# COS720 Project

u21456552 - Ivan Horak

May 17, 2025

## Contents

# 1 Research

## 1.1 How is AI used to detect phishing

There are a variety of different ways that AI is used to detect phishing attacks. Some of the more prevalent methods are the use of Machine Learning models which are trained on large datasets to detect and identify patterns in phishing attacks, Natural Language Processing is used to understand the context and intent of communications and to detect manipulation strategies often used by attackers like urgency. AI is also used not only to identify is some form of communication is a phishing attack but also to identify comprised accounts and victims of attacks by monitoring user behaviour for anomalies. Another way that it is used is to identify phishing attacks that are done by imitating legitimate websites and people, Computer Vision is often used for this to identify anomalies in logos or differences compared to legitimate companies and people. These different strategies for using AI to fight phishing are implemented in many different ways, they could be implemented on the server side to detect user anomalies, or they could be a hosted AI with an API endpoint that a plugin could send data to to get it classified and lastly it could be a feature built into a tool or application like an email client.[1]

These have a lot of advantages compared to traditional detection methods as traditional methods rely on humans to define the qualifiers for what is a phishing attack and then also implement it, however with AI it can adapt and learn new strategies and even identify unseen phishing tactics that would usually take a long time to develop using traditional methods.

## 1.2 Phishing Detection using AI

Phishing detection using AI can be defined as the use of any AI strategy to detect and warn a user or stakeholder about any phishing attacks, especially in email based phishing attacks, which are one of the most common, AI detection would be the method that a tool uses to analyze an email and make a prediction about an email on if it is legitimate or not. AI's role in this is to warn users of phishing attempts before it is too late, especially users that would otherwise not know better.

## 2 What is Phishing and How I Detect it

To best understand how to implement an AI detection system you need to understand what is a phishing attack, what are the strategies used in phishing attacks and what to look out for.

Below are the attributes of a phishing email and the signs I look out for:

- Does the email contain in links.
- Is the email trying to get me to do something
- Does the domain of the email address match the claimed domain. (i.e does the email address belong to the same domain as the receiving email address)
- Does the name the email set match the name in the email address
- Does the email contain phrases like, change password or banking details

Given these qualities of a phishing email they are the qualities that I trained my model on.

To do these things I have the following data points:

## 3 Model Selection Process

While there are lots of different AI models and techniques that are used to detecting phishing there are a couple that show up the most in discussions and implementations. The three that are most prevalent are: Naive Bayes, Logistic Regression and Random Forest. These are the three that I considered for my model.

I considered Naive Bayes because it works on the probability of a set of features to be in the same class together and assumes that the features are independent of each other, which is good for phishing detection since there are so many different strategies that are used for attacks and different combinations of strategies. So for my model it needed to be able to consider data such that it picks up the most common features and the features being able to come from different emails in the training dataset.

Since an email is either a phishing email or not, that binary relationship made me consider logistic regression as it takes in multiple independent variables and then a dependent binary variable used for categorization. The binary dependent variable being if it is phishing or not.

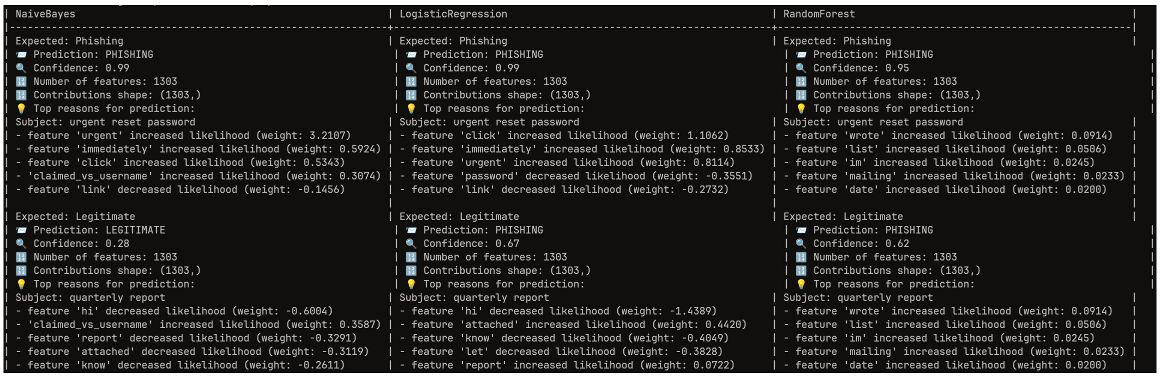
Finally I considered Random Forest because of how it can rank features on how important they were in making a prediction, this is useful as it will naturally find the features in an email that are most likely to used in a phishing attack which is needed for phishing detection as sometimes it is only one or two things in an email that can give away if it is phishing or not and the nature of the random forest being able to take advantage of this is important.

For my implementation I had a couple key requirements, First that it is able to take in multiple data points, the second that it not be too much overhead to implement and lastly that there is plenty of documentation on implementation. This caused me to land on using python for training my model. This is important for my model selection as it enabled me to rapidly iterate and modify my code to evaluate different models performances.

Due to pythons ease of use and specifically the scikit-learn package I was able to implement a program that would train all three models at the same time, then it would feed all three the same data that didn't appear in the datasets used for training. With this I could easily compare the different models perfomances and how often each model correctly predicted if the email was legitimate or not.

