SPAM DETECTION

Web Mining Final Project

Team Members: Rozerin Yıldız, Ayşe Oduncu

Student Ids: 180709040, 190709065

Abstract

Spam emails, also known as unsolicited bulk emails, refer to unwanted and often malicious or deceptive messages sent to a large number of recipients. These emails are typically sent without the explicit consent of the recipients and are often used for fraudulent purposes, such as phishing attempts, spreading malware, or promoting scam schemes. In our project, we are building models to classify emails as spam or not spam.

1. Introduction

In this project, we are processing dataset and building models to classify emails as spam or not spam.

GitHub Repository: https://github.com/Web-Mining-Final-Project/Web-Mining-Final-Project

2. Dataset

We used Spam or Not Spam Dataset to build a spam or not spam model that detects if an email is spam or not. The collection consists of '0030228_easy_ham.tar.bz2' and '20030228_spam.tar.bz2' taken from https://spamassassin.apache.org/old/publiccorpus/ i.e. *Apache SpamAssassin's public datasets*. There are 2500 ham and 500 spam emails in the dataset. All the numbers and URLs were converted to strings as NUMBER and URL respectively. This dataset is the simplified spam and ham dataset. We used the csv filed named spam_or_not_spam.csv file.

The dataset contains 2 columns named email and label; email containing emails and label containing 0s and 1s, 0s meaning email is not spam and 1s meaning email is spam.

3. Data Preprocessing

For preprocessing, we preprocessed and analyzed email data by filtering stopwords, generating word clouds for ham and spam emails, and displaying the word clouds to visualize the most frequent words in each category.

