# Text Classification and MNIST Digit Classification

**Prof. Parag Singla**

**Hardeep Kaur 2019CS10354**


**Assignment 2**
**COL774, Machine Learning**
**Indian Institute of Technology, Delhi**

October 8, 2021

# Contents

# List of Figures

# Text Classification

## Problem Statement

In this problem, we will use the Naive Bayes algorithm for text classification. The data set for this problem is the Amazon Digital Music review data set. Given a user's review, task is to predict the 'overall' rating given by the reviewer.A review comes from one of the five categories (class label). Here, class label represents 'overall' rating given by the user along with the 'reviewText'.

## 1.1 Implementing a Naive Bayes Classifier

**Model used:** Multinoulli event model, i.e, bag of words model. The words appearing in reviews are stored in vocabulary during learning and predictions are made using the probabilities based on frequency of words.

**Parameters:** Parameters in the model are $\{\phi_k\}_{k=1}^{r}$ and $\theta_{j=l|k}$ whose expressions are given below:

$$\phi_k = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = k\}}{m} = \frac{\text{no. of reviews with rating k}}{\text{total no. of reviews}}$$

$$\theta_{j=l|k} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = k\} \sum_{i=1}^{m} 1\{x_j^{(i)} = l\}}{\sum_{i=1}^{m} 1\{y^{(i)} = k\} n^{(i)}}$$

$$= \frac{\text{total no. of occurrences of word j in all reviews with rating k}}{\text{total no. of words in all reviews of rating k}}$$

**Accuracy:**
1. Training data set: 69.918 %
2. Test data set: 65.464 %

**Confusion Matrix:**
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{vmatrix} 15 & 2 & 23 & 40 & 148 \\ 5 & 1 & 27 & 133 & 160 \\ 5 & 4 & 42 & 507 & 528 \\ 21 & 2 & 54 & 961 & 2070 \\ 105 & 26 & 85 & 890 & 8146 \end{vmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0.079 & 0.055 & 0.063 & 0.340 & 0.802 \end{vmatrix}$$

**Macro score:** 0.2583

**Total time taken:** 131.928 s

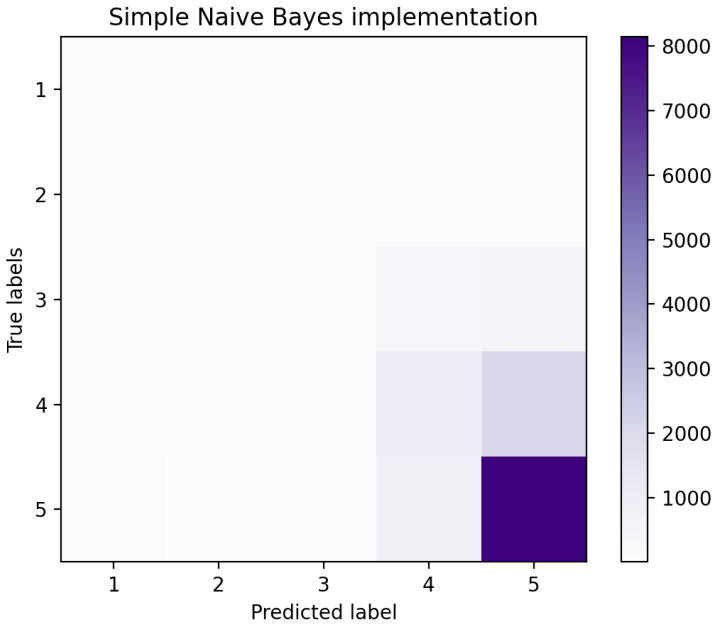



*Figure 1.1: Confusion Matrix for Naive Bayes Classifier*

## 1.2 Baseline Models

### 1.2.1 Random Prediction Model

The probability of a guess being correct is $\frac{1}{r}$ where r is the number of classes, so the expected number of correct predictions becomes equal to $\frac{m}{r}$ for a test data set of m examples. The accuracy is less for both the given data sets as the reviews do not have uniform ratings in both of them.

