COL774 Assignment 2

-Dhairya Gupta, 2019CS50428

1. Text Classification

We have used a naive Bayes classifier assuming a multinoulli model(Bag of Words). Laplace smoothing factor, $\alpha = 1$ in all cases. Log values were used to calculate likelihood to prevent underflow. While testing, we have ignored new words not present in the vocabulary.

(a)

Training set accuracy: 69.92%, Macro F1 score: 0.562

Test set accuracy: 66.76%, Macro F1 score: 0.29

(b)

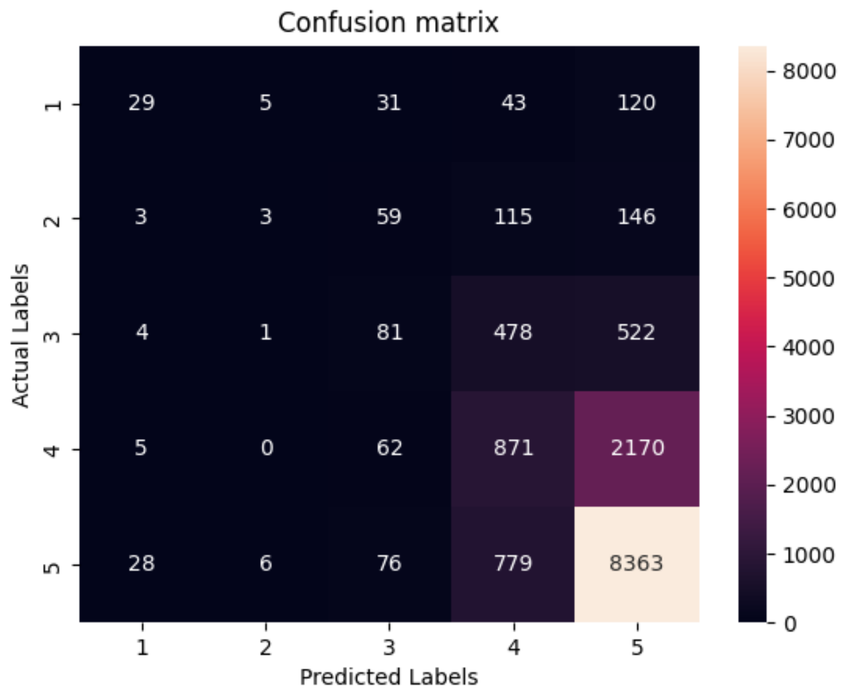
Random guess test accuracy: 19.88%

Majority guess test accuracy: 66.08%. (For rating = 5)

Random guess accuracy should be around 20% = $\frac{1}{5}$ as evident in our results.

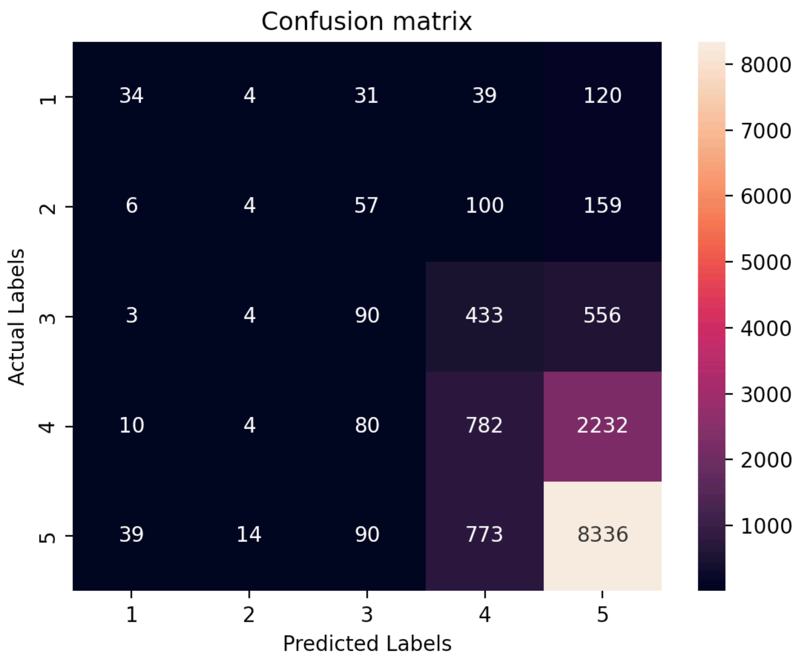
From above results, we can see that there is a huge class imbalance in the test_data. This is due to large number of positive reviews in the Amazon data which dominate the training, as well as test dataset. Therefore, the majority guess test accuracy is large. Our naive Bayes model perform slightly better than the majority guess and much better than random guessing.

