ISSN: 0974-5823 Vol. 7 No. 6 June, 2022

*International Journal of Mechanical Engineering*

# Comparison of Feature Representation Schemes to Classify SMS Text using Data Balancing

**Suruchi Gautam**[1]*

[1] Department of Computer Science, Rajdhani College (University of Delhi), Raja Garden, New Delhi, India, 110015

suruchigaur29@gmail.com

**Abstract**

Text classification is a widely studied problem by data scientists. In this paper we address the problem of spam classification of SMS text. We note that the dataset is highly imbalanced with non-spam samples having a majority over the spam samples, thus affecting the Machine Learning classifiers to be biased towards the majority class. We address this issue by using Random Over Sampling to create more data points for the spam class. We investigate the effect of three feature representation schemes – Count-based, frequency-based and hashing-based on a balanced dataset and on an imbalanced dataset. We compare the performance of five different Machine Learning classifiers - k-Nearest Neighbor, Linear Support Vector Machine, RBF Support Vector Machine, Random Forest, and Decision Tee, for the purpose of Ham-Spam classification and compare these results using Accuracy, Precision, Recall, F1 measure metrics. Our experiments indicate that the classification accuracy of all classifiers (except k-NN, in this case) is enhanced on a balanced dataset. The highest classification accuracy of 99.95% and 97.7% was achieved using Linear SVM on frequency based balanced dataset and Hashing based imbalanced dataset, respectively.

**Keywords:** SMS Classification, Hashing, Machine Learning Classifiers, Data imbalance

## 1. Introduction

In this digital era most of the communication relies heavily on messages through smartphones, emails, etc. One of the most common uses of smartphones is texting. Sending an SMS message does not require an active internet connection, making texting a popular form of communication. This also makes it an attractive medium for spammers to send spam messages. Spam is an unsolicited message sent in bulk to a large number of recipients. Other than being quite a nuisance, it could also be a serious security concern as it may be used to extract personal and sensitive information such as credit card number, Bank account number, Aadhar card number, or ATM number. Some spam SMSs contain links, clicking on which allows malware to be installed on the device thus stealing personal information stored on the phone. People are generally on-the-go and less cautious while using their smartphones and may inadvertently click on these malicious links which take them to a legitimate looking app or to a fake site. Users are tricked to enter their confidential information there and fall prey to cyber fraud.

Therefore, it is very important to develop efficient means for detecting and filtering spam and non-spam, or ham messages. The short length of SMS messages, use of shorthand notations and slang, makes it more difficult to classify a message as Ham or Spam. A number of approaches have been used which basically work on first collecting the datasets, then data cleaning and tokenizing, followed by applying Machine Learning Classifiers and comparing the performance of each classifier.

In this paper, after cleaning, we tokenize the data using three techniques, such as, Bag of words, TF-IDF and Hash vectorizer. We then perform two experiments. In the first experiment we apply five different Machine Learning classifiers - K Nearest Neighbor, Linear Support Vector Machine, RBF Support Vector Machine, Random Forest, and Decision tree, for the purpose of Ham-Spam classification and compare these results using Accuracy, Precision, Recall, F1 measure metrics.

In the second experiment, we take into account the imbalance between the Ham and Spam classes. We also note that most Ham-Spam datasets consist of imbalanced data with non-spam messages having a majority over spam messages. When we apply a classifier on such imbalanced data for training, the classification tends to be biased towards the majority class, i.e., the non-spam or ham class. Although the models may classify ham messages with high accuracy, spam messages are largely misclassified. This is misleading in terms of assessing the classifiers and models used, thus making it difficult to predict different classes accurately. Therefore, as part of the second experiment, before applying the above-mentioned classifiers, we first balance the dataset and then test the comparative performance of each classifier. The main contributions of this study are:

Collection of datasets containing 5784 SMS messages

Bag of Words, Term Frequency- Inverse Document Frequency, Hashing for feature extraction

Addressing the issue of data imbalance using Random Over Sampling

Classification into Spam and Ham using five different classifiers - k-Nearest Neighbors, Linear SVM, RBF SVM, Random Forest, Decision Tree

Evaluation metrics - Accuracy, Precision, Recall, F 1 measure, AUC score

Performance Comparison of classifiers on the three models - Bag of Words model, TF-ID model and Hash Vectorizer model, using imbalanced and balanced dataset

The paper is organized as follows: Section 2 illustrates some of the recent work done for Ham-Spam classification. In Section 3 we discuss the datasets used, the preprocessing steps followed and our proposed methodology for classification. In Section 4 we present and discuss the results obtained by the proposed methodology. We also compare the performance of classifiers on an imbalanced and balanced dataset. Section 5 concludes the paper with future work.

