COL 774 : Assignment 2

Aaveg Jain : 2021CS10073

October 2023

§ 1.0.Introduction

In this assignment we explore text classification using a Naive Bayes model, and also explore image classification using Support Vector Machines(SVM).

§ 2.0.Text Classification

Part a) **Naive Bayes model**

We implement the vanilla version of Naive Bayes model. We use the "bag of words" version, where we assume that for each position in the text, the occurrence of words obey a multinoulli distribution. We also incorporate Laplace smoothing with smoothening factor 1.

i) The accuracy over the training set is **84.29 %** and over the validation set is **66.44 %** .

ii) The word clouds for the Positive, Negative and Neutral classes are :

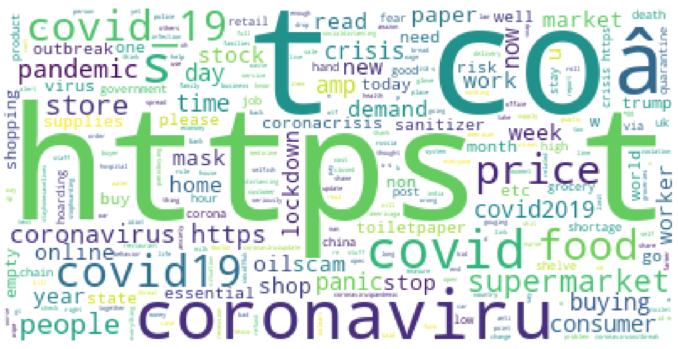


Figure 1: Positive Tweets

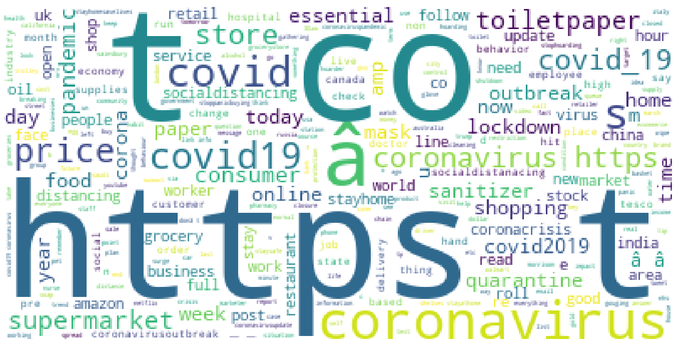


Figure 2: Neutral Tweets

Figure 3: Negative Tweets

Part b) **Comparison with random and all positive predictor**

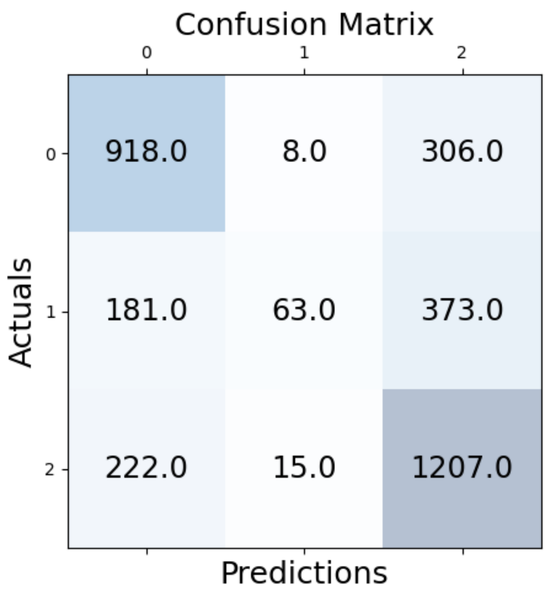
i) The validation accuracy we obtain by using a random predictor is **31.67 %**, close to 33.33 % as expected.

ii) The validation accuracy we obtain by using an all-positive predictor is **43.85 %**.

iii) Improvement over random and positive baseline are respectively : **39.90 %, 27.72 %** .

