PHISHING EMAIL DETECTION PROGRESS REPORT

Asmita Chandra
Student# 1009051339
asmita.chandra@mail.utoronto.ca

Riya Kapoor
Student# 1009030558
riya.kapoor@mail.utoronto.ca

Joonseo Park
Student# 1008819281
joonseo.park@mail.utoronto.ca

Lee Martins
Student# 1008866835
leah.martins@mail.utoronto.ca

ABSTRACT

This progress report outlines the development of a phishing email detection system using a fine-tuned Bidirectional Encoder Representations from Transformers (BERT) model. This document reflects both technical progress and collaborative efforts made to meet project milestones. – Total Pages: 9

1 PROJECT DESCRIPTION

As communication migrates online, our project combats the increasing threat of phishing emails, with a deep learning system that accurately detects deceptive emails. Phishing emails often contain linguistic cues and patterns that traditional rule-based systems do not capture. Deep learning, specifically a Bidirectional Encoder Representation from Transformers (BERT) model, is well-suited because it learns such patterns in emails, allowing the model to make accurate predictions. Figure 1, below illustrates inputs including the raw email data, separate prediction architectures, and the output of the final model which is the accuracy of our predictions.

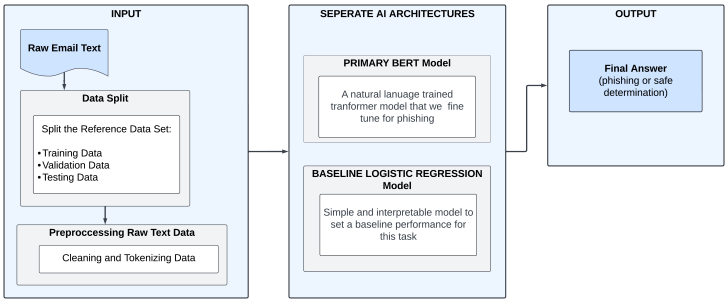

Figure 1: Explicit diagram of our AI model highlighting the BERT model

2 INDIVIDUAL RESPONSIBILITIES

The following section outlines the individual and team responsibilities that lead to our model.

2.1 WEEKLY WORKFLOW & COMMUNICATION:

- **Weekly Meetings:** Thursdays, 7–8pm EST (Discord)
- **Code Management:** GitHub & Colab (to avoid overwriting)
- **Task Updates:** Share progress in a shared Google Drive and discuss during meetings.

Our team has established a clear task division where progress is discussed and documented regularly.

Table 1: Individual and Team Responsibilities

Team Member	**Responsibilities Progress**
Asmita (Meeting Co-ordinator)	• Scheduled and led weekly meetings. • Completed data pre-processing and cleaning by June 26. • Applied and fine-tuned pre-trained BERT model by July 7.
Joon (Delivery Lead)	• Set up Hugging Face Transformers for tokenization by June 29. • Took comprehensive notes explaining BERT tokenization. • Co-implemented the logistic regression model with Riya by July 5.
Lee (Scribe)	• Documented meeting notes and action items. • Completed tokenization for baseline model (LR) by July 3. • Assisted in tokenization of the dataset with Riya by June 26.
Riya (Editor)	• Managed final edits ensuring alignment with the rubric. • Completed tokenization using Hugging Face with Lee by June 26. • Co-implemented the logistic regression model with Joon by July 5.

2.2 Project Management & Progress:

With consistent weekly updates, clear documentation, and issue tracking, the team is on track with all major deliverables. In case of technical or personal delays, members have committed to stepping in. The following table is the project plan from this stage onward.

Table 2: Project Task Schedule

	Task	**Internal Deadline**	**Description**
✓	Preprocessing Data (Tokenization)	June 26, 2025	Includes splitting into training/sample data
✓	Install and Import Hugging Face Transformers	June 29, 2025	Set up environment, test with sample data
✓	Baseline model using Logistic Regression: Bi-nomial Regression	July 5, 2025	Evaluate performance for comparison
✓	Apply Pretrained BERT model	July 7, 2025	Use pre-built models
✓	Progress Report	July 10, 2025	Provide updates on projects
	Fine-tune BERT (hyperparameter tuning)	July 20, 2025	Adjust learning rate, batch size, and other hyperparameters.
	Apply L2 Regularization	July 25, 2025	Compare performance with or without regularization
	Evaluate Model (accuracy)	August 1, 2025	Generate final metrics
	Final Configuration and Adjustments	August 8, 2025	Finalize project for submission
	Project Presentation	August 14, 2025	Create a pre-recorded video and demonstrate how the model performs.
	Final Deliverable	August 14, 2025	Final documentation of the entire project.