(c)



Shown above is the Confusion matrix for the naive Bayes classifier in part(a). We observe that most of our predictions are for class 5 in each class. This is likely due to the huge imbalance in the training examples, not enough unique words corresponding to each class. Class 5 has the largest number of training examples by a large margin. Therefore, for class 5, maximum number of labels correspond to the correct data. As visible by its diagonal entry. We can also see 2 has negligible prediction, possibly because the words are indistinguishable from other ratings such as 1 or 3. We see our data is close to majority prediction. Confusion is evident between 4 and 5 where prediction data is more in number and prediction accuracy is higher. It is possible that classification with multinoulli naive Bayes is unable to give us a good fit for the given data.

(d)



Test accuracy: 66.04%

Macro F1 score: 0.29

We can see confusion matrix above being nearly identical as the one without stemming. On incorporating stemming and stopword removal, our accuracy remains nearly the same. This is possibly due to the data still being highly imbalanced and the distribution of words not being affected much by stopword removal.

Also, we can attribute the slight decrease on stemming which removes words with different meanings but similar stem-word, leading to poorer accuracy.

(e)**Feature Engineering:**

We have incorporated 2 features into our model. The features are:

1. Unigrams + bigrams

2. Frequency scaling according to TF-IDF model

Bigrams: In this case, we have also added tuples of words occuring together in our dictionary. This is applied on our stemming/stopword removed model. This is intuitive as words occuring together can have different sentiments in the natural language.

Frequency scaling: For every label, each word has a corresponding frequency parameter in the Bag of Words model. For each word frequency, we multiply by by a term corresponding to document frequency of the word. In particular, if $f_{w|y=k}$ is the frequency of word w among words corresponding to label k, and there are N text documents. $f'_{w|y=k} = f_{w|y=k} . \log \frac{N}{N_w}$ where N_w is the number of documents in which word w occurs. By doing this, we are decreasing the frequency of words which occur in a large number of documents, and hence increase the relative frequency of unique words which can lead to better classification.

On using only (unigrams + bigrams):

Test accuracy: 66.19%

Macro F1 score: 0.169

On using only (unigrams + TF-IDF scaled frequency):

Test accuracy: 64.65%

Macro F1 Score: 0.335

On using both features together:

Test accuracy: 66.27%

Macro F1 score: 0.263

We observe that using both features gives a slightly high accuracy than simply using stemming/stopword removal. Scaled frequency, however gives lower accuracy. Again, the reason for this is the class imbalance in the data. Due to larger number of 5/5 ratings, the scaling factor reduces frequency of words corresponding to label 5 and recall for label 5 decreases. However, f1_scores for other parameters increase

(f) F1 score is a much more suitable way of measuring accuracy. F1 score for each class is the Harmonic Mean of metrics precision and recall. Precision for class 1 is the fraction of correct predictions for class 1 out of all predictions predicting class = 1. Recall is the fraction of correct predictions for class 1 out of all examples with label 1. As we have seen above, F1 scores vary differently than accuracy. Given that the data set is highly imbalanced, even a very bad predictor such as majority guess gives a competitive accuracy which is not desirable. Hence, we can use F1 scores for each class. Unlike normal accuracy, it is not a weighted sum of correct predictions and hence, can overlook the class imbalance and gives a better metric for validation.