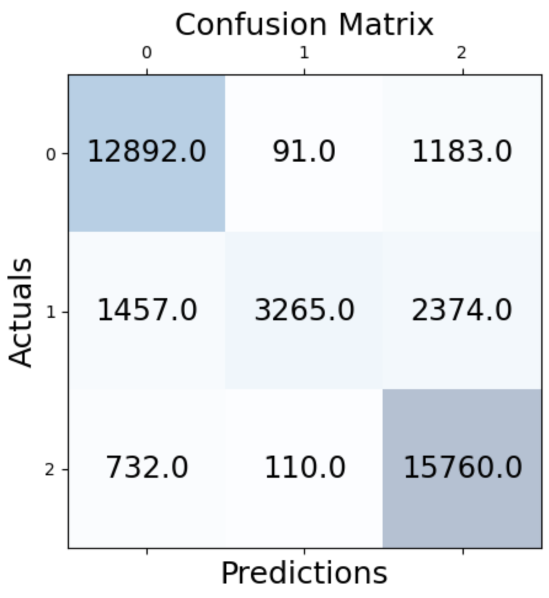
Part c) **Confusion matrices**

i) Confusion matrices for part a, b



	0	1	2
0	918.0	8.0	306.0
1	181.0	63.0	373.0
2	222.0	15.0	1207.0

Figure 4: Confusion matrix, Naive Bayes, Validation



	0	1	2
0	12892.0	91.0	1183.0
1	1457.0	3265.0	2374.0
2	732.0	110.0	15760.0

Figure 5: Confusion matrix, Naive Bayes, Training

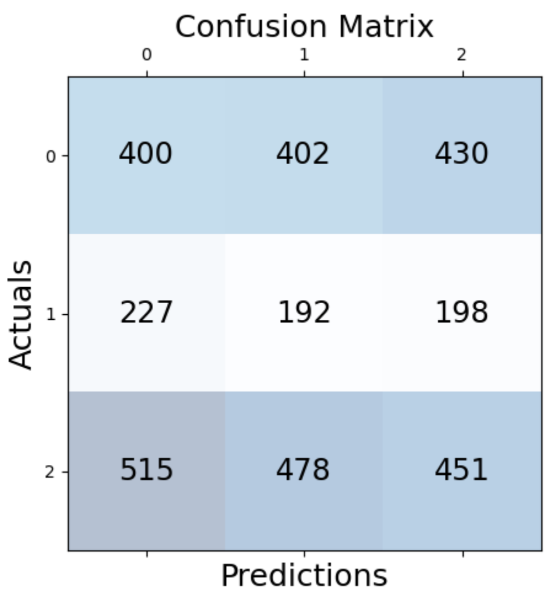


Figure 6: Confusion matrix, Random Predictor , Validation

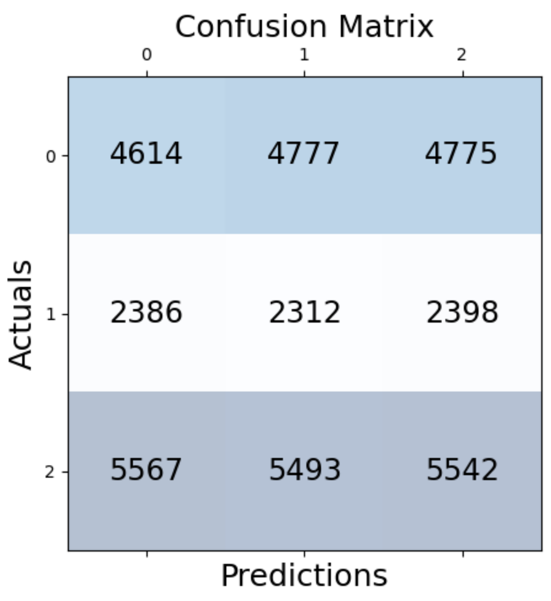


Figure 7: Confusion matrix, Random Predictor, Training

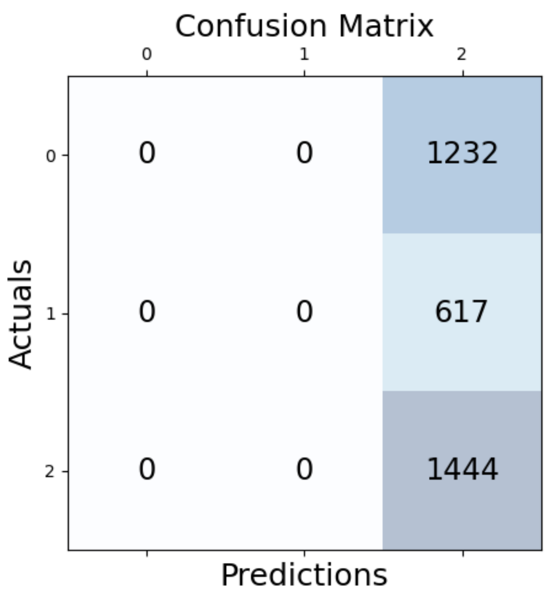


Figure 8: Confusion matrix, All Positive, Validation

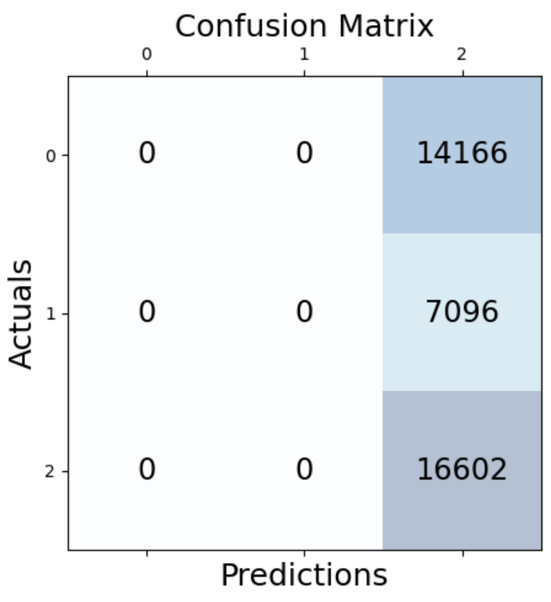


Figure 9: Confusion matrix,All Positive, Training

ii) For all cases, class 2 or "Positive" has the highest value of diagonal entry. This means that the maximum number of correct predictions have been made for "Positive" label.

Part d) **Preprocessing**

i) The preprocessing on the text has been done in the following order:

- We first got rid of links (http and https) as these mostly do not contribute to the meaning of the tweets.
- It was observed that the tweets had characters like Â which were present due to encoding issues. These were also removed.

- The text was then tokenized into tokens, splitting using whitespaces and treating punctuation as seperate tokens.
- The resulting tokens were then lemmatized (not stemmed as lemmatized resulted in slightly better accuracy and also overstemming may change the meaning of words)
- Finally stopwords and punctuations were removed.

ii) New word clouds obtained :

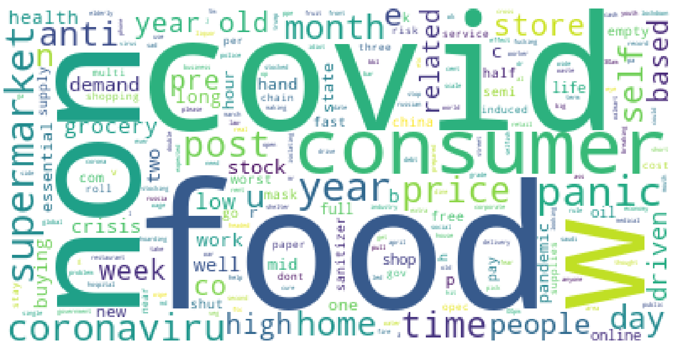


Figure 10: Positive Tweets