## 2. Related Work

In the extant literature, researchers have proposed various methodologies for the detection and classification of SMS text. Mehul Gupta et.al [1] proposed a SMS classification scheme using supervised Machine Learning algorithms namely, Support Vector Machine (SVM), Naive Bayes, Decision Tree, Logistic Regression, Random Forest, AdaBoost, Artificial Neural Network and Convolutional Neural Network. They compared the performance of each of the classifiers on the metrics Accuracy, Precision, Recall, CAP curve, AR and Positive % on two datasets. The first dataset contains 5574 messages and the second one has 2000 messages, respectively. They achieved the highest accuracy of 99.19% and 98.25% and 0.9926 and 0.9994 AR values on each of the two datasets. They observed that CNN showed significant improvement against traditional classifiers. SVM gave an accuracy of 98.57% and 96.25% and Naive Bayes 98.48% and 96.75%, respectively, on the two selected datasets.

Saeed W. [2] performed content-based Spam message filtering by comparing the performance of three automatic Machine Learning tools - mljar AutoML, H2O AutoML and TPOT AutoML. They used Information Gain (IG) to select features and experimented with 50, 100 and 200 feature sets. They used 9 ML models such as Baseline, Decision Tree, Random Forest, XGBoost, Light GBM, CatBoost, Extra Trees, Neural Networks, and Nearest Neighbor with the mljar AutoML tool. For the H2O AutoML, Random Forest, Generalized Linear Model (GLM), Gradient Boosting Machine (GLM), XGBoost were used and for the TPOT AutoML Naive Bayes, Decision Tree, Extra Trees, Random Forest, Gradient Boosting, Logistic Regression, XGBoost, Nearest Neighbor and Neural Networks were used. Their experiments concluded that every tool gave its best performance with the 200 feature set. Their experiments also indicate that Stacked Ensemble models and Gradient Boosting models yield the best performance for each feature size. The Stacked Ensemble model built using H2O AutoML with 200 features gave the best performance with 0.8370 as Log loss, 1087 as True Positive and 268 as True Negative values. The training time was 2 hours, 3 hours and 4 hours on 50, 100 and 200 feature size sets. The mljar AutoML and TPOT AutoML took the entire time allotted for training, whereas, H2O AutoML took less time and finished before time.

In a study proposed by Navaney P, et. al [3], they applied text mining techniques on the SMS text and visualized the data using WordCloud to detect catch phrases. They used the SMS Spam dataset containing

5574 records with 4827 Ham and 747 Spam messages. They used three classifiers - SVM, Naive Bayes Maximum, Entropy, and compared their performance using crosstable. They were able to achieve an accuracy of 97.4% through the SVM classifier.

In another study conducted by GuangJun, L. et.al [4], the authors collected the SMS Spam dataset from the Kaggle repository containing 4900 Ham and 672 Spam samples. They used 70% of the data for training and 30% for validation. They compared the performance of Logistic Regression, k-Nearest Neighbors and Decision Tree classifiers and found that the performance of Logistic Regression with hyperparameter C=1 was the best with a 99% accuracy value.

Zhao, C, et.al [5] proposed a heterogeneous ensemble framework for handling data imbalance in spam detection. The framework proposed by them has two modules - a base module and a combining module. In the base module they use six different classifiers such as SVM, CART, Gaussian Naive Bayes, k-Nearest Neighbors, Random Forest and Linear Regression. The results of these classifiers are used as input to the combining module which generates metadata using bootstrap, bagging and cross-validation. The metaclassifier is a Deep Neural Network. Softmax algorithm is used between the hidden layer and the output layer. Cost-sensitive modified cross entropy is used as the loss function. They assigned higher costs for misclassification of Spam as Ham. They experimented on the real Spam Twitter dataset and observed that their model achieves best performance with 3 hidden layers, with 64 hidden units in the first layer, 32 hidden units in the second layer, and 16 hidden units in the third layer. They evaluated the model on the basis of TPR (True Positive Rate), FPR (False Positive Rate), Precision, G-mean and Kappa Score. They concluded that their method achieves better results as compared to ensemble learning methods.

