COL 774: Assignment 2 Report

Parth Thakur, 2021CS50615

Wednesday Oct 4, 2023

1 Text Classification

1.1 Naïve Bayes Multiclass Classification

The prior probability for a class (e.g., Positive) is computed as:

$$P(\text{class}) = \frac{\text{Number of samples in the class}}{\text{Total number of samples}} \tag{1}$$

The likelihood of a word given a class is:

$$P(\text{word}|\text{class}) = \frac{\text{Number of times word appears in class} + 1}{\text{Total number of words in class} + \text{Size of vocabulary}} \tag{2}$$

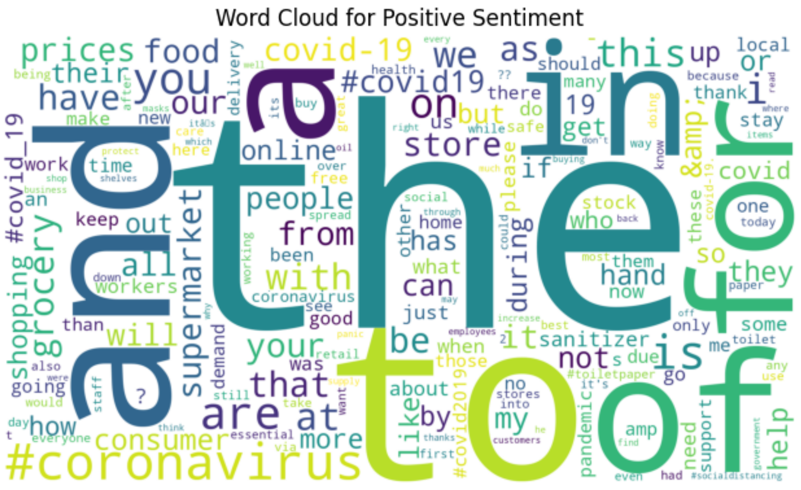
The class prediction for a given document (or tweet) is:

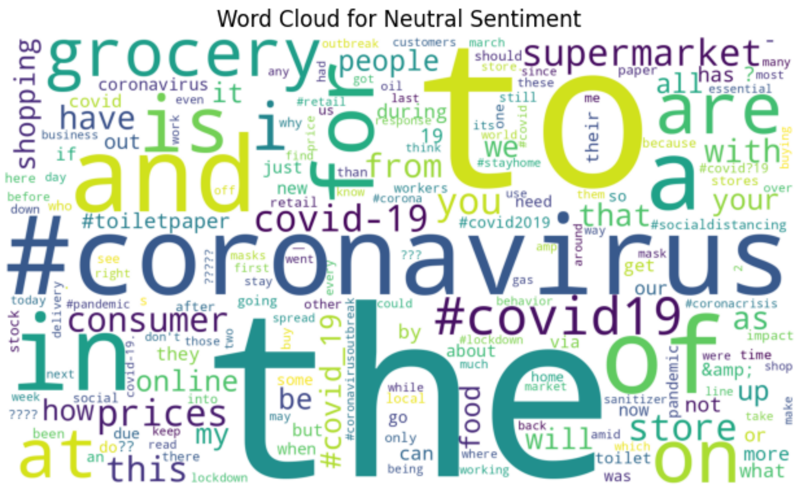
$$\text{argmax}_{\text{class}} \left(\log(\text{prior of class}) + \sum_{\text{word in document}} \log(\text{likelihood of word given class}) \right) \tag{3}$$

