**1. What model did we choose? How many models did we choose?**

For this project, we selected three supervised machine learning models to classify emails as phishing or legitimate:

* Logistic Regression
* Multinomial Naive Bayes
* Random Forest

Each model was trained using textual features extracted through TF-IDF vectorization. This range of models allows us to compare linear, probabilistic, and ensemble-based approaches for spam detection.

**2. Why did we choose these models?**

We selected these three models because they represent commonly used, interpretable, and effective approaches for text classification tasks:

**Logistic Regression** is a simple yet powerful linear model, ideal for linearly separable data. It is widely used in spam detection due to its ability to assign clear decision boundaries and interpret feature weights.

**Multinomial Naive Bayes** is particularly suited for discrete features such as word counts or frequencies. It assumes independence between features and is computationally efficient, making it a strong baseline for NLP tasks.

**Random Forest** is a decision-tree ensemble model. It can capture complex feature interactions and is less sensitive to irrelevant features. It’s useful when the relationship between features and labels is nonlinear.

**3. What is the difference between these models?**

The core differences lie in how each model learns from data and makes predictions:

**Logistic Regression** computes a linear boundary using a weighted sum of features and applies a sigmoid function to output probabilities. It assumes a linear relationship between feature inputs and the log-odds of the class label.

**Naive Bayes** uses Bayes' theorem and assumes that all features are conditionally independent given the class. It calculates the probability of an email being phishing based on the likelihood of words appearing in that class, making it fast but sometimes limited by its strong independence assumption.

**Random Forest** builds multiple decision trees on random subsets of data and features, and aggregates their votes for final classification. Unlike the other two, it does not assume linearity or independence, making it robust to overfitting and more adaptable to irregular patterns in the data.

**4. Correlation Matrix with Features**

Since the input features are derived from TF-IDF vectorization of email text, traditional Pearson correlation is not ideal. However, a heatmap of the most frequent n-gram TF-IDF features was generated to show feature co-occurrence tendencies.

In traditional structured datasets, a correlation matrix helps identify multicollinearity. For text data, feature co-occurrence is more meaningfully explored via term similarity (e.g., cosine similarity) or via PCA/UMAP visualizations, which are not applied here but can be considered for future dimensionality analysis.

**5. Which model are we using?**

After evaluating all models on interpretability, computational efficiency, and generalization, we selected Logistic Regression with TF-IDF features as the primary model. It offers a good balance between accuracy and simplicity and performs robustly on phishing detection tasks, especially with word and character n-gram features.

6. Comparison of 2–3 Models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **How It Works** | **Strengths** | **Limitations** |
| Logistic Regression | Finds a linear decision boundary in feature space using weighted summation. | Interpretable, fast, performs well with high-dimensional sparse data like TF-IDF. | Assumes linear separability; limited in capturing complex relationships. |
| Naive Bayes | Uses probability theory assuming feature independence for each class. | Very fast, works well with small datasets and high-dimensional text data. | Independence assumption may not hold in real-world email text. |
| Random Forest | Builds many decision trees and averages their outputs for classification. | Captures complex feature interactions, robust to overfitting, good with imbalanced data. | Slower, less interpretable, and may require more memory for training. |

7 Dataset Description

We used the Phishing Email Detection dataset available on Kaggle:

<https://www.kaggle.com/datasets/subhajournal/phishingemails>

This dataset contains a large collection of email texts labeled as phishing (1) or legitimate (0). The emails include subject lines and full bodies of messages. The data is text-heavy and suitable for natural language processing techniques like TF-IDF. It is balanced well enough to train binary classification models, with necessary preprocessing applied.

**8. Feature Selection**

Feature selection was handled by using TF-IDF vectorization, which transforms text into numeric features based on term frequency and inverse document frequency. Both word-level and character-level n-grams were used:

* Word n-grams (1,2) capture simple token patterns.
* Character n-grams (3,4,5) help detect obfuscated or intentionally misspelled words used in phishing (e.g., "baank", "paypa1").

We also limited the feature size (e.g., max\_features=5000) to avoid overfitting and reduce computational load.

**9. Suggestions for Improvement**

To improve the model or extend this project, the following suggestions are proposed:

Use Deep Learning Models: Implement LSTM, GRU, or transformer-based models like BERT for contextual understanding of email language.

Add Metadata: Include email headers, sender domain, or time-based metadata for additional classification power.

Real-Time System: Deploy the trained model in a real-time email server for spam filtering.

Model Explainability: Use tools like SHAP or LIME to visualize and explain predictions for end-users and system transparency.