COL774 Assignment 2

Rishabh Kumar (2021CS10103)

March 2025

1 Text Classification using Naïve Bayes

1.1 Naïve Bayes Multiclass Algorithm

The parameters used for the Naïve Bayes model are defined as follows:

$$\phi_{k|y=c} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = c\} + 1}{\sum_{i=1}^{m} 1\{y^{(i)} = c\} n_i + |V|} \tag{1}$$

$$\phi_{y=c} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = c\}}{m} \tag{2}$$

where:

- $|V|$ is the vocabulary size,
- $\phi_{k|y=c}$ represents the probability of the word k occurring in class c, computed using Laplace smoothing to prevent zero probabilities,
- $\phi_{y=c}$ represents the prior probability of class c.

In this implementation, no additional preprocessing was performed on the dataset, and the raw descriptions were directly tokenized for model training. The class labels are defined as follows:

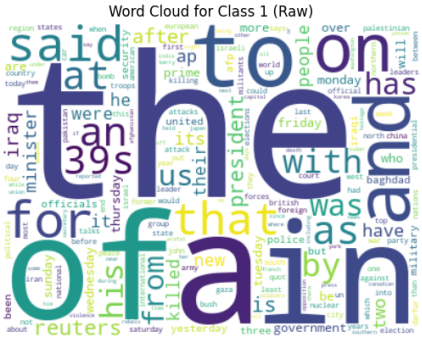
- World : 1
- Sports : 2
- Business : 3
- Science/Technology : 4

1.1.1 Accuracy Results

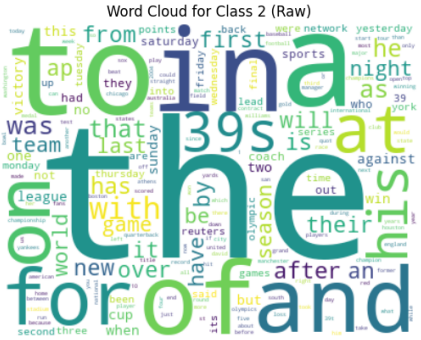
- **Training Accuracy**: 91.22%
- **Test Accuracy**: 89.26%

1.1.2 Word Clouds

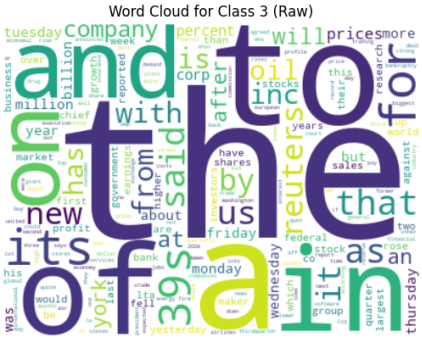
The following word clouds visualize the most frequent words in each category:



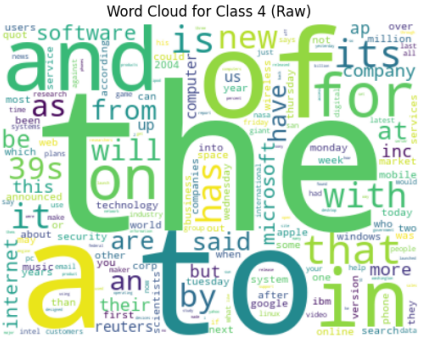
(a) World News



(b) Sports



(c) Business



(d) Science/Technology

Figure 1: Word clouds representing the most frequent words in each category.

