Japneet Singh 2019MT10696 Semester 1 2022-23

Report: COL774 Assignment 2 (Naïve Bayes & SVM)

Part 1. Text Classification

The dataset for this problem is the Large Movie Review Dataset. Given a movie review, the task is to predict the sentiment of the review as either positive or negative. We have been provided with a subset of the Large Movie Review Dataset, with the training and test splits containing 25,000 reviews (samples) and 15,000 reviews respectively. A review comes from one of the two categories (class labels) — **positive** or **negative**.

(1a) Simple Naïve Bayes

Minimal pre-processing of the reviews was done. The reviews were tokenised and special characters were removed.

Vocabulary dictionaries are created separately for positive and negative reviews, as the sample data give to us is separated itself.
Laplace smoothing is done to account for words not seen in the training set appearing in the test set, this gets rid of any zero probability.

The model will 2|V| + 1 parameters, i.e., the following:

$$\theta_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

$$\theta_{k|y=1} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1 \wedge x_j^{(i)} = 1\} + c}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} + 2c}$$

$$\theta_{k|y=0} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0 \wedge x_j^{(i)} = 1\} + c}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} + 2c}$$

$\theta_{k|y=1}$ models the probability of a word being in the review given the fact that it is positive. Similarly, $\theta_{k|y=0}$ models the probability of a word being in the review given the fact that it is negative. These. 2 parameters are present for each word in the vocabulary, given 2|V| such parameters.

As given in the problem statement, Laplace Smoothing Coefficient c is taken to be 1.
During test time, we take log and add the log-terms to prevent underflow.

 # List of libraries used is given on the last page

Accuracy over Training Set was obtained as: **91.144%**
Accuracy over Test Set was obtained as: **82.613%**
Further obtained measurements were,
Test Set Precision: 0.9229800869764249
Test Set Recall: 0.8065
Test Set F1: 0.8608175899242182

Find the confusion matrix below, **note: 0 is the Positive Class, 1 is the Negative Class**

Figure 1 – Confusion Matrix for Simple Naive Bayes

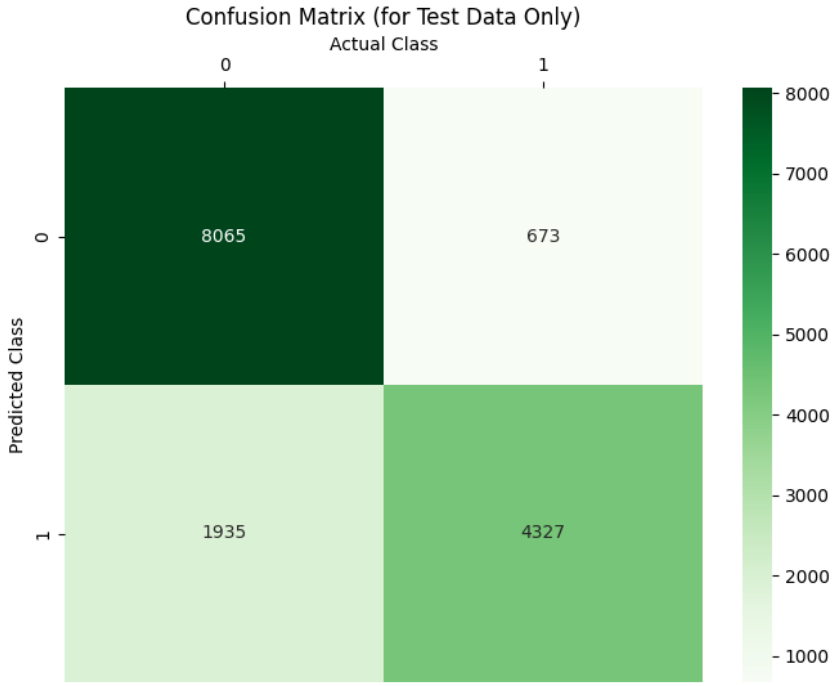


Figure 3 - Word Cloud for Positive Reviews

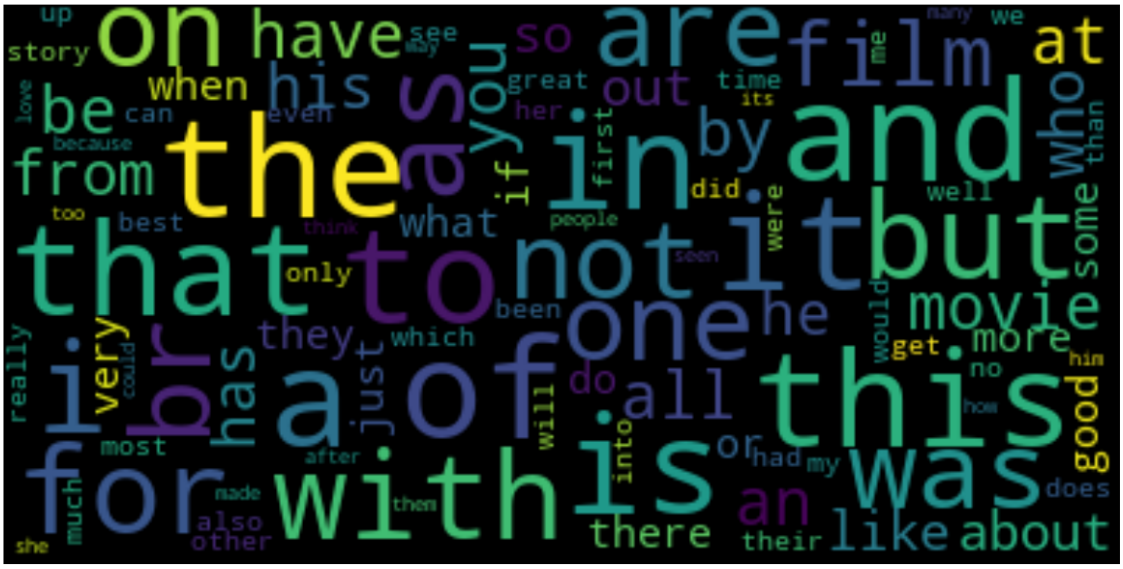


Figure 2 - Word Cloud for Negative Reviews

(1b) Random and Fixed Guessing

In Random Guessing, we guess Reviews to be Positive or Negative with equal probability.
Test Set Accuracy by Random Prediction: **49.406666666666666%**

