COL 774: Machine Learning
Assignment 2

Name: Suraj Reddy
Entry Number: 2025AIY7586

1 Text Classification using Naive Bayes

1.1 (a)Implement Naive Bayes Multiclass Classification:

Implement algorithm to classify each sample into one of the given 14 categories.

$$P(\text{document} \mid \text{class}) = \prod_{i=1}^{N} P(\text{word}_i \mid \text{class})$$

$$P(\text{class} \mid \text{document}) \propto P(\text{class}) \prod_{i=1}^{N} P(\text{word}_i \mid \text{class})$$

Each word is generated independently from a single (fixed) Multinoulli distribution $P(\text{word} \mid \text{class})$.

- Implement the Naive Bayes Classifier using only the **content** text.
- With Laplace smoothing set to 1.

Data	Accuracy	nof samples
Train set	0.9753	140000
Test set	0.9592	35000

Table 1: Train and test accuracy of Naive Bayes classifier

(b) Construct a word cloud representing the most frequent words for each class:

Most frequent images of class 1

Most frequent images of class 2

Most frequent images of class 3

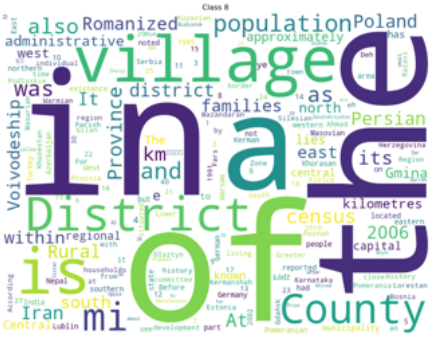
Most frequent images of class 4

Most frequent images of class 5

Most frequent images of class 6

Most frequent images of class 7

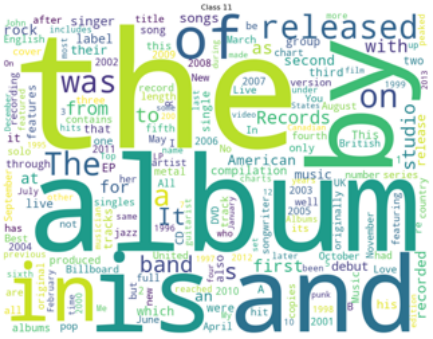
Most frequent images of class 8



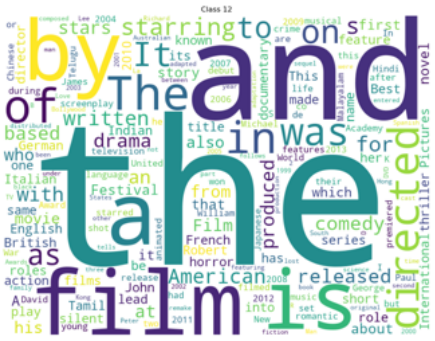
Most frequent images of class 9

Most frequent images of class 10

Most frequent images of class 11



Most frequent images of class 12



Most frequent images of class 13

Most frequent images of class 14

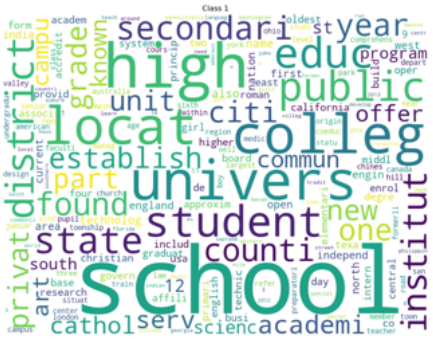
1.2 (a) Text Classification using Naive Bayes with stemming and removing the stop-words.

Implement Naive Bayes Multiclass Classification: Implement algorithm to classify each sample into one of the given 14 categories with stemming and removing the stop-words.

- Implement the Naive Bayes Classifier using only the **content** text.
- With Laplace smoothing set to 1.
- Implement stemming and remove the stop-words.

(b) Construct a word cloud representing the most frequent words for each class:

