A PROJECT REPORT

on

"Email Spam Detection"

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN

COMPUTER SCIENCE & ENGINEERING

BY

AVIKSHITH SUBUDHI 2128015
SANJANA MOHANTY 2128047

UNDER THE GUIDANCE OF

Dr. Siddharth Swarup Rautaray
(Associate Professor)

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

November 2023

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

“EMAIL SPAM DETECTION“

submitted by

AVIKSHITH SUBUDHI 2128015
SANJANA MOHANTY 2128047

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2023, under our guidance.

Date: 18/11/2023

Dr. Siddharth Swarup Rautaray
(Associate Professor)
Project Guide

Acknowledgement

We are profoundly grateful to **Dr. Siddharth Swarup Rautaray** of **School of Computer Engineering** at **KIIT University, Bhubaneswar** for his expert guidance and continuous encouragement throughout to see that this project meets its target since its commencement to its completion. We are grateful to him for his guidance, constructive feedback, and constant support throughout the project.

AVIKSHITH SUBUDHI

SANJANA MOHANTY

ABSTRACT

Spam mails, also known as junk mail or unsolicited commercial email (UCE), are emails sent in bulk to a large number of individuals. They're frequently marketable in nature, promoting products or services, but they can also be used to spread malware or phishing swindles.

Spam emails are a major problem for both individuals and businesses. They can clutter up inboxes, waste time, and even pose a security risk. In addition, they can damage the reputation of businesses that are associated with spam.

This project report presents an analysis of email spam detection that aims to predict the likelihood of spam and non-spam mails. These models were developed using various machine learning algorithms and trained on a large dataset. The report provides a detailed overview of the models, including its design, implementation, and evaluation. The results indicate how accurately different algorithms detect spam mails.

Keywords- Machine Learning**,** SVM, Random Forest, Naive Bayes.

CONTENTS

1. INTRODUCTION

The practice of "using email to send unsolicited emails or advertising emails to a group of recipients" is known as email spam or electronic mail spam. When an email is unsolicited, it indicates that the receiver hasn’t granted permission to receive it. Over the past few years, the popularity of spam mail has increased. Spam takes up storage space, time, and message speed. Automatic Email filtering could be used to detect spam but it can be bypassed easily. Manually banning certain email addresses can also prevent spam mail.

Machine learning algorithms are used to detect spam mail. They are more efficient and use a set of training data where emails are pre-classified. Machine learning has various algorithms that can be used for email filtering.

The purpose of this report is to present an analysis of various machine-learning algorithms used in email spam detection. The goal of our study is to develop an understanding of machine learning models that can detect spam. We begin by providing an overview of the problem of spam emails.

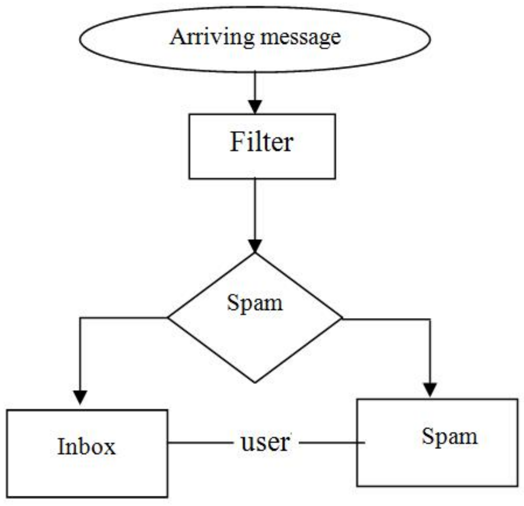


Fig 1.1 How spam filtering works

1.1 Basic Concepts Used

1. *Machine learning*:

Machine learning is a subfield of artificial intelligence (AI) that involves the development of algorithms and statistical models that allow computer systems to learn and improve from data without being explicitly programmed.

1.1) SVM

1.2) Random Forest

1.3) Naive Bayes

2. *Confusion Matrix*:

A confusion matrix is a table used to evaluate the performance of a classification model. It summarizes the number of correct and incorrect predictions made by the model on a set of test data and breaks down these predictions into different categories based on the true and predicted class labels. The four categories in a binary classification problem are true positives, true negatives, false positives, and false negatives.

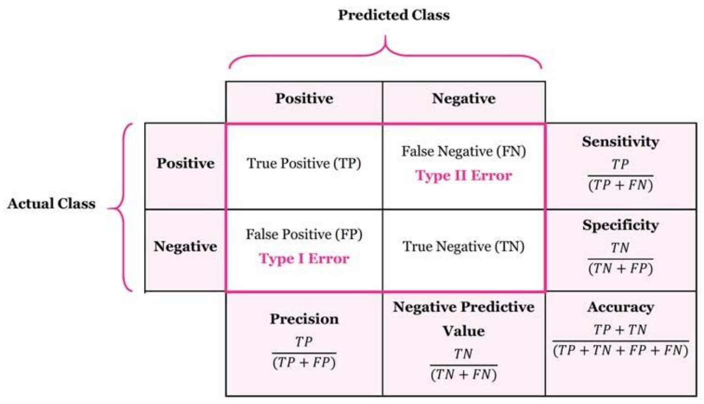
2.1) Precision

2.2) Recall

2.3) Specificity

2.4) Sensitivity

2.5) F1 Score



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Fig 1.2 Confusion matrix

1.2 Problem Statement

Spams can be a nuisance and a security threat as they may lead to the spreading of viruses, malware, and phishing scams. Spam filters help in filtering unwanted and dangerous virus-infected emails to get into email inboxes. However, different machine-learning algorithms can be used to classify spam emails.