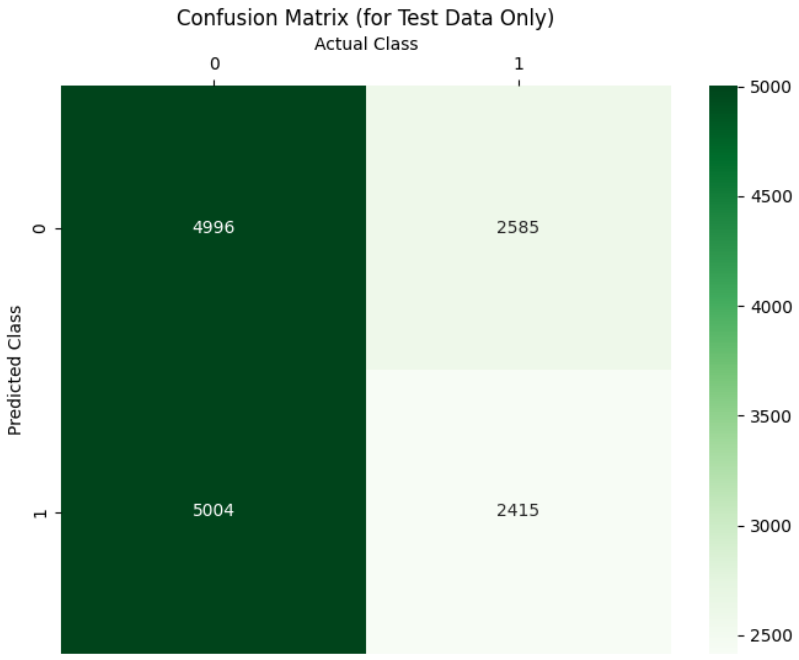


Figure 4 - Confusion Matrix for Random Guessing Class Prediction

Next, we try the model where we always predict the review to be positive.
Test Set Accuracy by Always Predicting Positive: **66.66666666666666%**

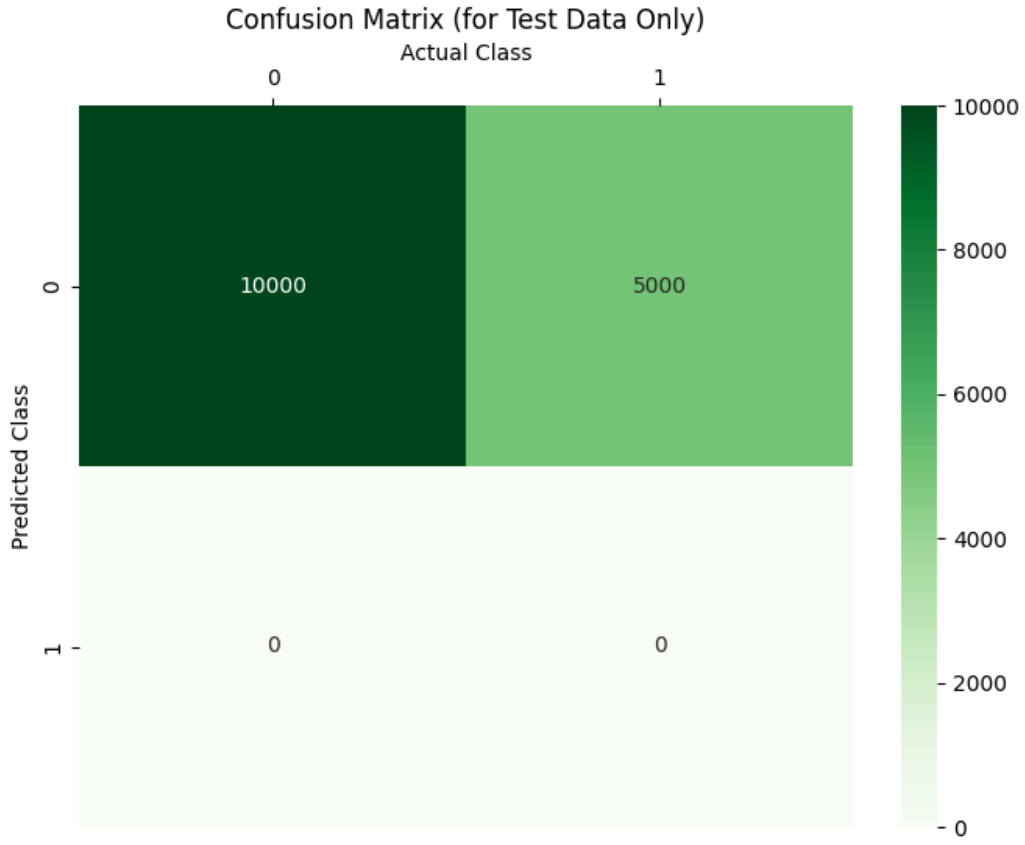


Figure 5 - Confusion Matrix for Always Predicting Positive

Hence, we can see:
Our model in (1a) gives an improvement of –
33.2 percentage points over the random guessing model's baseline (1.67x improvement) and
15.95 percentage points over the model which always predicts positive (1.24x improvement)

(1c) Confusion Matrices

Please note, in the following confusion matrices, **0 corresponds to positive class and 1 corresponds to negative class.**

For (1a) the confusion matrix is given below,
Positive reviews have a much higher diagonal entry than the negative reviews in all 3 confusion matrices. This also results from the fact that there are 10k positive reviews in the 15k test set. 13% of the negative reviews have been misclassified as positive and 19% of the positive reviews have been misclassified as negative. This indicates that negative reviews have a much distinct vocabulary making them easier to classify correctly.