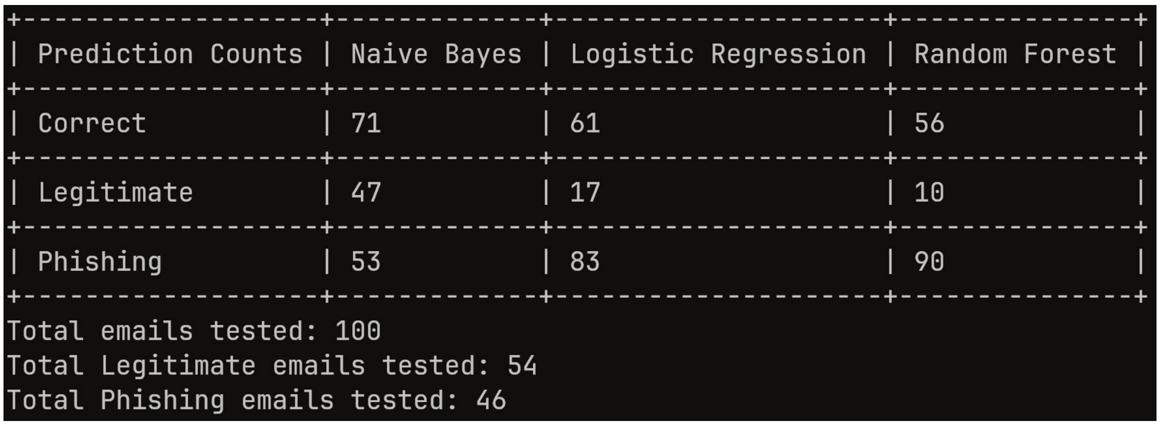
With this in mind I could then iterate on the training methods and see how the different models compared and why they made different predictions. And using a cli implementation I got the models to output the reasoning side by side for easy comparison. As seen below.

With this information and a simple switch I implemented to toggle it to machine readable output, I could also get a comparison of how many each model got correct. As seen below.



```
| NaiveBayes                                                     | LogisticRegression                                             | RandomForest                                                   |
|----------------------------------------------------------------+----------------------------------------------------------------+----------------------------------------------------------------|
| Expected: Phishing                                             | Expected: Phishing                                             | Expected: Phishing                                             |
| 📨 Prediction: PHISHING                                         | 📨 Prediction: PHISHING                                         | 📨 Prediction: PHISHING                                         |
| 🔍 Confidence: 0.99                                             | 🔍 Confidence: 0.99                                             | 🔍 Confidence: 0.95                                             |
| 🔢 Number of features: 1303                                     | 🔢 Number of features: 1303                                     | 🔢 Number of features: 1303                                     |
| 🔢 Contributions shape: (1303,)                                 | 🔢 Contributions shape: (1303,)                                 | 🔢 Contributions shape: (1303,)                                 |
| 💡 Top reasons for prediction:                                  | 💡 Top reasons for prediction:                                  | 💡 Top reasons for prediction:                                  |
| Subject: urgent reset password                                 | Subject: urgent reset password                                 | Subject: urgent reset password                                 |
| - feature 'urgent' increased likelihood (weight: 3.2107)       | - feature 'click' increased likelihood (weight: 1.1062)        | - feature 'wrote' increased likelihood (weight: 0.0914)        |
| - feature 'immediately' increased likelihood (weight: 0.5924)  | - feature 'immediately' increased likelihood (weight: 0.8533)  | - feature 'list' increased likelihood (weight: 0.0506)         |
| - feature 'click' increased likelihood (weight: 0.5343)        | - feature 'urgent' increased likelihood (weight: 0.8114)       | - feature 'im' increased likelihood (weight: 0.0245)           |
| - 'claimed_vs_username' increased likelihood (weight: 0.3074)  | - feature 'password' decreased likelihood (weight: -0.3551)    | - feature 'mailing' increased likelihood (weight: 0.0233)      |
| - feature 'link' decreased likelihood (weight: -0.1456)        | - feature 'link' decreased likelihood (weight: -0.2732)        | - feature 'date' increased likelihood (weight: 0.0200)         |
|                                                                |                                                                |                                                                |
| Expected: Legitimate                                           | Expected: Legitimate                                           | Expected: Legitimate                                           |
| 📨 Prediction: LEGITIMATE                                       | 📨 Prediction: PHISHING                                         | 📨 Prediction: PHISHING                                         |
| 🔍 Confidence: 0.28                                             | 🔍 Confidence: 0.67                                             | 🔍 Confidence: 0.62                                             |
| 🔢 Number of features: 1303                                     | 🔢 Number of features: 1303                                     | 🔢 Number of features: 1303                                     |
| 🔢 Contributions shape: (1303,)                                 | 🔢 Contributions shape: (1303,)                                 | 🔢 Contributions shape: (1303,)                                 |
| 💡 Top reasons for prediction:                                  | 💡 Top reasons for prediction:                                  | 💡 Top reasons for prediction:                                  |
| Subject: quarterly report                                      | Subject: quarterly report                                      | Subject: quarterly report                                      |
| - feature 'hi' decreased likelihood (weight: -0.6004)          | - feature 'hi' decreased likelihood (weight: -1.4389)          | - feature 'wrote' increased likelihood (weight: 0.0914)        |
| - 'claimed_vs_username' increased likelihood (weight: 0.3587)  | - feature 'attached' increased likelihood (weight: 0.4420)     | - feature 'list' increased likelihood (weight: 0.0506)         |
| - feature 'report' decreased likelihood (weight: -0.3291)      | - feature 'know' decreased likelihood (weight: -0.4049)        | - feature 'im' increased likelihood (weight: 0.0245)           |
| - feature 'attached' decreased likelihood (weight: -0.3119)    | - feature 'let' decreased likelihood (weight: -0.3828)         | - feature 'mailing' increased likelihood (weight: 0.0233)      |
| - feature 'know' decreased likelihood (weight: -0.2611)        | - feature 'report' increased likelihood (weight: 0.0722)       | - feature 'date' increased likelihood (weight: 0.0200)         |
```