**Accuracy:**

1. Training data set: 19.874 %
2. Test data set: 19.885 %

**Confusion Matrix:**
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{vmatrix} 41 & 54 & 51 & 30 & 52 \\ 57 & 58 & 84 & 72 & 55 \\ 213 & 255 & 206 & 201 & 211 \\ 641 & 626 & 617 & 609 & 615 \\ 1884 & 1852 & 1781 & 1865 & 1870 \end{vmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0.026 & 0.036 & 0.107 & 0.206 & 0.310 \end{vmatrix}$$

**Macro score:** 0.1376

**Total time taken:** 216.486 s

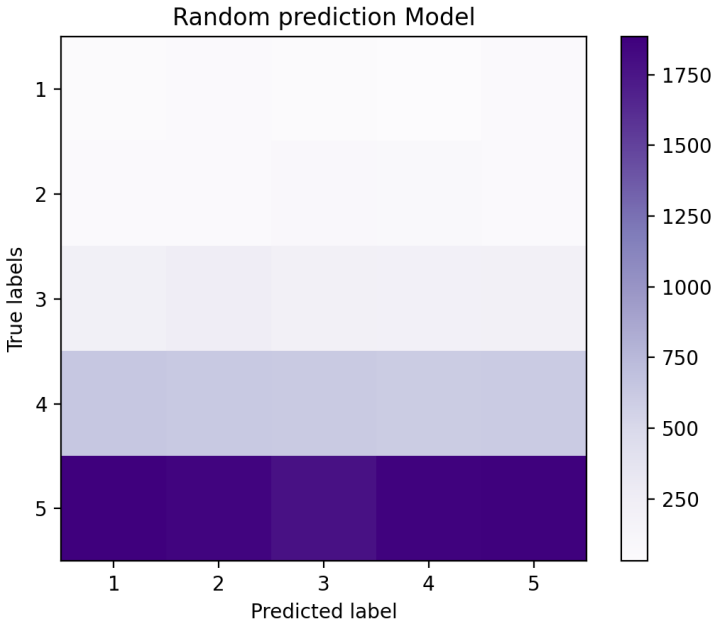



*Figure 1.2: Confusion Matrix for Random Prediction*

### 1.2.2 Majority Prediction Model

This always predict a single rating, which occurs for majority of the reviews in the data set. Since the given data set has a significant amount of reviews having rating 5, the accuracy is high, close to our Naive Bayes implementation.

**Accuracy:**

1. Training data set: 51.864 %
2. Test data set: 66.085 %

**Confusion Matrix:**
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{vmatrix} 0 & 0 & 0 & 0 & 228 \\ 0 & 0 & 0 & 0 & 326 \\ 0 & 0 & 0 & 0 & 1086 \\ 0 & 0 & 0 & 0 & 3108 \\ 0 & 0 & 0 & 0 & 9252 \end{vmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0 & 0 & 0 & 0 & 0.795 \end{vmatrix}$$

**Macro score:** 0.1591

**Total time taken:** 232.982 s

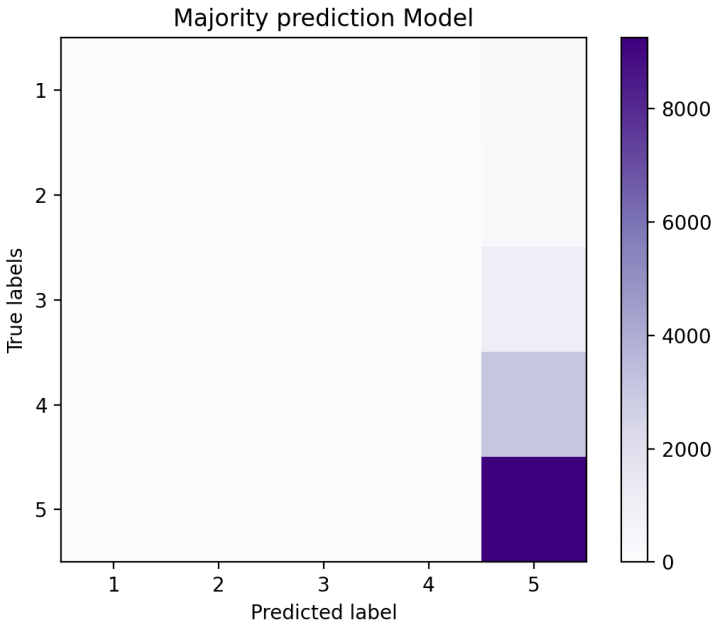



*Figure 1.3: Confusion Matrix for Majority Prediction*

## 1.3 Analysing the Confusion Matrix

Confusion matrix for part (a) is 2.1. Highest diagonal entry is for rating = 5. This happens because the number of reviews having rating = 5 are significantly high in both training data set and test data set.

### 1.3.1 Observations

1. There is a significant amount of mis-classification of documents of class 4 into class 5 by the model. This may be due to similarity of words used in reviews having rating equal to 4 and 5.
2. There are very few documents of ratings 2 and 3 mis-classified into rating 1. This may be due to large number of training examples for rating 1 and also due to the difference in type of words used by rating 1 reviewers.
3. The accuracy of majority prediction model is better than the Naive Bayes implementation due to large number of training and test examples of class 5.