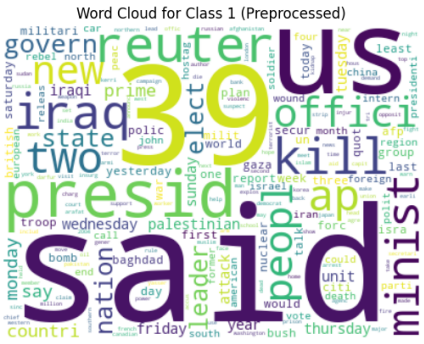
1.2 Stemming and Stopword removal

1.2.1 Code Implementation

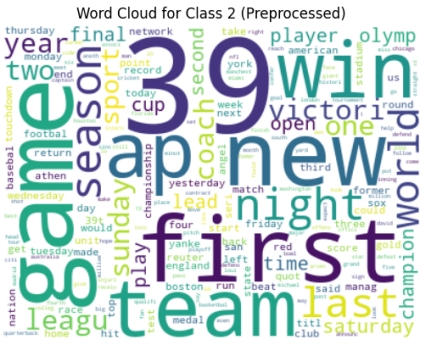
```python
def preprocess_tokens(tokens, remove_stopwords=True, stem=True):
    """Removes English stopwords and applies stemming."""
    if remove_stopwords:
        stop_words = set(stopwords.words('english'))
        tokens = [token for token in tokens if token not in stop_words]
    if stem:
        stemmer = PorterStemmer()
        tokens = [stemmer.stem(token) for token in tokens]
    return tokens
```

1.2.2 Word Clouds

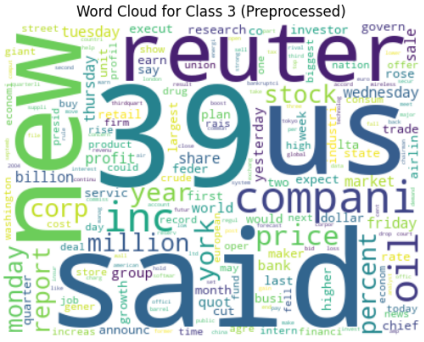
The following word clouds visualize the most frequent words in each category for preprocessed description:



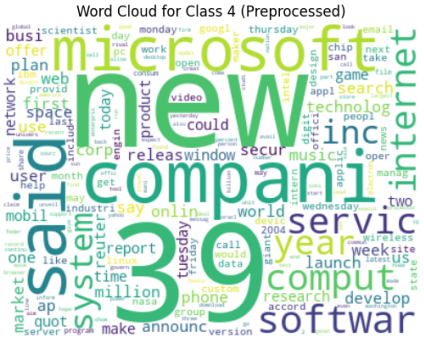
(a) World News



(b) Sports



(c) Business



(d) Science/Technology

Figure 2: Word clouds representing the most frequent words in each category.

1.2.3 Accuracy

- Training Set : 91.02%
- Test Set : 89.22%

1.2.4 Observation

Here we see a slight decrement in the accuracy after preprocessing which suggest that in this context stemming and stopword removal does not help mnuch with the accuracy.

1.3 Feature Engineering : Unigrams + Bigrams

```python
def create_bigrams(tokens):
    """Creates bigrams from a list of tokens."""
    return [tokens[i] + '_' + tokens[i+1] for i in range(len(tokens)-1)]

def combine_unigrams_bigrams(tokens):
    """Returns a list containing both unigrams and bigrams."""
    return tokens + create_bigrams(tokens)
```

The model trained using both unigrams and bigrams achieved the following performance metrics:

- **Training Accuracy:** 95.53%
- **Test Accuracy:** 90.37%

The introduction of bigrams as features led to a significant improvement in training accuracy, increasing from approximately 91.02% in the preprocessed unigram model to 95.53%. This suggests that bigrams capture additional contextual information that was previously lost in unigram-based feature extraction.

However, the test accuracy only saw a marginal improvement from 89.22% to 90.37%, indicating that while the model has learned better representations of the training data, it may have also introduced slight overfitting.

1.4 Analysis and Observations

Table 1 summarizes the classification performance across different experiments.

Experiment	Accuracy	Precision (Macro)	Recall (Macro)	F1-score (Macro)
Raw Description	89.26%	0.89	0.89	0.89
Preprocessed Description	89.22%	0.89	0.89	0.89
Unigrams + Bigrams	90.37%	0.90	0.90	0.90

Table 1: Performance comparison across different text representations.

From the results, we observe that preprocessing (stopword removal and stemming) does not significantly impact accuracy, suggesting that the classifier is robust to noise. However, incorporating bigrams leads to a notable increase in training accuracy (95.53%) and a modest improvement in test accuracy (90.37%), indicating enhanced feature representation while maintaining generalization. The slight performance gap between training and test accuracy suggests minimal overfitting. Overall, using both unigrams and bigrams improves classification performance by capturing contextual relationships between words. Precision, recall, and F1-score exhibit marginal improvements with bigrams, confirming better feature representation. The close match between training and test metrics suggests minimal overfitting. Overall, leveraging both unigrams and bigrams provides a richer text representation and improves classification effectiveness.

