BigFish

The AI Phishing Detection Algorithm

Adam Kablawi

Introduction:

Phishing emails remain the leading computer infection vector worldwide. A multitude of empirical studies has been conducted on the issue of malware attack vectors (Antal; Daugherty), stating that “attackers often target emails”, arriving in employee inboxes on “corporate endpoints”. Phishing emails contain an attachment carrying a malicious file, which if the user is persuaded to open, may result in a plethora of attacks on the operating system and/or network of the user. This is an effective scam, identified in “41%” computer infection incidents. The approach presented in this paper ensures organizational safety, hypothesizing that the widespread implementation of the AI phishing recognition system *BigFish* will heavily reduce compromising incidents and protect company privacy from malicious emails. This paper first establishes the methodology and process of developing the machine learning algorithm (MLA), a thorough step-by-step explanation of the algorithmic structure, and finally an evaluation of various authenticity metrics.

Methodology:

The goal of *BigFish* is to develop an accurate AI algorithm that can recognize potential phishing email scams, specifically living up to an accuracy of above 85%. The program will be designed in Python (using the Google Colab IDE) as it is both efficient and advanced. First, a dataset must be collected containing a large number of both phishing and legitimate emails to ensure a high level of diversity (Shantanu Dhakad). Such a dataset is then formatted to UTF-8 character format and is then pre-processed in the following procedure:

1. Loading Data: The code reads the dataset containing legitimate and phishing emails.
2. Combining Data: The legitimate and phishing emails are combined, ignoring the order.
3. Preprocessing: Text data is transformed into numerical data using TF-IDF Vectorization: a measure of word originality by comparing the number of times a word appears in a file with the number of files the word appears in.

After making the data essentially eligible for the code to understand, it will be processed through a previously existing learning algorithm. Out of numerous routes, logistic regression (LR) was selected: an MLA that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. It is easy to interpret, less demanding and excels with binary inputs. After the model is trained to manipulate the training set, it will use the analyzed patterns to predict labels for the test set.

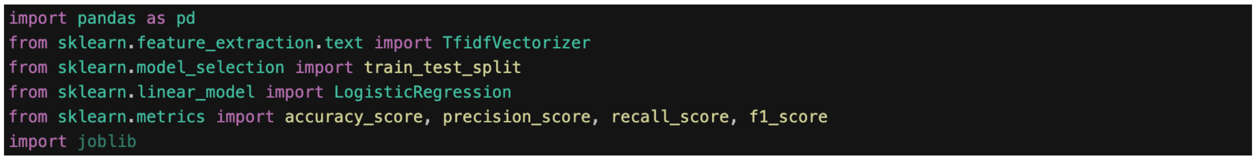
To improve the MLA, the performance of the model is evaluated using numerous metrics (accuracy, precision, recall, and F1 score) which locate the strengths and weaknesses within the algorithm. These metrics will then be utilized to reconsider certain areas of the code which will be altered to improve the accuracy.

- Accuracy: A measure of how often the MLA correctly predicts the outcome
- Precision: A measure of the quality of positive predictions within an MLA (true positives divided by total positive predictions)
- Recall: A measure of how often the MLA correctly identifies positive instances (true positives divided by the sum of true positives & false negatives)
- F1 Score: The harmonic mean of precision and recall

In the analysis of MLA performance, computing false negatives is indispensable to the critical process. False negatives are the most dangerous of the 4 possible outcomes within a binary computing system: false negatives, false positives, true negatives, and true positives. To further depict the situation, imagine receiving a negative COVID-19 swab test result while infected; the result allows for a scenario in which the misinformed infected individual can harm numerous others. This same scenario plays out with phishing emails. If the MLA outputs a negative result after analyzing an email, and the email is conversely phishing, the user will be compromised after clicking on a malicious link. Therefore, the minimization of MLA false negatives is necessary for the functionality of the program.