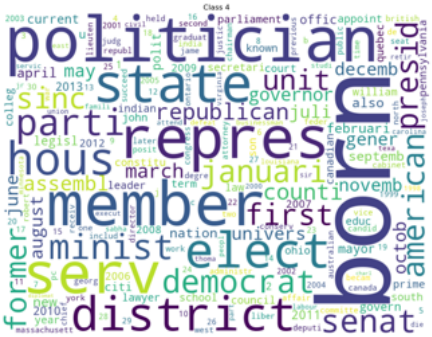
Most frequent images of class 1



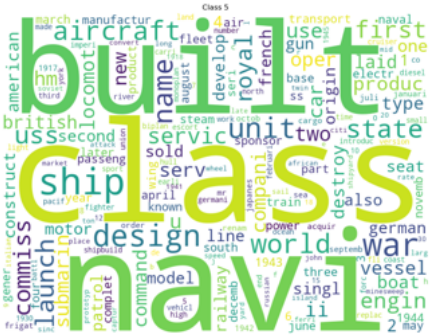
Most frequent images of class 2

Most frequent images of class 3

Most frequent images of class 4

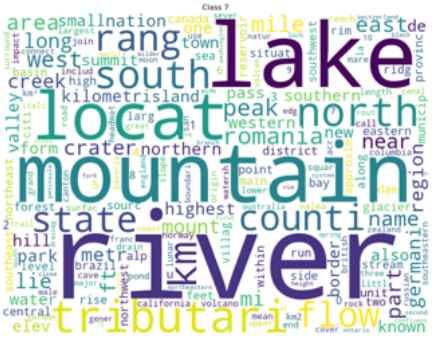


Most frequent images of class 5



Most frequent images of class 6

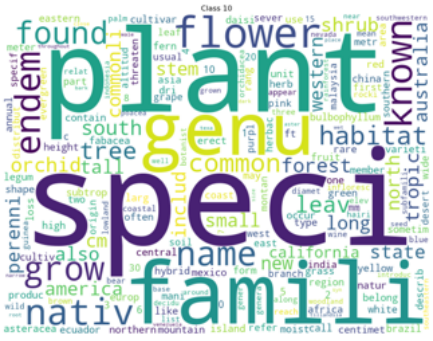
Most frequent images of class 7



Most frequent images of class 8

Most frequent images of class 9

Most frequent images of class 10



Most frequent images of class 11

Most frequent images of class 12

Most frequent images of class 13

Most frequent images of class 14

(c) Learn a new model on the transformed data. Report the test set accuracy.

Data	Accuracy	nof samples
Train set	0.9691	140000
Test set	0.9529	35000

Table 2: Train and test accuracy of Naive Bayes classifier with smoothing and removing stop words.

(d) How does your accuracy change over the validation set? Comment on your observations.

- After smoothing and removing stop words, the test accuracy decreases.
- Number of words in the vocabulary decreases, which effects the training

1.3 (a) Text Classification using Naive Bayes with stemming and removing the stop-words and Bigrams.

Implement Naive Bayes Multiclass Classification: Implement algorithm to classify each sample into one of the given 14 categories with stemming and removing the stop-words and Train a model that utilizes both unigrams (individual words) and bigrams as features.

- Implement the Naive Bayes Classifier using only the **content** text.
- With Laplace smoothing set to 1.
- Implement stemming and remove the stop-words.
- Train a model that utilizes both unigrams and bigrams as features.

(b) Learn a new model on the transformed data. Report the test set accuracy:

Data	**#raw**	**#unigrams**	**#bigrams**
Train set	0.9753	0.9691	0.9919
Test set	0.9592	0.9529	0.9648

Table 3: Train and test acc of NBC with raw, unigram and bigrams.

1.4 Analysis

Analyze the performance of different models to identify which one works best for classifying based on the content text. Implement performance metrics such as accuracy, precision, recall, F1-score.

Table 4: Aggregated Performance Metrics for Raw, Unigram, Bigram, and Bigram (No Stemming) Models

Metric	Raw Model	Unigram Model	Bigram Model	Bigram Model(No Stem)	only Bigram Model(No Stem)
Accuracy	0.9592	0.9592	0.9647	0.9678	0.9677
Mean Precision	0.9587	0.9587	0.9638	0.9638	0.9686
Mean Recall	0.9627	0.9627	0.9721	0.9731	0.9702
Mean F1-score	0.9591	0.9591	0.9680	0.9700	0.9690

Among all Naive Bayes Classifier (NBC) variants, the bigram-based model achieved the best performance with a test accuracy of 0.9648 and an F1-score of 0.96 (Table **??**). This shows that using bigrams helps capture contextual relationships between words, improving classification accuracy. The unigram and raw text models performed slightly worse, indicating that single-word features and unprocessed text lack sufficient contextual information. Hence, the bigram model was selected as it provides the best balance between accuracy, precision, recall, and F1-score.

Table 5: Per-Class Precision, Recall, and F1-score for Raw, Unigram, Bigram, and Bigram (No Stemming) Models

Class	Raw Model			Unigram Model			Bigram Model			Bigram Model (No Stemming)		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
1	0.9378	0.8688	0.9020	0.9378	0.8688	0.9020	0.9439	0.8760	0.9087	0.9439	0.8760	0.9087
2	0.9530	0.9808	0.9667	0.9530	0.9808	0.9667	0.9562	0.9860	0.9709	0.9562	0.9860	0.9709
3	0.9573	0.8960	0.9256	0.9573	0.8960	0.9256	0.9639	0.8988	0.9302	0.9639	0.8988	0.9302
4	0.9821	0.9896	0.9859	0.9821	0.9896	0.9859	0.9826	0.9912	0.9869	0.9826	0.9912	0.9869
5	0.9642	0.9804	0.9722	0.9642	0.9804	0.9722	0.9706	0.9784	0.9745	0.9706	0.9784	0.9745
6	0.9634	0.9892	0.9761	0.9634	0.9892	0.9761	0.9637	0.9876	0.9755	0.9637	0.9876	0.9755
7	0.9467	0.9512	0.9489	0.9467	0.9512	0.9489	0.9543	0.9616	0.9580	0.9543	0.9616	0.9580
8	0.9595	0.9852	0.9722	0.9595	0.9852	0.9722	0.9763	0.9872	0.9817	0.9763	0.9872	0.9817
9	0.9992	0.9436	0.9706	0.9992	0.9436	0.9706	0.9992	0.9620	0.9802	0.9992	0.9620	0.9802
10	0.9933	0.9552	0.9739	0.9933	0.9552	0.9739	0.9963	0.9616	0.9786	0.9963	0.9616	0.9786
11	0.9732	0.9868	0.9799	0.9732	0.9868	0.9799	0.9751	0.9888	0.9819	0.9751	0.9888	0.9819
12	0.9388	0.9932	0.9652	0.9388	0.9932	0.9652	0.9398	0.9936	0.9660	0.9398	0.9936	0.9660
13	0.9546	0.9676	0.9611	0.9546	0.9676	0.9611	0.9631	0.9824	0.9727	0.9631	0.9824	0.9727
14	0.9103	0.9412	0.9255	0.9103	0.9412	0.9255	0.9234	0.9500	0.9365	0.9234	0.9500	0.9365