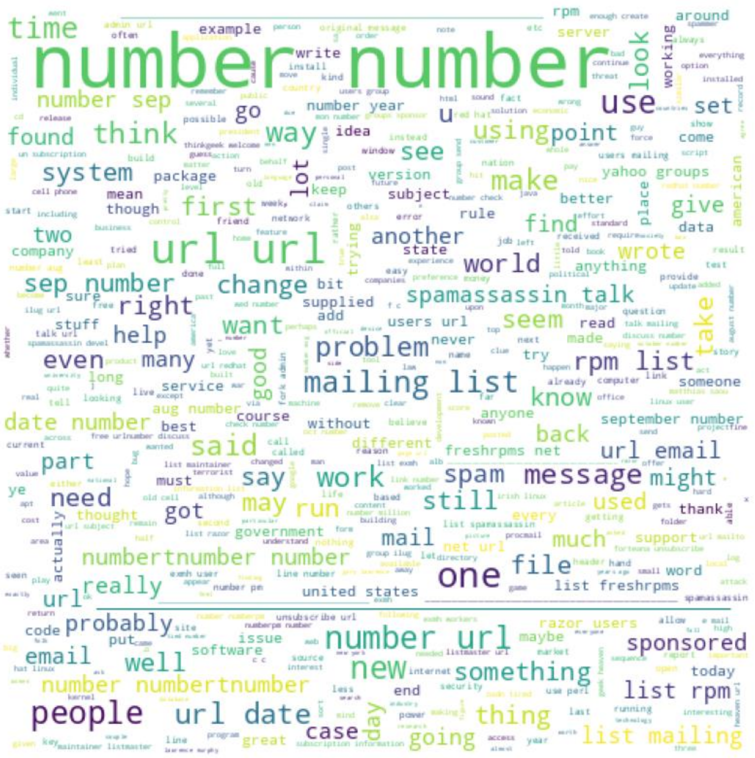


Figure 1 Ham words

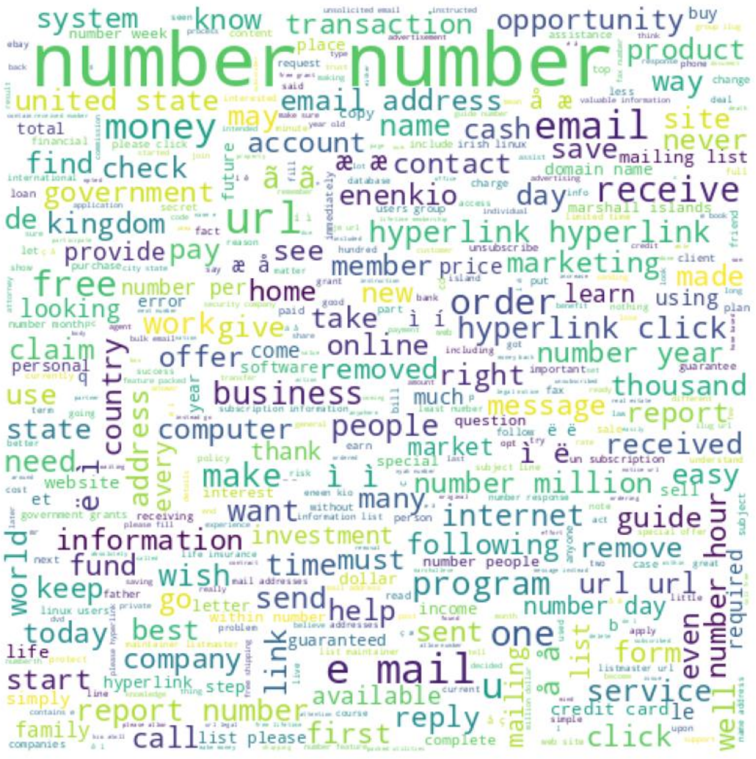
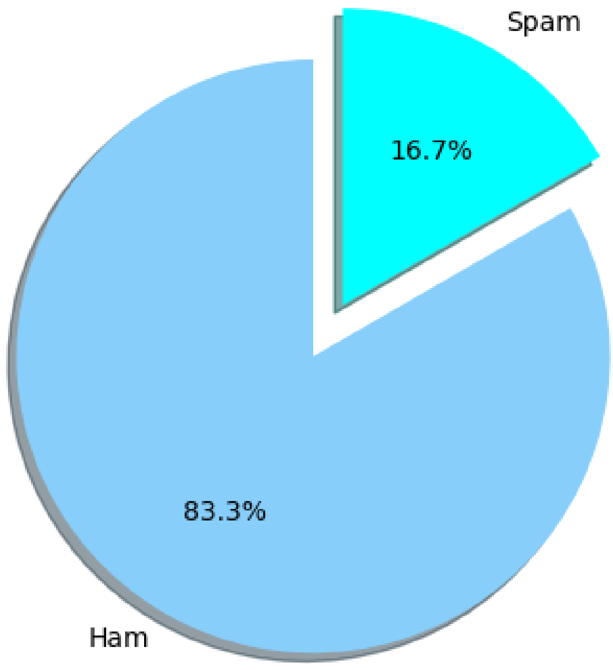


Figure 2 Spam Words

Then we generated a pie chart to visualize the distribution of two classes ('Ham' and 'Spam') in the dataset.



Then we obtained separate datasets for training and testing machine learning models. The training set are used to train the models, while the testing set are used to evaluate its performance on unseen data.

Then, we removed missing values, creating a **CountVectorizer** object to convert the text data into numerical features, and transformed the data into a document-term matrix for both the training and testing sets. These processed representations then are used as input features for training and evaluating a machine learning model.