1.3 Requirements

OS: Windows, Linux or MacOS

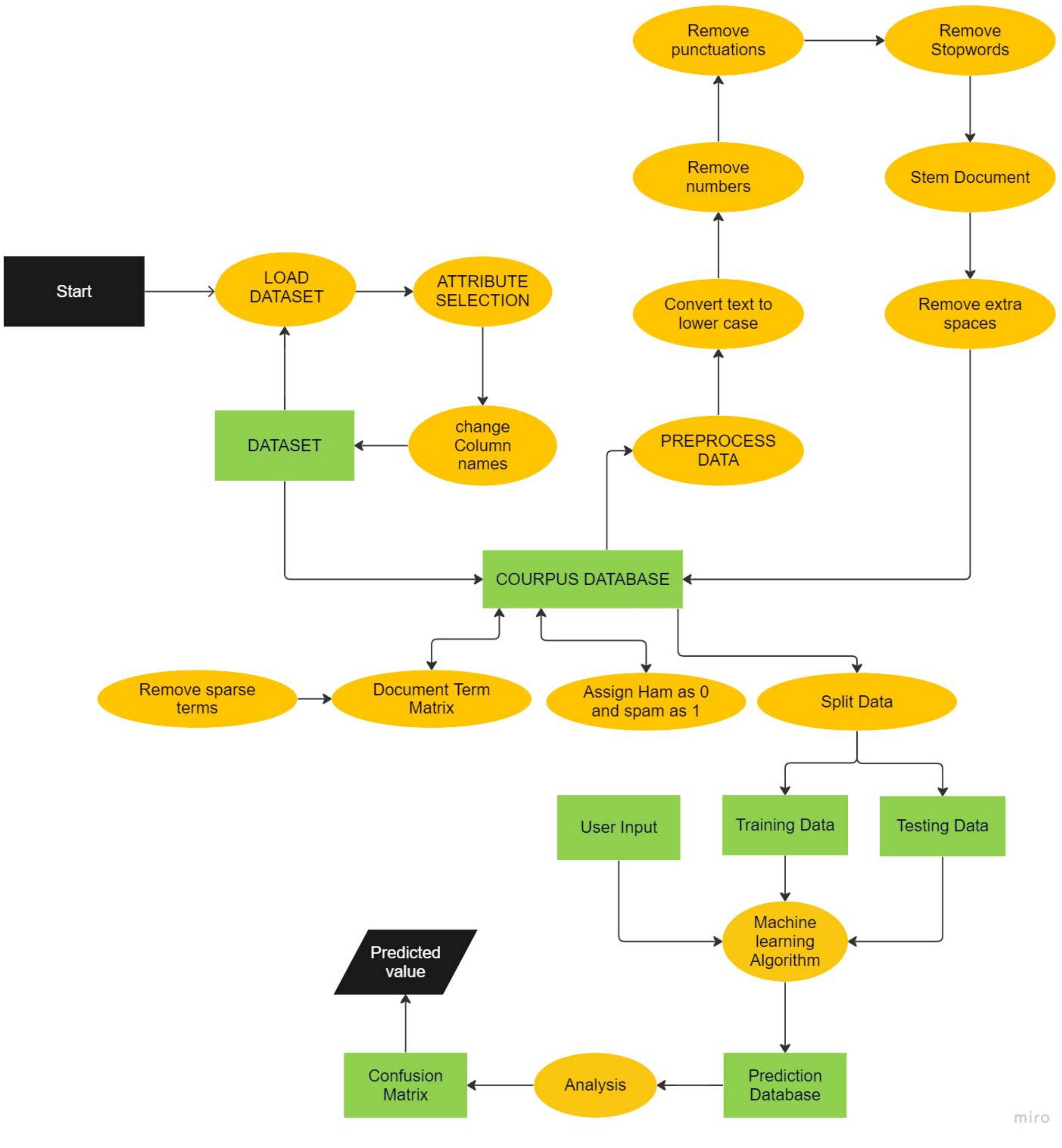
Language: R

IDE: R Studio

Libraries: tm, snowballC, ggplot2, randomForest, e1071, caret

- tm: The tm package is often used to create a corpus object. This package can be used to read data in many different formats– including text within data frames, .txt files, or .doc files.
- snowballC: An R interface to the C 'lib stemmer' library that implements Porter's word stemming algorithm for collapsing words to a common root to aid comparison of vocabulary.
- ggplot2: ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics.
- randomForest: Random Forest in R Programming is an ensemble of decision trees. It builds and combines multiple decision trees to get more accurate predictions. It's a non-linear classification algorithm.
- e1071: Offers quick and easy implementation of SVMs. Provides most common kernels, including linear, polynomial, RBF, and sigmoid.
- caret: Caret stands for classification and regression training and is arguably the biggest project in R. This package is sufficient to solve almost any classification or regression machine learning problem.

