# ml_proposal

# Phishing URL Detection

Final Project Report for CS 7641

## Introduction/Background

Our project focuses on detecting phishing URLs using machine learning. Based on this research paper, most approaches leverage models like SVM, Neural Networks, Random Forest, Decision Tree, and XGBoost, achieving accuracy rates between 80% and 98%. For our project, we will be working with the PhiUSIIL Phishing URL dataset, which includes over 230,000 URLs. Key features of this dataset include URL structure, uses HTTPS, number ratio, special char ratio, and the domain length.

## Problem

Phishing is a common and dangerous scamming technique which can result in the loss of valuable personal and sensitive information. What makes phishing so dangerous and difficult to safeguard against is the necessity for end users to differentiate between genuine correspondence and phishing. According to one study, research showed that “from a group of 179 participants and a set of 20 messages (11 phishing examples and nine legitimate mails), the overall level of successful classification was only 42%, with misclassification in 32% of cases and participants simply unsure in the other instances.” (Source) This study showcases the difficulty for end users to differentiate between real requests and phishing scams. The goal of our project is to utilize a dataset of common phishing URLs and data regarding its legitimacy and apply machine learning models to predict which URLs are phishing URLs and which URLs are legitimate.

## Methods

To detect phishing URLs in our project, we will apply the following combination of supervised and unsupervised models alongside effective data preprocessing techniques. We will leverage established libraries like scikit-learn and XGBoost for efficient implementation.

### Data Preprocessing Methods:

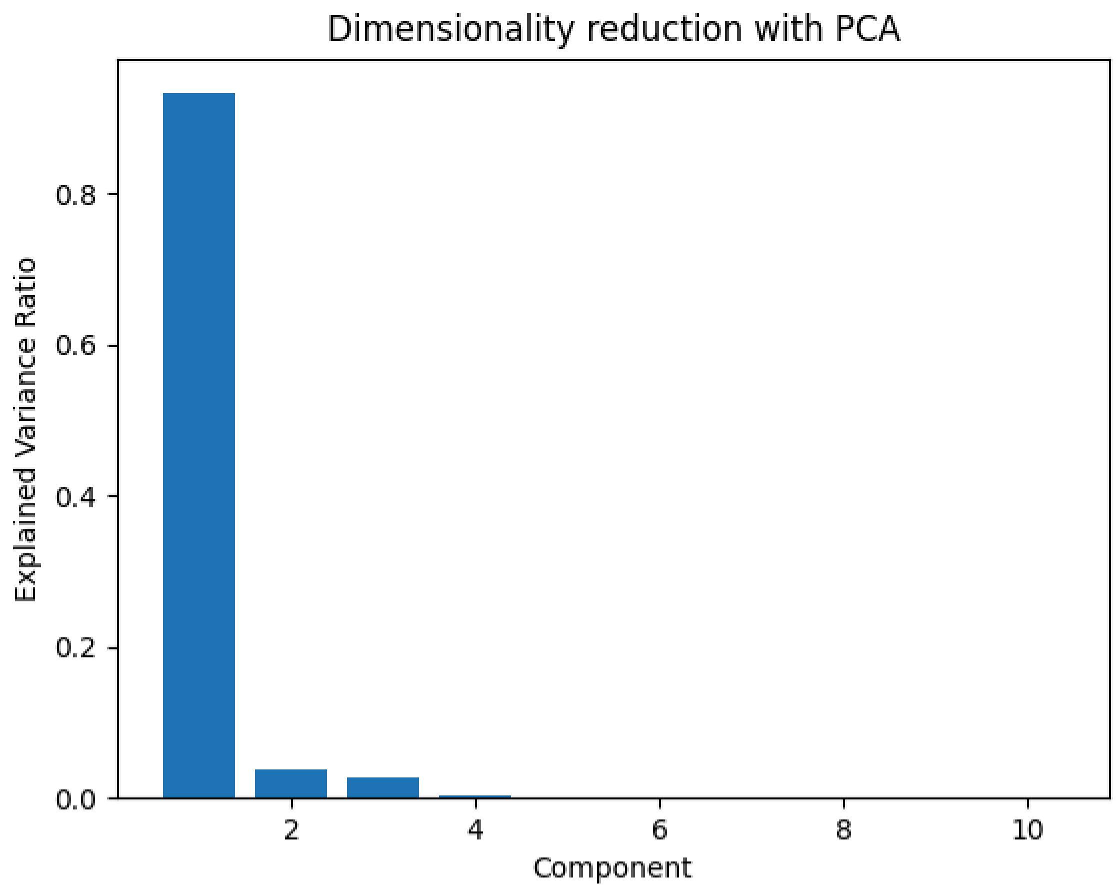
- **Feature Engineering:** We will derive new features from URLs, such as subdomain count and URL length, using pandas for extraction (e.g., detecting suspicious keywords with str.contains()) [1].