1.5 Implement Naive Bayes Multiclass Classification for Title Features.

(a) Learn model on title column data: Reporting the train and test set accuracy for 3 different models:

- RAW: Training model on raw title column data.
- UNIGRAM: Training model on title column with stemming and stop-words.
- BIGRAM: Training model on title column with stemming and stop-words and Bigram.

Data	#raw	#unigrams	#bigrams
Train set	0.8950	0.8795	0.9480
Test set	0.6601	0.6532	0.6542

Table 6: Train and test acc of NBC with raw, unigram and bigrams for title column data.

(b) Construct a word cloud representing the most frequent words for each class:

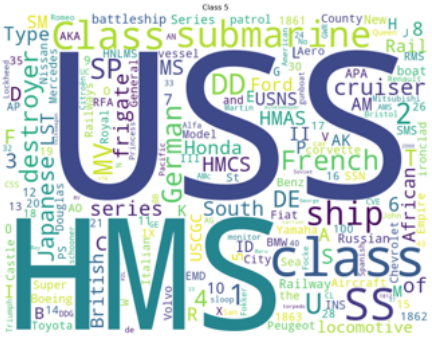
Most frequent images of class 1

Most frequent images of class 2

Most frequent images of class 3

Most frequent images of class 4

Most frequent images of class 5



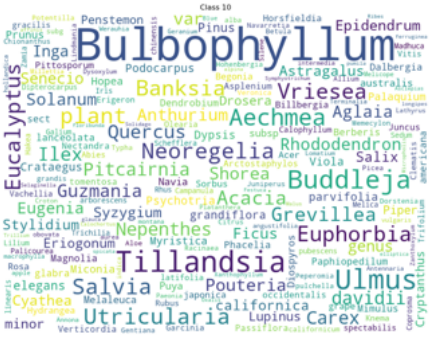
Most frequent images of class 6

Most frequent images of class 7

Most frequent images of class 8

Most frequent images of class 9

Most frequent images of class 10



Most frequent images of class 11

Most frequent images of class 12

Most frequent images of class 13

Most frequent images of class 14

(c) Construct a word cloud representing the most frequent words for each class with stemming and stop-words:

Most frequent images of class 1

Most frequent images of class 2

Most frequent images of class 3

Most frequent images of class 4

Most frequent images of class 5

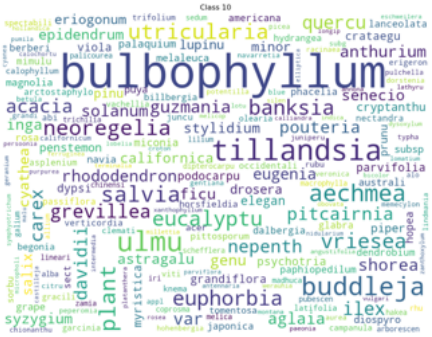
Most frequent images of class 6

Most frequent images of class 7

Most frequent images of class 8

Most frequent images of class 9

Most frequent images of class 10



Most frequent images of class 11

Most frequent images of class 12

Most frequent images of class 13

Most frequent images of class 14

(d)Analysis: Analyze the performance of different models to identify which one works best for classifying based on the content text. Implement performance metrics such as accuracy, precision, recall, F1-score.

Class	Raw				Unigram				Bigram			
	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
1	0.6601	0.2939	0.7296	0.4190	0.6532	0.2907	0.6968	0.4103	0.6542	0.2926	0.6980	0.4124
2	0.6601	0.8297	0.9668	0.8930	0.6532	0.8280	0.9684	0.8927	0.6542	0.8305	0.9704	0.8950
3	0.6601	0.4740	0.3320	0.3905	0.6532	0.4680	0.3340	0.3898	0.6542	0.4669	0.3300	0.3867
4	0.6601	0.5491	0.4828	0.5138	0.6532	0.5442	0.4752	0.5074	0.6542	0.5465	0.4752	0.5083
5	0.6601	0.5143	0.5384	0.5261	0.6532	0.5152	0.5436	0.5290	0.6542	0.5136	0.5368	0.5249
6	0.6601	0.8884	0.8848	0.8866	0.6532	0.8682	0.8696	0.8689	0.6542	0.8631	0.8752	0.8691
7	0.6601	0.7780	0.8720	0.8223	0.6532	0.7657	0.8708	0.8149	0.6542	0.7677	0.8776	0.8190
8	0.6601	0.8980	0.7676	0.8277	0.6532	0.8896	0.7732	0.8273	0.6542	0.8928	0.7760	0.8303
9	0.6601	0.9231	0.5616	0.6983	0.6532	0.9128	0.5572	0.6920	0.6542	0.9137	0.5592	0.6938
10	0.6601	0.9660	0.4996	0.6586	0.6532	0.9380	0.4844	0.6389	0.6542	0.9377	0.4876	0.6416
11	0.6601	0.9403	0.7492	0.8339	0.6532	0.9227	0.7496	0.8272	0.6542	0.9235	0.7484	0.8268
12	0.6601	0.6536	0.7056	0.6786	0.6532	0.6347	0.7004	0.6659	0.6542	0.6360	0.7004	0.6667
13	0.6601	0.6667	0.5376	0.5952	0.6532	0.6460	0.5460	0.5918	0.6542	0.6407	0.5556	0.5951
14	0.6601	0.6495	0.6144	0.6314	0.6532	0.6431	0.5760	0.6077	0.6542	0.6399	0.5680	0.6018