2. BLOCK DIAGRAM



3. IMPLEMENTATION

3.1 METHODOLOGY

3.1.1 DATA COLLECTION

- The first step for a prediction system is data collection and deciding about the training and testing dataset. In this project, we have used 75% of the dataset for training and 25% of the dataset for testing.
- The spam dataset was taken from the dataset repository of the Center for Machine Learning and Intelligent Systems at the University of California, Irvine.

3.1.2 ATTRIBUTE SELECTION

- In a dataset, attributes are the variables or features that are measured or observed for each instance or data point. They can be thought of as columns in a table, where each column represents a different aspect of the data.

SL.NO	ATTRIBUTE	DETAILS
1	V1	Type of email (spam or ham).
2	V2	Contains the text contents.

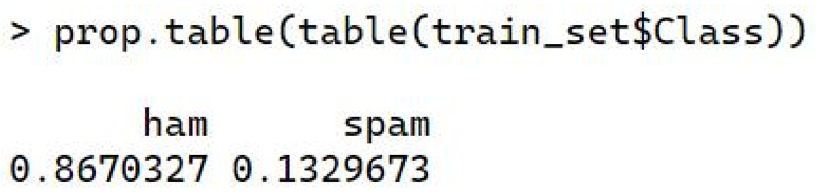
Table 3.1 Table depicts the names and details of all the attributes present in the spam dataset

3.1.3 NLP PRE-PROCESSING

- Perform tokenization, lemmatization, segmentation, and POS tagging to the content attributes of the dataset (v2 attribute). Store data in the corpus.
- Then it is to Create a document term matrix of the newly pre-processed data stored in the corpus data variable.

3.1.4 SPLITTING DATASET

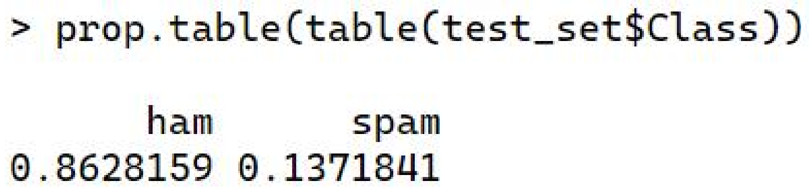
- Splitting the dataset into
    - Training data: 75%



```
> prop.table(table(train_set$Class))

      ham      spam
0.8670327 0.1329673
```

    - Testing data: 25%



```
> prop.table(table(test_set$Class))

      ham      spam
0.8628159 0.1371841
```

