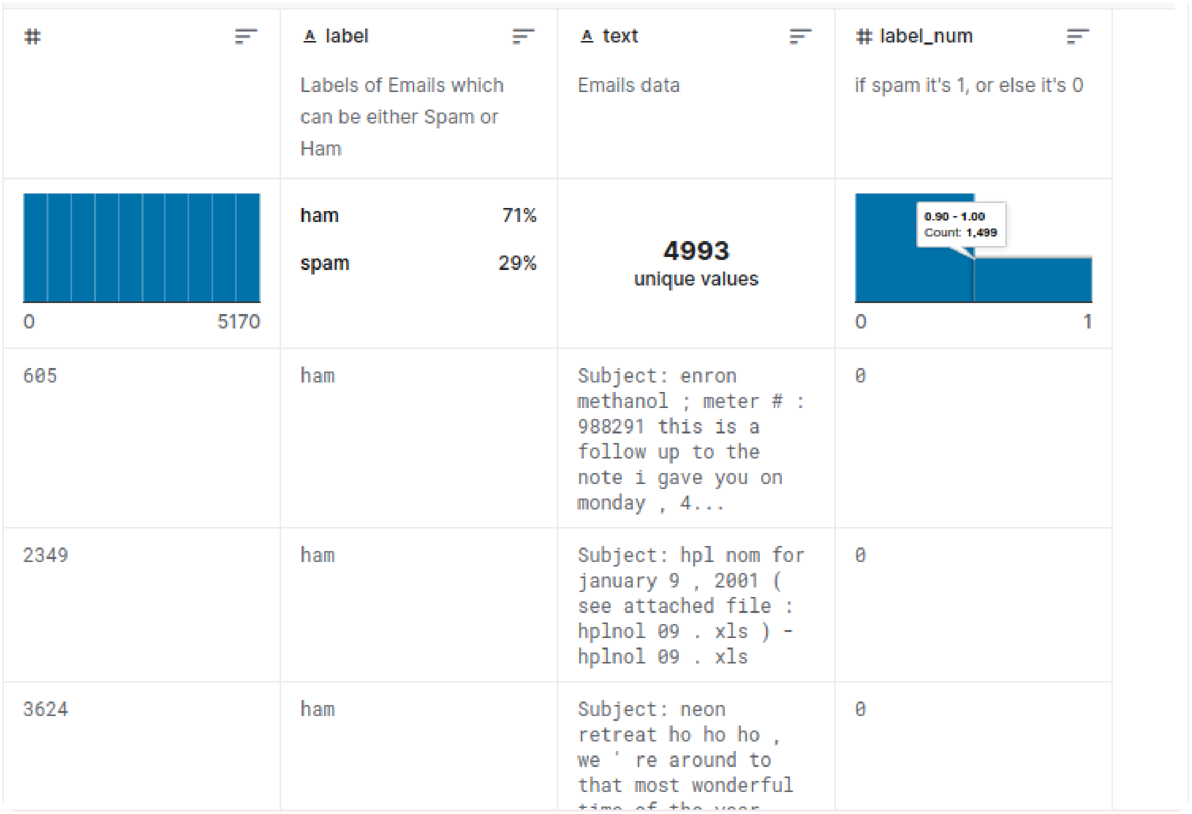
Name - NILESH TIWARI
Roll Number - CS22M059

1. DataSet Chosen

I chose the dataset publicly available on Kaggle website for spam-ham classification.
The link for the dataset is - https://www.kaggle.com/datasets/venky73/spam-mails-dataset. This dataset contains 5171 emails. This dataset is in CSV format with the column labels as ‘text , ‘label’ and ‘label_num. ‘text’ contains the text of the email, and ‘label_num’ contains the label=1 for spam and 0 for non-spam. This dataset has a large number of emails with a substantial number of emails categorized as spam. Hence, it is preferred. The following figure gives a overview of the dataset.



#	label	text	# label_num
	Labels of Emails which can be either Spam or Ham	Emails data	if spam it's 1, or else it's 0
0 5170	ham 71% spam 29%	4993 unique values	0.90 - 1.00 Count: 1,499 0 1
605	ham	Subject: enron methanol ; meter # : 988291 this is a follow up to the note i gave you on monday , 4...	0
2349	ham	Subject: hpl nom for january 9 , 2001 (see attached file : hplnol 09 . xls) - hplnol 09 . xls	0
3624	ham	Subject: neon retreat ho ho ho , we ' re around to that most wonderful	0

