**I. ABSTRACT**

Being the fast and economical means of communication has prompted many to use email as the main communication medium for both official and personal purposes. However, the rapid increase in the number of e-mail users has resulted in a dramatic increase in the number of spams in recent years. Spam mail has become an increasing menace as it increases the chances of virus threats, communication overload, wastage of time, irritation and disturbance, etc., to the users. Hence, there is a need for developing efficient spam filters. Several classification algorithms for mining text are being employed to classify e-mails as legitimate or otherwise. Comparison of these algorithms using some machine learning technique are to be conducted in order to determine which algorithm is better efficient in classifying the e-mails. In this project, two algorithms classifying spam on UCI dataset, namely, Naïve Bayes and Support Vector Machine, were investigated for their classification accuracy in the COLAB environment. The in-depth analysis of the previous studies and descriptions of two algorithms on the basis of parameters, such as ‘accuracy’, ’precision’ and ’recall’ to measure the performance of these two classification algorithms was performed and the result was analyzed. The result reveals that the Support Vector Machine (SVM) gave the most accurate result among these two algorithms.

Keywords- email classification; text mining; classification algorithm;

**II. INTRODUCTION**

Spam could be defined as bulk unsolicited emails received without one’s permission that causes discomfort and concern (Kelman, 2004). The senders of such mails usually do not intend to target the recipients personally. Their addresses are gathered based on specific information, and numerous recipients are simultaneously targeted (Kelman, 2004). In recent years, spam mails have been increasing alarmingly, prompting a need for anti-spam filters that are reliable, accurate, and can effectively classify legitimate mails from spam. Several text mining and machine-learning techniques to classify spam mail have been used such as Naïve Bayes (Androutsopoulos et al., 2000; Metsis et al., 2006), rule learning, (Cohen, 1996) and Support Vector Machines (SVM) (Drucker and Vapnik, 1999; Wang et al., 2006).

Spammers gather email IDs from sources, such as chat rooms, websites, newsgroups and malware that gather address details of users. These details are available to other spammers for a price. After purchasing the same, bulk messages are sent, the volumes of which create enormous productivity losses to IT firms. They pose serious security threats and are carriers for phishing of classified information. Hence, the classification of emails is of prime importance. In spite of the increased number of classification algorithms available today, there is none that can claim to be 100% accurate as each utilizes only limited features for classification. Therefore, selecting the best algorithm is a difficult task as their advantages need to be weighed against drawback.

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser. It is capable of performing tasks such as pre-processing, statistical processing and visualization of data (<https://colab.research.google.com/>). Algorithm such as Naïve Bayes and Support Vector Model (SVM) have been reviewed in this project, and their applicability in classifying spam mail has been presented. Description of these algorithms and a comparison of their performance using the Colaboratory environment have been reported in this project.

We give a short review of relevant literature, followed by a detailed description of the four classification algorithms. We then present the experimental details followed by results and discussion. Lastly, we present the conclusions followed by avenues for future work..

**III. ALGORITHM FOR TEXT CLASSIFICATION**

DESCRIPTION OF CLASSIFICATION ALGORITHMS

1. NAIVE BAYES CLASSIFICATION ALGORITHM

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes’ theorem.

Bayes’ Theorem

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:



Where A and B are events and P (B)? 0.

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
* P (A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).
* P (A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

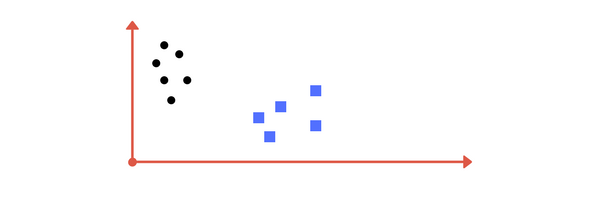
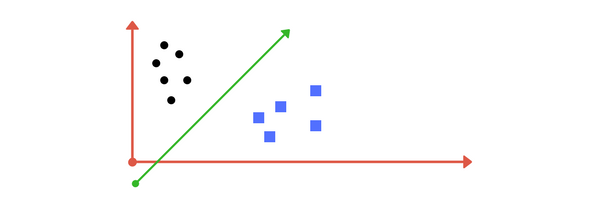
Naive Bayes is an easy method for creating classifiers which are models designating labels of class to problem instances, presented as vectors of feature values, where the class labels are taken from a limited group. They form a multiple set of algorithms having a common idea that all such classifiers consider that a specific feature’s value does not depend on any other’s value, the class variable being given.

