BRAC UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence
Section: 05

Group: 8

Project Title:

Email Spam Detection

Detects Spam emails by differentiating it from real emails using models like Naive Bayes, Support Vector Machine & Random Forrest Classifier.

	Name	ID
Submitted by:	Sadman Sharif	20101107
	Fardin Rahman	20101072
	Md. Seam Iltimas	20301036
	Md. Abu Bakar Siddique Khan	20301041
Submitted To:	Syed Zamil Hasan	Faculty
	Md. Nowroz Junaed Rahman	Faculty

Table of Contents:

1. Introduction

1.1. Background

Since the inception of email communication, the problem of email spam detection has persisted. Spam emails were first sent to numerous email addresses in the early 1990s, and they rapidly became a source of annoyance for email users. In response to the increasing volume of spam, email providers started to devise techniques to identify and remove unwanted messages. Initially, rule-based filtering systems were employed, which compared emails against predetermined criteria to determine if they were spam. However, spammers soon learned to circumvent these systems, prompting email providers to explore more advanced spam detection methods.

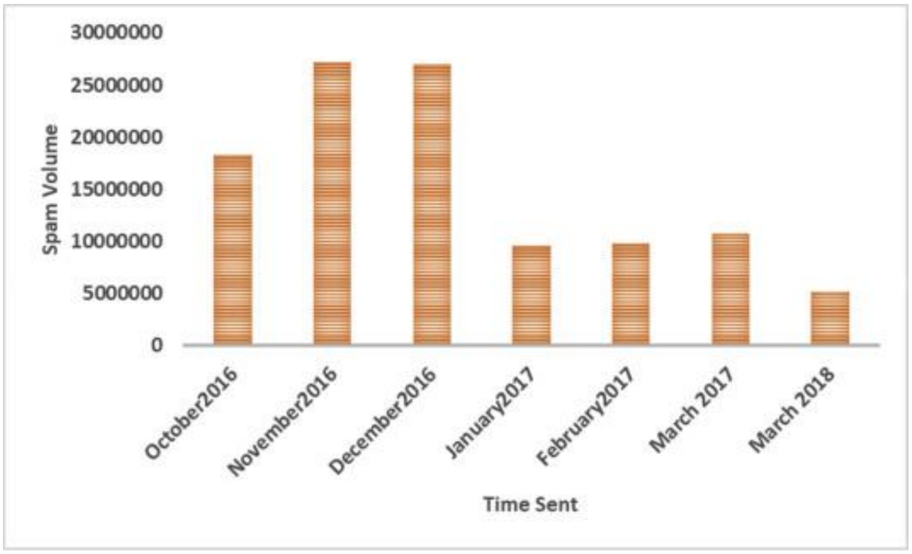


Figure: Spam volume vs Year

1.2. Statement of Problem

According to social networking experts, about 40% of social network accounts are utilized for spamming purposes. Spammers exploit popular social networking tools to focus on specific groups of people, review pages, or fan pages and include hidden links to pornographic or fraudulent sites through fake accounts. These spam emails tend to have similar characteristics and are sent to the same types of individuals or organizations.

1.3. Scope of Research

There is a vast and continually developing scope of research in using machine learning for detecting email spam. Machine learning algorithms have the potential to improve the accuracy of spam detection by recognizing patterns in large email datasets. Researchers have explored various machine learning techniques, such as supervised, unsupervised, and semi-supervised learning, to teach models how to differentiate between spam and legitimate emails. By examining the unique features of spam emails, one can improve their detection by using artificial intelligence (AI) to extract characteristics from the email's headers, subject, and body. These features are then used to categorize emails as either spam or legitimate. Learning-based classifiers are commonly used for spam detection because they assume that spam emails have certain traits that distinguish them from legitimate ones. However, there are several factors that make the identification process for spam in these models complicated.