Table 7: Per-class accuracy, precision, recall, and F1-score for Naive Bayes using raw text, unigram, and bigram models.

Based on the per-class performance metrics shown in Table 7, the Bigram model consistently outperforms the Raw and Unigram models across most classes. It achieves slightly higher precision, recall, and F1-scores, indicating better contextual understanding by capturing word pairs rather than isolated words. Although the overall accuracy improvement is modest, the Bigram model demonstrates more stable and balanced performance across all classes, making it the most effective choice for content-based text classification.
When comparing the best-performing models from both feature sets, the accuracy obtained using the title features (0.6542 with bigrams) is notably lower than that achieved using the content features (0.9648 with bigrams). This indicates that the content text provides richer and more discriminative information for classification than titles, which are typically shorter and contain fewer contextual cues. Hence, models trained on full content text are more effective for accurate classification.

1.6 (a) Text Classification using Naive Bayes with both features title and content together.

Implement Naive Bayes Multiclass Classification: Implement NBC to classify each sample into one of the given 14 categories with a model that utilizes both title and content features. Logic setup:

- No stemming and no stop-words removal.
- Concatenating title and content to get single Tokenized Description.
- Using both unigrams and bigrams.

Results are as shown in table 8:

Data	Accuracy	nof samples
Train set	0.9923	140000
Test set	0.9687	35000

Table 8: Train and test accuracy of Naive Bayes classifier

Comparision with previous models:

Feature Set	Train Acc.	Test Acc.	Observation
Content (Unigram)	0.9753	0.9592	Good baseline
Content (Unigram + st-sw)	0.9691	0.9529	Slightly lower due to preprocessing
Content (Unigram + st-sw + Bigram)	0.9919	0.9648	Strong contextual learning
Title (Unigram)	0.8950	0.6601	Weak, limited info
Title (Unigram + st-sw)	0.8795	0.6532	Slightly worse with preprocessing
Title (Unigram + st-sw + Bigram)	0.9480	0.6542	Minor bigram gain
Title + Content (Unigram + Bigram)	**0.9923**	**0.9687**	Best — uses both context and summary

Table 9: Comparison of Naive Bayes models using title, content, and combined features.

We extend the Multinomial Naive Bayes model by learning separate conditional multinomial parameters θ_y^{title} and $\theta_y^{\text{content}}$ for each class y. With Laplace smoothing (α), the MAP estimates are

$$\hat{\theta}_{y,v}^{\text{title}} = \frac{N_{y,v}^{\text{title}} + \alpha}{N_y^{\text{title}} + \alpha V}, \qquad \hat{\theta}_{y,v}^{\text{content}} = \frac{N_{y,v}^{\text{content}} + \alpha}{N_y^{\text{content}} + \alpha V}.$$

At prediction time we combine the title and content log-likelihoods (optionally with weights $\lambda_{\text{title}}, \lambda_{\text{content}}$) and choose the class with maximum posterior score. This separate-parameter approach captures different language patterns in titles and contents and allows flexible weighting; empirically it often improves test accuracy over both single-source and concatenated-text models.

The model with separate parameters for title and content achieves the highest test accuracy (0.9703), outperforming both the single-feature and concatenated models. This shows that learning distinct distributions for title and content improves classification performance.

Data	Accuracy	nof samples
Train set	0.9923	140000
Test set	0.9703	35000

Table 10: Train and test accuracy of NBC with 2 features.

Model	Train Accuracy	Test Accuracy
Title only	0.9842	0.6601
Content only	0.9918	0.9648
Concatenated (joint θ)	0.9923	0.9687
Separate $\theta^{(title)}$, $\theta^{(content)}$	0.9923	**0.9703**

Table 11: Comparison of Naive Bayes models using different feature representations.

1.7 Analysis

(a) What is the validation set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the articles (random prediction)? (b) What accuracy would you obtain if you simply predicted each sample as positive? (c) How much improvement does your algorithm give over the random/positive baseline?

Model / Baseline	**Validation Accuracy**
Random Prediction	0.0695
Always Predict One Class (e.g., Class 0)	0.0714
Naive Bayes (best model)	**0.9703**

Table 12: Comparison of baseline accuracies with the best Naive Bayes model.