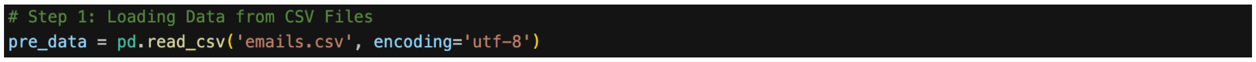
Structure:

The following images are the entirety of the code, split into distinct sections:



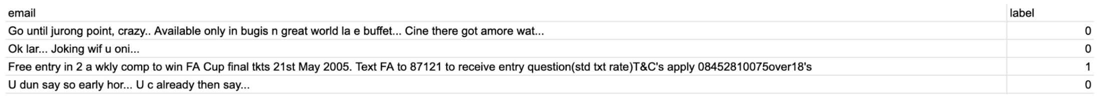
```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import joblib
```

Python libraries (collections of pre-written code and functions) are first imported, which are machine learning affiliated. Two of the key tools include the Term Frequency-Inverse Document Frequency (TF-IDF)–a method used to transform text documents into vectors, highlighting the relevance of each word. It operates on the bag-of-words model, generating a matrix that indicates the importance of both less relevant and highly relevant words within the document; and Logistic Regression–a supervised machine learning algorithm used for binary classification tasks. It predicts the probability of an outcome, event, or observation, resulting in a binary or dichotomous outcome limited to two possibilities: yes/no or 0/1.

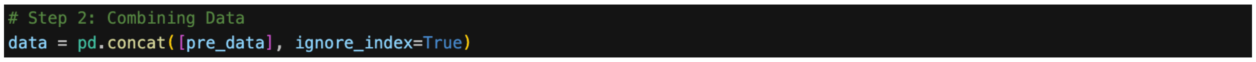


```python
# Step 1: Loading Data from CSV Files
pre_data = pd.read_csv('emails.csv', encoding='utf-8')
```

This loads the two columns from this CSV file which is composed of the written emails and their 'label' (the classification of whether the email is phishing or not using 0 and 1).

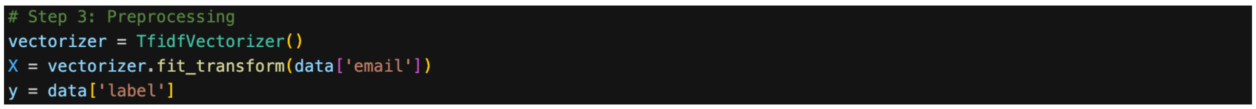


email	label
Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	0
Ok lar... Joking wif u oni...	0
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	1
U dun say so early hor... U c already then say...	0



```python
# Step 2: Combining Data
data = pd.concat([pre_data], ignore_index=True)
```

The entire table is turned into a concatenated list, ignoring the index (order) of the elements within the table (randomizing it). This makes it effortless to use the TF-IDF Vectorizer.



```python
# Step 3: Preprocessing
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(data['email'])
y = data['label']
```

The TF-IDF Vectorizer here turns the entirety of the list into vectors–a mathematical technique–that allows the dataset to recognize highly repetitive words. The model then splits up the independent and dependent variables (the vectorized emails, and the label [0 or 1]).

```python
# Step 4: Training the Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
```

The independent and dependent variables are split up into two sections: training and testing. The test set is 20% of the data, and the training set is 80%. The random state ensures the replicability of the split by setting a seed for the random number generator.

To train the model, it is specified here that the model to be used is Logistic Regression which was imported in the beginning. Training data points are then fed into the logistic regression model to help the model predict the probability of such outcomes.

```python
# Step 5: Evaluating the Model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

The model here is evaluated based on four parameters mentioned earlier in this paper, aiding in the evaluation process of understanding further potential improvements. Such analytical equations are provided by the Python Library–increasing efficiency.

```python
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
```

This section prints each of the statistics for the user to view and provide the developer with feedback.

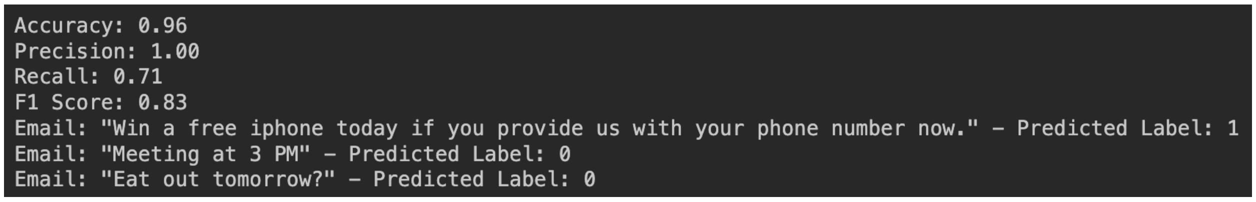
```python
# Using the model to make predictions
new_emails = ["Win a free iphone today if you provide us with your phone number now.", "Meeting at 3 PM", "Eat out tomorrow?"]
X_new = vectorizer.transform(new_emails)
y_new_pred = model.predict(X_new)
```

Here, examples of emails a user may provide the program to evaluate are inputted. Hence, the previous algorithm is used to analyze the string for possibilities of phishing.