4. Building Models

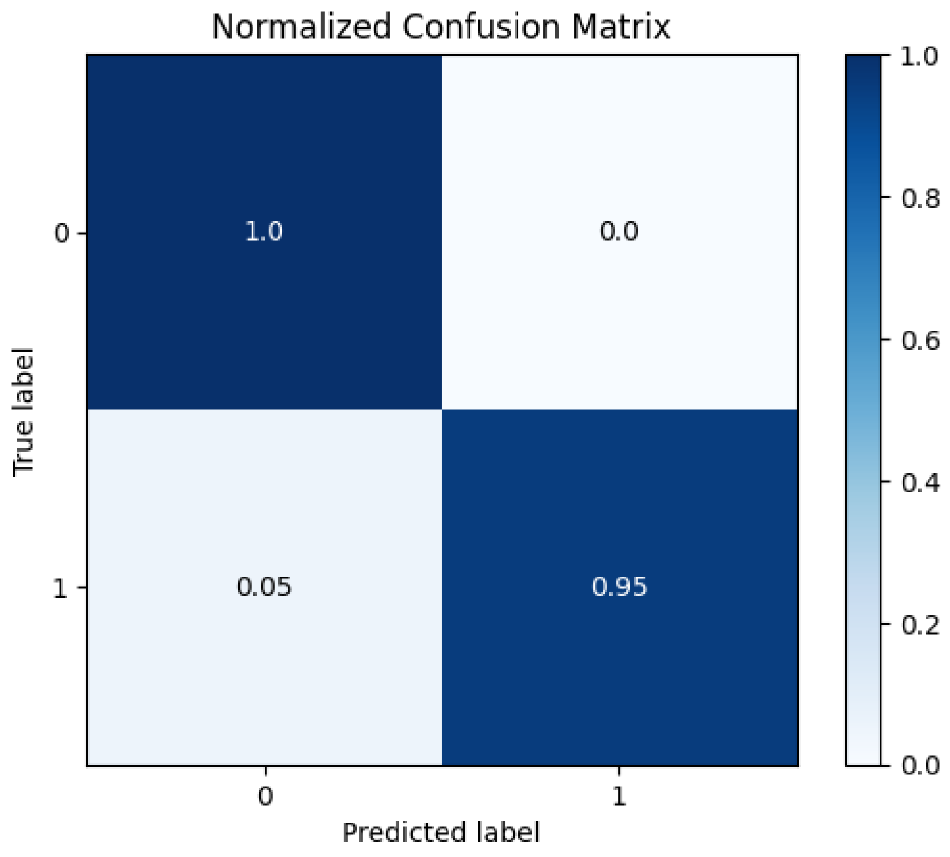
We have used;

- Naive Bayes Classifier
- K-Nearest Neighbors Algorithm
- Decision Tree Learning
- Support Vector Machine (SVM)
- Random Forest

data models.

A. *Naive Bayes Classifier:*

In web mining, the Naive Bayes classifier is a machine learning algorithm used for categorizing web documents into different classes or categories. It applies Bayes' theorem and assumes that the features (e.g., words or terms) in the document are conditionally independent given the class label. By calculating the probability of a document belonging to a specific class based on the occurrence or frequency of its features, the Naive Bayes classifier assigns the document to the class with the highest probability. It is widely employed in web mining tasks such as text classification, sentiment analysis, and spam detection due to its simplicity, efficiency, and effectiveness in handling large amounts of data.



B. *K-Nearest Neighbors Algorithm:*

In web mining, the K-Nearest Neighbors (KNN) algorithm is a classification algorithm used to categorize web documents into different classes based on their features. It operates by finding the K nearest neighbors to a given document in the feature space and assigning the document to the class that is most prevalent among its neighbors. The algorithm measures the similarity between documents based on their feature vectors, which could be word frequencies, TF-IDF values, or other representations. KNN is a non-parametric algorithm, meaning it does not assume any underlying probability distribution of the data. KNN can be useful in web mining for tasks such as document categorization, recommendation systems, and identifying similar documents based on their features. However, it can be computationally expensive, especially with large datasets, and the choice of K and the similarity metric can impact its performance.

