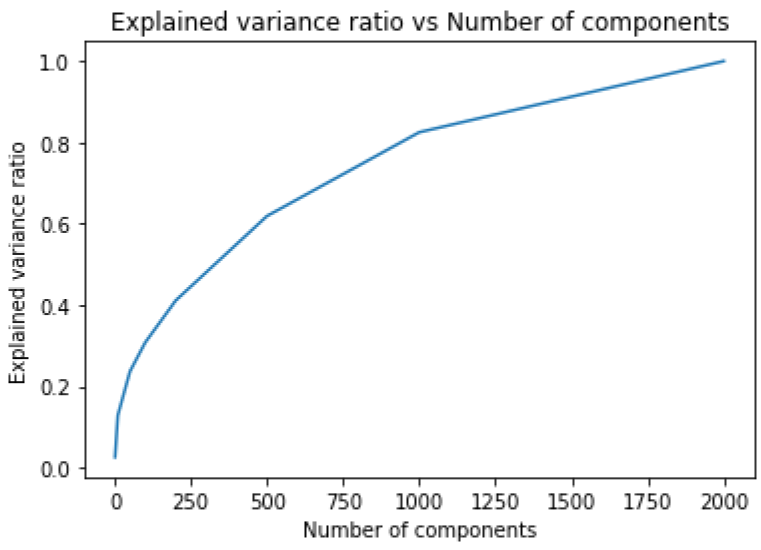


Figure 4: Explained variance ratio vs number of components

Concavity of the figure suggests that as the number of components increases, the contribution of those components to the explained variance decreases. Using the first terms effect in explaining the data variance is better then using the last terms effect. Also, note that 90% of the data variance can be explained with only 1000 features.

Comparison of NMF and LSI:

To compare two methods, dimensionality is reduced to 50 features. Then, reconstruction residual MSE error using NMF and LSI are computed. The reconstruction step for each method is different. For LSI, one should find the k right and left singular vectors and form a new U and V matrices. Also, one should find the k singular values and construct the diagonal S matrix. Then, U*S*V.T will give the reconstructed data matrix. For NMF, W and H matrices should be computed using the NMF function with 50 components. Multiplying those matrices will give the reconstructed data matrix.

The reconstruction residual MSE for LSI is **1853.74**.
The reconstruction residual MSE for NMF is **1875.75**.

Results indicate that the LSI is performing better than NMF with a slight difference. LSI was able to capture more information with k = 50, since matrices are not limited to be non-negative.

For classification algorithms, LSI with k = 50 is used.

Classification Algorithms:

In this part, using the transformed data matrix, classifiers are trained and their performances are evaluated. Also, binary classification tasks will be performed in this section. Each document will be either labeled as sports or climate.

In order to evaluate the performance some metrics should be computed. These metrics are accuracy, precision, recall, f1 score , confusion matrix and ROC curve.

Accuracy: In mathematical terms, accuracy is defined as (TP + TN)/(TP + FP + TN + FN) which calculates the percentage of correct labeling over all labeling. Therefore, getting a high accuracy means that the classifier is able to correctly classify each label.

Precision: Precision is related to positive labeling. In mathematical terms, precision is defined as (TP /(TP + FP). It calculates the percentage of correctly labeled actual positives over all labeled positives.

Recall: In mathematical terms, recall is defined as (TP/ (TP + FN). It calculates the percentage of correctly labeled actual positives over all actual positives.

F1 Score: In mathematical terms, F1 score is defined as (2* Precision * Recall / (Precision + Recall)). Therefore, F1 score is the harmonic mean of Precision and Recall. F1 Score can be chosen over accuracy since it also includes the balance of the correct labeling, whereas accuracy does not include the balance of correct labeling. Suppose there is an imbalance at labeling. The accuracy can be reported as above. Let's say TP = 10, TN = 80, FN = 5, FP = 5. Then accuracy will be 90% which is near to what we found.Recall and precision will be 66% which is very low. F1 score will be 66 indicating that the classifier is performing poorly. Therefore, when TN and TP are not equal in number, one should check F1 Score and False Positive Rate to distinguish the performance of the classifier. In our case, let's assume that there is an imbalance between the numbers of TN and TP.

Confusion matrix: Confusion matrix reports the TN,FN,FP,TP in a matrix format.
TN is the count of [prediction == 0 && actual == 0] at matrix[0,0].
FN is the count of [prediction == 0 && actual == 1] at matrix[0,1].
FP is the count of [prediction == 1 && actual == 0] at matrix[1,0].
TP is the count of [prediction == 1 && actual == 1] at matrix[1,1].
This is not in the case of multi-class classification.

The ROC curve: ROC is related to the relationship between true positive rate and false positive rate in the context of different threshold values. AUC which area under the curve is an important metric that shows the usefulness of the test procedure. If the number is high, AUC indicates that the model is performing well. If the number is low, the model is performing poorly. True positive rate is also called sensitivity which is defined as TP/(TP+FN). This ratio indicates the probability that an actual positive will test positive. False positive rate which is also called false alarm ratio is defined as FP/FP+TN.

SVM:

A support vector machine (SVM) is a type of supervised learning algorithm that can be used for classification or regression tasks. The basic idea behind SVMs is to find the best boundary that separates the different classes in the training data. The boundary is chosen so that it maximally separates the classes while also maximizing the margin, which is the distance between the boundary and the closest data points of each class. Once the boundary is determined, new data can be classified by seeing on which side of the boundary it falls. SVMs are particularly useful when the data has many features and when the classes are well-separated by a linear boundary.

As the γ value decreases, the model is more open to making some mistakes. On the contrary, when the γ value increases, the model is penalized for making mistakes, therefore the model overfits the data more.

Question 5

SVM with γ = 1000:

```
****** Performance of SVM with C = 1000 *****
Accuracy:  0.9412698412698413
Precision:  0.9154411764705882
Recall:  0.9467680608365019
F-1 Score:  0.9308411214953272
Confusion Matrix:
```

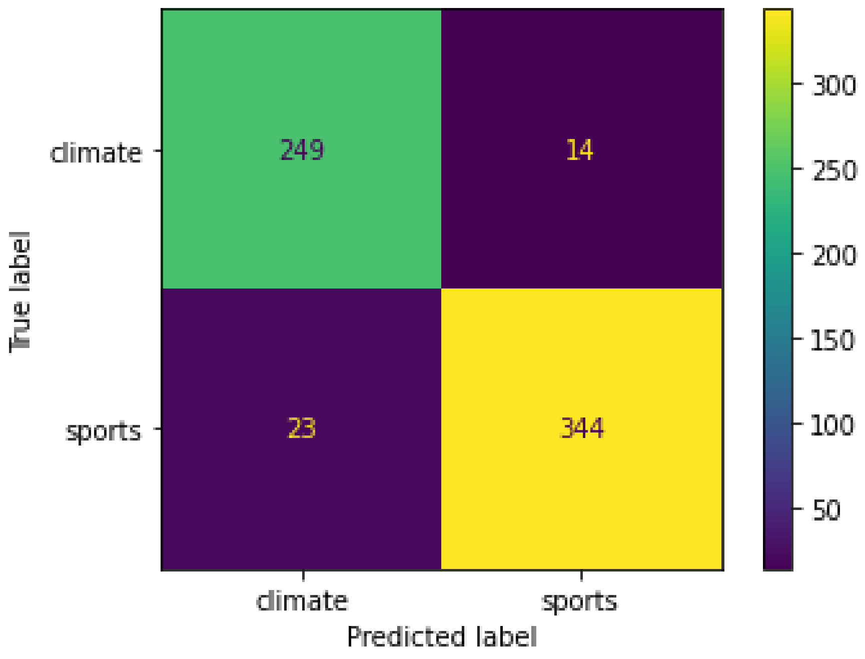


Figure 5: Confusion matrix of SVM with C = 1000

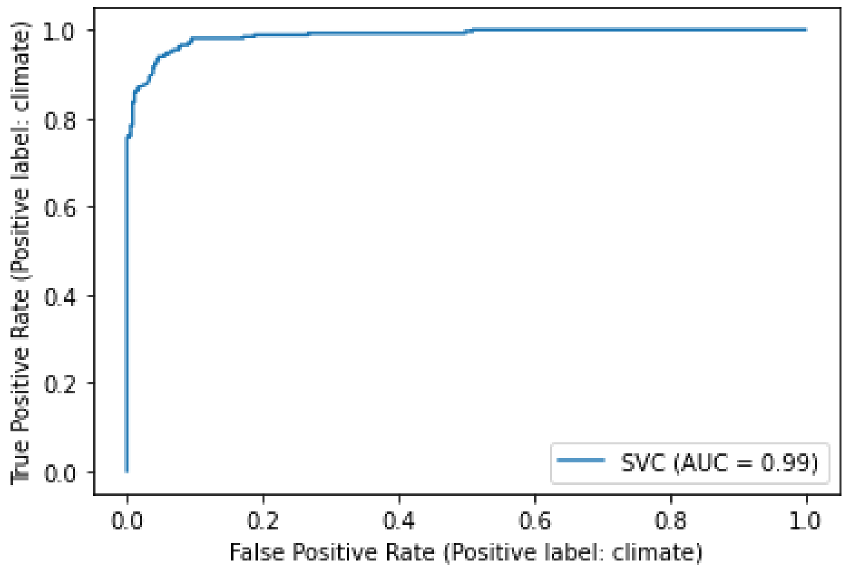


Figure 6: ROC of SVM with C = 1000

SVM with γ = 0.0001:

```
****** Performance of SVM with C = 0.0001 *****
Accuracy:  0.5825396825396826
Precision:  0.0
Recall:  0.0
F-1 Score:  0.0
Confusion Matrix:
```

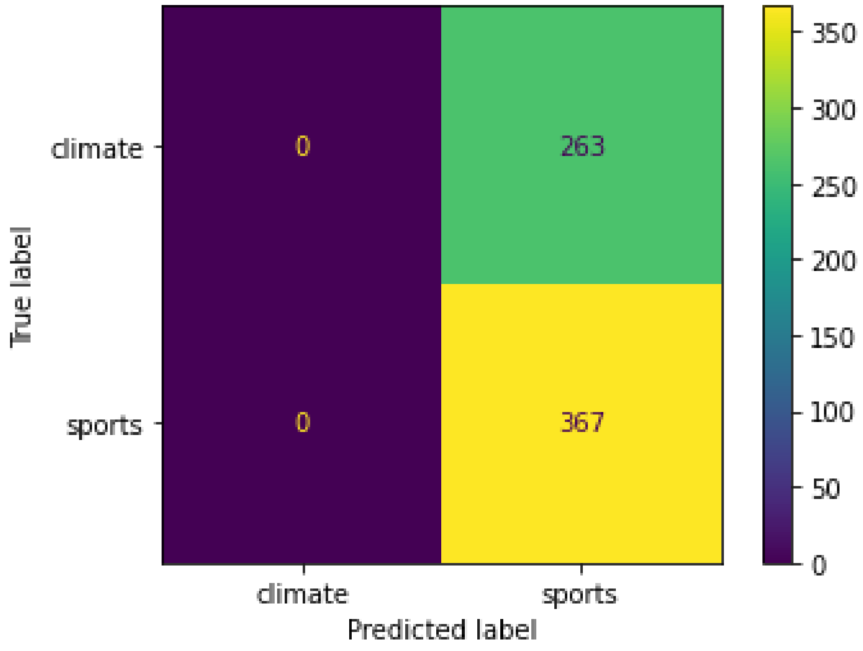


Figure 7: Confusion Matrix of SVM with C = 0.0001

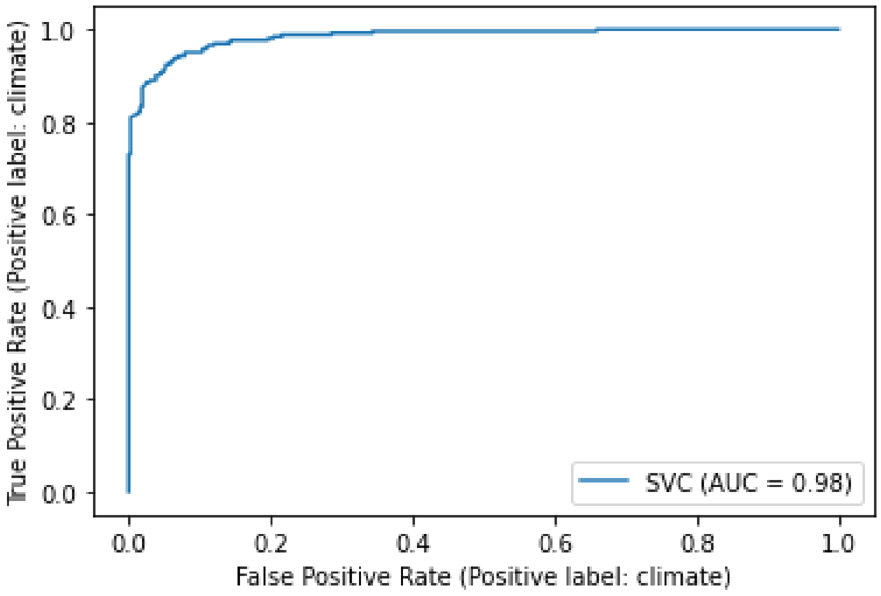


Figure 8: ROC of SVM with C = 0.0001

SVM with γ = 100000:

```
****** Performance of SVM with C = 100000 *****
Accuracy:  0.9412698412698413
Precision:  0.9124087591240876
Recall:  0.9505703422053232
F-1 Score:  0.931098696461825
Confusion Matrix:
```