Figure 1: CLI comparison



```
+-------------------+-------------+---------------------+---------------+
| Prediction Counts | Naive Bayes | Logistic Regression | Random Forest |
+-------------------+-------------+---------------------+---------------+
| Correct           | 71          | 61                  | 56            |
+-------------------+-------------+---------------------+---------------+
| Legitimate        | 47          | 17                  | 10            |
+-------------------+-------------+---------------------+---------------+
| Phishing          | 53          | 83                  | 90            |
+-------------------+-------------+---------------------+---------------+
Total emails tested: 100
Total Legitimate emails tested: 54
Total Phishing emails tested: 46
```

Figure 2: Model Comparison

This led me to choose the Naive Bayes Model as it correctly identifier the most amount of emails. It also did not drastically prefer either predicting an email to be phishing or legitimate.

## 4 Design

### 4.1 Function Requirements

**R1** Model Training

**R1.1** Dataset Preprocessing

**R1.1.1** Url detection

**R1.1.2** Domain Extraction

**R1.1.3** Sender vs Receiver Domain comparison

**R1.1.4** Sender email username vs sender claimed identity

**R1.2** Text Cleaning

**R1.2.1** Remove HTML from email

**R1.2.2** Remove URLS

**R1.2.3** Remove stop words

**R1.2.4** Remove punctuation

**R1.3** Model Attributes

**R1.3.1** Subject, body, sender, receiver attributes

- **R1.3.2** Url presence
- **R1.3.3** Are the sender and receiver domains the same
- **R1.3.4** Is the sender who they claim to be (from username vs name)

**R1.4** Train Model

- **R1.4.1** Dump Model
- **R1.4.2** Train model on fitness using dataset

**R2** Email Analysis/Prediction

**R2.1** Email ingest

- **R2.1.1** Extract url presence from email
- **R2.1.2** Extract sender, receiver and identities from email
- **R2.1.3** Check domains and claimed identity

**R2.2** Prediction output

- **R2.2.1** Should output if email is legitimate or phishing
- **R2.2.2** Should give confidence of prediction
- **R2.2.3** Give features of email and how they affected prediction

**R2.3** Prediction reasoning

- **R2.3.1** Should give top 5 reasons for prediction
- **R2.3.2** Should give weighting of reason
- **R2.3.3** Should say if it increased or decreased likelihood of phishing

**R3** Frontend

**R3.1** Email input

- **R3.1.1** Should have input for raw email text
- **R3.1.2** Should have input for structured email

**R3.2** Result output

- **R3.2.1** Should display if email is phishing or not
- **R3.2.2** Should display confidence
- **R3.2.3** Should display top reasons
- **R3.2.4** Should display how it affected prediction

## 4.2 Security Requirements

**S1** API/AI model security

**S1.1** Sanitize/Validate data ingest

- **S1.1.1** Limit size of input
- **S1.1.2** Remove any HTML/scrips
- **S1.1.3** Validate input, e.g. email body, subject, etc.

**S1.2** Application logging

- **S1.2.1** Log out activity
- **S1.2.2** Log out errors and exceptions
- **S1.2.3** Log out access/endpoint usage

- **S1.3** Backend Monitoring
  - **S1.3.1** Use APM for activity traces
  - **S1.3.2** Use log ingester like elastic search
  - **S1.3.3** Get metrics using elastic agent

**S2** Frontend security

- **S2.1** Sanitize inputs
  - **S2.1.1** Limit text input lengths
  - **S2.1.2** Validate inputs, e.g. length, type
- **S2.2** Application logs
  - **S2.2.1** Log file for user activity
  - **S2.2.2** Log file for errors
  - **S2.2.3** Logs for metrics
- **S2.3** DOM purification
  - **S2.3.1** DOM purifier to prevent XSS
  - **S2.3.2** DOM validation/static DOM
- **S2.4** Frontend Monitoring
  - **S2.4.1** Use Real User Monitoring (RUM) APM for user observability on the client side
  - **S2.4.2** Get metrics for frontend server
  - **S2.4.3** Ingest frontend logs into elastic search