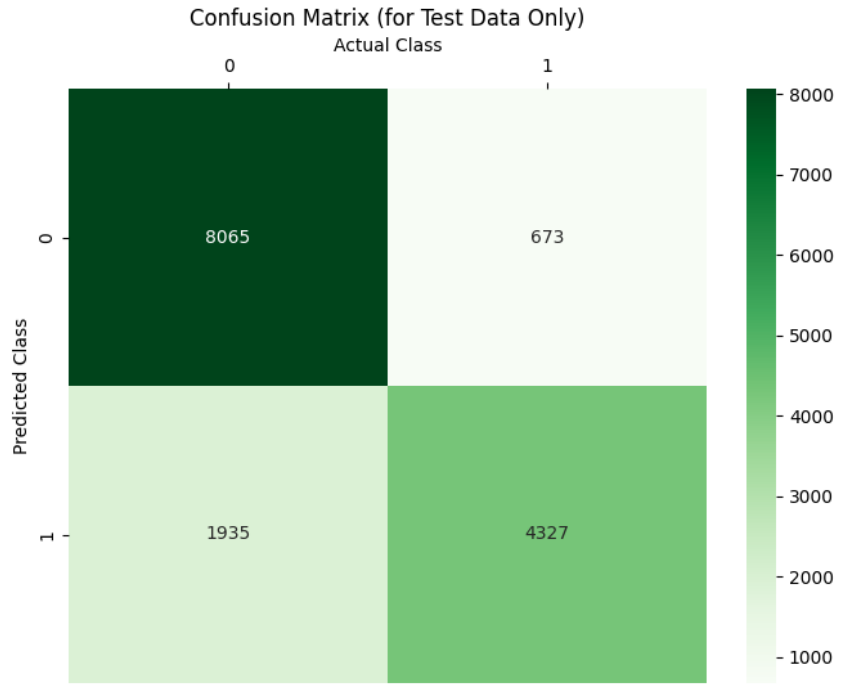


Figure 6 - Confusion Matrix for (1a) Simple Naive Bayes

Positive reviews have a much higher diagonal entry than the negative reviews in all 3 confusion matrices.

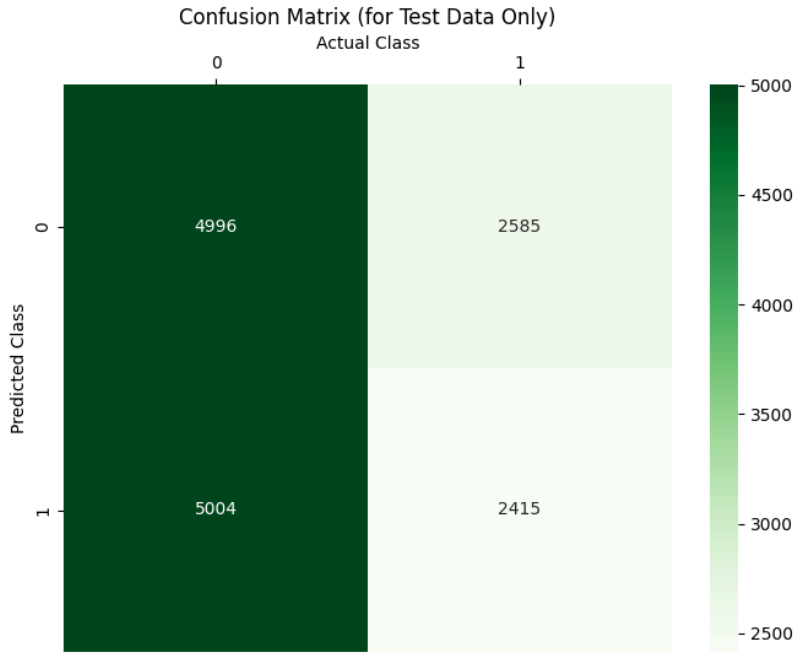


Figure 7 - Confusion Matrix for (1b) Random Predictions

In random predictions, we see a much more regular trend, almost equal number of test samples have been classified positive and negative.
Diagonal values of positive is higher than negative but the proportion of the reviews classified correctly is the same for both positive and negative classes (~50% each).

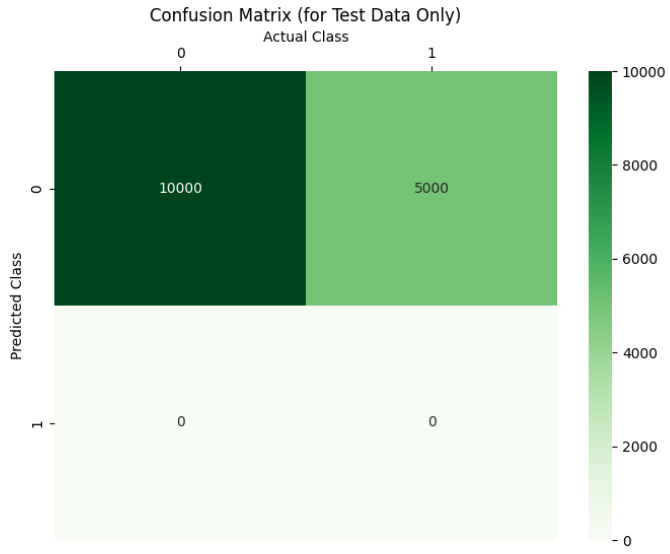


Figure 8 - Confusion Matrix for (1b) always predicting positive

For always predicting positive cases, true negatives are zero – which is inline with the model. All 10k positive reviews in the test set get classified correctly, giving us the accuracy we obtain.

Positive reviews have a much higher diagonal entry than the negative reviews in all 3 confusion matrices.

We observe that the sum of numbers in each columns equals the number of test samples for that class. Here, in all 3 matrices, sum of column 0 is 10k and column 1 is 5k – the number of positive and negative reviews in the test set respectively. The sum of numbers in each rows is the number of predictions made for each class. Row 0's sums give the number of test samples predicted to be in class 0 and so on.

We also observe that as the accuracy of the model increases, more and more fraction of the test sets gets included in the diagonal elements. For a perfectly accurate model, the confusion matrix would be a diagonal matrix – i.e., all classifications will be correct / actual class = predicted class for all samples.

(1d) Naïve Bayes with **Stop Word Removal** and **Stemming**

Here, we pre-process the reviews much more. In addition to the work done in (1a), we have performed stop word removal and also stemmed the remaining words in the review to group similar root words together.

Figure 9 - Negative Reviews Word Cloud for (1d)

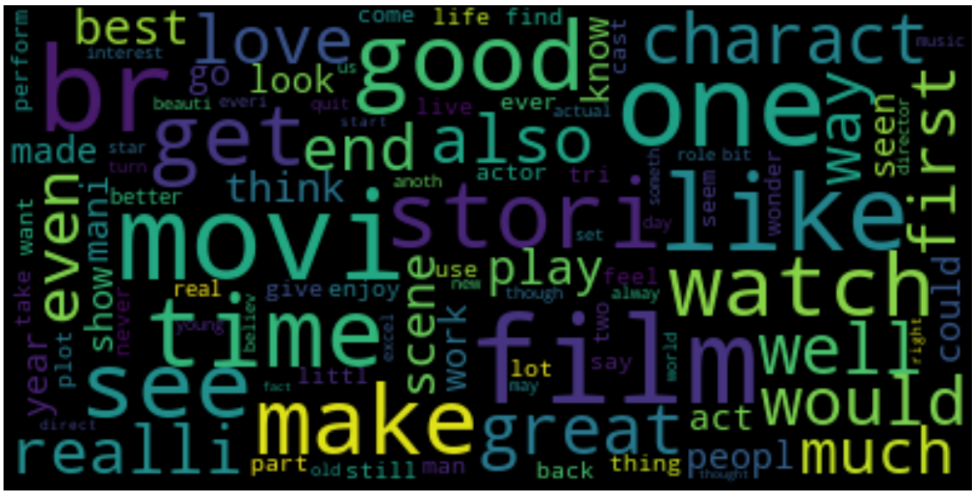


Figure 10 - Positive Reviews Word Cloud for (1d)

