

Spam Filter Report

By Christo-Odyseus Keramitzis

Aim of the Project

The purpose of this project is to develop modern and relevant defences against common phishing/spam email attacks. According to modern risk assessment tools and approaches, employees are statistically the most at risk with one of the largest financial and operational impacts from a cyber attack. In Deloitte's press release, Lumpur, K. (2020, January 9) reports "91% of all cyber attacks begin with a phishing email to an unexpected victim", depicting its scalability and mass application. Due to employees' general high level clearance and access to sensitive information, phishing/email attacks have historically been the most successful throughout all levels of management as a result of its simplicity.

Key content of the Project

This paper examines the effectiveness of modern defences such as spam filtering to reduce the success of email attacks by employing a supervised machine learning algorithm trained on a csv table. This table, filled with labelled data such as spam and non spam headers, will teach the algorithm to detect unwanted and potential harmful emails. The relevant features have been extracted to reach the desired result. Furthermore, the algorithm takes advantage of support vector machine learning (SVM) for its reliability in supervised learning. To take full advantage of SVM and unsupervised machine learning, a large data set will be used.

Current Development of this topic

Currently, the key areas of development in phishing/spam email detection are namely: advanced machine learning and deep learning, and natural language processing.

Oluchukwu et al. (2024) highlight the effectiveness of Naive Bayes, SVMs, neural networks, and decision trees for classifying spam emails based on content and sender information. They also cite the use of machine learning algorithms for dynamic malware detection, demonstrating high accuracy in identifying malware by its behaviour. They recommend late multi-modal fusion frameworks with CNNs and continuous bag-of-words for tackling hybrid spam emails incorporating both images and text.

Natural Language Processing (NLP) comprises Natural Language Understanding (NLU) and Natural Language Generation (NLG). Garg & Girdhar (2021) defines that NLU interprets text for meaning, while NLG converts data into human-readable language. Assem Utaliyeva et al. (2023) assert that NLP-based spam filters outperform traditional ML-based filters due to their superior semantic understanding and notes NLP's resilience against spam emails rewritten by AI chatbots like ChatGPT.

Solution for project

The reasonable solution chosen to mitigate spam email is to create a supervised machine learning algorithm to filter potential spam. Support vector machine learning algorithm is a prescribed method to automatically detect and filter out potential phishing attacks and spam email. As it is one of the most widely used, catalogued and reliable methods of supervised ML; SVM can be specifically implemented for a variety of use cases due to its customisation due to its variety of kernels. The recommended kernel used will be sigmoid due to its efficiency in classifying linear datasets; this will be achieved through the support vector machine learning classifier (SVC).

Method description

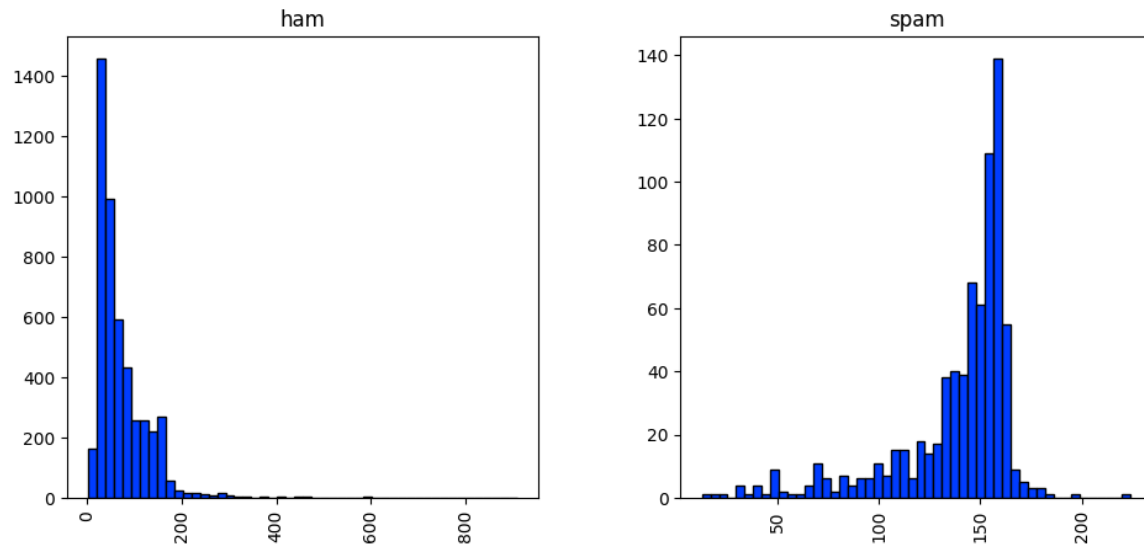
SVC is a reliable classifier that is easily modifiable to generate the desired variance and bias when vectorising. This tool is also incredibly efficient at classifying based on labelled data. SVM works by plotting data points through support vectors and regression lines; a decision boundary is then generated that classifies the data. This is depicted on a hyperplane with the axis being the features such as spam and non-spam. SVM, through the use of different kernels, has the ability to plot linear and non linear datasets. Svm is better suited due to its relatively low variance when plotting few feature data on a hyperplane through linear classification. Furthermore, the limited features within the dataset helps in overcoming SVC's disadvantages from a large number of features and datasets, thus outputting a more accurate classification. Additionally, K-fold cross-validation is necessary to output an accuracy score; with the addition of features without extending the number of samples, overfitting occurs as well as a large performance penalty. Thus, due to this use case of a spam filtering algorithm with limited parameters, svm using svc is suitable and provides a positive performance to accuracy exchange.