Liu, S. [6] handle data imbalance between ham and spam class using three methods - Random Over Sampling (ROS), Random Under Sampling (RUS) and propose a Fuzzy-based Over Sampling (FOS) function. They used real time Twitter data for their experiment. Treating spam tweets as positive class and ham tweets as negative class, they define a class imbalance ratio n/m, where n is the number of spam tweets and m is the number of ham tweets, and n is much less than m. They experimented on different datasets with 10 different values of the imbalance ratio ranging from 2, 4, 6, ..., 20 and compared the performance on the basis of True Positive Rate, False Positive Rate and F-measure. They concluded that using an imbalanced dataset, Random Forest classifier identifies 66% of spam tweets. After balancing the dataset using ROS, RUS, FOS, the detection rate improves to 74%, 79%, 74% respectively. The ensemble approach successfully detects 75% of spam tweets. As the imbalance ratio increases, the true positive rate and false positive rates decrease. They concluded that the ensemble approach performs better in identifying spam tweets with an F-measure of 76% and then decreases to 55% as the imbalance ratio increases from 2 to 20.

In another similar study, Saeed, R M, et.al [7] present four different detection methods in Arabic spam reviews. They experiment on two publicly available datasets - Deceptive Opinion Spam Corpus (DOSC) and Hotel Arabic Reviews Dataset (HARD). After preprocessing they extract features using N-gram, negation handling and content-based feature extraction methods. For Spam detection, they use four approaches - Rule-based classifier, Machine Learning classifier, Majority voting ensemble and Stacking ensemble classifier. Their experiments indicate that by handling negation the classification accuracy increases from 66.33% to 82.38% on DOSC dataset and from 71.45% to 98.35% on the HARD dataset. They concluded that with the triple combination of N-gram features and negation handling the classification accuracy increases, also improving specificity, recall, precision and F1 score. The authors were able to achieve the highest accuracy with the Decision Tree for the DOSC dataset and Boosting for the HARD dataset. The majority voting ensemble classifier gave accuracy of 74.38% and 98.21%, while the stacking ensemble classifier by combining K-means classifier with rule-based classifier gave accuracy of 95.25%, and 99.98% on DOSC and HARD datasets, respectively.

Baccouche, A, et. al [8] proposed a LSTM model that performs best for binary as well as multi-class text classification. They worked on the Youtube spam comments dataset which contains 2394 comments on videos and Nigerian Fraudulent emails containing 11,000 emails with nearly balanced Fraud or Non-fraud labels. After preprocessing they used Word2Vec embedding and the LSTM classifier model. They also created a joint dataset consisting of bigrams present in both datasets. In their experiment they applied SVM, Naive Bayes and Stochastic Gradient Descent (SGD) as baseline models. Their experiments show that Naive Bayes performed best in all the three datasets and the gave better results with the joint dataset with the highest accuracy of 92.7%. They concluded that by combining text from different sources within a similar domain the performance of the original classifiers was enhanced.

Jin Zhipeng, et. al [9] worked on spam detection in Weibo, a platform similar to Twitter, used for information sharing in China. Their model works in three stages - Feature extraction, expanding keyword features and under-sampling ensembles to handle class imbalance. They categorize features as general, sentence pattern and keyword features. For general features they choose the number of hashtags, "@username" mentions, retweets, Special symbols, punctuations, number sequences, mention of other social tools, length of tweets, etc. Robot generated tweets generally have a fixed pattern. These patterns are extracted and encoded as regular expressions. These are used as sentence features. Since most spam messages are generated from e-commerce sites, they use product advertisements to expand keyword features and generate unigram and bigram features as keyword features. Each feature maintains a list of n-gram characters with term frequencies greater than a threshold value. For classification, they use a majority voting ensemble on three classifiers - C4.5, logistic regression and random forest. To balance the spam and ham class, they have Random Under Sampling and generate more spam samples so that the imbalance rate is approximately 1:2. Their experiment concludes that by expanding the keyword feature set the performance of the model greatly improves.

## 3. Proposed Framework

In this paper, we propose a framework that first uses unigram and bigram to represent features, thereafter, represents them using three tokenizing techniques - count based, frequency based and Hash based, on the preprocessed data. Machine Learning Classifiers are trained and tested on these features. The performance of these classifiers is evaluated using accuracy, precision, recall, F1-measure, AUC and ROC curve.

We conduct this study on the SMS dataset which contains 5677 text messages, with 747 spam and 4827 ham messages. As a first step, we preprocess and clean the data by removing stop words, punctuation, spaces, and lower case conversion. In the next step we create tokens using unigrams and bigrams from the cleaned text which are then represented using three techniques - count-based (Bag of Words), frequency-based (TF-IDF) and hashing based (Hash Vectorizer).