Fig. 1. Dataset

2. Features extracted

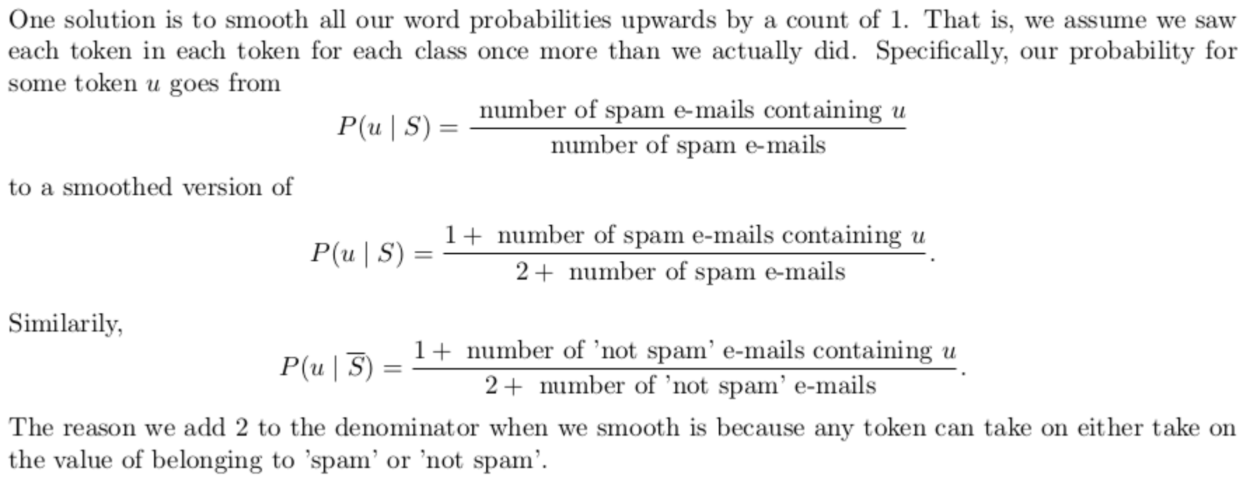
The ‘text’ part of each datapoint(email) was preprocessed using a function that eliminates any URLs, HTML tags, punctuation symbols, digits, and extra whitespaces. From this preprocessed text, we obtain a large number of unique words. We choose the top 1000 words by frequency as the features for each datapoint. For the Naive Bayes algorithm, each datapoint has a particular feature with value=1 if the word is present in the datapoint else 0. For SVM, each datapoint has a particular feature with value=count of that word in the datapoint.

3. Algorithm used

Two algorithms are used for classification -

a. **Naive Bayes**

1. For each feature, find the conditional probabilities of each feature given the label of the class. Here, I have used Laplacian Smoothing to deal with the case when a given word and class never occur together in the training data, then the probability estimate for that word conditioned on that class will be zero. The formula used for Laplacian smoothing is given below.



One solution is to smooth all our word probabilities upwards by a count of 1. That is, we assume we saw each token in each token for each class once more than we actually did. Specifically, our probability for some token u goes from

$$P(u \mid S) = \frac{\text{number of spam e-mails containing } u}{\text{number of spam e-mails}}$$

to a smoothed version of

$$P(u \mid S) = \frac{1 + \text{ number of spam e-mails containing } u}{2 + \text{ number of spam e-mails}}.$$

Similarily,

$$P(u \mid \overline{S}) = \frac{1 + \text{ number of 'not spam' e-mails containing } u}{2 + \text{ number of 'not spam' e-mails}}.$$

The reason we add 2 to the denominator when we smooth is because any token can take on either take on the value of belonging to 'spam' or 'not spam'.

Fig. 2. Laplacian Smoothing

2. Using the Bayes theorem, we obtain P(y_test=1 | x_test) and P(y_test=0 | x_test).
3. If P(y_test=1 | x_test) > P(y_test=0 | x_test), then predict 1 else 0.
4. To find the above-mentioned probabilities, we have to find the product of conditional probabilities for each feature. This can cause underflow, hence we used the log of the probability values and summed them.

b. **SVM**

1. I used the SVM library function of sklearn for the classification algorithm.
2. The parameters selected to fit the model are kernel='rbf' and class_weights='balanced'.
3. RBF kernels used to make the data linear separable in higher dimensions.
4. As the number of datapoints for non-spam class is less than half of the spam class, the classes become imbalance. Due to this difference in each class, the algorithms tend to get biased towards the majority values present and don't perform well on the minority values. When the class_weights = 'balanced', the model automatically assigns the class weights inversely proportional to their respective frequencies. So, the classes become balanced.