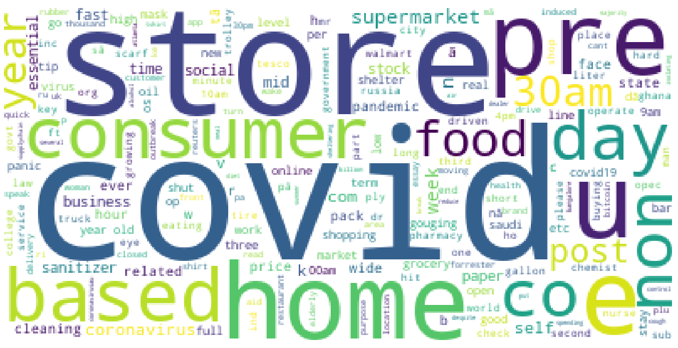


Figure 11: Neutral Tweets

store
pre
covid
consumer
food
day
based
home
coe
non
30am
year
supermarket
post
u

Figure 12: Negative Tweets

iii) A new model was trained which used preprocessing as given above, and the resulting validation accuracy is **69.69 %**.

iv) The new model performed significantly better than the previous version, with an increase of **3.25 %** over the baseline model. Also, the word clouds obtained have fewer non-sensical words like http and Â.

We can conclude that **lemmatizing, stop word and punctuation removal, and removing links** improve accuracy as this decreases noise and redundant words in the data, focusing more on sentiment-holding words.

Part e) **N-grams**

i) After training model based on bigrams, we get validation set accuracy as **67.41 %**.

ii) We add the following set of new features after preprocessing (not using bigrams) :

- We map all unknown words in the text (unknown means which are not present in the vocabulary from training data), to a new feature `UNK`, and set its probability of occurrence for the i^{th} class as $1/(N_i + |V|)$,(where V is the vocabulary and N_i is the total number of words occuring in all training texts of the i^{th} class)
- We use a simple lexicon AFINN, which is used to assign sentiment values to sentences, to get a set of words with a high positive sentiment value (≥ 3), and another set with a high negative value (≤ -3). We then increase the weight of these words in the training and prediction, by adding duplicates (2 additional copies) to the feature vector if they occur in the text. The idea behind this is to give more emphasis to words that hold high sentiment value.