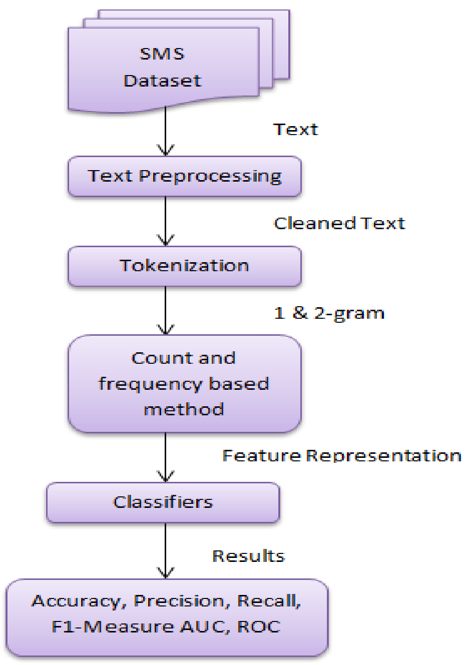

**Figure 1.** Proposed framework used for SMS classification.

The study is conducted in two phases. In the first phase, we experiment on the imbalanced dataset by evaluating and comparing the performance of five different machine learning classifiers - K Nearest Neighbor, Linear Support Vector Machine, RBF Support Vector Machine, Random Forest, and Decision Tree on the basis of accuracy, precision, recall, f1-measure.

In the second phase, we balance the dataset using Random Over Sampling by generating more samples for the spam class. Our study is motivated by the observation that the number of ham or non-spam samples are much more tha n the number of spam samples. Thus, by applying ML Classifiers on such imbalanced data, classification of non-spam messages is quite accurate, however, spam messages get largely misclassified as the classifiers tend to be biased towards the majority class. This could be misleading, particularly for multi-class identification. Therefore, in the second experiment, we try to balance out the two classes and compare the performance of each of the five classifiers on the three tokenized models mentioned above.

Based on these two experiments, we draw conclusions about the effect of tokenizing techniques on balanced and imbalanced datasets for text classification.

The proposed framework is depicted in Figure 1. Each block of the proposed framework is described in the following subsections.

### *3.1 Datasets*

The data set has been collected from https://archive.ics.uci.edu/ml/datasets/sms+spam+collection. The dataset contains 5574 text messages, with 4287 Ham and 747 Spam messages. The details of the dataset are shown in Table 1. Figure 2 shows that the dataset is highly imbalanced.

**Table 1.** Dataset details.

| Category | Numbers |
|---|---|
| Spam | 747 |
| Ham | 4827 |

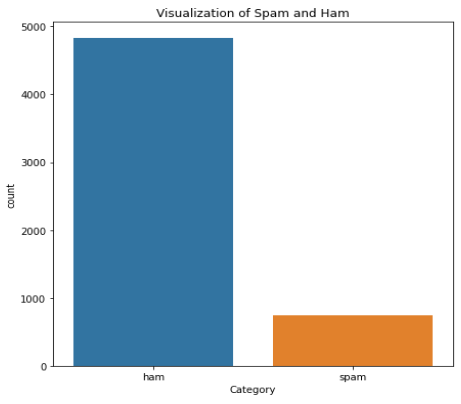

**Figure 2.** Graph representing number of Ham and Spam samples.

### *3.2 Preprocessing*

In the text preprocessing step, the data is preprocessed and cleaned by converting into lowercase, removing punctuation, special symbols and stop words. The stop words are provided in the NLTK library. Stop words occur in both spam as well as in ham messages and do not contribute to the classification procedure, therefore, by removing these words we can select more relevant features. We also remove numeric data such as mobile numbers and phone numbers from the text.

### *3.3 Tokenization*

Tokenization is the process of converting the text into tokens. After the text is pre-processed and cleaned, each word from the text is converted into a token by applying the tokenization process. We create unigrams and bigrams from the text. Unigram is a single word sequence and Bi-gram is the combination of two words occurring one after another in a SMS.

### *3.4 Count based, Frequency based and Hashing based methods*

Three methods, count based, frequency based and hashing based, are used to convert the unigram and bigram into numeric vectors. Count based method uses the bag of words model. The Frequency-based method uses the Term Frequency-Inverse Document Frequency (TF-IDF) model, and the Hashing based model uses Hash Vectorizer to convert the tokens of both unigram and bigram into real valued vectors.

Bag of words model counts the number of times a token appears in the document which results in a term document matrix. Term document matrix is a matrix in which rows represent the documents and columns represent the tokens.