1.5 Title Features Only

In this section, we evaluate the classification performance of models trained using only the title features. Following the same methodology applied to description features, we conduct experiments on raw, preprocessed, and unigram-bigram feature representations. Table 2 summarizes the key performance metrics.

Experiment	Accuracy	Precision (Macro)	Recall (Macro)	F1-score (Macro)
Raw Title	86.45%	0.86	0.86	0.86
Preprocessed Title	86.04%	0.86	0.86	0.86
Unigrams + Bigrams (Title)	87.45%	0.87	0.87	0.87
Best Title Model	87.45%	0.87	0.87	0.87

Table 2: Performance comparison of models trained on title features.

1.5.1 Comparison with Description-Based Models

To assess the effectiveness of title-based classification, we compare the best-performing title model with the best-performing description model. Table 3 presents the key performance metrics for both models.

Feature Set	Accuracy	Precision (Macro)	Recall (Macro)	F1-score (Macro)
Best Description Model	90.37%	0.91	0.91	0.91
Best Title Model	87.45%	0.87	0.87	0.87

Table 3: Performance comparison of best title vs. best description models.

1.5.2 Observations

- The best **description-based model outperforms the best title-based model** in all evaluation metrics, achieving a higher accuracy of **90.37% vs. 87.45%**.

- **Titles provide limited context,** leading to lower classification accuracy compared to descriptions, which contain more detailed information about each article.
- **Unigrams + Bigrams improve performance** for both feature sets, but the gain is more substantial for description features due to their richer textual content.

Overall, description-based models yield better classification performance, confirming that news descriptions provide more useful information than titles alone.

1.6 Combined Title and Description

1.6.1 Concatenated Title and Description

In this experiment, we concatenate title and description features and train a Naïve Bayes classifier using unigrams and bigrams. This ensures the model learns a single set of parameters for both features.

Feature Set	Accuracy
Best Description Model	90.37%
Best Title Model	87.45%
Combined Model	91.18%

Table 4: Accuracy comparison of combined vs. single feature models.

- The combined model achieves 91.18% accuracy, slightly higher than the best description-based model (90.37%), indicating improvement from adding title features.
- Training accuracy increases to 95.85%, suggesting higher model complexity but minor overfitting.
- Title features alone are less informative than descriptions, and their inclusion does not significantly enhance performance.

1.6.2 Separate Parameters for Title & Description

In this experiment, we allow the model to learn separate parameters for title and description, denoted as θ_{title} and θ_{desc}. The maximum likelihood estimates for these parameters with Laplace smoothing are given by:

$$\theta^{(\text{title})}_{k|y=c} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = c\} + \alpha}{\sum_{i=1}^{m} 1\{y^{(i)} = c\} n_i + |V_{\text{title}}|\alpha}$$

$$\theta^{(\text{desc})}_{k|y=c} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = c\} + \alpha}{\sum_{i=1}^{m} 1\{y^{(i)} = c\} n_i + |V_{\text{desc}}|\alpha}$$

where V_{title} and V_{desc} are the vocabulary sizes of title and description, respectively.

Feature Set	Accuracy
Best Description Model	90.37%
Best Title Model	87.45%
Joint Concatenation Model	91.18%
Separate Parameter Model	90.28%

Table 5: Accuracy comparison of separate vs. single parameter models.

Observations

- The separate parameter model achieves 90.28% accuracy, lower than the best description model (90.37%) but slightly better than the title-only model (87.45%).
- Compared to the joint concatenation model (91.18%), the separate parameter model does not provide a significant advantage, suggesting that a unified feature space is more effective.
- Training accuracy (92.04%) indicates minimal overfitting, but title and description features may not contribute equally to classification.

1.7 Baseline Comparisons

To evaluate the effectiveness of our best model, we compare it against simple baseline classifiers.