1.1.1 Accuracy Report

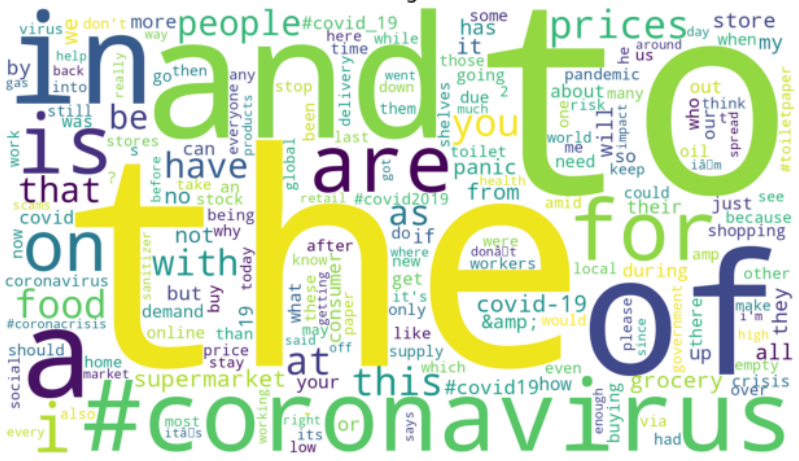
- Training: 80.23 percent
- Validation: 69.96 percent

1.1.2 Word Cloud Construction





Word Cloud for Negative Sentiment



1.2 Random and Positive Baseline Accuracy

- Random:
  - Training: 33.32 percent
  - Validation: 32.82 percent
- Positive:
  - Training: 43.84 percent
  - Validation: 43.85 percent

For three classes, the accuracy of random guessing is:

$$\text{Accuracy} = \frac{1}{3} \tag{4}$$

The accuracy when predicting all samples as positive is:

$$\text{Accuracy} = \frac{\text{Number of positive samples}}{\text{Total number of samples}} \tag{5}$$

Comparing to the Naive Bayes Classifier I have created, as expected, it performs better than both Random and Positive Prediction.

The improvement of our Naïve Bayes classifier over:

- Random guessing is $\approx$ 34.42%.
- Predicting all as ”Positive” is $\approx$ 23.90%.

Method	**Training Accuracy**	**Validation Accuracy**
Random	33.32%	32.82%
Positive	43.84%	43.85%
Naïve Bayes	80.23%	69.96%

Table 1: Comparison of different methods for text classification.

1.3 Confusion Matrix

	Predicted Positive	**Predicted Neutral**	**Predicted Negative**
Actual Positive	True Positives (TP)	False Neutral (FN)	False Negative (FN)
Actual Neutral	False Positive (FP)	True Neutrals (TN)	False Neutral (FN)
Actual Negative	False Positive (FP)	False Neutral (FN)	True Negatives (TN)

Table 2: Confusion Matrix Format

1.3.1 Naive Bayes

Training Set

	Positive	Negative	Neutral
Positive	14770	1565	267
Negative	1699	12252	215
Neutral	2238	1500	3358

Validation Set

	Positive	Negative	Neutral
Positive	1196	226	22
Negative	254	952	26
Neutral	290	171	156

1.3.2 Random

Training Set

	Positive	Negative	Neutral
Positive	5575	5528	5499
Negative	4834	4689	4643
Neutral	2407	2337	2352

Validation Set

	Positive	Negative	Neutral
Positive	465	505	474
Negative	401	427	404
Neutral	205	223	189

1.3.3 Positive

Training Set

	Positive	Negative	Neutral
Positive	16602	0	0
Negative	14166	0	0
Neutral	7096	0	0

Validation Set

	Positive	Negative	Neutral
Positive	1444	0	0
Negative	1232	0	0
Neutral	617	0	0

1. Naive Bayes:
   - Training Set: The category with the highest value on the diagonal is Positive with 14770 correct predictions.
   - Validation Set: The category with the highest value on the diagonal is Positive with 1196 correct predictions.
2. Random:
   - Training Set: The category with the highest value on the diagonal is Positive with 5575 correct predictions.
   - Validation Set: The category with the highest value on the diagonal is Positive with 465 correct predictions.
3. Positive:
   - Training Set: The category with the highest value on the diagonal is Positive with 16602 correct predictions.
   - Validation Set: The category with the highest value on the diagonal is Positive with 1444 correct predictions.