Examples of the prediction

A dataset of approximately 5000 cells consisting of spam and non-spam was used to generate an initial example. The features involve the message or body of the email, labels spam and ham and character lengths of messages.

	label	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

The graph below shows the spread of each mail's character length. On average spam includes more characters than non-spam; this may be due to the coercion spam attempts on users to redirect them and steal information.



The following results depict multiple methods of supervised learning and their accuracy results. SVC has a 0.98% accuracy score in detecting spam.

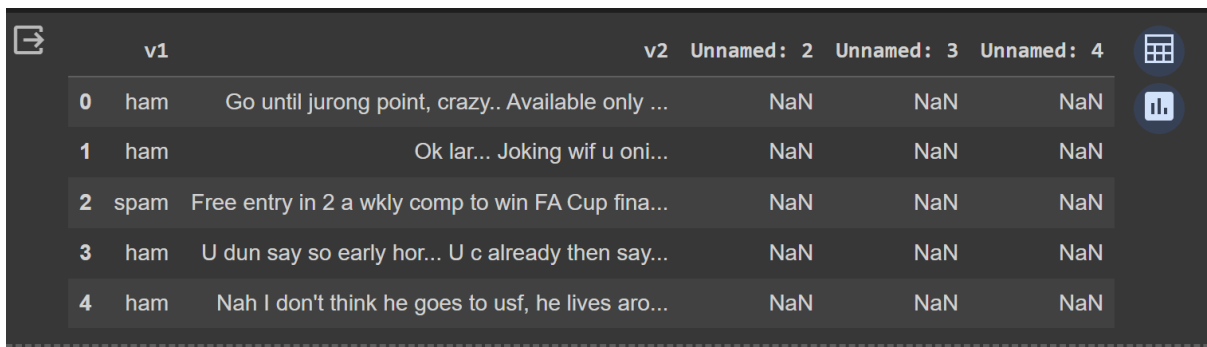
	0	1
0	SVC	[0.9784688995215312]
1	KN	[0.9246411483253588]
2	NB	[0.9844497607655502]
3	DT	[0.958732057416268]
4	LR	[0.9431818181818182]
5	RF	[0.9700956937799043]
6	AdaBoost	[0.9671052631578947]
7	BgC	[0.9665071770334929]
8	ETC	[0.9772727272727273]

WARNING:root:Quickchart encountered

Key method code

Text-preprocessing/feature extraction

Before training the algorithm on the labelled data, some small text processing to extract additional features and reduce noise from punctuation is necessary. This is important as having a readable and efficient dataset will reduce unnecessary processing and the chances of noise creating false positives.



