Machine Learning for cybersecurity

Pétia Georgieva
Course Instructor TAA 2023/24
DETI, University of Aveiro
petia@ua.pt

Tiago Pereira
Student TAA 2023/24
DETI, University of Aveiro
tfgp@ua.pt

Vítor Santos
Student TAA 2023/24
DETI, University of Aveiro
maedje@typst.app

***Abstract*—This report revolves around the work developed by Tiago & Vítor for the final TAA project of the 2023/24 class. The theme *Machine Learning for cybersecurity*. Through context analysis, dataset manipulation and model exploration we now present the results obtained from the work developed over the last half of the semester.**

***Index terms*—Scientific writing, Typesetting, Document creation, Syntax**

I. Introduction

Exploitation. Since the dawn of human era, exploitation of others in seek of personal gain has always been a problem. *Why?* Because, in order for the perpetuators to obtain their so desired advantage, there's always the need of a victim. Through the years, technology has evolved rapidly, and exploitation methods have not only evolved with it, but now use it as a powerful weapon. E-mail exploits, such as spam & phishing are common ways to breach users rights by obtaining their classified data, or violating their privacy and/or invasion of personal space through unwanted ads and requests. As evidenced by a study developed by APWG [1], exploits continue to multiply year after year, causing millions in losses. As engineers, it is our duty to find ways to solve these problems and boost common users safety, privacy and will over the ever-growing global network that, nowadays, contains most of our information. As a start, we decided to address the referenced problems regarding e-mails exploits through spam, by exploring machine learning capabilities. But, *how?* Let's dive into it.

II. Problem Complexity

E-mails are related to natural language expression. This means we're dealing with an **infinite** space of possibilities. In order for our models to be effective, they need to accurately extrapolate classifications from training data. In order to achieve this, the following requirements must be met:

1. Dataset includes a wide-range of suffice examples to represent the whole e-mail paradigm
2. Dataset examples must includes ham(normal) & spam e-mails, properly identified.
3. Models present good performance, ideally similar, on training *and* test data.

Our work throughout this project was based on these premisses.

III. State of the Art

After analyzing the complexity of problem in hands and setting base requirements, we started to analyze existing solutions. Articles such as *Leveraging Large Language Models for Effective Phishing Email Detection* [2] were of help to understand how phishing e-mails are detected. We realized that **Recurrent Neural Networks** (RNN) were the go-to tool regarding this topic. These networks, when trained by large IT companies, managed to obtain accuracies on test data upwards of 99%. Was also interesting to see, in the first referenced article [2], that **Large Language Models** (LLMS) were now being used as a way to parse e-mails content text and identify feral spam examples, with the intent to *phish* users personal data. In fact, a solution based on ChatGPT, described in said article, managed to obtain 99.7% on test data.

We also extended our analysis to public code projects, such as those present in Kaggle & GitHub. From the examples found in Kaggle, 2 catched our attention. Both used RNN algorithms, specifically, **Long Short Term Memory** (LSTM) NN to identify spam e-mails. The first [3] project achieved accuracy on test data of 98.4% and the second [4] managed to obtain 97.7%. Both used the same Kaggle Spam dataset.

Figure 1: Dataset used on referred Kaggle Examples.

So, as a result of our SOA research, we decided to develop **RNN** & **CNN** solutions, but also, due to our curiosity, other models, like **Naive Bayes Network** and **SVM**.

With this superficial overview, we moved towards reviewing the existing academic literature on spam detection, focusing on methodologies that represent the current SoA in this field of natural language processing.

A. Feature extraction

Machine learning models cannot directly use unstructured text as input, requiring it to be parsed into well-structure numerical vectors and raising questions on how two fundamentally distinct information mediums can be encoded into one another without losing meaning. Feature extraction from text is a cornerstone of NLP and has therefore been the subject of much experimentation and discussion [5].

1) *Pre-processing:*

As

2) *Bag-of-words & term frequencies:*

This is an old, simple, yet highly popular model for representing text as numerical vectors: create a vocabulary of all words that appear in the corpus of documents and count the occurrences of each word at the level of each document, representing them with a vector of word occurrences [6]. Besides simplicity, the bag-of-words (BoW) model is adequately performant in processing most small to medium sized datasets.

In the spam detection sub-domain of text classification, BoW can yield a highly satisfactory performance (>95% scores across different methods) [7], not to mention its role as a fundamental step in classification models that inherently subsist on word occurrences. A different way to represent words is with term frequencies.

In a similar vein to BoW, term frequency inverse document frequency (TF-IDF) reflects the importance of terms in a corpus of documents by multiplying the frequency of a word in a document with the logarithm of the total number of documents divided by the number of documents containing that word [8].

3) *Word embeddings:*

B. Naive Bayes

Naive Bayes classifiers have historically been a popular and surprisingly effective solution for spam detection, given its simplicity. As such, the literature is extremely rich in this topic. In summary, NB classifiers predict the class of a given sample based on the computed probabilities of learnt features and classes [9]. Probabilities are modelled and calculated using Bayes' theorem:

$$P(C_k \mid x) = \frac{P(C_k) * P(x_i \mid C_k)}{P(x)} \tag{1}$$

where C_k is class k, x is an example and x_i is a word that composes the example (feature).