1. SUPPORT VECTOR MACHINE (SVM)

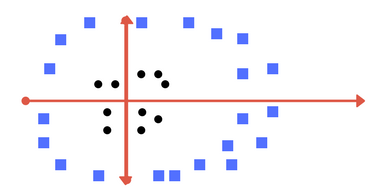
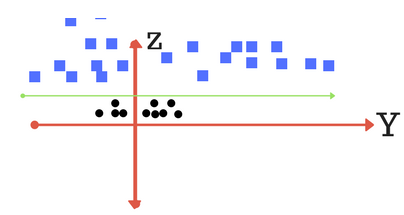
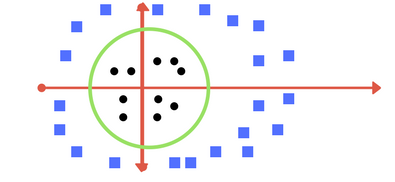
A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side

Suppose you are given plot of two label classes on graph as shown in image

Can you decide a separating line for the classes?

Let’s assume value of points on z plane, w = x² + y².



TUNING PARAMETERS: KERNEL, REGULARIZATION, GAMMA AND MARGIN.

# ****Kernel****

# The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra.

**Linear kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

f(x) = B(0) + sum(ai \* (x,xi))

The **polynomial kernel** can be written as K(x,xi) = 1 + sum(x \* xi)^d and **exponential** as K(x,xi) = exp(-gamma \* sum((x — xi²)). [Source for this excerpt: <http://machinelearningmastery.com/>].

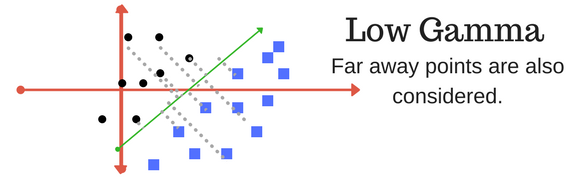
**\*Polynomial and exponential kernels calculates separation line in higher dimension. This is called kernel trick**

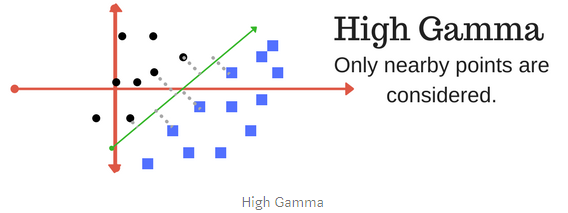
# Regularization

# The Regularization parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

# Gamma

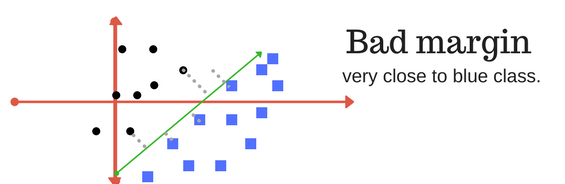
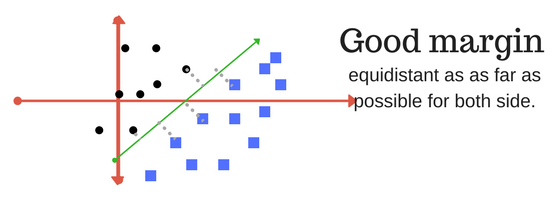
The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.



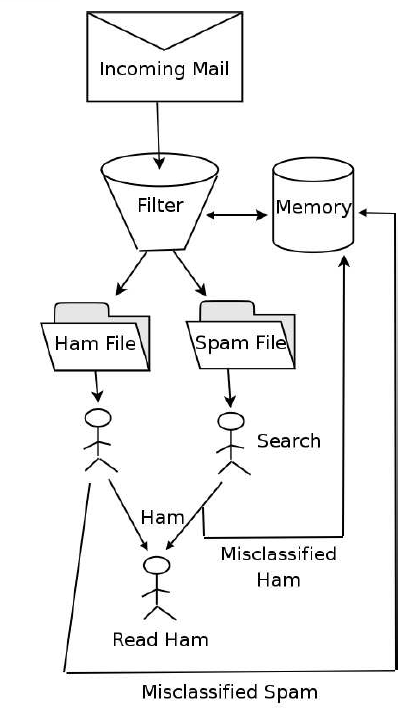


# Margin

A margin is a separation of line to the closest class points.