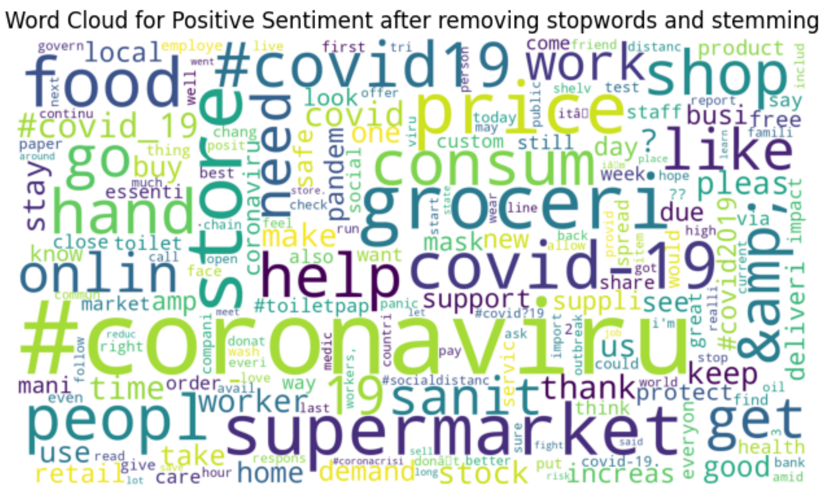
The highest value on the diagonal entry indicates that this category was the most accurately predicted among the three. For all three methods (Naive Bayes, Random, and Positive), the "Positive" category has the highest number of correctly predicted instances. This means that the model is best at correctly identifying positive samples.

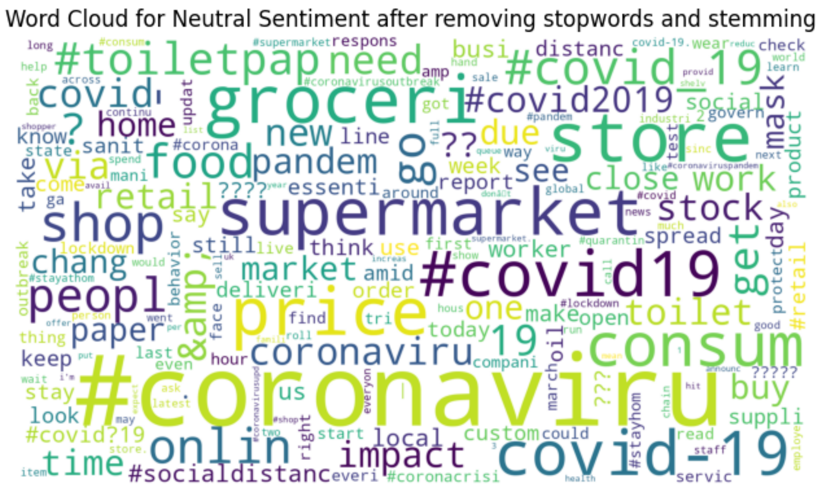
1.4 Stopword Removal and Stemming

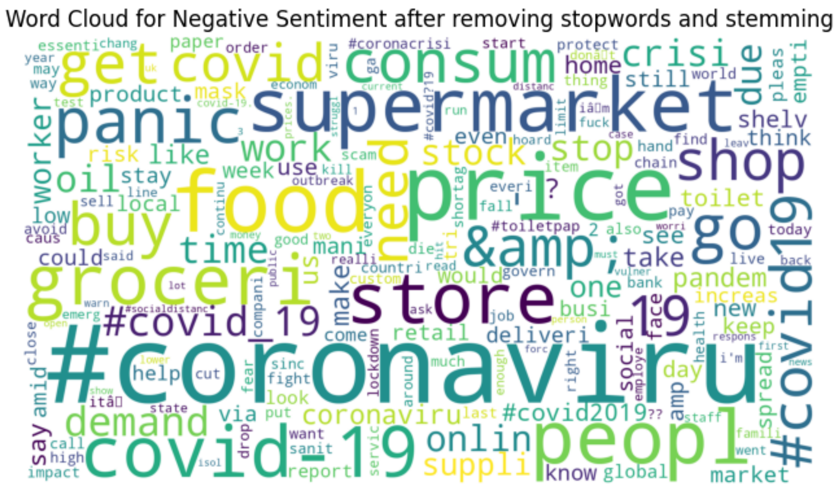
1.4.1 Data Transformation

Stop Word removal is performed using the set of English stop words in the nltk corpus. Stemming is done using PorterStemmer from nltk.stem.