Many authors demonstrate the effectiveness of NB classifiers, including Kumar et al. [10] in their article that explores Multinomial NB applied to spam detection, achieving highly satisfactory performance metrics.

C. K-Nearest Neighbors

KNN is one of the simplest machine learning algorithms in the field. In classification tasks, it conducts a majority vote of the K-nearest points to the sample to predict. Compared to other algorithms, KNN tends to fall short with sub-par accuracies rounding 80% [11], [10].

D. Support Vector Machines

Support vector machines, used for finding the best margin (distance between the decision boundary and the nearest data point from each class) that maximizes the separation between different classes, present an interesting advantage in spam detection as text-extracted features tend to be high-dimensional, with the number of feature dimensions often surpassing the number of samples [11], [10].

E. Convolutional Neural Networks

Convolutional Neural Networks (CNN), while famous for their applications in computer vision and image classification, have great potential in uncovering hidden parts in documents represented by word embeddings, while utilizing one-dimensional filters. CNNs have demonstrated high performance metrics for relatively simple architectures, as evidenced in Zhang et al's paper [12].

IV. Data Description

The chosen dataset for this project was ***Spam Email Classification*** [13], from Kaggle. This dataset results from a combination of two datasets, one from TREC [14] and the other from EROM [15].

This resulting dataset, of about 139 MB in size, possesses 83446 examples records of e-mails, labeled either as spam(1) or not-spam/ham(0).

We managed to analyse each example individually and identify its language. We used python library *langdetect* for that effect. The results showed that the dataset was mainly of english e-mails.

From the class histogram of the dataset records, we can realize that spam & ham examples are evenly represented, without need for data balancing.

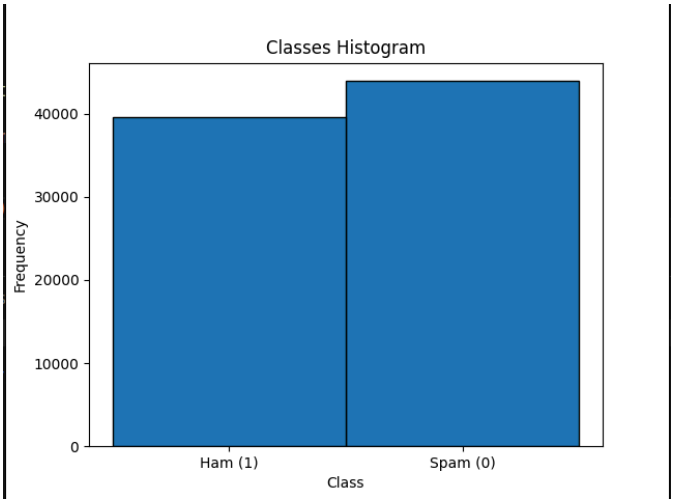

Figure 2: Class histogram of chosen dataset.

This equiparation is further evidenced from the resulting pie chart.

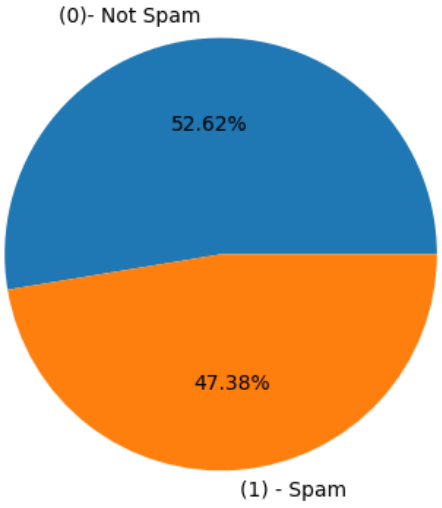

Figure 3: Class histogram of chosen dataset.

Each example consists of e-mails bodies (*body section*), with texts, links & attachments represented as text, each one with a corresponding label, just as shown in the following examples.

	label	text
0	1	ounce feather bowl hummingbird opec moment ala...
1	1	wulvob get your medircations online qnb ikud v...
2	0	computer connection from cnn com wednesday es...
3	1	university degree obtain a prosperous future m...
4	0	thanks for all your answers guys i know i shou...

Figure 4: Dataset records format examples (extracted form the Dataset first rows).

Since each example lied upon the natural language field, language itself was an important parameter to address.

Using *langdetect*, we realized the vast majority of the dataset was composed of english e-mails.

