Machine Learning Approaches to SMS Spam Detection

Umamaheswarababu Maddela
Liu-Id: umama339, Course Code: 732A81

Abstract

This project tackles the escalating issue of SMS spam, which poses a significant challenge to both mobile users and service providers. With the widespread adoption of mobile technology globally, the rise in SMS spam has become a major concern. By leveraging machine learning algorithms like the Support Vector Classifier, Random Forest Classifier, and XGBoost Classifier, alongside text mining techniques such as TF-IDF and Word2Vec, the study aims to effectively differentiate between legitimate messages and spam. Utilizing a dataset comprising 3068 messages collected from personal mobile inbox, the research assesses the performance of various models in SMS spam detection. Results highlight the Random Forest Classifier, Support Vector Classifier, and XG Boost Classifier as strong performers with TF-IDF representation, demonstrating high accuracy and ROC-AUC scores. Conversely, the baseline Random Classifier falls short, emphasizing the need for sophisticated models in text classification needs. The study underscores the importance of employing advanced techniques to enhance SMS security and improve user experience in mobile communication realms.

1 Introduction

As communication technology grows, mobile services have become a convenient tool for everyone to use. In 2016, smartphones were owned by fewer than half of the global population. However, by 2020, the number increased to 78.05%. It's projected that by 2025, nearly 87% of mobile users in the US will have smartphones.[1] One of the smartphone applications, SMS (Short Message Service), a communication tool, is favored for its simplicity, with a 160-character limit and direct delivery. It's preferred by 9 out of 10 consumers for business communication and enjoys high engagement rates, making it a rapidly growing marketing channel.[2] 23 billion texts are sent worldwide each day in 2020 and this is expected to continue.[3] Most users don't open business emails daily, unlike text messages, which typically open within minutes, indicating higher open rates for text marketing.[4]

SMS spam refers to the unwanted (junk) messages sent to users' mobile devices without their consent (Almeida et al., 2013). These messages not only cause annoyance but also fill users' inboxes, making it harder to access important communications and pose security threats. To address this issue, the project utilizes machine learning algorithms and text mining techniques such as Support Vector Classifier, Random Forest Classifier, and XGBoost Classifier, along with vectorization methods like TF-IDF and Word2Vec. By analyzing message content and patterns, these models aim to distinguish between legitimate messages and spam, enhancing user experience and security while optimizing engagement with SMS services.

2 Theory

This section covers theory behind machine learning algorithms and text mining techniques like Support Vector Classifier, Random Forest Classifier, and XGBoost Classifier, focusing on their application in SMS spam detection and evaluation using metrics like accuracy and ROC-AUC.

2.1 Machine learning algorithms

2.1.1 Support Vector Classifier (SVC):

The Support Vector Classifier (SVC) (Evgeniou and Pontil, 2001) is a powerful classification algorithm that finds the hyperplane separating data

[1] https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

[2] https://financesonline.com/sms-marketing-statistics/

[3] https://www.forbes.com/sites/forbestechcouncil/2021/01/06/the-past-present-and-future-of-messaging/

[4] https://techjury.net/blog/sms-marketing-statistics/

classes while maximizing the margin. Mathematically, the hyperplane is represented as:

$$f(x) = sign(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b)$$

where $K(x_i, x)$ is the kernel function that computes the inner product in a higher-dimensional space without explicitly mapping the data into that space. The optimization problem to find the optimal hyperplane is formulated as:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i$$

subject to $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$ for $i = 1, 2, ..., n$, where C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. SVC can handle non-linear data using different kernels like linear, polynomial, or radial basis function (RBF) kernels, mapping data into higher-dimensional spaces.

SVC is ideal for spam classification, handling high-dimensional data efficiently, and separating classes distinctly. In SMS spam detection, where features are complex, SVC identifies patterns and distinguishes between legitimate messages and spam. Training involves finding the optimal hyperplane to separate data classes while maximizing the margin. This process enables SVC to generalize from training data, accurately classifying unseen SMS messages, making it ideal for spam detection tasks.

2.1.2 Random Forest Classifier:

Random Forest Classifier (RFC) (Liaw and Wiener, 2001) is a versatile ensemble learning method that constructs many decision trees during training and outputs the mode of the classes as the prediction of individual trees. It operates by creating decision trees on randomly selected data samples and features, thereby preventing overfitting and enhancing robustness. The final prediction is obtained through a voting mechanism, where each tree contributes to the final decision. RFC is suitable for spam classification due to its ability to handle high-dimensional data, deal with noisy and irrelevant features, and mitigate overfitting issues common in other classifiers.