## 1.4 Stemming and Stop word Removal

The data set provided is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as of, the, and etc. (called stop words). These words may not be relevant for classification. Similarly, the raw data treats different forms of the same word separately, e.g., eating and eat would be treated as separate words. Merging such variations into a single word is called stemming. This section gives the performance of Naive Bayes after performing stemming and removal of stop words.

**Accuracy:**
1. Training data set: 68.589 %
2. Test data set: 65.95 %

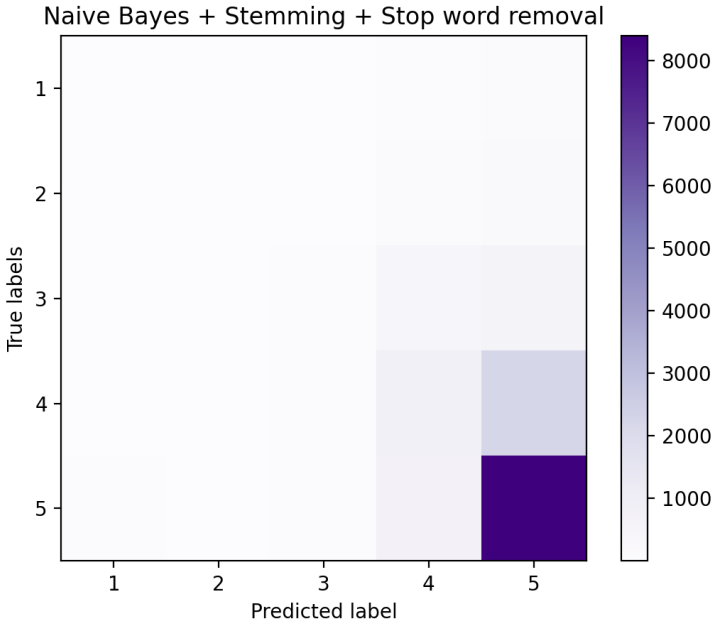



*Figure 1.4: Confusion Matrix for Naive Bayes + Stemming + Stopword Removal*

**Confusion Matrix:**

$$\begin{vmatrix} 15 & 2 & 21 & 39 & 151 \\ 3 & 3 & 29 & 106 & 185 \\ 1 & 2 & 44 & 430 & 609 \\ 5 & 3 & 57 & 781 & 2262 \\ 39 & 22 & 84 & 717 & 8390 \end{vmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0.103 & 0.016 & 0.066 & 0.301 & 0.804 \end{vmatrix}$$

**Macro score:** 0.2585

**Total time taken:** 263.171 s

## 1.5 Feature Engineering

### 1.5.1 Using uni-grams and bi-grams

Instead of using only uni-grams, we tried using both uni-grams (words) and bi-grams (two consecutive words). It is expected that accuracy would increase as occurrence of two consecutive words leads to a better estimate of rating.

**Accuracy:**

1. Training data set: 78.292 %
2. Test data set: 66.492 %

**Confusion Matrix:**
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{vmatrix} 3 & 0 & 9 & 39 & 177 \\ 0 & 1 & 5 & 70 & 250 \\ 0 & 0 & 8 & 302 & 776 \\ 0 & 0 & 14 & 472 & 2622 \\ 11 & 10 & 23 & 383 & 8825 \end{vmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0.024 & 0.005 & 0.013 & 0.215 & 0.805 \end{vmatrix}$$