3.1.5 VISUALIZING FREQUENCIES OF WORDS > 100

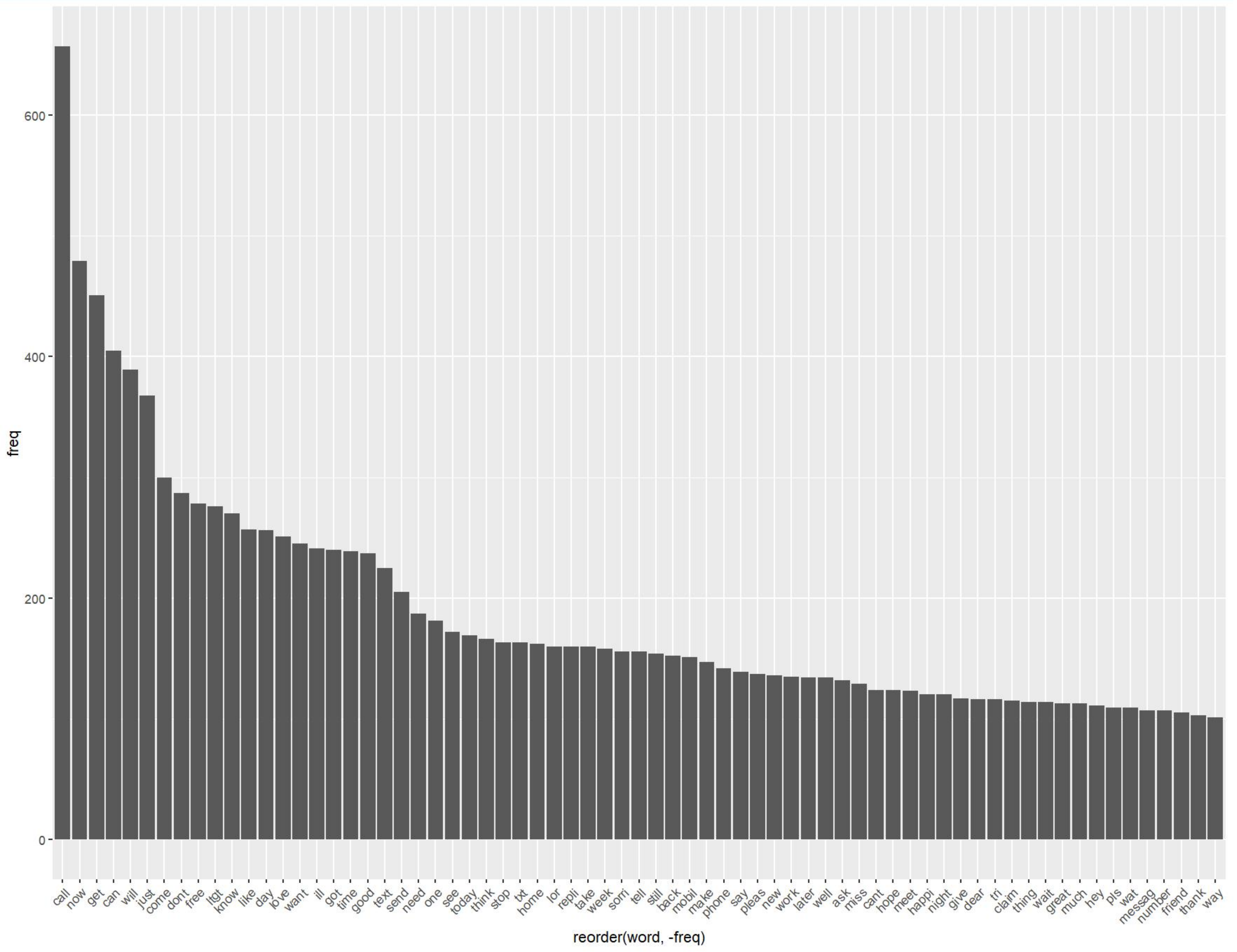
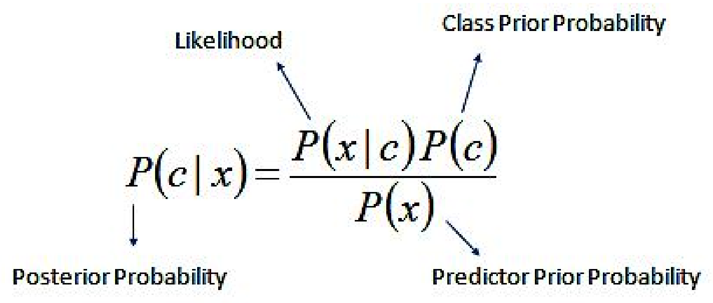


Fig 3.1 Frequency of words.

3.1.6 MACHINE LEARNING ALGORITHMS

- Machine learning algorithms are those algorithms that help in the prediction of specific data computationally.
- NAIVE BAYES
  - Naive Bayes is a simple and efficient probabilistic classifier based on Bayes' theorem. It is a popular choice for many classification tasks, including spam filtering, document classification, and sentiment analysis.



$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

- SVM
  - Support vector machines (SVMs) are versatile supervised learning algorithms used for classification and regression. They are particularly well-suited for binary classification problems, which involve classifying data points into two categories. SVMs work by finding the optimal hyperplane, which is a decision boundary that separates the two classes with the maximum margin. This margin is the distance between the hyperplane and the nearest data points of each class.

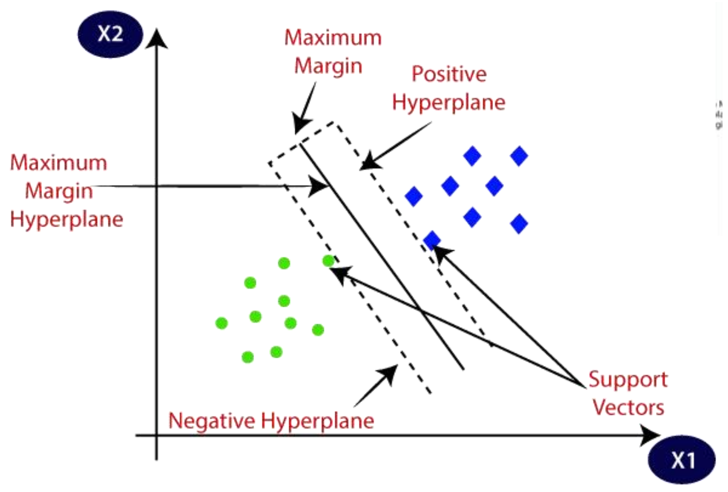


Fig 3.2 Suport Vector Machine

- RANDOM FOREST
  - Random forest is a popular machine-learning algorithm for both classification and regression tasks. It is an ensemble learning method, meaning that it combines the predictions of multiple decision trees to make a final prediction. This makes random forests less susceptible to overfitting than individual decision trees, and they often achieve higher accuracy.