## 4.3 Use Case Diagrams

### 4.3.1 Model Training

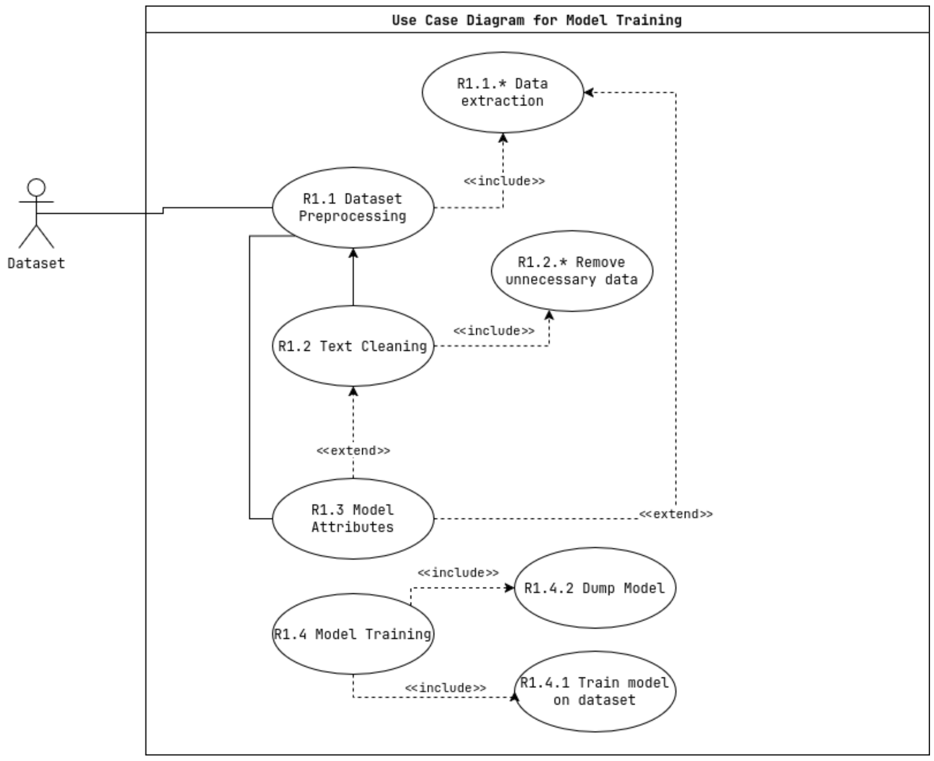


Figure 3: Use Case Diagram on how the model is trained

### 4.3.2 Model Running/API/Backend

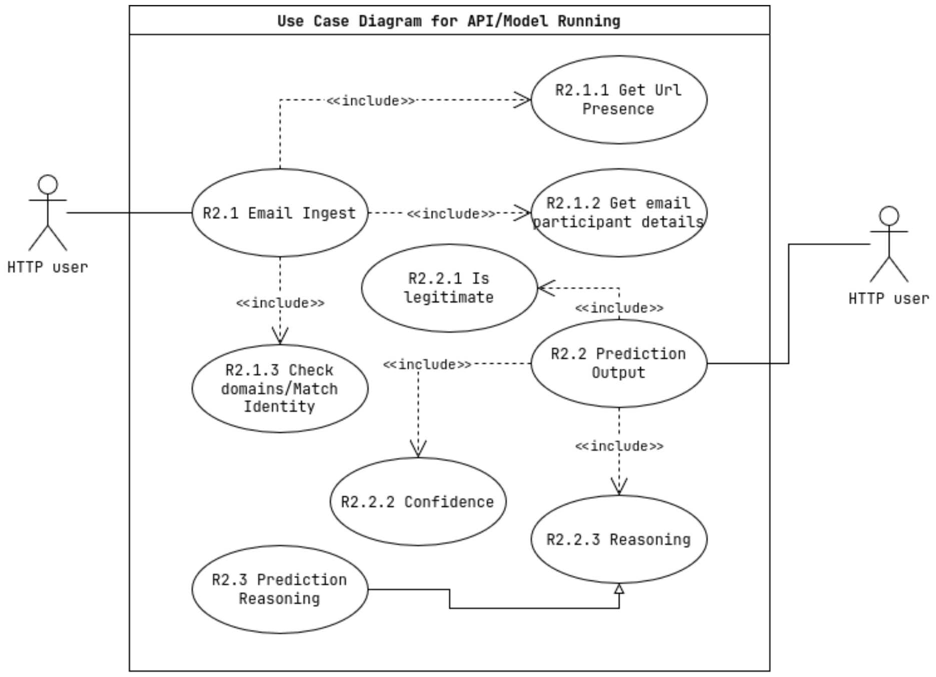


Figure 4: Use Case Diagram for how the Model is run and how the backend works

### 4.3.3 Frontend

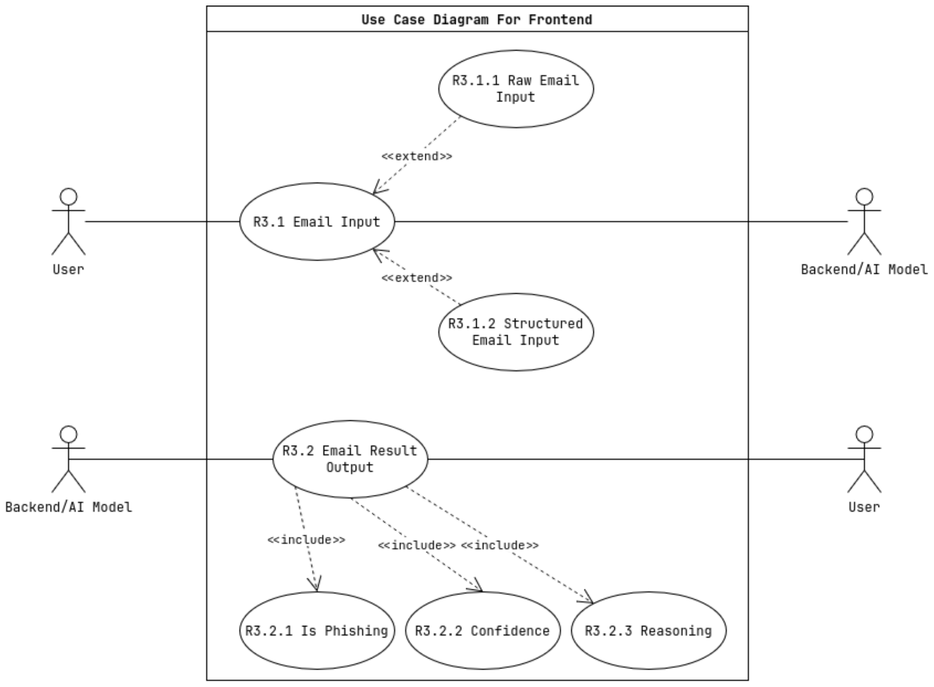


Figure 5: Use Case Diagram for how the frontend

## 4.4 Architectural Diagram

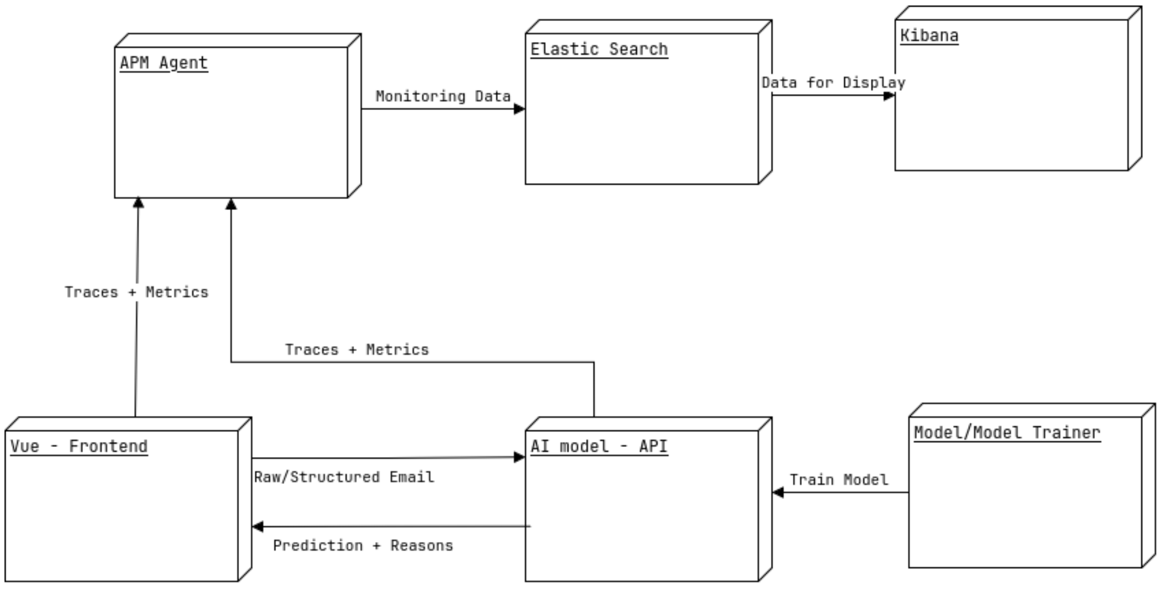


Figure 6: Architectural Diagram

# 5 How the Model works and what it does

## 5.1 Preprocessing

In the preprocessing step I ingest the data from multiple datasets from the phishing dataset provided to use. In this ingestion I first load and normalize the data in a loop over the different csv's. I load the data into dataframes from pandas.

The first step in my preprocessing is to detect if there is a URL in the email, first by checking if that was provided by the dataset and if not the body and subject are scanned to detect the presence of any URLs.

The next step is splitting out the senders domain and receivers domain and then adding a boolean to the dataframe entry which represents whether the sender's domain is the same as the receiver's domain. This is to train it for phishing attacks that come from outside an organization.

Next the senders claimed name (i.e the part of the sender header before the email address, e.g. ”From: John Doe ¡john.doe@company.com¿”) as well as the username of the email address are extracted, then a similarity is calculated using the levenshstein distance of the strings which is the number of single-character edits required to change one word into another. This is usefull for training the model on phishing detection as many phishing tactics involve pretending to be someone else. The levenshstein distance is used to account for the fact that rarely does someones claimed name fully match their username.

Finally I clean and remove all stopwords from the subject and body of the email. This is to ensure that only words and details that impact the meaning and intent of the email are left over and are used in the training process. This is important as if the text wasn't cleaned then the most common words will effect the models predictions in negative ways.

## 5.2 Model output - confidence/feature highlights

The main outputs for the model/api are the prediction which is either ”phishing” or ”legitimate” these are based on the confidence, if the confidence is higher than 0.5 then the email is labled as phishing else it is labled as legitimate. Along with the prediction the confidence is also output so that a user would be able to evaluate the chances of it being phishing or not.

The other thing that is output is the top 5 features that contributed to the prediction along with if the feature increased or decreased the likelihood of the email being phishing and finally the weighting by which the feature effected the prediction.

The different kinds of features that could be outputed are the different words that contributed, if the claimed vs actual identity were the same, the presence of a url and if the domains matched.

## 5.3 Model Performance

In the development of the model I made a csv with 100 entries, made up of a mixture of legitimate and phishing emails, none of these emails appeared in the training data.