Accuracy over Training Set was obtained as: **90.096%**
Accuracy over Test Set was obtained as: **84.04%**
Further obtained measurements were,
Test Set Precision: 0.9032018659881256
Test Set Recall: 0.8519
Test Set F1: 0.8768011527377522

Accuracy over the test set has increased by 1.41% over the model in (1a).
We see that removal of insignificant words like 'the', 'and', 'is' etc does improve the test accuracy as it results in a more differentiated vocabulary for positive and negative reviews. The train accuracy however, slightly goes down I this. Case from (1a).

(1e) Feature Engineering

i. Unigrams + Bigrams

We include pairs of consecutive words in the vocabulary hoping that this would add more context of the reviews to the vocabulary, making classifying correctly much more probable.
The statistics obtained were:
Accuracy over Training Set was obtained as: **99.80%**
Accuracy over Test Set was obtained as: **86.34%**
Further obtained measurements were,
Test Set Precision: 0.9281714408787421
Test Set Recall: 0.8619
Test Set F1: 0.8938089806076946

ii. Additional Features

a. Unigrams + Bigrams + Trigrams

We now include sets three consecutive words in the vocabulary hoping that like bigrams, this would too give the model more context of the reviews, making classifying correctly even easier.
The statistics obtained were:
Accuracy over Training Set was obtained as: **97.564%**
Accuracy over Test Set was obtained as: **70.16%**
Further obtained measurements were,
Test Set Precision: 0.9592617226471566
Test Set Recall: 0.5769
Test Set F1: 0.720494567253653

b. Unigrams + Trigrams

We now use only trigrams and unigrams (single words like (1a) and (1d)) in the vocabulary to see how the model behaves.
The statistics obtained were:
Accuracy over Training Set was obtained as: **74.26%**
Accuracy over Test Set was obtained as: **56.44%**
Further obtained measurements were,
Test Set Precision: 0.9466632311259985
Test Set Recall: 0.3674
Test Set F1: 0.5293566745911678

<u>Some comments:</u>

Adding bigrams helps improve the model a lot, as we can see in the different measures given in the table below, (unigram + bigram) model performs better than the previous too in all aspects.

Adding trigrams however, seems to overfit the model, the train accuracy for (unigram +bigram + trigram) is very high but the test accuracy falls way below the previous three models indicating severe overfitting. The other parameters for this model also indicate the same, namely very low recall and F1 score.

I also tried the (unigram + trigram) model, which performed much much worse than the unigram + bigram model, indicating severe overfitting here as well. The train accuracy is much lower than the previous models and the test accuracy falls off a cliff here.

Hence, just adding new features may not always be a good idea – the kind of features we add or remove makes a huge difference to the performance of our model.

iii. Comparison of various feature sets

MODEL DESCRIPTION	TRAIN SET ACCURACY	TEST SET ACCURACY	PRECISION	RECALL	F1 SCORE
Unigrams only	0.91	0.82	0.92	0.80	0.86
Unigrams only (with stop word removal and stemming)	0.90	0.84	0.90	0.85	0.87
Unigrams + Bigrams	0.99	0.86	0.92	0.86	0.89
Unigrams + Bigrams +Trigrams	0.97	0.70	0.95	0.57	0.72
Unigrams + Trigrams	0.74	0.56	0.94	0.36	0.53

The best performing model for us is the (Unigram +Bigram) model, its measurements are highlighted in yellow in the table above.
Here, the higher accuracy means higher number of true positives and true negatives being classified. False-Positives or False-Negatives here do not have a high cost, (compared to cancer diagnosis cases for example) hence, we might want to have a higher number of correct classifications (true positives and true negatives).
F1 score becomes really important when the distribution of classes is really different – occurrence of one class is much less than the other (again, cases like cancer diagnosis).
In this case, we can consider the accuracy to be a good measure however, looking at the other values like Precision, Recall and F1-score is also a good practice. A really low value for either of these will indicate something wrong with our model.
For the (Unigram + Bigram) case, all the measurements are in good, high 80+ ranges and the model can be considered as our best one out of all our different attempts.

Part 2. Binary Image Classification

Note: My entry number is 2019MT10696, hence, the results are for the classification of class 1 from class 2
The training, validation and test set have been sampled from the original test set according to the classes 1 and 2.
For modelling, class 1 was labelled -1.0 and Class 2 was labelled 1.0 and all results have been stated accordingly.

For classification, we used SVM with a soft-margin i.e. allowing some data points to be on the wrong side of the margin or inside the margin.

The parameter C is set to 1.0 in this formulation which sets the importance/weightage of the penalty of the outliers in the data to the decision boundary.

$$\min \frac{1}{2} w^T w + C \sum_i \xi_i$$

such that $$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i \; ; \; \xi_i \geq 0 \; ;$$

Dual form gives us:

$$\max W(\alpha)_\alpha = \sum_{i=1}^{m} \alpha_i - 0.5 \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)} x^{(j)} >$$

such that $0 \leq \alpha_i \leq C \; ; \; \sum \alpha_i \, y^{(i)} = 0 \; ; \quad w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \; ; \quad b = Y - W^T X$

(2a) CVXOPT with Linear Kernel

1543 support vectors were obtained (keeping a threshold of 1e-4 to choose among the output alpha values). This constitutes to around 38% of the samples as support vectors.