- **(a) Random Baseline:** By randomly assigning one of the four categories to each article, we achieve an accuracy of 25.26%. This aligns with the expected performance of a random classifier in a balanced four-class problem.
- **(b) Positive Baseline:** Predicting all samples as a fixed class results in an accuracy similar to the majority class baseline, which achieves 25.00%.
- **(c) Improvement Over Baselines:** The best-performing model from Experiment 6 achieves 91.18% accuracy, demonstrating a substantial improvement of over 66 percentage compared to the baselines. This confirms that our model effectively captures meaningful patterns in the data rather than relying on random or trivial heuristics.

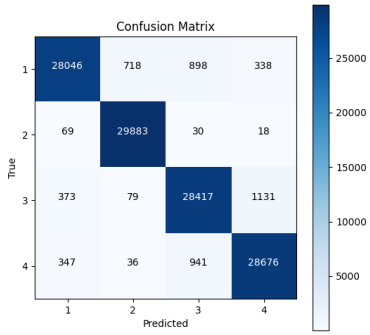
1.8 Confusion Matrix Analysis

To further analyze the performance of our best models, we explore their confusion matrices. The two models considered are:

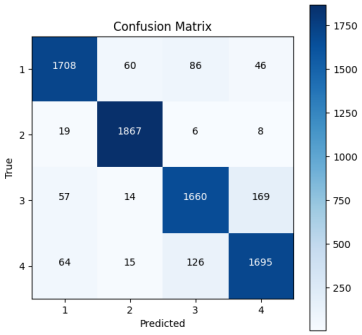
- **Model 6a (Concatenated Unigram + Bigram)**: Achieves 91.18% accuracy, combining both title and description features.
- **Model 6b (Separate Parameters for Title & Description)**: Achieves 90.28% accuracy, learning distinct parameters for each feature.

1.8.1 Confusion Matrices

The confusion matrices for the best-performing model (Model 6a) on both training and test data are presented below.



(a) Confusion Matrix for Training Data (Model 6a)



(b) Confusion Matrix for Test Data (Model 6a)

Figure 3: Confusion matrix for model 6a

1.8.2 Observations

- **(a) Highest Diagonal Entry:** In each confusion matrices, the highest value appears at (2,2), indicating that category 2 (corresponding to the 'Sports' class) is classified most accurately.
- **(b) Interpretation:** A high diagonal value signifies that the model is correctly predicting this category more often than others. This suggests that class 2 has distinct features that make it easier to classify, whereas other categories may have more overlapping characteristics leading to occasional misclassifications.

While Model 6a achieves slightly higher accuracy, Model 6b demonstrates **less overfitting**, making it a strong alternative. The confusion matrices help identify which categories are well-separated and which may need further feature refinement.

1.9 Additional Feature Engineering

In this experiment, we enhance the Naïve Bayes model by incorporating discriminative keyword count features. The key steps in this process include:

- Identifying the most discriminative words for each class based on their relative probabilities.
- Creating new features by counting occurrences of these words in each sample.
- Categorizing these counts into ‘none‘, ‘low‘, and ‘high‘ bins to introduce structured categorical features.
- Combining these categorical features with existing title and description token features.

1.9.1 Results

The enhanced model is trained and evaluated using these additional features. The results are as follows:

- **Training Set Accuracy:** 91.88%
- **Test Set Accuracy:** 90.25%

1.10 Comparison and Observations

- The enhanced model achieves an accuracy of 90.25% on the test set, which is comparable to the best-performing models (91.18% for concatenated unigram-bigram and 90.28% for separate parameter learning).
- The newly introduced features contribute positively but do not drastically improve test accuracy, suggesting that title and description already capture most of the discriminative power.
- Training accuracy remains high (91.88%), indicating that the model is effectively utilizing the new features without significant overfitting.

2 Image Classification using SVM

Binary Classification

2.1 Binary Image Classification using SVM and CVXOPT

In this experiment, we build a binary image classifier using Support Vector Machines (SVMs) and solve the optimization problem through the CVXOPT package. The dataset consists of RGB images that have been pre-processed—each image is resized to 100×100 pixels, center-cropped, flattened into a 30,000-dimensional vector, and normalized by scaling the pixel values to lie in [1] (i.e., by dividing by 255).