For our best model in feature engineering(using both features)
F1 scores : [0.1496063, 0.0060241, 0.0854430, 0.27016129, 0.80547428] corresponding to classes 1-5.
Macro F1 score = average of F1 scores = 0.2633418

We can see that other models such as only TFIDF gave a lower accuracy but higher F1 score indicating better classification for each class. Hence, normal accuracy metric is not ideal in our case.

(g) Summarized review gives a slightly better classification than reviewText. This is because of less number of redundant words and more number of unique/ determining words in the document text. Results are given below

No modifications:

Accuracy: 66.5%

Macro F1 score: 0.314

When only stemming is done:

Test Accuracy: 65.9%

Macro F1 score: 0.315

Bigrams without stemming

Accuracy: 67.9%

Macro F1 score: 0.294

Bigrams with stemming

Accuracy: 67.3%

Macro F1 score: 0.274

Bigrams with TF-IDF:

Accuracy: 63.9%

Macro F1 score: 0.34

From above results, we can see that using unigrams + bigrams lead to higher accuracy. Also, we can see that stemming slightly reduces the accuracy. This is because stemming can remove information from text. In case of summarized review, this leads to more inaccuracies. We also see that TF-IDF reduces accuracy by a larger amount. This is because TF-IDF helps in reducing redundant words, but in a summarized review words occuring in multiple documents may be important for characterization. Hence, TF-IDF may not be ideal to use in summarized review. Also, note that TF-IDF still increases F1 scores, again due to the class imbalance with very few documents for low ratings.

MNIST digit classification:

(a) For binary classification, the labels are 8 and 9

(i) The dual SVM problem for a linear kernel can be written as minimizing the expression $\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)^T}x^{(j)} - \sum_{i=1}^{m}\alpha_i$, with constraints

$0 \leq \alpha_i \leq C, \forall i \in \{1, 2, \ldots, m\}$ and $\sum_{i=1}^{m}\alpha_i y^{(i)} = 0$. Here, we need to find

P, q, G, h, A, b such that above sexpression is of form $\min_{\alpha} \frac{1}{2}\alpha^T P\alpha + q^T\alpha$ with

$G\alpha \preceq h$ and $A\alpha = b$.

After computation, we find P = QQ^T where $Q_{ij} = x_j^{(i)}y^{(i)}$. This can be calculated using numpy without using explicit for loops. The second term in equation gives $q = [(-1)\ (-1)\ldots\ (-1)]^T$. We can also see trivially that $A = y^T, b = 0$. For $G\alpha_i \preceq h$, we have $\alpha_i \preceq C$, and $-\alpha_i \preceq 0$. This can be represented as below

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \\ -1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & -1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & -1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & -1 \end{bmatrix} \qquad h = \begin{bmatrix} C \\ C \\ C \\ \vdots \\ C \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Results obtained for CVXOPT are:

Validation accuracy (20% separated from train_set) = 97.7%

Test accuracy = 97.4%

Bias = 1.21, no.of support vectors = 189

Time taken = 13.487s

(ii) In this case, P changes other parameters are not affectd from linear kernel case. $P_{ij} = y^{(i)}y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)}) = y^{(i)}y^{(j)}e^{-\gamma(x^{(i)}-x^{(j)})^T(x^{(i)}-x^{(j)})}$. This can also been done using matrix manipulation to avoid any for loops during calculation. In this case, we cannot store w as it is an infinite dimensional vector. Hence, we store α, b, y and use it with kernel function to get prediction.

Results obtained for CVXOPT gaussian kernel are:
Validation accuracy (20% separated from train_set) = 98.7%
Test accuracy = 98.4%
Bias = -0.40, no.of support vectors = 1313
Time taken = 11.201s

Gaussian kernel yields higher accuracy as non-linear separator is able to fit the data better than linear separator.