3 NOTABLE CONTRIBUTIONS

The following section outlines the contributions made regarding data processing, baseline model, and primary model.

3.1 DATA PROCESSING

Our model is trained on four diverse datasets from Kaggle to academic research datasets:

1. **Phishing No More Dataset (Kaggle)**, Labels: 1 = phishing, 0 = not phishing
   - Fields: body, sender, recipient, date, URLs, label, (Alam & Khandakar, 2024).
2. **University of Twente Dataset (Academic)**, Labels: 1 = phishing, 0 = not phishing
   - Fields: email text, email type (Miltchev et al., 2024).
3. **Cybercop Dataset (Kaggle)**, Labels: "phishing email" or "safe email"
   - Fields: email text, email type, (Chakraborty, 2023).
4. **Human-LLM Phishing Emails (Kaggle)**, Labels: 1 = phishing, 0 = not phishing
   - Fields: text, label, (Greco, 2024).

Across all sources of data we have a total of 98147 samples before cleaning.

3.1.1 DATA PROCESSING STEPS

1. **Source Collection:** All `.csv` files were loaded from a shared directory.
2. **Corruption Filtering:**
   - Files that failed to load due to encoding or malformed rows were skipped.
3. **Standardization of Columns:**
   - Column names like `"Email Text"` were renamed to `"body"`.
   - Column names like `"Email Type"` were renamed to `"label"`.
   - Labels `"phishing"/"safe"` were changed to binary: 1 = phishing and 0 = safe.
   - If the `urls` column was missing, it was added based on the presence of hyperlinks.
4. **Column Filtering:** Only the essential columns were retained: `body`, `urls`, and `label`.
5. **Output Format:**
   - Each cleaned dataset was saved as a separate sheet in a consolidated Excel file.
   - All sheets are concatenated into a single DataFrame named `phishing_df`.
6. **Final Cleaning Steps:**
   - Rows with missing values were dropped.
   - Data was shuffled randomly with a fixed seed (`random_state=42`).

After data cleaning, the dataset contains 67,126 samples, with 36,604 phishing and 30,522 safe. Figure 2, illustrates the distribution between the two classes.

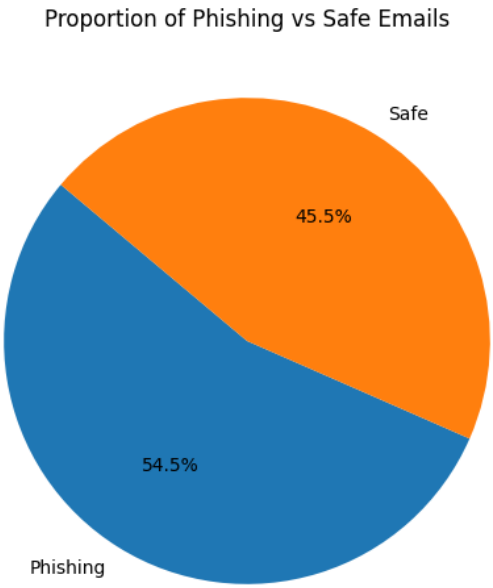

Figure 2: Percentage Division between Safe and Phishing Emails.

Along with labeling division it was found that from the set of emails, 38111 contained URLs and 29015 did not contain URLs. This presence of URLs across the phishing and non-phishing emails can be seen in Figure 3.