This csv of emails was used to evaluate how well the model performed over iterations and allowed me to compare the performance and accuracy of different models, either previous iterations of the model or completely different machine learning strategies. Using this dataset I managed to get my final model to correctly identity 71 of the 100 emails. This is a high success rate considering the complexity of the task. The model also tended to identify emails as phishing rather than legitimate most of the time. This is preferable as it is better to tend to the side of caution rather than labeling emails as legitimate when they aren't.

## 5.4 Model Performance Metrics

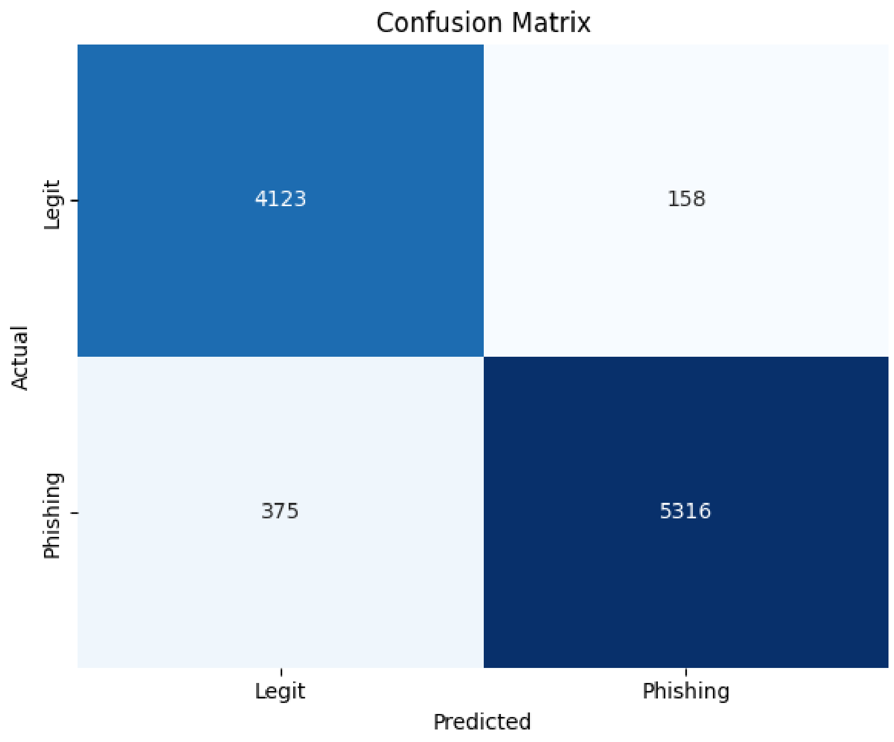


Figure 7: Naive Bayes Confusion Matrix

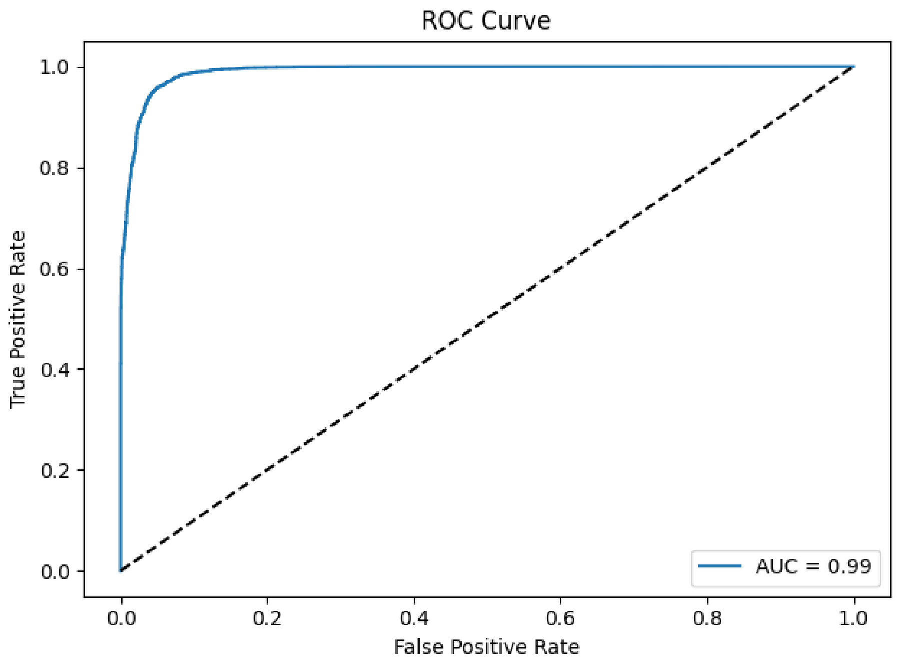


Figure 8: Naive Bayes ROC Curve

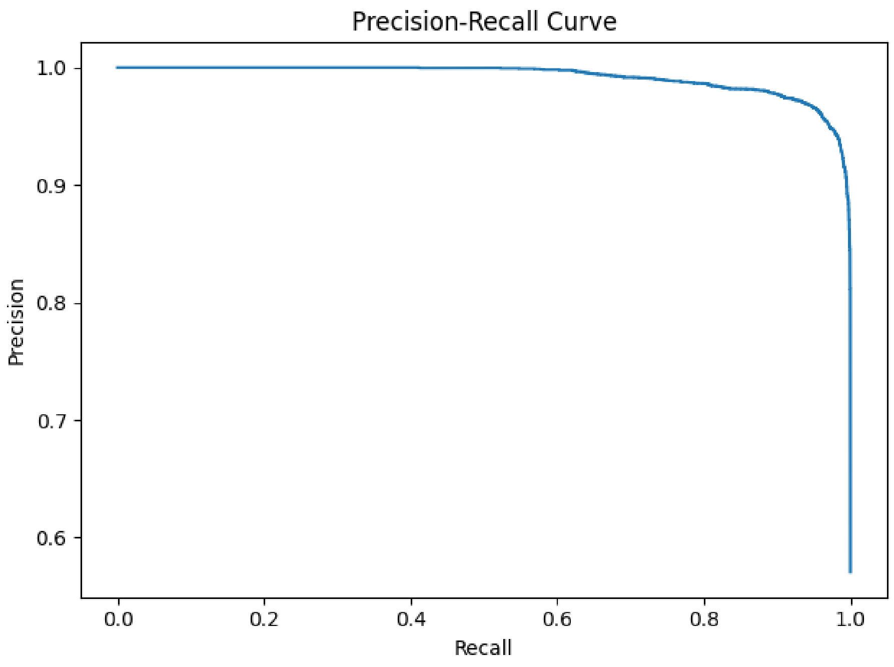


Figure 9: Naive Bayes Precision Recall Curve

## 6 Security Features

### 6.1 Frontend

To ensure that the frontend of my application is secure I tried to follow and treat the application as if it would actually be deployed. With this in mind the following are security features I implemented on the Vue frontend:

- I used the dompurify package which is an XSS sanitizer for HTML
- I did input validation on the inputs for the raw email input and structured email inputs
- I limited input lengths to the lengths found in the RFC documents for emails

