Project Progress Timeline

# September 27

* Team meeting

# September 29

* Start of the project

# September 30

* Email body preprocessing
  + Duplicates
  + Null values
  + Tokenization
  + Lemmatization

# October 3 – October 4

* Email body feature engineering
* Talked about the potential issue of imbalanced classes
* Talked about the use of different models to consider

# October 10

* Defined Logistic Regression model for the meeting
  + Accuracy 92%

# October 11

* Team meeting
* Talked about also have URL prediction model

# October 15 – October 17

* Talked about different innovations to incorporate to the project
  + XAI (CBR, SHAP, LIME)
  + Use of stacking models
  + Continual learning
  + How to avoid poisoning text and obfuscated words
  + BERT/DistilBERT
  + EML files
  + File analysis using entropy
  + Page ranking
* Made a flow chart to visualize the process from the input email to the output result

A diagram of a company

Description automatically generated

# October 18

* Planned new tasks before meeting on November 1st
* Team meeting

A screenshot of a computer program

Description automatically generated

# October 20

* Started URL prediction code

# October 24 – October 27

* Did preprocessing for a dataset, but found many duplicates
* Wrote code to extract information from EML files
  + Email body
  + Email subject
  + Sender’s IP address
* Defined a regex to capture links
* Created the website
* Considered checking for misspelled words in emails, but found the accuracy was worse
  + Decided not to consider misspelled/obfuscated words
* Got accuracies of 85-87% for URL predictions
  + Increased the accuracy to 91% using a stacking model

# October 29 – October 30

* Made functions to extract the features from a URL to make predictions
* Found a problem where the functions extract the features correctly 100-70% of the time
  + Seems like the dataset contains inconsistent data
* Found a new dataset to perform URL prediction with an accuracy of 98%
* Updated the EML preprocessing code to better extract the body of an email
* Wrote functions to extract features from links for the new dataset
  + Found a similar issue where the dataset inconsistently gets features
  + Mostly get around 100-99.99%, but also get 60-70% for some of the features
* Upgraded the stacking model for email predictions
* Talked about potentially modifying the URL dataset by replacing the feature with the result of our functions to ensure we can consistently extract URL features
* Found a page ranking dataset (Top 10 Million Domains)

# October 31

* Worked on the website
* Found an issue, where the URL model would overfit with many duplicates if we removed the “URL” column
  + Decided to keep the “URL” column and to use TF-IDF on that column to do NLP analysis as well as use the link features for predictions
  + Got a testing accuracy of 99.5%
* Tested the new URL prediction model on 2 separate datasets
  + Found that it accurately predicted 45-55% of time
  + Decided to combine the original dataset with the 2 other datasets, and got an accuracy of 91% using Logistic Regression

# November 1

* Team meeting
* Found an issue where the website wouldn’t “clean” the EML email body, and it resulted in bad results because the model would make predictions on the body which contained HTML code
* Talked about fixing 2 problems:
  + Fixing the EML preprocessing to make predictions from an EML file
  + Making predictions on links
* Fixed the issues, and now the website can make prediction by either providing an EML file, or by pasting the email directly to the website
* Talked about creating our own dataset with our own features which will be extracted from different functions

# November 2 – November 3

* Finished code for calculating the page ranking score of links
  + Decided not to use this score as a feature for the dataset since it takes more than 15 seconds per link
  + It would take too much time to compile a dataset of 50,000 links
* Talked about potentially combining this page ranking score with the final prediction score from the model
  + Still unsure about how to combine the score and unsure how we can reduce the runtime of the algorithm
* Created functions to extract features from a link

# November 4

* Compiled the dataset using 50,000 links with the new features and got an accuracy of 93.5% using a stacking model
  + Took 588 minutes to compile the dataset
* Talked about potentially increasing the accuracy of the model by creating a new dataset with more features (more functions)
* Talked about how we can combine the score of the email body prediction and the URL prediction
  + Decided to consider 70% of the URL prediction score and 30% of the email body prediction since we believe that the URL prediction is more important to predict phishing/malicious emails

# November 5

* Started compiling a new dataset with new features
  + Got a compilation error after 1120 minutes
  + Fixed the issue (missing comma) and compiled it again
    - Made 2 other version (another dataset of 50,000 and one of 75,000)
* Included the 93.5% URL prediction model to the website to see if it would work well
  + The model works on the website
* Talked about implementing CBR with FAISS or ANNOY

# November 6

* Compiled the new dataset with the added features
  + Got an accuracy of 98%

# November 7

* Updated the EML preprocessing code to only consider visible text in the email and to avoid hidden text
* Started to code function to check if a link is already considered phishing
  + If so, then we wouldn’t have to perform a prediction
* Started to code function to check if the IP of the email sender is a part of a list of known suspicious/blacklisted Ips
* Started to code function to perform analysis on email attached files

# November 8

* Team meeting (canceled)
* Compile the dataset of 75,000 links and got an accuracy of 98.3%
* Wrote code for case-based reasoning (CBR) using Faiss

# November 10 – November 11

* Modified the code for CBR
  + Had an issue where the exported file for Faiss was 17.5 Gb, and we were unable to upload it to GitHub
  + Used cosine similarity using sparse matrix (csr\_matrix) for a fast runtime
  + Using cosine similarity also allowed us to increase our parameters for the vectorizer (ngram\_range and max\_features) which increases the accuracy
    - The biggest file is now 427 Mb
* Created a file called get\_scores to compute different scores such as checking if an input is already in our database, computing the CBR score, checking if the sender’s IP is blacklisted, etc

# November 14

* Worked on file analysis
  + Now able to
    - Extract file extensions
    - Get size of files
    - Extract the content of pdf, docx, txt and json files
    - Get the entropy of a file
    - Check if a file has macros and if pdf files contain javascript

# November 15

* Team meeting