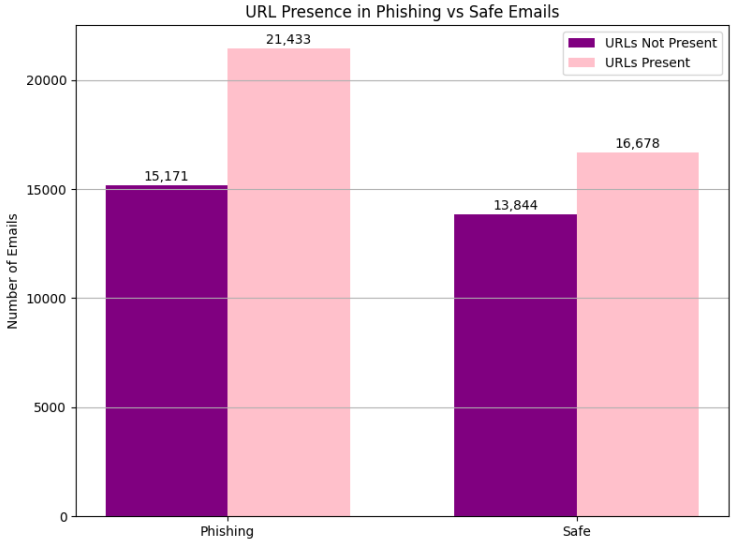

Figure 3: Division of URL and Non-URL emails between Safe and Phishing category.

The final set of cleaned data can be observed from the phishing data frame in Figure 4 below, which displays what the first five rows of our cleaned dataset look like.

body	urls	label
They look and feel exactly like the real thing. ...	1	1
CNN Alerts: My Custom Alert ...	0	1
Ranked #1 Men's Supplement by GQ in 2007, discover the secrets of thousands here http://www.risepat.com/	1	1
No girl will now resist a temptation to go with you! ...	1	1
Dear Sarah, I hope this email finds you well. My name is Jennifer Smith, and I'm thrilled to offer you an ...	1	0

Figure 4: Depiction of the samples in the dataframe head from our cleaned data

3.1.2 New Data Source and Plan

Only 45.5% of the cleaned data were safe emails, so new testing samples will target a 50/50 safe-to-phishing ratio to check for phishing bias in the model. New samples will be collected from University of Toronto Outlook accounts (manual collection) and Gmail's promotions/spam folders (using a Python script with the Gmail API). These will replace 10% and 50% of the original test set with Outlook and Gmail samples, respectively.

Error analysis will identify features linked to misclassifications (e.g., email length, sender domain, exclamation marks). These features will be stored in data frames, and a Seaborn heatmap will visualize correlations between them and prediction errors. New emails with features that the model most often misclassified can be extracted. These new emails can be used to replace approximately 5% of the training and validation datasets, allowing the model to better learn how to predict labels for new emails (Jain, Sejal, 2021).

3.1.3 Challenges Faced

Standardizing data posed challenges as "sender" fields were dropped due to inconsistent and missing data across datasets, which could introduce noise and reduce model reliability. In contrast, "url" field was retained and reconstructed a binary indicator (checking for "http" in email bodies) since the presence of URLs is a strong phishing indicator. This decision ensured consistency across datasets and preserved valuable predictive information.

3.2 Baseline Model

A baseline Binary Logistic Regression (BLR) model was implemented as it is effective and widely used for binary tasks. It predicts the probability that an email is phishing (class = 1) or safe (class = 0) by applying the sigmoid function to map outputs between 0 and 1, using a 0.5 threshold for final classification.

BLR was chosen as the baseline because it outputs normalized probabilities (0–1) that easily convert to labels, providing a clear, linearly separable decision boundary. Unlike linear regression, BLR is resilient to outliers since it does not fit a line through all data points, (Rishabh, 2023). The model is also computationally efficient because it only needs to store feature weights and uses fast dot product and sigmoid calculations for predictions, unlike other large algorithms such as KNN and decision trees that require more memory and slower computations. This makes it well-suited for large text datasets in phishing detection, (MathWorks, 2023).

3.2.1 LR Implementation and Results

Using this set-up, the logistic regression model achieved a test accuracy of 98.44%, indicating strong performance on this dataset. The high accuracy suggests that, with a straightforward BoW representation, the model effectively captures phishing patterns based on repeated word and phrase usage within the dataset. Figure 5 below, shows the baseline model pipeline.

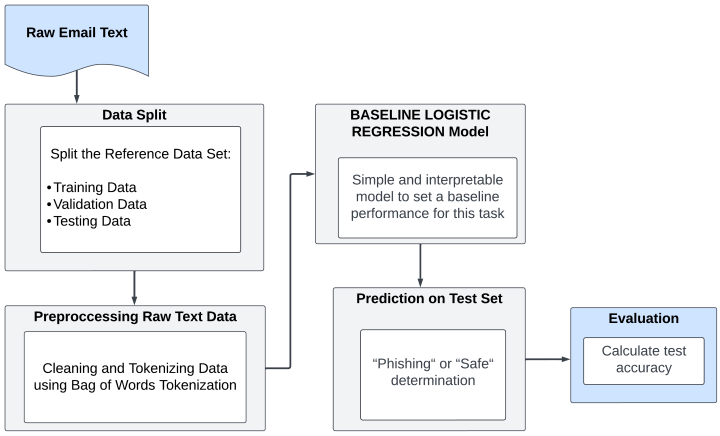

Figure 5: Depiction of the samples in the dataframe head from our cleaned data

Quantitative Results

- **Accuracy:** 98.44% in the validation set.
- From the confusion matrix generated in Figure 6, we have the following calculations:

Table 3: Precision, Recall, and F1 Score calculated from the confusion matrix.