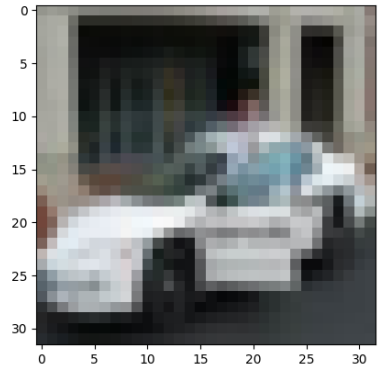
Bias/**b** was found to be 0.07047682389961614

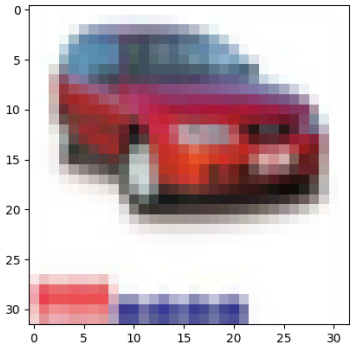
The weights array obtained was (printed fully in the terminal output)

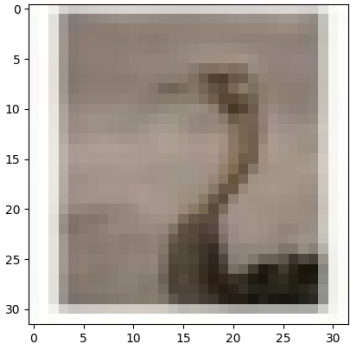
w [[-0.74568524]
[0.03275555]
[-0.4837071]
...
[-0.09054527]
[0.09773743]
[-1.04533169]]

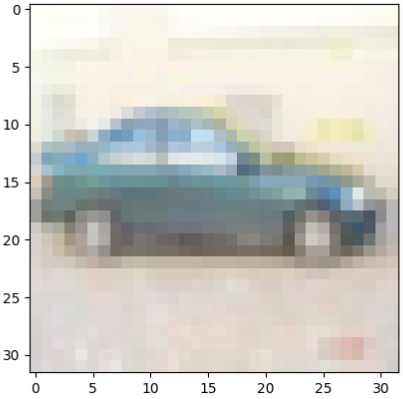
The accuracies were found to be –
Train Set: 94.95%
Test Set: 78.15%

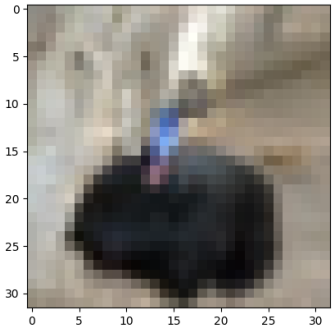
Top 5 Coefficients Plots –











(2b) CVXOPT with Gaussian Kernel

1872 support vectors were obtained (keeping a threshold of 1e-4 to choose among the output alpha values). Around 46.8% of the samples are support vectors.

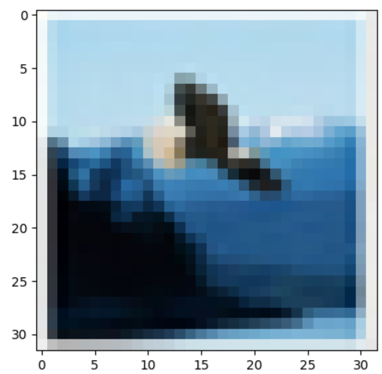
1161 of these support vectors were common with the support vectors in part (2a).

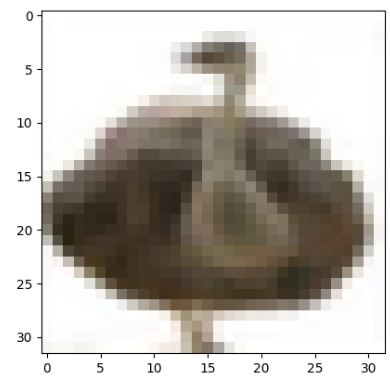
Bias/**b** was found to be -6.086002620715735

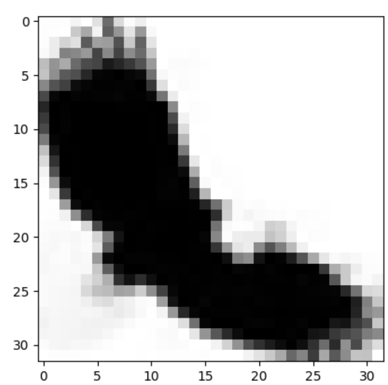
The accuracies were found to be –
Train Set: 89.35%
Test Set: 87.5%

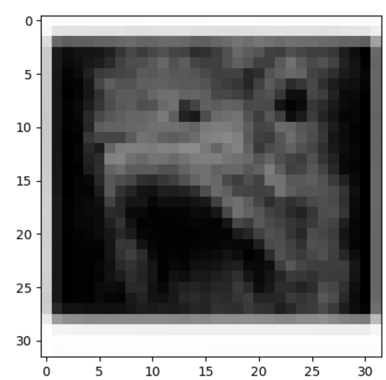
Train accuracy went down while using Gaussian Kernel as compared to Linear Kernel. However, **Test accuracy increased by over 9 percentage points!**
This result is consistent with the Scikit-learn implementation's results as well!
This might suggest that the test data is better suited to the Gaussian Kernel rather than being more linearly separable using a linear kernel.

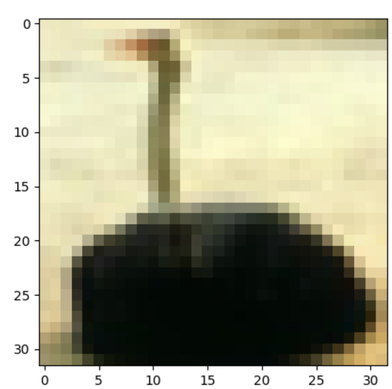
Top5 Coefficient Plots –











(2c) scikit-learn SVM – Linear and Gaussian Kernels

i. Linear Kernel

1541 support vectors were obtained (compared to **1543 in (2a) above**). This constitutes to around 38% of the samples as support vectors.
1533 support vectors were common with (2a) case

Bias/**b** was found to be 0.01147231

The weights array obtained was (printed fully in the terminal output)
w [[-0.7448432]
[0.0337600]
...
[-0.09044527]
[0.09674643]
[-1.04531148]]

The accuracies were found to be –
Train Set: 94.85%
Test Set: 78.05%

ii. Gaussian Kernel

1865 support vectors were obtained (compared to **1872 in (2b) above**). Around 46% of the samples are support vectors.
1812 support vectors were common with (2b) case

Bias/**b** was found to be -6.08461227

The accuracies were found to be –
Train Set: 89.325%
Test Set: 87.4%

Computational Cost Comparison:

Time taken to solve using CVXOPT / LINEAR KERNEL: **38.64611196517944s**
Time taken to solve using CVXOPT / GAUSSIAN KERNEL: **46.045270919799805s**

Time taken to solve using SKL SVM / LINEAR KERNEL: **28.526849031448364s**
Time taken to solve using SKL SVM / GAUSSIAN KERNEL: **18.71180009841919s**

Part 3. Multi Class Image Classification

In this part we make 5C2=10 One-vs-One SVM Classifiers using Gaussian Kernels (each similar to parts (2b) and (2c) above) to perform Multi-Class Classification. The class receiving the most votes out of the 10 classifiers will be the one the test input is finally classified as.