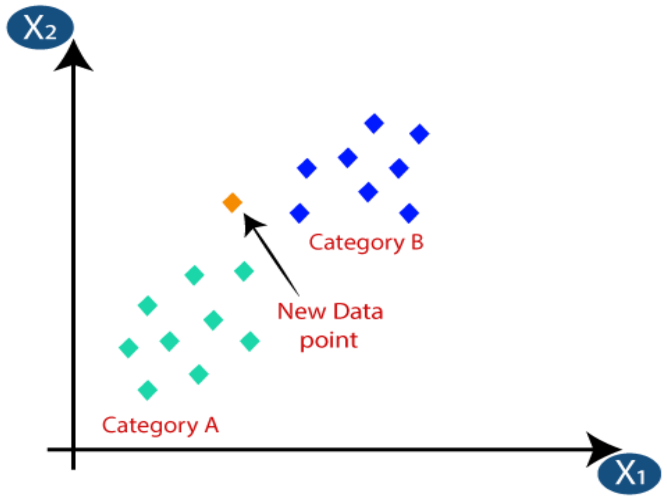
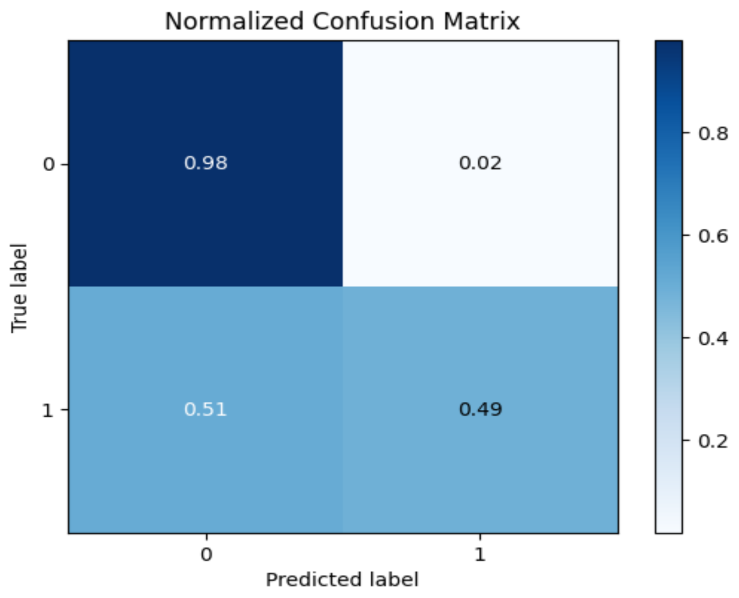
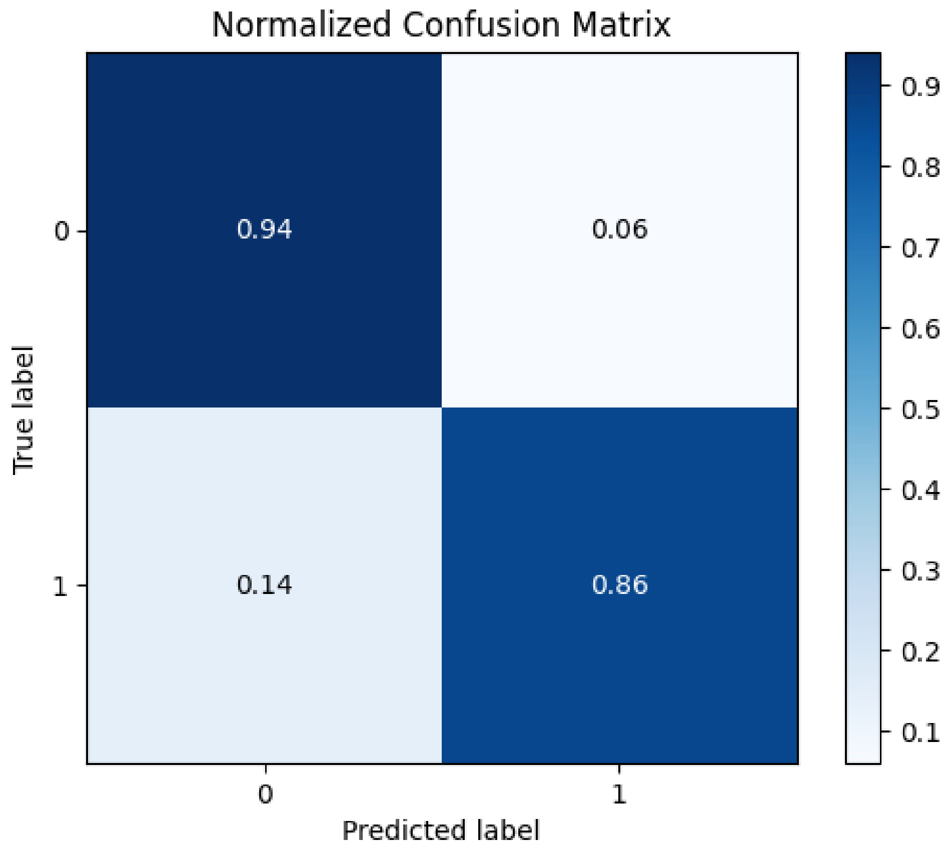