2. Dataset Description

2.1. Source

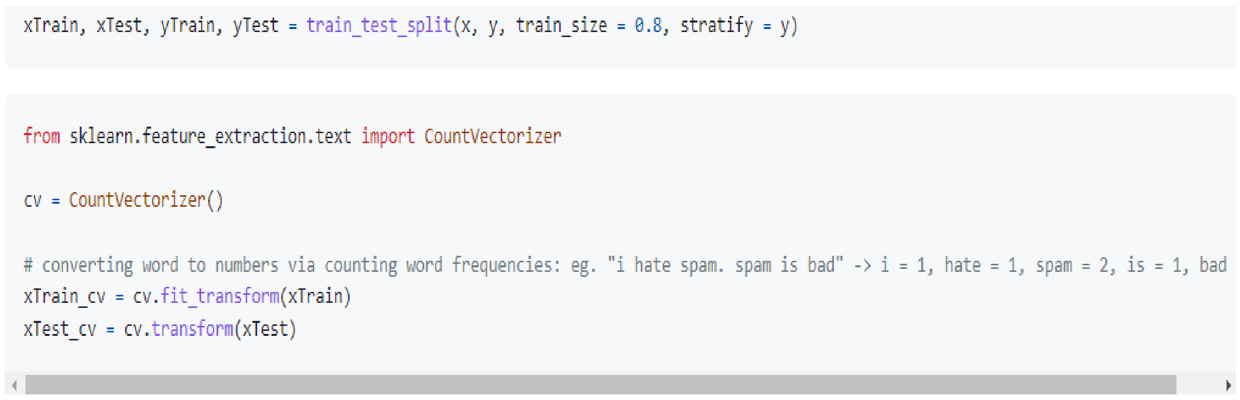
This dataset has been retrieved from Kaggle. Kaggle provides its users with access to datasets for use in developing AI models, as well as the ability to publish their own datasets and collaborate with other data scientists and machine learning engineers. Additionally, Kaggle offers the opportunity to participate in competitions aimed at solving data science challenges.

2.2. Link

https://www.kaggle.com/datasets/rockinjas123/spam-ham-emails

2.3. Total amount of Features

In email spam detection we are using a dataset that consists of 5000+ emails. In this project we have converted these emails in a sparse matrix. A sparse matrix is a matrix in which many or most of the elements have a value of zero. This is in contrast to a dense matrix, where many or most of the elements have a non-zero value. The column of this sparse matrix consists all the features. Here the features are basically the frequency of words used in each email. Based on the volume or frequency of each word it is decided that the email is spam or not.



```
xTrain, xTest, yTrain, yTest = train_test_split(x, y, train_size = 0.8, stratify = y)
```

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer()