During training, the Random Forest algorithm constructs multiple decision trees independently and in parallel. Each tree is trained on a random subset of the training data and features, known as bootstrapping and random feature selection, respectively. The hyperparameters of RFC, such as the number of trees (n_estimators) and the maximum depth of the trees (max_depth), can be adjusted to optimize performance and prevent overfitting. The training process involves growing each decision tree recursively, splitting nodes based on feature importance and purity measures like Gini impurity or entropy. The resulting ensemble of trees forms a robust classifier capable of accurately distinguishing between spam and legitimate messages.

2.1.3 XG Boost Classifier:

XGBoost (Extreme Gradient Boosting) Classifier (Chen and Guestrin, 2016) is a powerful and scalable machine learning algorithm known for its efficiency and accuracy in various classification tasks, including spam detection. It belongs to the ensemble learning family and works by sequentially building a series of decision trees, each subsequent tree correcting the errors of its predecessors. XGBoost employs a gradient boosting framework, where each tree is grown to minimize a specified loss function, such as binary logistic loss for classification problems.

XGBoost's effectiveness in spam classification stems from its ability to handle high-dimensional data, learn complex patterns, and mitigate issues like overfitting. It leverages regularization techniques to control model complexity and prevent overfitting, allowing it to generalize well to unseen data. Moreover, XGBoost is highly efficient and scalable, making it suitable for large-scale datasets commonly encountered in spam detection tasks.

During training, XGBoost sequentially adds decision trees to the ensemble, with each tree learning from the mistakes of its predecessors. The algorithm optimizes the model parameters by minimizing the specified loss function, typically using techniques like gradient descent. Hyperparameters such as learning rate, tree depth, and regularization parameters can be fine-tuned to optimize performance and prevent overfitting.

2.1.4 Random Classifier:

The "Random Classifier" serves as a fundamental baseline model for comparison against more sophisticated algorithms like Support Vector Classifier, Random Forest Classifier, and XGBoost Classifier in spam detection tasks. Unlike the other algorithms, the Random Classifier assigns labels ran-

domly to incoming data points without considering any underlying patterns or features. Its purpose in the project is twofold: first, it provides a simple benchmark against which the performance of more advanced models can be evaluated, helping to gauge the effectiveness of the proposed approaches. Second, it highlights the necessity of employing more complex algorithms like SVC, Random Forest, and XGBoost to achieve accurate and reliable spam detection. While the Random Classifier lacks the sophistication and predictive power of other models, its inclusion underscores the importance of employing robust machine learning techniques tailored to the specific requirements of SMS spam detection.

2.2 Text representations

Text representations are ways to convert textual data into numerical format for machine learning algorithms to process and analyze.

2.2.1 TF-IDF Vectorization:

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a widely used technique in natural language processing and information retrieval for converting text documents into numerical vectors. It aims to capture the importance of a word in a document relative to a corpus of documents (Salton and Buckley, 1988).

Mathematically, the TF-IDF weight of a term t in a document d is calculated as:

$$TF - IDF(t, d) = TF(t, d) \times IDF(t)$$

Where: - $TF(t, d)$ represents the term frequency of t in d, i.e., the number of times t appears in d. - $IDF(t)$ is the inverse document frequency of t, computed as $\log\left(\frac{N}{DF(t)}\right)$, where N is the total number of documents in the corpus, and $DF(t)$ is the number of documents containing term t.

The TF-IDF vectors created for each document represent the importance of words in distinguishing documents from each other. Words that occur frequently in a particular document but are rare across the corpus are given higher TF-IDF weights.

TF-IDF vectorization is particularly appropriate for spam classification because it helps identify discriminative features that characterize spam messages, such as specific keywords or patterns commonly found in spam but not in legitimate messages. By capturing the unique characteristics of spam messages, TF-IDF vectors enable machine learning models to effectively distinguish between spam and non-spam content.

2.2.2 Word2Vec:

Word2Vec is a popular technique used to represent words as continuous vectors in a high-dimensional space, capturing semantic relationships between words (Mikolov et al., 2013). It is based on the distributional hypothesis, which posits that words appearing in similar contexts are semantically related.

The Word2Vec model consists of two architectures: Continuous Bag of Words (CBOW) and Skip-gram. In the CBOW architecture, the model predicts the target word given its context words, while in Skip-gram, it predicts context words given the target word.

Mathematically, the objective of Word2Vec is to maximize the probability of context words given the target word or vice versa. This is typically achieved using a neural network with a single hidden layer. The network learns to adjust word vectors during training to improve prediction accuracy.

