1. Introduction

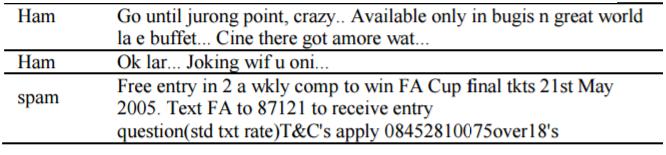
Mobile communication has revolutionized how we interact, but it has also opened doors for misuse, particularly through unwanted SMS messages. These unsolicited texts, often promising rewards like "reply to win a gold coin" or urging callbacks for inquiries, trick recipients into revealing personal information or losing money. The increasing prevalence of such spamSMS necessitates robust detection methods. This is where machinelearning comes in.

While there are parallels between email and SMS spam filtering, key differences exist. Collaborative content filtering, where users share information about spam emails, is a valuable technique that could be adapted for mobile spam. However, a significant distinction lies in message length: SMS messages are inherently brief. This brevity often leads spammers to craft messages with subtle yet deceptive patterns to capture attention. Machinelearningclassifiers are ideal for identifying these patterns by learning from existing data.

This paper focuses on using a supervised machine learning approach to detect spam in mobile text messages. We utilized a dataset of 5772 SMS messages from the UCI machine learning repository, each labeled as either legitimate ("ham") or spam as shown in Table 1. This is a binary classification problem, aiming to answer a simple "yes" or "no" question: "Is this message spam?"

The process involves converting the natural language text of the SMS messages into numerical vectors using vectorizers. Subsequently, various classifiers are employed to analyze their performance based on obtained scores. The classifier with the best performance is then used for making detailed predictions. All development and analysis for this project were conducted using Python.

Table 1: Example of ham and spam messages in UCI dataset.



1. Proposed Algorithm and Architecture

**Figure 1** illustrates a block diagram outlining the process of detecting spam in mobile text messages. The initial stage involves a dataset containing both 'ham' (legitimate) and spam messages, which is subsequently split into two subsets: one for training (teaching) and the other for testing. The training subset is used to fit the machine learning classifier, after which the model is evaluated to determine its accuracy using the test set.

The primary aim of this study is to categorize mobile text messages as either spam or non-spam through the application of a machine learning algorithm. The dataset used for this purpose is sourced from the UCI Machine Learning Repository. Each line in the dataset file represents a single text message and includes two columns: 'V1', which specifies the label (either 'ham' or 'spam'), and 'V2', which contains the actual text message.

The dataset comprises a total of 5,574 messages, of which 747 are labeled as spam and 4,827 as ham. This dataset is partitioned into a training set containing 4,400 messages and a test set consisting of 1,172 messages. The training set is employed to build the classifier, while the test set is used to assess its performance.

As the messages are originally in natural language, they must be transformed into a numerical format suitable for machine learning models. This is achieved using text vectorization techniques. In this context, three types of vectorizers are utilized to convert the raw text into vectors.

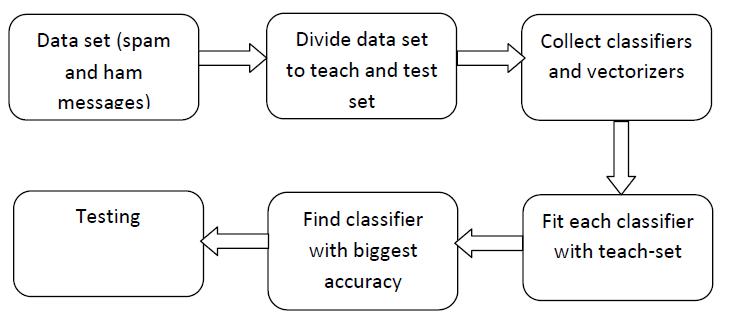


Fig. 1: Block diagram of detection of spam in mobile text messages

* 1. Vectorizers

Testing plays a major role in machine learning algorithms. But it’s not possible to fed the un-encoded messages directly to the algorithms. For that Sci kit-learn provides vectorizers which convert a collection of text documents into numerical feature vectors. There are three types of vectorizers namely count vectorizer, Tfidf vectorizer and hashing vectorizer.