**Macro score:** 0.2132

**Total time taken:** 1006.792 s

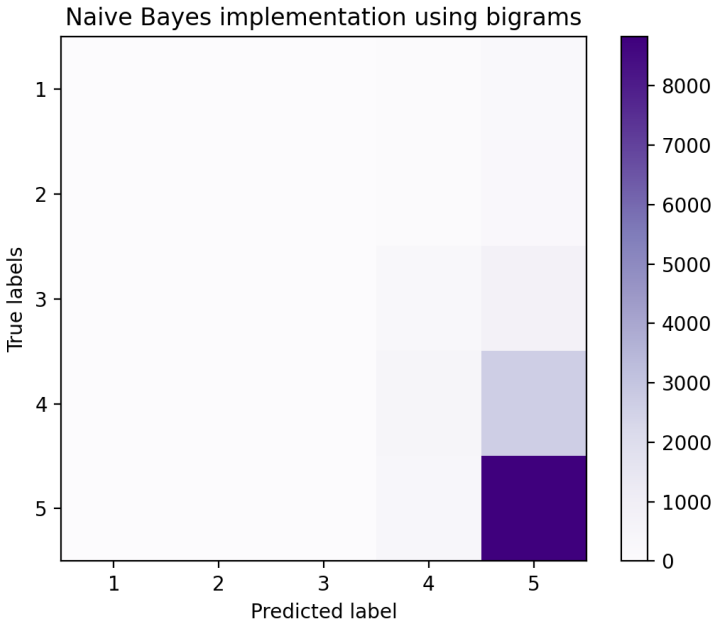



*Figure 1.5: Confusion Matrix for Naive Bayes using uni-grams and bi-grams*

### 1.5.2 Using uni-grams, bi-grams and tri-grams with different weights

In this model, we use unigrams, bi-grams and tri-grams. In practical scenario, the probability of three words occurring consecutively in two documents would be very less.Thus, we used the tri-grams in a restricted way, i.e., their contribution is considered only if a match is found in any class. The bi-grams are given more weight than uni-grams and tri-grams due to the same reason as mentioned in section 1.5.1.

**Parameters:** Parameters in the model are $\{\phi_k\}_{k=1}^{r}$ where $\phi_k$ are the same as in original model and $\theta_{pj=l|k}$ are defined as below:

$$\theta_{pj=l|k} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = k\} \sum_{i=1}^{m} 1\{x_{pj}^{(i)} = l\}}{\sum_{i=1}^{m} 1\{y^{(i)} = k\} n_p^{(i)}} \quad p \in \{1,2,3\}$$

$$x_{1j}^{(i)} = j^{th} \text{ word in } i^{th}\, doc$$
$$x_{2j}^{(i)} = j^{th} \text{ bigram in } i^{th}\, doc$$
$$x_{3j}^{(i)} = j^{th} \text{ trigram in } i^{th}\, doc$$

Similarly, $n_p^{(i)}$ represent the number of words, bigrams and trigrams in $i^{th}$ doc for p = 1,2 and 3 respectively.

**Predicting labels:** Now, while calculating P(y|x; $\theta$), we consider all the words, bigrams and trigrams to be seperate features and use $\log \theta_{pj=l|k}$ for words and trigrams as in original model. For giving more weight to bigrams, as discussed above, we use w

* $\log \theta_{pj=l|k}$ where $0 < w < 1$. This would work as it is equivalent to raising the corresponding probability to power w. As probability is less than or equal to 1, $w < 1$ would increase the weight.

**Accuracy:**
1. Training data set: 95.908 %
2. Test data set: 67.493 %

**Confusion Matrix:**
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{vmatrix} 11 & 1 & 4 & 21 & 191 \\ 2 & 0 & 6 & 68 & 250 \\ 4 & 1 & 9 & 238 & 834 \\ 16 & 1 & 2 & 318 & 2771 \\ 73 & 11 & 15 & 182 & 8971 \end{vmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0.065 & 0 & 0.016 & 0.161 & 0.805 \end{vmatrix}$$

**Macro score:** 0.2098

**Total time taken:** 7054.811 s

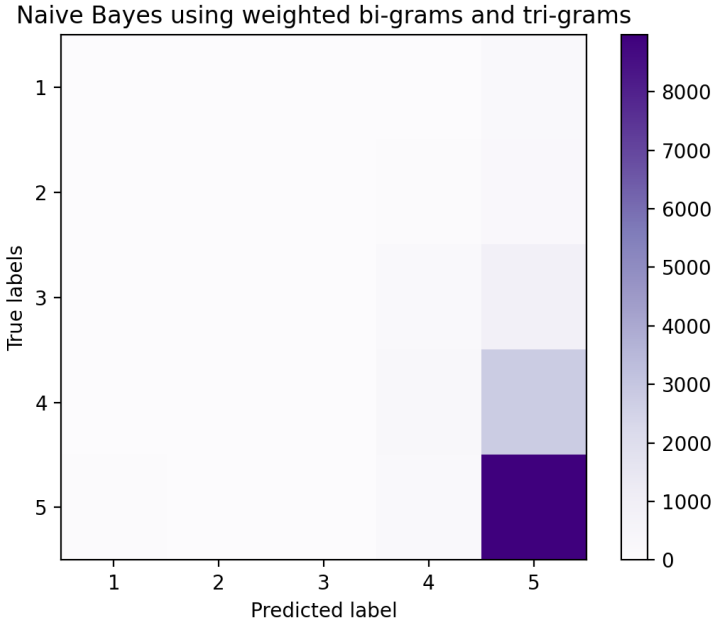



*Figure 1.6: Confusion Matrix for Naive Bayes using uni-grams, bi-grams and tri-grams*

**Observations**
1. Baseline implementation (Simple Naive Bayes Classifier):
Training set: 69.918 %
Test dataset: 65.464 %
2. Feature transformation 1:
Training set: 78.292 %
Test dataset: 66.492 %
3. Feature transformation 2:
Training set: 95.908 %
Test data set: 67.493 %
We observe an increase in accuracy over both training and test data sets. compared to the baseline model. This may be due to enriching features provided to the model.