Term frequency-inverse document frequency (TF-IDF) is calculated by multiplying the term frequency to inverse document frequency using equation (1):

tf-id f = tf * idf , where, (1)

tf(t,d) = number of times a token 't' occurs in document d number of words in d (2)

idf(t) = number of documents of documents containing 't' (3)

Hash vectorizer uses a hash function to map token strings in the input to feature indices. The hash function converts text documents to a sparse matrix containing occurrence counts of tokens in the document. These output values are in a predefined range. In this way, any words that are out of the vocabulary also get mapped to a vector of the same size as the training vectors.

### *3.5 Classifiers*

In this study, we have used five classifiers namely k-Nearest neighbor, Linear Support Vector Machine, RBF Support Vector Machine, Random Forest and Decision Tree for the purpose of classification.

## 4. Results and discussion

This section contains the tabulated results of our experiments on the balanced and imbalanced datasets. We also discuss the performance of classifiers using each of the three tokenizing schemes mentioned in the paper and the effect of balancing the dataset.

### *4.1 Presentation of results*

We present the results of our experiments in this subsection by tabulating the results of experiment 1 (Imbalanced dataset) and experiment 2 (Balanced dataset). Tables 2, 3 and 4 pertain to experiment 1, and Tables 5, 6, 7 to experiment 2. We evaluate and compare the performance of five classifiers (k-Nearest neighbor, Linear Support Vector Machine, RBF Support Vector Machine, Random Forest and Decision Tree) on Count based, Frequency based and Hashing based tokenized dataset. Table 8 presents a comparative analysis of the performance accuracy of all the classifiers.

**Table 2.** Classifier performance on Count based imbalanced dataset.

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| k-Nearest Neighbor | 0.85939397 | 0.429699 | 0.500000 | 0.462191 |
| Decision Tree | 0.961980 | 0.945212 | 0.892533 | 0.916460 |
| Linear SVM | **0.974892** | **0.985807** | **0.910714** | **0.943782** |
| RBF SVM | 0.859397 | 0.429699 | 0.500000 | 0.462191 |
| Random Forest | 0.888092 | 0.942393 | 0.602041 | 0.638927 |

**Table 3.** Classifier performance on Frequency based imbalanced dataset.

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| k-Nearest Neighbor | 0.937590 | 0.961980 | 0.780195 | 0.840773 |
| Decision Tree | 0.956958 | 0.928599 | 0.887478 | 0.906512 |
| Linear SVM | **0.971306** | **0.981059** | **0.900093** | **0.935286** |
| RBF SVM | 0.859397 | 0.429699 | 0.500000 | 0.462191 |
| Random Forest | 0.887374 | 0.942066 | 0.599490 | 0.635209 |

**Table 4.** Classifier performance on Hashing based imbalanced dataset.

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| k-Nearest Neighbor | 0.916786 | 0.950076 | 0.706215 | 0.768259 |
| Decision Tree | 0.954806 | 0.921008 | 0.886226 | 0.902506 |
| Linear SVM | **0.977044** | **0.981813** | **0.922635** | **0.949443** |
| RBF SVM | 0.865136 | 0.932179 | 0.520408 | 0.502838 |
| Random Forest | 0.862267 | 0.930935 | 0.510204 | 0.482906 |

**Table 5.** Classifier performance on count based balanced dataset.

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| k-Nearest Neighbor | 0.614747 | 0.783419 | 0.611529 | 0.544149 |
| Decision Tree | 0.987158 | 0.987285 | 0.987243 | 0.987158 |
| Linear SVM | **0.997929** | **0.997932** | **0.997925** | **0.997929** |
| RBF SVM | **0.997929** | **0.997954** | **0.997911** | **0.997929** |
| Random Forest | 0.946147 | 0.951745 | 0.945698 | 0.945939 |

**Table 6.** Classifier performance on Frequency based balanced dataset.

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| k-Nearest Neighbor | 0.926678 | 0.926990 | 0.926573 | 0.926647 |
| Decision Tree | 0.982187 | 0.982661 | 0.982334 | 0.9821862 |
| Linear SVM | **0.999586** | **0.999589** | **0.999582** | **0.9995863** |
| RBF SVM | 0.933306 | 0.939848 | 0.932810 | 0.9330004 |
| Random Forest | 0.951533 | 0.956147 | 0.951128 | 0.951376 |

**Table 7.** Classifier performance on Hashing based balanced dataset.

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| k-Nearest Neighbor | 0.910936 | 0.911270 | 0.910824 | 0.910896 |
| Decision Tree | 0.980530 | 0.980954 | 0.980670 | 0.980529 |
| Linear SVM | **0.994200** | **0.994195** | **0.994207** | **0.994200** |
| RBF SVM | 0.946562 | 0.948494 | 0.946301 | 0.946479 |
| Random Forest | 0.979702 | 0.980648 | 0.979532 | 0.979685 |

In experiment 1 (imbalanced dataset), we observed that Linear SVM classifier outperformed all the other classifiers in all the three models i.e., Count based, Frequency based and Hash based model. The highest accuracy was achieved with the Hashing based tokenized dataset with an accuracy of 97.70%, 0.9818 precision, 0.9226 recall and 0.9494 F1 score. Decision Tree was a close second, achieving the highest accuracy of 96.19% with the count-based tokenizer.