FeatureUnion in scikit-learn will assist in combining multiple feature extraction pipelines.

- **Normalization:** Continuous features like domain age will be normalized using MinMaxScaler or StandardScaler from sklearn.preprocessing to ensure consistency across the dataset [2].
- **Performing PCA:** Since we have a lot of features, we will perform PCA on our dataset to keep only the strongest components. With this, our models will have a easier and faster time training and produce better quality results.

## Preprocessing with PCA:

The dataset we used had many extracted features precalculated from the URL string such as *Length* or *Number of Question Marks*. However, it also had features regarding the website itself, such as number of lines of JavaScript code, which we removed to keep features related to the URL string itself. After narrowing it down to around 15 features, we performed **Principal Component Analysis** to see how many of the features could be condensed. Below are the explained variance ratios calculated for principal components:



As shown, this dimensionality reduction process allowed us to reduce the dataset to 10 principal components, capturing most of the essential information while minimizing redundancy. Notably, the first three components account for the majority of the variance within the data, suggesting that these components are the most influential in describing the underlying structure of our dataset. This finding allowed us to focus our analysis primarily on these three components for further interpretation and modeling.

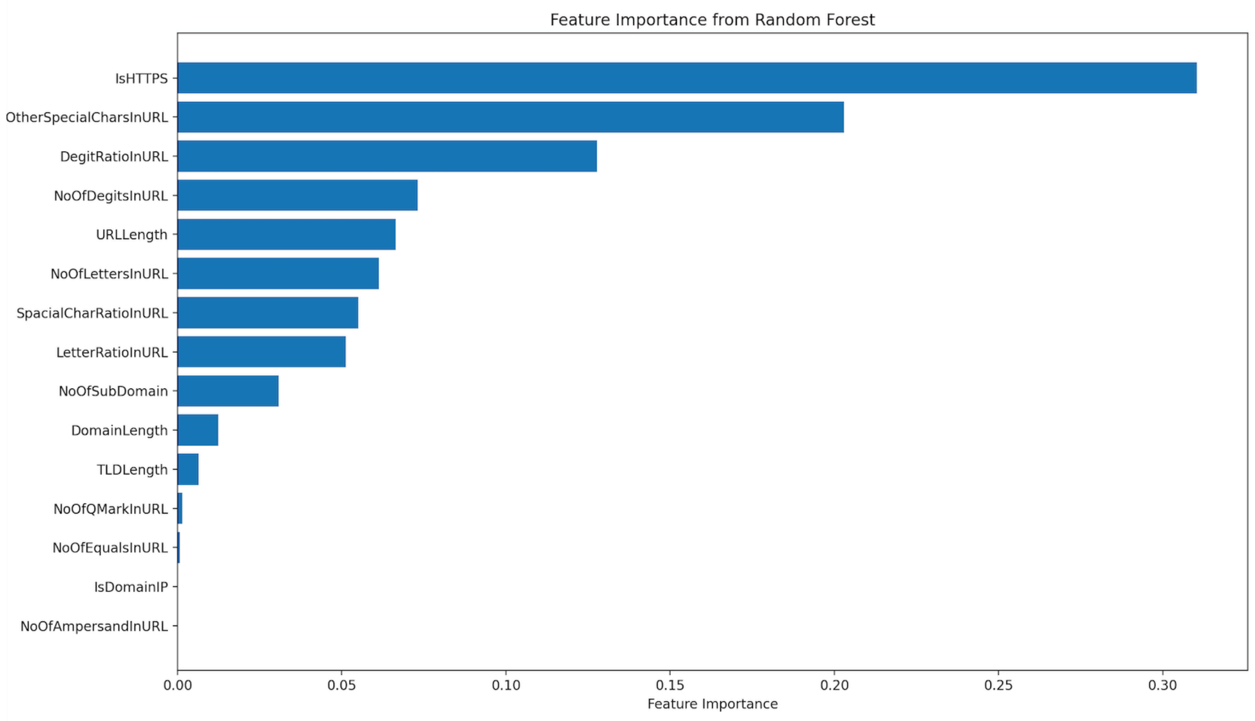
## Machine Learning Algorithms/Models:

- **Random Forest (Supervised):** RandomForestClassifier from sklearn.ensemble will help manage high-dimensional datasets and provide insights into feature importance [3].
- **Logistic Regression (Supervised):** LogisticRegression from sklearn.linear_model will provide a fast model for binary classification, making it ideal for large datasets [4]. It will allow us to easily understand the relationship between individual features, which is useful for feature importance analysis and model transparency.
- **Isolation Forest (Unsupervised):** IsolationForest from sklearn.ensemble will aid in identifying anomalous URLs that deviate from normal patterns [5].

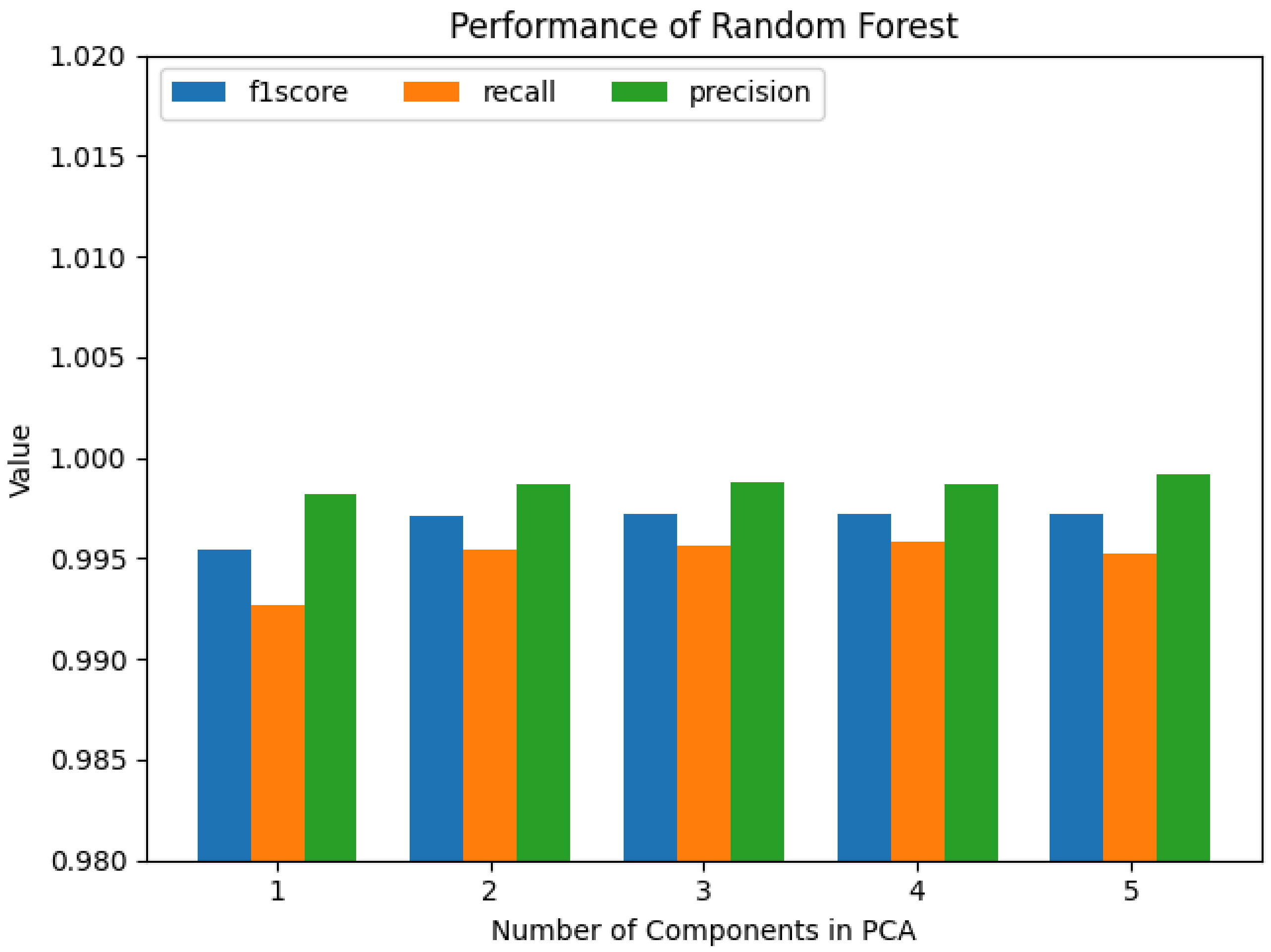
## Random Forest Model:

For our first model, we decided to implement Random Forests since they deal well with classification problems and the possibility of large amount of features.