## 1.6 F1 score

F1 score has already been reported for each model above.

## 1.7 Using Summary field for predictions

We used the words from summary along with the words from reviewText field both while training our model and making predictions. **Accuracy:**

1. Training data set: 54.482 %
2. Test data set: 65.871 %
We observe an increase in accuracy over test data sets. compared to the baseline model.
**Confusion Matrix:**
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{bmatrix} 0 & 0 & 1 & 5 & 222 \\ 0 & 0 & 0 & 7 & 319 \\ 0 & 0 & 0 & 20 & 1066 \\ 0 & 0 & 0 & 28 & 3080 \\ 8 & 0 & 9 & 41 & 9194 \end{bmatrix}$$

**F1 score:**

$$\begin{vmatrix} 0 & 0 & 0 & 0.017 & 0.794 \end{vmatrix}$$

**Macro score:** 0.1624

**Total time taken:** 415.992 s

# MNIST Digit Classification

## Problem Statement

In this problem, we will use Support Vector Machines (SVMs) to build a handwritten digit classifier. Each row in the (train/test) data file corresponds to an image of size 28x28, represented as a vector of grayscale pixel intensities followed by the label associated with the image. Every column represents a feature where the feature value denotes the gray-scale value (0-255) of the corresponding pixel in the image. There is a feature for every pixel in the image. Last column gives the corresponding label.

## 2.1 Binary Classification

Let d be the last digit of your entry number. Then, we would start with binary classification problem over the images of digits (d vs (d+ 1) mod 10) in this Section.
My entry number: 2019CS10354. So, the model is trained using CVXOPT to classify digits 4 and 5.

### 2.1.1 Linear Kernel

**Mathematical Formulation**
Goal: Minimize *W($\alpha$)* given by

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)T}x^{(j)} - \sum_{i=1}^{m}\alpha_i$$

subject to the constarints

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

While writing the problem for CVXOPT format, we had to find P,q,G,A,h and b such that problem becomes equivalent to

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

subject to

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

This gave us the following values for the parameters:

$$P = ZZ^T \qquad Z_{ij} = x_j^{(i)} y^{(i)}$$

$$q = \begin{bmatrix} 1 \\ 1 \\ . \\ \\ 1 \\ 1 \end{bmatrix} \qquad h = \begin{bmatrix} C \\ . \\ . \\ C. \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \qquad G = \begin{bmatrix} 1 & 0 & . & . & . & 0 \\ 0 & 1 & . & . & . & 0 \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ 0 & . & . & . & 0 & 1 \\ -1 & 0 & . & . & . & 0 \\ 0 & -1 & . & . & . & 0 \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ 0 & . & . & . & 0 & -1 \end{bmatrix} \quad A = y^T \qquad b = 0$$

**Results**
1. Accuracy on test data set: 98.665%
2. Accuracy on training data set: 100 %
3. Number of support vectors: 151
4. Value of parameter b: 1.4567
5. Time taken to learn the model: 34.436 s
6. Time taken to make predictions: 1.275 s
7. Number of iterations: 14

### 2.1.2 Gaussian Kernel

**Mathematical Formulation** The goal is again to minimise W($\alpha$) given by

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)}) - \sum_{i=1}^{m}\alpha_i$$

In this case, we get the following expression for P. Rest of the parameters are same as in previous case.

$$P_{ij} = y^{(i)}y^{(j)}\phi(x^{(i)T})\phi(x^{(j)})$$

**Results**
1. Accuracy on test data set: 99.893%
2. Accuracy on training data set: 100 %
3. Number of support vectors: 1815
4. Value of parameter b: 0.2471
5. Time taken to learn the model: 30.793 s
6. Time taken to make predictions: 165.653 s
7. Number of iterations: 9

