**SMS Spam Classification: A Comparative Analysis of Machine Learning and Deep Learning Techniques**

**Author: Balaita Radu Ionut**

**Mentor: radu-ionut.balaita@student.tuiasi.ro**

**Summary**

**This paper investigates the effectiveness of various machine learning and deep learning models for classifying SMS messages as "ham" (legitimate) or "spam." Utilizing the SMS Spam Collection dataset, we employ extensive preprocessing and feature extraction techniques, including CountVectorizer and TF-IDF Vectorizer. Ten traditional models, such as Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), and Random Forest (RF), are compared alongside a Bidirectional Long Short-Term Memory (BiLSTM) deep learning model. Performance is evaluated based on accuracy, precision, recall, and F1-score. The BiLSTM model demonstrates competitive performance with top traditional models like SVC, RF, and Extra Trees Classifier (ETC), which all exhibit high accuracy and precision. Notably, Naive Bayes achieves perfect precision but lower recall. These findings offer valuable insights for selecting optimal models in practical spam detection applications.**

**Introduction**

**The ubiquity of mobile devices and SMS messaging has led to a significant increase in unsolicited and potentially harmful messages, commonly referred to as spam. SMS spam not only degrades user experience but also poses serious security threats, such as phishing and malware distribution. Consequently, the development of robust and efficient SMS spam classification systems is of paramount importance.**

**This paper aims to explore and evaluate various machine learning and deep learning techniques for SMS spam classification. The primary objective is to develop a model that can accurately distinguish between legitimate (ham) and spam messages. This research will provide insights into the effectiveness of different algorithms for this specific task and offer a comparative analysis to guide the selection of the most suitable approach for practical implementation. This work is relevant because it will help refine our model selection process for future projects.**

**Dataset and Methodology**

**Dataset**

**The SMS Spam Collection dataset, a widely recognized public dataset for SMS spam research, is employed in this study. It comprises 5,574 SMS messages in English, each labeled as either "ham" (legitimate) or "spam."**

**Data Preprocessing**

**The raw text data undergoes several preprocessing steps to prepare it for model training:**

**1. Data Cleaning: The dataset is initially cleaned by removing any unnecessary columns and renaming the relevant columns to "target" (for labels) and "text" (for message content).**

**2. Label Encoding: Categorical labels ("ham" and "spam") are converted into numerical representations (0 and 1, respectively) using LabelEncoder.**

**3. Duplicate Removal: Duplicate messages are removed to prevent bias in the model training and evaluation process.**

**4. Text Transformation:**

**• All text is converted to lowercase.**

**• Tokenization is performed using the NLTK library to split the text into individual words.**

**• Special characters, stop words (common words like "the," "a," "is"), and punctuation are removed.**

**• Stemming is applied using the Porter Stemmer algorithm to reduce words to their root form.**

**Feature Extraction**

**Two feature extraction techniques are used to transform the preprocessed text into numerical vectors:**

**1. CountVectorizer: This method creates a vocabulary of all unique words in the corpus and represents each document as a vector of word counts.**

**2. TF-IDF Vectorizer: TF-IDF assigns weights to words based on their frequency within a document and across the entire corpus.**

**Traditional Machine Learning Models**

**The paper explored ten distinct traditional machine learning models for SMS spam classification, each with its own approach to learning from the data. Let's break them down a little further:**

**1. Support Vector Classifier (SVC): At its core, SVC seeks to establish an optimal boundary—a hyperplane—that best divides spam from ham messages within a high-dimensional space. The goal is to maximize the margin between the two classes, leading to robust classification. SVC uses something called kernels to handle non-linear relationships between data points. In this study a sigmoid kernel with a specific gamma value was used, influencing how the model forms the decision boundary. The model aims to be effective even when dealing with complex data patterns.**

**2. K-Nearest Neighbors (KNN): KNN operates on the principle that data points that are close to each other are likely to belong to the same class. It classifies a new message based on the majority class among its 'k' nearest neighbors in the feature space, determined by a distance metric. KNN is a simple model, without any explicit training phase, however, this also implies the model may need a large dataset to perform well, and is sensitive to the choice of the value for k.**

**3. Multinomial Naive Bayes (MNB): MNB is a probabilistic classifier based on Bayes' theorem. It calculates the probability of a message being spam or ham based on the frequency of words and relies on a strong independence assumption between features (words). Despite its simplicity, it is often effective for text classification tasks.**