These ideas combined push the validation accuracy to **71.57 %**.

iii) We have performed preprocessing in both bigrams model and in the above model.
We observe that bigrams model actually performs worse than in part d (decrease of **2.28 %**), whereas it performs better than the baseline version (increae of **0.77 %**). Thus, we can conclude that wrt to part d, bigrams is performing worse since it may be overfitting the model with the training data (interestingly the training data accuracy for bigrams is **94.3 %**), but adding of additional features helps it to perform better than the baseline.

After adding additional features as told in ii) (bigrams not used), the accuracy becomes better than part a) as well as part d) by **5.13 % and 1.88 %** respectively, a significant improvement. Thus adding the above set of features improves our accuracy. This can be due to the following facts :

- Earlier we were ignoring words which were not present in our vocabulary, but now we assign a positive probability to each of them. Effectively we are giving more preference to classes with lower N_i for i = 0,1,2. This helps in reducing bias in our model to those classes which occur more frequently than others.
- By giving more weight to words which hold high sentiment value, we are focusing less on words with lower sentiment value, and more on words with higher sentiment value. This is useful for predicting the sentiment of the tweet and hence improves accuracy.

Part f) **Domain Adaptation**

i) Using domain adaptation, the accuracies we get for using 1%, 2%, 5%, 10%, 25%, 50%, and 100% of the target domain training data are respectively : **41.75 %, 42.44 %, 43.37 %, 44.66 %, 46.98 %, 50.46 %, 53.08 %**.

ii) By learning from scratch we get accuracies (in same order as above) : **40.32 %, 39.53 %, 43.44 %, 45.76 %, 46.49 %, 50.00 %, 52.65 %**.

iii) Plot of accuracies for both parts above (alg1 denotes domain adaptation, alg2 denotes learning from scratch)