**IV.ARCHITECTURE**

****

**V.CODE**

IMPORTING THE LIBRARIES AND UPLOADING DATA SET ON COLAB

# Spam classification with Naive Bayes and Support Vector Machines.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from collections import Counter

from sklearn import feature\_extraction, model\_selection, naive\_bayes, metrics, svm

from IPython.display import Image

import warnings

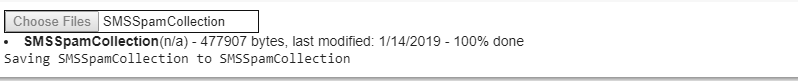
warnings.filterwarnings("ignore")

#%matplotlib inline

**#** Uploading UCI data on google colab.

from google.colab import files

upload = files.upload()

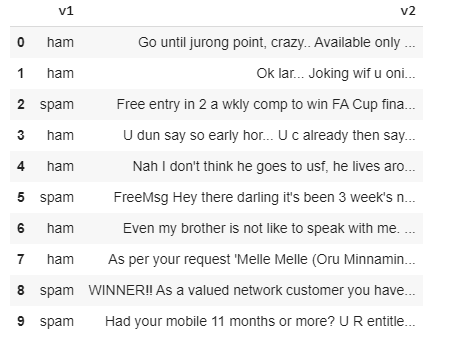


FETCING AND FEATURE EXTRACTION THE UCI DATA SET

# Reading and verifying the UCI dataset

data = pd.read\_csv('SMSSpamCollection', sep='\t', names=["v1", "v2"])

data.head(10)



#feature extraction using count vectorizer

f = feature\_extraction.text.CountVectorizer(stop\_words = 'english')

X = f.fit\_transform(data["v2"])

np.shape(X)



TRAINING THE DATA SET

data["v1"]=data["v1"].map({'spam':1,'ham':0})

X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(X, data['v1'], test\_size=0.33, random\_state=42)

print([np.shape(X\_train), np.shape(X\_test)])

print([np.shape(y\_train), np.shape(y\_test)])



MODELING THE DATA USING NAIVE BAYES

Train different bayes models changing the regularization parameter α.

Evaluating the accuracy, recall and precision of the model with the test set.

list\_alpha = np.arange(1/100000, 20, 0.11)

score\_train = np.zeros(len(list\_alpha))

score\_test = np.zeros(len(list\_alpha))

recall\_test = np.zeros(len(list\_alpha))

precision\_test= np.zeros(len(list\_alpha))

count = 0

for alpha in list\_alpha:

    bayes = naive\_bayes.MultinomialNB(alpha=alpha)

    bayes.fit(X\_train, y\_train)

    score\_train[count] = bayes.score(X\_train, y\_train)

    score\_test[count]= bayes.score(X\_test, y\_test)

    recall\_test[count] = metrics.recall\_score(y\_test, bayes.predict(X\_test))

    precision\_test[count] = metrics.precision\_score(y\_test, bayes.predict(X\_test))

    count = count + 1

SELECTING THE MODEL WITH THE MOST TEST PRECISION

best\_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()

bayes = naive\_bayes.MultinomialNB(alpha=list\_alpha[best\_index])

bayes.fit(X\_train, y\_train)

models.iloc[best\_index, :]

alpha 15.290010

Train Accuracy 0.975355

Test Accuracy 0.978793

Test Recall 0.841463

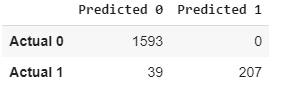
Test Precision 1.000000

Name: 139, dtype: float64

CONFUSION MATRIX FOR NAÏVE BAYES MODEL

m\_confusion\_test = metrics.confusion\_matrix(y\_test, bayes.predict(X\_test))

pd.DataFrame(data = m\_confusion\_test, columns = ['Predicted 0', 'Predicted 1'],index = ['Actual 0', 'Actual 1'])



MODELING THE DATA USING SUPPORT VECTOR MACHINE

Going to apply the same reasoning applying the support vector machine model with the gaussian kernel.

Train different models changing the regularization parameter C.

