Indian Institute of Technology Delhi

COL 774 - Machine Learning

Assignment 2

Naive Bayes Classifier and Support Vector Machines

Garv Nagori
2021CS10549

Contents

§1 Part 1. Text Classfication

§1.1 a) Naive Bayes Classifier

In this part, I implemented a Naive Bayes Classifier based on the Bag of Words Model. We model each word with its frequency in each of the classes and assign probabilities according to them.
The Prior Probability Pr[C] is given by:

$$P(C) = \frac{\text{Number of Documents in Class C}}{\text{Total number of training documents}}$$

The Likelihood Probability of each word Pr[$w \mid C$] is given by:

$$P(w|C) = \frac{\text{Frequency of w in class C } + 1}{\text{Total count of words in Class C } + \text{ Vocabulary Size}}$$

We estimate the class C of a particular text document d by Pr[$C \mid d$]

$$P(C|d) = \frac{P(C)\prod_{i=1}^{n} P(w_i|C)}{P(d)}$$
$$P(C|d) \propto P(C)\prod_{i=1}^{n} P(w_i|C)$$

Since, $P(d)$ is constant for all, we can ignore it and compare only the numerators for all the classes. Instead of comparing the actual probability values, we can compare their logarithms since they are easier to calculate and store.

$$log(P(C|d)) = log(P(C)) + \sum_{i=1}^{n} log(P(w_i|C))$$

The values of $P(C)$ for the three classes were found to be:

$$pc = [0.44, 0.19, 0.37]$$

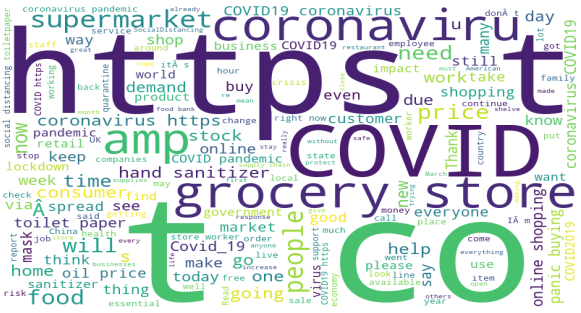
These show the percentage of training examples of each class.
The Naive Bayes Classifier was created for the test data and had the following statistics.

$$\begin{aligned} \text{Correct} &= 31800 \\ \text{Incorrect} &= 6064 \\ \text{Accuracy} &= 83.98\% \end{aligned}$$

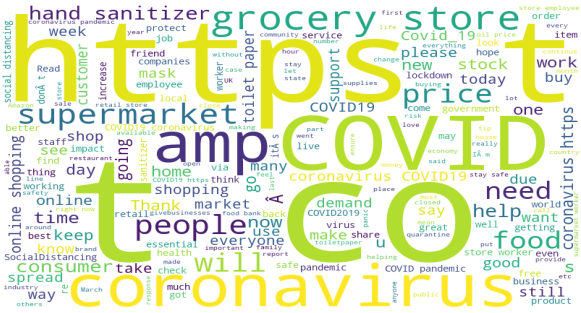
The accuracy of the Validation data was also calculated.

$$\begin{aligned} \text{Correct} &= 2188 \\ \text{Incorrect} &= 1105 \\ \text{Accuracy} &= 66.44\% \end{aligned}$$

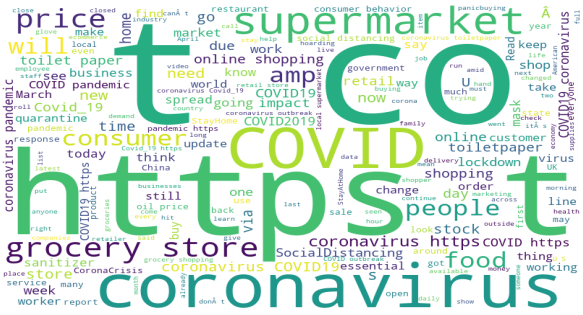
I also created word clouds to represent the most frequent words in all of the classes.



(a) Word cloud for all classes



(b) Wordcloud for all 'Positive' documents



(c) Wordcloud for all 'Neutral' documents

(d) Wordcloud for all 'Negative' documents

Figure 1: Word Clouds for all different classes

§1.2 b) Baseline Models for Comparison

For comparison, we set baseline as **Random Model** and **Always Positive Model**.
For the Random Model, we output a random choice for each input. The Naive Bayes Model should perform much better than this model because it should learn on the data.
The Validation Set Accuracy for this model is

$$\begin{aligned} \text{Correct} &= 1082 \\ \text{Incorrect} &= 2211 \\ \text{Accuracy} &= 32.86\% \end{aligned}$$

This is what we expected since this model will have a 1 in 3 chance of guessing the correct output.