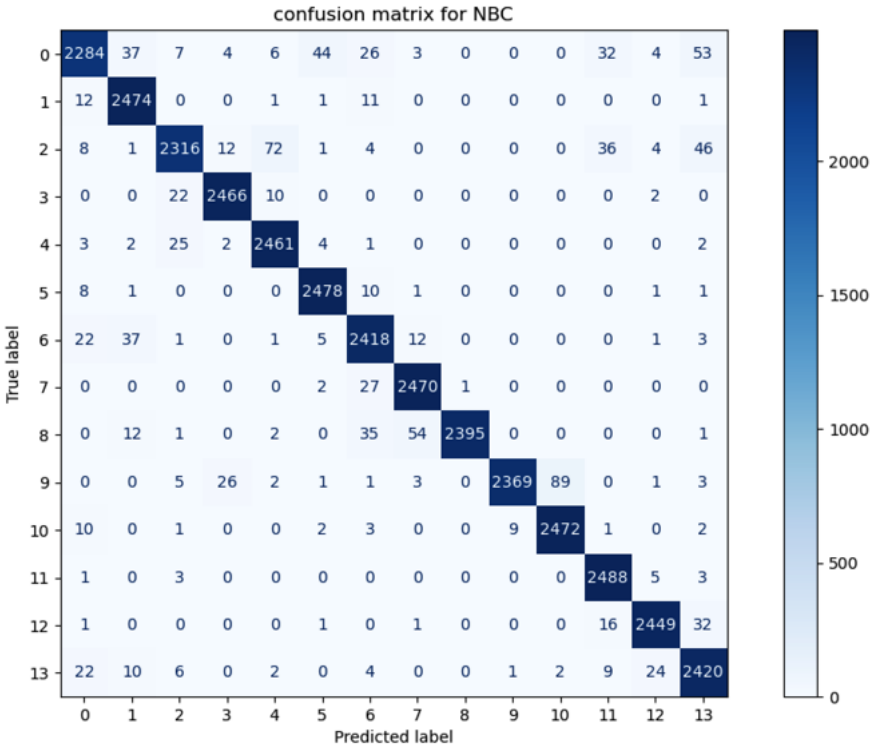
(a) With 14 categories, random guessing yields an expected accuracy of 0.0695.

(b) Predicting all samples as a single (most frequent) class gives around 0.0714 accuracy.

(c) The Naive Bayes model achieves 0.9703, which is an improvement of roughly +92.8% over the random baseline and +92.6% over the positive-class baseline, demonstrating a significant performance gain.

1.8 Plotting confusion matrix

Plotting confusion matrix for my best performing model. That is, claculating theta for title and content features and estimating likelihoods with unigram, bigrams, smoothing and stop-words. From the confusion matrix (Figure **??**),



Confusion matrix for bigram model.

the diagonal entries represent the number of correctly classified samples for each category. Table 13 shows the diagonal values for all 14 categories.
The highest diagonal entry is observed for **Class 11 (2488 samples)**, indicating that the model performs best on this category. This suggests that the textual features for Class 11 are more distinctive and less prone to overlap with other classes, leading to more accurate classification.

Class Index	Correct Predictions (Diagonal Value)
0	2284
1	2474
2	2316
3	2466
4	2461
5	2478
6	2418
7	2470
8	2395
9	2369
10	2472
11	**2488**
12	2449
13	2420

Table 13: Diagonal values from the confusion matrix representing correct classifications per class.

1.9 Feature Engineering and Model Enhancement

To further improve classification performance, we introduced an additional feature based on the **average document length** (number of tokens) for each article. This feature captures structural information, as certain categories (e.g., politics, technology) tend to have longer articles than others.

The Naive Bayes model was retrained by incorporating document length as a prior adjustment to class probabilities.

Table 14: Performance of Naive Bayes Classifier with Title, Content, and Average Document Length Features

Class	**Precision**	**Recall**	**F1-Score**
Class 0	0.9605	0.9132	0.9362
Class 1	0.9641	0.9888	0.9763
Class 2	0.9728	0.9308	0.9513
Class 3	0.9884	0.9904	0.9894
Class 4	0.9655	0.9840	0.9746
Class 5	0.9802	0.9912	0.9857
Class 6	0.9608	0.9616	0.9612
Class 7	0.9714	0.9920	0.9816
Class 8	0.9996	0.9632	0.9811
Class 9	0.9967	0.9788	0.9877
Class 10	0.9853	0.9912	0.9882
Class 11	0.9599	0.9956	0.9774
Class 12	0.9757	0.9808	0.9783
Class 13	0.9438	0.9612	0.9524
Macro Avg.	**0.9722**	**0.9773**	**0.9748**
Training Accuracy		0.9987	
Test Accuracy		0.9731	

The Naive Bayes classifier trained with title, content, and average document length features achieved a high training accuracy of **99.87%** and a test accuracy of **97.31%**. Performance across all 14 classes remained consistent, with an average precision of **0.9722**, recall of **0.9773**, and F1-score of **0.9748**. This demonstrates that integrating lexical and structural document features significantly enhances classification effectiveness.