*Count vectorizers*:Theyare used to convert the text message data into a matrix of token of unique words count.

*Tfidf vectorizer*: Tfidf means Term Frequency Inverse Document Frequency which performs normalization to a sparse matrix of occurrence counts using Tfidf transformation.

|  |  |
| --- | --- |
| tfidf(t; d) = tf(t; d) \* idf(t) | (1) |

*Hashing vectorizer*: Both count and Tfidf vectorizer produces several problems while it dealing with larger datasets. In order to overcome such things another type of vectorizer is hashing vectorizer. It converts a collection of text messages into matrix of token occurrences. But the demerit is that it is not possible to invert the model.

* 1. Classifiers

A classifier is a mathematical function used in classification algorithms to assign input data to specific categories or groups. Classification, in this context, is a form of supervised learning. Selecting an appropriate classifier for a given problem can be challenging. To address this, multiple classifiers are initially tested, including AdaBoost, Random Forest, Stochastic Gradient Descent (SGD), One-vs-Rest, Decision Tree, Ridge Classifier, among others. The classifier that yields the highest accuracy is chosen as the best performer.

SGD (Stochastic Gradient Descent) is known for its efficiency and suitability for large-scale applications and natural language processing tasks. The One-vs-Rest strategy is used when fitting individual classifiers for each class in a multi-class setting. The Passive-Aggressive classifier is another option known for its speed and ease of implementation when handling large datasets.

The Calibrated Classifier is used to adjust the predicted probabilities to better reflect the true likelihood of outcomes. The RidgeClassifierCV is a variation of the Ridge Classifier that includes built-in cross-validation to automatically determine the optimal hyperparameters.

By applying these classifiers along with different vectorization techniques, their performance can be compared. Among the tested combinations, the One-vs-Rest classifier paired with the TF-IDF vectorizer achieved the highest performance score.

1. Results

The One-vs-Rest classifier, when used in combination with the TF-IDF vectorizer, achieved an accuracy of 98.8%. Table 2 presents the performance of various classification models applied to the spam detection dataset. The first column lists the different classifiers, while the second column shows the accuracy scores obtained by each.

Out of a total of 1,172 test messages, the model correctly classified 1,158 messages and misclassified 14. The correct predictions refer to instances where spam messages were accurately identified as spam and legitimate (ham) messages were correctly labeled as ham, as illustrated in Figure 2. Incorrect predictions occurred when spam messages were mistakenly classified as ham, or vice versa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 2: Classifier and its corresponding scores | |  |  |  |
|  |  |  |  |  |
|  | Classifier | Score |  |  |
|  | OVR Classifier with Tfidf vectorizer | 0.988 |  |  |
|  | Calibrated Classifier with Count Vectorizer | 0.987 |  |  |
|  | Passive aggressive Classifier with Tfidf vectorizer | 0.986 |  |  |
|  | OVR with Count Vectorizer | 0.986 |  |  |
|  | Calibrated Classifier with Tfidf vectorizer | 0.985 |  |  |
|  | Passive aggressive Classifier with Hashing Vectorizer | 0.984 |  |  |
|  | SGD Classifier with Tfidf | 0.984 |  |  |
|  | SGD Classifier with Hashing vectorizer | 0.984 |  |  |
|  | Ridge ClassifierCV with Tfidf Vectorizer | 0.983 |  |  |
|  | Ridge ClassifierCV with Count Vectorizer | 0.982 |  |  |

1. Conclusions

With the growing use of text messaging, the issue of SMS spam has become increasingly prevalent. Detecting spam messages is a complex task due to the limited number of features available in the text data. In this study, various classifiers were employed for the learning process in machine learning, including One-vs-Rest (OvR), Stochastic Gradient Descent (SGD), Calibrated Classifier, Passive-Aggressive Classifier, and Ridge Classifier, along with different types of text vectorizers.

Among these combinations, the One-vs-Rest classifier paired with the TF-IDF vectorizer delivered the highest accuracy, making it the most effective model for prediction. The One-vs-Rest approach involves training one classifier per class, with each classifier distinguishing a single class from all others.

The entire implementation was carried out in Python using Google Colab, which offers a convenient environment for handling datasets and executing code.