(iii)Results for LIBSVM:
For linear kernel:
Test accuracy = 97.4%
Bias = -1.21, no.of support vectors = 189
Time taken = 1.194s

For gaussian kernel:
Test accuracy = 98.4%
Bias = -0.40, no.of support vectors = 1296
Time taken = 4.506s

We observe that data obtained from LibSVM implementation is nearly identical. No. of support vectors differs due to testing with different values of epsilon threshold in CVXOPT and LIBSVM. We observe that LibSVM approach is more efficient by a factor of 3-5.

(b)

(i) Results for Multiclass one vs one using CVXOPT:

Test Accuracy: 97.24%,

Time taken: 1698.32s

(ii) Results for Multiclass one vs one using LIBSVM:

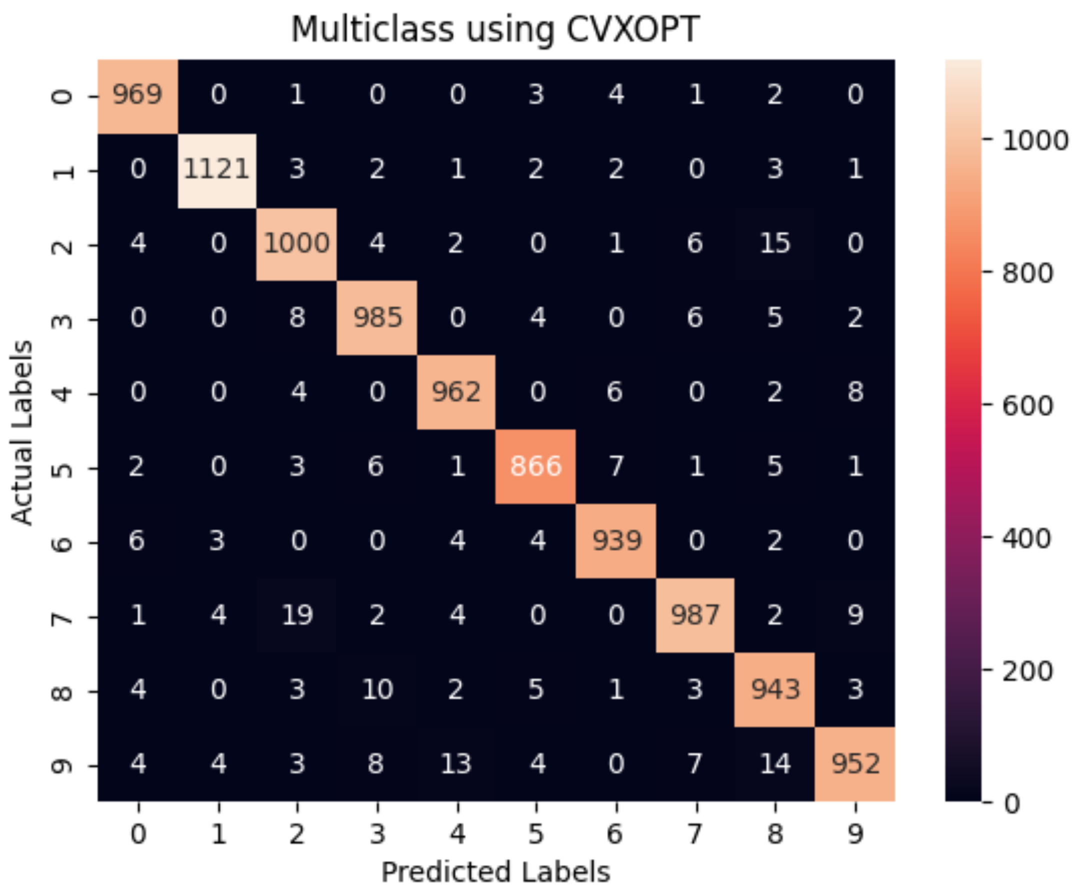
Test Accuracy: 97.23%

Time taken: 363.52s

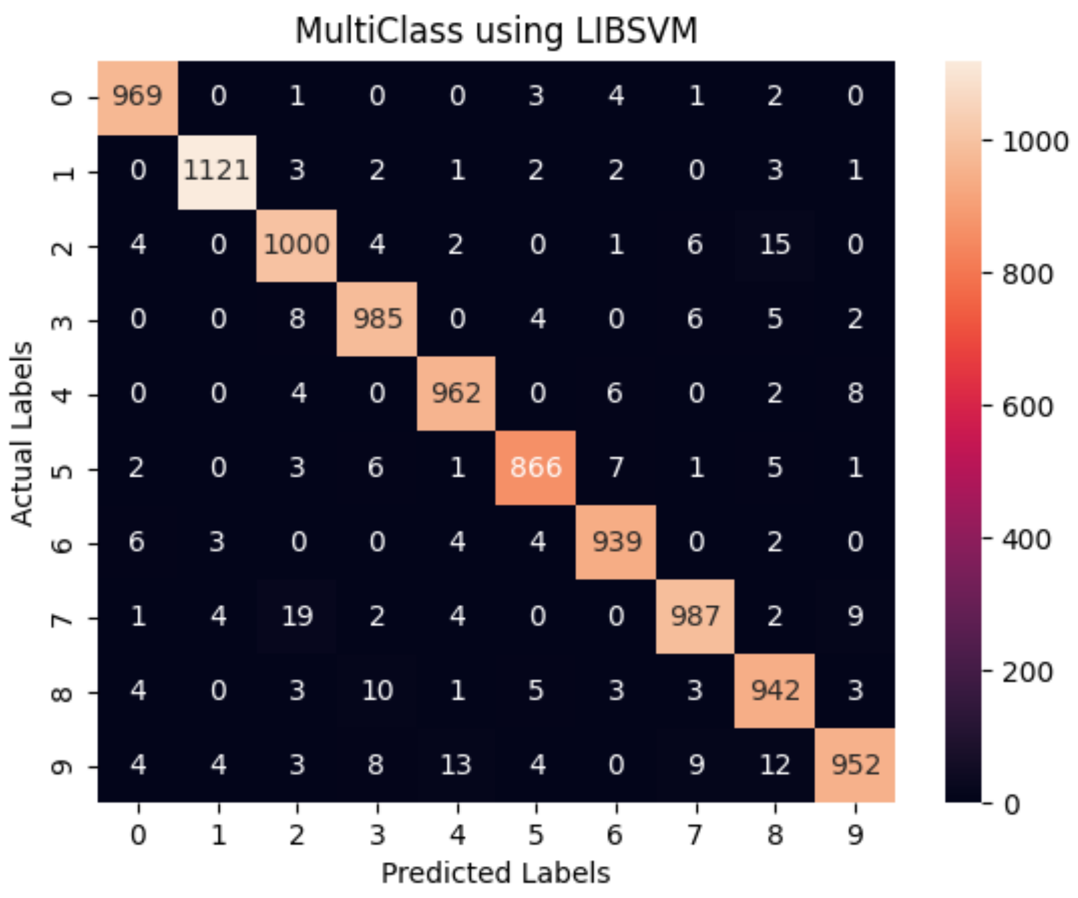
We observe the test accuracy is identical in both cases. However, LIBSVM is computationally efficientby a factor of 5 for getting the same results.