The Always Positive Model will always output 'Positive' no matter the input.
The Validation Set Accuracy for this model is

$$\begin{aligned} \text{Correct} &= 1444 \\ \text{Incorrect} &= 1849 \\ \text{Accuracy} &= 43.85\% \end{aligned}$$

This is a little bit higher than Random Guessing since there are a larger number of 'Positive' examples compared to the other two classes.
In contrast, the Naive Bayes model has an accuracy of 66.44% which is a lot higher than both of these models.

§1.3 c) Confusion Matrix

The Confusion Matrix is a good way to estimate which classes are being misclassified into which classes the most. This can help identify bugs or mistakes in the trained model and help improve the model. The colums are represented by the values predicted by the model whereas the rows are the actual values. Hence, the diagonal elements are the correctly classified examples and the rest are misclassified.

Confusion Matrix for the Naive Bayes Classifier:

Classes	Positive	Neutral	Negative
Positive	15709	164	729
Neutral	2357	3312	1427
Negative	1225	162	12779

Training Set Matrix

Classes	Positive	Neutral	Negative
Positive	1194	22	228
Neutral	349	91	177
Negative	310	19	903

Validation Set Matrix

Confusion Matrix for the Random Prediction Model:

Classes	Positive	Neutral	Negative
Positive	5559	5424	5619
Neutral	2318	2360	2418
Negative	4797	4667	4702

Training Set Matrix

Classes	Positive	Neutral	Negative
Positive	471	497	476
Neutral	213	186	218
Negative	390	417	425

Validation Set Matrix

Confusion Matrix for the Always Positive Prediction Model:

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

Training Set Matrix

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

Validation Set Matrix

In all the matrices, the diagonal entry of 'Positive - Positive' is the highest, which means that the most correctly classified class is the 'Positive' class. In Random Model, this is because there are a larger number of 'Positive' class examples in the validation set.
In the Naive Bayes Classifier also, the most correctly classified class is 'Positive'. The most misclassified is classifying 'Neutral' examples as 'Positive', suggesting that the model is less accurate while detecting differences between these two.

§1.4 d) Stemming and Stopword Removal

Stopwords are words that are present in almost all of the documents and are irrelevant for classification such as 'of', 'the' and 'and'.
Stemming is a process that converts each word to some 'root' version of the word, generally a noun. In this part, I perform stemming and remove stopwords using the 'nltk' library. I also remove punctuation and tokenize the words using nltk's word_tokenize function.
The validation accuracy after this preprocessing is given by,

$$\begin{aligned} \text{Correct} &= 2272 \\ \text{Incorrect} &= 1021 \\ \text{Accuracy} &= 69.00\% \end{aligned}$$

I also created wordclouds for this transformed data to visualise what the preprocessing has done.

(a) Word cloud for all classes

(b) Wordcloud for all 'Positive' documents

(c) Wordcloud for all 'Neutral' documents

(d) Wordcloud for all 'Negative' documents

Figure 2: Word Clouds for all different classes after preprocessing

The accuracy over the validation set increases because previously, we were considering different forms of the same word as different parameters, but they are related and may appear together more frequently in some class. Also words like 'of', 'to', and 'the' which generally appear in every document are removed so we don't consider those while including our observation. This also decreases training time and leads to an increase in accuracy since those were being considered as noise before.

§1.5 e) Feature Engineering

For the first part, we consider "bigrams" as an additional feature. The intuition being that words sometimes occur in pairs and bigrams will be able to capture which pairs occur more frequently in which class.

The accuracy for this model is,

Training Set	Validation Set
Correct = 36616	Correct = 2275
Incorrect = 1248	Incorrect = 1018
Accuracy = 96.71%	Accuracy = 69.08%

This model performed marginally better than the previous ones but the training accuracy increase to almost 97%. This shows that the model has overfit to the training data and performs no better on validation data.