# converting word to numbers via counting word frequencies: eg. "i hate spam. spam is bad" -> i = 1, hate = 1, spam = 2, is = 1, bad
xTrain_cv = cv.fit_transform(xTrain)
xTest_cv = cv.transform(xTest)
```

Figure: Converting mails to Sparse Matrix

2.4. Type of Problem

This is a classification type problem because our main goal here is to classify email as either spam or real. A classification problem in machine learning is one in which a class label is anticipated for a specific example of input data. Problems with categorization include spam email detection where it indicates that whether the email is spam or not.

2.5. Kind of Features

The dataset used for this problem involves categorical data, which is information that has been grouped into different categories. Categorical data can be grouped based on variables present in the dataset, such as whether an email is classified as spam or real. Although categorical data may be represented using numerical values (e.g., 1 for "Yes" and 0 for "No"), these numbers do not have any mathematical significance. As a result, they cannot be added or subtracted.

2.6. Correlation of Features

A correlation is a statistical measure that describes the connection between two variables. It is a fundamental aspect of data exploration and plays a pivotal role in many advanced machine learning techniques. In our given dataset, the frequency of individual words is the defining feature. However, no two features in our dataset are correlated with each other, as none of them surpasses the threshold value that determines the degree of correlation. This implies that there is no linear relationship between the features, and they are independent of one another.

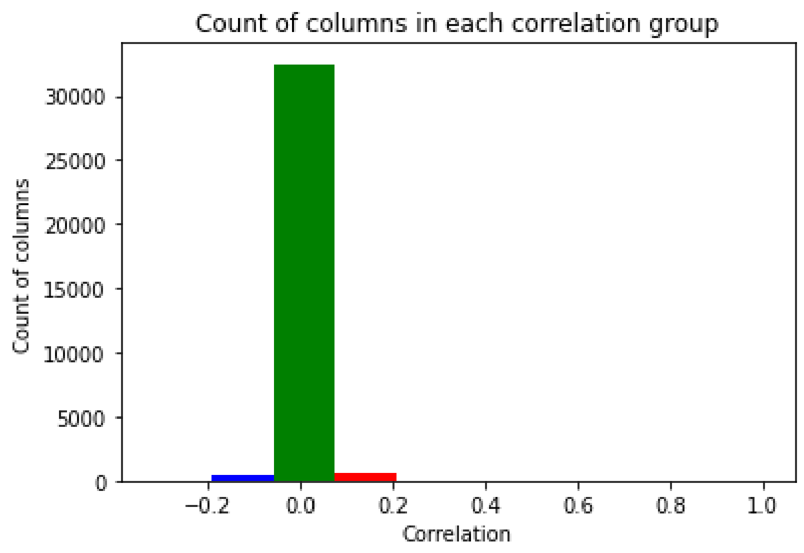


Figure: 0% Correlation with Frequency of word (x) and Email (y)

3. Dataset Pre-processing

3.1. Faults in the Dataset

The problems or faults with this dataset are mentioned below:

1. There were a lot of duplicate rows in this dataset which were making the process redundant.

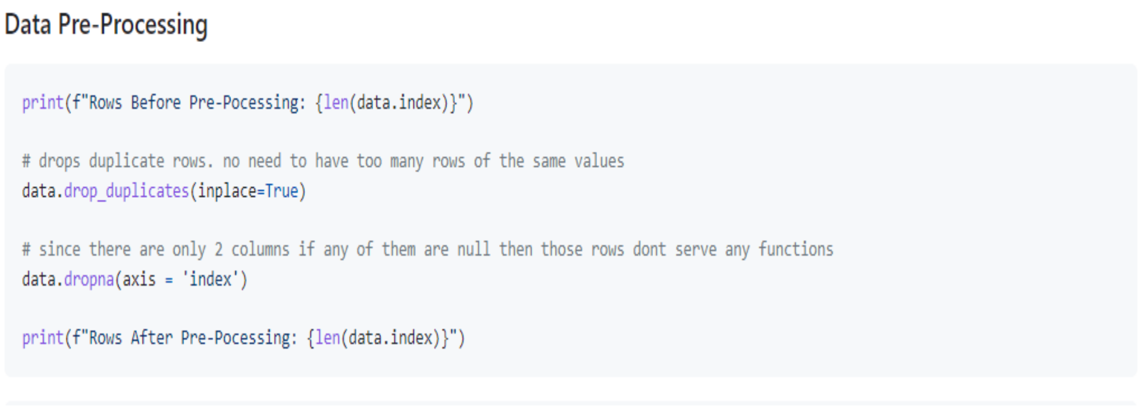
2. Multiple rows with null values. Rows with null values were of no use as it didn't play any role feature

3.2. Solution of the problems in dataset

Missing values can be handled by deleting the rows or columns having null values. If columns have more than half of the rows as null then the entire column can be dropped. The rows which are having one or more columns values as null can also be dropped. Solutions are mentioned below:

1. We dropped duplicate rows. No need to have too many rows of the same values

2. Since there are only 2 columns if any of them are null then those rows don't serve any functions



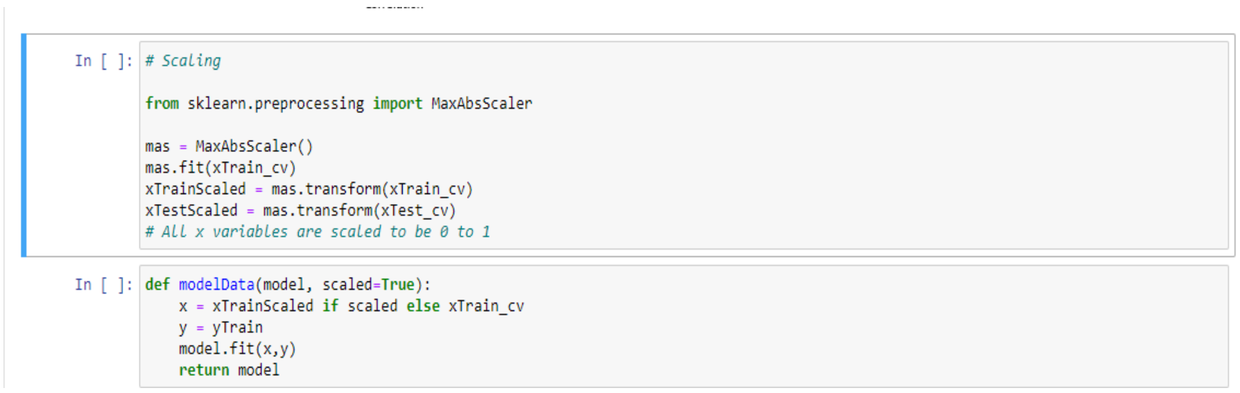
Data Pre-Processing

```
print(f"Rows Before Pre-Pocessing: {len(data.index)}")

# drops duplicate rows. no need to have too many rows of the same values
data.drop_duplicates(inplace=True)

# since there are only 2 columns if any of them are null then those rows dont serve any functions
data.dropna(axis = 'index')

print(f"Rows After Pre-Pocessing: {len(data.index)}")
```

Figure: Data Pre-Processing (To avoid Redundancy)

4. Feature Scaling

In feature scaling we have adjusted all the variables scaled to be 0 to 1. We have approached this step for the simplicity of calculation.