(iii)

Confusion matrices:


Multiclass using CVXOPT

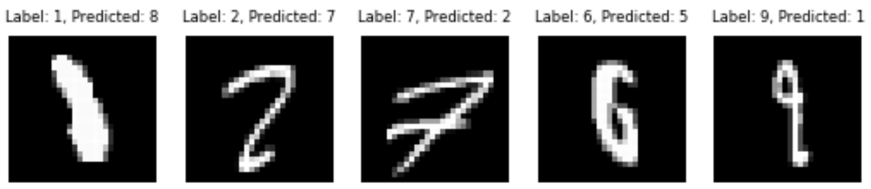
Actual Labels \ Predicted Labels	0	1	2	3	4	5	6	7	8	9
0	969	0	1	0	0	3	4	1	2	0
1	0	1121	3	2	1	2	2	0	3	1
2	4	0	1000	4	2	0	1	6	15	0
3	0	0	8	985	0	4	0	6	5	2
4	0	0	4	0	962	0	6	0	2	8
5	2	0	3	6	1	866	7	1	5	1
6	6	3	0	0	4	4	939	0	2	0
7	1	4	19	2	4	0	0	987	2	9
8	4	0	3	10	2	5	1	3	943	3
9	4	4	3	8	13	4	0	7	14	952

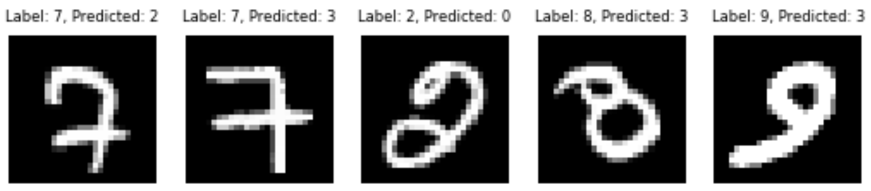


	0	1	2	3	4	5	6	7	8	9
0	969	0	1	0	0	3	4	1	2	0
1	0	1121	3	2	1	2	2	0	3	1
2	4	0	1000	4	2	0	1	6	15	0
3	0	0	8	985	0	4	0	6	5	2
4	0	0	4	0	962	0	6	0	2	8
5	2	0	3	6	1	866	7	1	5	1
6	6	3	0	0	4	4	939	0	2	0
7	1	4	19	2	4	0	0	987	2	9
8	4	0	3	10	1	5	3	3	942	3
9	4	4	3	8	13	4	0	9	12	952

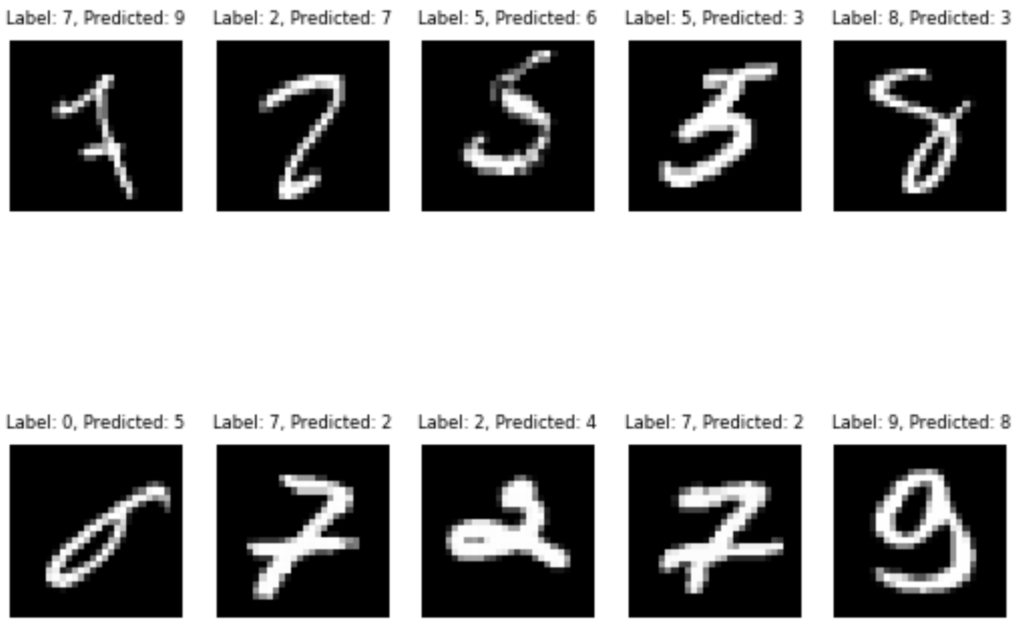
As we observe, both the confusion matrices are nearly identical with negligible differences. We observe that the most common misclassified label is digit 7, which is interpreted as 2 a lot. The example misclassified digits are shown below. Also, 8 is interpreted as 3 often. This is intuitive as these digits are similar in shape/structure.