Evaluate the accuracy, recall and precision of the model with the test set

list\_C = np.arange(500, 2000, 100)

score\_train = np.zeros(len(list\_C))

score\_test = np.zeros(len(list\_C))

recall\_test = np.zeros(len(list\_C))

precision\_test= np.zeros(len(list\_C))

count = 0

for C in list\_C:

    svc = svm.SVC(C=C)

    svc.fit(X\_train, y\_train)

    score\_train[count] = svc.score(X\_train, y\_train)

    score\_test[count]= svc.score(X\_test, y\_test)

    recall\_test[count] = metrics.recall\_score(y\_test, svc.predict(X\_test))

    precision\_test[count] = metrics.precision\_score(y\_test, svc.predict(X\_test))

    count = count + 1

SELECTING THE MODEL WITH THE MOST TEST PRECISION

best\_index = models['Test Precision'].idxmax()

models.iloc[best\_index,:]

C 500.000000

Train Accuracy 0.993303

Test Accuracy 0.986406

Test Recall 0.898374

Test Precision 1.000000

Name: 0, dtype: float64

best\_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()

svc = svm.SVC(C=list\_C[best\_index])

svc.fit(X\_train, y\_train)

models.iloc[best\_index, :]

C 1000.000000

Train Accuracy 0.998393

Test Accuracy 0.988581

Test Recall 0.914634

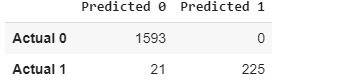
Test Precision 1.000000

Name: 5, dtype: float64

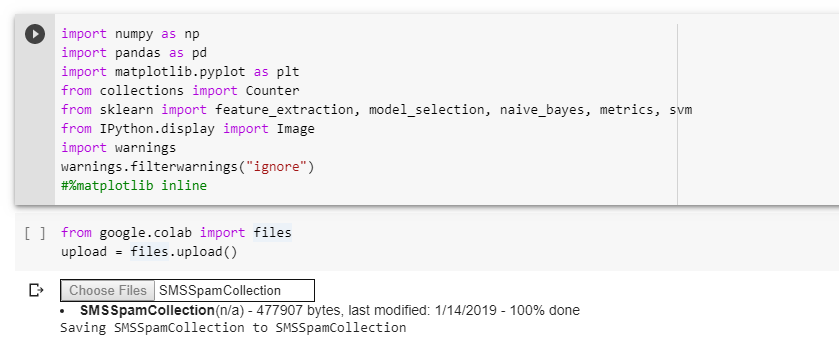
CONFUSION MATRIX FOR SUPPORT VECTOR MODEL

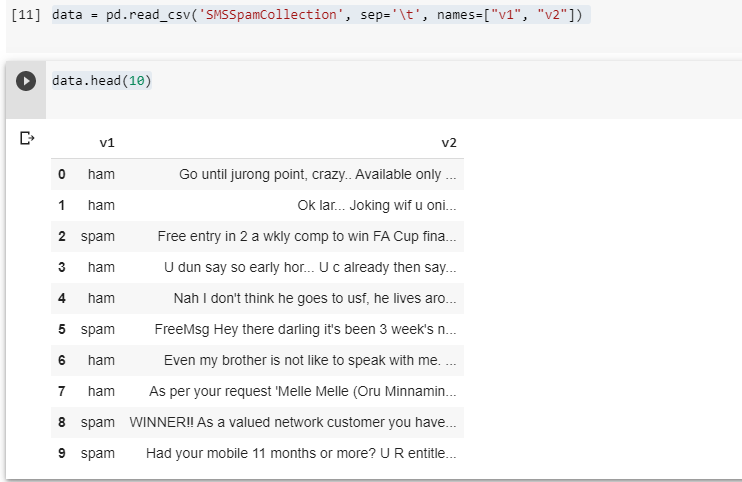
m\_confusion\_test = metrics.confusion\_matrix(y\_test, svc.predict(X\_test))

pd.DataFrame(data = m\_confusion\_test, columns = ['Predicted 0', 'Predicted 1'],index = ['Actual 0', 'Actual 1'])