As a result of experiment 2 (balanced dataset), the best performance was achieved by Linear SVM again. We got the best accuracy of 99.95% with TF-IDF tokenizer and 0.9995 as precision, recall and F1 score values. The second highest accuracy of 99.79% was achieved with the RBF classifier on the count-based model and Linear SVM with count based model.

*4.2 Comparative analysis of results*

We compare the performance of the classifiers used on balanced and unbalanced datasets on each of the three tokenizing schemes. The summarized results of our experiments are listed in Table 8.

**Table 8.** Comparative analysis of all classifiers employed on Balanced and Imbalanced datasets using the three tokenizing techniques.

| | Classification Accuracy % | | | | | |
|---|---|---|---|---|---|---|
| | Count based Tokenizer | | TF-IDF Tokenizer | | Hash based Tokenizer | |
| Classifier | Balanced Dataset | Imbalanced Dataset | Balanced Dataset | Imbalanced Dataset | Balanced Dataset | Imbalanced Dataset |
| k-Nearest Neighbor | 61.47 | 85.93 | 92.66 | 93.75 | 91.09 | 91.67 |
| Decision Tree | 98.71 | 96.19 | 98.21 | 92.85 | 98.08 | 95.48 |
| Linear SVM | **99.79** | **97.48** | **99.95** | **97.13** | **99.42** | **97.70** |
| RBF SVM | **99.79** | 85.93 | 93.33 | 85.93 | 94.65 | 86.51 |
| Random Forest | 94.61 | 88.80 | 95.15 | 88.73 | 97.97 | 86.22 |

The results of our experiments are summarized in Table 8. We make the following observations:

1. The performance of all the classifiers gets enhanced when working on a balanced dataset. This enhancement is seen in all the three tokenizing techniques used in the study. However, k-Nearest Neighbor is an exception to this observation where the classification accuracy actually decreases on the balanced dataset.

2. The best performance is achieved by Linear SVM classifier with both the balanced and imbalanced datasets in each of the three tokenizing models, with Decision Tree coming in as a close second.

3. We see a significant improvement in the performances of RBF SVM and Random Forest when we employ them on a balanced dataset.

4. The highest accuracy of 97.70% is achieved with Linear SVM and the Hash based model and 97.48% with the Bag of Words model for the imbalanced dataset.

5. With the balanced dataset, Linear SVM and the TF-IDF tokenizer yield the best results with an accuracy of 99.95%. RBF SVM also gives the same accuracy (99.95%) with the count-based model and a balanced dataset.

6. The Decision Tree classifier gives second best results in all the cases with 98.71% as its highest accuracy with the count-based model using a balanced dataset.

7. The k-Nearest Neighbor proves to be a poor classifier for this dataset with the lowest values for accuracy, precision, recall and F1 score.

Graphical representation renders clarity in visualizing results. Figure 3, 4 and 5 represent our results on balanced and imbalanced datasets using the three tokenizing schemes.

