

# Mobile Money SMS Fraud Detection

Stanford CS 229 Final Project Report

**Josh Nkoy x Arafat Mohammed**

Department of Computer Science

Stanford University

joshnkoy@stanford.edu x arafatm@stanford.edu

## Abstract

In recent times, the growing decentralization and ubiquitization of financial technology has driven economic growth in emerging economies, especially in sub-Saharan Africa. Key to this spread of greater financial freedom and expediency in this region has been the use of mobile money transfer (MMT) services, which allow greater speed, volume, and anonymity of transactions on an individual basis than traditional money wires. However, with such power comes much possibility for fraud, especially preying on vulnerable populations using one or more text messages preceding a call. In this project, we present the Machine learning model best suited for MMT fraud detection in text messages.

## 1 Introduction

With more than 2 billion persons and 200 million small businesses in emerging economies worldwide lacking access to formal savings and checking accounts — but 68% of *that* population having access to a mobile phone — MMT has nearly closed the gap of easy transactional capability with nearly 1.1 billion MMT accounts operational worldwide. Of that staggering amount, the African continent is home to over half of those accounts, to such an extent that MMT has over 50 times the reach of banking services. With wider adoption of digitalized finance, emerging economies in Africa and elsewhere could collectively add 6% more GDP — that is, \$3.7 trillion. It is clear that MMT has and will continue to drive the way people in emerging economies circulate their money.

However, considering the relative ease of MMT compared to banking, MMT has also been exploited for not-so-good purposes; all things considered, it is far easier (without proper regulation) for people to launder money or defraud vulnerable people. A 2020 Interpol report lists various types of new fraud that are possible in MMT systems — email-like advance fee scams, fake charity drives, using fake employee details, and most importantly for our purposes SMS scams. Fortunately, we can wield the power of machine learning to try and make predictive systems that can pick up on such fraud on the fly, keeping more money legitimately in the MMT flow for the people who need it most. Given the large corpus of research on classification of text, we hope to uncover which of a machine learning methods can robustly identify fraudulent intent within SMS messages, measuring metrics for accuracy, precision, and other dynamics along the way.

## 2 Related Work

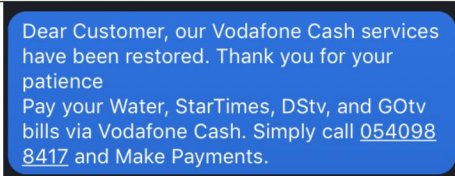
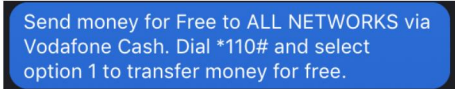
There has been ample work on applying machine learning algorithms, specifically ones most useful for spam-ham classification, to fraud detection in the mobile money transfer sphere. One of the most comprehensive comparisons of classification techniques for fraud is in a paper from Botchey, Qin, and Hughes-Lartey [4]; this work compared the efficiency of support vector machines, gradient-boosted decision trees, and naive Bayes algorithms in classification, with added model construction tactics over the dataset (such as oversampling and undersampling) to overcome the inherent imbalance in spam-ham data — where well over 90% of the data are spam instances. Another project from Magagula [5] compared two plain ReLU neural network classifiers over similar datasets. Yet another

project from Bandyopadhyay and Dutta [6] analyzed the efficacy of recurrent neural networks in classifying fraudulent messages, achieving an accuracy of 99% (and F1 score of 0.99) in the process. Additionally, Adedoyin [7,8] curated a fraud detection system model consisting of data preprocessing, then clustering, then feature extraction, and finally case-based reasoning classification achieved by  $k$ -nearest neighbor variants. Our work aims to be a fairly wider survey of multiple techniques that are covered in these studies, with more emphasis on feature extraction and the effect it has on efficiency over a plethora of models, be it probabilistic, plain neural network, or deep learning models.

### 3 Dataset

Due to the intrinsically private nature of financial transactions, gathering real data especially in the emerging mobile money transactions domain can be hard and no existing datasets was found. Consequently, about 50 real mobile money SMS messages were manually gathered and tagged as either ham (legitimate) or spam (fraud). A typical example of a MMT fraud/spam message is show in Table 1 below. This data was combined with two SMS Spam Collection Datasets sourced from Kaggle with each having about 5000 messages, in English, each and tagged according being ham or spam. A total of about 11,000 messages was used for this project with 80 percent for training and 20 percent for testing. A distribution of the messages is shown below in figure 1.