Model / Baseline	**Validation Accuracy**
Random Prediction	0.0695
Always Predict One Class (e.g., Class 0)	0.0714
Naive Bayes (Previous Best Model)	0.9703
Naive Bayes (Title + Content + Avg. Doc Length)	**0.9731**

Table 15: Comparison of baseline accuracies with improved Naive Bayes models. The addition of average document length as a feature led to the highest accuracy.

COL 774: Machine Learning Assignment 2

Name: Suraj Reddy
Entry Number: 2025AIY7586

2. Image Classification using SVM

1 Using Support Vector Machines (SVMs) to build a binary image classifier using CVXOPT with linear Kernel.

SVM Dual Problem (Linear Kernel, Soft Margin)
Given training data $\{(x_i, y_i)\}_{i=1}^m$ with $y_i \in \{-1, 1\}$, the dual optimization problem is:

$$\begin{aligned} \text{maximize}_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m, \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

In standard quadratic programming form for CVXOPT:

$$\begin{aligned} \text{minimize}_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \boldsymbol{\alpha}^\top P \boldsymbol{\alpha} + q^\top \boldsymbol{\alpha} \\ \text{subject to} \quad & G\boldsymbol{\alpha} \leq h, \\ & A\boldsymbol{\alpha} = b \end{aligned}$$

Where:

$$\begin{aligned} & P_{ij} = y_i y_j \langle x_i, x_j \rangle, \quad q_i = -1, \\ & G = \begin{bmatrix} -I \\ I \end{bmatrix}, \quad h = \begin{bmatrix} 0 \\ C \cdot \mathbf{1} \end{bmatrix}, \\ & A = y^\top, \quad b = 0 \end{aligned}$$

Here, $C = 1.0$ is the regularization parameter for soft-margin SVM.
Results:

- My classes for binary Classification are :frog and horse.

number of support vectors	1274
Test accuracy	0.7898
total execution time	18.6249

Table 1: Train and test accuracy of Naive Bayes classifier

Plot top 5 support vectors

(a) Image 1

(b) Image 2

(c) Image 3

(d) Image 4

(e) Image 5

Figure 1: Five top support vectors for SVM binary classification with CVXOPT

2 Using Support Vector Machines (SVMs) to build a binary image classifier using CVXOPT with gaussian Kernel.

Results:

- My classes for binary Classification are : frog and horse.

nof support vectors with linear Kernel	1274
Test accuracy of linear Kernel	0.7898
total execution time	18.6249
nof support vectors with gaussian Kernel	2058
Test accuracy of gaussian Kernel	0.8483
total execution time	40.7535
nof matching support vectors	1084

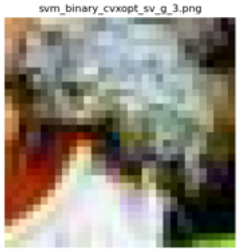
Table 2: Train and test accuracy of Naive Bayes classifier with linear and gaussian Kernel a

Plot top 5 support vectors

(a) Image 1

(b) Image 2

(c) Image 3



(d) Image 4

(e) Image 5

Figure 2: Five top support vectors for SVM binary classification with CVXOPT

3 Using Support Vector Machines (SVMs) to build a binary image classifier using LIBSVM with linear and gaussian Kernel.

Results:

model	test acc	nof sv	total execution time(s)
CVXOPT with linear Kernel	0.7898	1274	18.6249
CVXOPT with gaussian Kernel	0.8483	2058	40.7535
LIBSVM with linear Kernel	0.7898	1262	8.9805
LIBSVM with gaussian Kernel	0.8483	2029	7.5164

Table 3: comparision of CVXOPT and LIBSVM

Table 4: Number of Matching Support Vectors Across Models

Model	CX Linear	CX Gaussian	LB Linear	LB Gaussian
CX Linear	1274	1084	1262	1077
CX Gaussian	1084	2058	1074	2029
LB Linear	1262	1074	1262	1067
LB Gaussian	1077	2029	1067	2029

Table 5: Comparison of Weight and Bias for Linear Kernel SVM

SVM Model	Weight Vector (w)	Bias (b)	Remarks
Custom CVXOPT Linear	$[-0.0620, 0.0849, 0.1493, \ldots, 0.4715, 0.4211, 0.6112]$	-1.0221	–
Scikit-learn Linear	$[-0.0626, 0.0850, 0.1482, \ldots, 0.4717, 0.4208, 0.6109]$	-1.0279	–
Weight difference norm: 0.0296, **Cosine similarity:** 0.999998			

4 Multi-Class Image Classification

Implement Multi-class classification problem using Support Vector Machines (SVMs) with gaussian Kernel and CVXOPT.