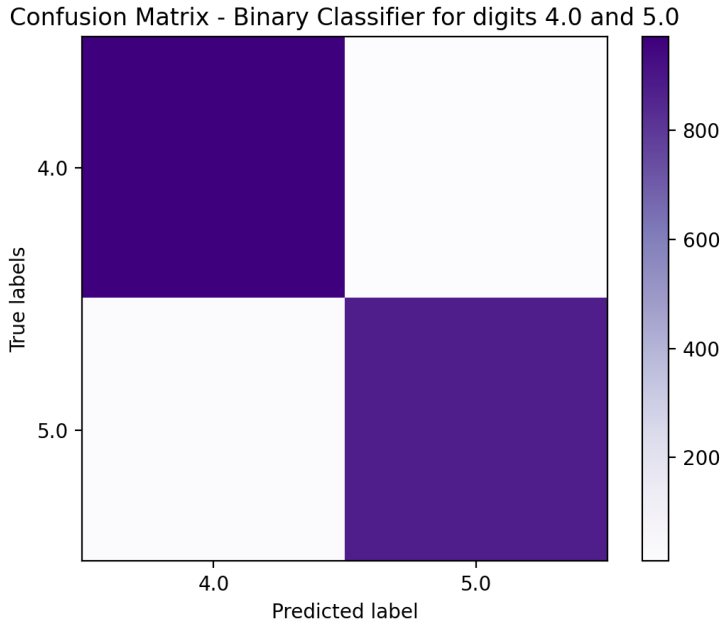



*Figure 2.1: Confusion Matrix for Binary Classification using Linear SVM*

### 2.1.3 Using LIBSVM

#### Linear Kernel

1. Accuracy on test data set: 98.665%
2. Accuracy on training data set: 100 %
3. Number of support vectors: 150
4. Time taken to learn the model: 0.375 s
5. Time taken to make predictions: 1.681 s

#### Gaussian Kernel

1. Accuracy on test data set: 99.893%
2. Accuracy on training data set: 100 %
3. Number of support vectors: 1459
4. Time taken to learn the model: 2.586 s
5. Time taken to make predictions: 2.652 s

**Observations**

1. The accuracy on both training and test data sets using LIBSVM matches exactly with our implementation.
2. Number of support vectors are almost same for Linear Kernels.
3. For Gaussian Kernels, number of support vectors are less using LIBSVM.
4. The time taken in learning the model and making predictions is considerably less using LIBSVM for the Gaussian kernels.
5. The time taken is less for Linear Kernel as compared to Gaussian, probably due to less computations needed for Linear SVM implementation.

## 2.2 Multi Class Classification

### 2.2.1 One vs One Multi-Classifier

In order to extend the above binary classifier to the multi-class setting, we train a model on each pair of classes to get $\binom{k}{2}$ classifiers, k being the number of classes. During prediction time, we output the class which has the maximum number of votes from all the $\binom{k}{2}$ classifiers. We train a soft margin Gaussian Kernel with C =1.0 and $\gamma = 0.05$. In case of ties, label with the highest score is chosen by considering the prediction of binary classifier for the digits predicted equal number of times.

**Results:**
1. Accuracy on test data set: 97.01%
2. Accuracy on training data set: 99.94 %

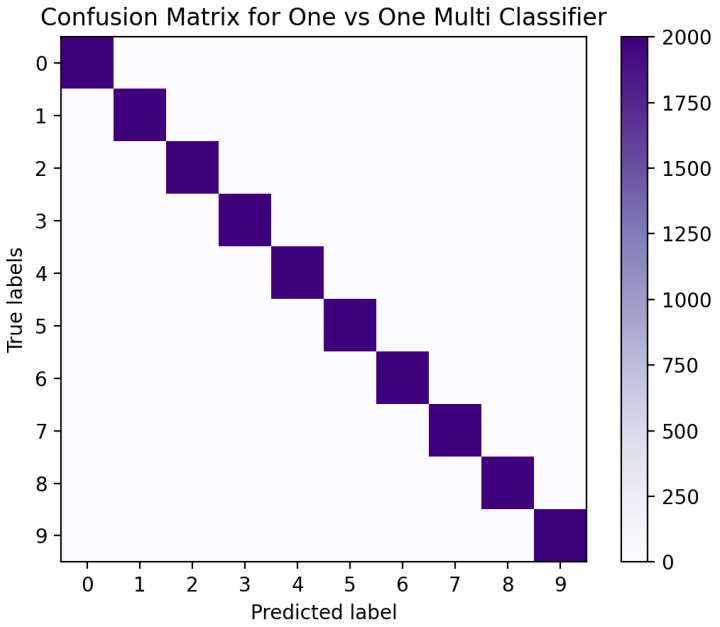



*Figure 2.2: Confusion Matrix for Multi Classifier using Gaussian Kernels*

### 2.2.2 Multi Class Classification using LIBSVM

**Results**
1. Accuracy on test data set: 99.92%
2. Accuracy on training data set: 99.92 %
3. Time taken to learn the model: 117.05 s