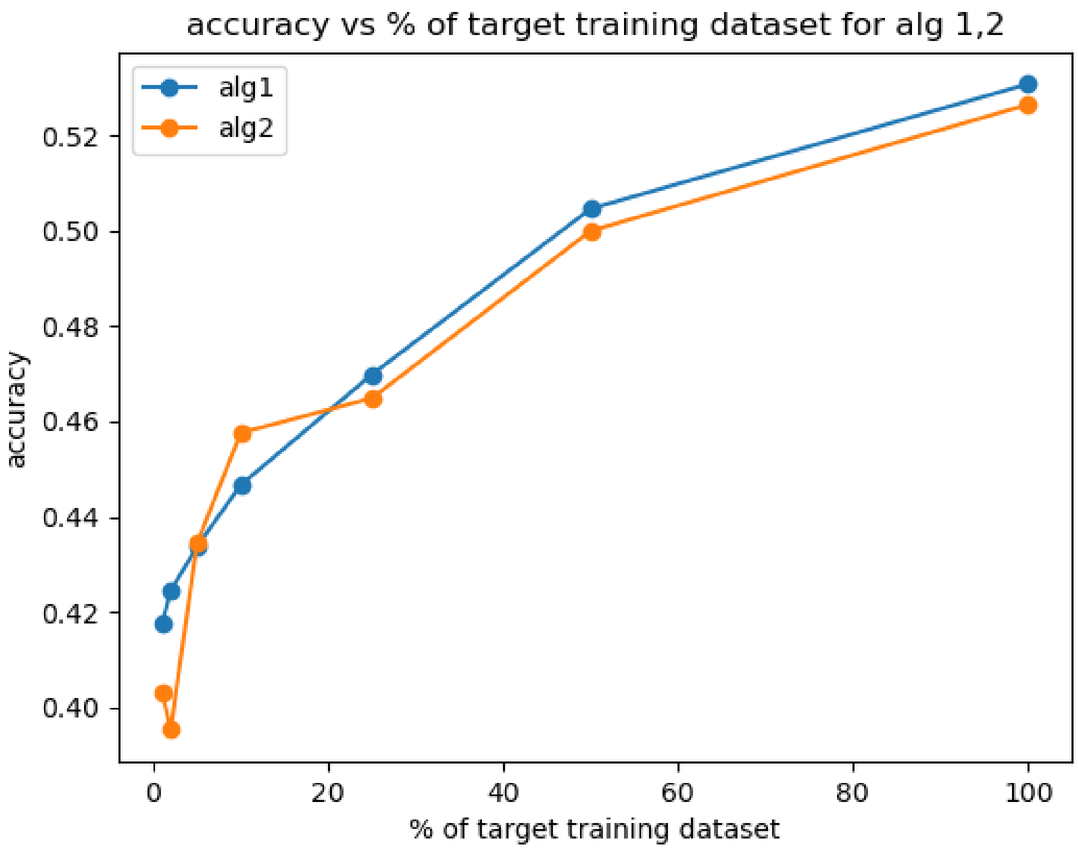


Figure 13: Domain adaptation vs Learning from scratch

iv) We observe that in general when we increase percentage of training data used from target domain, the validation accuracy on the target domain also increases.

We also observe that in almost all cases, domain adaptation gives better accuracy than learning from scratch. The percentage of training data from target domain represents how much data is available from target domain. Thus we observe that in almost all cases, using source domain training data makes up for insufficient target domain training data, and provides better accuracy than learning from scratch by leveraging the similarity between source and target domains.

§ 3.0.Binary Image Classification

We use the datasets 3 and 4 .

The SVM dual optimization problem is :

$$\min_{\alpha} \frac{1}{2} \sum_i \sum_j y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} > - \sum_i \alpha_i$$

$$\sum_i y^{(i)} \alpha^{(i)} = 0$$

$$C \geq \alpha_i \geq 0 \ \forall \ i$$

where i,j go from 1 to m = number of training examples, $< x^{(i)}, x^{(j)} >$ is the kernel of x$^{(i)}, x^{(j)}$, x$^{(i)}$ is the i^{th} training example, $y^{(i)}$ is the class of the i^{th} training example, C is the regularization parameter, and α_i is the weight of the $i^{(th)}$ support vector.

Expressing this in the form

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

$$G\alpha \succeq h$$

$$A\alpha = b$$

where

$$P_{ij} = y^{(i)}y^{(j)} < x^{(i)}, x^{(j)} > \text{, P is of shape m * m}$$

$$q_i = -1 \,\forall i, \text{q is of shape m * 1}$$

$$c = 0$$

$$G = \begin{bmatrix} -\mathbb{I} \\ C * \mathbb{I} \end{bmatrix}, \text{G is of shape 2m * m}$$

$$h_i = 0 \,\forall i = 1...m$$

$$h_i = C \,\forall i = m+1...2m$$

$$\text{h is of shape 2m * 1}$$

$$A = Y^T$$

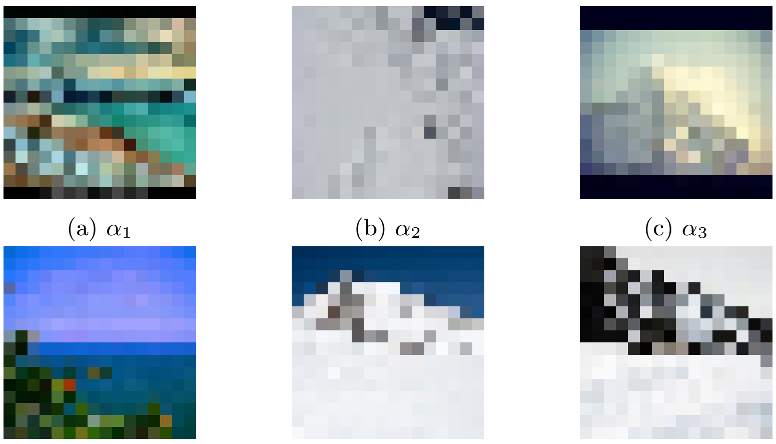
$$b = 0$$

Part a) **Linear Kernel (C = 1.0)**

i) We get **2904** support vectors out of 4760 training examples. We take a margin of 1e0-5 for values of alpha. The percentage of training samples is **61 %**.

ii) The bias term is **0.725** and the validation accuracy is **72.75 %**. The weight vector is not shown for brevity.

iii) Reshaping the top 6 coefficient support vectors, and the weight vector to 16 x 16 x 3, we get



(a) α_1 (b) α_2 (c) α_3

(d) α_4 (e) α_5 (f) α_6

Figure 14: Linear kernel top 6 support vectors

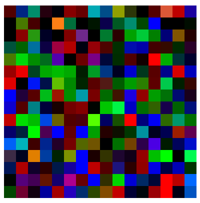


Figure 15: Linear Kernel Weight vector

Part b) **Gaussian Kernel (C = 1.0, $\gamma = 0.001$)**

i) We get **3453** support vectors, which is **72.54%** of the training set. There are **2698** support vectors common with linear kernel case **56.68 %**. Thus we get **549** more SV in this case **11.53 %**.