Once trained, Word2Vec embeddings capture semantic relationships between words. Words with similar meanings are represented by vectors that are close together in the embedding space. This makes Word2Vec embeddings useful for various natural language processing tasks, including sentiment analysis, text classification, and information retrieval.

In the context of spam classification, Word2Vec embeddings provide a powerful representation of text data, enabling machine learning models to capture nuanced semantic information present in spam and non-spam messages. By leveraging semantic similarities between words, Word2Vec embeddings help identify subtle patterns and linguistic cues indicative of spam content.

The hyperparameters in Word2Vec include the dimensionality of the word vectors, the window size (the maximum distance between the current and predicted word within a sentence), the training algorithm (either CBOW or Skip-gram), and the number of iterations over the corpus during training.

2.3 Evaluation Metrics

2.3.1 Accuracy:

Accuracy is a common evaluation metric used to measure the overall correctness of a classification model. It represents the ratio of correctly classified

instances to the total number of instances in the dataset (Hossin and M.N, 2015). Mathematically, accuracy is calculated as:

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions}$$

While accuracy provides a straightforward measure of classification performance, it may not be suitable for imbalanced datasets where one class dominates the other (Fawcett, 2006). In such cases, accuracy alone can be misleading and may not reflect the true performance of the model, especially when the minority class is of interest.

2.3.2 ROC-AUC (Receiver Operating Characteristic - Area Under the Curve):

ROC-AUC is a performance metric commonly used in binary classification tasks, particularly when dealing with imbalanced datasets (Fawcett, 2006). The ROC curve is a graphical representation of the true positive rate (Sensitivity) against the false positive rate (1 - Specificity) for different threshold values. The area under the ROC curve (AUC) quantifies the model's ability to distinguish between the positive and negative classes. A perfect classifier would have an AUC of 1, while a random classifier would have an AUC of 0.5.

Mathematically, ROC-AUC is computed by integrating the area under the ROC curve:

$$ROC - AUC = \int_0^1 TPR\, d(FPR)$$

ROC-AUC is preferred over accuracy, especially in scenarios where class imbalances exist, as it provides a more comprehensive assessment of the model's performance across different thresholds. It is less sensitive to class distributions and provides insights into the classifier's ability to rank instances correctly.

Definitions:

- TP (True Positive): Correctly classified positive instances.
- FP (False Positive): Incorrectly classified negative instances.
- TN (True Negative): Correctly classified negative instances.
- FN (False Negative): Incorrectly classified positive instances.

True Positive Rate (TPR) and False Positive Rate (FPR):

$$TPR = \frac{TP}{TP + FN}; FPR = \frac{FP}{FP + TN}$$

3 Data

The dataset used in this project was created specifically for this purpose. It was gathered from my personal mobile inbox, comprising messages from various sources including family, friends, insurance companies, e-commerce platforms, and banks, covering personal, business, and marketing communications. The dataset consists of 3068 messages, including those from family and friends with their permission. It's structured in CSV format with two columns: "Text" for message content and "Spam" indicating whether the message is spam ('1') or not ('0'). Each message is manually labeled after carefully reviewing and categorizing them according to predefined criteria outlined in (Almeida et al., 2013). The dataset includes 2669 non-spam messages and 399 spam messages which indicate the data is imbalanced. Common examples of spam message words include 'cashback,' 'save,' 'discount,' 'offer,' 'sale,' 'free,' 'lottery,' etc., identified during content analysis.

To protect privacy, sensitive information such as social security numbers, mobile numbers, email addresses, credit card numbers, bank account numbers, and one-time passwords (OTP) were removed from the dataset using the Python package 're'.

The preprocessing phase involved several steps to clean and prepare the dataset for analysis. Initially, all text was converted to lowercase using the Python package 'nltk' to ensure consistency in the dataset. Punctuations and special characters were removed to focus solely on the textual content. Next, the text was tokenized into individual words or tokens for further processing. Stop words, such as "the" and "is", were filtered out using 'nltk' to eliminate common words that don't contribute significantly to the analysis. Lastly, lemmatization was applied to reduce words to their base or dictionary form, enhancing the consistency and accuracy of the dataset for analysis purposes.

4 Method

This section explains how the problem was addressed using methods on Google Colab. Preprocessing and machine learning algorithms were applied, benefiting from Colab's resources.

4.1 Text Representations

4.1.1 TF-IDF Vectorization:

The dataset was preprocessed to remove noise and standardize the text format. TF-IDF vectorization was applied using the TfidfVectorizer from scikit-learn to convert text into numerical features. Each message was transformed into a numerical vector representation using TF-IDF.