Hence, we try to add features that don't overfit to the training data.
I added features corresponding to number of exclamation marks, question marks and hashtags. The intuition behind this was the fact that these often help in judging the sentiment of the text since users use these to convey sentiments which may not be captured with just words alone.
I also added a multiplier to the log of probability based on the number of capitalisations since uppercase letters generally indicate a stronger sentiment than lowercase letters.
The new accuracy for this model was,

Correct = 2308
Incorrect = 985
Accuracy = 70.09%

As we can see, the new accuracy is certainly better than the previous models and this also does not overfit to the training data and performs reasonably better on the validation data.

§1.6 f) Domain Adaptation

We use Source Data, in this case Coronavirus tweets to predict on Target Data, here general purpose tweets.
We have different data splits corresponding to the percentages of Target Data in the Training Set.

Domain: 1%	Domain: 2%	Domain: 5%	Domain: 10%
Correct = 1256	Correct = 1273	Correct = 1292	Correct = 1339
Incorrect = 1762	Incorrect = 1745	Incorrect = 1726	Incorrect = 1679
Accuracy = 41.62%	Accuracy = 42.18%	Accuracy = 42.81%	Accuracy = 44.37%

Domain: 25%	Domain: 50%	Domain: 100%
Correct = 1427	Correct = 1517	Correct = 1585
Incorrect = 1591	Incorrect = 1501	Incorrect = 1433
Accuracy = 47.29%	Accuracy = 50.26%	Accuracy = 52.52%

The second time, I did not include any additional Source Data and instead trained only on the Target Data. The accuracies obtained were as follows:

Domain: 1%	Domain: 2%	Domain: 5%	Domain: 10%
Correct = 1231	Correct = 1298	Correct = 1365	Correct = 1415
Incorrect = 1787	Incorrect = 1720	Incorrect = 1653	Incorrect = 1603
Accuracy = 40.79%	Accuracy = 43.01%	Accuracy = 45.23%	Accuracy = 46.88%

Domain: 25%	Domain: 50%	Domain: 100%
Correct = 1474	Correct = 1568	Correct = 1601
Incorrect = 1544	Incorrect = 1450	Incorrect = 1417
Accuracy = 48.84%	Accuracy = 51.95%	Accuracy = 53.05%

The accuracies were graphed with respect to the split of Source and Target Data.

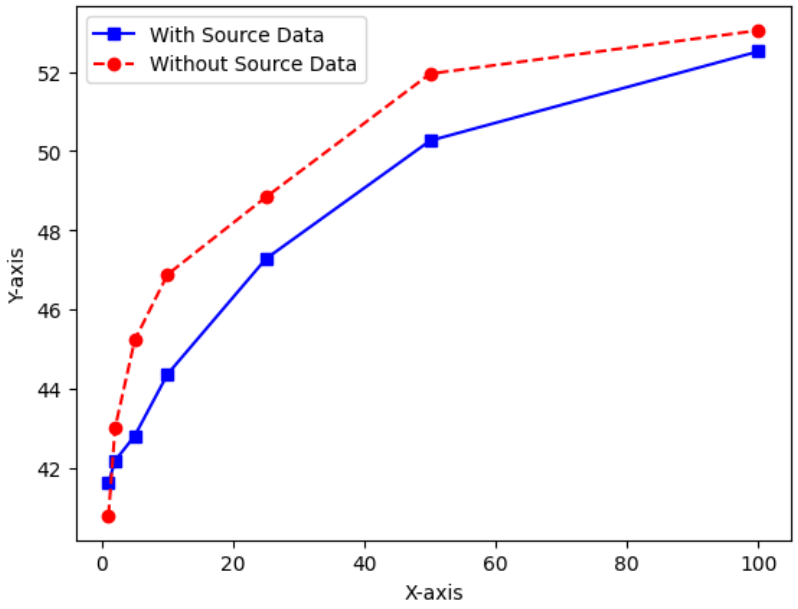


Figure 3: Accuracies vs Source-Target Split

§2 Part 2. Image Classification

§2.1 Binary Classification

Classes 3 and 4 based on entry number 2021CS10549

§2.1.1 a) Support Vector Machines

We classify the images in classes 3 and 4 using Support Vector Machines. In this part, I use a general purpose Convex Optimization Solver 'CVXOPT' to solve the dual problem.
The dual problem for SVMs with l_1 regularization is given by,

$$\begin{aligned}
&\max_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\
&\text{s.t. } 0 \leq \alpha_i \leq C, i = 1 \dots m \\
&\sum_{i=1}^{m} \alpha_i y^{(i)} = 0
\end{aligned}$$

The solution is converged to in 12 iterations using 'CVXOPT' and it takes about 65 seconds to find it.
The threshold I set to be $1e-05$ and above to consider it a support vector.

$$\begin{aligned}
\text{Number of support vectors} &= 2950 \\
\text{Total number of training examples} &= 4760 \\
\text{Percentage of examples which are support vectors} &= 61.97\%
\end{aligned}$$

The weight w and the bias b was calculated according to the following equations.