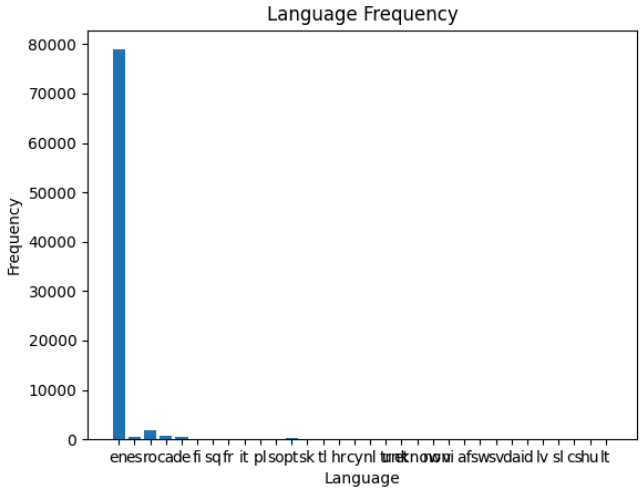

Figure 5: Different Languages Present in the dataset

V. General Data Preprocessing

As depicted from the last figure (section IV), over 4500 examples were of languages different from english. This should have a negative impact upon trained models since most of the training data provides information to identify english spam e-mails, and so, it would be possible that the obtained models would not correctly classify these examples. So, in order to solve this problem, the original dataset was filtered, and only english examples were let through. This meant our models would apply to the detection of **english** spam e-mails.

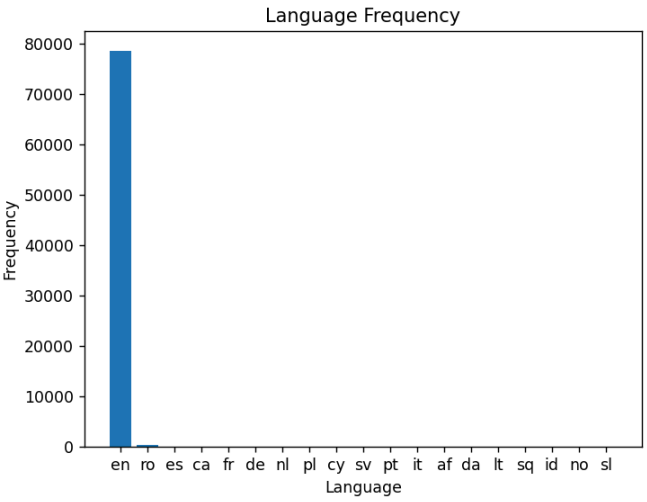

Figure 6: Cleaned Dataset with only English examples

DISCLAIMER: Still detected *ro* examples due to langdetect not fully-deterministic behaviour.

After filtering, the number of examples was reduced to **78903**. Though quantity was reduced, quality was surely improved! After analysing the chosen dataset and certifying it possesses all desired requirements(defined in section **II**), we were prepared to start taking the next steps to solve the context problem.

VI. KNN

A. *Context*

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed to recognize patterns in sequences of data, such as text, time series, or speech. Unlike traditional neural networks, RNNs have connections that form directed cycles, allowing information to persist and be used in future time steps. This makes them particularly well-suited for tasks involving sequential data where context and order are important. RNNs are capable of processing inputs of variable length by maintaining a hidden state that captures information about previous inputs, which can be leveraged to make predictions based on the sequence's history.

For email spam detection, RNNs can be highly effective because they can learn the context and sequential patterns in text. Spam detection often involves analyzing the text of emails to identify characteristics and patterns typical of spam. RNNs, especially their more advanced variants like LSTM (Long Short-Term Memory) networks, can capture long-range dependencies in text, making them adept at understanding the context and nuances that distinguish spam from legitimate emails.

However, RNNs are computationally intensive and can be slower to train compared to other models like traditional machine learning algorithms (e.g., Naive Bayes, SVM) or even more modern architectures like Transformers. Additionally, for spam detection, simpler models combined with feature engineering (e.g., keyword detection, frequency analysis) might be sufficient and more efficient. So, while RNNs are a powerful tool for email spam detection, their performance usually lies under more complex models()

B. *Methodology*

To train a KNN model for spam email classification, we utilized the scikit-learn library, leveraging the KNeighborsClassifier module for its implementation. The dataset was preprocessed using techniques such as text tokenization, term frequency-inverse document frequency (TF-IDF) transformation, and feature selection with chi-squared statistics.

Firstly, the dataset was loaded and split into training and testing sets using a 70/30 ratio. During the training phase, we employed the GridSearchCV function from scikit-learn to explore and optimize hyperparameters efficiently. Specifically, we varied the number of neighbors (n_neighbors), the use of IDF (inverse document frequency) in TF-IDF weighting, and the proportion of features selected via chi-squared test (chi_portion).

The KNNClassifier class, extending scikit-learn's BaseEstimator and ClassifierMixin, encapsulates our model. Within this class, the fitpreprocess method preprocesses the training data by converting text into numerical features using CountVectorizer and applying TF-IDF weighting. It further selects the most informative features based on chi-squared scores.

After hyperparameter tuning and model training, the best-performing model was selected based on the F1 score metric. This model was then serialized using joblib and saved to disk as 'knn_classifier.sav' for future use.

During testing, the saved model was loaded and evaluated on the test dataset to generate classification reports detailing precision, recall, and F1 scores for each class (spam and non-spam). Additionally, a Receiver Operating Characteristic (ROC) curve was plotted to visualize the model's performance in distinguishing between spam and non-spam emails, with the Area Under the Curve (AUC) providing a summary of the model's predictive power.