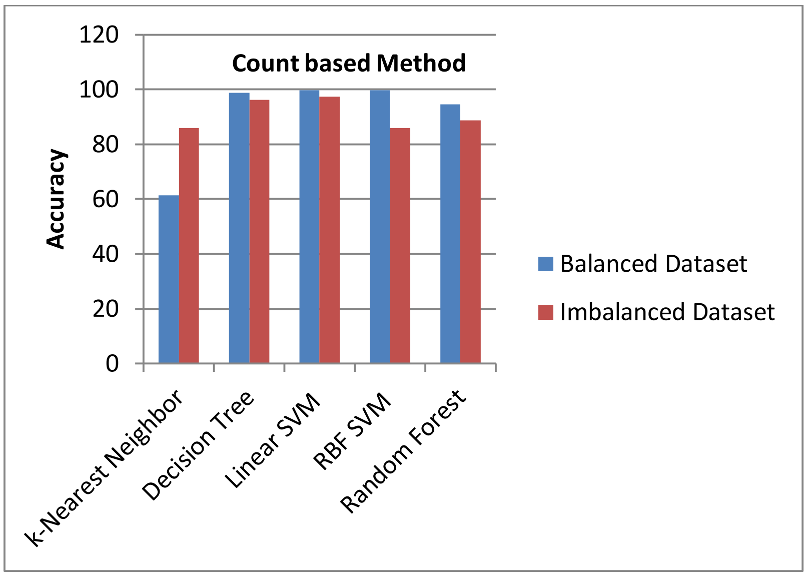

**Figure 3.** Performance accuracy comparison of count-based method.

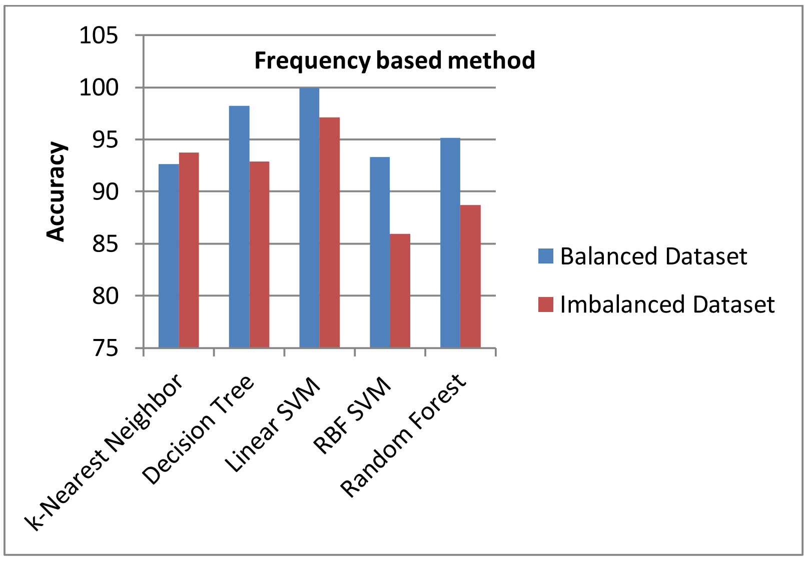

**Figure 4.** Performance accuracy comparison of frequency-based method.

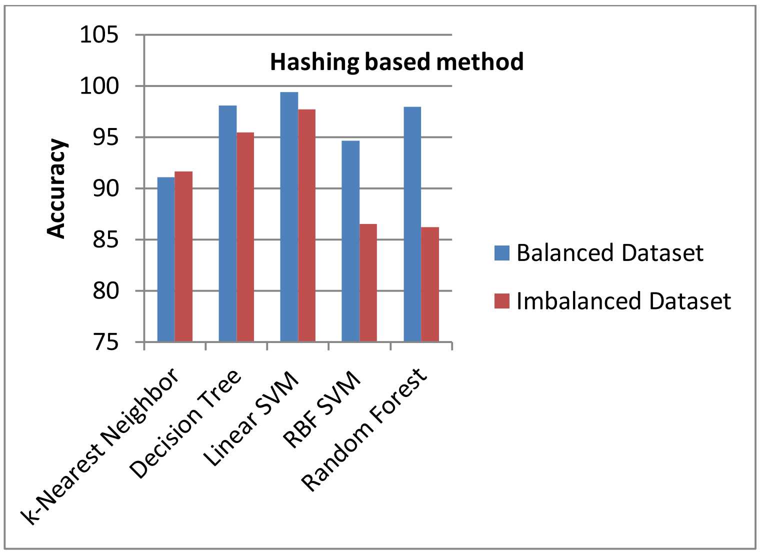

**Figure 5.** Performance accuracy comparison of hashing-based method.

## 5. Conclusions and future work

We used supervised learning classifiers for ham-spam classification of SMS data set. Our observations were made under two categories (a) Balanced data set and (b) Imbalanced data set. We found that Linear SVM performed best in both the categories, whereas KNN gave the lowest score in both categories.

For future study, we can experiment with other sample generation techniques like Random Under Sampling, and study the performance of the classifiers for each case.

We used supervised learning classifiers for this study. As an extension to this work, we can employ deep learning classifiers and compare their performance. The same classifiers and tokenizing methods can be tested on multi-class classification as well.

## References

[1] Gupta M, Bakliwal A, Agarwal S, Mehndiratta P. A comparative study of spam SMS detection using machine learning classifiers. In2018 Eleventh International Conference on Contemporary Computing (IC3) 2018 Aug 2 (pp. 1-7). IEEE.