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$$
$$y^{(i)}(w^T x^{i)} + b) = 1 \text{ if } 0 < \alpha_i < C$$

This was then used to predict the classes for the training set and validation set again.
The accuracies are given by,

Training Set		Validation Set	
Correct	= 3626	Correct	= 284
Incorrect	= 1134	Incorrect	= 116
Accuracy	= 76.18%	Accuracy	= 71.00%

The support vectors corresponding to the top 6 coeffecients α are shown below.

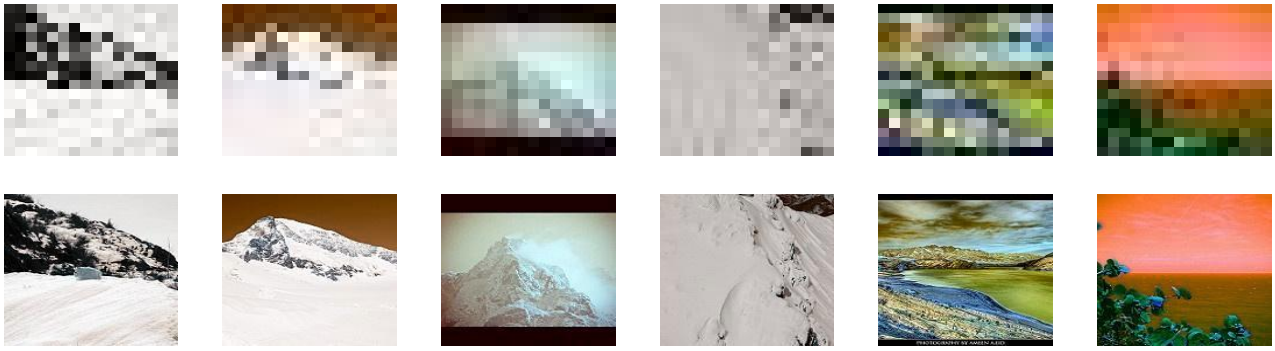


Figure 4: The Support Vectors. The top row are resized, whereas bottom row are the original images

Also, the weights vector can be visualised as an image

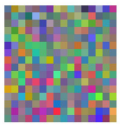


Figure 5: Weights imagined as an image

§2.1.2 b) Gaussian Kernel

Here we map x to a features using the feature mapping $\phi(x)$.
ϕ is chosen such that the Kernel matrix $K(x,z) = \phi(x)^T\phi(z)$ is Gaussian.

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

Here, we map x to an infinite dimensional feature space. So we can't calculate weights since they would be infinite dimensional. Hence, we represent it differently in terms of support vectors to calculate it efficiently.

$$\begin{aligned} w^T\phi(x) + b &= (\sum_{i=1}^{m} \alpha_i y^{(i)} \phi(x)^{(i)})^T \phi(x) + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle \phi(x^{(i)}), \phi(x) \rangle + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b \end{aligned}$$

The threshold I set to be $1e-05$ and above to consider it a support vector.

$$\begin{aligned} \text{Number of support vectors} &= 3436 \\ \text{Total number of training examples} &= 4760 \\ \text{Percentage of examples which are support vectors} &= 72.18\% \end{aligned}$$

The accuracies are given by,

Training Set		Validation Set	
Correct	= 3490	Correct	= 311
Incorrect	= 1270	Incorrect	= 89
Accuracy	= 73.32%	Accuracy	= 77.75%

The support vectors corresponding to the top 6 coeffecients α are shown below.