Precision	Recall	F1 Score
0.9766	0.9922	0.9844

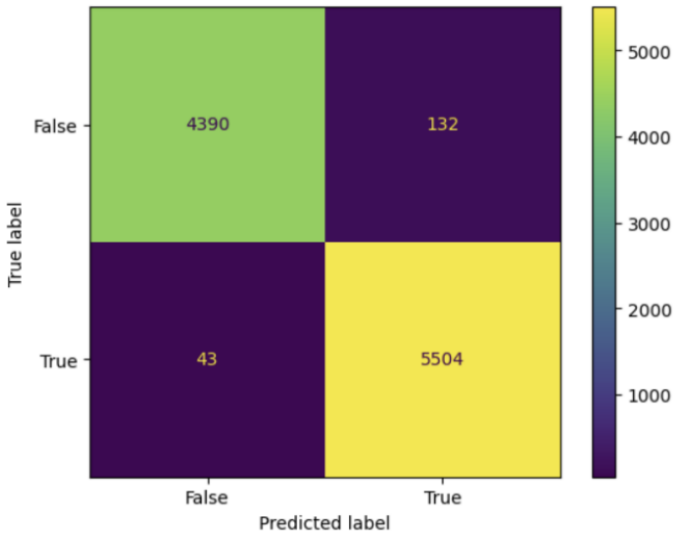

Figure 6: Depiction of the samples in the dataframe head from our cleaned data

- The high recall suggests that 99.2% of all phishing emails were identified, and the high F1 score implies an overall reliable model with strong precision and recall.

Qualitative Observations

- The model consistently identified phishing emails that contained repeated action phrases.
- Incorrect classifications generally involved email bodies with non-standard phrasing.

3.2.2 CHALLENGES FACED

1. **Tokenization Type:**
   - TF-IDF tokenization reduces the weight of frequently occurring words, which may suppress phishing indicators.
   - BoW better aligned with phishing detection objectives, emphasizing recurring phrase patterns.
2. **Using an imported binary logistic regression model**
   - Due to the model being imported, the architecture of the model could not be modified during fine tuning.
   - However, due to the computational cost of a custom-built model, as well as the strong performance of the imported model, the pre-initialized model was kept.

3.3 PRIMARY MODEL

The current model has the following inputs: an input_id associated with each email body, an attention_mask to separate padding from text, the binary value of email classifications, and the binary value of whether a url is present. Training is performed over 3 epochs with a batch size of 8. The following layers are defined in the list below.

1. **Forward Pass**
   - **BERT Encoder:** Custom neural network classifier that is pre-trained for encoding text input and uses a Hugging Face Transformer to handle case-sensitive text.
   - **Dropout Layer:** `dropout_layer = 0.3`.
   - **Linear Mapping:** CLS token (an embedded token from the input) mapped to logit score (a value between 0 to 1 where phishing = 1, not phishing = 0).
2. **Optimizer**
   - **AdamW Optimizer:** `learning_rate = 1e-5`, `weight_decay = 2e-5`.
   - Acts as regularization to prevent overfitting, which is common when using BERT.
3. **Scheduler Step:** Learning rate gradually increases by 10%.
4. **Track Training Loss:** Outputs error loss value per epoch.
5. **Early Stopping:** Based on the error loss value, either continue with the model or stop training if loss is increasing.