```
In [ ]: # Scaling

        from sklearn.preprocessing import MaxAbsScaler

        mas = MaxAbsScaler()
        mas.fit(xTrain_cv)
        xTrainScaled = mas.transform(xTrain_cv)
        xTestScaled = mas.transform(xTest_cv)
        # All x variables are scaled to be 0 to 1
```

```
In [ ]: def modelData(model, scaled=True):
            x = xTrainScaled if scaled else xTrain_cv
            y = yTrain
            model.fit(x,y)
            return model
```

5. Dataset Splitting

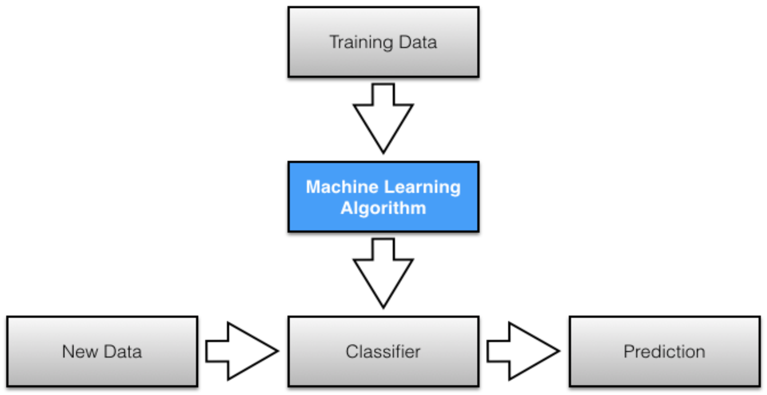
5.1. Train Set

we first divided the dataset into groups based on the target class variable. The division is then done in such a way that each group is represented in both the training and testing sets, with the same proportion as in the original dataset. This ensures that the model trained on the training set is able to learn the characteristics of all classes, including the less prevalent ones, and is able to generalize well to new data. For this project we have used 70% of our data for train set.

5.2. Test Set

we first divided the dataset into groups based on the target class variable. The division is then done in such a way that each group is represented in both the training and testing sets, with the same proportion as in the original dataset. This ensures that the model trained on the training set is able to learn the characteristics of all classes, including the less prevalent ones, and is able to generalize well to new data. For this project we have used 30% of our data for test set.

6. Model Training & Testing



6.1. Naive Bayes Model

The Naive Bayes algorithm, which is a straightforward yet robust classifier technique, is based on Bayes' Theorem Formula. It assumes that the predictors are independent of each other. According to Bayes' Theorem, when given a hypothesis A and evidence B, the relationship between the likelihood of hypothesis A before obtaining evidence P(A) and the likelihood of hypothesis A after obtaining evidence P(A|B) can be calculated.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Here:

- A, B = events
- P(A|B) = probability of A given B is true
- P(B|A) = probability of B given A is true
- P(A), P(B) = the independent probabilities of A and B

The model has been created by analyzing a large set of emails that have been previously labeled as spam or ham. The frequency of each word is counted to determine the probability of spam or ham. For instance, if there are 5000 randomly selected emails in the training set, with 1500 labeled as spam and 3500 as ham, the probability of spam (P(S)) is 0.3, while the probability of ham (P(H)) is 0.7. To calculate the probabilities of a particular word, say "tasty," occurring in spam or ham emails, we can count the number of emails with the word "tasty" in the spam and ham categories separately and divide it by the total number of emails in each category. For example, if there are 230 spam emails with the word "tasty" and 16 ham emails with the word "tasty," we can calculate the probabilities accordingly.

$$P(\text{“tasty”} \mid S) = 230/1500 = 15.33\%$$

$$P(\text{“tasty”} \mid H) = 16/3500 = 0.46\%$$

6.2. Support Vector Machine Model

SVM is a classifier that belongs to the Kernel Methods subcategory of Machine Learning. It is based on statistical learning theory. In SVM, the assumption is that linear categories can be separated into frames of acceptance, with a maximum margin hyperplane separating the categories. In cases where data cannot be linearly separated, it is mapped to a higher dimensional space so that it can be separated linearly in the new space. If two categories are linearly separable, then the SVM algorithm is used