	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

This code shows the renaming of column titles in addition to dropping unnecessary tables that will work against the algorithm.

```
[ ] sms = sms.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
    sms = sms.rename(columns = {'v1':'label','v2':'message'})
```

Now the data can be clearly organised and can accurately be queried if the dataset needs to be altered to provide a better reading. Below is a simple example query of 'I'll call later'.

```
[ ] sms.groupby('label').describe()

      message
      count unique top          freq
label
ham      4825   4516          Sorry, I'll call later      30
spam      747    653  Please call our customer service representativ...      4
time: 62.8 ms (started: 2024-03-28 11:12:32 +00:00)
```

Generating additional features can aid the algorithm in providing more accurate classifications due to the increased parameters it must consider when training with datasets. Providing character lengths of messages as a feature allows for additional sorting and

analysis of spam message types. This is done by inserting a column that counts characters and sorting from the top using head().

```
[ ] sms['length'] = sms['message'].apply(len)
    sms.head()
```

	label	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

Punctuation such as full stops, quotation, commas and exclamation are removed to reduce noise and unnecessary characters.

```
[ ] def text_process(text):
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = [word for word in text.split() if word.lower() not in stopwords.words('english')]
    return " ".join(text)

[ ] import nltk
    nltk.download('stopwords')
    text_feat = text_feat.apply(text_process)
```

Training of classifiers

In this section, classifiers have been imported and different 'settings' such as kernel type and gamma levels will be used when the algorithm iterates through the features.

SVC will be using cross-validation, it is important to prevent extreme biasness and ensure the algorithm does not completely rely on the generated features evident within the dataset. This is to ensure the algorithm can detect/function outside of this controlled environment i.e. detecting unknown spam within an independent email folder rather than from a curated list of

data. The industry accepted ratio is 30% sample size to be used for testing thus providing a practical performance to accuracy in training ratio.

```
[ ] features_train, features_test, labels_train, labels_test = train_test_split(features, sms['label'], test_size=0.3, random_state=111)
time: 13.7 ms (started: 2024-03-28 11:12:55 +00:00)
```

In regards to SVM, the support vector classifier will use sigmoid kernel and a varying gamma. Sigmoid is better suited for text based solutions rather than rbf, polynomial and other kernels. Gamma represents the effect each data point has on another; a very high gamma will create more bias and less variance and vice versa.

```
[ ] svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier(n_neighbors=49)
mnb = MultinomialNB(alpha=0.2)
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=31, random_state=111)
abc = AdaBoostClassifier(n_estimators=62, random_state=111)
bc = BaggingClassifier(n_estimators=9, random_state=111)
etc = ExtraTreesClassifier(n_estimators=9, random_state=111)

[ ] clfs = {'SVC': svc, 'KN': knc, 'NB': mnb, 'DT': dtc, 'LR': lrc, 'RF': rfc, 'AdaBoost': abc, 'BgC': bc, 'ETC': etc}
```

These functions then make the classifier usable so a prediction through iteration of the dataset can occur. In the photo below, the different training parameters and what the prediction is built from (features) are depicted.

```
[ ] def train_classifier(clf, feature_train, labels_train):
    clf.fit(feature_train, labels_train)

[ ] def predict_labels(clf, features):
    return (clf.predict(features))
```

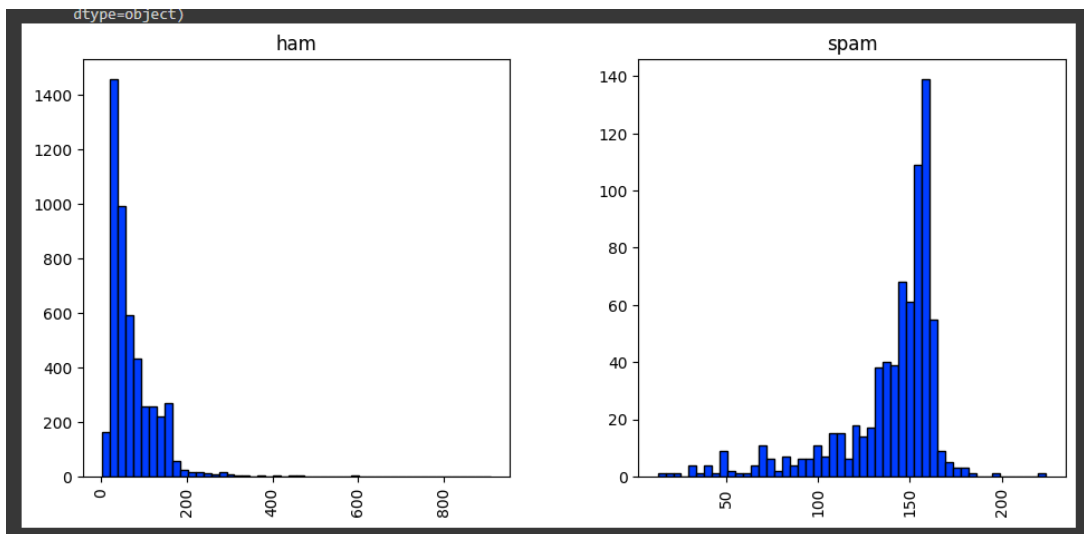
Here the classifiers iterate through the features and datasets in addition to training from the training sample size and seed. The result is a prediction/accuracy score that represents how effective the algorithm is at detecting spam

```
[ ] pred_scores = []
for k,v in clfs.items():
    train_classifier(v, features_train, labels_train)
    pred = predict_labels(v, features_test)
    pred_scores.append((k, [accuracy_score(labels_test, pred)]))
```