**Observations**
1. The models obtained by using LIBSVM are slightly better than our implementation.

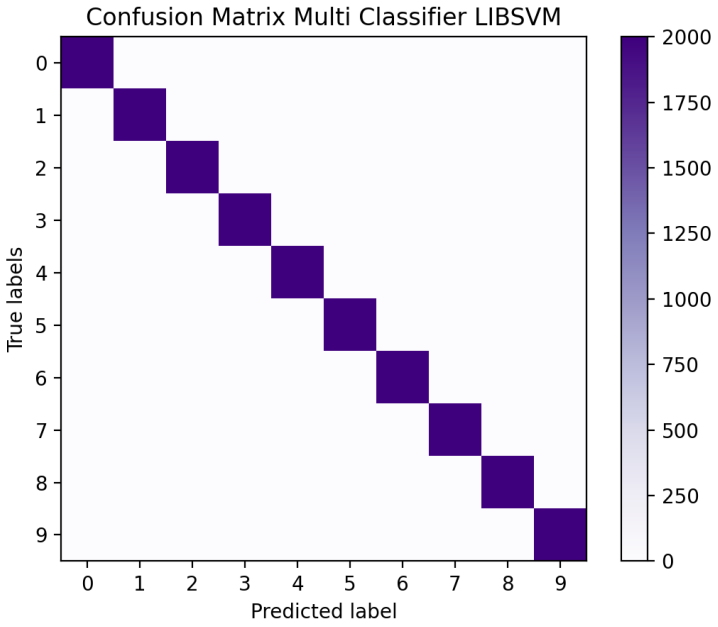



*Figure 2.3: Confusion Matrix for Multi Classifier using LIBSVM Gaussian Kernels*

2. The time taken to learn the models by our implementation was about 20-25 minutes while it is only a few seconds using LIBSVM.

### 2.2.3 Analysing the Confusion Matrices

Confusion Matrix for our implementation of Multi Classifier on Training data set (Cij = number of documents of class j, classified into class i by our model )

$$\begin{vmatrix} 2000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1998 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1997 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1997 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1996 \end{vmatrix}$$

Confusion Matrix for our implementation of Multi Classifier on Test data set
(Cij = number of documents of class j, classified into class i by our model )

$$\begin{vmatrix} 972 & 0 & 9 & 0 & 2 & 3 & 6 & 2 & 9 & 8 \\ 0 & 1124 & 0 & 0 & 0 & 0 & 4 & 4 & 0 & 6 \\ 1 & 3 & 984 & 3 & 9 & 4 & 1 & 7 & 1 & 3 \\ 0 & 1 & 8 & 993 & 0 & 7 & 0 & 10 & 10 & 8 \\ 0 & 0 & 0 & 0 & 995 & 1 & 1 & 2 & 2 & 21 \\ 0 & 1 & 0 & 3 & 0 & 855 & 2 & 0 & 2 & 0 \\ 4 & 2 & 0 & 0 & 5 & 7 & 941 & 0 & 2 & 1 \\ 1 & 0 & 14 & 3 & 1 & 1 & 1 & 985 & 2 & 6 \\ 2 & 4 & 17 & 6 & 1 & 12 & 2 & 4 & 943 & 7 \\ 0 & 0 & 0 & 2 & 9 & 2 & 0 & 14 & 3 & 949 \end{vmatrix}$$

Confusion Matrix for LIBSVM implementation of Multi Classifier on Training data set
(Cij = number of documents of class i, classified into class j by our model )

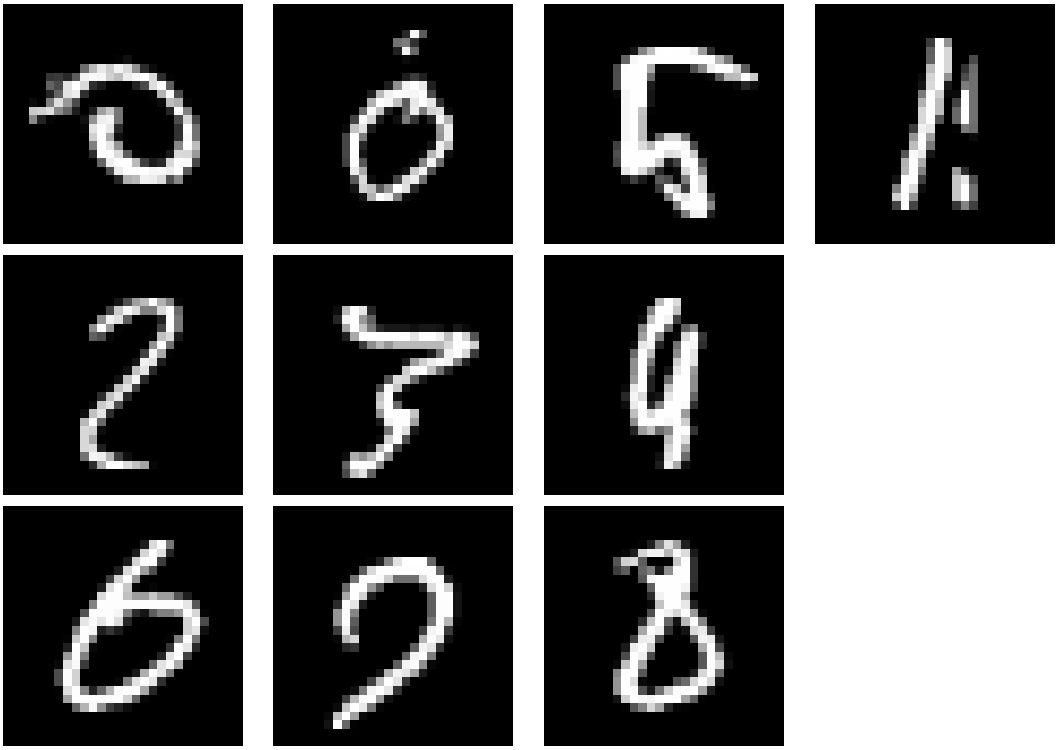
$$\begin{vmatrix} 2000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1997 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1999 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1999 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1999 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1995 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1999 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 1996 \end{vmatrix}$$