4. Results obtained

Following results are obtained :

a. Naive Bayes

```
Accuracy on train data set = 80.03866602223296
Confusion Matrix on train data set :
[[2112  826]
 [   0 1200]]
Precision on train data set = 0.5923000987166831
Recall on train data set = 1.0

Accuracy on test data set = 83.4462729912875
Confusion Matrix on test data set :
[[563 171]
 [  0 299]]
Precision on test data set = 0.6361702127659574
Recall on test data set = 1.0
```

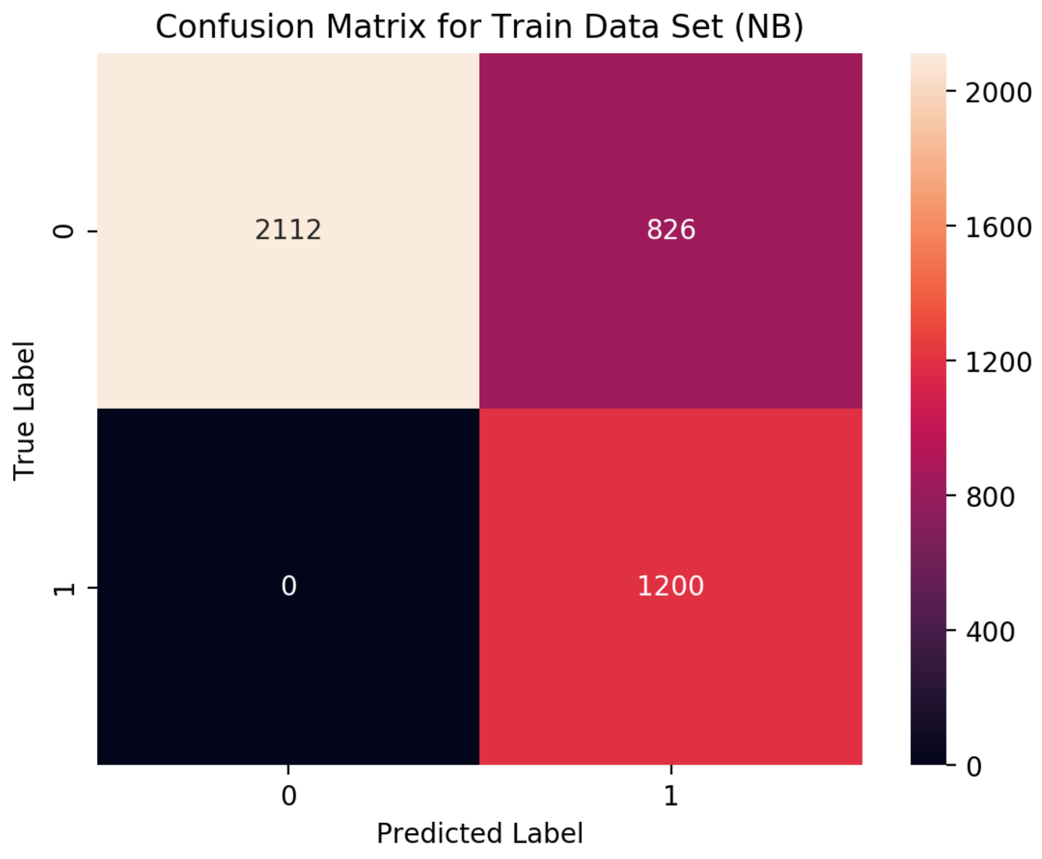


Fig. 3. Confusion matrix for train dataset (Naive Bayes)