Figure 3KNN Algorithm



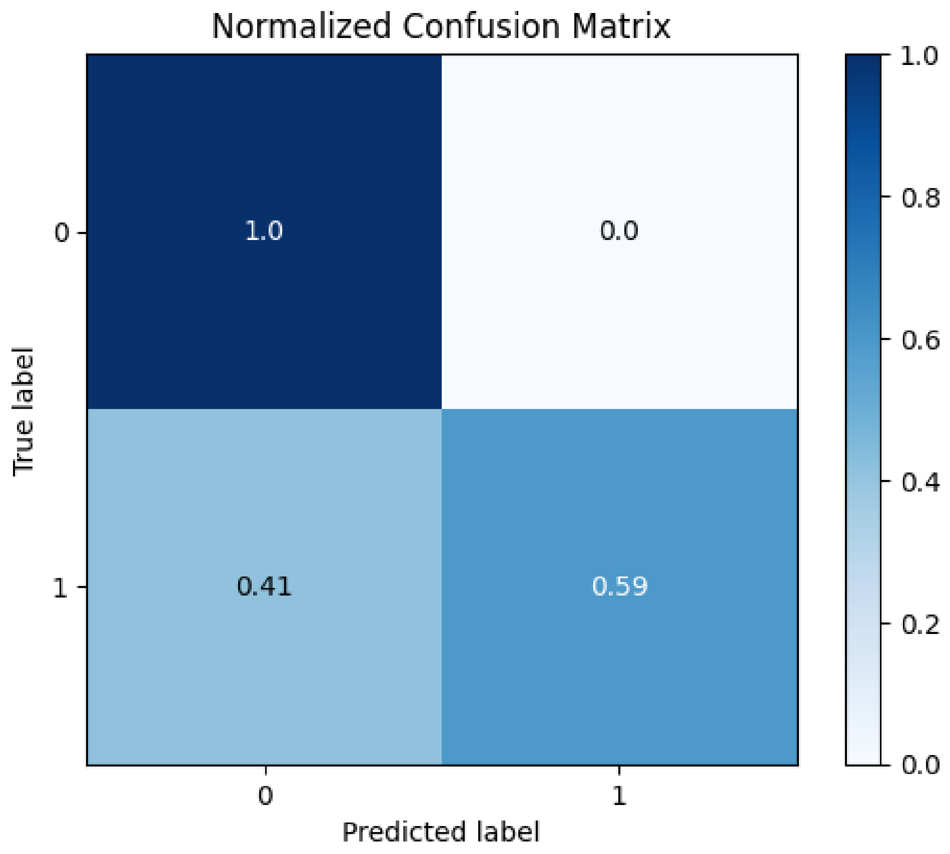
C. Decision Tree Learning:

Decision Tree Learning is a machine learning algorithm that builds a tree-like model to make predictions or classify instances based on a series of decision rules learned from the training data. The algorithm recursively partitions the data based on the values of different features, selecting the most informative features at each step to maximize the predictive accuracy. Each internal node of the tree represents a decision based on a feature, and each leaf node represents a predicted class label or outcome. Decision Tree Learning is popular in various domains, including web mining, as it provides interpretability, handles both numerical and categorical features, and can handle missing values. It is capable of learning complex decision boundaries and can handle large datasets efficiently.