Table 1: Sample of Real Mobile Money related SMS message

Fraud (spam)	Legitimate (ham)
	

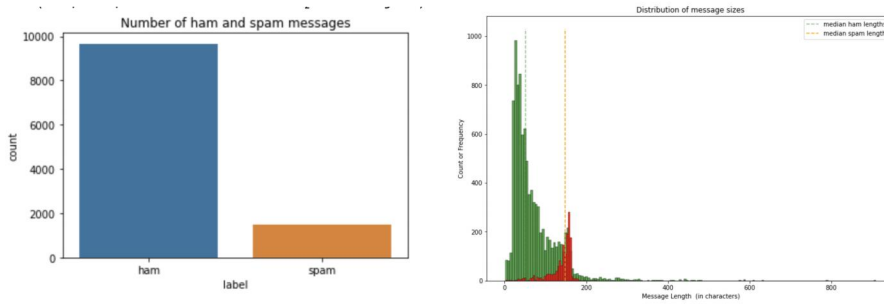


Figure 1.

### 4 Features

The purpose of extraction is to represent the messages in a meaningful number space in order to be used by the Machine Learning Models. Four feature extraction methods were employed namely: Term Frequency-Inverse Dense Frequency(TF— IDF), CountVectorizer, Tokenizer, and Transformer Based feature extractor. The last two were used for the deep learning models presented in this paper. Before extracting the features, the messages are normalized by removing punctuation and all stop words and converted to lower case.

#### 4.1 TF-IDF

TF-IDF represents the collection of messages by assigning a score for each word in the using the formula below:

$$\text{Term frequency: } \text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

Inverse document frequency:  $\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D: t \in d\}|}$$

t: represent the specific term or word, d: specific document or sentence, N: total number of documents in the corpus.

## 4.2 CountVectorizer

The CountVectorizer represents the collection of sentences in a matrix space of token/word counts.

## 4.3 Tokenizer

A keras Tokenizer transforms each word into the collection of sentences to a sequence of integers using word indexing.

## 4.4 DistillBert Based feature extractor

A pretrained transformer based model, DistilBert, was used to obtain an embedding feature vector from the collection of messages.

## 5 Methods

Our project benchmarked learning over our processed data with the following models: logistic regression, (multinomial) naive Bayes classification, support vector machines, k-nearest neighbors algorithm, random forest classifier, LSTM recurrent neural networks, convolutional neural networks, and a combination of the latter two. Over all our following examples, let our dataset be denoted by  $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$ , with each  $x$  representing the messages and  $y$  representing its spam or ham label.

**Logistic regression** assumes that our **hypotheses** for predicting the (probability of the) label  $y^{(i)} \in [0, 1]$  (with either 0 being ham and 1 being spam) for the SMS message data  $x^{(i)}$  is related through the parameters  $\theta$  in the **logistic/sigmoid function**  $g$  such that

$$h_{\theta}(x^{(i)}) = g(\theta^T x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}.$$

This generates a linear decision boundary at  $y = 0.5$  that allows us to classify each message. Over a full each iteration of the algorithm, we utilize that  $p(y^{(i)} = 1|x^{(i)}; \theta) = h_{\theta}(x^{(i)})$  and that  $p(y^{(i)} = 0|x^{(i)}; \theta) = 1 - p(y^{(i)} = 1|x^{(i)}; \theta) = 1 - h_{\theta}(x^{(i)})$  to obtain the following objective function to maximize:

$$J(\theta) = \sum_{i=1}^n \left( y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right).$$

A **multinomial naive Bayes classifier** first embeds each SMS message  $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)})$  as a vector of  $d$  words, with each word  $x_j^{(i)} \in \{1, \dots, |V|\}$  taking a value representing one of the  $|V|$  possible words in the whole dataset-wide vocabulary. As such, the overall probability for an SMS message would be  $p(y^{(i)}) \prod_{j=1}^d p(x_j^{(i)}|y^{(i)})$ , with  $y^{(i)}$  being a Bernoulli variable for the label and  $x^{(i)}|y^{(i)}$  being a multinomial variable over the message given the label. Letting our respective parameters for any word  $j$  be  $\theta = \{\phi_y, \phi_{k|y=1}, \phi_{k|y=0}\}$ , such that  $\phi_y = p(y)$ ,  $\phi_{k|y=1} = p(x_j = k|y = 1)$ , and  $\phi_{k|y=0} = p(x_j = k|y = 0)$ , we obtain the following likelihood function to maximize:

$$\begin{aligned} \mathcal{L}(\theta) &= \prod_{i=1}^n p(x^{(i)}, y^{(i)}; \theta) \\ &= \prod_{i=1}^n \left( p(y^{(i)}; \phi_y) \prod_{j=1}^d p(x_j^{(i)}|y^{(i)}; \phi_{k|y=0}, \phi_{k|y=1}) \right). \end{aligned}$$