(3a) CVXOPT Gaussian Kernel One-vs-One Classification

The **Test Set Accuracy** was found to be **59.04%**.
Time taken to train the 10 classifiers was: 410.8 seconds.

(3b) scikit-learn SVM Gaussian Kernel One-vs-One Classification

The **Test Set Accuracy** was found to be **59.30%**.
The test set accuracy is almost the same (just .26% higher) as that of the CVXOPT implementation with Gaussian Kernel.
Time taken to train the 10 classifiers was: 157.6 seconds. (~2.5x less than CVXOPT case)
The less time taken to learn in the sklearn implementations is consistent with the results obtained in Part 2. We can conclude that sklearn was around 2.5x faster to train the same model than using CVXOPT on the same hardware configurations (all computations performed locally on an intel i7 8^{th} gen machine)

(3c) Confusion Matrices

Both the Confusion Matrices i.e., for part (a) using the CVXOPT package and for part (b) using the sklearn package have similar patterns. Classes 0 and 1 are classified correctly the most. Classes 2 and 4 are classified incorrectly among each other the most i.e., Class 2 is often misclassified as Class 4 and vice versa.
Class 3 is also sometimes misclassified as class 2 or 4 but not as much as the number of misclassifications between Class 2 and Class 4.
The second confusion matrix, obtained. Using the sklearn implementation gives the same trends as well.

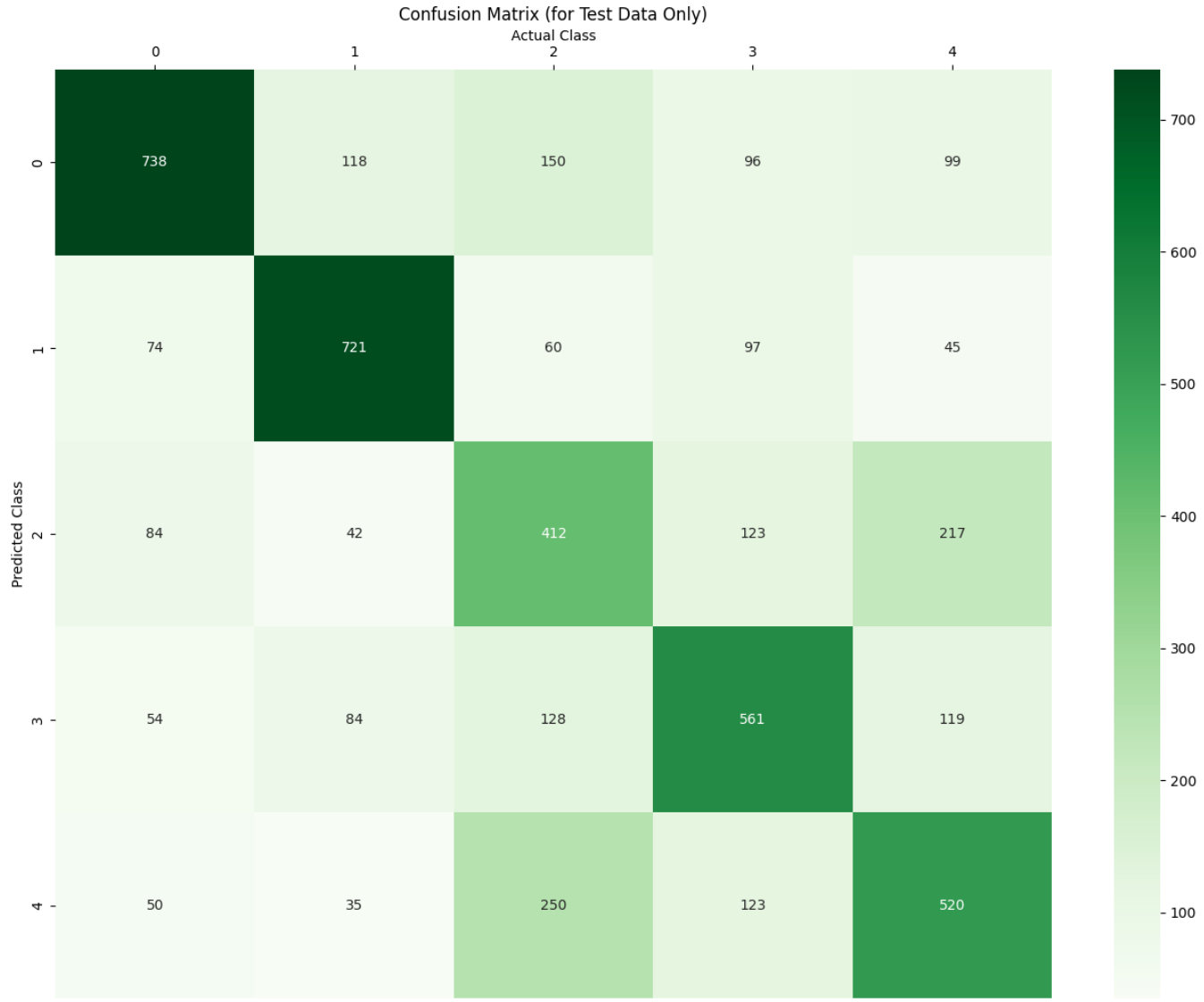


Figure 11 - Confusion Matrix for Multiclass Classifier using CVXOPT + Gaussian Kernel