Metrics and results

Data Metrics and results

These following metrics were generated using sigmoid kernel, 1.0 gamma and a dataset consisting of 5573 (747 being spam) cells with a mixture of spam and non-spam. Below is a depiction of the dataset used with all relevant features (labels as titles, length of mail body in characters on x axis). As shown below, spam is positively skewed to the right with on average having a higher character count than non-spam messages.



Algorithm metrics and results

The figure below depicts the level of accuracy of SVM in comparison to other supervised machine learning methods. This is a positive outcome as at most 22.41 emails may be misclassified as spam or skewed towards being detected as spam. This result changes as we reduce the gamma with more variance occurring. Initial settings for svc:

```
[ ] svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier(n_neighbors=49)
mnb = MultinomialNB(alpha=0.2)
dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=31, random_state=111)
abc = AdaBoostClassifier(n_estimators=62, random_state=111)
bc = BaggingClassifier(n_estimators=9, random_state=111)
etc = ExtraTreesClassifier(n_estimators=9, random_state=111)

[ ] clf = {'SVC': svc, 'KN': knc, 'NB': mnb, 'DT': dtc, 'LR': lrc, 'RF': rfc, 'AdaBoost': abc, 'BgC': bc, 'ETC': etc}
```

Gamma and accuracy:

Accuracy with gamma 1.0:

	0	1
0	SVC	[0.9784688995215312]
1	KN	[0.9246411483253588]
2	NB	[0.9844497607655502]
3	DT	[0.958732057416268]
4	LR	[0.9431818181818182]
5	RF	[0.9700956937799043]
6	AdaBoost	[0.9671052631578947]
7	BgC	[0.9665071770334929]
8	ETC	[0.9772727272727273]

WARNING:root:Quickchart encountered

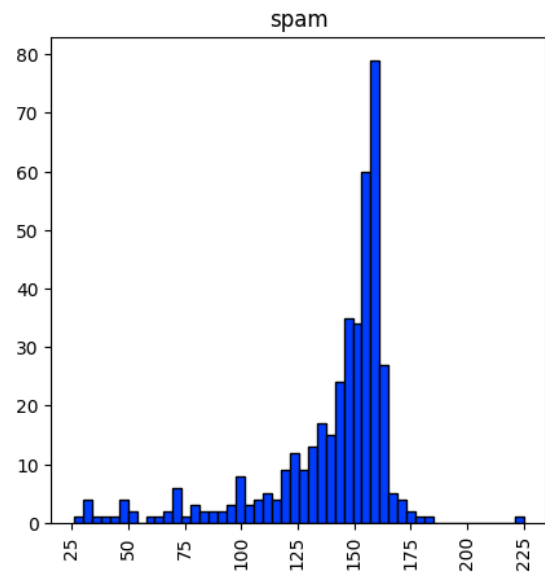
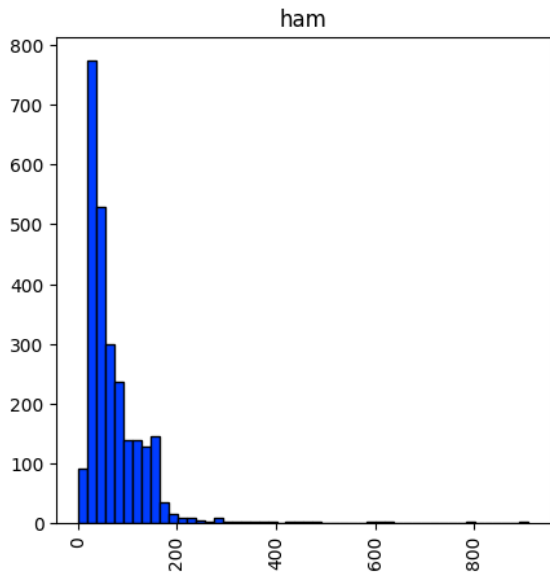
Changing gamma from 1.0 to .01:

	0	1
0	SVC	[0.861244019138756]
1	KN	[0.9246411483253588]
2	NB	[0.9844497607655502]
3	DT	[0.958732057416268]
4	LR	[0.9431818181818182]
5	RF	[0.9700956937799043]
6	AdaBoost	[0.9671052631578947]
7	BgC	[0.9665071770334929]
8	ETC	[0.9772727272727273]

Changing the dataset size without altering any other parameters

	0	1
0	SVC	[0.9722222222222222]

time: 30.3 ms (started: 2024-03-29 10:36:10 +00:00)



Metric speed:

To test the speed of svc testing, other classifiers imported were removed allowing us to see the speed of svc through the classifiers iteration code. The speed comes out to 1.29s for a large dataset of over 5000 cells.

```
clfs = {'SVC' : svc}

time: 587 µs (started: 2024-03-29 09:59:25 +00:00)
```

```
[48] pred_scores = []
     for k,v in clfs.items():
         train_classifier(v, features_train, labels_train)
         pred = predict_labels(v,features_test)
         pred_scores.append((k, [accuracy_score(labels_test,pred)]))

time: 1.29 s (started: 2024-03-29 09:59:25 +00:00)
```

```
0 SVC [0.9784688995215312]
time: 38.9 ms (started: 2024-03-29 09:59:27 +00:00)
```

[Discussion of metrics](#)

- Accuracy: the Sigmoid kernel tends to stay closer to the extremes (1.0 and 0), this provides strong variance in spam detection that aids in prioritising detection with the disadvantage of false positives. This can be shown through the gamma metric which heavily affects the accuracy. Furthermore, the size of the dataset created a minor drop in accuracy from 0.98 to 0.97; resulting in the advantage of svc in classifying data accurately regardless of large changes within the size of the dataset.
- Variance: gamma additionally affects the variance and classification as it represents the impact/relation each point has on another. This means that on a hyperplane, points may be closer to the decision line or overlap. This is useful for prioritising the algorithms detection such as more aggressive spam detection with more false positives.
- Speed: For limited specifications in energy consumption and computing power, speed becomes an increasingly important metric when considering SVM. Due to the wide application, research and documentation of SVM, it is a highly efficient supervised learning algorithm method; paired with mathematically sound equations evident within the kernels e.g. the sigmoid kernel (Sitorus, I. (2020, September 23) right:

$$K(x, x_i) = \tanh(\alpha x_i \cdot x_j + \beta)$$
 SVM becomes feasible for low to high end hardware thus not being restricted even with large datasets.

Reference

- Assem Utaliyeva, Millati Pratiwi, Park, H., & Choi, Y.-H. (2023). ChatGPT: A Threat to Spam Filtering Systems. 2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application

(HPCC/DSS/SmartCity/DependSys).

<https://doi.org/10.1109/hpcc-dss-smartcity-dependsys60770.2023.00150>

- Garg, P., & Girdhar, N. (2021). A Systematic Review on Spam Filtering Techniques based on Natural Language Processing Framework. 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence).
<https://doi.org/10.1109/confluence51648.2021.9377042>
- Lumpur, K. (2020, January 9). 91% of all cyber attacks begin with a phishing email to an unexpected victim | Deloitte Malaysia | Risk Advisory | Press releases. Deloitte Malaysia.
<https://www2.deloitte.com/my/en/pages/risk/articles/91-percent-of-all-cyber-attacks-begin-with-a-phishing-email-to-an-unexpected-victim.html>
- Oluchukwu, U. W., Okwudili, A. S., Chinedu, A. D., Asogwa, E. C., Sylvanus, A. K., & Department of Computer Science, Nnamdi Azikiwe University Awka, Nigeria. (2024). View of Hybrid Machine Learning Algorithms for email and malware spam filtering: a review. European Journal of Theoretical and Applied Sciences, 2(2).
[https://doi.org/10.59324/ejtas.2024.2\(2\).07](https://doi.org/10.59324/ejtas.2024.2(2).07)
- Sitorus, I. (2020, September 23). Introduction to SVM and Kernel Trick — Part 1 (Theory). Analytics Vidhya.
<https://medium.com/analytics-vidhya/introduction-to-svm-and-kernel-trick-part-1-theory-d990e2872ace>