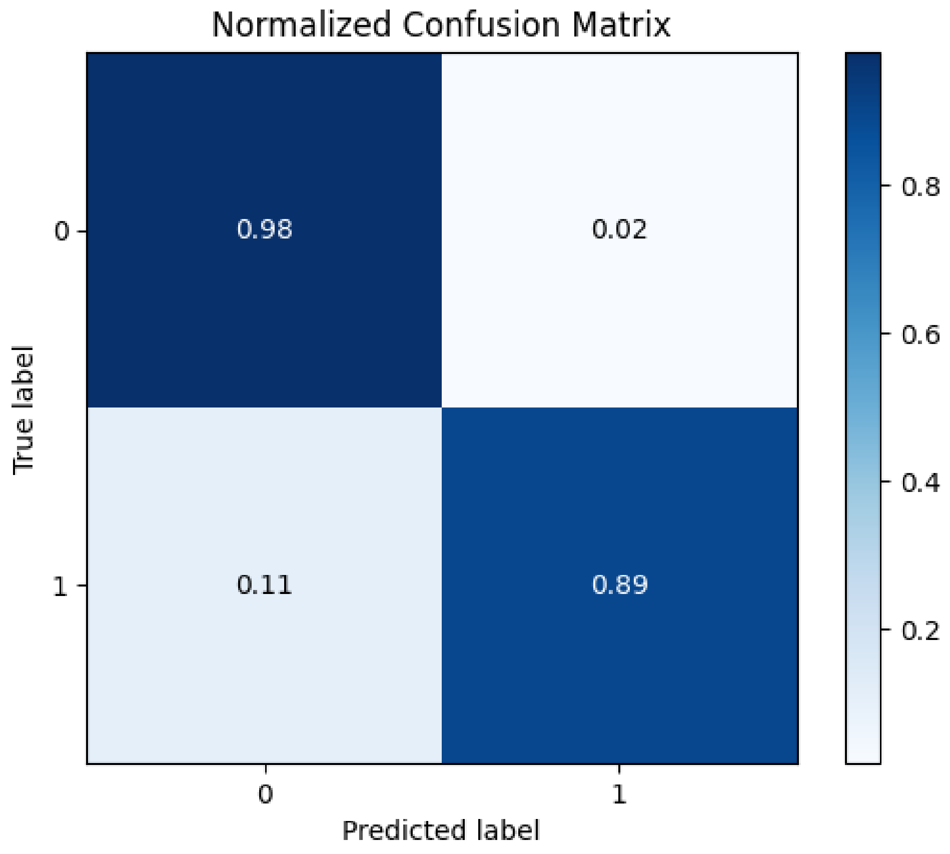
D. Support Vector Machine (SVM):

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for classification and regression tasks. It aims to find an optimal hyperplane that maximally separates data points of different classes by creating a high-dimensional decision boundary. By identifying support vectors, which are key data points close to the decision boundary, SVM can generalize well to new, unseen data. SVM is particularly effective in handling high-dimensional data and can handle both linearly separable and nonlinearly separable datasets through the use of kernel functions. It has become widely used in various domains, including web mining, for tasks such as text classification, sentiment analysis, and spam detection, due to its ability to achieve high accuracy and handle complex datasets.



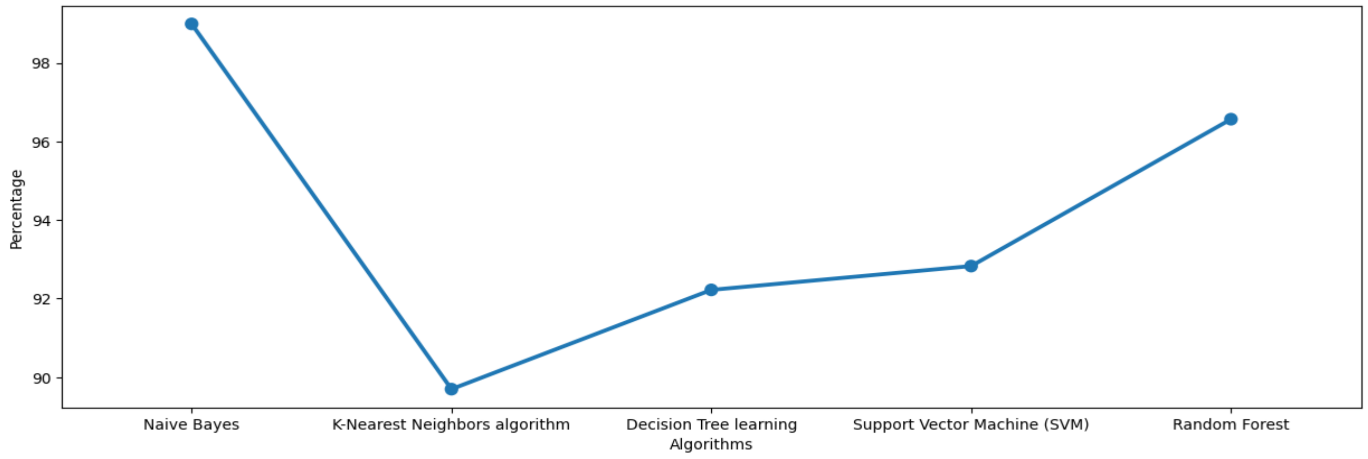
E. Random Forest:

Random Forest is a versatile ensemble learning algorithm used for both classification and regression tasks. It constructs a multitude of decision trees and combines their predictions to make accurate and robust predictions. Each decision tree is built using a random subset of the training data and a random subset of features, which introduces diversity and reduces overfitting. Random Forest leverages the concept of majority voting (classification) or averaging (regression) to determine the final prediction. It is known for its ability to handle high-dimensional data, handle missing values, and provide feature importance rankings. Random Forest is widely used in web mining for tasks such as web page classification, recommendation systems, and anomaly detection due to its effectiveness, scalability, and ability to handle complex datasets.



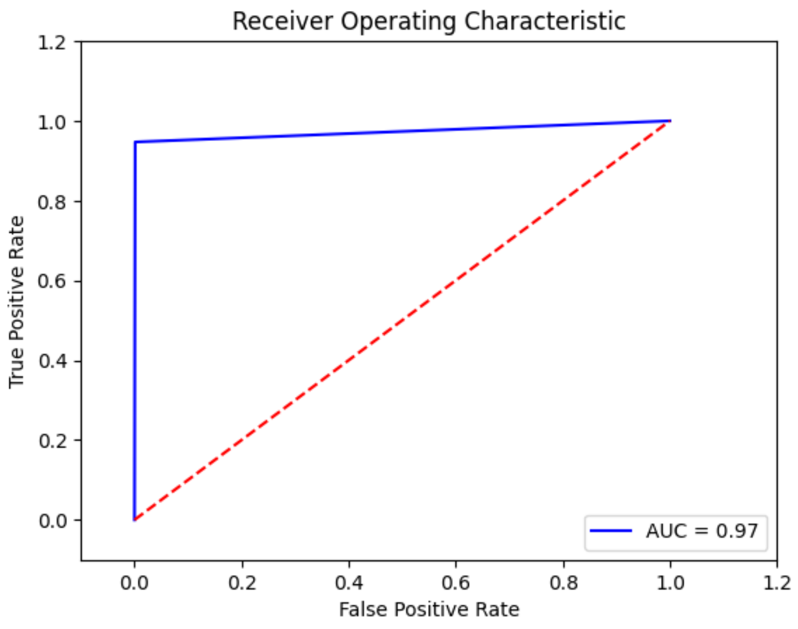
Method Comparison

We compared used methods accuracy percentages.



Roc Accuracy

In web mining, ROC accuracy refers to the evaluation of classification models that are used to predict binary outcomes, such as whether a web page is relevant or irrelevant, spam or non-spam, or fraudulent or legitimate. ROC (Receiver Operating Characteristic) analysis is a commonly used technique to assess the performance of these models. ROC accuracy measures the ability of a classification model to accurately distinguish between positive and negative instances by plotting the true positive rate (sensitivity) against the false positive rate (1 - specificity). The ROC accuracy is determined by the area under the ROC curve (AUC), with a higher value indicating better performance. It provides a comprehensive evaluation of the model's performance across various classification thresholds, making it a popular metric in web mining tasks.



5. Conclusion

We used Spam or Not Spam Dataset to create several models and compare their accuracies finding whether an email is spam or not. In conclusion, the performance evaluation of various classification algorithms on a spam detection project reveals significant differences in accuracy. Naive Bayes achieved the highest accuracy of 98.99%, making it a promising choice for accurately classifying spam and non-spam emails. Random Forest also performed impressively well, with an accuracy of 96.57%, showcasing its ability to effectively classify spam emails. Support Vector Machine (SVM) and Decision Tree learning achieved accuracies of 92.83% and 91.52% respectively, indicating their potential as viable options for spam detection. However, the K-Nearest Neighbors algorithm had a lower accuracy of 89.70%, suggesting that it may not be as suitable for this particular project. Overall, Naive Bayes and Random Forest demonstrate the highest effectiveness in accurately classifying spam emails, and further experimentation or fine-tuning may be required to improve the performance of the other algorithms.