C. *Tested Configurations*

To obtain the best possible model, we tested different configurations, with variations on number of neighbors, proportion of selected features according to their chi-quare score(-more details at [16]) and IDF vectorization. In total, 18 different configurations were tested and analyzed.

```
param_grid = {
    'n_neighbors': [3, 5, 7],
    'use_idf': [True, False],
    'chi_portion': [0.5, 0.7, 1.0]
}
```

Figure 7: KNN: Tested Configurations with different hyperparameters

D. *Results*

From the indicated configurations, we obtained a best-fit model, according to its loss on validation data. This loss variation can be seen on the loss graphic.

This graphic allow us to understand that the model had reach stability at the minimum of its validation loss without risking severe overfitting, due to prolonged training on a stationary validation loss state.

The following report indicates performance metrics on *test* data.

```
=== Best model's params:
 {'chi_portion': 0.5, 'n_neighbors': 3, 'use_idf': False}

Classification report:
              precision    recall  f1-score   support

           0       0.99      0.69      0.81     11953
           1       0.78      0.99      0.87     13082

    accuracy                           0.85     25035
   macro avg       0.89      0.84      0.84     25035
weighted avg       0.88      0.85      0.85     25035
```

Figure 8: Best KNN model report (metrics on test data)

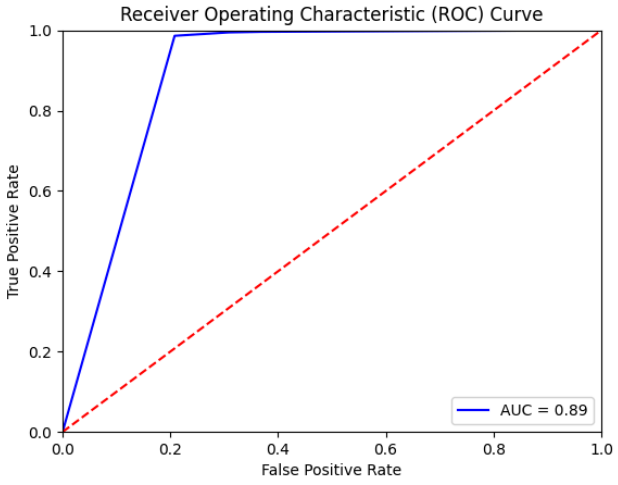

Figure 9: Best KNN model ROC Curve

With a AUC of **0.89**, our models is a valid option at classifying spam e-mails, correctly classifying 85% of all cases, and **99%** of all spam e-mails. The fact almost every single spam e-mail was correctly classified was a success! After all, this was the primary goal, in order to ensure users safety and privacy. However, **31%** of non-spam e-mails were wrongfully classifies as spam, which is a high value, and deeply impacts the commodity of users, if this models was to be widely adopted. This showed us that there was still room for improvement through the adoption of more complex models.

When compared to the KNN model in found in the SOA phase (see [17]), we realize our model falls behind. These changes depend, not only on the quality of the datasets, as a IEEE renowned research article possesses elevated access to high-quality resources, but also parameter tweaking, and libraries.

Data	Support	Precision	Recall	F1- Score
Ham	949	0.99	0.93	0.96
Spam	166	0.71	0.93	0.80
avg / total	1115	0.94	0.93	0.94
Accuracy	0.931835650224216			

Figure 10: IEEE article best KNN model report

However, our model recall on spam is noticeably better. So, in resume, our model is better at identifying spam messages, but is deeply punished by its high quantity of false positives(FP). Due to this tradeoff, each model be preferred by different users, at different contexts.

VII. Random Forest

A. *Context*

Random Forest is a versatile machine learning algorithm known for its effectiveness in classification tasks. It operates by constructing multiple decision trees during training and outputs the mode of the classes (for classification) of the individual trees. Each tree in the forest is trained independently on a subset of the data and features, using techniques like bagging (bootstrap aggregating) to reduce overfitting and improve generalization.

In the context of spam detection problems, Random Forests can be particularly suitable due to several reasons:

Ensemble Learning: Random Forests are an ensemble of decision trees, which tends to perform well by reducing variance and improving robustness compared to individual decision trees. This makes them less susceptible to overfitting, which is beneficial when dealing with noisy or imbalanced datasets typical in spam detection.

Feature Importance: Random Forests provide a measure of feature importance, indicating which features (words or tokens in the case of text data) are most influential in making predictions. This can help in understanding which words or patterns contribute most to identifying spam emails.

Non-linear Relationships: They can capture complex non-linear relationships between features and target variables, which is crucial when the boundary between spam and non-spam emails is not linearly separable.

Scalability: Random Forests are scalable and can handle large datasets efficiently. They are also parallelizable, which can leverage multicore architectures for faster training.

In conclusion, Random Forests are well-suited for spam detection tasks due to their robustness, ability to handle high-dimensional data, and capability to provide insights into feature importance. When properly tuned, they can de-

liver high accuracy and generalization performance in distinguishing between spam and legitimate emails.