Confusion Matrix LIBSVM implementation of Multi Classifier on Test data set
(Cij = number of documents of class i, classified into class j by our model )

$$\begin{vmatrix} 2000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1997 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1999 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1999 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1999 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1995 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1999 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 1996 \end{vmatrix}$$

**Observations**
1. The diagonal elements are almost always order of magnitude larger than non diagonal entries. This indicates high accuracy of model.
2. Digits 3 and 8 have the highest rate of being mis-classified.
3. This makes intuitive sense as the digits 8 looks just like the digit 3 plus its mirror image. There may be ambiguity due to the handwriting which gets reflected in the feature vectors and hence leads to mis-classification.

Following are some of the images which were mis-classified by our model.



### 2.2.4 k-fold Cross Validation

We use 5-fold cross validation to estimate the best value of the C parameter for the Gaussian kernel case by fixing  as 0.05 and varying the value of C in the set { $10^{-5}, 10^{-3}$, 1, 5, 10}.
The results are tabulated below:

| C | Validation Set 1 Accuracy | Validation Set 2 Accuracy | Validation Set 3 Accuracy | Validation Set 4 Accuracy | Validation Set 5 Accuracy | Max Validation Set Accuracy | Test Set Accuracy |
|---|---|---|---|---|---|---|---|
| $10^{-5}$ | 9.15 | 9.4 | 9.25 | 9.0 | 9.375 | 9.4 | 72.08 |
| $10^{-3}$ | 9.15 | 9.4 | 9.25 | 9.0 | 9.375 | 9.4 | 72.08 |
| 1 | 97.39 | 97.75 | 96.95 | 96.95 | 97.3 | 97.75 | 97.23 |
| 5 | 97.55 | 97.8 | 97.125 | 97.075 | 97.475 | 97.8 | 97.3 |
| 10 | 97.55 | 97.8 | 97.125 | 97.075 | 97.475 | 97.8 | 97.3 |

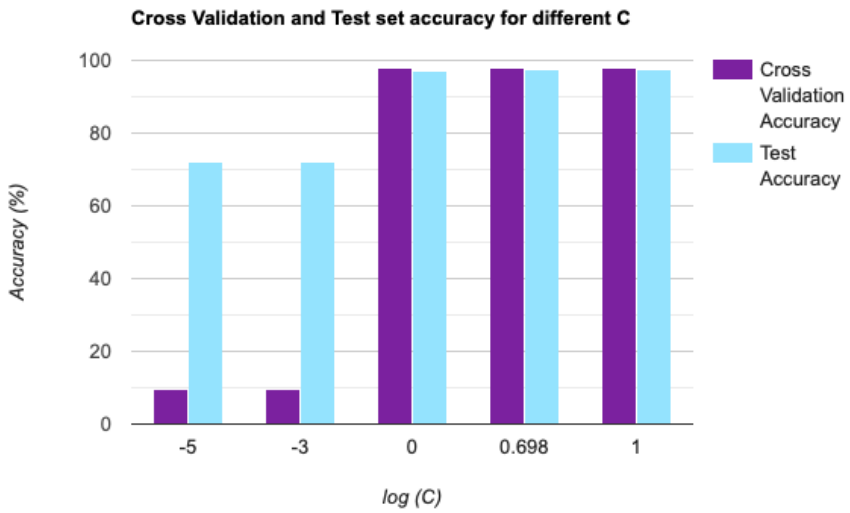



*Figure 2.4: Plot of max validation set accuracy and test set accuracy for different values of C*