1.4.2 Word Clouds







1.4.3 Model Accuracy

- Training: 73.46 percent
- Validation: 65.65 percent

1.4.4 Observations

The wordclouds created after removing stop words and stemming more accurately represent the words with positive and negative sentiment. However, validation accuracy decreases slightly.

Here are a few potential reasons for this reduction:

1. Loss of Contextual Information: By removing stopwords, we may have inadvertently removed some contextual information from the tweets. In some cases, stopwords can provide valuable context. For instance, the phrase "not good" is negative, but if we remove the stopword "not", the sentiment could be misinterpreted.

2. Over-Stemming: Stemming can sometimes be aggressive, merging words that should be distinct. This can lead to loss of valuable information. For example, the words "universe" and "university" might both be stemmed to "univers", even though they have very different meanings.

3. Variability in Data: The nature of tweets is such that many words or phrases are used in a colloquial or slang manner. Some of these might be treated as stopwords or might get inappropriately stemmed, leading to loss of sentiment information.

4. Model Simplicity: The Naïve Bayes classifier assumes that features (words, in this case) are independent given the class label. This is a strong assumption, especially for textual data where word order and context matter. The preprocessing might exacerbate the impact of this assumption by removing or altering words that provide context.

1.5 Feature Engineering

1.5.1 Using Bigrams

- Validation Accuracy: 66.17 percent

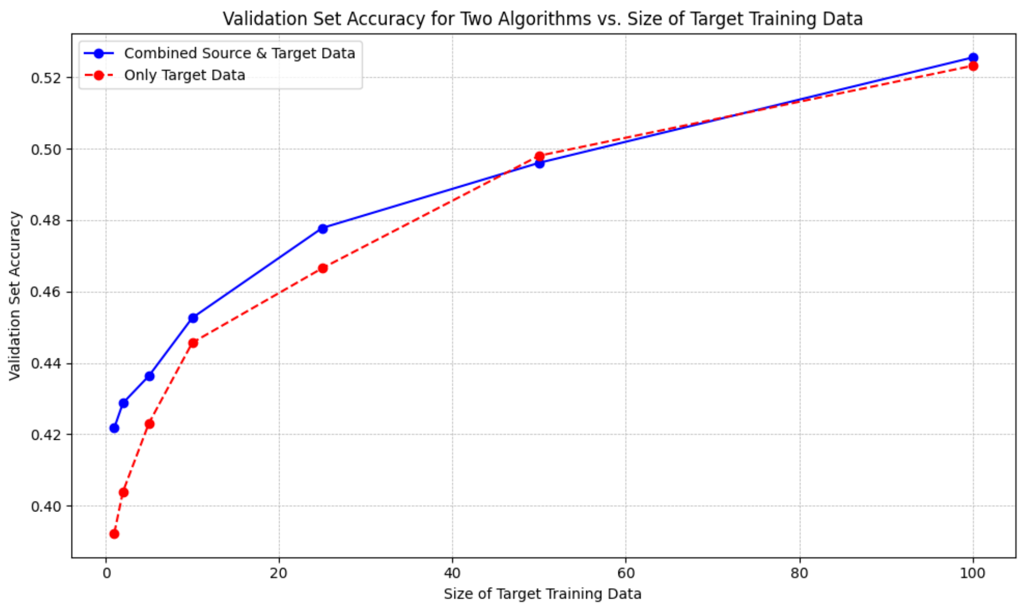
1.5.2 Additional Feature: Number of Words

Intuitively, the length of a tweet might give some information about its sentiment, with very short tweets possibly being more neutral or negative, while longer tweets might be more descriptive and positive.