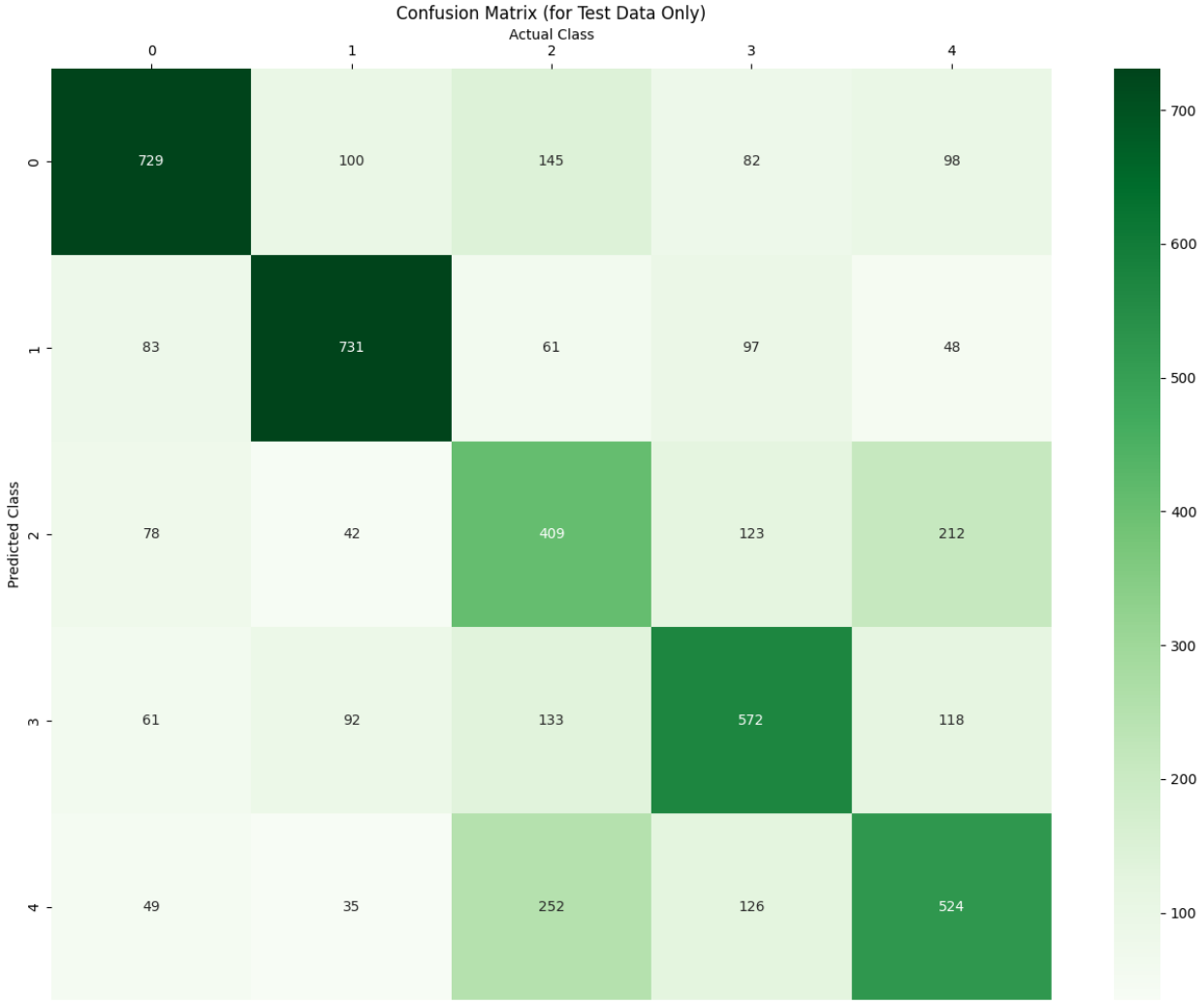
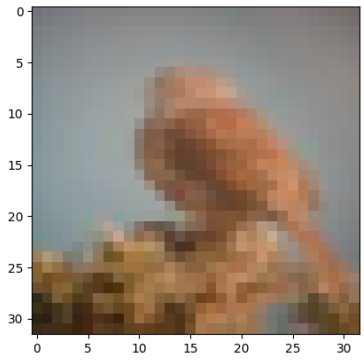
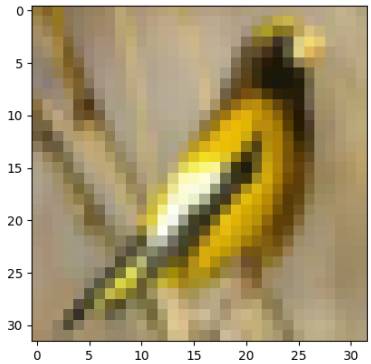
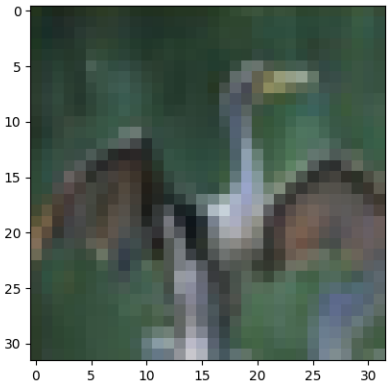
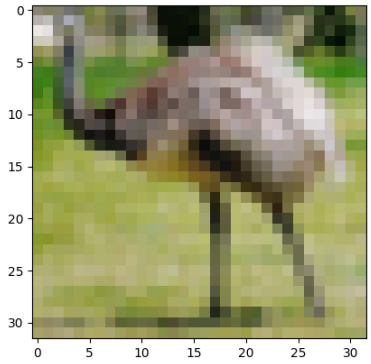
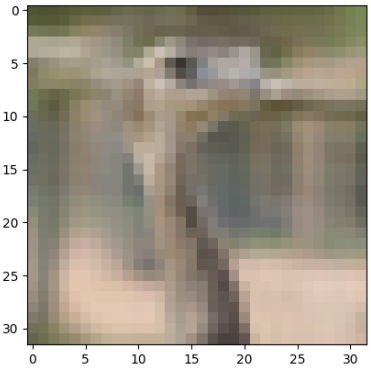


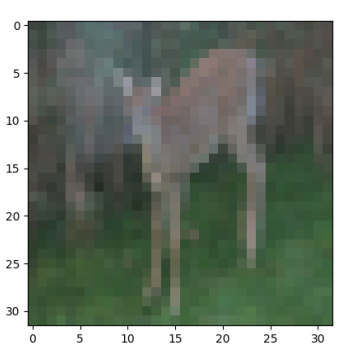
Figure 12 - Confusion Matrix for Multi Class Classifier using sklearn SVM + Gaussian Kernel

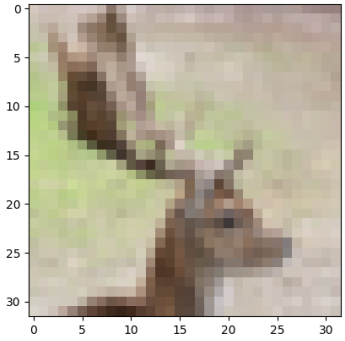
5 samples belonging to Class 2 misclassified as Class 4