ii) We get the validation accuracy as **77.75 %**. The bias term is **-2.784**.

iii) Reshaping the top 6 coefficient support vectors to 16 x 16 x 3, we get

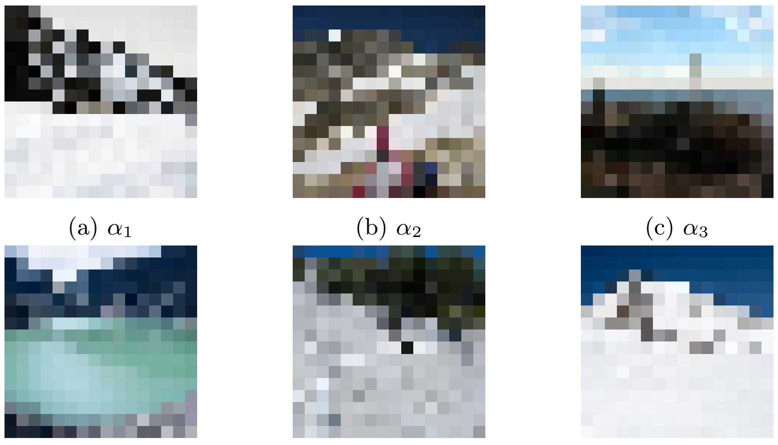

(a) α_1 (b) α_2 (c) α_3 (d) α_4 (e) α_5 (f) α_6

Figure 16: Gaussian kernel top 6 support vectors

iv) Validation accuracy, as compared to linear kernel (**72.75 %**) has an increase of **5%**.

Part c) **Comparison with Scikit SVM**

i) We state the nSV for all models till now :

- Number of support vectors for sklearn linear svm is **2899**.
- Number of support vectors for sklearn gaussian svm is **3398**.
- Number of support vectors for our linear svm is **2904**.
- Number of support vectors for our gaussian svm is **3453**.
- Number of support vectors common for linear svm are **2899**.
- Number of support vectors common for gaussian svm is **3398**.

  From the numbers, we conclude that the sv used in both of our implemented models is also used by the corresponding sklearn models, along with a few additional sv.

ii) The bias obtained in this case for linear kernel is **0.725**, the same as obtained by our model. The weight vectors are also almost identical, the percentage error using the norm of their difference being 0.247 %.

iii) The validation accuracy is **72.75 % and 77.75 %**, the same as obtained by our models. Hence our models have been implemented correctly.

iv) Training times :

- Time taken by our linear svm is **80 sec**.
- Time taken by our gaussian svm is **62 sec**.
- Time taken by sklearn linear svm is **12 sec**.
- Time taken by sklearn gaussian svm is **7 sec.**
  Hence there is a reduction of about 6 times in case of linear kernel, and about 9 times in case of gaussian kernel if we use scikit learn !!.

§ 4.0.Multi-Class Image Classification

Part a) **Using Gaussian kernel model implemented (C = 1.0, $\gamma = 0.001$)**

i) The validation accuracy is **53 %**.

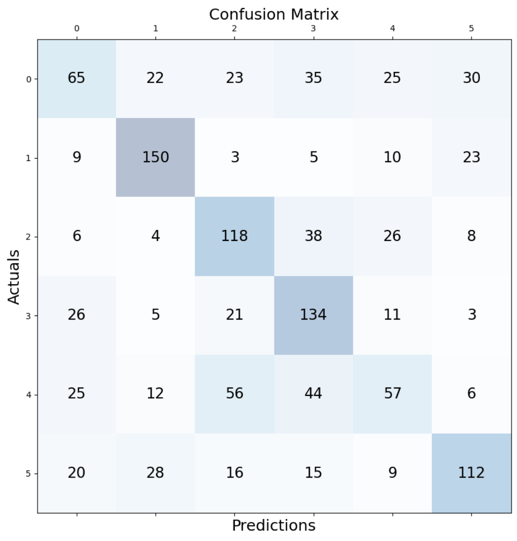
Part b) **Comparison with Scikit learn (C = 1.0, $\gamma = 0.001$)**

i) The validation accuracy is **55.833 %**.

ii) The validation accuracy for both are almost identical, with scikit learn svm being higher by **2.833 %**. This again shows that our model has been implemented correctly.
The times taken by scikit and our model are respectively **82 sec and 19 min** . Thus scikit learn svm takes around **14 times** less than our implementation.