Multiclass using CVXOPT





Multiclass using LIBSVM



(iv)

Results: (Here validation accuracy corresponds to highest accuracy among all models)

Validation accuracy for C = 10: 97.625%

Validation accuracy for C = 5: 97.625%

Validation accuracy for C = 1: 97.525%

Validation accuracy for C = 1e-3: 17.15%

Validation accuracy for C = 1e-5: 17.15%

Test accuracy for C = 10: 97.29%

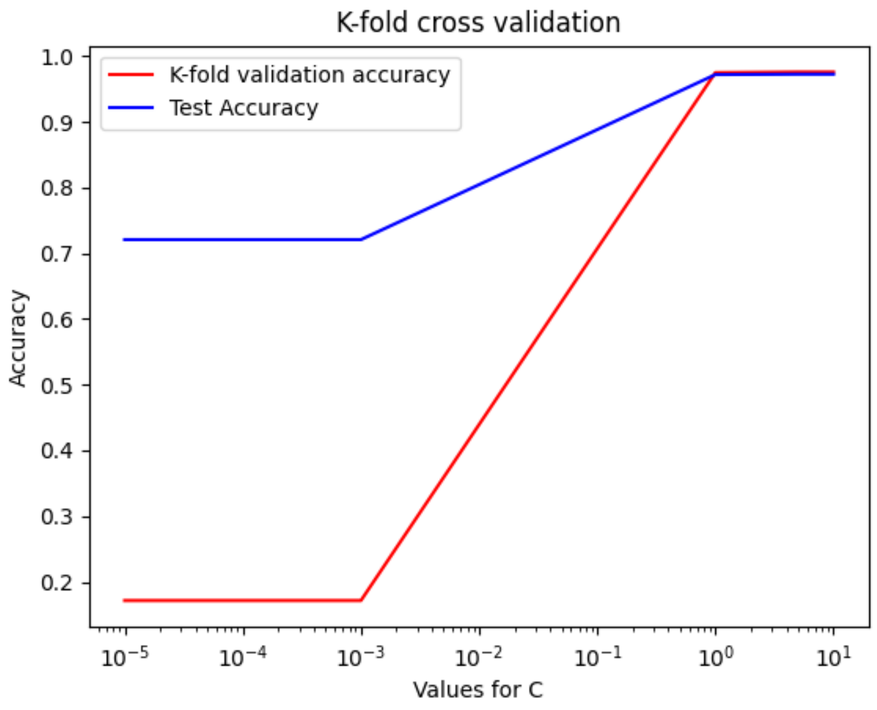
Test accuracy for C = 5: 97.29%

Test accuracy for C = 1: 97.23%

Test accuracy for C = 1e-3: 72.1%

Test accuracy for C = 1e-5: 72.1%

We observe that accuracies for 1e-5 and 1e-3 are very poor. For larger C values the result saturates at around 97.2%. The C values are indicative of penalty/loss associated with the regularization parameter ζ_i. Higher values of C correspond to hard SVMs. For smaller values of C, margin can be adjusted even if accuracy is low. High accuracies for large C's indicate the data is almost perfectly separable using a gaussian kernel. Plot for train and validation accuracies is given below.


K-fold cross validation
K-fold validation accuracy
Test Accuracy
Accuracy
Values for C
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}