The binary classification task considers two weather classes selected based on the my entry number (2021CS10103). For my experiment, the two classes chosen are **“glaze”** and **“hail”**.

SVM Dual Formulation

Given a training set

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}, \quad \mathbf{x}_i \in \mathbb{R}^{30000}, \quad y_i \in \{-1, +1\},$$

the soft-margin SVM dual optimization problem can be formulated as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j \, y_i \, y_j \, K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, m, \\ & \sum_{i=1}^{m} \alpha_i \, y_i = 0, \end{aligned}$$

where for a *linear kernel* we have:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j.$$

Here, $C = 1.0$ is used to allow for soft margins, thereby accommodating noisy data.

Once the optimal Lagrange multipliers $\boldsymbol{\alpha}$ are found, the weight vector and bias term can be computed as:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \, y_i \, \mathbf{x}_i,$$

and the bias b is estimated by averaging over the support vectors:

$$b = \frac{1}{N_{\text{SV}}} \sum_{i \in \text{SV}} \Big(y_i - \mathbf{w}^\top \mathbf{x}_i \Big),$$

with a threshold (e.g., $\alpha_i > 10^{-5}$) used to designate support vectors.

2.1.1 Support Vectors:

- Number of support vectors: 966.
- Percentage of training samples as support vectors: 98.27%.

The extremely high number of support vectors (98.27% of the training examples) indicates that nearly all the data points lie very close to the decision boundary. This typically suggests that the data might not be linearly separable or that the chosen features (in this case, the raw pixel information) do not provide a clear margin between the two classes.

2.1.2 Weight Vector and Bias:

- The computed weight vector has a norm of $\|\mathbf{w}\| = 2.2030$.
- The bias term is $b = -1.6080$.
- The resulting decision function is: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} - 1.6080$, and the test set accuracy is 64.37%.

2.1.3 Visualization

The top-5 support vectors, selected based on the highest α_i coefficients, have been reshaped into images of dimensions 100 x 100 x 3 and plotted. Similarly, the weight vector $\mathbf{w}$ has been reshaped and visualized as an image.

Figure 4: Top-5 Support Vectors (reshaped from the highest α_i values).

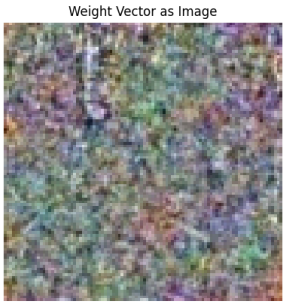


Figure 5: Visualization of the weight vector $\mathbf{w}$ (reshaped to image dimensions).

2.2 CVXOPT SVM with Gaussian Kernel

For this experiment, we solve the SVM dual problem using a Gaussian (RBF) kernel defined as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\Big),$$

with $\gamma = 0.001$ and $C = 1.0$. In this case, the matrix P is computed as

$$P_{ij} = y_i \, y_j \, K(\mathbf{x}_i, \mathbf{x}_j),$$

and unlike the linear kernel, the weight vector $\mathbf{w}$ is not explicitly formed.

2.2.1 Support Vectors

- **Number of support vectors:** 895.
- **Percentage of training samples as support vectors:** 91.05%.
- **Common support vectors (with linear SVM):** 879.

2.2.2 Test Set Accuracy

Since the Gaussian kernel does not allow for an explicit representation of $\mathbf{w}$ or b, we use the learned model directly to classify the test examples. The Gaussian SVM achieves a test set accuracy of 72.06%.

2.2.3 Visualization

The top-5 support vectors, selected based on the highest dual coefficients, are reshaped into images of dimensions $100 \times 100 \times 3$ as shown in Figure 6.

Figure 6: Top-5 Support Vectors (Gaussian SVM) reshaped from the highest α_i values.

2.2.4 Accuracy Comparison

The Gaussian SVM achieves higher test accuracy (72.06%) than the linear SVM (64.37%) because its non-linear kernel better captures complex data patterns, leading to a more flexible decision boundary.