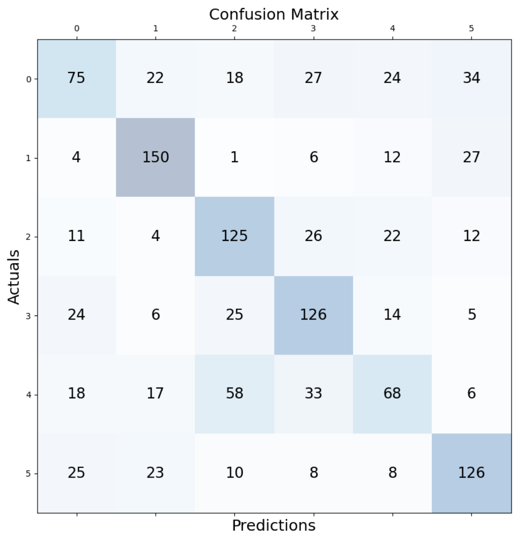
Part c) **Confusion matrices and visualisation of misclassified images**

The confusion matrices for the above two cases are


Confusion Matrix
Actuals
Predictions

Actuals \ Predictions	0	1	2	3	4	5
0	65	22	23	35	25	30
1	9	150	3	5	10	23
2	6	4	118	38	26	8
3	26	5	21	134	11	3
4	25	12	56	44	57	6
5	20	28	16	15	9	112

Figure 17: Our Multiclass model

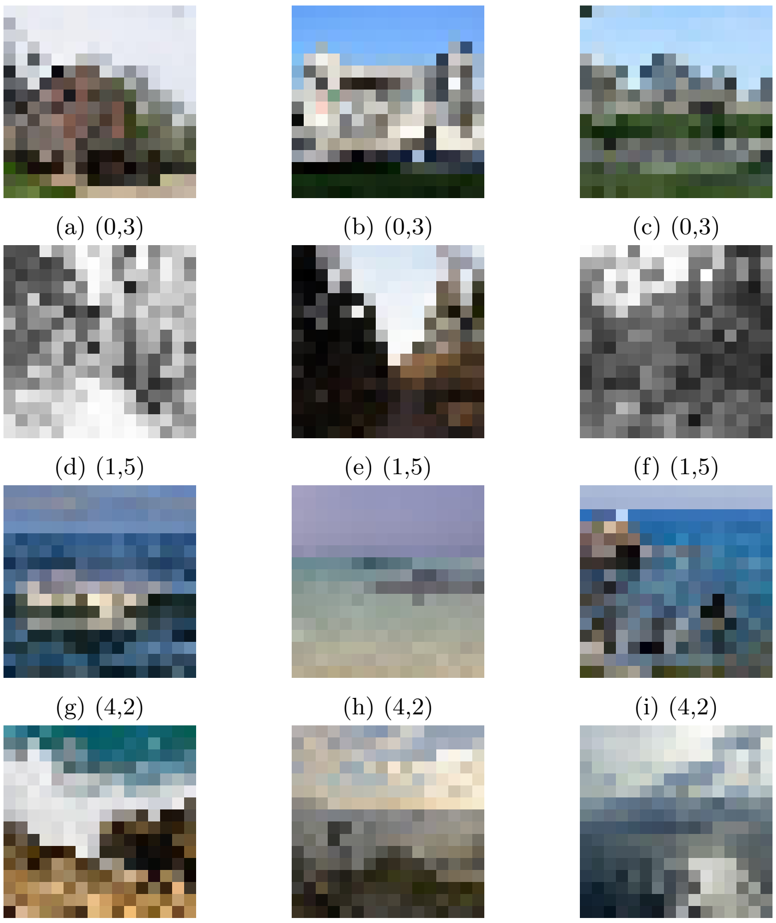

Confusion Matrix
Actuals
Predictions

Actuals \ Predictions	0	1	2	3	4	5
0	75	22	18	27	24	34
1	4	150	1	6	12	27
2	11	4	125	26	22	12
3	24	6	25	126	14	5
4	18	17	58	33	68	6
5	25	23	10	8	8	126

Figure 18: Sklearn SVM Multiclass model

Observations : set 0 has images of houses and buildings, 1 has of forests and greenery, 2 has of ice and glaciers, 3 has of cloudy sky and mountains, 4 has of coasts and oceans, 5 of cities.