- Validation Accuracy: 66.32 percent

- Improvement of about 0.2 percent

1.6 Domain Adaptation



1.6.1 Observations:

- For both algorithms, the validation accuracy increases as more target training data is added. This indicates that having more target domain data generally improves the model's performance on unseen target domain data.

- The first algorithm (combined source and target data) shows a more gradual increase in validation accuracy as the dataset size increases. By contrast, the second algorithm (only target data) starts with a lower validation accuracy for smaller datasets but catches up and eventually surpasses the first algorithm as the dataset size increases.

- The results indicate the value of domain adaptation. The first algorithm, which leverages data from the source domain, provides a head start in performance, especially when the target domain data is limited. As more target domain data becomes available, the advantage of using source domain data diminishes, and the second algorithm's performance surpasses the first.

In conclusion, these observations highlight the benefits of domain adaptation, especially when target domain data is limited. However, as more target domain data becomes available, the advantage of using only target domain data becomes apparent. The results also emphasize the importance of having a sufficiently large dataset to prevent overfitting and ensure better generalization to unseen data.

2 Image Classification

2.1 Binary Classification

2.1.1 Using CVXOPT, Linear Kernel

Given the standard SVM dual formulation:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

subject to:

$$0 \leq \alpha_i \leq C,$$
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

The CVXOPT standard quadratic problem format is:

$$\min_{\alpha} \frac{1}{2} \alpha^T P \alpha + q^T \alpha$$

subject to:

$$G\alpha \leq h,$$
$$A\alpha = b$$

To put the SVM dual in this form:

- P is an $m \times m$ matrix where $P_{ij} = y^{(i)} y^{(j)} \langle x^{(i)}, x^{(j)} \rangle$
- q is an m sized column vector with all elements being -1.
- G is a $2m \times m$ matrix constructed such that it considers both $\alpha_i \geq 0$ and $\alpha_i \leq C$.

- h is a $2m$ sized vector with first m elements being 0 and next m elements being C.
- A is a $1 \times m$ row vector with elements as the labels $y^{(i)}$.
- b is a scalar 0.

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$

For a chosen support vector x_s:

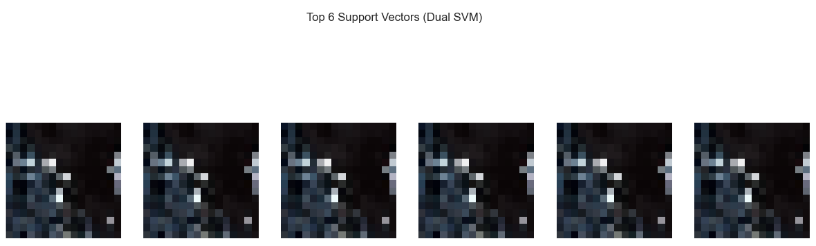
$$b = y_s - \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x_s \rangle$$

Ensure $0 < \alpha_s < C$ for the chosen x_s.

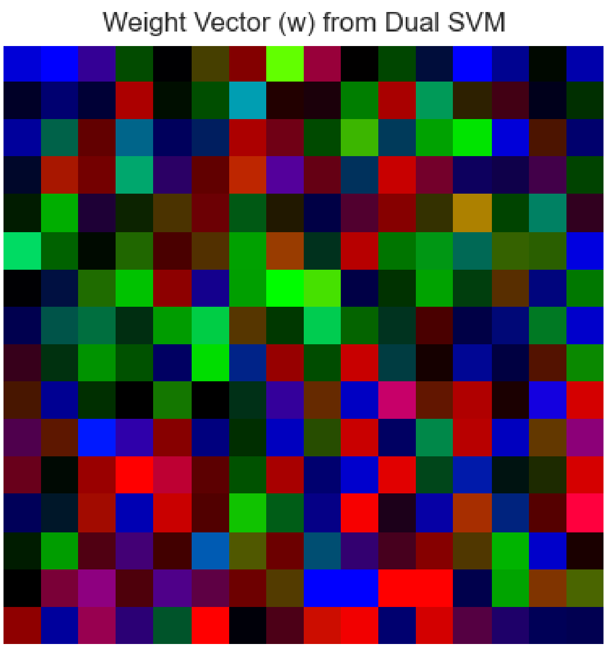
$$\text{prediction} = \text{sign}(\langle w, x \rangle + b)$$