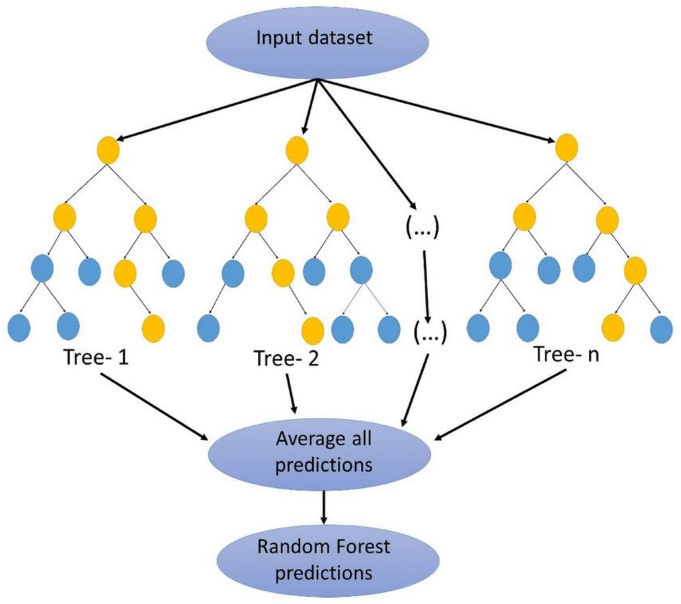
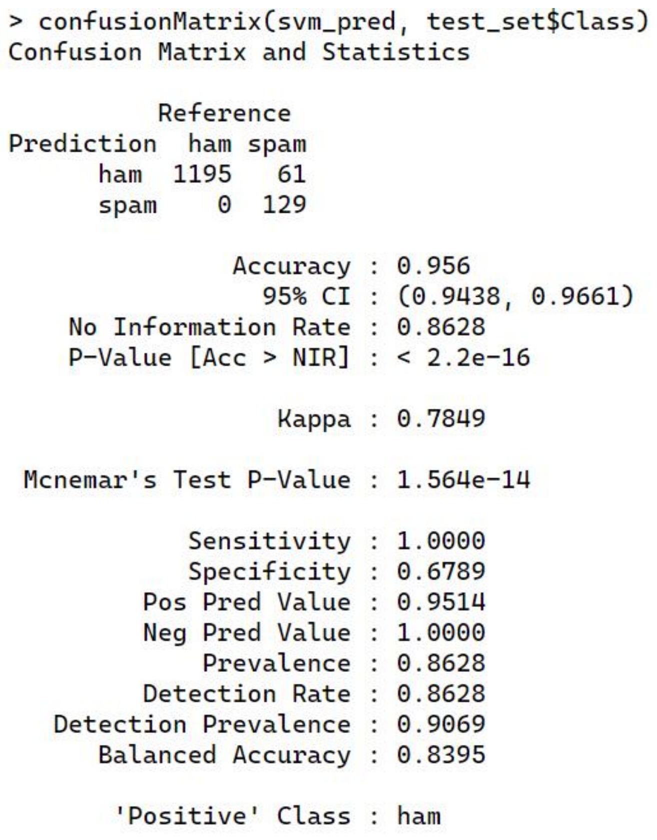


Fig 3.3 Random Forest

3.2 ANALYSIS

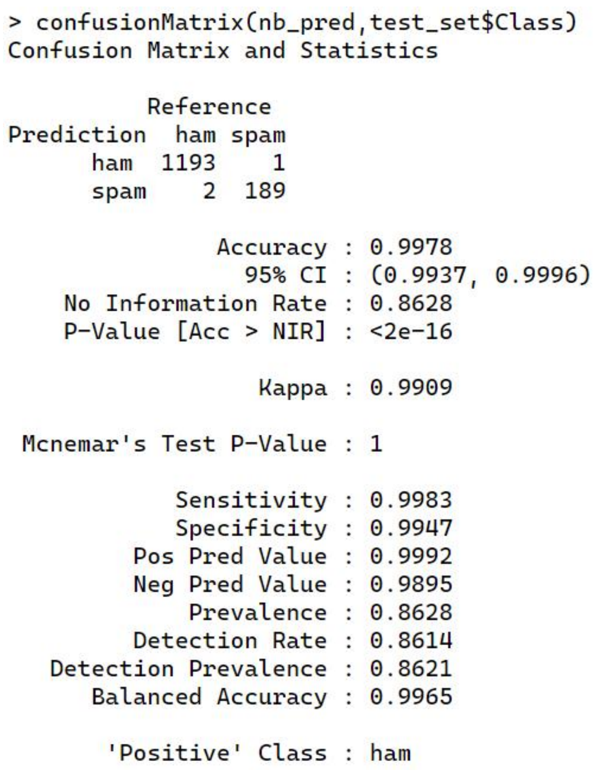
- 3.2.1 SVM (Support vector machine)
  - Confusion Matrix :



```
> confusionMatrix(svm_pred, test_set$Class)
Confusion Matrix and Statistics

          Reference
Prediction  ham spam
      ham  1195   61
      spam    0  129
                                          
               Accuracy : 0.956           
                 95% CI : (0.9438, 0.9661)
    No Information Rate : 0.8628          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.7849          
                                          
 Mcnemar's Test P-Value : 1.564e-14       
                                          
            Sensitivity : 1.0000          
            Specificity : 0.6789          
         Pos Pred Value : 0.9514          
         Neg Pred Value : 1.0000          
             Prevalence : 0.8628          
         Detection Rate : 0.8628          
   Detection Prevalence : 0.9069          
      Balanced Accuracy : 0.8395          
                                          
       'Positive' Class : ham             
```