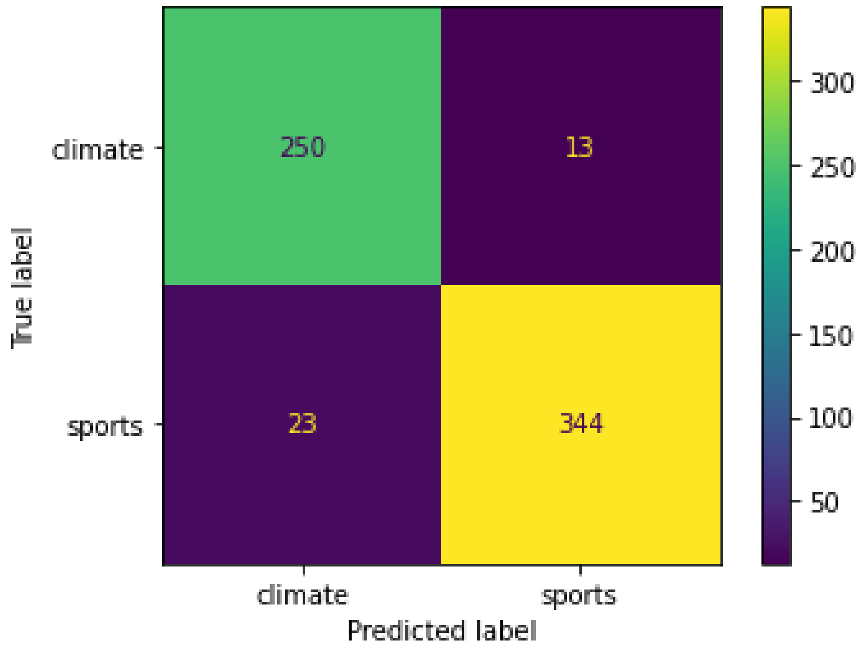


Figure 9: Confusion Matrix of SVM with C = 100000

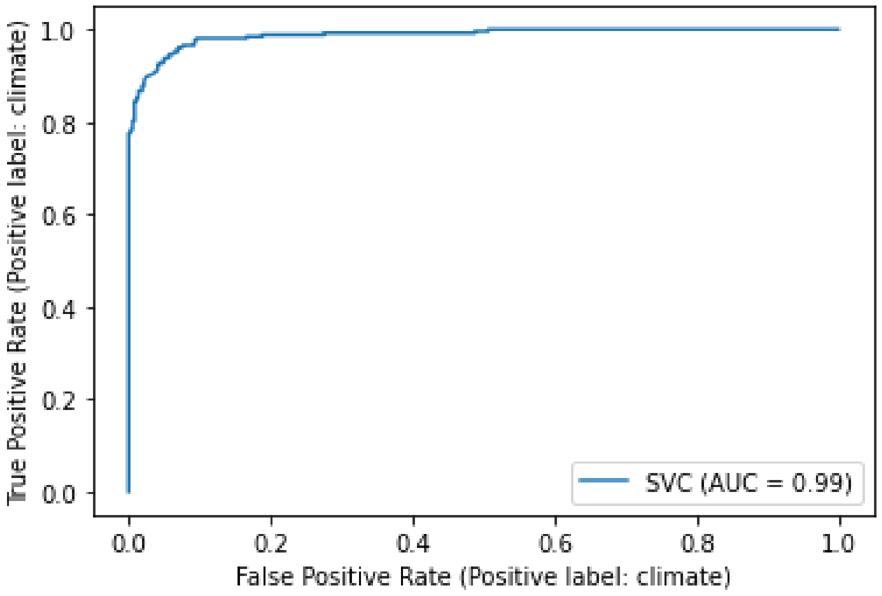


Figure 10: ROC of SVM with C = 100000

Which one performs better:

It is observed that SVM C = 100000 works best among three SVM models. High gamma value cares more about misclassified labels than other gamma values which results in a better performing model. It is noteworthy to mention that using a high gamma value can lead to overfitting since the model will try to label everything correctly according to the training data. Therefore, the model can lose the generalization property.

Soft Margin SVM:

The model with soft margin has only predicted test labels as one class which is the sports label regardless of the training data. Labeling all test data as the sports label is intuitive since the model does not care to get high accuracy. Therefore, the model tries to minimize the norm of the weight vector. There are 263 climate samples and 367 test sports labels. Soft margin SVM classifies all test labels as the majority class.

ROC Curve of Soft Margin SVM:

Although we get low accuracy from the soft margin SVM, AUC of the soft margin SVM is 0.98 which is high. Therefore, the ROC curve does not reflect the performance of the soft margin SVM. ROC curve of the soft margin SVM looks reasonable since there is an exchange between TPR and FPR.

Cross Validation:

Average validation accuracy is used to find the best gamma value among the set { 10^k: -3 <= k <= 6, k ∈ ℤ}.

Results(Note that C is gamma parameter for SVM, best one is marked by bold):

C = 0.001	Average Accuracy = 0.5488095238095239
C = 0.01	Average Accuracy = 0.5488095238095239
C = 0.1	Average Accuracy = 0.9234126984126985
C = 1.0	Average Accuracy = 0.9321428571428572
C = 10.0	Average Accuracy = 0.9384920634920635
C = 100.0	Average Accuracy = 0.9428571428571427
C = 1000.0	**Average Accuracy = 0.9464285714285714**
C = 10000.0	Average Accuracy = 0.942063492063492
C = 100000.0	Average Accuracy = 0.9428571428571428
C = 1000000.0	Average Accuracy = 0.944047619047619

Best C(gamma) value is 1000.0 with average accuracy score 0.9464285714285714

Results of SVM with Best gamma:

```
****** Performance of SVM with best c *****
Accuracy:  0.9412698412698413
Precision:  0.9154411764705882
Recall:  0.9467680608365019
F-1 Score:  0.9308411214953272
```

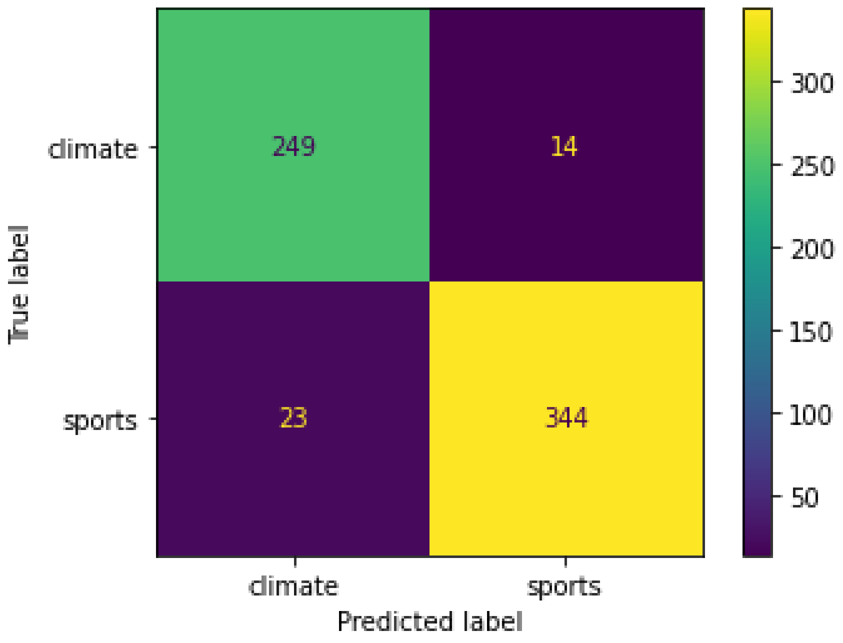


Figure 11: Confusion Matrix of SVM with best gamma(1000)

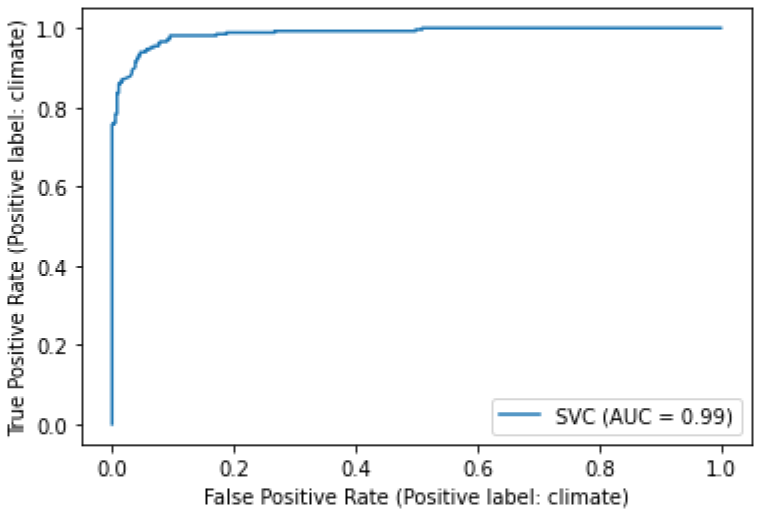


Figure 12: ROC of SVM with best gamma(1000)

Logistic Regression:

Logistic Regression is a type of supervised machine learning algorithm that is used for binary classification and also it can be used for multi-class classification. It is a generalized linear model that uses a logistic function to model a binary dependent variable. The logistic function, also called the sigmoid function, maps any input value to a value between 0 and 1, which can be interpreted as the probability of the input value belonging to a certain class.

Given a set of input features, the algorithm will learn the coefficients of the features that will be used to make predictions. The predicted probability is then thresholded at 0.5 to predict the class label. The goal of the logistic regression is to find the best coefficients that maximize the likelihood of the observed data.

In order to generalize models and avoid overfitting one can use L1 and L2 regularization techniques. In L1 regularization, $\lambda|w|$ term is added. In L2 regularization, $\lambda|w|^2$ is added. L1 regularization shrinks some coefficients to zero, therefore feature selection is also performed while training the classifier. In L2 regularization, coefficients get smaller and regularization avoid overfitting.