**Support vector machines** focus not only on finding a decision boundary but also finding one that increases the **margin** of as many data points from that decision boundary as possible — those with a smaller margin from the boundary being harder to predict than those farther away. This means that we parametrize a linear classifier with slope  $w$  and intercept  $b$  such that  $h_{w,b}(x) = g(w^T x + b)$ , where  $g$  is the sign function. We then find the optimal margin classifier by solving the constraints for  $i \in \{1, \dots, n\}$  that

$$\min_{w,b} \frac{1}{2} \|w\|^2 \text{ s.t. } y^{(i)} (w^T x^{(i)} + b) \geq 1.$$

Simply put, classification with the  **$k$ -nearest neighbors algorithm** uses some distance metric to group points into  $k$  clusters, based on the minimal distance achieved from other points. The algorithm, with  $k = 2$  and using the Euclidean distance  $\|\cdot\|$  as we did, is as follows. We first pick a data point  $(x^{(i)}, y^{(i)})$ ; then iterating over each point in the dataset  $(x^{(k)}, y^{(k)})$  we find and store each distance  $d_{k,i} = \|(x^{(k)}, y^{(k)}) - (x^{(i)}, y^{(i)})\|$ . We then pick the smallest  $k = 2$  distances and then return their mode.

**Random forest classifiers** generate via bootstrap a set number (we used 100) of **decision trees** that take in a sequence of points  $\{(x^{(i)}, y^{(i)})\}$  and decides based on its features which of the two classes, ham or spam, a message will belong to. The key idea of this algorithm is that multiple trees acting in committee, with each of them deciding classification via certain features, is better than just one; bootstrapping over the data will allow us to feed each tree in the committee different combinations of data.

Recurrent neural networks have long been shown to be effective at natural language classification tasks, in our case taking in as input one message  $x^{(i)}$  and concluding if the output  $p(y^{(i)})$  gives us a spam or ham prediction. The **long short-term memory (LSTM)** unit, comprised of a cell with input, output, and forget gates, is no exception. Given the numbers of input features  $d$  and hidden units  $h$ , the variables of LSTM units at time  $t$  involve an input vector  $x_t \in \mathbb{R}^d$ , activation vectors for the input/output/forget gates  $i_t, o_t, f_t \in \mathbb{R}^h$ , cell state and cell input activation vectors  $c_t, \tilde{c}_t \in \mathbb{R}^h$ , the hidden state or output vector  $h_t \in \mathbb{R}^h$ , and weight and bias parameters  $W_q \in \mathbb{R}^{h \times d}$ ,  $U_q \in \mathbb{R}^{h \times h}$ , and  $b_q \in \mathbb{R}^h$  for each gate type  $q$ . This gives the following set of equations for forward pass, where  $\odot$  is the Hadamard element-wise product and  $\sigma$  is the sigmoid function:

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t &= o_t \odot \tanh(f_t). \end{aligned}$$

This model will be very strong at determining spam or ham based on long-term dependencies. The parameters for the LSTM model we used are below in Figure 2 on the left.

**Convolutional neural networks (CNNs)** have also been shown to be effective for natural language classification, taking the input of messages as a matrix and convolving them over multiple layers to analyze spam or ham probabilities in the end. The parameters for the CNN model we used are below in Figure 2 on the right.

Model: "model_9"			Model: "model_5"		
Layer (type)	Output Shape	Param #	Layer (type)	Output Shape	Param #
Inputs (InputLayer)	[(None, 150)]	0	Inputs (InputLayer)	[(None, 150)]	0
embedding_9 (Embedding)	(None, 150, 50)	50000	embedding_5 (Embedding)	(None, 150, 64)	64000
dropout_10 (Dropout)	(None, 150, 50)	0	conv1d_2 (Conv1D)	(None, 148, 32)	6176
lstm_7 (LSTM)	(None, 64)	29440	max_pooling1d_2 (MaxPooling1D)	(None, 74, 32)	0
FC1 (Dense)	(None, 256)	16640	leaky_re_lu_4 (LeakyReLU)	(None, 74, 32)	0
leaky_re_lu_7 (LeakyReLU)	(None, 256)	0	flatten_1 (Flatten)	(None, 2368)	0
dropout_11 (Dropout)	(None, 256)	0	out_layer (Dense)	(None, 1)	2369
out_layer (Dense)	(None, 1)	257	activation_5 (Activation)	(None, 1)	0
activation_9 (Activation)	(None, 1)	0			
Total params: 96,337			Total params: 72,545		
Trainable params: 96,337			Trainable params: 72,545		
Non-trainable params: 0			Non-trainable params: 0		

Figure 2.