4.1.2 Word Embeddings:

Word2Vec model was trained on the preprocessed text data using the Word2Vec class from the Gensim library. A document vector was generated by averaging the word vectors of each document. The Word2Vec model provided a dense vector representation for each message in the dataset. The hyperparameters of the Word2Vec model, such as vector size, window size, and minimum word count, were set to default values. Further tuning of these hyperparameters could potentially improve the performance of the Word2Vec model.

4.2 Data Splitting:

The dataset was split into training(80%) and test(20%) sets using stratified sampling to ensure a balanced representation of both classes (spam and non-spam) in each set.

4.3 Model Training and Evaluation

4.3.1 Dummy Classifier:

Dummy classifiers were trained using both TF-IDF vectorization and Word2Vec embeddings. They served as baselines for evaluating the performance of more sophisticated models.

4.3.2 Support Vector Classifier (SVC):

Grid search with cross-validation was used to optimize hyperparameters for SVC with both TF-IDF and Word2Vec representations. The hyperparameters explored included the choice of kernel (linear or radial basis function) and the regularization parameter (C). The best SVC models were selected based on the highest area under the ROC curve (AUC) score. Performance metrics such as accuracy, AUC score, and classification report were computed for each SVC model.

4.3.3 Random Forest Classifier:

Random Forest models were trained and optimized using grid search with cross-validation for both TF-IDF and Word2Vec representations. The hyperparameters considered included the number of trees (n_estimators), maximum depth of each tree (max_depth), minimum number of samples required to split an internal node (min_samples_split), and the minimum number of samples required to be at a leaf node (min_samples_leaf). The best Random Forest models were selected based on AUC score. Performance metrics including accuracy, AUC score, and classification report were calculated.

4.3.4 XGBoost Classifier:

XGBoost models were trained and hyperparameters were tuned using grid search with cross-validation for both TF-IDF and Word2Vec representations. The hyperparameters explored included the learning rate, number of estimators (n_estimators), maximum tree depth (max_depth), subsample ratio of the training instances (subsample), column subsample ratio of the features (colsample_bytree), and gamma, which is the minimum loss reduction required to make a further partition on a leaf node of the tree. The best XGBoost models were selected based on AUC score. Performance metrics such as accuracy, AUC score, and classification report were generated for each XGBoost model.

This methodology ensured a comprehensive exploration of text representations and machine learning algorithms for spam classification, with rigorous evaluation using multiple performance metrics.

5 Results

Here are the results of different classifiers with their respective text representations, accuracy scores, and ROC-AUC scores presented in Table 1.

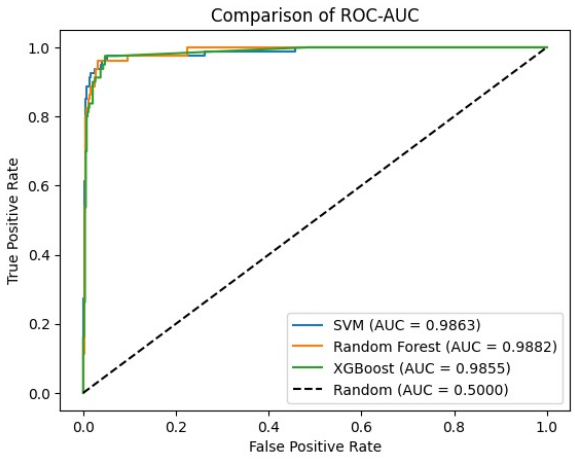

Figure 1: ROC-AUC Plot

The three models, namely the Random Forest Classifier, Support Vector Classifier (SVC),

Classifier	Text representation	Accuracy	ROC-AUC
Random Classifier (base)	TF-IDF/Word2Vec	0.4902	0.5000
SVC	TF-IDF	0.9577	0.9863
SVC	Word2Vec	0.9007	0.9508
Random Forest	TF-IDF	0.9642	0.9882
Random Forest	Word2Vec	0.9397	0.9739
XG Boost	TF-IDF	0.9658	0.9855
XG Boost	Word2Vec	0.9479	0.9797

Table 1: The results of different classifiers with their respective text representations

and XG Boost Classifier, demonstrated strong performance with TF-IDF representation, exhibiting slight variations in accuracy and ROC-AUC scores. The Random Forest Classifier emerged as the top performer, achieving an accuracy of 96.42% and an ROC-AUC score of 98.82%, signifying its effectiveness in distinguishing between spam and non-spam messages using TF-IDF features. Similarly, the SVC exhibited strong performance with an accuracy of 95.77% and an ROC-AUC score of 98.63%. The XG Boost Classifier also demonstrated competitive performance, achieving an accuracy of 96.58% and an ROC-AUC score of 98.55% with TF-IDF representation. In contrast, the Random Classifier, serving as the baseline model, yielded poor results with an accuracy of 49.02% and an ROC-AUC score of 50%, highlighting the necessity of employing more sophisticated models for effective text classification tasks. Figure 1 shows the visual representation of ROC-AUC plots of all the models with TF-IDF text representation. These observations underscore the robustness of TF-IDF as a text representation method and emphasize the importance of model selection in achieving optimal performance in text classification applications.