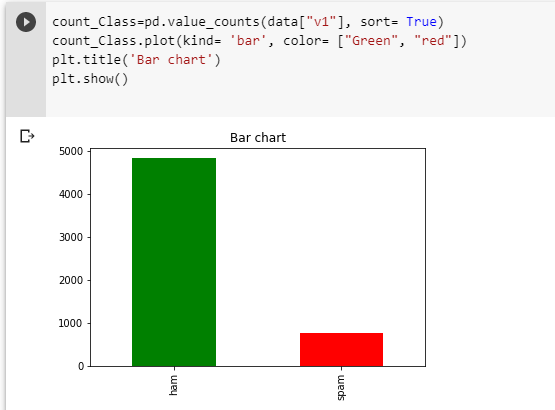


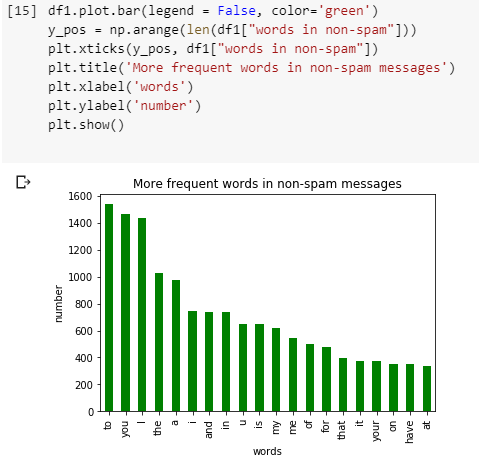
**VI. SCREENSHOTS OF OUTPUTS**

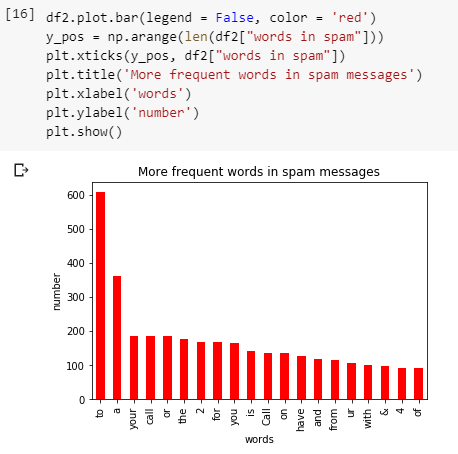
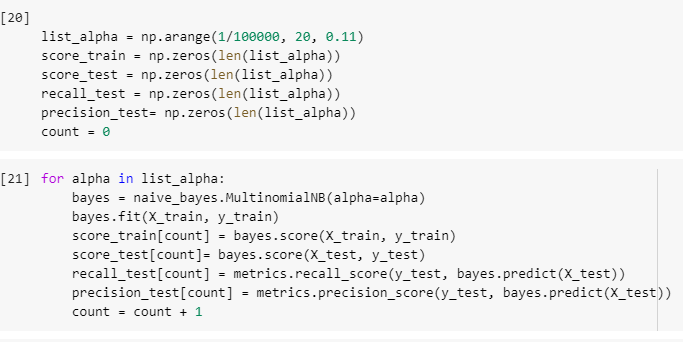