```python
# Print predictions
for email, label in zip(new_emails, y_new_pred):
    print(f'Email: "{email}" - Predicted Label: {label}')
```

The model ultimately outputs the predictions of whether the email is phishing or not, printing it alongside the previous evaluative metrics.

Evaluation:



```
Accuracy: 0.96
Precision: 1.00
Recall: 0.71
F1 Score: 0.83
Email: "Win a free iphone today if you provide us with your phone number now." - Predicted Label: 1
Email: "Meeting at 3 PM" - Predicted Label: 0
Email: "Eat out tomorrow?" - Predicted Label: 0
```

Various issues arose in the development of this program, causing minor setbacks in the authenticity metrics of the algorithm. Fortunately, the 85% accuracy goal was exceeded at 96%, accompanied by a fantastic precision of 100%. The various emails given to the program were additionally predicted correctly and accurately.

However, the recall and f1 score came short at 71% and 83%; observing the computational equations of recall and f1 score, the only distinction between them and the other two metrics is the incorporation of false negative rates. The high frequency of false negatives was detrimental to the metrics.

Various limitations contribute to the high false-negative rate associated with the software. Because the computer used to design *BigFish* was a 2020 MacBook Air M1, limitations arose with the number of files that could be downloaded, processed, and uploaded. Boundaries were therefore set with the types of machine learning available, strictly allowing the usage of Logistic Regression and not a more advanced learning algorithm such as Random Forest. The computational limitations additionally forced the usage of a dataset with only 5000 emails. Research safety limitations allowed strictly the incorporation of a "Spam/Ham" email dataset, and not phishing emails (although quite similar). Additionally, this set is heavily composed of "Ham" emails (legitimate), which is highly reflected by the incredible precision of the algorithm, and the relatively weak recall. With further investment, a larger, real phishing email dataset can be incorporated into the training of the algorithm, and computed with a much more efficient system that can operate random forest computing.

To mass-implement such a program, the evaluative metrics must be nearly perfect. Technology frequently tends to induce addictive and reliant behaviors from users, which can be a major complication with this program. By implementing such an MLA, users begin to leave the thinking to the machine and re-allocate their energy and time to other tasks to increase their efficiency. With a limited MLA and reliant users, the software may not be as effective as intended. Therefore, after improving the algorithm, accompanying the software must be both a disclaimer of imperfection and a correction of asymmetric information about phishing scams.

Conclusion:

Despite numerous limitations, the findings from the development and implementation of the AI phishing recognition system, *BigFish*, demonstrate promising results in mitigating phishing threats. The system, leveraging a sophisticated machine learning algorithm, effectively identifies and neutralizes malicious threats before they can compromise operating systems or networks. Although not foolproof, *BigFish* substantially reduces the incidence of successful phishing scams as represented by the evaluative figures above. These findings underscore the potential of advanced AI systems in enhancing cybersecurity measures, thereby offering a viable solution to the pervasive issue of employee asymmetric information. The continued refinement and widespread implementation of *BigFish* can further bolster its effectiveness, ensuring greater protection for companies against malicious email-based attacks.

Bibliography:

Antal, Gabriella. “What Are the Main Ransomware Attack Vectors?” *Heimdal Security Blog*, Heimdal Security, 25 Jan. 2023, heimdalsecurity.com/blog/main-ransomware-attack-vectors/#:~:text=Phishing,-Without%20a%20doubt&text=Phishing%20remains%20the%20most%20popular,ransomware%2C%20because%20it%20never%20fails. Accessed 19 May 2024.

Daugherty, Taelor. “Phishing Is the Top Attack Method Used by Threat Actors.” *Securitymagazine.com*, Security Magazine, 7 Feb. 2024, www.securitymagazine.com/articles/100361-phishing-is-the-top-attack-method-used-by-threat-actors#:~:text=Phishing%20remains%20the%20leading%20infection,2022%2C%20compared%20to%202021%20data. Accessed 19 May 2024.

Shantanu Dhakad. “Email Spam Detection Dataset (Classification).” *Kaggle.com*, 2022, www.kaggle.com/datasets/shantanudhakadd/email-spam-detection-dataset-classification. Accessed 20 May 2024.