Question 6:

No regularization:

```
****** Performance of Logistic Regression without penalty term *****

Accuracy:  0.9412698412698413
Precision:  0.9154411764705882
Recall:  0.9467680608365019
F-1 Score:  0.9308411214953272
```

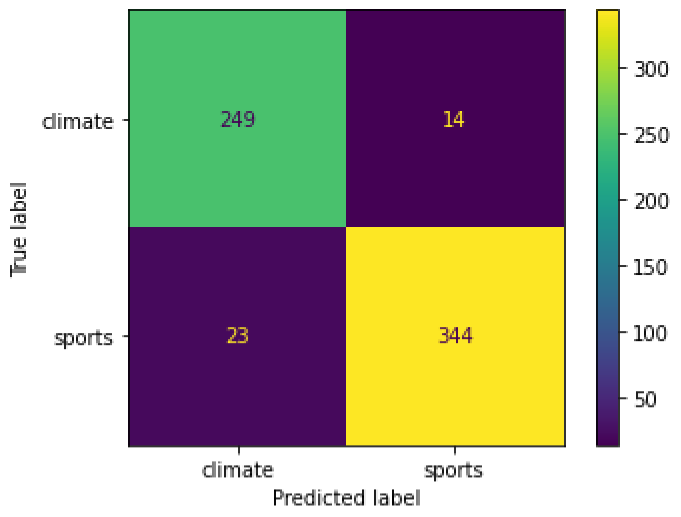


Figure 13: Confusion Matrix of Logistic Regression without penalty term

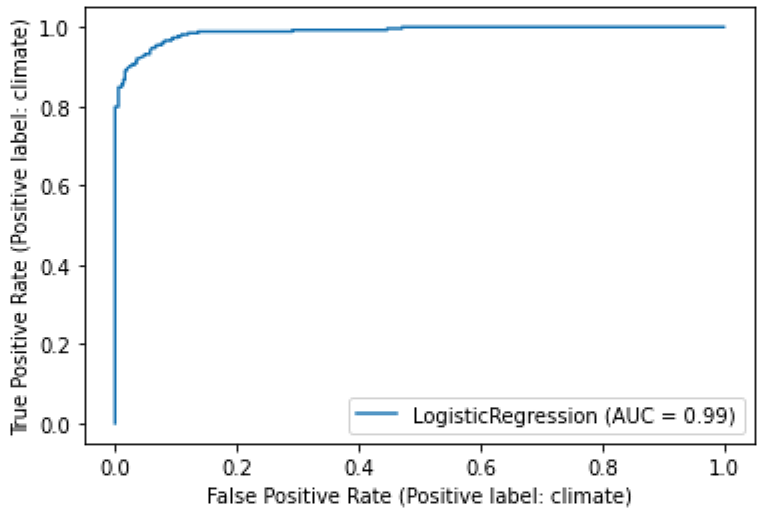


Figure 14: ROC of Logistic Regression without penalty term

Using CV find regularization parameter for L1:

```
penalty =  1e-05       Average Accuracy =  0.4511904761904762
penalty =  0.0001      Average Accuracy =  0.4511904761904762
penalty =  0.001       Average Accuracy =  0.4511904761904762
penalty =  0.01        Average Accuracy =  0.4511904761904762
penalty =  0.1         Average Accuracy =  0.9095238095238096
penalty =  1.0                Average Accuracy =  0.932936507936508
penalty =  10.0        Average Accuracy =  0.9424603174603174
penalty =  100.0       Average Accuracy =  0.9468253968253968
penalty =  1000.0      Average Accuracy =  0.9468253968253968
penalty =  10000.0     Average Accuracy =  0.9472222222222223
penalty =  100000.0 Average Accuracy =  0.9472222222222223
```

(L1) Best penalty value is 10000.0 with average accuracy score 0.947

Using CV find regularization parameter for L2:

```
penalty =  1e-05       Average Accuracy =  0.5488095238095239
penalty =  0.0001      Average Accuracy =  0.5488095238095239
penalty =  0.001       Average Accuracy =  0.5488095238095239
penalty =  0.01        Average Accuracy =  0.5674603174603174
penalty =  0.1         Average Accuracy =  0.913095238095238
penalty =  1.0         Average Accuracy =  0.9281746031746032
penalty =  10.0        Average Accuracy =  0.9357142857142856
penalty =  100.0       Average Accuracy =  0.942063492063492
penalty =  1000.0      Average Accuracy =  0.9464285714285714
penalty =  10000.0     Average Accuracy =  0.9464285714285714
penalty =  100000.0  Average Accuracy =  0.9468253968253968
(L2) Best penalty value is  100000.0  with average accuracy score  0.947
```

Compare performance between no-regularization, L1, L2 using the parameters found above:

```
****** Performance of Logistic Regression with best penalty L1 regularization *****
Accuracy:  0.9396825396825397
Precision:  0.9120879120879121
Recall:  0.9467680608365019
F-1 Score:  0.9291044776119403
```

Confusion Matrix:

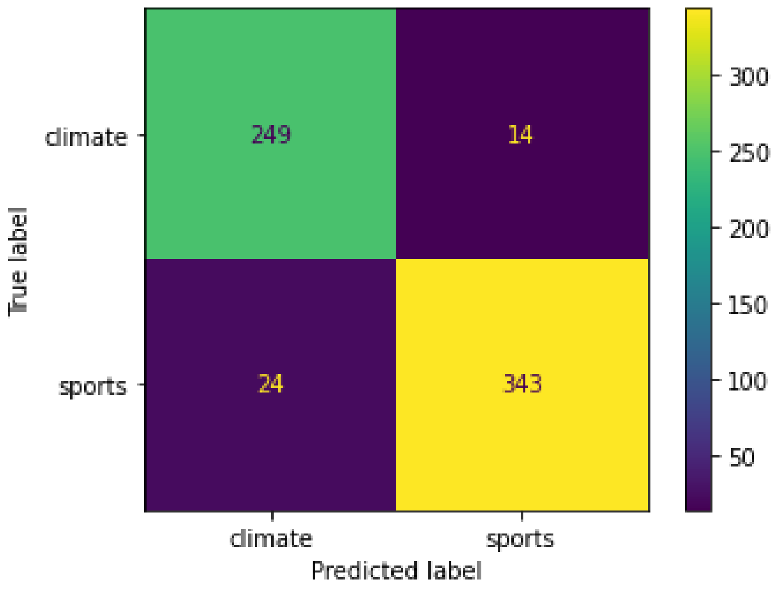


Figure 15: Confusion Matrix for L1 regularized Logistic Regression, penalty = 100000

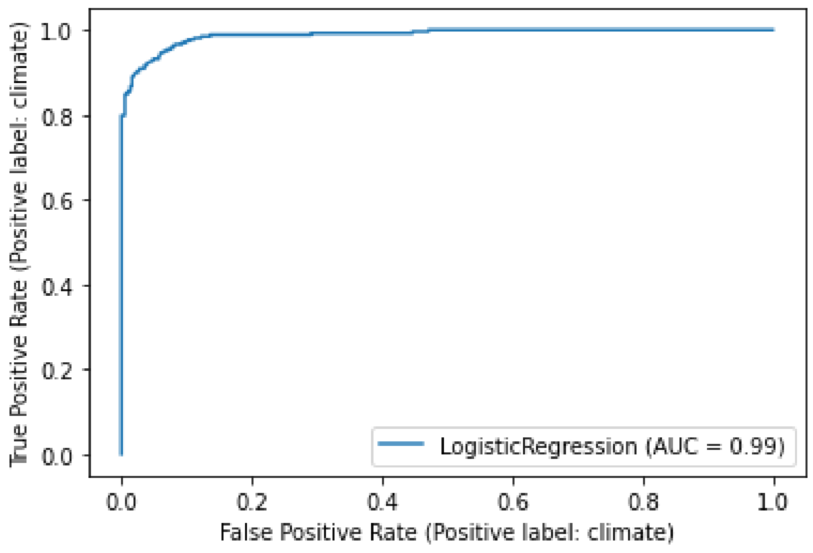


Figure 16: ROC for L1 regularized Logistic Regression, penalty = 100000

****** Performance of Logistic Regression with best penalty L2 regularization *****
Accuracy: 0.9396825396825397
Precision: 0.9120879120879121
Recall: 0.9467680608365019
F-1 Score: 0.9291044776119403
Confusion Matrix:

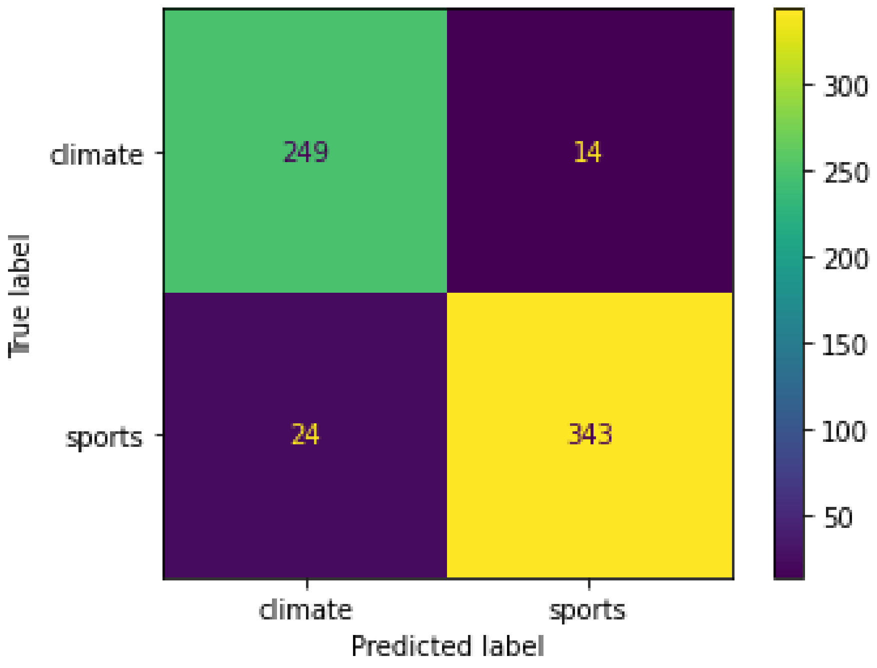


Figure 17: Confusion Matrix for L2 regularized Logistic Regression, penalty = 100000

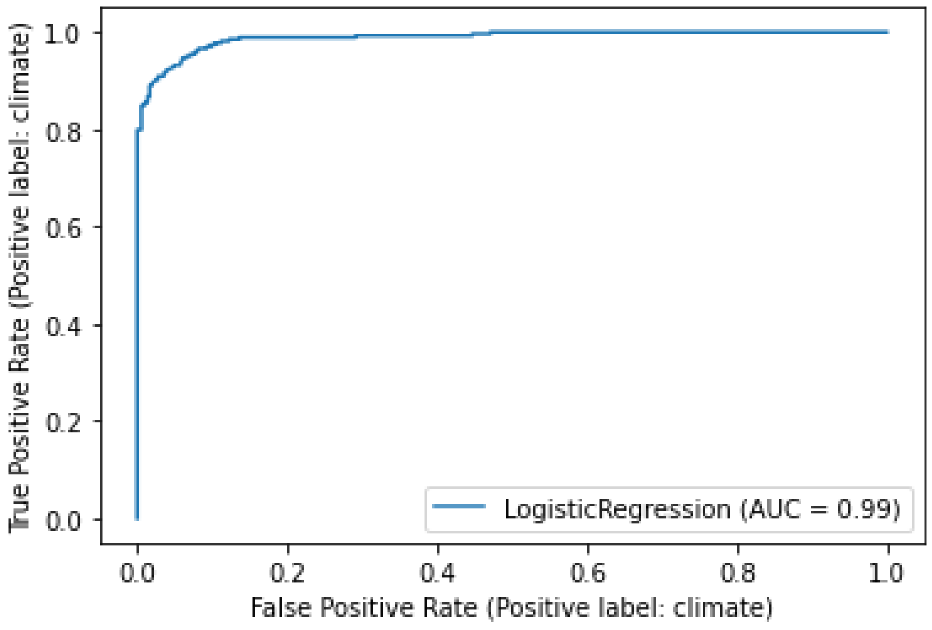


Figure 18: ROC for L2 regularized Logistic Regression, penalty = 100000

No regularization has performed slightly better than L1 and L2 regularized models at all metrics.

How does the regularization parameter affect the test error?

In our case, the regularization parameters slightly increased the test error.

How are the learnt coefficients affected?Why might one be interested in each type of regularization?

L1 regularization forces some weights to shrink to zero. Therefore, L1 regularization is also performing feature selection on the training dataset. L1 regularization can be used to find the important features of the dataset. In result, L1 regularization will produce sparse weight vectors, this is important when the feature set size is large.