B. Methodology

To train a Random Forest model for spam email classification, we utilized the scikit-learn library, leveraging the RandomForestClassifier module for its implementation. The dataset was preprocessed using techniques such as text tokenization, term frequency-inverse document frequency (TF-IDF) transformation, and feature selection with chi-squared statistics.

Firstly, the dataset was loaded and split into training and testing sets using a 70/30 ratio. During the training phase, we employed the GridSearchCV function from scikit-learn to explore and optimize hyperparameters efficiently. Specifically, we varied the number of estimators (n_estimators), the use of IDF (inverse document frequency) in TF-IDF weighting, and the proportion of features selected via chi-squared test (chi_portion).

The ForestClassifier class, extending scikit-learn's BaseEstimator and ClassifierMixin, encapsulates our model. Within this class, the fit preprocess method preprocesses the training data by converting text into numerical features using CountVectorizer and applying TF-IDF weighting. It further selects the most informative features based on chi-squared scores.

After hyperparameter tuning and model training, the best-performing model was selected based on the F1 score metric. This model was then serialized using joblib and saved to disk as 'random_forest_classifier.sav' for future use.

During testing, the saved model was loaded and evaluated on the test dataset to generate classification reports detailing precision, recall, and F1 scores for each class (spam and non-spam). Additionally, a Receiver Operating Characteristic (ROC) curve was plotted to visualize the model's performance in distinguishing between spam and non-spam emails, with the Area Under the Curve (AUC) providing a summary of the model's predictive power.

C. Tested Configurations

Also were, multiple configurations were tested, with variations on number of estimators(small decision trees), proportion of selected features according to their chi-quare score(more details at [16]) and IDF vectorization. In total, 30 different configurations were tested and analyzed.

```
param_grid = {
    'n_estimators': [50, 100, 200, 300, 400],
    'use_idf': [True, False],
    'chi_portion': [0.5, 0.7, 1.0]
}
```

Figure 11: Random Forest: Tested Configurations with different hyper-parameters

D. Results

- **Classification Performance**

The best-performing random forest spam classifier demonstrated excellent performance on the test data. The confusion matrix and the classification report provide insights into the model's predictive capability, specifically its precision, recall, F1 score, and accuracy across both classes (spam and non-spam).

```
=== Best model's params:
 {'chi_portion': 0.5, 'n_estimators': 400, 'use_idf': True}

Classification report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     11843
           1       0.99      0.99      0.99     13192

    accuracy                           0.99     25035
   macro avg       0.99      0.99      0.99     25035
weighted avg       0.99      0.99      0.99     25035
```

Figure 12: Best Random Forest model report (metrics on test data)

Precision: The model achieved a precision of 0.98 for class 0 (ham) and 0.99 for class 1 (spam). This indicates that 98% of the emails predicted as non-spam were indeed non-spam, while 99% of the emails predicted as spam were indeed spam. Recall: The recall scores were 0.98 for class 0 and 0.99 for class 1, showing that the model was able to identify 98% of the actual non-spam emails and 99% of the actual spam emails. F1 Score: The F1 scores were 0.98 for class 0 and 0.99 for class 1, reflecting a balanced performance in terms of precision and recall. Accuracy: The overall accuracy of the model on the test data was 99%, indicating that the model correctly classified 99% of the emails.

Spam Recall: The recall for spam (class 1) is particularly crucial in spam detection tasks, as it measures the model's ability to correctly identify all spam emails. A recall of 0.99 for spam means that the model successfully identified 99% of the actual spam emails. This high recall is vital for minimizing the number of spam emails that go undetected, which is a key objective in spam filtering systems, exceptionally achieved by this model.

ROC Curve and AUC: The Receiver Operating Characteristic (ROC) curve further illustrates the model's performance. The ROC curve plots the true positive rate (recall) against the false positive rate (1-specificity), providing a visual representation of the trade-off between sensitivity and specificity.

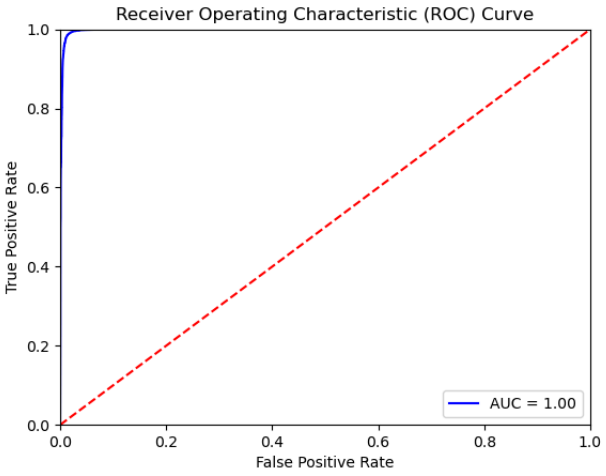

Figure 13: Best Random Forest model ROC Curve

AUC (Area Under the Curve): The AUC value for the ROC curve was found to be 1.00, which indicates perfect discrimination between the classes. An AUC of 1.00 signifies that the model is exceptionally effective in distinguishing between spam and non-spam emails. Conclusion

The random forest spam classifier exhibited robust performance metrics across all evaluation criteria. Its high precision, recall, and F1 scores, along with an accuracy of 99%, affirm its reliability in spam detection tasks. The perfect AUC score further underscores its predictive power, making it an ideal choice for practical spam classification applications.