to create two bordering parallel plates and draw them away from each other to separate the data. The categories that are farthest from the boundary of the plates are the best separator, and the nearest training data to the hyperplane separator plates is called the "support vector." The proper use of the SVM algorithm lies in its ability to generalize, because even with large dimensions, overfitting can be avoided due to the optimization algorithm used in data compression.

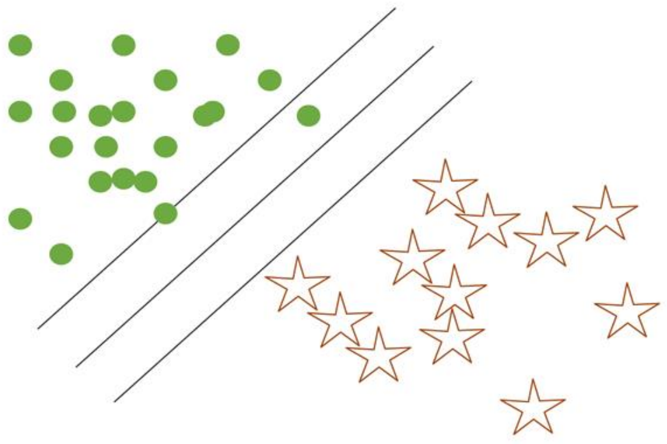


Figure: Support Vector Machine Model

The given image shows circles and stars which are referred to as objects that can be assigned to two different classes - stars or dots. The decision boundary is defined by isolated lines that separate the green and brown objects. The objects present in the lower part of the plane are brown stars, while the upper part contains green dots, indicating that the two types of objects are classified into different classes. When a new object, represented by a black circle, is presented to the model, it uses the training examples provided during the training phase to classify it into one of the two classes.

6.3. Random Forest Model

The random forest algorithm is a supervised classification technique that constructs a forest of decision trees. The more trees in the forest, the more robust it is likely to be. The decision tree is used as the base model, and the forest is made up of a number of these trees. The random forest classifier can handle unbalanced and missing data, and has relatively fast runtimes. However, it has some limitations, including its inability to predict beyond the range of the training data for regression tasks and the tendency to over-fit noisy datasets. The algorithm works by growing multiple classification trees, each of which is grown independently using a random subset of the input variables. The effectiveness of the algorithm can be determined by how well it performs on a given dataset. It works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