6 Discussion

In this project, the objective was to develop machine learning models capable of distinguishing between spam and legitimate SMS messages by employing various machine learning algorithms in conjunction with two distinct text representations: TF-IDF and Word2Vec. Interestingly, all models exhibited superior performance with TF-IDF compared to Word2Vec. This outcome could be attributed to the nature of SMS messages, which often feature short, informal, and domain-specific language. TF-IDF, known for its ability to capture unique vocabulary and characteristics, appears to be more adept at handling the nuances present in SMS spam messages. On the other hand, Word2Vec, relying on dense vector representations learned from context, may struggle with the limited data available in SMS datasets, leading to less informative representations (Joseph and Yerima, 2022).

When comparing these results to existing literature, it is worth noting that Singh et al. (2022) achieved 98.70% accuracy using a linear SVM countVectorizer with a larger dataset. Despite the dataset is small in this project, the models demonstrated competitive performance, particularly in ROC-AUC scores, indicating their efficacy in SMS spam detection.

The project's limitations include a dataset sourced solely from an individual's mobile inbox, possibly limiting its representation of real-world SMS diversity. It is also constrained to the English language, potentially restricting its applicability across multilingual contexts. Furthermore, text conversion to lowercase may result in information loss, particularly affecting languages reliant on case sensitivity.

Future work needs expanding the dataset's size and diversity to enhance model robustness. Incorporating multiple languages could broaden the study's applicability, while exploring advanced neural networks may improve classification accuracy. Additionally, incorporating diverse text representations beyond TF-IDF and Word2Vec could enrich feature representations and enhance model performance.

7 Conclusion

In summarizing the analysis, the exploration delved into the domain of SMS spam detection utilizing machine learning models and diverse text representations. It was observed that the Random Forest Classifier, Support Vector Classifier, and XG Boost

Classifier exhibited strong performance, particularly when combined with TF-IDF representation. These models showed high accuracy and ROC-AUC scores, indicating their ability to effectively distinguish between spam and legitimate messages. By contrasting these models with a baseline Random Classifier, the importance of employing advanced algorithms for accurate classification was highlighted.

In assessing the extent of problem resolution, this project has made significant progress in addressing the issue of SMS spam. While some limitations remained, such as dataset size and language constraints, the models performed comparably well to established benchmarks in the literature. This suggests a meaningful advancement in tackling SMS spam, which enhances user experience and strengthens mobile communication security.

Furthermore, the project underscores the importance of continuous innovation and improvement in machine learning applications. It highlights the need for diverse datasets and considerations of different languages to improve model reliability and applicability. Looking ahead, further research can explore advanced neural network architectures and alternative text representations to advance SMS spam detection and create a safer mobile communication environment.

Ethics Statement

In this project, I have made sure to handle the data with care and responsibility. I have not misused the data or shared it with anyone publicly. I've taken steps to protect the privacy of the people whose data is included. Any personal information has been kept private. Respecting people's privacy and being honest with how data is used is important to me.

References

Tiago Almeida, Jose Gomez Hidalgo, and Akebo Yamakami. 2013. Contributions to the study of sms spam filtering: New collection and results (preprint).

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. pages 785–794.

Theodoros Evgeniou and Massimiliano Pontil. 2001. Support vector machines: Theory and applications. volume 2049, pages 249–257.

Tom Fawcett. 2006. Introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874.

Mohammad Hossin and Sulaiman M.N. 2015. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining Knowledge Management Process*, 5:01–11.

Prashob Joseph and Suleiman Y. Yerima. 2022. A comparative study of word embedding techniques for sms spam detection. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 149–155.

Andy Liaw and Matthew Wiener. 2001. Classification and regression by randomforest. *Forest*, 23.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Tarandeep Singh, Tushar Anupam Kumar, and Prashant Giridhar Shambharkar. 2022. Enhancing spam detection on sms performance using several machine learning classification models. In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1472–1478.