[2] Saeed W. Comparison of Automated Machine Learning Tools for SMS Spam Message Filtering. arXiv preprint arXiv:2106.08671. 2021 Jun 16.

[3] Navaney P, Dubey G, Rana A. SMS spam filtering using supervised machine learning algorithms. In2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence) 2018 Jan 11 (pp. 43-48). IEEE.

[4] GuangJun L, Nazir S, Khan HU, Haq AU. Spam detection approach for secure mobile message communication using machine learning algorithms. Security and Communication Networks. 2020 Jul 9;2020.

[5] Zhao C, Xin Y, Li X, Yang Y, Chen Y. A heterogeneous ensemble learning framework for spam detection in social networks with imbalanced data. Applied Sciences. 2020 Jan;10(3):936.

[6] Liu S, Wang Y, Zhang J, Chen C, Xiang Y. Addressing the class imbalance problem in twitter spam detection using ensemble learning. Computers & Security. 2017 Aug 1;69:35-49.

[7] Saeed RM, Rady S, Gharib TF. An ensemble approach for spam detection in Arabic opinion texts. Journal of King Saud University-Computer and Information Sciences. 2019 Oct 11.

[8] Baccouche A, Ahmed S, Sierra-Sosa D, Elmaghraby A. Malicious text identification: deep learning from public comments and emails. Information. 2020 Jun;11(6):312.

[9] Jin Z, Li Q, Zeng D, Wang L. Filtering spam in Weibo using ensemble imbalanced classification and knowledge expansion. In2015 IEEE International Conference on Intelligence and Security Informatics (ISI) 2015 May 27 (pp. 132-134). IEEE.

[10] Wang F, Liu Z, Wang C. An improved k NN text classification method. International Journal of Computational Science and Engineering. 2019;20(3):397-403.

[11] Fanny F, Muliono Y, Tanzil F. A comparison of text classification methods k-NN, Naïve Bayes, and support vector machine for news classification. Jurnal Informatika: Jurnal Pengembangan IT. 2018 May 13;3(2):157-60.

[12] Azam M, Ahmed T, Sabah F, Hussain MI. Feature extraction based text classification using k-nearest neighbor algorithm. IJCSNS Int. J. Comput. Sci. Netw. Secur. 2018 Dec 30;18(12):95-101.

[13] Goh YM, Ubeynarayana CU. Construction accident narrative classification: An evaluation of text mining techniques. Accident Analysis & Prevention. 2017 Nov 1;108:122-30.

[14] Wang Z, Qu Z. Research on Web text classification algorithm based on improved CNN and SVM. In2017 IEEE 17th International Conference on Communication Technology (ICCT) 2017 Oct 27 (pp. 1958-1961). IEEE.

[15] Vijayan VK, Bindu KR, Parameswaran L. A comprehensive study of text classification algorithms. In2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2017 Sep 13 (pp. 1109-1113). IEEE.

[16] Prasetijo AB, Isnanto RR, Eridani D, Soetrisno YA, Arfan M, Sofwan A. Hoax detection system on Indonesian news sites based on text classification using SVM and SGD. In2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE) 2017 Oct 18 (pp. 45-49). IEEE.

[17] Ong MS, Magrabi F, Coiera E. Automated categorisation of clinical incident reports using statistical text classification. Quality and Safety in Health Care. 2010 Dec 1;19(6):e55-.

[18] Fei G, Liu B. Breaking the closed world assumption in text classification. InProceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2016 Jun (pp. 506-514).

[19] Shah K, Patel H, Sanghvi D, Shah M. A comparative analysis of logistic regression, random forest and KNN models for the text classification. Augmented Human Research. 2020 Dec;5(1):1-6.

[20] Salles T, Gonçalves M, Rodrigues V, Rocha L. Improving random forests by neighborhood projection for effective text classification. Information Systems. 2018 Sep 1;77:1-21.

[21] Bouaziz A, Dartigues-Pallez C, da Costa Pereira C, Precioso F, Lloret P. Short text classification using semantic random forest. InInternational Conference on Data Warehousing and Knowledge Discovery 2014 Sep 2 (pp. 288-299). Springer, Cham.

[22] Pranckevičius T, Marcinkevičius V. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. Baltic Journal of Modern Computing. 2017;5(2):221.

[23] Ababneh J. Application of Naïve Bayes, Decision Tree, and K-Nearest Neighbors for Automated Text Classification. Modern Applied Science. 2019 Oct;13(11):31.

[24] Deng X, Li Y, Weng J, Zhang J. Feature selection for text classification: A review. Multimedia Tools & Applications. 2019 Feb 1;78(3).