L2 regularization forces all weights to shrink but those weights are not zero. L2 regularization is a technique used to prevent overfitting in logistic regression. Overfitting occurs when a model is too complex and is able to fit the noise in the data, as well as the underlying pattern. This can lead to poor performance on unseen data. L2 regularization, also known as weight decay, is a method of penalizing large weights in the model.

Difference between SVM and Logistic Regression:

Both logistic regression and linear SVM are trying to classify data points using a linear decision boundary, but the way they find this boundary is slightly different.

In logistic regression, the goal is to find the coefficients of the input features that maximize the likelihood of the observed data. The decision boundary is defined by the equation of the line that separates the two classes. Using the sigmoid function this linear equation turns into a probability. The predicted class label is determined by thresholding the predicted probability at 0.5.

In linear SVM, the goal is to find the hyperplane that maximally separates the two classes while also maximizing the margin, which is the distance between the boundary and the closest data points of each class. The decision boundary is defined by the equation of the hyperplane, which is of the same form as the equation of the line in logistic regression. However, in SVM the goal is to maximize the margin between the two classes, to do that the algorithm finds the support vectors which are the data points closest to the decision boundary and they are the ones that have the most impact on the position of the boundary, this makes SVM more robust to outliers.

In summary, logistic regression and linear SVM are similar in that they both use a linear decision boundary to classify data points, but they differ in the way they find this boundary: logistic regression maximizes the likelihood of the observed data, while linear SVM maximizes the margin and the robustness to outliers. Their performances differ due to the different ways that they utilize to find the linear boundary. However, metrics of each classifier are really close to each other, therefore there is no statistically significant difference in terms of the performance.

Naive Bayes:

Naive Bayes is a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. A Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to the presence of any other feature. However, it can perform poorly when the features are dependent on each other. Naive Bayes is also sensitive to class imbalance.

Gaussian Naive Bayes model is used for this section.

Question 7:

Performance of GaussianNB:

```
****** Performance of Gaussian Naive Bayes *****
Accuracy:  0.7888888888888889
Precision:  0.6675257731958762
Recall:  0.9847908745247148
F-1 Score:  0.7956989247311828
Confusion Matrix:
```

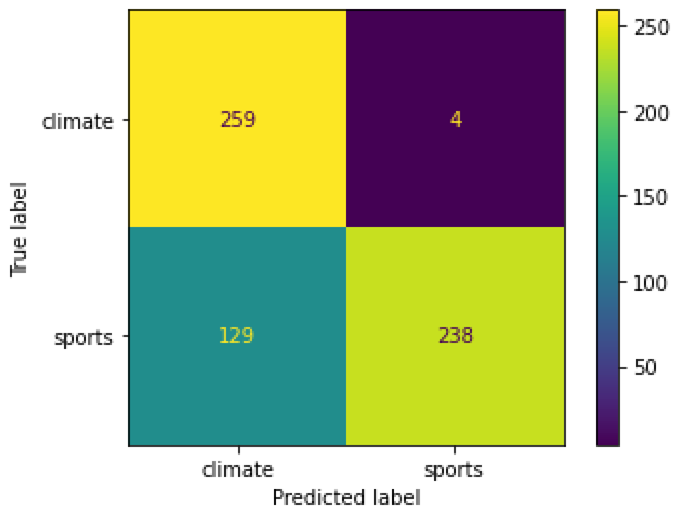


Figure 19: Confusion Matrix of GaussianNB

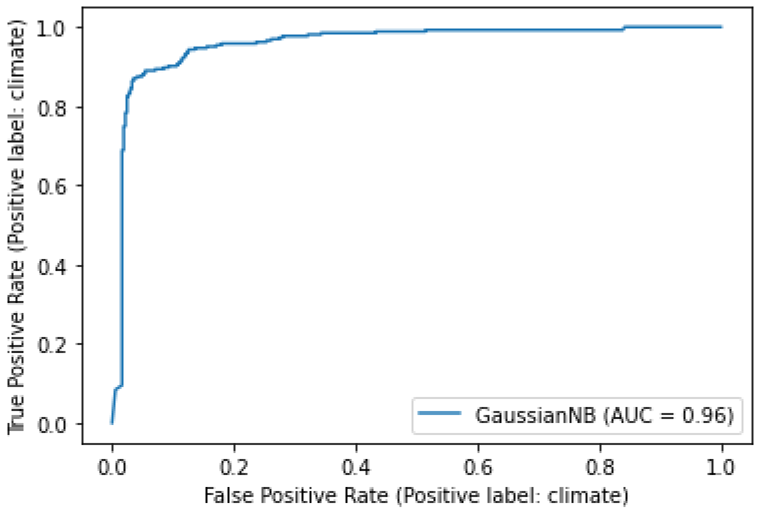


Figure 20: ROC of GaussianNB

Grid Search of Parameters:

Question 8:

Pipeline is created to perform feature extraction, dimensionality reduction and classification. 5-fold cross validation is used to assess method performance. Five best ranked pipelines are tested with a testing dataset.

Rank	min_df	Compression	Dim. Reduction	Classifier	Average accuracy
1	5	Lemmatization	LSI, k = 80	SVM, γ = 1000	0.95553
2	5	Lemmatization	LSI, k = 80	LogisticRegression,L2, γ = 100000	0.95409
3	3	Lemmatization	LSI, k = 80	LogisticRegression,L1, γ = 100000	0.94932
4	3	Stemming	LSI, k = 80	SVM, γ = 1000	0.94857
5	3	Stemming	LSI, k = 80	LogisticRegression,L2, γ = 100000	0.94831

Table 1: Five best combinations at grid search of parameters

Multiclass Classification:

In this part, rather than using the root labels of the dataset. We have used leaf labels to perform multiclass classification using SVM and Naive Bayes. Gaussian Naive Bayes can be directly used to perform multiclass classification, however since SVM is a binary classification technique, one should introduce One VS One and One VS the rest methods.

Question 9:

Class_weight parameter is used to solve the problem of class imbalance issue in the One VS the rest model. That parameter penalizes the penalty term of that class using the inverse proportion to the number of samples in that class.

Following results are obtained:

Results for Gaussian Naive Bayes:

Accuracy: 0.6190476190476191
Precision: 0.6311680561262082
Recall: 0.6089231734447665
F-1 Score: 0.6030937789733511
Confusion Matrix:

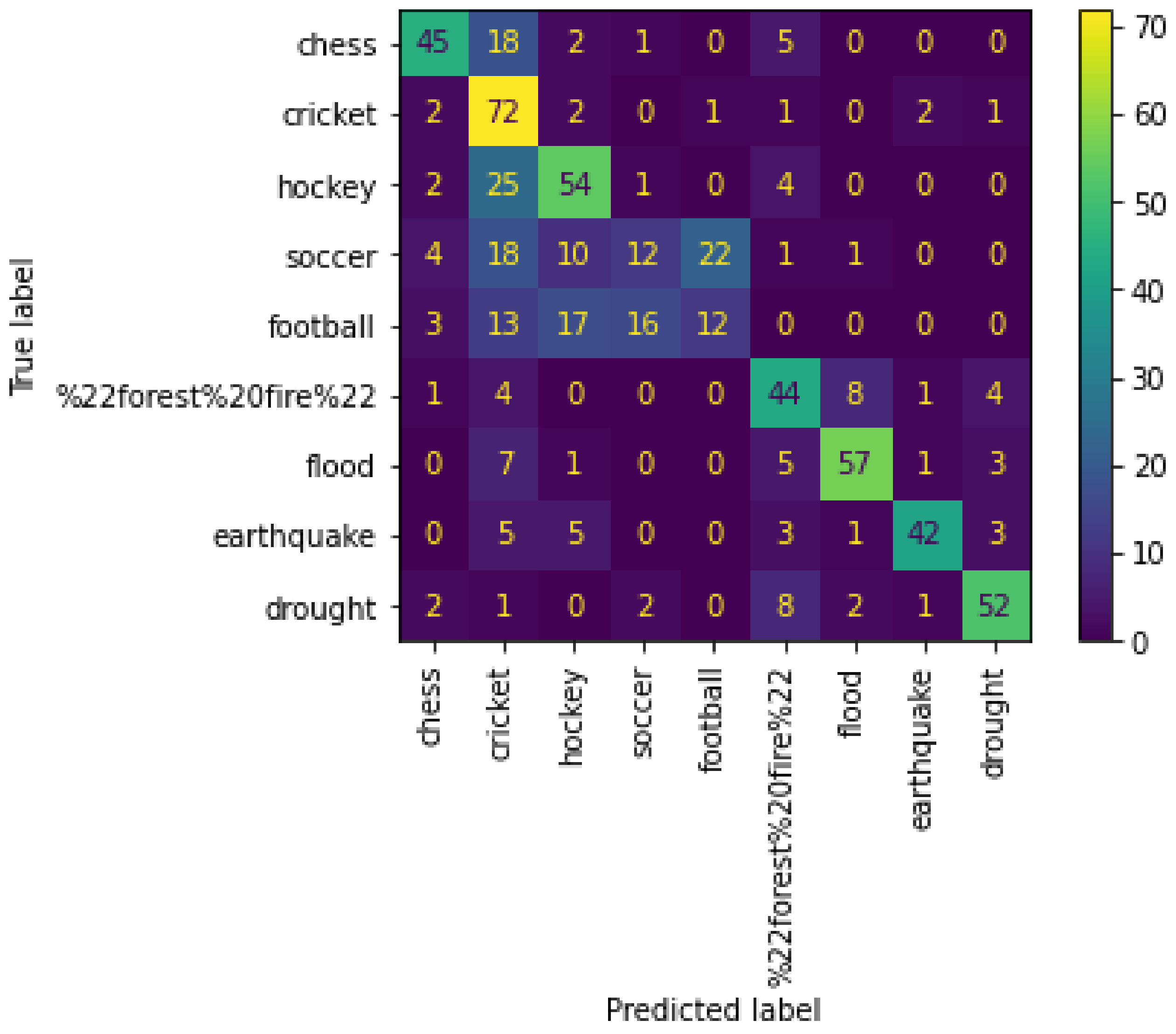


Figure 21: Confusion matrix for the multi-class gaussian naive bayes

Results of SVM OneVsOne with class imbalance:

Accuracy: 0.7380952380952381
Precision: 0.7310994751977695
Recall: 0.7281157937857858
F-1 Score: 0.7265404143453317
Confusion Matrix:

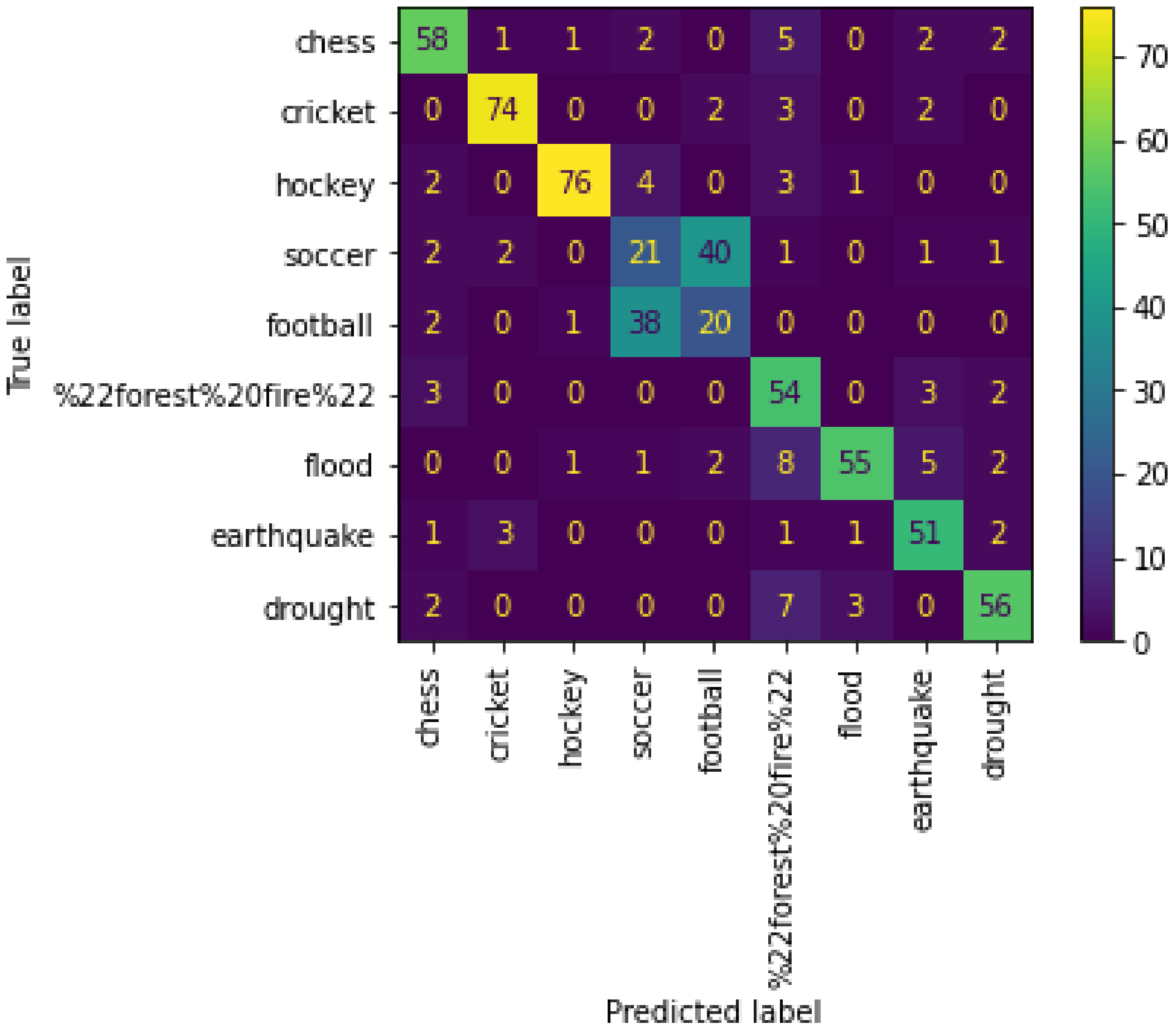
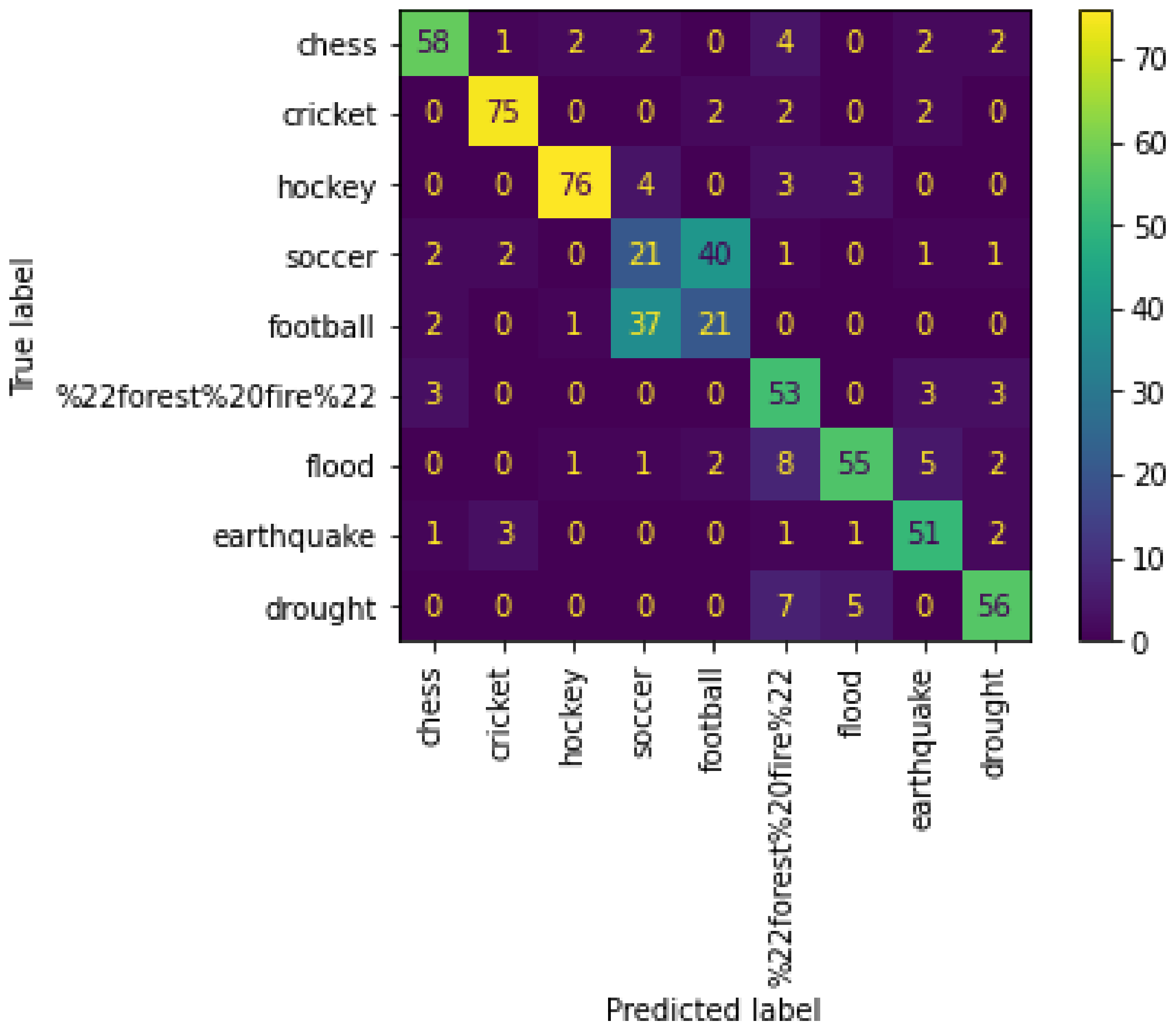


Figure 22: Confusion matrix for the OvO SVM with class imbalance

Results for SVM OvO with class balance:

Accuracy: 0.7396825396825397
Precision: 0.7307209528113184
Recall: 0.7295169148277005
F-1 Score: 0.7279322665710112
Confusion Matrix:



	chess	cricket	hockey	soccer	football	%22forest%20fire%22	flood	earthquake	drought
chess	58	1	2	2	0	4	0	2	2
cricket	0	75	0	0	2	2	0	2	0
hockey	0	0	76	4	0	3	3	0	0
soccer	2	2	0	21	40	1	0	1	1
football	2	0	1	37	21	0	0	0	0
%22forest%20fire%22	3	0	0	0	0	53	0	3	3
flood	0	0	1	1	2	8	55	5	2
earthquake	1	3	0	0	0	1	1	51	2
drought	0	0	0	0	0	7	5	0	56

Figure 23: Confusion matrix for the OvO SVM with class balance

Results for SVM OvR with class imbalance:

Accuracy: 0.7476190476190476
Precision: 0.7436562057736155
Recall: 0.738398125121176
F-1 Score: 0.7364042362189488
Confusion Matrix:

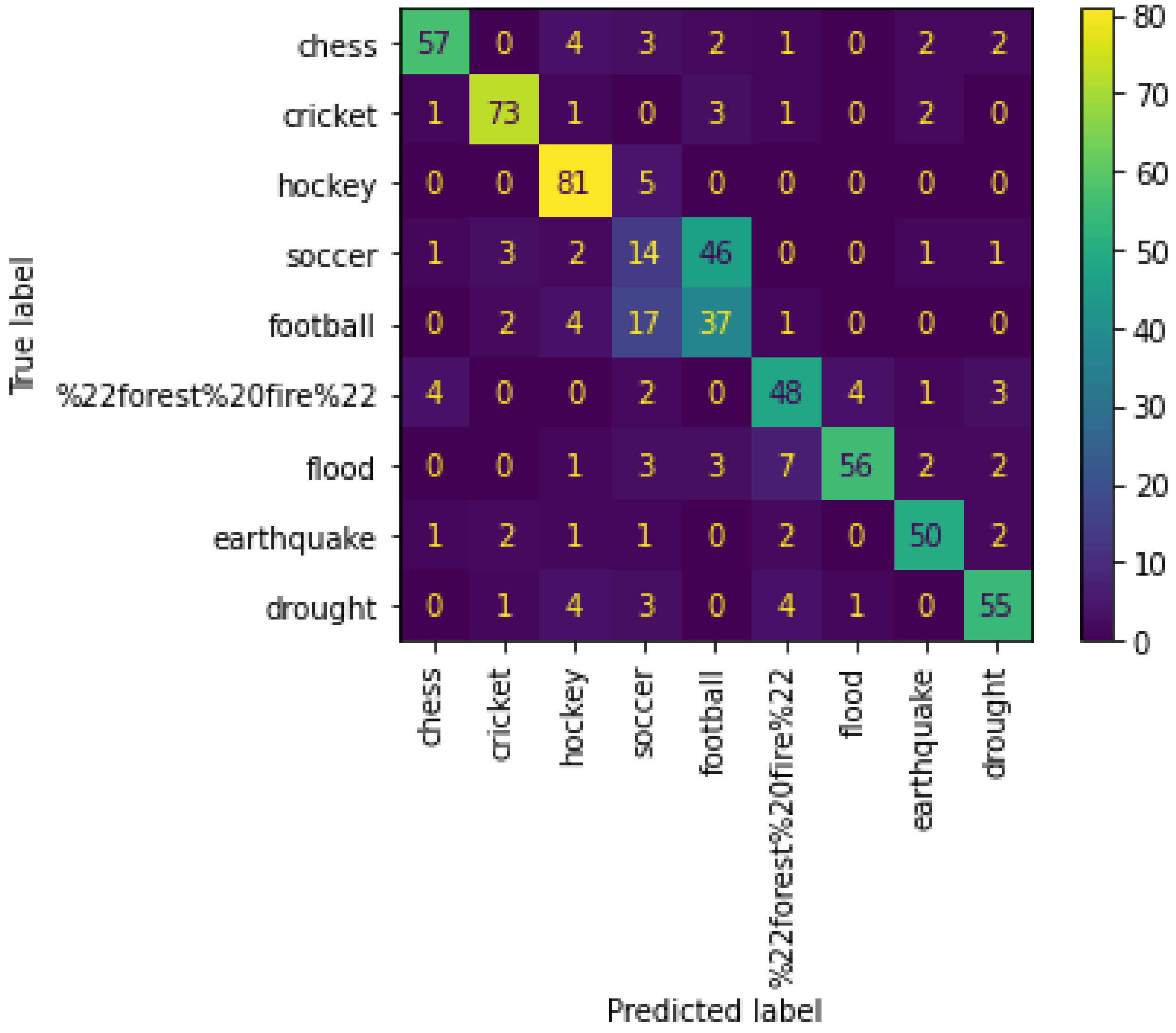


Figure 24: Confusion matrix for the OvR SVM with class imbalance

Results for SVM OvR with class balance:

Accuracy: 0.7698412698412699
Precision: 0.7541104194627795
Recall: 0.7615939397577111
F-1 Score: 0.7491046776462742
Confusion Matrix:

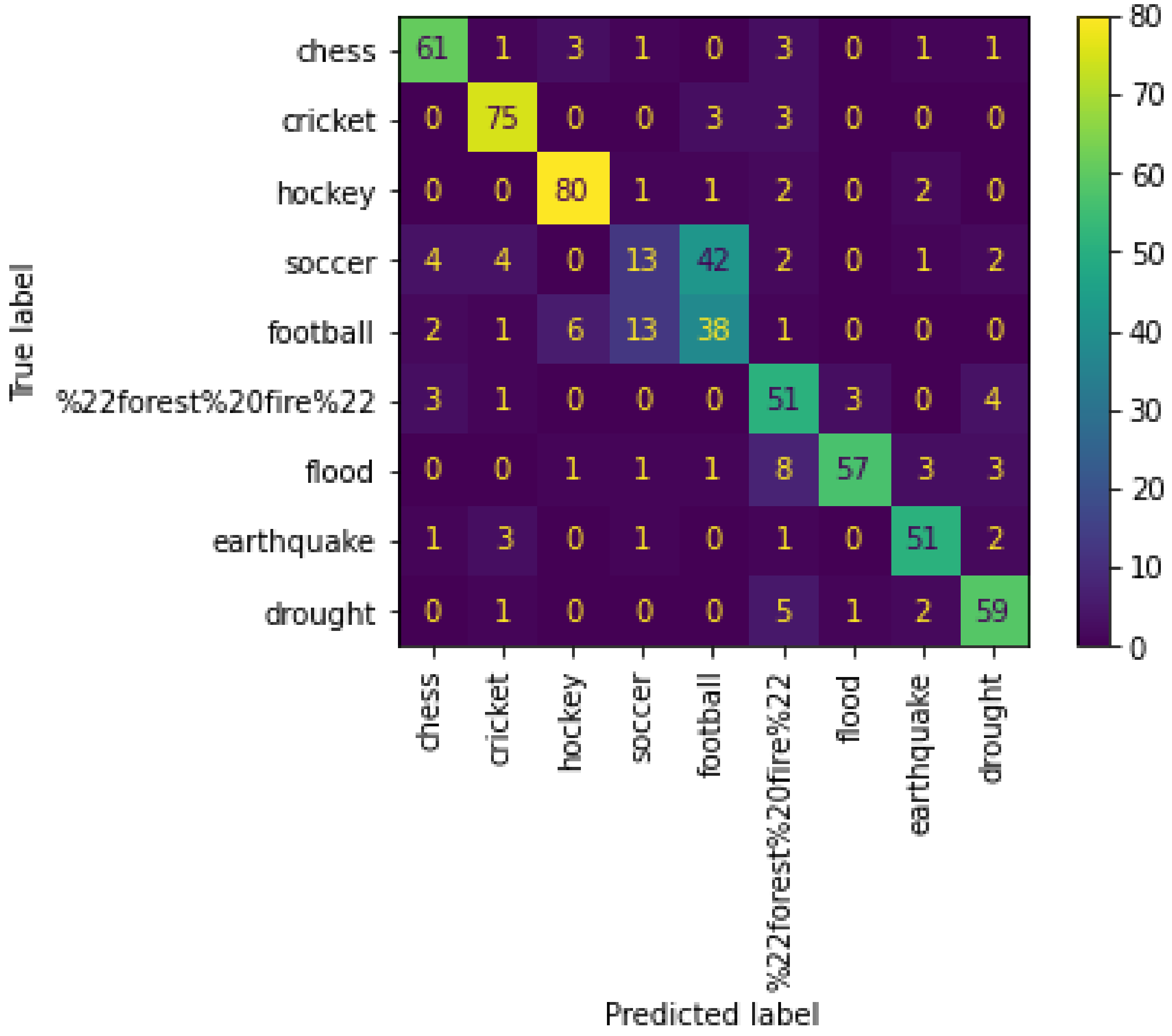


Figure 25: Confusion matrix for the OvR SVM with class balance

Interpretation of results:

As it can be clearly seen from the results, Gaussian Naive Bayes is performing poorly than the SVM. Best performing model is OneVsRest SVM with class balance, it has accuracy 0.76. The main reason for getting low accuracy scores in this part with each model is that classifiers are not able to clearly distinguish between soccer and football. It makes sense these terms are interchangeably used worldwide. American texts are using soccer to describe the sport played by foot, whereas the rest of the world uses football to describe the same sport.

Getting distinct visible blocks on the major diagonal is important since diagonals correspond to correct classification.

Since classifiers are not able to distinguish between football and soccer, we have decided to merge them into a single class called soccer_football.

Results after merging can be shown as below:

Results for GNB:

Accuracy: 0.6793650793650794
Precision: 0.7368410786872954
Recall: 0.6984671020413693
F-1 Score: 0.6965544849737366
Confusion Matrix:

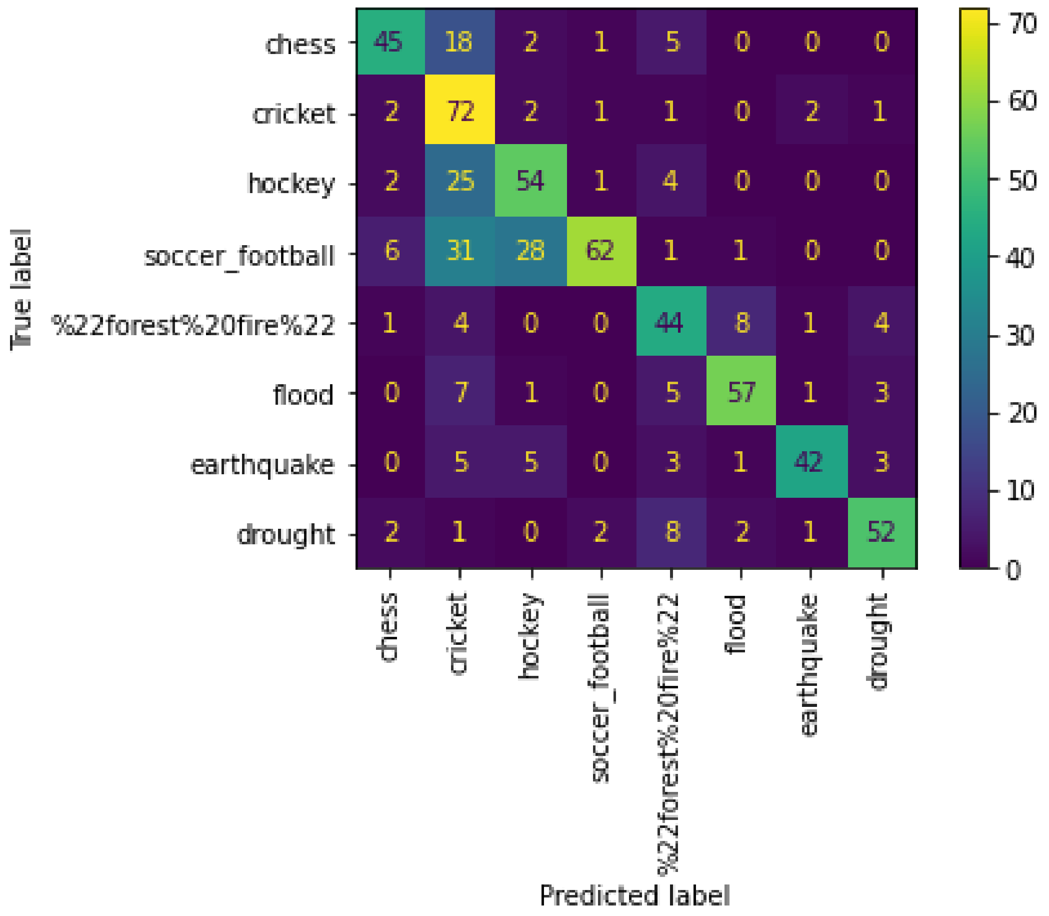


Figure 26: Confusion matrix for the Gaussian Naive Bayes, soccer and football merged

Results for SVM OneVsOne with class imbalance after merge:

Accuracy: 0.861904761904762
Precision: 0.8592181970040829
Recall: 0.8567435113902834
F-1 Score: 0.8544460096975244
Confusion Matrix:

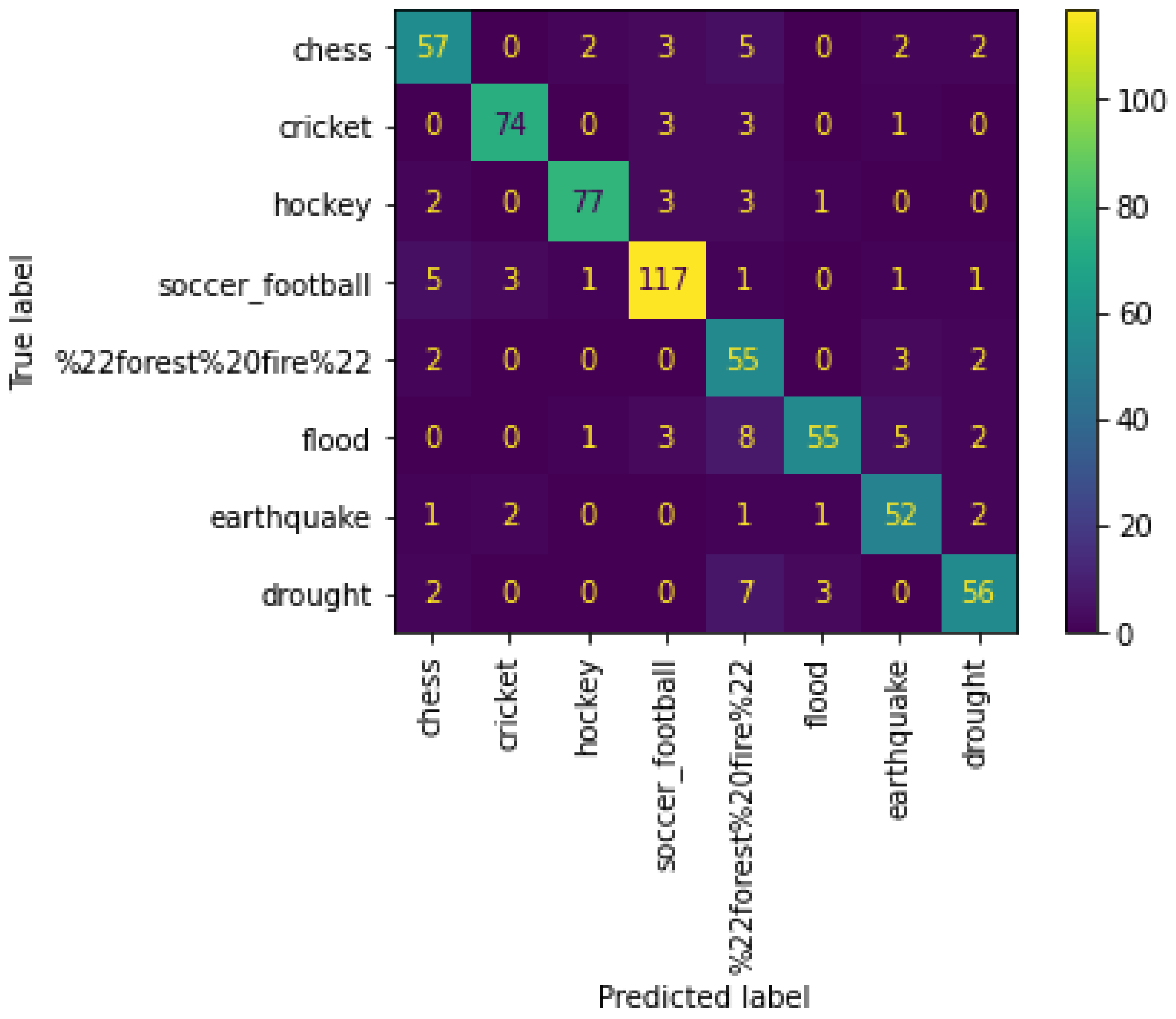


Figure 27: Confusion matrix for the SVM OneVsOne with class imbalance after merge

Results for SVM OneVsOne with class balance after merge

Accuracy: 0.8603174603174604
Precision: 0.8547404744709588
Recall: 0.8552118784820564
F-1 Score: 0.8522989926254795
Confusion Matrix:

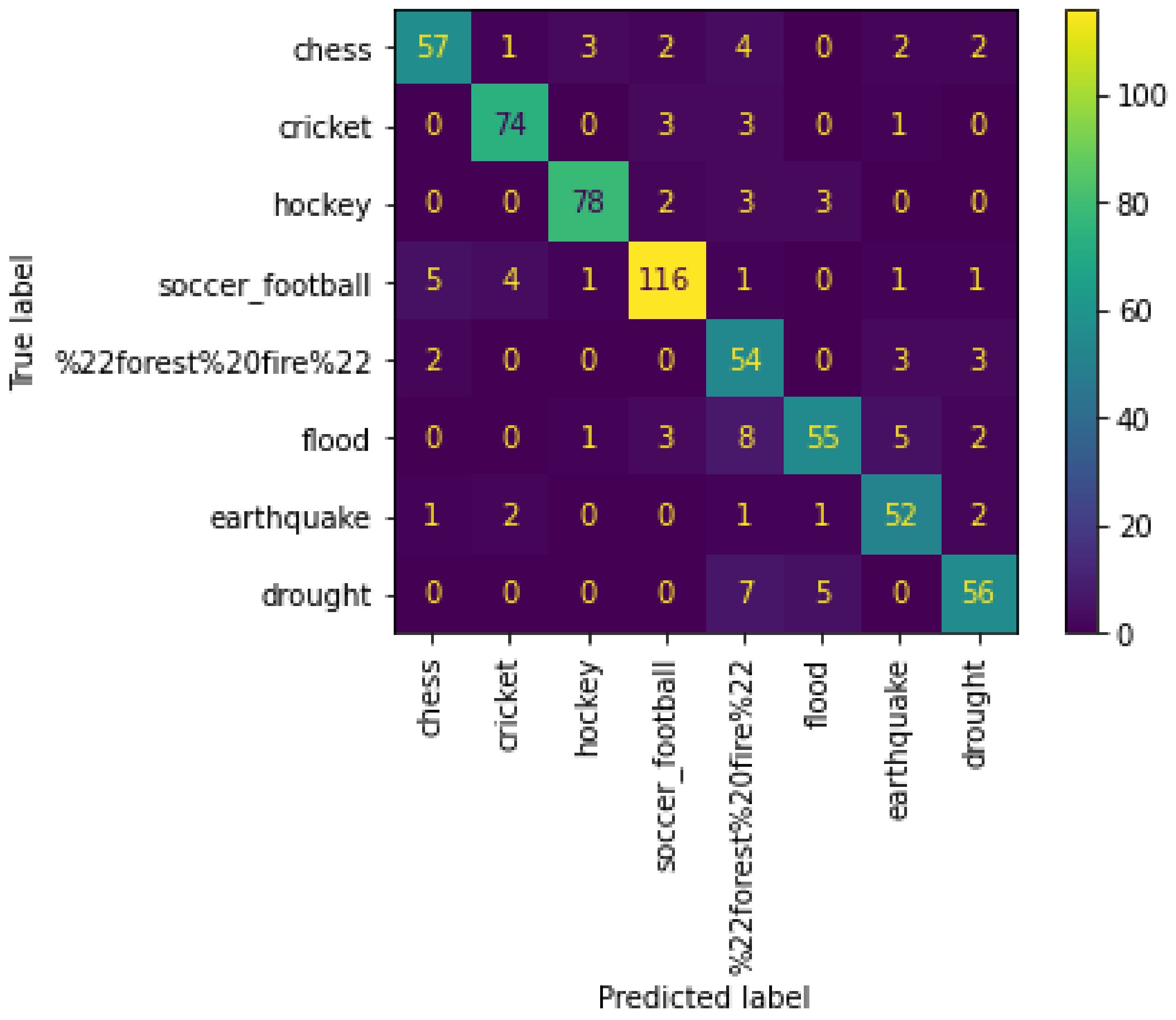
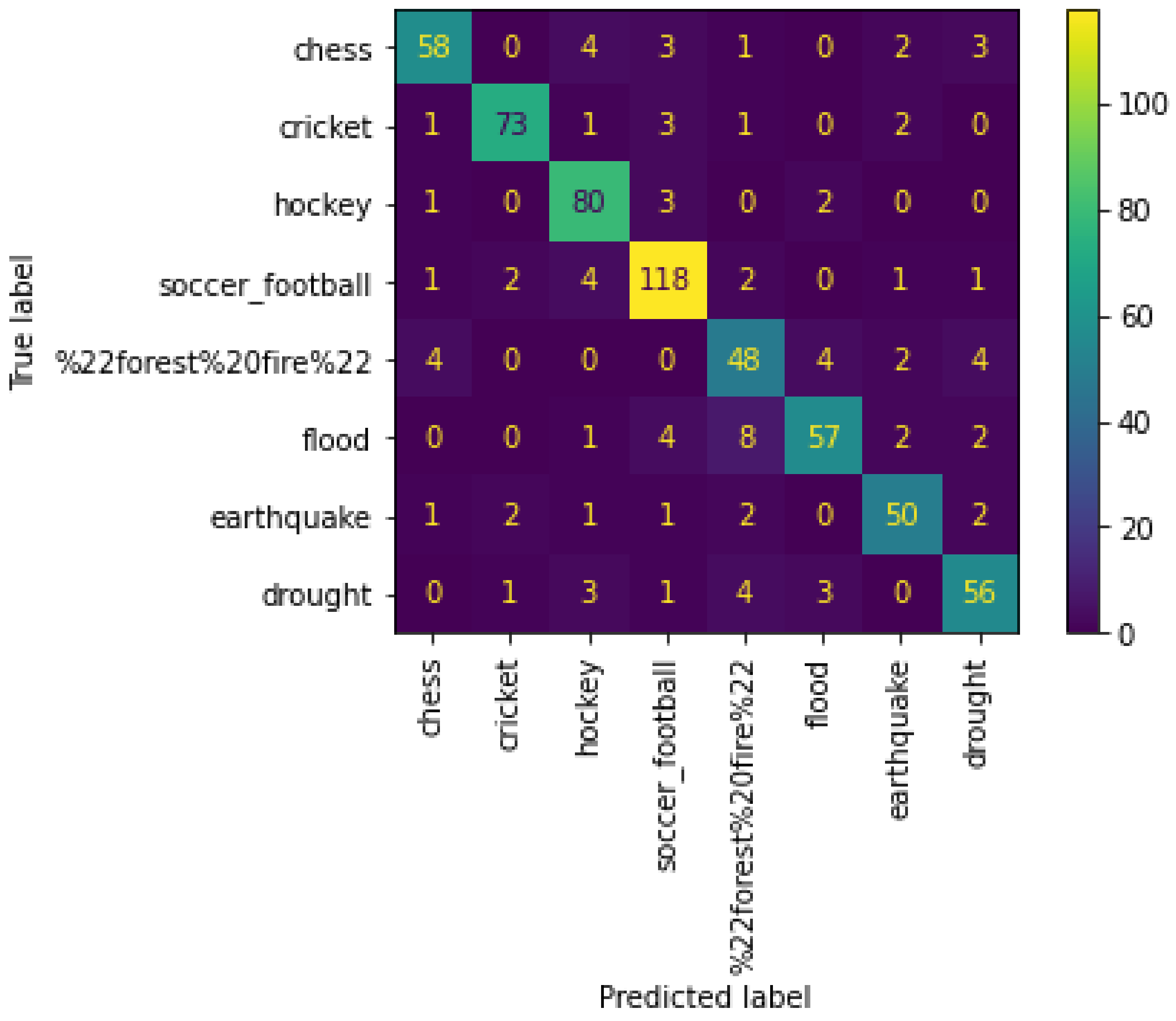


Figure 28: Confusion matrix for the SVM OneVsOne with class balance after merge

Results for SVM OneVsRest with class imbalance after merge:

Accuracy: 0.8571428571428571
Precision: 0.8518579149222214
Recall: 0.8473185092753417
F-1 Score: 0.848734915376803
Confusion Matrix:

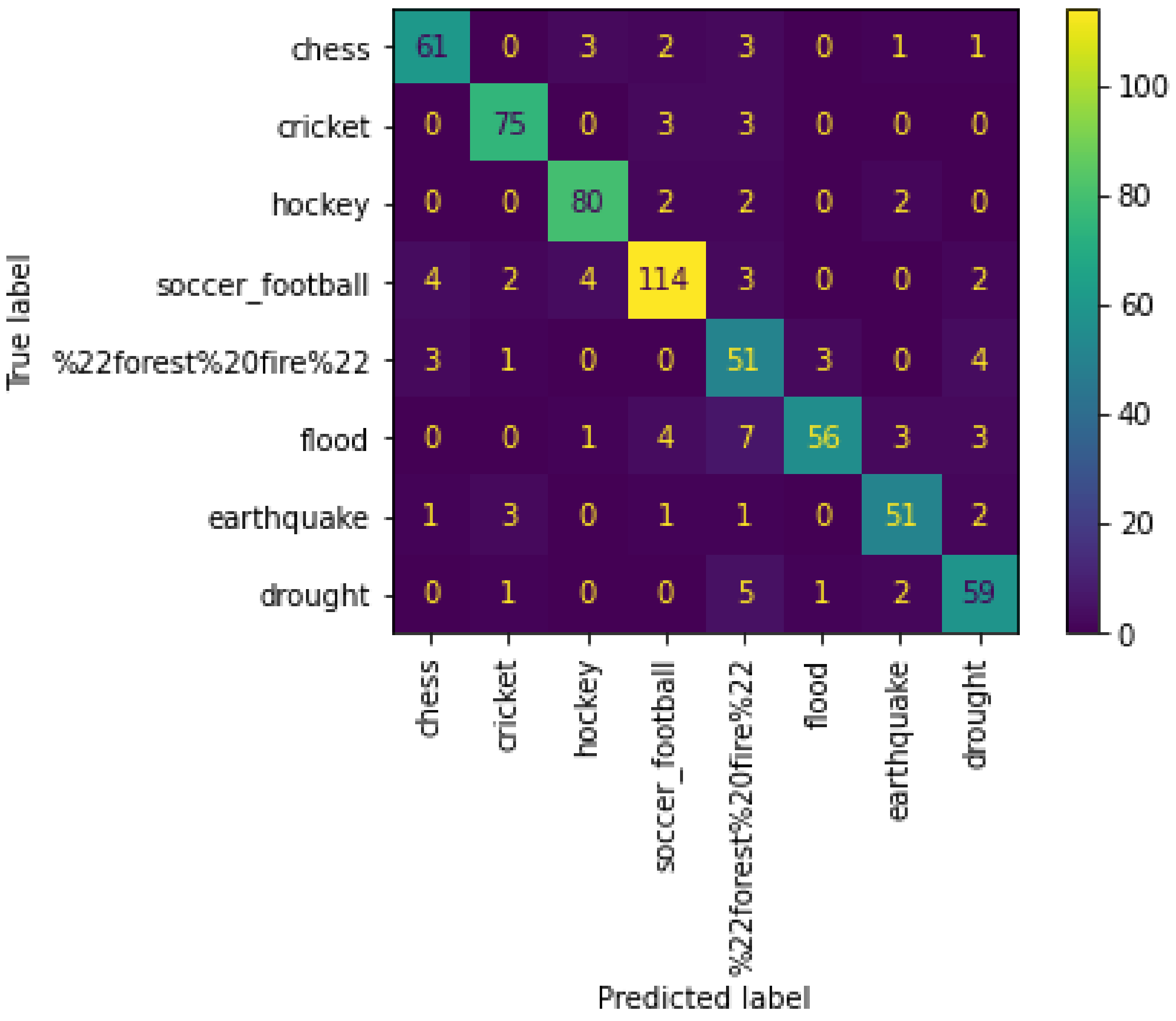


True \ Predicted	chess	cricket	hockey	soccer_football	%22forest%20fire%22	flood	earthquake	drought
chess	58	0	4	3	1	0	2	3
cricket	1	73	1	3	1	0	2	0
hockey	1	0	80	3	0	2	0	0
soccer_football	1	2	4	118	2	0	1	1
%22forest%20fire%22	4	0	0	0	48	4	2	4
flood	0	0	1	4	8	57	2	2
earthquake	1	2	1	1	2	0	50	2
drought	0	1	3	1	4	3	0	56

Figure 29: Confusion matrix for the SVM OneVsRest with class imbalance after merge

Results for SVM OneVsRest with class balance after merge

Accuracy: 0.8682539682539683
Precision: 0.8651588699620099
Recall: 0.8638031980347597
F-1 Score: 0.862375741635434
Confusion Matrix:



	chess	cricket	hockey	soccer_football	%22forest%20fire%22	flood	earthquake	drought
chess	61	0	3	2	3	0	1	1
cricket	0	75	0	3	3	0	0	0
hockey	0	0	80	2	2	0	2	0
soccer_football	4	2	4	114	3	0	0	2
%22forest%20fire%22	3	1	0	0	51	3	0	4
flood	0	0	1	4	7	56	3	3
earthquake	1	3	0	1	1	0	51	2
drought	0	1	0	0	5	1	2	59

Figure 30: Confusion matrix for the SVM OneVsRest with class balance after merge

Interpretation of figures and answers to Question 9:

Accuracy after merge has significantly increased by 8-12% for each model. This was expected since the most of the error were caused by the merged classes.