This analysis underscores the **effectiveness** and **reliability** of the random forest classifier in identifying spam emails, based on the evaluation metrics and visual performance indicators provided.

In comparison to the models found on SOA, KNN & RNN, our model surpassed the KNN models, both performance and timewise and we were surprised to realize that it matched top RNN LSTM models precision, not only in terms of accuracy, but also precision, with similar time performance!

VIII. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a type of deep learning algorithm often associated with image classification and computer vision tasks, especially effective for their ability to detect and discover new features from the sample data without the need for sophisticated manual feature engineering by domain experts. Despite this popular association, CNNs are nevertheless still applicable to the context of spam detection and, more broadly, text classification with the analysis of word-embedded sentences.

A. Methodology

- **Libraries**

We used the Keras and Tensorflow libraries to develop and implement the CNN, alongside dataset loading operations and metrics tuning.

- **Data Preprocessing**

Stopwords were removed using the NLTK package's English stopwords list, augmented with additional custom stopwords. The text data was tokenized using TensorFlow's TextVectorization layer, which converts spam mail message bodies into sequences of tokens encoded as integers, belonging to a vocabulary fitted on the training dataset.

- **Word Embedding**

A word embedding layer was trained on the training dataset to encode the tokens from the previous step into vectors with contextual meaning.

- **Layered architecture**

Based on the conducted SoA review, we opted for using a one-dimensional convolutional layer with a kernel size of 3 (3-grams), 32 filters and RELU activation. This layer is followed by a max pooling layer, one fully connected RELU-activated layer and a single sigmoid-activated neuron for final predictions.

- **Model Training:**

The model was trained on the preprocessed email data, with a validation split of 20% to monitor performance on unseen data. Early stopping was implemented to halt training if the validation loss did not improve for three consecutive epochs, ensuring the model does not overfit.

- **Hyperparameter Tuning**

We attempted to use the Hyperband hyperparameter tuning algorithm to search for the optimal hyperparameters for our CNN. While Hyperband did help us find hyperparameters with relatively high accuracies (97% on validation), we could not run the algorithm to its full length due to constant performance issues related with the GPU and its drivers.

Were it not for these performance limitations, we could've explored more relevant hyperparameters, especially in regard to anti-overfitting measures such as dropout rates and regulization terms.

- **Model Evaluation:**

The model's performance was evaluated on a separate test set, which was not used during the training process. Key metrics such as precision, recall, and F1-score were calculated to assess the classification performance. ROC curve and respective AUC were also plotted, alongside accuracy/loss metrics over epochs.

IX. LSTM - RNN

A. *Context*

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models are advanced types of deep learning neural networks, specifically designed to handle sequential data, making them highly effective for text classification tasks, such as spam email detection. Unlike traditional neural networks, RNNs maintain a form of memory by using their hidden states to retain information from previous inputs, allowing them to understand context over sequences. LSTM models, an improved version of RNNs, address the vanishing gradient problem commonly faced by standard RNNs, enabling them to learn long-term dependencies more efficiently. In the context of spam email detection, LSTM models excel at capturing the nuanced patterns and contextual cues within email texts. By analyzing sequences of words and understanding their relationships, LSTM models can more accurately classify emails as spam or non-spam. Their ability to remember and utilize information from earlier parts of the email makes them particularly suited for this task, as they can detect subtle spam indicators that might be missed by simpler models. This leads to higher accuracy and better performance in filtering out unwanted emails, thereby enhancing email security and user experience.

B. *Methodology*

To train an RNN LSTM model for spam email classification, we utilized the TensorFlow and Keras libraries, leveraging the Sequential API for its implementation. The dataset was preprocessed using techniques such as text tokenization, sequence padding, and embedding. The RNN LSTM model was chosen for its ability to retain long-term dependencies and contextual information, which is crucial for accurately identifying spam emails.

- **Data Preprocessing**

Stopwords were removed using NLTK's English stopwords list, augmented with additional custom stopwords. The text data was tokenized using TensorFlow's Tokenizer, which converts the text into sequences of integers, where each integer represents a specific word in the vocabulary. Sequence Padding:

The sequences were padded to ensure uniform input length, using a maximum sequence length of 100. This is crucial for feeding the data into the LSTM model, which requires inputs of fixed size.

- **Model Development**

An embedding layer was used to convert the input sequences into dense vectors of fixed size (embedding dimension of 100). The LSTM layer, with 128 units, was added to capture temporal dependencies in the data. Dropout was applied to prevent overfitting. A dense layer with a sigmoid activation function was used for the final binary classification output (spam or ham). Hyperparameter Tuning:

Hyperparameters such as embedding dimensions, LSTM units, dropout rates, batch sizes, and epochs were optimized using GridSearchCV from Scikit-learn. The grid search evaluated combinations of parameters to identify the best configuration for the model.