## 6 Experiments, Results, and Discussion

TF— IDF, CountVectorizer, Tokenizer, and Transformer Based features were used to train on five probabilistic models namely Naive Bayes, Logistic Regression, Support Vector Machine (SVM), K Nearest Neighbors, Random Forest and three Deep learning models including an ong Short-Term Memory(LSTM), Convolutional Neural Network (CNN) and a hybrid model of CNN and LSTM. The results from our experiments showed that the Random Forest Model with CountVectorizer features provided the best performance for this classification task with an accuracy of 99.821 % as seen in figures 3.

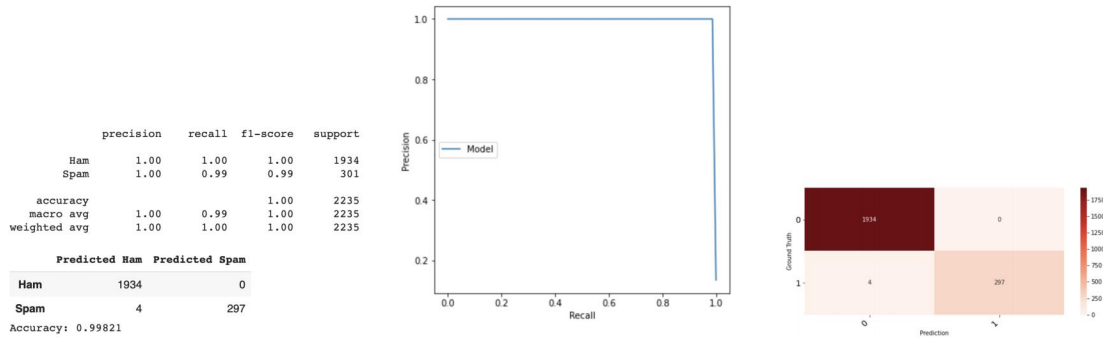


Figure 3.

Given the limit of the number of pages we are unable to display all the graphs.

## 7 Conclusion and Future Work

Our work showed that using a count vectorizer with a random forest algorithm to classify SMS messages was most highly accurate in identifying spam, which by the dataset construction meant that it identified nearly all fraudulent messages sent. We hope to build more on the data processing side to fully develop a system on this model that might most effectively, compared to current products on the market, keep vulnerable people in developing economies safe.

## 8 Contributions

Every part of the project was equally contributed to by both partners.

## 9 Link to Our Code

<https://github.com/Fatmangh/Mobile-Money-Fraud-Detection-using-Deep-Learning>

## References

- [1] E. A. Lopez-Rojas , A. Elmir, and S. Axelsson. "PaySim: A financial mobile money simulator for fraud detection". In: The 28th European Modeling and Simulation Symposium-EMSS, Larnaca, Cyprus. 2016
- [2] Almeida, T.A., Gomez Hidalgo, J.M., Yamakami, A. Contributions to the Study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011.
- [3] "Control of Fraud on Mobile Money Services in Ghana: An Exploratory Study | Emerald Insight." Emerald Insight, 7 May 2019, [www.emerald.com/insight/content/doi/10.1108/JMLC-03-2018-0023/full/htmlloginreload](http://www.emerald.com/insight/content/doi/10.1108/JMLC-03-2018-0023/full/htmlloginreload).

- [4] Botchey, Francis Effirim. "Mobile Money Fraud Prediction—A Cross-Case Analysis on the Efficiency of Support Vector Machines, Gradient Boosted Decision Trees, and Naïve Bayes Algorithms." MDPI, 31 June 2020, [www.mdpi.com/2078-2489/11/8/383/html](http://www.mdpi.com/2078-2489/11/8/383/html)B43-information-11-00383.
- [5] Magagula, Shile. "Mobile Money SMS Classification And Text Analysis: Exploring Possibilities For Enhanced Financial Inclusion." Bachelor thesis, May 2020.
- [6] Bandyopadhyay, Samir and Shawni Dutta. "Detection of Fraud Transactions Using Recurrent Neural Network during COVID-19." Preprints 2020, 2020060368 (doi: 10.20944/preprints202006.0368.v1).
- [7] Adedoyin, Adeyinka. "Predicting Fraud in Mobile Money Transfer." PhD thesis, June 2018.
- [8] Adedoyin, Adeyinka, et al. "Predicting Fraud in Mobile Money Transfer Using Case-Based Reasoning." In: Bramer M., Petridis M. (eds) Artificial Intelligence XXXIV. SGAI 2017. Lecture Notes in Computer Science, vol 10630. Springer, Cham.