Results:

- Number of Support Vectors Obtained: 2383
- Percentage of training samples constituting support vectors: 50.06 percent
- Intercept (b)= 1.041890005986661
- The Validation accuracy came to be 77.5 percent.



-



•

2.1.2 Using CVXOPT, Gaussian Kernel

The Gaussian (or Radial Basis Function - RBF) kernel between two points x and z is defined as:

$$K(x, z) = \exp(-\gamma \|x - z\|^2)$$

where γ is a parameter that controls the width of the Gaussian function.

section*SVM Dual Formulation with Gaussian Kernel

The dual formulation of the SVM optimization problem with the Gaussian kernel is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} K(x^{(i)}, x^{(j)}) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned}$$

where:

- α_i are the dual variables
- $y^{(i)}$ are the labels, taking values in { -1, 1 }
- $K(x^{(i)}, x^{(j)})$ is the Gaussian kernel between points $x^{(i)}$ and $x^{(j)}$
- C is a regularization parameter

CVXOPT solves the quadratic programming problem in the following standard form:

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

To express the SVM dual in this form:

- P is an $m \times m$ matrix with elements $P_{ij} = y^{(i)} y^{(j)} K(x^{(i)}, x^{(j)})$
- q is an m-sized column vector with all elements being -1
- G is a $2m \times m$ matrix constructed such that it considers both $\alpha_i \geq 0$ and $\alpha_i \leq C$
- h is a $2m$-sized vector with the first m elements being 0 and the next m elements being C
- A is a $1 \times m$ row vector with elements as the labels $y^{(i)}$
- b is a scalar 0

Results:

- Number of support vectors: 3058
- Validation Accuracy: 75.5 precent

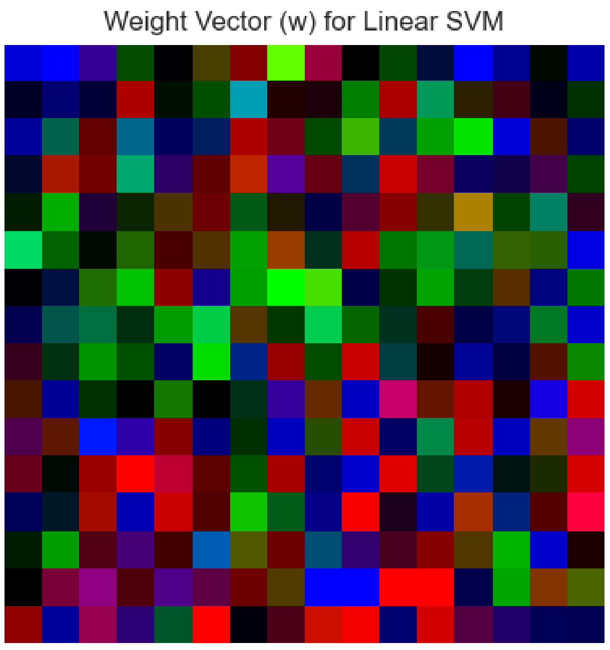
2.1.3 Using scikit-learn SVM function, Linear Kernel

Results:

- Number of Support Vectors Obtained: 2371
- Percentage of training samples constituting support vectors: 49.81 percent

- Intercept (b)= 1.06989668
- The Validation accuracy came to be 78 percent.