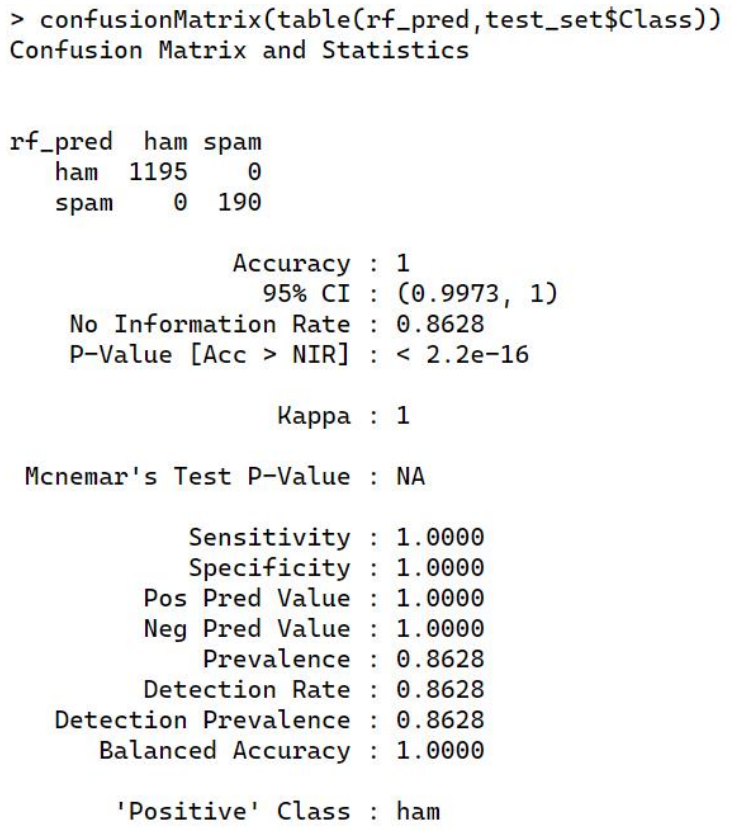
- 3.2.2 Random Forest
  - Confusion Matrix :



```
> confusionMatrix(nb_pred,test_set$Class)
Confusion Matrix and Statistics

          Reference
Prediction  ham spam
      ham  1193    1
      spam    2  189
                                          
               Accuracy : 0.9978          
                 95% CI : (0.9937, 0.9996)
    No Information Rate : 0.8628          
    P-Value [Acc > NIR] : <2e-16          
                                          
                  Kappa : 0.9909          
                                          
 Mcnemar's Test P-Value : 1               
                                          
            Sensitivity : 0.9983          
            Specificity : 0.9947          
         Pos Pred Value : 0.9992          
         Neg Pred Value : 0.9895          
             Prevalence : 0.8628          
         Detection Rate : 0.8614          
   Detection Prevalence : 0.8621          
      Balanced Accuracy : 0.9965          
                                          
       'Positive' Class : ham             
```

- 3.2.3 Naive Bayes Classifier
    - Confusion Matrix:



```
> confusionMatrix(table(rf_pred,test_set$Class))
Confusion Matrix and Statistics


rf_pred  ham spam
   ham  1195    0
   spam    0  190

               Accuracy : 1
                 95% CI : (0.9973, 1)
    No Information Rate : 0.8628
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.8628
         Detection Rate : 0.8628
   Detection Prevalence : 0.8628
      Balanced Accuracy : 1.0000

       'Positive' Class : ham
```

4. CONCLUSION

Features	**SVM**	**NAIVE BAYES**	**RANDOM FOREST**
Types of algorithm	Supervised learning	Probabilistic classifier	Ensemble learning
Principle	Maximizing margin between two classes	Bayes Theorem	Combining predictions of multiple decision trees
Advantages	High accuracy, robust to outliers, can handle high-dimensional data	Simple to handle and implement, efficient, robust to noise.	High accuracy, less susceptible to overfitting, can handle high-dimensional data.
Disadvantages	Can be computationally expensive, Sensitive to outliers, May not be suitable for all types of data	Naive independence assumption can be violated, Sensitive to noise	Can be difficult to interpret, Can be computationally expensive
Application	Image classification, Text classification, Spam filtering	Spam filtering, Document classification, Sentiment analysis	Classification, Regression, Feature selection

Table 4.1 Difference between SVM, Naive bayes and Random Forest

MACHINE LEARNING ALGORITHM PERFORMANCE ANALYSIS:

- **Random forest**
  - Random forest showed exceptional performance with 100% accuracy. But it may lead to overfitting.
- **Naive Bayes**
  - Naive Bayes model achieved 99.49% accuracy. It has 100% sensitivity.
- **Support vector machine**
  - SVM did not perform efficiently as it achieved 86.28% accuracy.