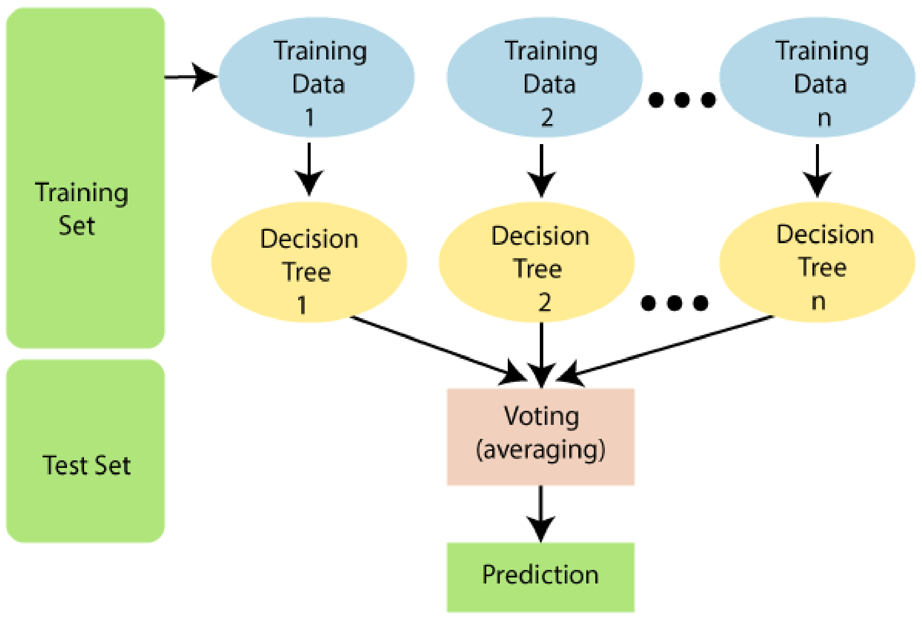


Figure: Random Forest Classifier steps

The steps of Random Forest Classifier are mentioned below:

Step-1: Select K data points randomly from the training set.
Step-2: Create decision trees using the selected data points.
Step-3: Decide the number of decision trees to build (N).
Step-4: Repeat Steps 1 and 2 to create N decision trees
Step-5: When given new data points, predict their category using each decision tree, and assign them to the category that has the most votes.

7. Model Selection/ Comparison Analysis

7.1. Bar chart Showcasing of all Models

The Bars charts for all the Models are mentioned below:

1.

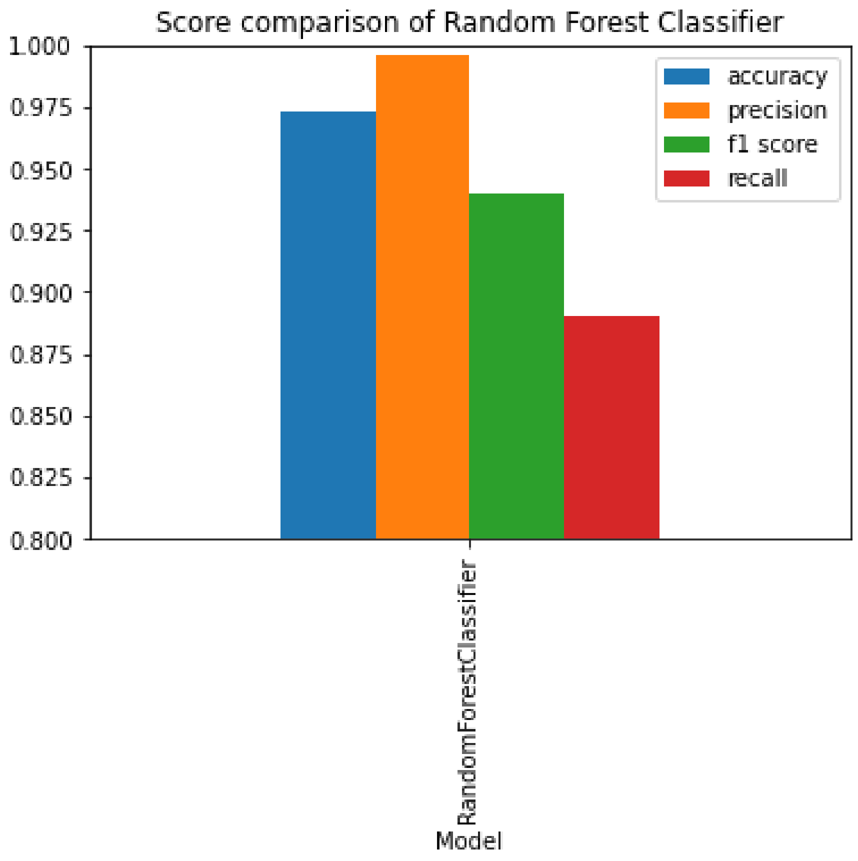


Figure: Bar chart (Random Forest Classifier)

2.

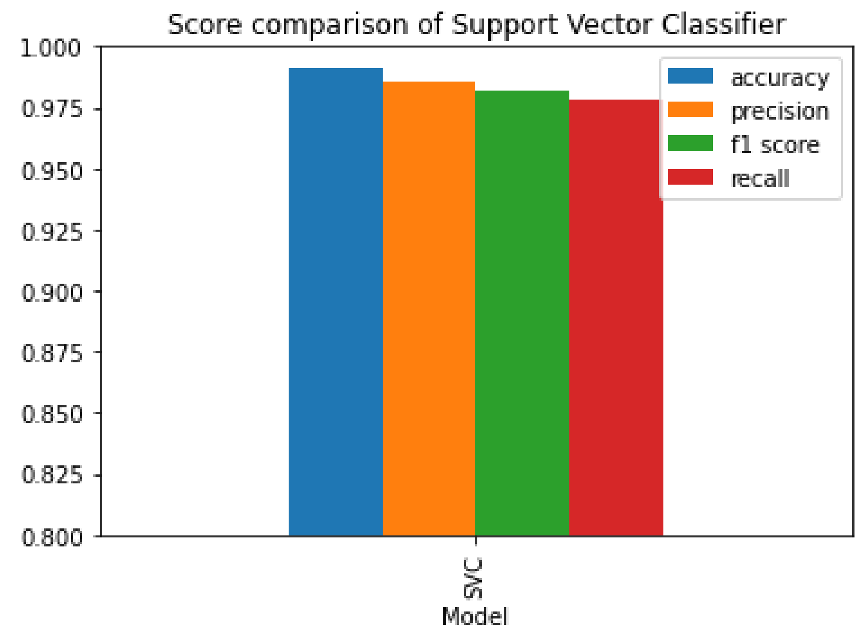


Figure: Bar chart (Support Vector Classifier)

3.

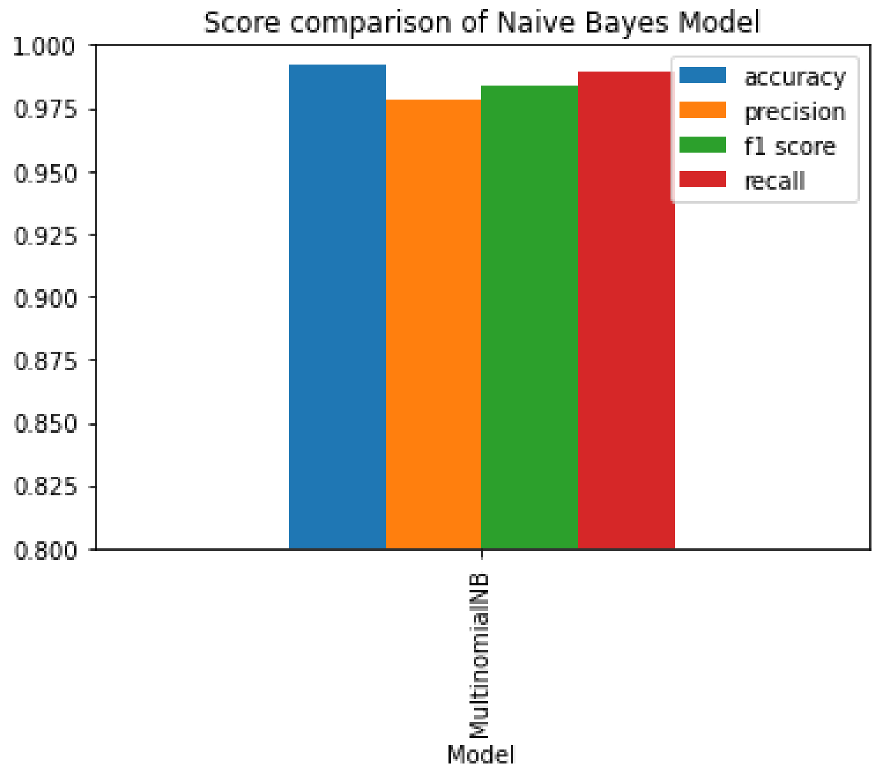


Figure: Bar chart (Naive Bayes Model)

7.2. Prediction Accuracy of all Models: $\frac{\text{True Positive+True Negative}}{\text{True positive+True Negative+False Positive+False Negative}}$

Models	Accuracy
Naive Bayes Model	98.8 %
Support Vector Machine	98.7 %
Random Forest Classifier	96.4 %

7.3. Recall Comparison of all Models: $\frac{\text{True Positive}}{\text{True positive+False Negative}}$

Models	Recall
Naive Bayes Model	96.7 %
Support Vector Machine	96.7 %
Random Forest Classifier	85.4 %

7.4. Precision Comparison of all Models: $\frac{\text{True Positive}}{\text{True positive+False Positive}}$

Models	Precision
Naive Bayes Model	98.5 %
Support Vector Machine	98.1 %
Random Forest Classifier	100 %

7.5. Confusion Matrix of all Models