I also added some observability features to better monitor the application for possible attacks or vulnerabilities.

- I used the RUM APM monitoring from elastic search to get client side metrics and traces.
- I also used elastic search to monitor the machine metrics that the application is running on for high resource usage.
- I added logging to a file to ensure that if the application goes down or there is unusual activity there is some trace to follow for an investigation.
- I also integrated with kibana to view the metrics and service traces as well as any logs that the application produces.

### 6.2 Backend

For the backend there is also a variety of security features that I implemented, including logging and error handling.

The most important security features for the running of the backend are:

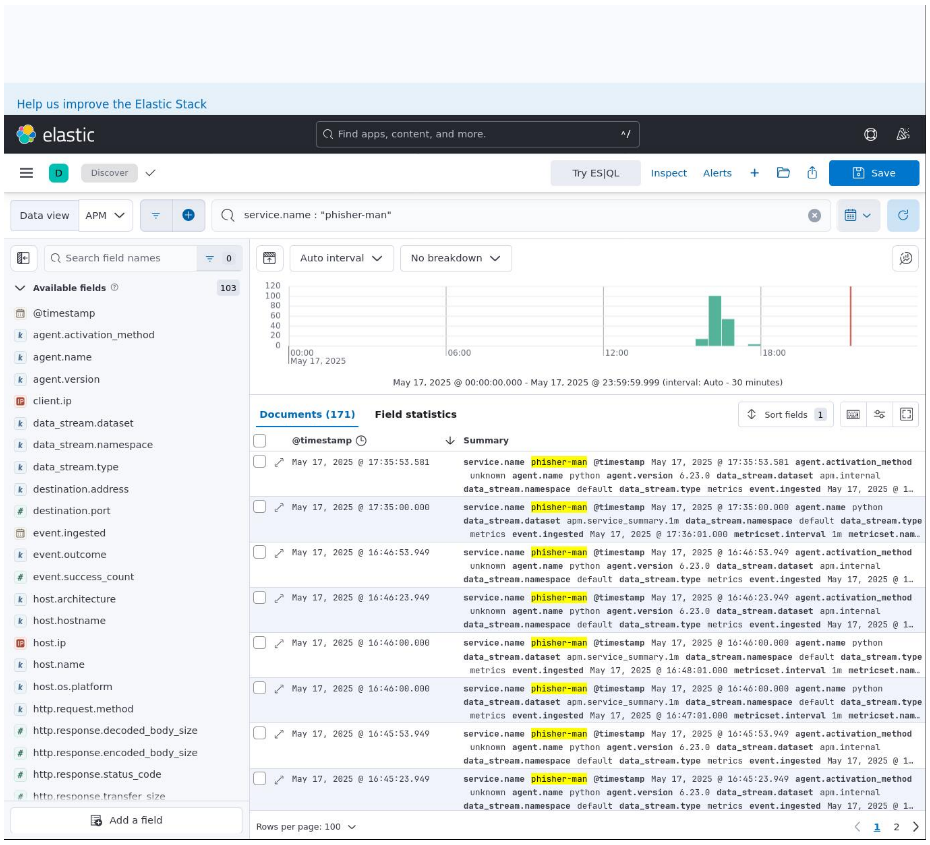
- Proper error handling and exception handling.
- Type checking for inputs.

- I also added input sanitization and validation with the same kinds of rules as the inputs for the frontend.

I also wanted to ensure proper observability for the backend so to ensure that any unusual activity is monitored and detected I integrated it using the FastApi/Starlette Elastic APM agent and I added logging to a file for proper persistence of any activity. The main features of these are:

- Elastic Search database for log ananlysis and storage
- Kibana for metrics observability and ananlysis
- Loggin to a file including timestamps, requests and other important details.