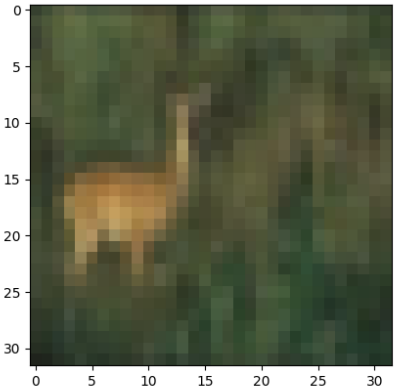






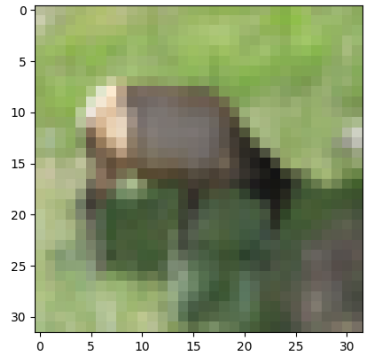
Class 2 is the class of birds, misclassified as 4 legged animals i.e. class 4. It is hard to make sense of the misclassification of the yellow bird. For the others, the colours scheme is very similar to the dominant colour scheme in class 4 and resemblance of some features to features in animals (only head visible in 2 of these, open wings maybe mistaken for legs and torso of the deer and so on) can also be seen. Hence, the misclassification can be made sense of.

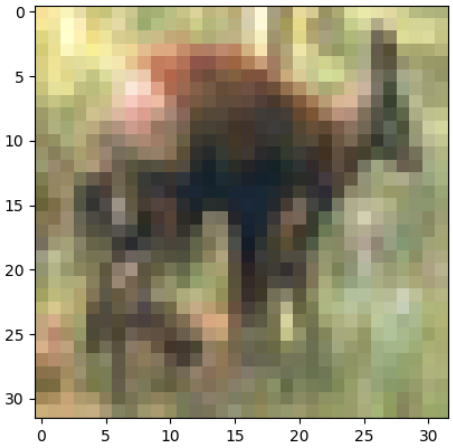
5 samples belonging to Class 4 misclassified as Class 2











Class 4 is the class of deer-like 4 legged animals. Due to the low resolution of the images, the neck and antlers in one image of the animal can be easily mistaken as the beak of the bird. Misclassification can be easily explained and made sense of in these cases.

(3d) 5-fold cross validation

$$\gamma = 0.001$$

C =>	1e-5 (1 in chart)	1e-3 (2 in chart)	1 (3 in chart)	5 (4 in chart)	10 (5 in chart)
5-FOLD CROSS VALIDATION ACCURACY	21.32%	20.55%	57.62%	60.29%	60.78%
TEST ACCURACY	37.72%	37.6%	59.3%	61.82%	62.6%

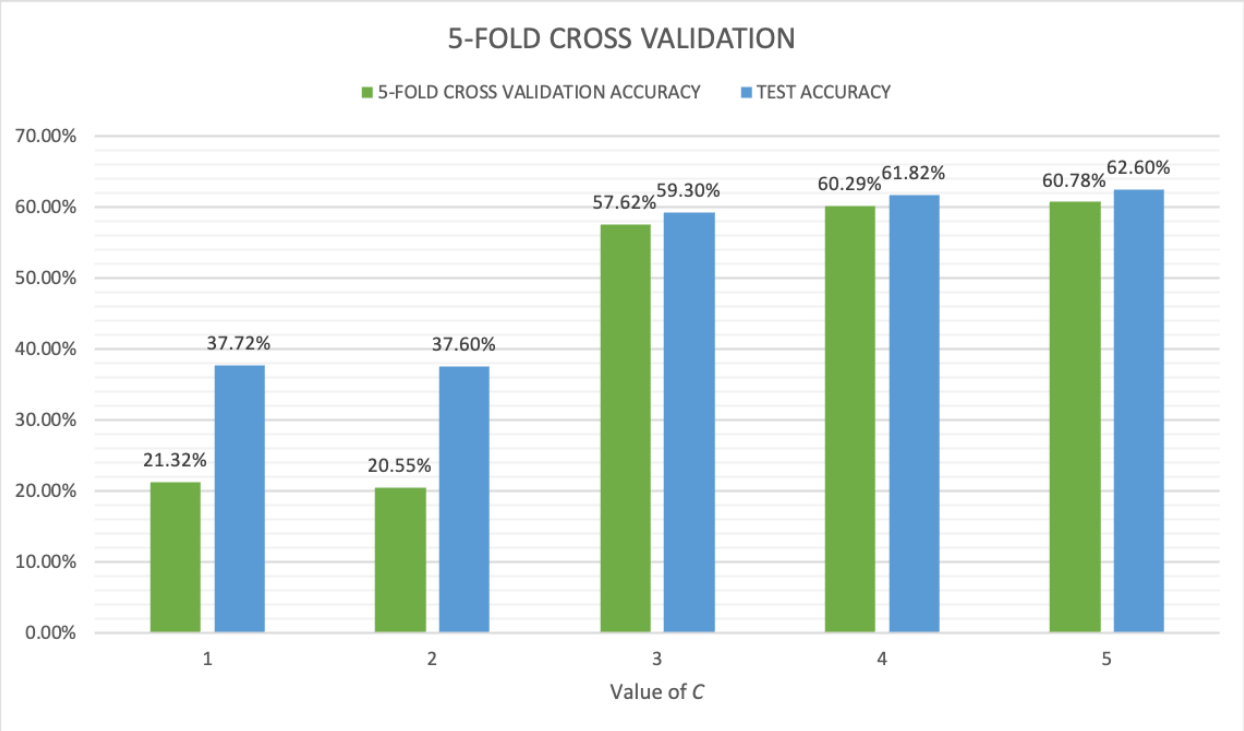


Figure 13 - 5 Fold Cross Validation Accuracies

Clearly, the value of accuracies, both validation and test increase as C increases initially.
For 1e-5 and 1e-3, the weightage of soft margin error is too low hence the model is not well fitted.
For values of C = 1, 5 and 10 the accuracies reach a plateau. **C=10 gets the best validation accuracy and also has the highest test accuracy.**

List of Libraries Used (Exhaustive list of imports)

```
import sys
import os
import math
from collections import defaultdict
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import seaborn as sb
import random
import pickle
import numpy as np
import cvxopt
import time
from scipy import spatial #(for gaussian kernel)
from sklearn import svm
```