Figure 7, below shows the flow and functionality of the architecture in greater detail

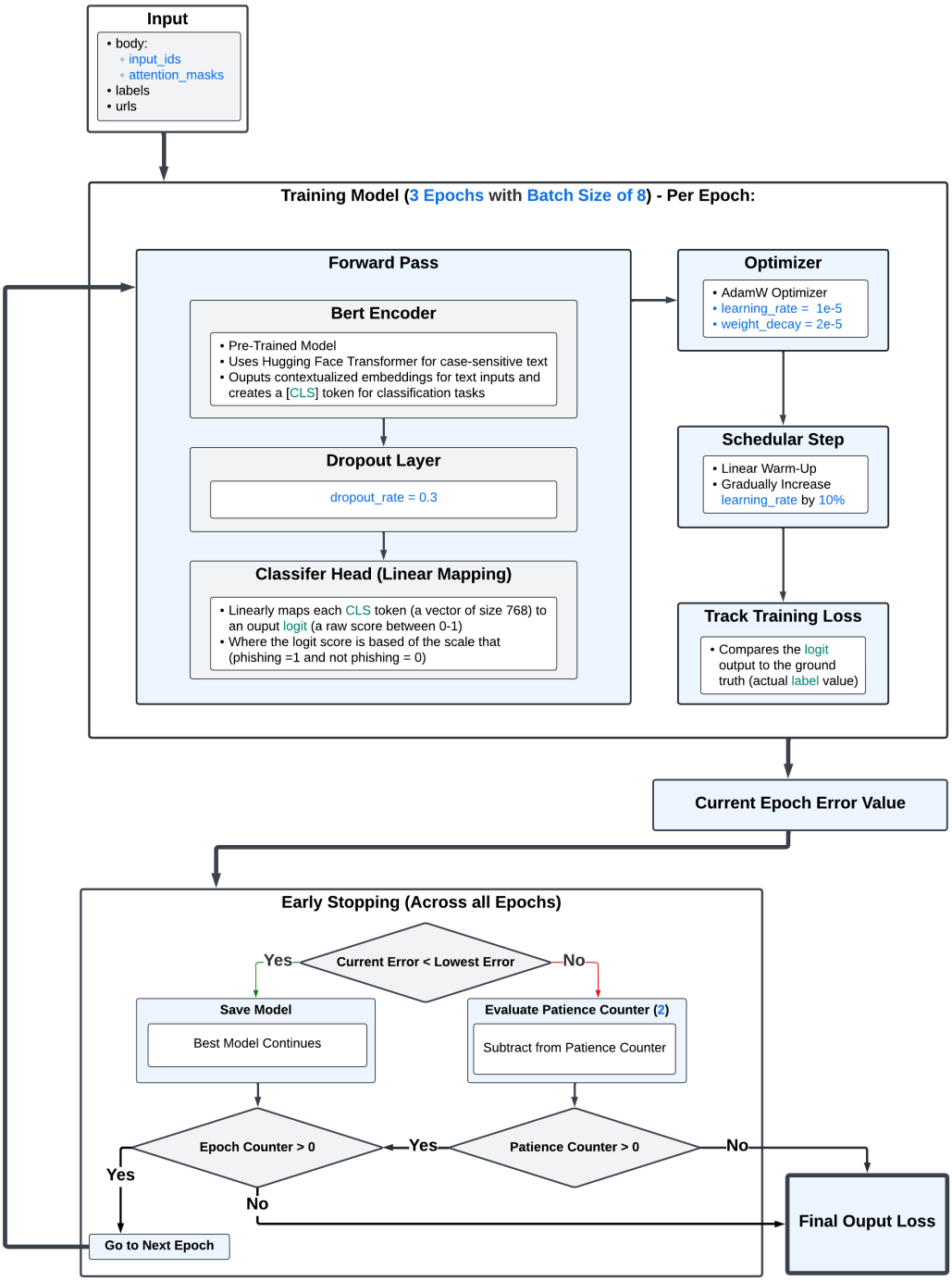

Figure 7: AI Model Architecture for BERT, (hyper-parameters are in blue text).

3.3.1 Results Across Models

Three models were tested before selecting the "best" and current model. The table below outlines the hyperparameters and results from each model.

Table 4: Validation Loss and Error Comparison Across BERT Classifier Variants

#	BERT type	Epoch, Batches	Optimizer	Layers	Val Loss	Val Error
1	Sequential Classifier (imported)	Epoch: 3 BS: 16	AdamW learning_rate = 2e-5 no weight_decay	1. Forward Pass	0.0394	0.0124
2	Sequential Classifier (imported)	Epoch: 3 BS: 8	AdamW learning_rate = 1e-5 weight_decay = 2e-5	1. Forward Pass 2. Early Stopping	0.0306	0.0092
3	Custom Classifier (self-built) Acts as NN	Epoch: 3 BS: 8	AdamW learning_rate = 1e-5 weight_decay = 2e-5	1. Forward Pass 2. Early Stopping 3. Dropout Layer 4. dropout_rate = 0.3 5. Scheduler Layer	0.0269	0.0084

More qualitatively, we can observe changes in the learning curves between iterations, as shown in Figure 8. You will see severe overfitting in the first iteration, which is slowly improving by our current iteration.