Since Naive Bayes Algorithm gave good results and incorporated and adapted to any type of database , we chose this algorithm to predict any user input Data.
After Incorporating User input data we were able to see significantly accurate results.

5. REFERENCES

- https://rpubs.com/redwanwalid/spam-detection#:~:text=The%20dataset%20contains%20the%20count,the%20predictors%20and%20the%20response.
- https://rpubs.com/aquaregis32/spamEmailClassification_byAr32
- https://applied-math-coding.medium.com/data-science-creating-an-email-spam-filter-by-using-bag-of-words-bfb42078e002
- https://encyclopedia.pub/entry/50115
- https://kharshit.github.io/blog/2017/08/25/email-spam-filtering-text-analysis-in-r
- https://towardsdatascience.com/spam-detection-in-emails-de0398ea3b48
- https://www.sciencedirect.com/science/article/pii/S1877050921007493
- https://www.researchgate.net/publication/342113653_Email_based_Spam_Detection
- https://search.r-project.org/CRAN/refmans/kernlab/html/spam.html
- https://github.com/norberte/Spam-detection
- https://github.com/topics/spam-detection?o=asc&s=stars
- https://www.r-bloggers.com/2014/09/build-a-spam-filter-with-r/
- https://www.pnrjournal.com/index.php/home/article/download/8948/12217/10743
- https://www.pnrjournal.com/index.php/home/article/download/8948/12217/10743

Image links :

- https://www.google.com/url?sa=i&url=https%3A%2F%2Fmanisha-sirsat.blogspot.com%2F2019%2F04%2Fconfusion-matrix.html&psig=AOvVaw0rn0oC-1-PDKtTg1vZieIf&ust=1700280964854000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLjzj7WWyoIDFQAAAAAdAAAAABAD
- https://www.google.com/imgres?imgurl=https%3A%2F%2Fuc-r.github.io%2Fpublic%2Fimages%2Fanalytics%2Fnaive_bayes%2Fnaive_bayes_icon.png&tbnid=i_i1ubQtQbnK1M&vet=12ahUKEwiAh8fXlsqCAxW3cGwGHf6QDdYQMygKegUIARCBAQ..i&imgrefurl=https%3A%2F%2Fuc-r.github.io%2Fnaive_bayes&docid=mPNZTJwY-4anIM&w=439&h=241&q=naive%20bayes&ved=2ahUKEwiAh8fXlsqCAxW3cGwGHf6QDdYQMygKegUIARCBAQ
- https://www.google.com/imgres?imgurl=https%3A%2F%2Fwww.researchgate.net%2Fpublication%2F355828449%2Ffigure%2Ffig3%2FAS%3A1085564153528357

%401635830071846%2FSchematic-diagram-of-the-random-forest-algorithm.png&tbnid=S4eUuGiM1H9izM&vet=12ahUKEwjP9_TklsqCAxVTSWwGHb5qD70QMygjegUIARC4AQ..i&imgrefurl=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FSchematic-diagram-of-the-random-forest-algorithm_fig3_355828449&docid=rRqDCMbBaMSBFM&w=850&h=750&q=Random%20Forest%20&ved=2ahUKEwjP9_TklsqCAxVTSWwGHb5qD70QMygjegUIARC4AQ

- https://www.google.com/imgres?imgurl=https%3A%2F%2Fstatic.javatpoint.com%2Ftutorial%2Fmachine-learning%2Fimages%2Fsupport-vector-machine-algorithm.png&tbnid=pCj4qS6NoIDTjM&vet=12ahUKEwjV4sj3lsqCAxUjUWwGHRK-DbAQMygCegQIARBx..i&imgrefurl=https%3A%2F%2Fwww.javatpoint.com%2Fmachine-learning-support-vector-machine-algorithm&docid=L3B6Vp_1rkllmM&w=600&h=400&itg=1&q=SVM&ved=2ahUKEwjV4sj3lsqCAxUjUWwGHRK-DbAQMygCegQIARB
- https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FBlock-Diagram-of-Spam-Filter_fig3_305001941&psig=AOvVaw0K9Gc3ipPrvKLkf-QLAUIf&ust=1700281913653000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCKj7rPqZyoIDFQAAAAAdAAAAABAD

DATASET link :

- https://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspamcollection.zip