- **Model Training:**

The model was trained on the preprocessed email data, with a validation split of 10% to monitor performance on unseen data. Early stopping was implemented to halt training if the validation loss did not improve for three consecutive epochs, ensuring the model does not overfit.

- **Model Evaluation:**

The model's performance was evaluated on a separate test set, which was not used during the training process. Key metrics such as precision, recall, and F1-score were calculated to assess the classification performance.

- **Model Implementation**

The RNN LSTM model was encapsulated using the KerasClassifier wrapper from Scikit-learn. The fit method preprocesses the training data by tokenizing and padding the text sequences. The GridSearchCV function was used to perform hyperparameter tuning and select the best model based on the F1 score metric. This model was then serialized using joblib and saved to disk as 'rnn_classifier_best.h5' for future use.

C. *Tested Configurations*

Also were, multiple configurations were tested, with variations on embedding dim, LSTM units, dropout rate and batch size. In total, 36 different configurations were tested and analyzed.

```
param_grid = {
    'model__embedding_dim': [50, 100, 200],
    'model__lstm_units': [128, 256],
    'model__dropout_rate': [0.2, 0.5],
    'batch_size': [32, 64, 128],
    'epochs': [20]
}
```

Figure 14: LSTM: Tested Configurations with different hyper-parameters

D. *Results*

From the tested solutions, the LSTM RNN model emerged as the top performer, achieving an impressive 98.5% accu-

racy on all test data. The detailed classification report further highlights the model's robustness:

Classification report:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	11530
1	0.98	0.99	0.99	12142
accuracy			0.99	23672
macro avg	0.99	0.99	0.99	23672
weighted avg	0.99	0.99	0.99	23672

Figure 15: Best LSTM RNN report(metrics on test data)

Precision: The model demonstrated excellent precision for both classes, with 0.99 for non-spam (ham) and 0.98 for spam emails. This high precision indicates a very low false positive rate, meaning the model is highly accurate in identifying true spam emails without misclassifying too many legitimate emails as spam.

Recall: The recall values are equally strong, with 0.98 for non-spam and 0.99 for spam emails. This high recall for spam detection is particularly crucial, as it indicates the model's effectiveness in identifying almost all spam emails, minimizing the risk of spam emails slipping through undetected.

F1 Score: Both classes have an F1 score of 0.99, suggesting a balanced performance in terms of precision and recall. The F1 score is particularly useful in situations where both false positives and false negatives are equally costly.

Overall, the macro and weighted averages of precision, recall, and F1 scores are all 0.99, reaffirming the model's consistent and reliable performance across both classes. This consistency is crucial for real-world applications where both precision and recall need to be maximized to ensure efficient spam detection.

This is, once again, evidenced by the ROC Curve. The AUC(=1.0) reflects that the models perfectly fitted the test data, both for positive and negative examples.

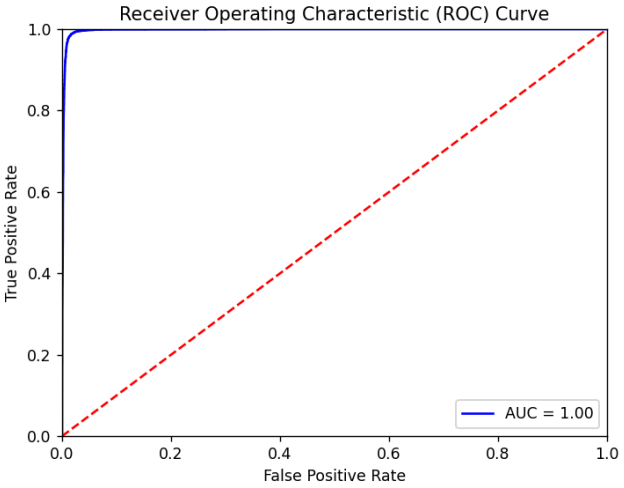

Figure 16: Best LSTM RNN ROC Curve

The obtained Model had the following structure:

=0.985 total time= 8.3min
Model: "sequential_29"

Layer (type)	Output Shape	Param #
embedding_29 (Embedding)	(None, 100, 100)	2,000,000
lstm_29 (LSTM)	(None, 256)	365,568
dense_29 (Dense)	(None, 1)	257

Total params: 2,365,825 (9.02 MB)
Trainable params: 2,365,825 (9.02 MB)
Non-trainable params: 0 (0.00 B)

Figure 17: Best LSTM RNN Structure, Params & Training Time

With a Training time of 3 x 8.3 min(3 epochs) = 25.9min, for 2.365 million parameters, the obtained performance is more than suffice!

Compared to the results obtained in the SOA section, our model surpasses the Kaggle examples and falls just shy of the Chat-GPT hybrid(see [2]).

X. Conclusions

We found specially interesting how such a simple empirical problem, such as spam detection to a Software Engineer, can turn into such a complex network of requirements.

Its also refreshing to see that we can present multiple satisfying solutions to a single problem. As expected, RNN LSTM presented itself as a prime solution, however, Random Forest, KNN and CNN also presented high-performance & efficiency on spam identification. KNN lacked the required precision on identifying non-spam e-mails. The high rate of FP would be disturbing to the common users daily life.