All results can be summarized below table:

Model(Merge = M)	Accuracy	Precision	Recall	F1 score
GaussianNB	0.6190	0.6311	0.6089	0.6030
OvO SVM	0.7380	0.7310	0.7281	0.7265
OVR SVM	0.7476	0.7436	0.7383	0.7364
OvO SVM, balanced	0.7396	0.7307	0.7295	0.7279
OvR SVM, balanced	**0.7698**	**0.7541**	**0.7615**	**0.7491**
GaussianNB(M)	0.6793	0.7368	0.6984	0.6965
OvO SVM(M)	0.8619	0.8592	0.8567	0.8544
OVR SVM(M)	0.8571	0.8518	0.8473	0.8487
OvO SVM, (M)balanced	0.8603	0.8547	0.8552	0.8522
OvR SVM,(M) balanced	**0.8682**	**0.8651**	**0.8638**	**0.8623**

Table 2: Summary of the results obtained for Question 9

We have observed that the best model is OvR SVM with balanced weighing before or after merging. Class imbalance slightly effects the model performance in the case of OneVsRest classifier. Class balancing increases accuracy by 1-2%. Also, class balancing does not have much effect when the model is OvO SVM.

Word Embedding:

Question 10:

a) The co-occurences are critical because the marginal probability distribution is distributed across all the classes and co-occurences reveal more informative results because the word to word relations are better captured in the co-occurences.

b) The “running” word would return the same vector. It will have high weights for both of the contexts (i.e. Running as physical movement and running for some competition).

c) The underlying concept between ||“king” - “queen”|| will be gender related concept and ||”husband” - “wife”|| will also capture the same concept therefore they are roughly equal. || `GLoVE["queen"] - GLoVE["king"] - GLoVE["wife"] + GLoVE["husband"]` ||2, will be roughly equal to zero.

d) Lemmatizing is more useful than stemming since stemming might produce non-words.

Question 11:

a) Feature engineering process firstly involves conversion of the embeddings to the same length array * 300 Glove embeddings. While keeping the embeddings size same for each keyword, it takes the average of all the related Glove embeddings.

b) A classifier model is chosen as Logistic Regression model. Maximum iteration number was chosen as 400. The penalty was determined as L2. The receiver operation characteristics were provided below:

Classifier	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.95	0.951	0.961	0.956

Table 3: Receiver Operating Characteristics of Logistic Regression Classifier

Question 12:

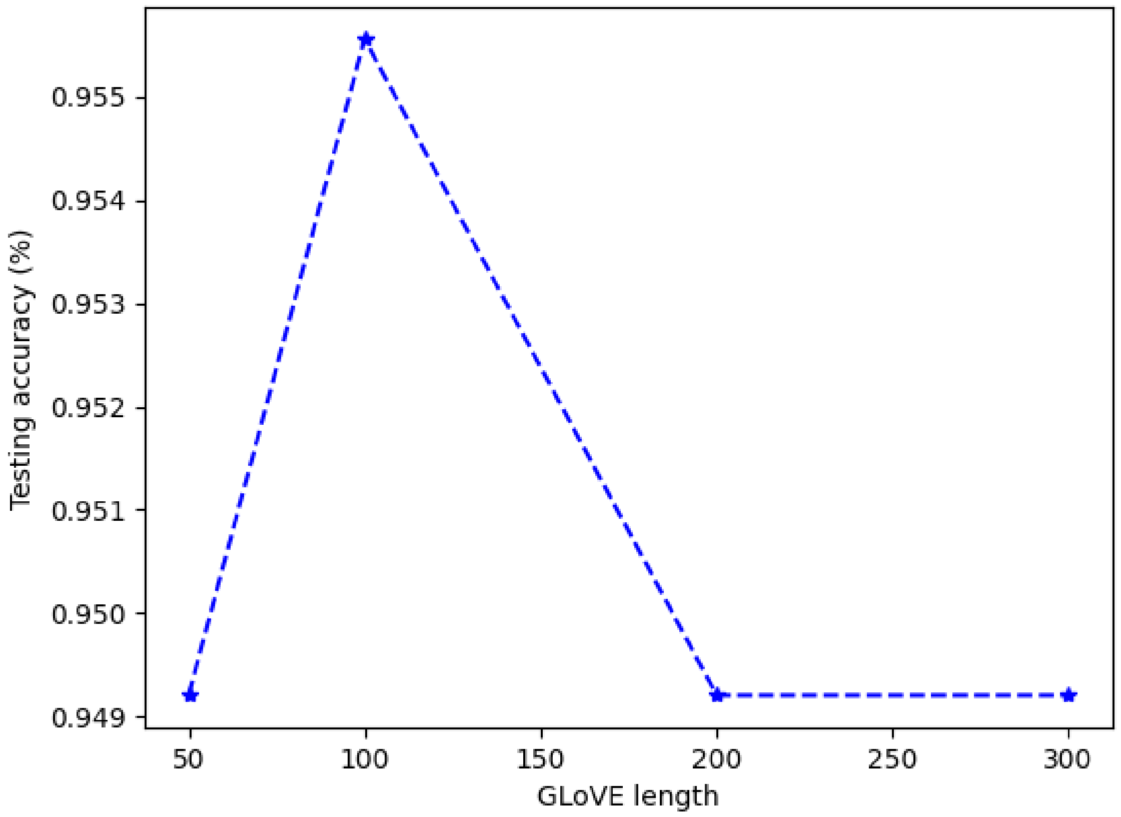


Figure 31: The accuracy vs Glove Length Graph

As shown in Fig. 31, the increase in the Glove length has a little impact on the accuracy when it is increased from 50 to 100. The reason is that the feature engineering process takes the average of the Glove embeddings and thus it is highly dominated by the most related words which can be covered only by 50 to 100 keywords. On the other hand, increasing the Glove length more than 100 have no improvements since the first 100 words are already enough for logistic regression classifier for this specific classification problem.

Question 13:

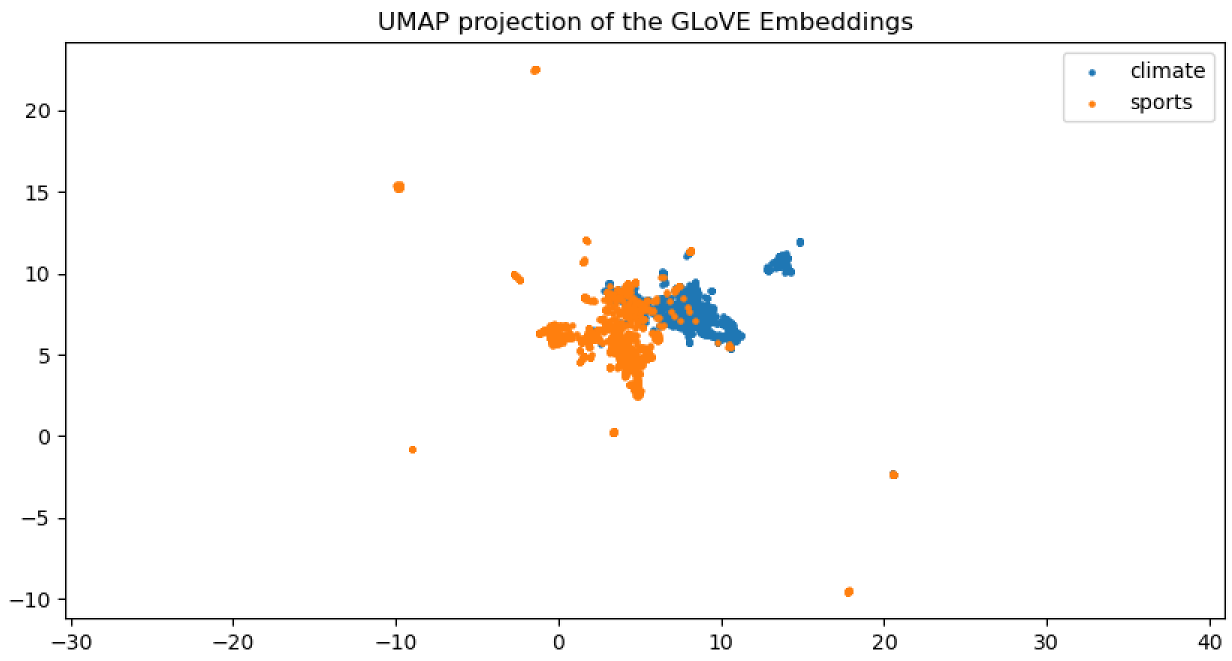


Figure 32: UMAP Projection of the GLoVE Embeddings

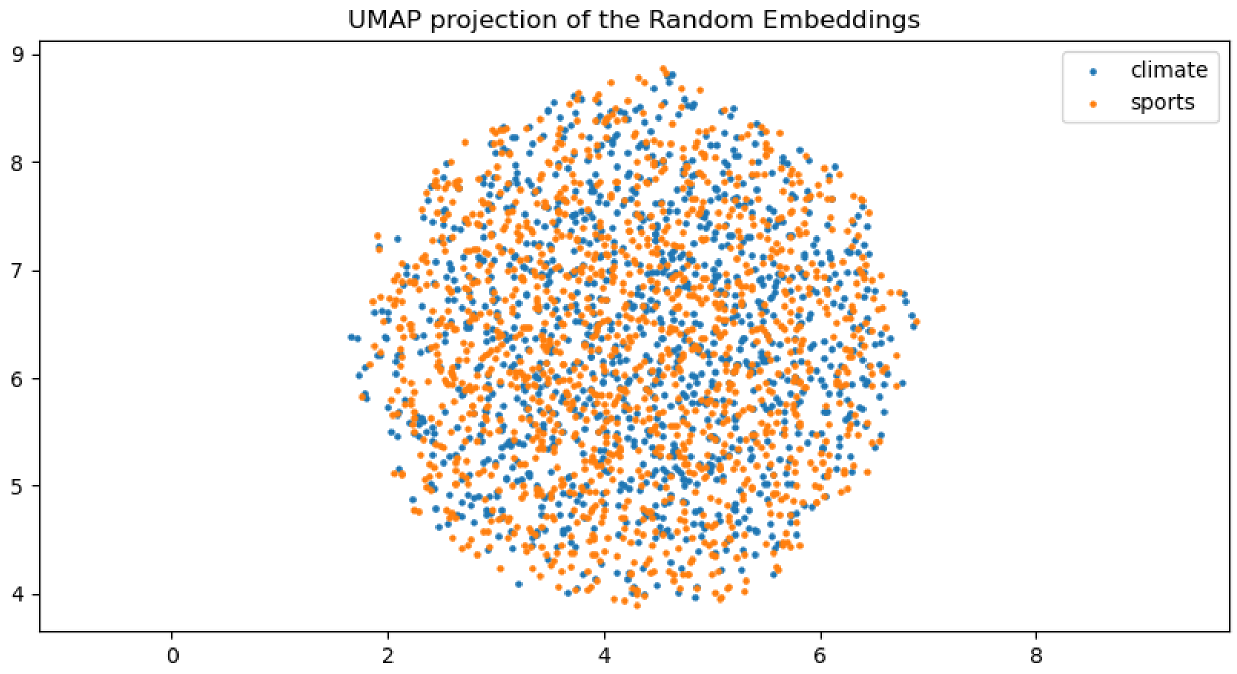


Figure 33: UMAP Projection of the Random Embeddings

When the random and Glove embeddings are compared it can be clearly seen that the same class related words forms a cluster on the other hand the random vectors have randomly reduced dimensions with UMAP and thus they don't reflect any cluster type shape or any correlation. The UMAP measures one sample’s probability distance to the multinomial probability distribution. Usually Hellinger’s distance is preferred as a probabilistic metric. As a result, the same class word embeddings have relatively similar probability distribution and thus they are close to each other in 2D plane.