6. APPENDIX (CODE)

```
# Import necessary libraries
library(tm)
library(SnowballC)
library(e1071)
library(caret)
library(ggplot2)
library(randomForest)



## DATASET (LOAD AND MODIFICATION)

# Read the spam collection dataset
data_text <- read.delim("SpamCollection", sep="\t", header=F, colClasses="character", quote="")
# Examine the data structure
str(data_text)
# Display the first few rows of the data
head(data_text)
# Extract the column names
colnames(data_text)
# Rename the columns
colnames(data_text) <- c("Class", "Text")
# Convert the "Class" column to a factor with two levels
data_text$Class <- factor(data_text$Class)
# Display the proportion of each class
prop.table(table(data_text$Class))



## TEXT CLEANING

# Clean the texts
corpus = VCorpus(VectorSource(data_text$Text))
# Convert all text to lowercase
corpus = tm_map(corpus, content_transformer(tolower))
# Remove numbers from the text
corpus = tm_map(corpus, removeNumbers)
# Remove punctuation from the text
corpus = tm_map(corpus, removePunctuation)
# Remove stop words from the text
corpus = tm_map(corpus, removeWords, stopwords("english"))
# Stem the words in the text
corpus = tm_map(corpus, stemDocument)
# Strip whitespace from the text
corpus = tm_map(corpus, stripWhitespace)
```

```
## BAG OF WORDS

# Create a document-term matrix
dtm = DocumentTermMatrix(corpus)
# Remove sparse terms from the document-term matrix
dtm = removeSparseTerms(dtm, 0.999)
# Display the dimensions of the document-term matrix
dim(dtm)
# Inspect a portion of the document-term matrix
inspect(dtm[40:50, 10:15])
# Define a function to convert counts to binary values
convert_count <- function(x) {
  ifelse(x > 0, 1, 0)
}
# Apply the convert_count function to get final training and testing DTMs
datasetNB <- apply(dtm, 2, convert_count)
# Convert the document-term matrix to a data frame
dataset = as.data.frame(as.matrix(datasetNB))



##  VISUALIZE FREQUENCY OF WORDS

# Identify frequently occurring terms
freq <- sort(colSums(as.matrix(dtm)), decreasing = TRUE)
tail(freq, 10)
# Find frequently appearing terms
findFreqTerms(dtm, lowfreq = 60)
# Plot word frequencies
wf <- data.frame(word = names(freq), freq = freq)
head(wf)
pp <- ggplot(subset(wf, freq > 100), aes(x = reorder(word, -freq), y = freq)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
# Display the plot
pp



##  SPLIT DATA

# Add the Class label back to the dataset
dataset$Class = data_text$Class
str(dataset$Class)
# Set the seed for reproducibility
set.seed(222)
# Split the data into training and testing sets
split <- sample(2, nrow(dataset), prob = c(0.75, 0.25), replace = TRUE)
train_set <- dataset[split == 1,]
test_set <- dataset[split == 2,]
# Display the proportion of each class in the training set
```

```
prop.table(table(train_set$Class))
# Display the proportion of each class in the testing set
prop.table(table(test_set$Class))



##  RANDOM FOREST

# Train a Random forest classifier
rf_classifier = randomForest(x = train_set[-1210], y = train_set$Class, ntree = 300)
# Display the Random forest classifier summary
rf_classifier
# Predicting the Test set results
rf_pred = predict(rf_classifier, newdata = test_set[-1210])
# Evaluate the classifier performance using a confusion matrix
confusionMatrix(table(rf_pred,test_set$Class))



##  NAIVE BAYES CLASSIFIER

# Create control limit for NB model
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# Train a Naive Bayes classifier
system.time( classifier_nb <- naiveBayes(train_set, train_set$Class, laplace = 1, trControl = control,tuneLength = 7) )
# Make predictions on the testing set
nb_pred = predict(classifier_nb, type = 'class', newdata = test_set)
# Evaluate the classifier performance using a confusion matrix
confusionMatrix(nb_pred,test_set$Class)


## SVM

# Train a support vector machine classifier
svm_classifier <- svm(Class~., data = train_set)
# Display the classifier summary
svm_classifier
# Make predictions on the testing set
svm_pred <- predict(svm_classifier, test_set)
# Evaluate the classifier performance using a confusion matrix
confusionMatrix(svm_pred, test_set$Class)



##  NEW DATA FROM USER (PREDICTION USING NB)

# Get user input for the email to be tested
email_text <- readline(prompt = "Enter the email you want to test: ")
# Create a new data frame for the user input
new_test_set <- data.frame(Text = email_text)
# Clean the user input text
new_corpus <- VCorpus(VectorSource(new_test_set$Text))
new_corpus <- tm_map(new_corpus, content_transformer(tolower))
new_corpus <- tm_map(new_corpus, removeNumbers)
new_corpus <- tm_map(new_corpus, removePunctuation)
new_corpus <- tm_map(new_corpus, removeWords, stopwords("english"))
new_corpus <- tm_map(new_corpus, stemDocument)
new_corpus <- tm_map(new_corpus, stripWhitespace)
# Create a document-term matrix for the user input
new_dtm <- DocumentTermMatrix(new_corpus)
# Convert the new document-term matrix to a data frame
new_test_set <- as.data.frame(as.matrix(new_dtm))
# Make a prediction on the user input
nb_pred <- predict(classifier_nb, type = 'class', newdata = new_test_set)
# Determine and display the spam or not spam message
if (nb_pred == "ham") {
  print("This email is likely not spam.")
} else {
  print("This email is likely spam.")
}
```