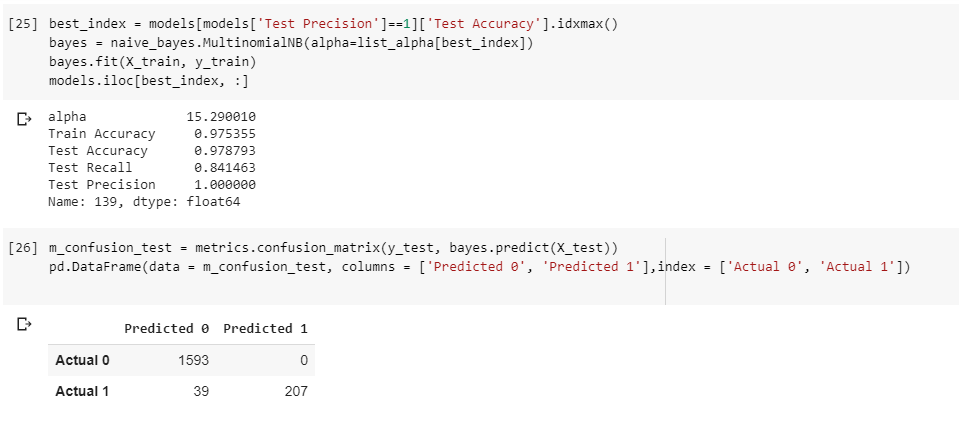


## DISTRIBUTION SPAM/NON-SPAM PLOTS

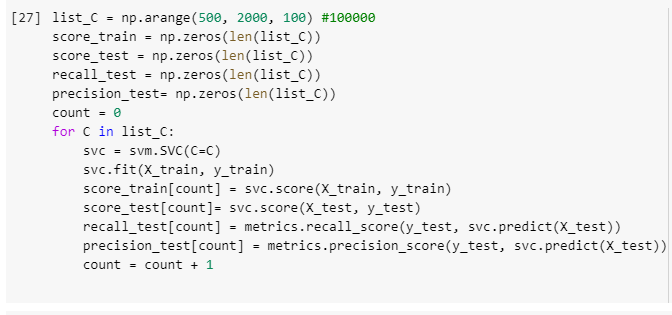


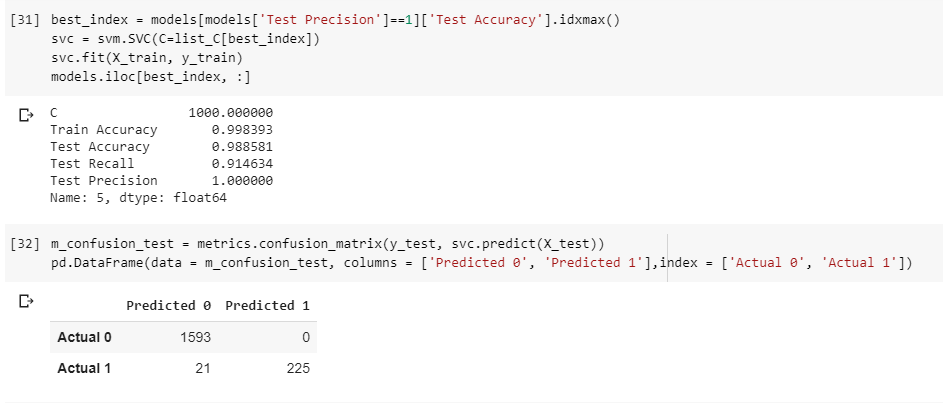




SVM MODEL SCREEN SHORT





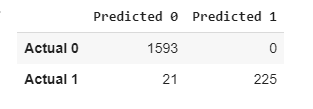
## **VII. CONCLUSION**

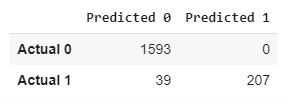
**The best model I have found is support vector machine with 98.8% accuracy.**

**It classifies every non-spam message correctly (Model precision)**

**It classifies the 91.4% of spam messages correctly (Model recall)**

**Confusion matrix of Support Vector Machine**



**Confusion matrix of Naïve Bayes**

**VIII. REFERENCES**

1. J. Han, M. Kamber, J. Pei, “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2011.
2. S. Murugan, R. Karthika, “A Literature Review on Text Mining Techniques and Methods”, International Journal of Comp
3. R. Lourdusamy, S. Abraham, "A Survey on Text Pre-processing Techniques and Tools", International Journal of Computer Sciences and Engineering, Vol.06, Issue.03, pp.148-157, 2018.
4. R. Lourdusamy, S. Abraham, "A Survey on Text Pre-processing Techniques and Tools", International Journal of Computer Sciences and Engineering, Vol.06, Issue.03, pp.148-157, 2018.
5. A. McCallum, R. Rosenfeld, T.M. Mitchell, and A.Y. Ng,”Improving Text Classification by Shrinkage in a Hierarchy of Classes”. In ICML, Vol. 98. 359–367, 1998.
6. C.D. Manning, P. Raghavan, and H. Schütze, “Introduction to information retrieval”,Vol. 1, Cambridge university press, Cambridge, 2008.
7. .H. Turtle and W.B. Croft,“Inference networks for document retrieval”, In Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, pp. 1–24, 1989
8. P.V. Arivoli, T.Chakravarthy, G.Kumaravelan, “International Journal of Advanced Research in Computer Science”, 8, (8), 299- 302, 2017.
9. F. Sebastiani. “Machine Learning in Automated Text Categorization”, ACM Computing Surveys, 34(1), 2002.
10. S. Z. Mishu, S. M. Rafiuddin, "Performance Analysis of Supervised Machine Learning Algorithms for Text Classification", 19th Int. Conf. Comput. Inf. Technol, pp. 409-413, 2016.
11. F. Colas ,P. Brazdil., “Artificial Intelligence in Theory and Practice”, ed. M. Bramer, (Boston: Springer), pp. 169-178, 2006.
12. <https://archive.ics.uci.edu/ml/datasets.php>