## 6.3 ElasticSearch and Kibana



service.name : "phisher-man"

Available fields 103

@timestamp
agent.activation_method
agent.name
agent.version
client.ip
data_stream.dataset
data_stream.namespace
data_stream.type
destination.address
destination.port
event.ingested
event.outcome
event.success_count
host.architecture
host.hostname
host.ip
host.name
host.os.platform
http.request.method
http.response.decoded_body_size
http.response.encoded_body_size
http.response.status_code
http.response.transfer_size

May 17, 2025 @ 00:00:00.000 - May 17, 2025 @ 23:59:59.999 (interval: Auto - 30 minutes)

Documents (171) Field statistics Sort fields 1

```
@timestamp                    Summary
May 17, 2025 @ 17:35:53.581   service.name phisher-man @timestamp May 17, 2025 @ 17:35:53.581 agent.activation_method
                               unknown agent.name python agent.version 6.23.0 data_stream.dataset apm.internal
                              data_stream.namespace default data_stream.type metrics event.ingested May 17, 2025 @ 1…
May 17, 2025 @ 17:35:00.000   service.name phisher-man @timestamp May 17, 2025 @ 17:35:00.000 agent.name python
                              data_stream.dataset apm.service_summary.1m data_stream.namespace default data_stream.type
                               metrics event.ingested May 17, 2025 @ 17:36:01.000 metricset.interval 1m metricset.nam…
May 17, 2025 @ 16:46:53.949   service.name phisher-man @timestamp May 17, 2025 @ 16:46:53.949 agent.activation_method
                               unknown agent.name python agent.version 6.23.0 data_stream.dataset apm.internal
                              data_stream.namespace default data_stream.type metrics event.ingested May 17, 2025 @ 1…
May 17, 2025 @ 16:46:23.949   service.name phisher-man @timestamp May 17, 2025 @ 16:46:23.949 agent.activation_method
                               unknown agent.name python agent.version 6.23.0 data_stream.dataset apm.internal
                              data_stream.namespace default data_stream.type metrics event.ingested May 17, 2025 @ 1…
May 17, 2025 @ 16:46:00.000   service.name phisher-man @timestamp May 17, 2025 @ 16:46:00.000 agent.name python
                              data_stream.dataset apm.service_summary.1m data_stream.namespace default data_stream.type
                               metrics event.ingested May 17, 2025 @ 16:48:01.000 metricset.interval 1m metricset.nam…
May 17, 2025 @ 16:46:00.000   service.name phisher-man @timestamp May 17, 2025 @ 16:46:00.000 agent.name python
                              data_stream.dataset apm.service_summary.1m data_stream.namespace default data_stream.type
                               metrics event.ingested May 17, 2025 @ 16:47:01.000 metricset.interval 1m metricset.nam…
May 17, 2025 @ 16:45:53.949   service.name phisher-man @timestamp May 17, 2025 @ 16:45:53.949 agent.activation_method
                               unknown agent.name python agent.version 6.23.0 data_stream.dataset apm.internal
                              data_stream.namespace default data_stream.type metrics event.ingested May 17, 2025 @ 1…
May 17, 2025 @ 16:45:23.949   service.name phisher-man @timestamp May 17, 2025 @ 16:45:23.949 agent.activation_method
                               unknown agent.name python agent.version 6.23.0 data_stream.dataset apm.internal
                              data_stream.namespace default data_stream.type metrics event.ingested May 17, 2025 @ 1…
```

Rows per page: 100 1 2

Figure 10: Kibana log streams

# 7 Code Base

## 7.1 Dependencies

## 7.2 Docker

To be able to run this project with the APM monitoring make sure to run the command 'docker compose up -d' in the home directory of this project.

### 7.2.1 Frontend

The project is split into two areas for proper separation of concerns. The first area is the frontend which is a Vue project since it is one of the easiest frontend frameworks to work with and the npm packages mean that implementing and maintaining features should be easy. Vue is also one of the most modern frameworks and has a very active community. This means that it is well maintained and most security concerns have already been addressed or there is a package for it.

To install the dependencies for the vue project, ensure that you have nodejs installed and then cd into the email-parser-ui directory and run 'npm install'.

#### 7.2.2 Backend

For the backend I used python as it has a lot of libraries for machine learning and also it has a very simple recognisable syntax which makes it easy to understand and code and maintain. I also used FastApi in python to create the endpoints that are used for the frontend.

To install the dependencies I recommend using 'uv' which is a python virtual environment manager written in rust for speed and easy setup. To install all the dependencies with uv run the command 'uv sync'. Then to activate the virtual environment run 'source .venv/bin/activate'. This will make all the packages and tools for the project available.

### 7.3 Starting the applications

#### 7.3.1 Frontend

To start the Vue frontend run the command 'npm run dev'.

#### 7.3.2 Backend

Now to start the API run 'uvicorn app.main:app –reload' this will start the API.

### 7.4 Accessing the UI

To be able to see the frontend there should be a link in the output from starting the vue frontend that will serve the vue frontend. The python backend is always accessible by 'http://localhost:8000/' and the swagger docs are accessible at 'http://localhost:8000/doc'.

The kibana portal is accessible at 'http://localhost:8200'.

### 7.5 Code Repository

The code in this project is accessilbe at 'https://github.com/ChuufMaster/phisher-man'.

## 8 Bibliography

## References

[1] G. Baran, “Ai-powered phishing detection – does it actually work?.” https://cybersecuritynews.com/ai-powered-phishing-detection/, CISO, CYBER AI, Cyber Security, Cyber Security News.