2.3 scikit-learn SVM using LIBSVM

Using scikit-learn's SVM implementation (LIBSVM), we obtain the following results:

2.3.1 Support Vector Comparison

- **Linear Kernel:** 687 support vectors, all common with those from CVXOPT.
- **Gaussian Kernel:** 867 support vectors, all matching with the CVXOPT Gaussian case.

2.3.2 Weight Vector and Bias Comparison (Linear Kernel)

- CVXOPT: $\|\mathbf{w}\| = 2.20298$, $b = -1.60800$.
- scikit-learn: $\|\mathbf{w}\| = 2.19619$, $b = -1.24677$.

2.3.3 Test Set Accuracy

- **Linear Kernel:** Test accuracy is 63.56%.
- **Gaussian Kernel:** Test accuracy is 71.66%.

2.3.4 Computational Cost Comparison

- scikit-learn Linear SVM training time: 21.1414 sec.
- scikit-learn Gaussian SVM training time: 17.8972 sec.
- LIBSVM exhibits significantly faster training times for both Linear and Gaussian kernels compared to CVXOPT. This suggests that LIBSVM is more efficient in terms of computational speed

2.4 SVM Optimization using SGD

We optimize the SVM objective with stochastic gradient descent (using scikit-learn's SGDClassifier with hinge loss) and compare it with the LIBLINEAR implementation (scikit-learn's SVC with a linear kernel):

- **SGD SVM:** Test set accuracy is 65.99% with a training time of 4.5093 sec.
- **LIBLINEAR SVM:** Test set accuracy is 63.56% with a training time of 21.1414 sec.

Thus, the SGD-based solver not only achieves slightly higher accuracy but is also significantly faster than the LIBLINEAR approach.

Multi-Class Classification

2.5 Multi-Class Image Classification using One-vs-One SVM

In this experiment, we extend the binary SVM formulation to a multi-class setting using a one-vs-one approach. Given 11 classes, we train $\binom{11}{2} = 55$ binary classifiers using the Gaussian kernel with $C = 1.0$ and $\gamma = 0.001$. At test time, each classifier votes for one class, and in the case of ties, the class with the highest aggregated score is selected.

2.5.1 Results

- **Multi-class Training Samples:** 5484
- **Multi-class Test Samples:** 1378
- **Test Set Accuracy:** 53.77%

This confirms that the one-vs-one multi-class SVM, implemented via our CVXOPT solver, achieves a test accuracy of 53.77% on the given dataset.

2.6 scikit-learn Multi-Class SVM with Gaussian Kernel

Using scikit-learn's SVC (LIBSVM) with a Gaussian kernel (with $C = 1.0$ and $\gamma = 0.001$) on the multi-class dataset (5484 training samples, 1378 test samples), we obtained:

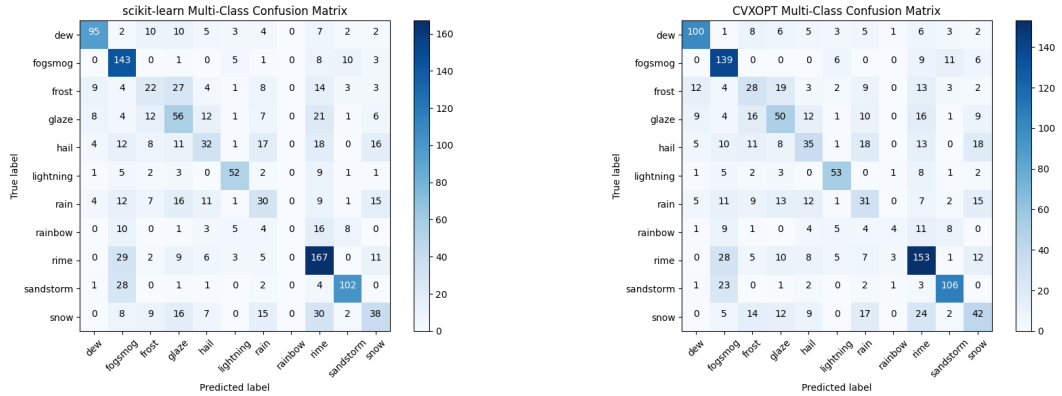
- **Test set accuracy:** 53.48%
- **Training time:** 212.2514 seconds

Compared to the CVXOPT multi-class SVM (which achieved 53.77% test accuracy), the LIBSVM-based approach yields a very similar classification performance. However, the training time for scikit-learn's implementation is noticeably quantified (212.25 sec), indicating a potential difference in computational cost between the two solvers.