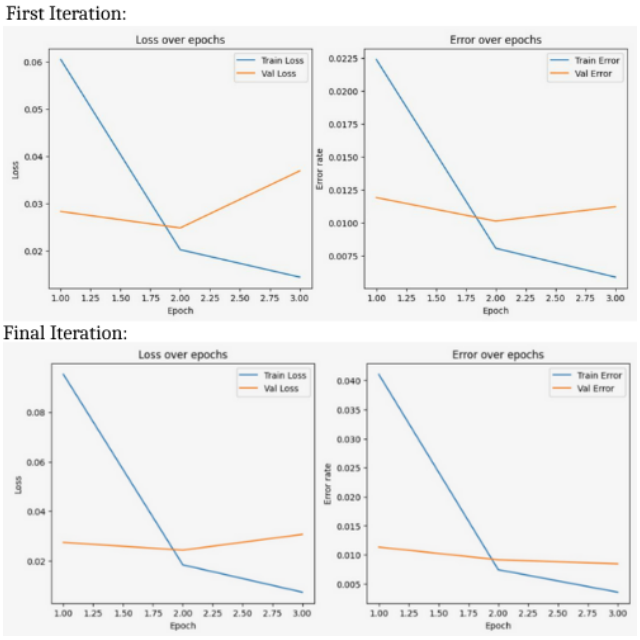

Figure 8: Learning Curve Improvement between iterations.

It is evident that our model needs to be further fine-tuned due to the prevalent overfitting. Our final model being changed to a custom neural network based BERT Classifier from the original Sequential Classifier allows us to implement more logistical changes such as adding more safety-net layers and changing more hyperparameters to manage overfitting.

3.3.2 Challenges Faced

The main challenge was the test time required for each model iteration. Notoriously known for demanding runtimes, testing the model required hours between iterations. Efforts were taken to reduce runtime cost, i.e. tokenizing was performed for each sample as it entered BERT, rather than tokenizing the whole dataset and backlogging it onto RAM. Efforts were also made to run the code on local GPU rather than colab to help mitigate run time. We are still working on ways to optimize our model's runtime moving forward and making our model more accurate.

COLAB LINK

The Google Colab notebook for this project can be accessed here: https://colab.research.google.com/drive/1m7JTrTuzCpVcSVN6c8YH6NbRZ2r5XRM4?usp=sharing

GITHUB LINK

The GitHub repository for this project can be accessed here: https://github.com/leahmdmartins10/Phishing-Email-Detection-System-BERT.git

REFERENCES

Naser Abdullah Alam and Amith Khandakar. Phishing no more dataset, 2024. URL https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset. Accessed: 2025-07-11.

Subhadeep Chakraborty. Cybercop phishing emails dataset, 2023. URL https://www.kaggle.com/datasets/subhajournal/phishingemails. Accessed: 2025-07-11.

Francesco Greco. Human-llm generated phishing and legitimate emails. https://www.kaggle.com/datasets/francescogreco97/human-llm-generated-phishing-legitimate-emails, 2024. Accessed: 2025-07-11.

Jain, Sejal. Error analysis for machine learning classification models. https://heartbeat.comet.ml/error-analysis-for-machine-learning-classification-models-8d35e240d9d3, 2021. Accessed: 2025-07-11.

MathWorks. Choosing the best machine learning classification model and avoiding overfitting, 2023. URL https://www.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html. Accessed: 2025-07-11.

Radoslav Miltchev, Dimitar Rangelov, and Genchev Evgeni. Phishing email validation dataset. https://research.utwente.nl/en/datasets/phishing-validation-emails-dataset, 2024. Accessed: 2025-07-11.

Robu Rishabh. Why logistic regression beats linear regression for classification, 2023. URL https://medium.com/@RobuRishabh/why-logistic-regression-beats-linear-regression-for-classification-9091a1cf877e. Accessed: 2025-07-11.