**4. Decision Tree Classifier (DT): A decision tree model creates a tree-like structure to represent decision rules. Each internal node of the tree corresponds to a feature, each branch to a decision based on that feature, and each leaf node contains the classification outcome. The model builds a series of cascading questions based on word features, and the answers determine if a message is classified as ham or spam. Here, the tree was constrained to a maximum depth of five, possibly to avoid overfitting.**

**5. Logistic Regression (LR): This model estimates the probability of a binary outcome (spam or ham) using a logistic function. It is a linear model that finds a linear decision boundary in the feature space. The model utilizes a 'liblinear' solver with an 'l1' regularization penalty, which helps prevent overfitting by shrinking less important feature coefficients towards zero, which could help to improve model performance.**

**6. Random Forest Classifier (RF): This is an ensemble method that constructs multiple decision trees during training. For each decision, a different subset of features is used, and a random subset of the training data is used, and then the final prediction is a combination of the results of each tree. It leverages the combination of many trees in a random way to achieve more robust classifications. This model was built using 50 trees and a particular random state to allow for reproducibility of results.**

**7. AdaBoost Classifier (ABC): AdaBoost is another ensemble technique that builds a strong classifier by combining several weak ones. It assigns higher weights to misclassified instances in each iteration, making the model pay more attention to instances it struggles with. The version used here was trained using 50 estimators, and a random state was set for consistency.**

**8. Bagging Classifier (BC): Bagging involves training multiple versions of the same model (typically a decision tree), each on different subsets of the training data that are sampled with replacement. The final prediction of a new example is the average prediction of all the base models. It's a technique to reduce variance in model predictions and enhance its stability, with the model being initialized to 50 base models here.**

**9. Extra Trees Classifier (ETC): Extra Trees is similar to Random Forest but introduces even more randomness in the way that decision trees are built. Like Random Forest, it builds many decision trees, but the way in which the feature split is chosen at each node is random, which can contribute to greater diversity in the trees and a more effective model. 50 trees were built, and a random state was fixed.**

**10. Gradient Boosting Decision Tree (GBDT): GBDT builds an ensemble of decision trees sequentially, with each new tree being created to correct the errors of the previous ones. It works iteratively to minimize errors. 50 estimators were used and a random seed was set.**

**Bidirectional LSTM (BiLSTM) Model**

**The BiLSTM model is a deep learning approach that leverages the power of recurrent neural networks to understand the sequential nature of text. It has a more complex architecture compared to traditional models, which includes the following:**

**• Embedding Layer: This layer transforms each word into a vector of 100 dimensions. It learns to capture the semantic context of each word, mapping words with similar meanings to similar vectors. It essentially converts discrete words into a continuous vector space.**

**• Bidirectional LSTM Layer: This is where the model processes the sequence of word embeddings in both forward and backward directions. In the forward direction, it looks at each word in context with the words coming before it in the sentence, and in the backward direction, each word is looked at in context of the words coming after. This approach allows the model to better capture the context of each word within the whole sentence. 128 units are used in each direction, and recurrent dropout (0.2) helps to prevent overfitting by dropping connections randomly during training.**

**• Dropout Layer: Following the LSTM layer, the dropout layer, with a rate of 0.2, randomly deactivates a fraction of neurons. This further reduces overfitting and makes the model more robust.**

**• Dense Output Layer: The final layer is a fully connected layer with a single output unit and a sigmoid activation function. The sigmoid activation constrains the output to a range between 0 and 1, which can be interpreted as the probability that a given SMS message is spam.**

**The model was trained using the rmsprop optimizer, which adjusts the learning rate for each parameter in a more sophisticated way than basic optimizers, with binary\_crossentropy as the loss function, which measures the difference between the predicted probabilities and the true labels. The performance was monitored by the metric accuracy. The model was trained using 10 epochs, and using a batch size of 64, and with 20% of the training data being set aside for validation. The model training involves the iterative process of refining its parameters to minimize the loss function, while validation on a held-out set helps to monitor the model performance on unseen data to avoid overfitting. The model prediction output is a value between 0 and 1. These probabilities are converted to final binary classifications using a threshold of 0.5: if greater than 0.5 the SMS message is predicted to be spam, and if less than or equal to 0.5, the SMS message is classified as ham.**

**Evaluation Metrics**

**Model performance is assessed using:**

**1. Accuracy: Overall correctness of the classification, calculated as (True Positives + True Negatives) / Total Predictions.**

**2. Precision: Proportion of true positives among the predicted positives, calculated as True Positives / (True Positives + False Positives). It measures how many of the messages classified as spam are actually spam.**