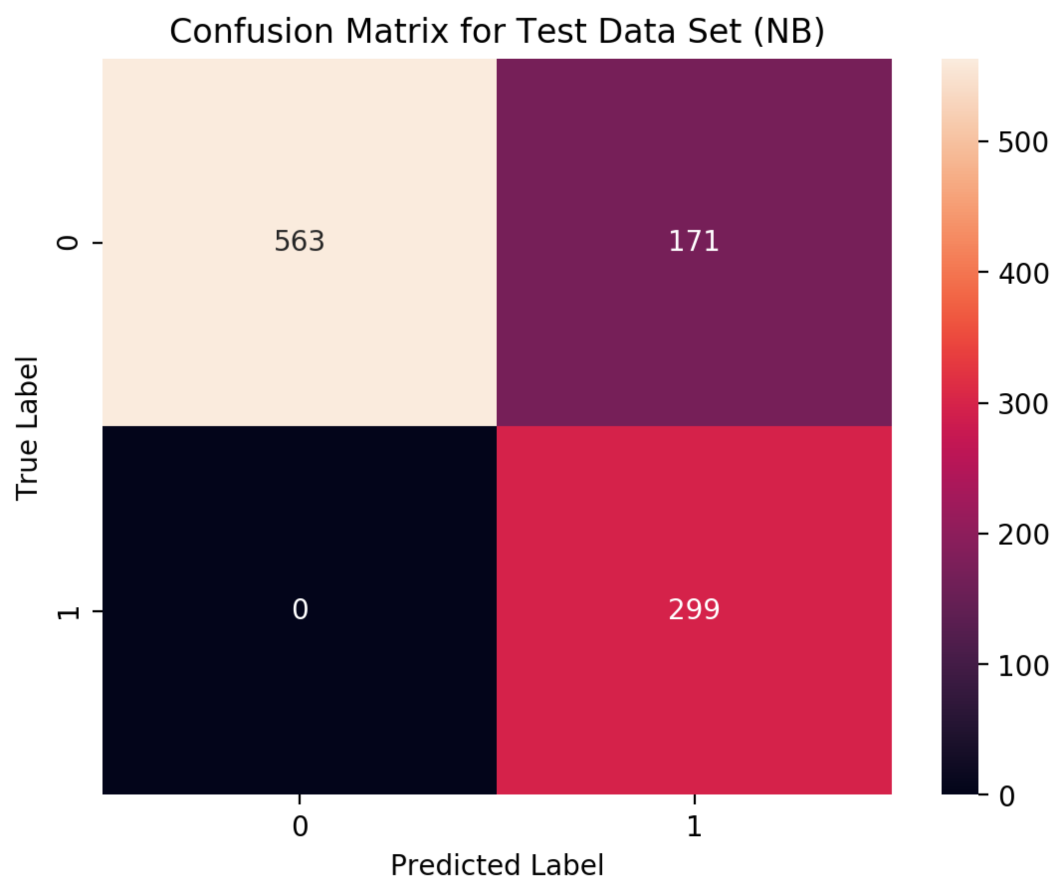


Fig. 4. Confusion matrix for test dataset (Naive Bayes)

b. SVM

```
Accuracy on train data set = 92.70178830352828
Confusion Matrix on train data set :
[[2636  302]
 [   0 1200]]
Precision on train data set = 0.7989347536617842
Recall on train data set = 1.0

Accuracy on test data set = 93.80445304937076
Confusion Matrix on test data set :
[[674  60]
 [  4 295]]
Precision on test data set = 0.8309859154929577
Recall on test data set = 0.9866220735785953
```