2.7 Confusion Matrix of Multi-Class SVM Classification Results

In this section, we analyze the confusion matrices for both our CVXOPT and scikit-learn (LIBSVM) multi-class SVM implementations, examine the most common misclassification patterns, and discuss selected examples of misclassified images.

Confusion Matrices



(a) scikit-learn Multi-Class Confusion Matrix

(b) CVXOPT Multi-Class Confusion Matrix

Figure 7: Confusion Matrices

Observations from Confusion Matrices

Examining both confusion matrices reveals several important patterns:

- Both CVXOPT and scikit-learn implementations show similar misclassification trends, indicating these errors likely stem from inherent dataset challenges rather than algorithm differences.
- The "rime" class is most frequently misclassified as "fogsmog" in both models (28-29 instances), likely due to their visual similarity as both feature white/gray atmospheric conditions.
- "Fogsmog" appears to be a common incorrect prediction for many classes, suggesting it may have less distinctive visual features.
- Strong diagonal elements for "fogsmog" (139-143), "rime" (153-167), and "sandstorm" (102-106) indicate these classes are more reliably identified by both models.
- The "snow" class shows significant confusion with multiple other classes including "rime," "rain," and "glaze" in both implementations, reflecting the visual similarity of these winter/precipitation phenomena.
- The "hail" class shows poor classification performance in both models, with frequent confusion with "rain," "rime," and "snow" classes.

Misclassified Examples

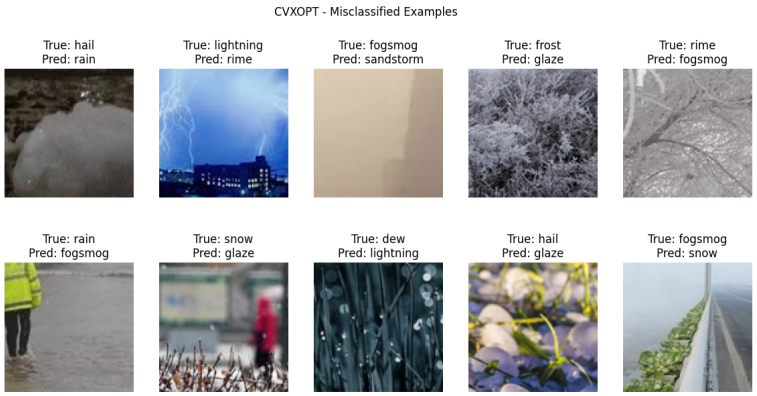


Figure 8: CVXOPT - Misclassified Examples

Figure 9: scikit-learn - Misclassified Examples

Analysis of Misclassifications

The misclassified examples provide valuable insights into why the models struggle with certain classes:

- Many misclassifications occur between visually similar weather conditions. For example, hail being classified as rain or snow, which is understandable given they are all forms of precipitation with similar visual appearance.
- The "lightning" image misclassified as "rime" in the CVXOPT model shows how distinctive features can be lost in the flattened representation—the bright flash against dark sky is visually distinct to humans but challenging for the model.
- Several "dew" samples are misclassified as "lightning" or "rime," likely due to the common presence of water droplets which can create similar light reflection patterns.
- In the scikit-learn examples, images of "glaze" are frequently confused with "dew," "rime," or "frost"—all phenomena involving thin layers of water or ice on surfaces.
- The "fogsmog" images misclassified as "snow" or "sandstorm" demonstrate how atmospheric conditions with reduced visibility present classification challenges.

Overall, these results make intuitive sense given the inherent visual similarity between many weather phenomena. The models perform reasonably well considering the challenges, but certain class distinctions remain difficult without additional context or temporal information. Future improvements could focus on feature engineering to better capture the distinguishing characteristics of easily confused classes like hail, rain, and snow, or dew, frost, and glaze.