One-vs-One Multi-Class SVM:

1. For each pair of classes (c_p, c_q), train a binary SVM using Gaussian kernel:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

   Solve the dual optimization problem:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \quad 0 \le \alpha_i \le C, \quad \sum_i \alpha_i y_i = 0$$

2. For a test sample, collect predictions from all $\binom{k}{2}$ classifiers.

3. Assign the test sample to the class with the highest number of votes:

$$\hat{y} = \arg \max_{c \in \{1,\ldots,k\}} \text{votes}(c)$$

Results:

Test accuracy	0.4381
total execution time	1707.6119

Table 6: test accuracy of Naive Bayes classifier

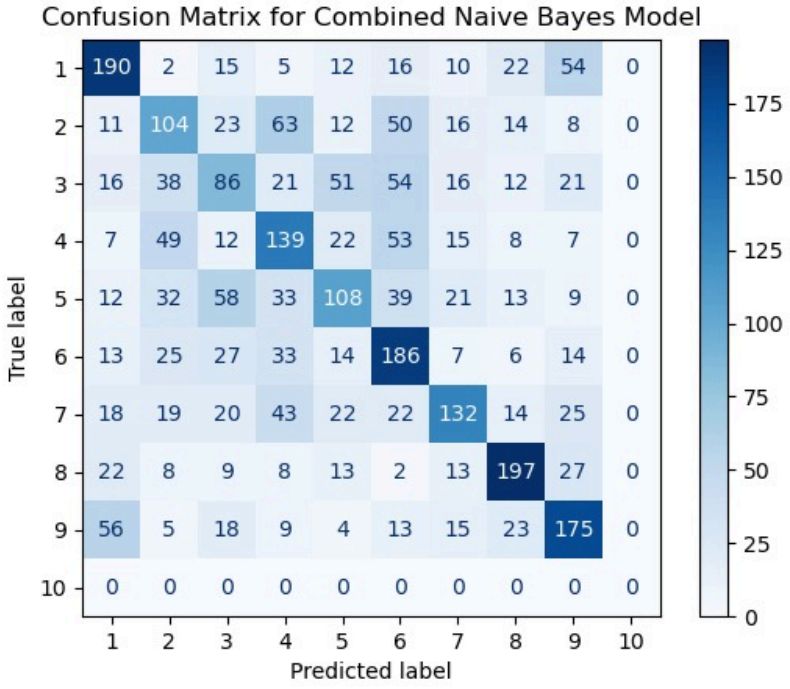


Figure 3: confusion matrix for CVXOPT multi class classifier

Observations on Confusion Matrices

- The LIBSVM OvO model demonstrates a clearer diagonal dominance in the confusion matrix, indicating stronger class-wise prediction accuracy compared to the Combined Naive Bayes Model.
- In the Naive Bayes model, significant misclassifications are observed among visually similar classes such as *cat*, *dog*, and *deer*, as well as between *airplane* and *ship*. This suggests that the probabilistic independence assumption limits its discriminative power on complex image features.
- The LIBSVM OvO model shows improved separability, particularly for classes like *automobile*, *ship*, and *truck*, where prediction counts along the diagonal are substantially higher.
- However, some degree of confusion remains between *cat* and *dog*, as well as between *horse* and *deer*, likely due to overlapping feature distributions in the dataset.
- Overall, the SVM model provides a more consistent and reliable performance across most classes, reflecting the advantage of margin-based learning over the conditional independence assumption of Naive Bayes.

Table 7: Most Frequent Misclassifications in the SVM Classifier

Misclassification	**Count**
airplane → ship	95
bird → deer	63
dog → cat	58
truck → automobile	56
automobile → truck	54
cat → frog	54
deer → frog	53
cat → dog	51
bird → frog	50
deer → bird	49

Figure 4: misclassified images using CVXOPT multi class classifier

5 Multi-Class Image Classification

Implement Multi-class classification problem using Support Vector Machines (SVMs) with gaussian Kernel and LIBSVM.

One-vs-One Multi-Class SVM: Results:

Test accuracy	0.4369
total execution time	1186.5481

Table 8: test accuracy of multi class Naive Bayes classifier

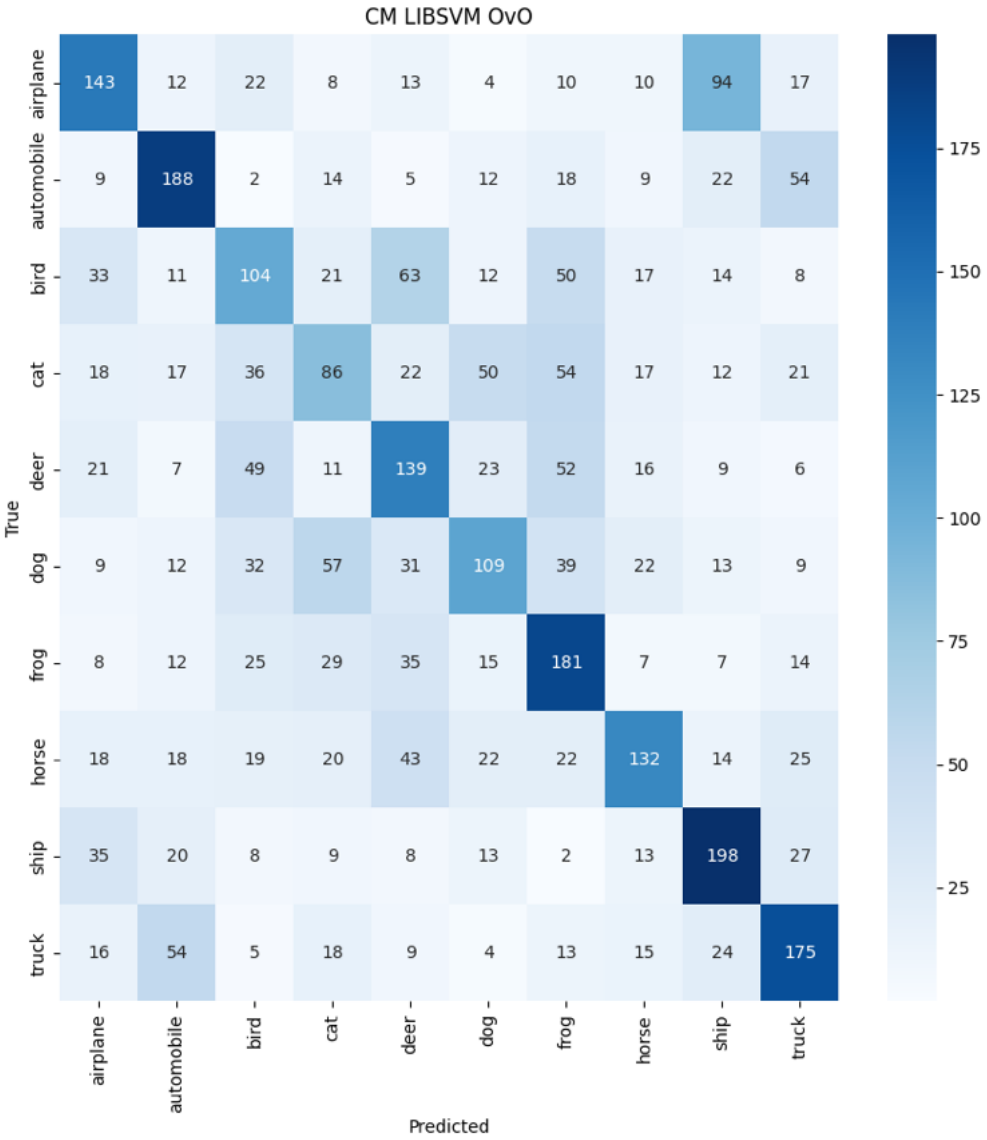


Figure 5: confusion matrix for LIBSVM multi class classifier

Table 9: Most Frequent Misclassifications

Actual → Predicted	**Count**
airplane → ship	94
bird → deer	63
dog → cat	57
automobile → truck	54
cat → frog	54
truck → automobile	54
deer → frog	52
bird → frog	50
cat → dog	50
deer → bird	49

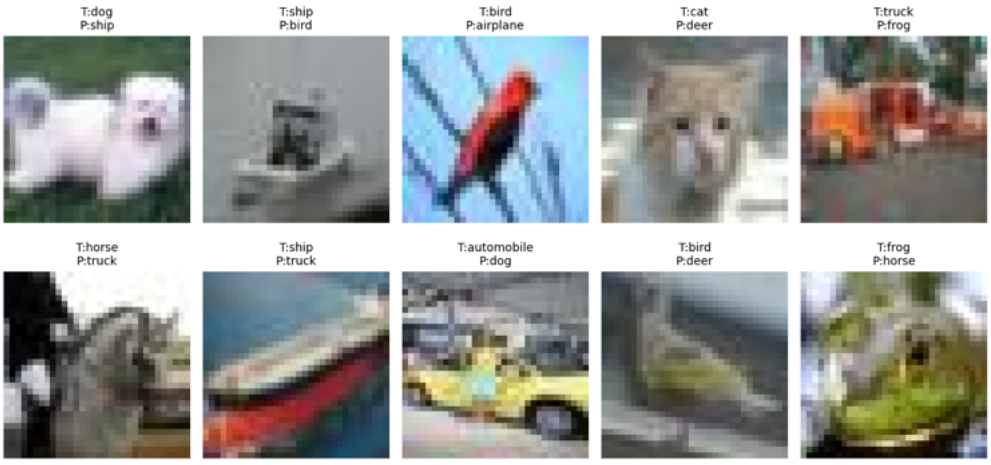


Figure 6: misclassified images using LIBSVM multi class classifier

6 5-fold cross validation and hyperparameter tuning

Q1.8 5-Fold Cross Validation and Hyperparameter Tuning

We performed 5-fold cross-validation on the training set to determine the optimal value of the regularization parameter C for the Gaussian kernel SVM, with $\gamma = 0.001$ fixed. The values of C considered were $\{10^{-5}, 10^{-3}, 1, 5, 10\}$. The corresponding cross-validation accuracies, test accuracies, and execution times are summarized below:

C	5-Fold CV Accuracy	Test Accuracy
10^{-5}	≈ 0.10	–
10^{-3}	≈ 0.10	–
1	0.425	–
5	0.461	0.4688
10	0.4692	0.4732

From the above results, we observe that both cross-validation and test accuracies increase with C up to a certain point, indicating better fitting of the training data. However, after $C = 10$, the improvement is marginal, suggesting that the model has reached an optimal regularization balance.

The highest 5-fold cross-validation accuracy was achieved at

$$C^* = 10$$

Training the SVM with $C = 10$ on the entire training set yielded a test accuracy of 0.4732, which improved upon previous results, confirming that appropriate tuning of C enhances model generalization performance.

Figure 7: SVM Accuracy vs C