-



-

2.1.4 Using scikit-learn SVM function, Gaussian(RBF) Kernel

Results:

- Number of Support Vectors Obtained: 3054
- Percentage of training samples constituting support vectors: 64.16 percent

- The Validation accuracy came to be 75.25 percent.

-

2.1.5 Comparison of b and Support Vectors

For Gaussian Kernel

- b found using cvxopt = -1.06832676477997
- b found using libsvm = -1.0622098653371628
- The bs obtained are approximately equal
- 3054 support vectors match

For Linear Kernel

- b found using cvxopt = 1.041890005986661
- b found using libsvm = 1.06989668
- 2371 support vectors match
- 383 entries in w are not within a margin of 1e-5

Between Linear and Gaussian Kernel

- 2206 support vectors match when using cvxopt
- 2190 support vectors match when using libsvm

2.2 Multi-Class Image Classification

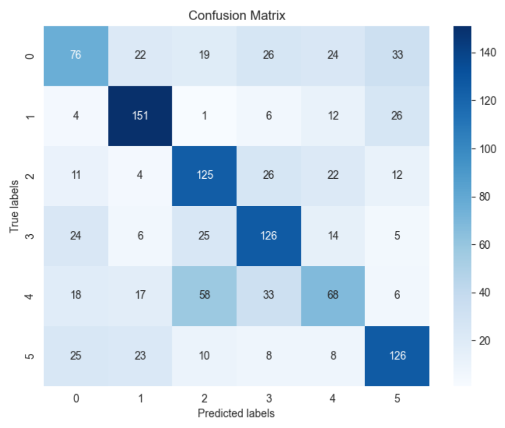
2.2.1 One-vs-One Multi-Class SVM using CVXOPT

- Validation Accuracy: 56 percent

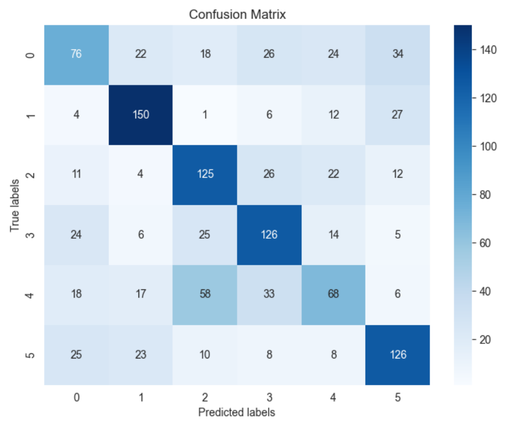
2.2.2 Using scikit-learn SVM function

- Validation Accuracy: 55.91 percent
- The validation accuracy is almost similar to what was obtained using CVXOPT
- The training time for LIBSVM was considerably smaller compared to CVXOPT(2 minutes vs 17 minutes)

2.2.3 Confusion Matrix and Observations



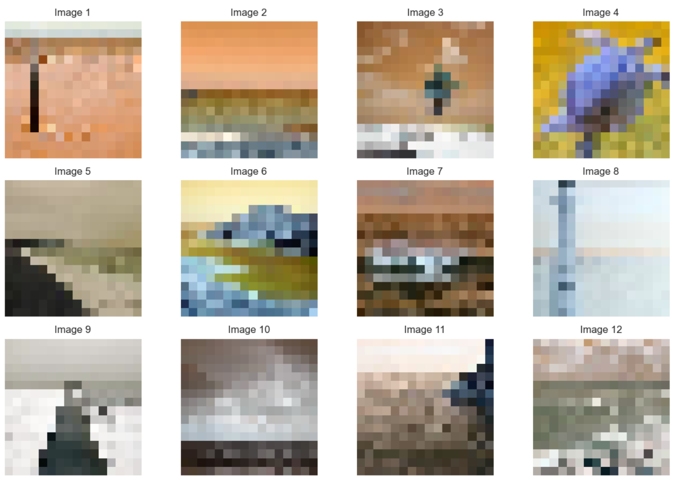
For CVXOPT:



For LIBSVM:

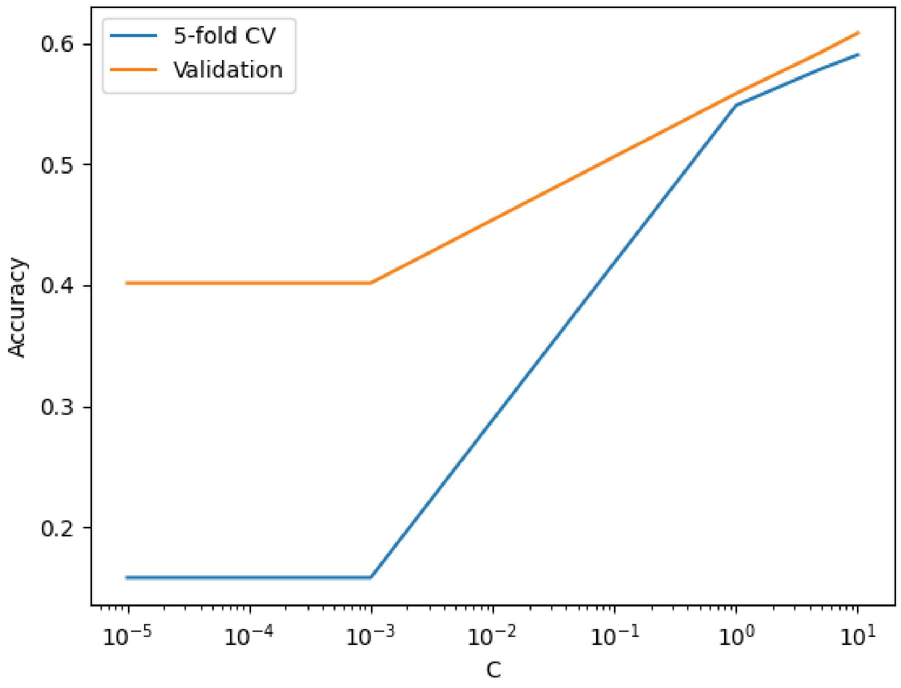
- We can see that 58 examples of class 4 are predicted as class 2, this is the highest number among classes being misclassified

Some misclassified images:



This misclassification makes sense. Th mapping for Intel image classification dataset is ’buildings’ - 0, ’forest’ - 1, ’glacier’ - 2, ’mountain’ - 3, ’sea’ - 4, ’street’ - 5. Classifying into glacier and sea is a more intricate task compared to others, as they are somewhat similar in images.

2.2.4 Validation and Cross-Validation



As is evident from the graph, higher values of C are better, thus 10 is the best value. The model achieves the highest cross-validation accuracy of 59.03 with C=10. The validation accuracy for this setting is 60.83.

It has the highest validation and cross validation accuracy.

$C = 1 \times 10^{-5}$ and $C = 0.001$ both lead to poor performance with a CV accuracy of 15.81% and a validation accuracy of 40.17%. This might indicate underfitting.

$C = 1$ has a CV accuracy of 54.85% and a validation accuracy of 55.83%.

$C = 5$ results in a CV accuracy of 57.87% and a validation accuracy of 59.25%.

As mentioned, $C = 10$ gives the best results.

There's an upward trend in performance as C increases from 1×10^{-5} to 10. However, after $C = 10$, the performance starts decreasing, suggesting that $C = 10$ might be an optimal value in this range.

Very low values of C (like 1×10^{-5} and 0.001) lead to poor performance, suggesting potential underfitting. The model might be too regularized and fails to capture the underlying patterns in the data.

Note that the cross validation accuracy for lower vlaues of C such as C=1e-5 is very less, it is almost the same as a random classifier. This is because the number of elements of each class might not be same in each of the folds. If the folds are kept uniform, we get higher cross validation accuracy for lower values of C(around 40 percent). This is called Stratified KFold.