A. *Possible Future Directions*

- **Hybrid Models:**

Combining the strengths of LSTM models with traditional machine learning approaches could lead to improved per-

formance. For example, using LSTM-generated embeddings as features for a Random Forest model could leverage the sequential processing capability of LSTMs and the interpretability of Random Forests.

- **Attention Mechanisms:**

Integrating attention mechanisms into LSTM models can enhance their ability to focus on relevant parts of the text, potentially improving performance in spam detection by highlighting important words and phrases.

- **Transfer Learning:**

Leveraging pre-trained language models, such as BERT or GPT, and fine-tuning them for spam detection can provide a significant boost in performance. These models have been trained on vast amounts of data and can capture nuanced language patterns effectively.

- **Explainability:**

Developing methods to improve the interpretability of LSTM models, such as visualization tools and techniques for extracting feature importance, can make these models more transparent and easier to trust in practical applications.

- **Data Augmentation:**

Implementing data augmentation techniques to generate diverse training examples can help in addressing the challenge of limited labeled data and reduce overfitting. By continuously exploring these future directions, we can enhance the effectiveness and robustness of spam detection systems, ensuring they remain adaptable to evolving spam tactics and capable of providing reliable email filtering solutions. We could explore existing dictionaries to combine words in a random manner or create structured e-mails, for example, through generative AI(such as Chat-GPT).

- **Multi-Language Support:**

Allow for the classification of spam e-mails of multiple languages inter and intra e-mails. This would be possible by training multiple models, with different languages as sources or training one generalist model, which could lead to decreased performance.

By continuously exploring these future directions, we can enhance the effectiveness and robustness of spam detection systems, ensuring they remain adaptable to evolving spam tactics and capable of providing reliable email filtering solutions.

$$a + b = \gamma \tag{2}$$

References

[1] G. Smith, "Top Phishing Statistics for 2024: Latest Figures and Trends." [Online]. Available: https://www.stationx.net/phishing-statistics/

[2] H. N. D. C. Takashi Koide Naoki Fukushi, "ChatSpamDetector: Leveraging Large Language Models for Effective Phishing Email Detection," 2024.

[3] K. R. MARYADA, "Simple LSTM for text classification." [Online]. Available: https://www.kaggle.com/code/kredy10/simple-lstm-for-text-classification

[4] F. QURESHI, "Email Spam Detection." [Online]. Available: https://www.kaggle.com/code/mfaisalqureshi/email-spam-detection-98-accuracy

[5] A. Tabassum and R. R. Patil, "A survey on text pre-processing & feature extraction techniques in natural language processing," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 6, pp. 4864–4867, 2020.

[6] W. A. Qader, M. M. Ameen, and B. I. Ahmed, "An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges," in *2019 International Engineering Conference (IEC)*, 2019, pp. 200–204. doi: 10.1109/IEC47844.2019.8950616.

[7] S. Kaddoura, G. Chandrasekaran, D. Elena Popescu, and J. H. Duraisamy, "A systematic literature review on spam content detection and classification," *PeerJ Computer Science*, vol. 8, p. e830, 2022, doi: 10.7717/peerj-cs.830.

[8] P. Joseph and S. Y. Yerima, "A comparative study of word embedding techniques for SMS spam detection," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2022, pp. 149–155. doi: 10.1109/CICN56167.2022.10008245.

[9] G. I. Webb, E. Keogh, and R. Miikkulainen, "Naïve Bayes," *Encyclopedia of Machine Learning*, vol. 15, no. 1. Springer, pp. 713–714, 2010.

[10] N. Kumar, S. Sonowal, and Nishant, "Email Spam Detection Using Machine Learning Algorithms," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2020, pp. 108–113. doi: 10.1109/ICIRCA48905.2020.9183098.

[11] N. Ahmed, R. Amin, H. Aldabbas, D. Koundal, B. Alouffi, and T. Shah, "Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges," *International Journal of Distributed Sensor Networks*, vol. 18, p. 1862888–1862889, 2022, doi: 10.1155/2022/1862888.

[12] Y. Zhang and B. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification." 2016.

[13] P. SINGHVI, "Spam Email Classification Dataset." [Online]. Available: https://www.kaggle.com/datasets/purusinghvi/email-spam-classification-dataset/data

[14] TREC, "2007 TREC Public Spam Corpus." [Online]. Available: https://plg.uwaterloo.ca/~gvcormac/treccorpus07/

[15] ENROM, "Enron-Spam." [Online]. Available: http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/index.html

[16] D. Guru and V. Kumar N, "Interval Chi-Square Score (ICSS): Feature Selection of Interval Valued Data," 2020, pp. 686–698. doi: 10.1007/978-3-030-16660-1_67.

[17] A. R. Yeruva, D. Kamboj, P. Shankar, U. S. Aswal, A. K. Rao, and C. S. Somu, "E-mail Spam Detection Using Machine Learning – KNN," in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*, 2022, pp. 1024–1028. doi: 10.1109/IC3I56241.2022.10072628.