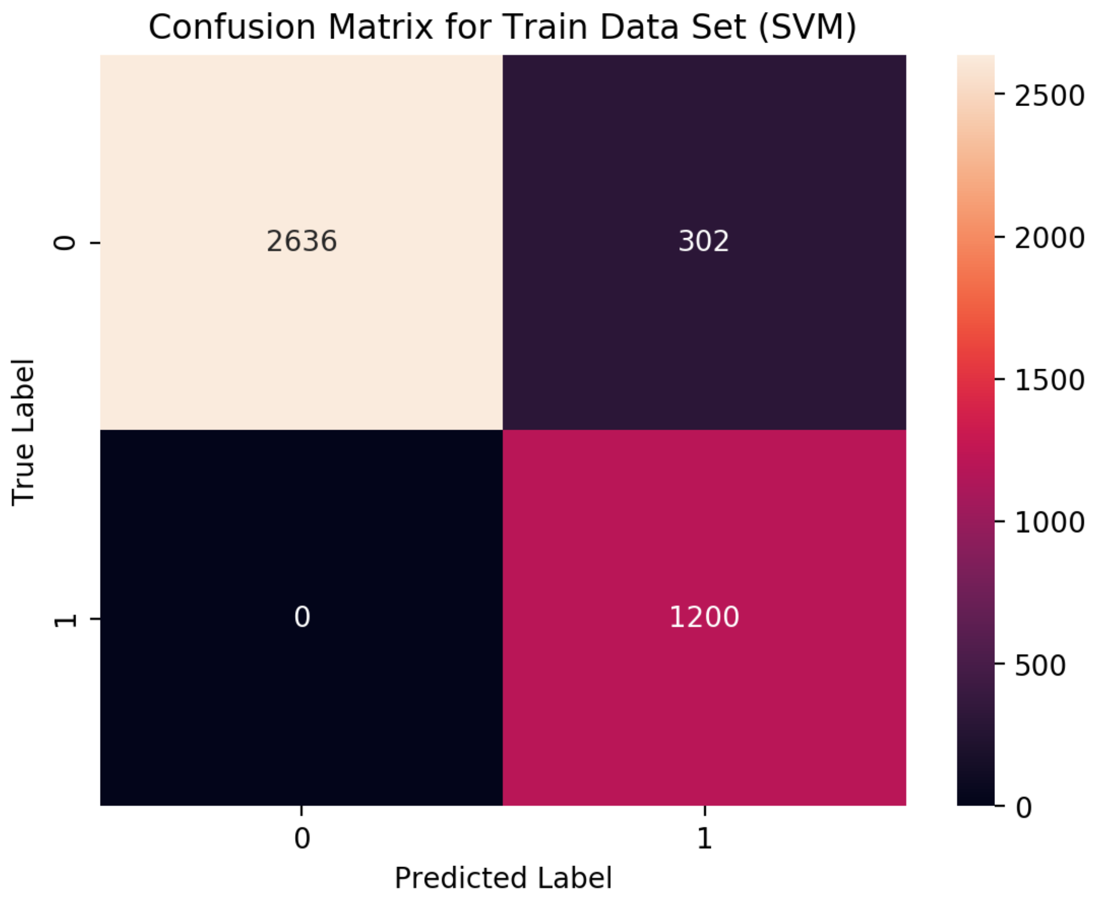


Fig. 5. Confusion matrix for train dataset (SVM)

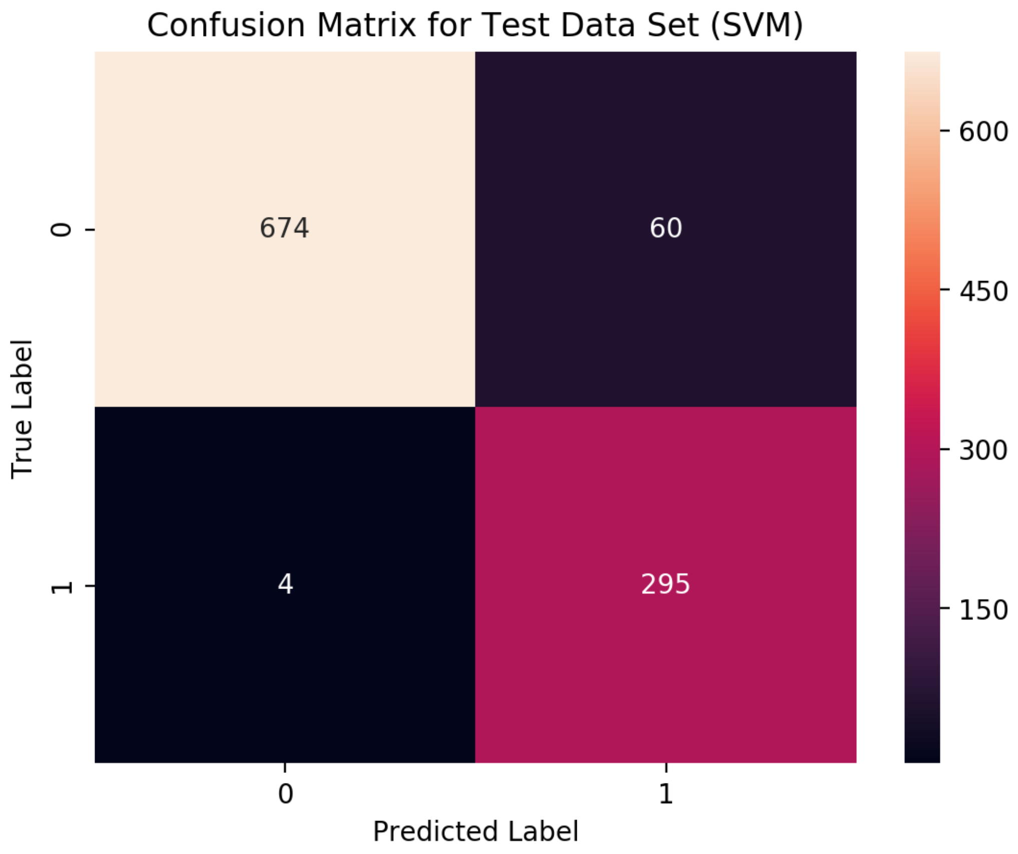


Fig. 6. Confusion matrix for test dataset (SVM)

5. Classifier chosen for prediction

The classifier chosen for the prediction is SVM because we can see from the results obtained, that its accuracy and precision on the dataset are higher than the accuracy and precision of the Naive Bayes algorithm. False positive means that the email was not spam but we labeled it spam. Increasing precision involves decreasing false positives. We don’t want to miss important emails, hence we would decrease False positives and care more for precision.