1.

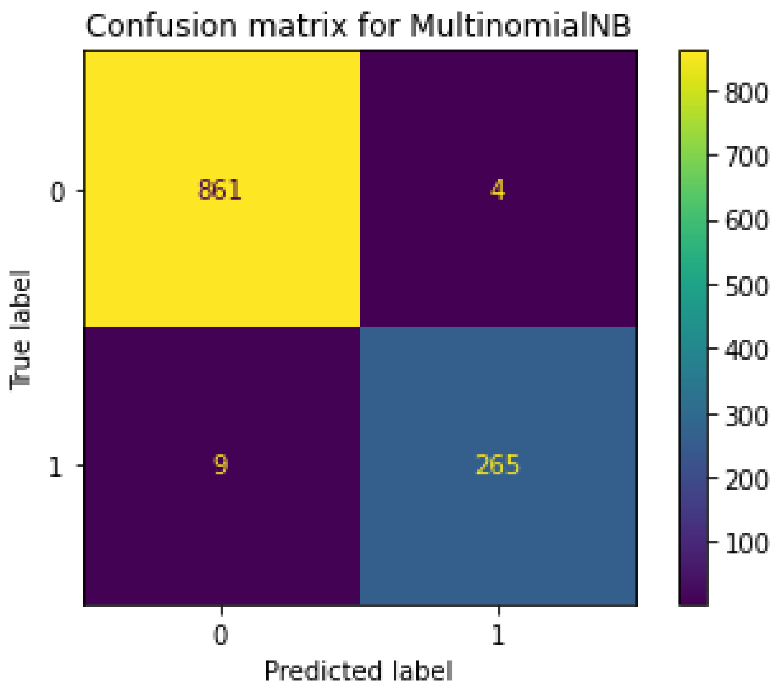


Figure: Confusion Matrix (Naive Bayes Model)

2.

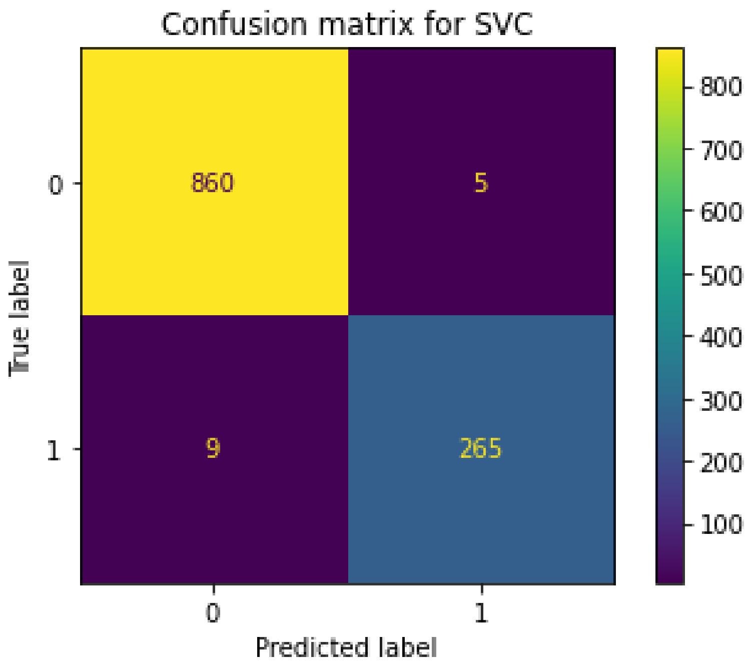


Figure: Confusion Matrix (Support Vector Machine)

3.

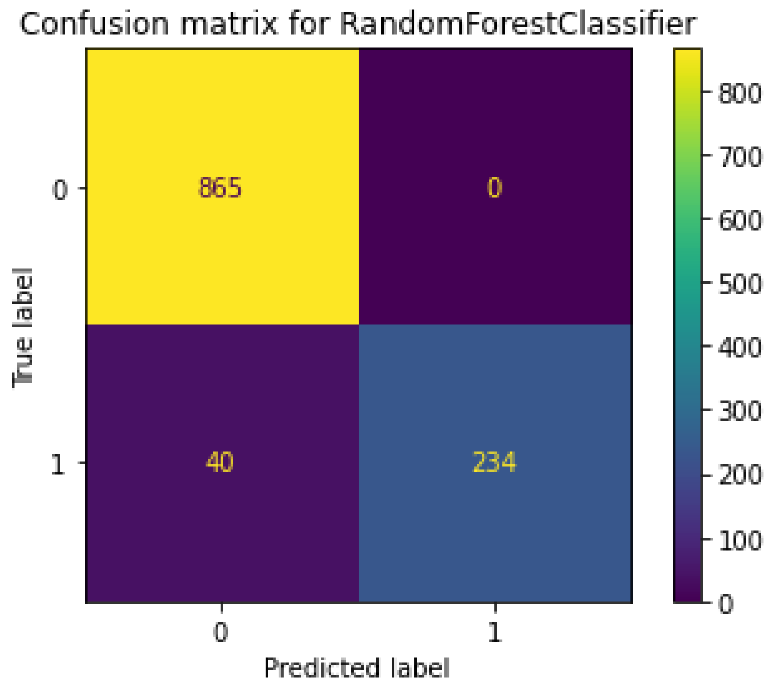


Figure: Confusion Matrix (Random Forest Classifier)

8. Conclusion

8.1. Major Finding.

In this study, we focus on training machine learning algorithms for filtering out spam mails. Throughout the process, we have used three models, e.g., Naive Bayes Model, Random Forest Model and SVM model. We have come to the conclusion that the Naive Bayes Model comparatively performs better than other two models. Email spam detection method helps us to detect the spam messages and therefore prevents them from creeping into the user inbox. Hence, it improves the quality of user experience. With this study, we can carry out the research for further development to increase the effectiveness of spam filters. Our project shows the necessity of email spam detection where the challenges as well as the progress in this field has also been discussed elaborately. To conclude, qualitative research in the field of spam filtering using machine learning, deep learning has been a need for providing better experience for the users.

8.2. References

- https://www.hindawi.com/journals/scn/2022/1862888/
- https://www.javatpoint.com/machine-learning-random-forest-algorithm
- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
- https://www.geeksforgeeks.org/support-vector-machine-algorithm/
- https://www.springboard.com/blog/data-science/bayes-spam-filter/
- https://courses.cs.washington.edu/courses/cse312/18sp/lectures/naive-bayes/naivebayesnotes.pdf
- https://medium.com/analytics-vidhya/email-spam-classifier-using-naive-bayes-a51b8c6290d4
- https://www.sciencedirect.com/science/article/pii/S2405844018353404