Of these the **most frequent pair of misclassified classes are (first is actual, second is prediction) : (4,2), (4,3), (0,3), (1,5).**



(a) (0,3) (b) (0,3) (c) (0,3)

(d) (1,5) (e) (1,5) (f) (1,5)

(g) (4,2) (h) (4,2) (i) (4,2)

(j) (4,3) (k) (4,3) (l) (4,3)

Figure 19: 12 examples of misclassified objects among most misclassified pair of classes

Based on the above observations, it is easy to make sense of the visualisations of the misclassified examples as shown. For eg - images of 4 and 2 both are predominantly white and blue, thus they are misclassified as each other the most. Similiar logic holds for 4 and 3, and so on.

Part d) **Cross Validation ($\gamma = 0.001$)**

i) For each value of C in { 10^5, 10^3 , 1, 5, 10 }, we get the cross validation accuracy and accuracy(in %) as { 15.95, 15.95, 55.09, 57.95, 59.15 } and { 40.17, 40.17 55.92, 59.25, 60.83 } respectively.

ii) Plot of cross validation accuracy and validation accuracy versus log C :

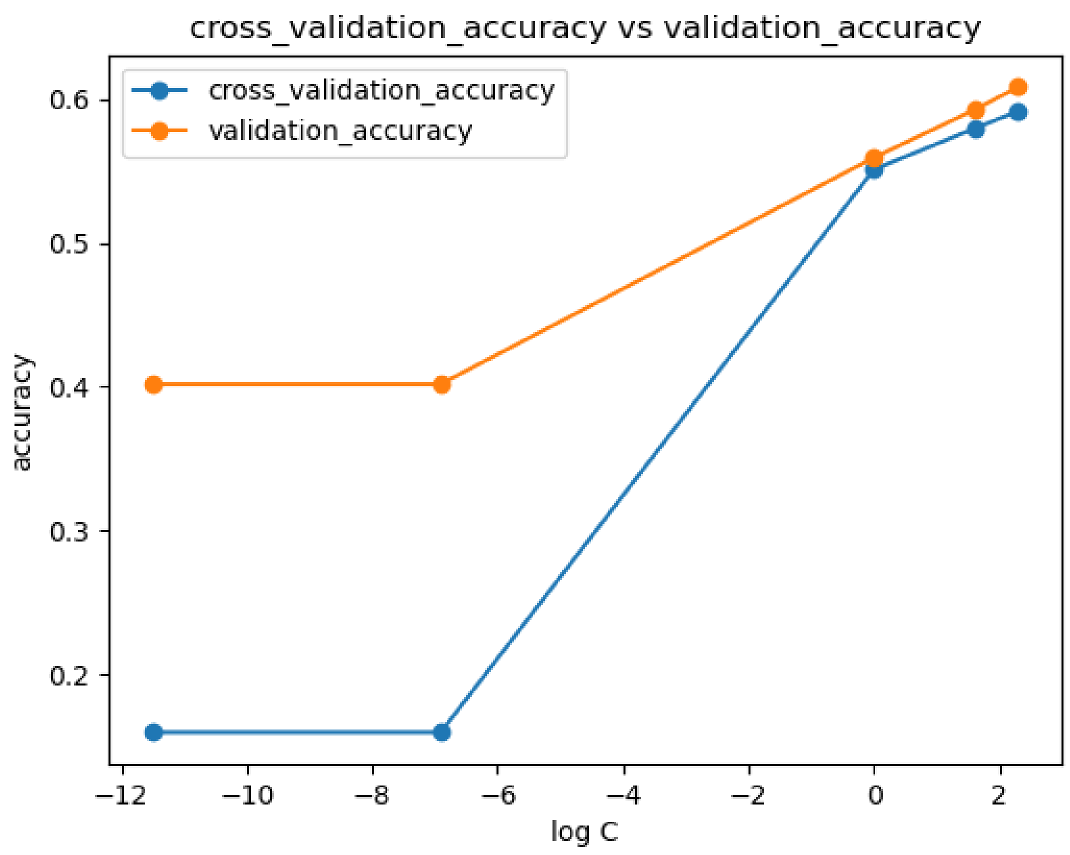


Figure 20: Cross validation accuracy and validation accuracy against log C

Observations : We observe that the value of C which gives the best cross validation accuracy is also the same which gives the best validation accuracy. In fact as the cross validation accuracy (CVA) increases, the validation accuracy (VA) also increases with C. Hence C = 10 gives best value of both accuracies.

Thus cross validation accuracy serves as a good measure of the generalized accuracy, i.e we can say for our case that if a model with C = C1 has better cross validation accuracy than model with C = C2, then its generalized accuracy is also expected to be better.