**3. Recall: Proportion of true positives among the actual positives, calculated as True Positives / (True Positives + False Negatives). It measures how many of the actual spam messages were correctly identified.**

**4. F1-Score: Harmonic mean of precision and recall, providing a balanced measure of performance, calculated as 2 \* (Precision \* Recall) / (Precision + Recall)**

**Results and Discussion**

**Traditional Machine Learning Models**

**The table below summarizes the performance of the traditional machine learning models:**

**Model Accuracy Precision Recall F1-Score**

**SVC 97.58% 97.48% 84.06% 90.27%**

**KNN 90.52% 100.00% 28.99% 44.94%**

**NB 97.10% 100.00% 78.26% 87.80%**

**DT 92.94% 82.83% 59.42% 69.20%**

**LR 95.65% 96.97% 69.57% 81.01%**

**RF 97.68% 97.50% 84.78% 90.70%**

**ADA 92.36% 83.91% 52.90% 64.89%**

**Bgc 95.94% 86.92% 81.88% 84.33%**

**ETC 97.78% 96.75% 86.23% 91.19%**

**GBDT 95.07% 93.07% 68.12% 78.66%**

**BiLSTM Model**

**The BiLSTM model achieved the following results:**

**precision recall f1-score support**

**Ham 0.98 1.00 0.99 896**

**Spam 0.97 0.88 0.92 138**

**accuracy 0.98 1034**

**macro avg 0.98 0.94 0.96 1034**

**weighted avg 0.98 0.98 0.98 1034**

**Discussion**

**The traditional models displayed a range of performance. Extra Trees Classifier (ETC) and Random Forest (RF) stood out with the highest accuracies (97.78% and 97.68% respectively) and strong F1-scores (91.19% and 90.70%). Their ensemble nature, combining multiple decision trees, likely contributed to their robustness and ability to generalize well. Support Vector Classifier (SVC) also performed exceptionally well with an accuracy of 97.58% and an F1-score of 90.27%, demonstrating its effectiveness in finding an optimal hyperplane for separating the classes.**

**Naive Bayes (NB) and K-Nearest Neighbors (KNN) achieved perfect precision (100%), meaning they never incorrectly classified a ham message as spam. However, their recall values differed significantly. NB's recall of 78.26% indicated it missed a portion of actual spam messages, while KNN's very low recall of 28.99% suggested it misclassified the majority of spam as ham, making it less practical despite its precision. The low recall of KNN may be because of the small number of spam examples in the dataset, which makes it difficult for the model to correctly classify spam SMS messages.**

**Logistic Regression (LR), Gradient Boosting Decision Tree (GBDT), and Bagging Classifier provided a reasonable balance between precision and recall, with F1-scores of 81.01%, 78.66%, and 84.33% respectively. Decision Tree (DT) and AdaBoost had lower F1-scores (69.20% and 64.89%), suggesting they might not be the best choices for this task if a balance between false positives and false negatives is crucial.**

**The BiLSTM model achieved an accuracy of 97.58%, a precision of 96.74%, a recall of 84.78%, and an F1-score of 90.38%. These results are competitive with the top-performing traditional models, such as SVC, RF, and ETC. This demonstrates the potential of deep learning techniques, particularly recurrent neural networks, in effectively capturing the sequential information in text data for spam classification. The ability of BiLSTMs to process text in both directions helps them understand the context of words more comprehensively, leading to improved classification accuracy. The fact that the BiLSTM has a very good performance compared to some of the traditional models is a positive result and may indicate that the BiLSTM architecture is more suitable to the type of data we are working with.**

**Conclusions**

**This study compared ten traditional machine learning models and a BiLSTM model for SMS spam classification, using the SMS Spam Collection dataset and evaluating performance with accuracy, precision, recall, and F1-score.**

**Top performers included Extra Trees Classifier (ETC), Random Forest (RF), and Support Vector Classifier (SVC), demonstrating high accuracy and strong F1-scores. Naive Bayes (NB) achieved perfect precision but lower recall, while K-Nearest Neighbors (KNN) had very low recall despite perfect precision. The BiLSTM model performed competitively with the best traditional models, highlighting the potential of deep learning for this task.**

**Ultimately, the optimal model choice depends on specific application needs and the desired balance between precision and recall. Traditional models like ETC, RF, and SVC offer excellent performance, while BiLSTM showcases the power of deep learning for text data. Future work should investigate advanced deep learning architectures, additional features, and class imbalance techniques to further improve spam detection systems. These findings provide valuable insights for developing more effective spam filtering solutions.**