2.8 Hyperparameter Tuning with 5-Fold Cross-Validation

In this experiment, we employ 5-fold cross-validation to determine the optimal regularization parameter C for our Gaussian kernel SVM. This systematic approach divides the training data into 5 equal parts, using each fold as a validation set while training on the remaining 4 folds.

2.8.1 Cross-Validation and Test Accuracy Results

C Value	5-fold CV Accuracy	Test Accuracy
10^{-5}	16.92%	16.84%
10^{-3}	16.92%	16.84%
1	53.03%	53.48%
5	53.76%	54.93%
10	53.26%	53.92%

Table 6: Comparison of 5-fold CV and Test Accuracy for different C values ($\gamma = 0.001$)

2.8.2 Analysis of CV and Test Accuracy Trends

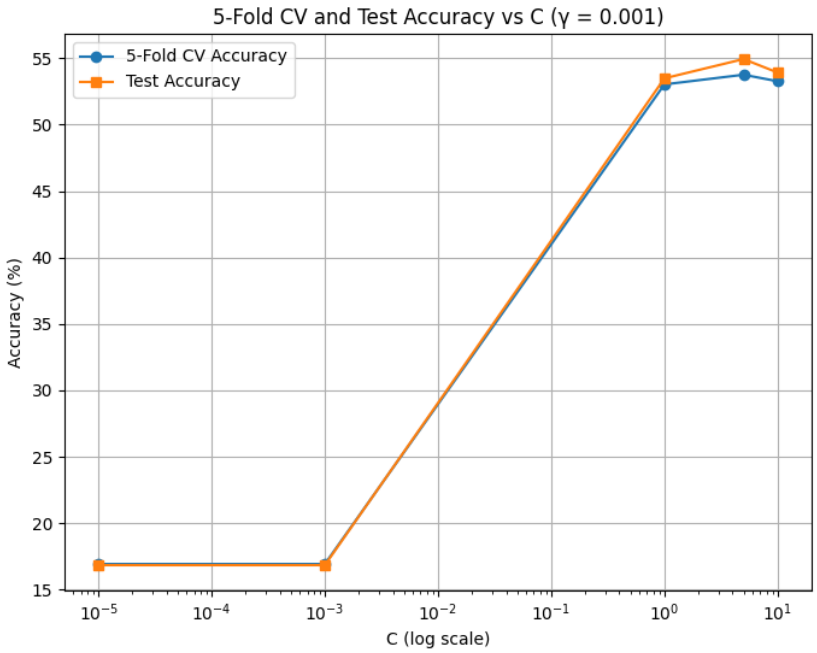


Figure 10: 5-Fold CV and Test Accuracy vs C ($\gamma = 0.001$)

The plot shows that both cross-validation and test accuracies follow a similar trend across different C values. For very small values of C (10^{-5} and 10^{-3}), the model performs poorly, achieving only around 17% accuracy, which is close to random guessing for 11 classes.

There is a dramatic improvement in performance when C increases to 1, with accuracy jumping to approximately 53%. The peak performance is achieved at C = 5, with a CV accuracy of 53.76% and

test accuracy of 54.93%. As C increases further to 10, there is a slight decline in performance.

The best C value based on 5-fold cross-validation is 5.

2.8.3 Final Model Performance and Observations

Training the final SVM model with C = 5 on the entire training set yields a test accuracy of 54.93%. This represents an improvement of approximately 1.45% over the previous model using C = 1 (which achieved 53.48% accuracy). This improvement demonstrates the value of hyperparameter tuning through cross-validation. The results suggest that a moderate value of C = 5 provides the best balance between model complexity and generalization ability. Very small C values (10^{-5}, 10^{-3}) result in underfitting, as the model places too little emphasis on minimizing training errors. As C increases beyond the optimal value (C = 5), the model begins to show signs of overfitting, with test accuracy starting to decline. The close alignment between cross-validation and test accuracies indicates that our 5-fold CV procedure effectively estimates model performance on unseen data. These findings highlight the importance of proper hyperparameter selection in achieving optimal SVM classification performance for multi-class image data.