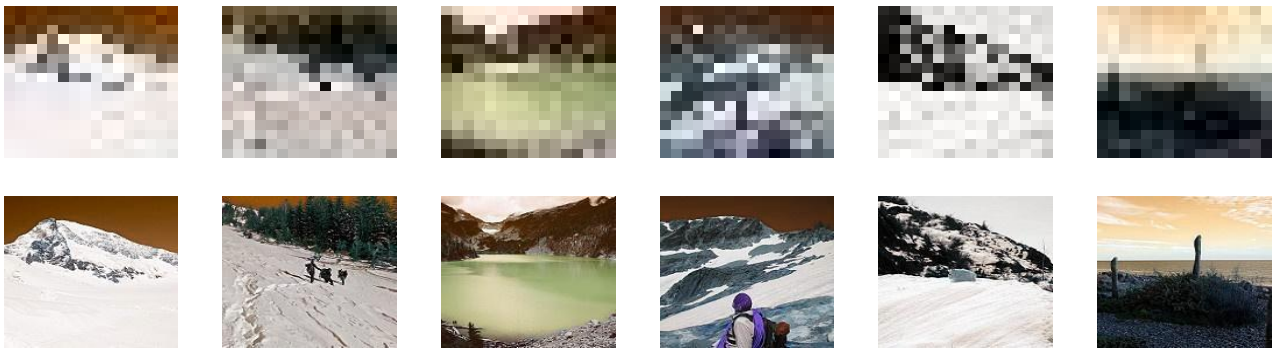


Figure 6: The Support Vectors. The top row are resized, whereas bottom row are the original images

Using Gaussian Kernel, we increase the number of support vectors but also increase the Validation Set Accuracy. The training set accuracy is slightly less, which is good since it shows that the model has not overfit to the training set. The Gaussian Kernel is a better choice in this case. However, it takes longer time to train and also predict which is one of its drawbacks.

§2.1.3 c) Scikit Learn

Here, we use the scikit-learn library to train the SVMs for both Linear and Gaussian Kernels.
The number of support vectors is given by

Linear Kernel: 2904
Gaussian Kernel: 3076

All 2904 support vectors matched with my model's support vectors for the Linear Kernel and 2686 support vectors matched for the Gaussian Kernel.

The norm of the difference between weights of my model and scikit-learn's model is 0.0236.
The difference between bias is 0.5515.

The accuracy is given by,

Linear Kernel	Gaussian Kernel
Correct $= 290$	Correct $= 311$
Incorrect $= 110$	Incorrect $= 89$
Accuracy $= 72.50\%$	Accuracy $= 77.75\%$

For the Linear Kernel, my model took about 65 seconds whereas scikit learn's model took only 8 seconds
For the Gaussian Kernel, my model took about 52 seconds whereas scikit learn's model took only 5 seconds.

§2.1.4 d) Size changed

Here, instead of resizing images to (16, 16), we resized it to (32, 32).
Now, time increased for training the models.
For linear model it took about 63 seconds and for Gaussian it took about 38 seconds.
The validation set accuracies also changed. For linear, it became 67% and for Gaussian, 78.75%.

§2.2 Multi Class Classification

§2.2.1 a) One vs one Classifier

In this model, we train $\binom{k}{2}$ models, for each pair of classes. For prediction, we then run all the models and count the number of votes for each class. If there is a tie, we break it using the maximum distance of the point from the decision boundary.
The time it took to train all 15 classifiers was about 25 minutes.
The validation set accuracy is given by,

$$\begin{aligned} \text{Correct} &= 667 \\ \text{Incorrect} &= 553 \\ \text{Accuracy} &= 55.58\% \end{aligned}$$

§2.2.2 b) Scikit Learn's One vs One Classifier

We used SciKit Learn's Multi Class Classifier.
The time it took to train was about 60 seconds.
The validation set accuracy is given by,

$$\begin{aligned} \text{Correct} &= 781 \\ \text{Incorrect} &= 419 \\ \text{Accuracy} &= 65.08\% \end{aligned}$$

The accuracy for scikit learn's model is 10% better. It also took very less time compared to our own implementation.

§2.2.3 c) Misclassifications

The confusion matrix for our model is given by,

Classes	0	1	2	3	4	5
0	73	22	18	27	26	34
1	4	150	1	6	12	27
2	9	4	123	26	23	15
3	23	6	25	126	15	5
4	18	17	57	33	69	6
5	25	23	10	8	8	126

Our Model

The confusion matrix for scikit-learn's model is given by,

Classes	0	1	2	3	4	5
0	116	15	17	16	8	28
1	7	154	0	3	7	29
2	12	1	139	19	21	8
3	19	5	21	125	26	4
4	20	8	37	29	104	2
5	31	12	7	6	1	143

SciKit Learn's Model

The most misclassified classes are:
Class 4 often misclassified as classes 2 and 3.
Classes 0 and 1 often misclassified as classes 5.

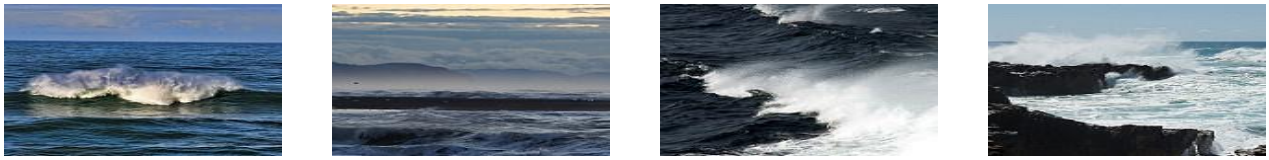


Figure 7: Images of class 4 misclassified as class 2

Figure 8: Images of class 4 misclassified as class 3

Figure 9: Images of class 0 misclassified as class 5

Figure 10: Images of class 1 misclassified as class 5

These results make sense because some images can be viewed as from a different class, especially if the image is scaled down to (16, 16).

§2.2.4 d) K-Fold Cross Validation

K-Fold Cross Validaiton is typically used for a range of hyperparameter values and the ones which give the best K-Fold Cross Fold Validation Accuracy are reported as the best hyperparameters.
Here we do 5-Fold Cross Validation for 5 different values of C for a fixed value of $\gamma = 0.001$.

$$\begin{aligned} \text{Value of C} &= 1e-05 \\ \text{5-Fold Cross Validation Accuracy} &= 15.896\% \\ \text{Validation Accuracy} &= 40.167 \end{aligned}$$

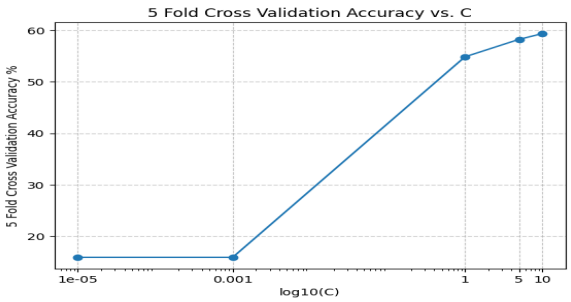
$$\begin{aligned} \text{Value of C} &= 0.001 \\ \text{5-Fold Cross Validation Accuracy} &= 15.896\% \\ \text{Validation Accuracy} &= 40.167 \end{aligned}$$

$$\begin{aligned} \text{Value of C} &= 1 \\ \text{5-Fold Cross Validation Accuracy} &= 54.853\% \\ \text{Validation Accuracy} &= 55.916 \end{aligned}$$

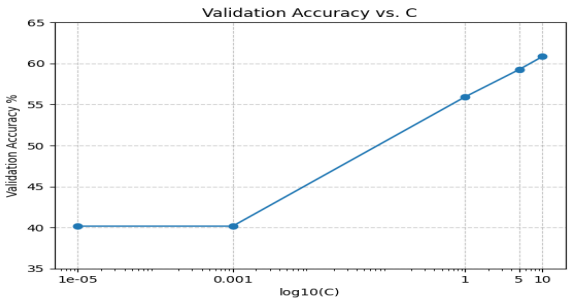
$$\begin{aligned} \text{Value of C} &= 5 \\ \text{5-Fold Cross Validation Accuracy} &= 58.249\% \\ \text{Validation Accuracy} &= 59.250 \end{aligned}$$

$$\begin{aligned} \text{Value of C} &= 10 \\ \text{5-Fold Cross Validation Accuracy} &= 59.370\% \\ \text{Validation Accuracy} &= 60.833 \end{aligned}$$

5-Fold Cross Validation and Accuracy on Validation set were plotted against the value of C (logarithmic graph).



(a) 5-Fold Cross Validation Accuracy vs C



(b) Validation Accuracy vs C

Value of $C = 10$ gives the best accuracy for both 5-Fold Cross Validation Accuracy and the Accuracy on the Validation Set. This shows that this value of C is best to eliminate overfitting to the training data and handle noise effectively.

§3 Acknowledgements

I have used the style file from here[1] to produce this document.

[1] https://github.com/vEnhance/dotfiles/blob/main/texmf/tex/latex/evan/evan.sty