- **Feature Importance:** After preprocessing, we used a Random Forest model to assess the importance of the remaining features. The bar chart below shows the feature importance scores, with each bar representing a feature's contribution to the model's predictions. Longer bars indicate more important features. This visualization helped us identify which URL-related features are most influential for the model.



- **Results:** We show our results on differing number of features from PCA analysis.



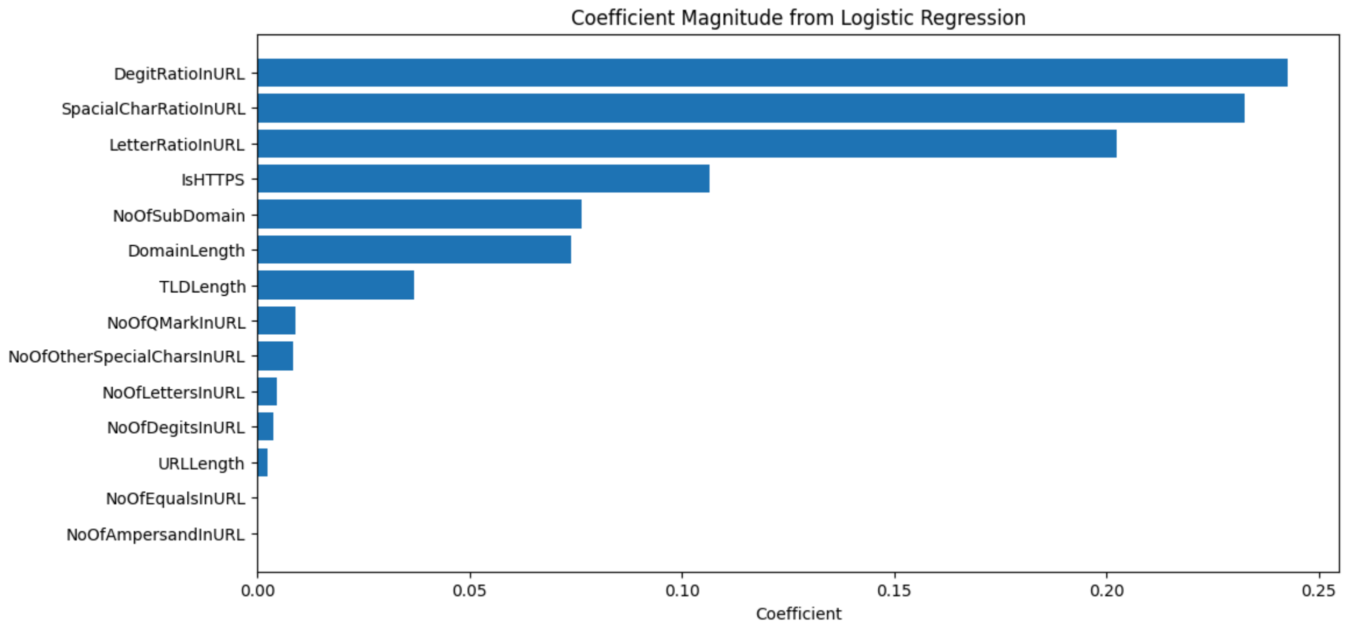
The Random Forest model performs extremely well even using just one component achieving an F1 Score of 0.995 and there is a small, noticeable increase to two components with an F1 score of 0.997. We found it very interesting that our model can predict phishing URLs with such high accuracy. This performance likely reflects the ability of the model to capture distinctive patterns in the dataset, especially given that the majority of variance is explained by the three strongest components of PCA. We think that these components reflects the key features like HTTPS usage and the presence of special characters and serve as strong indicators, enhancing the model's ability to distinguish phishing URLs from legitimate ones

## Logistic Regression

The second model we implemented was Logistic Regression which classifies using a linear regression as input into a sigmoid function to map it to a (0,1) space. Many of our features are binary inputs which do not work as well in a regression scenario, but we decided to try this method for another type of model.

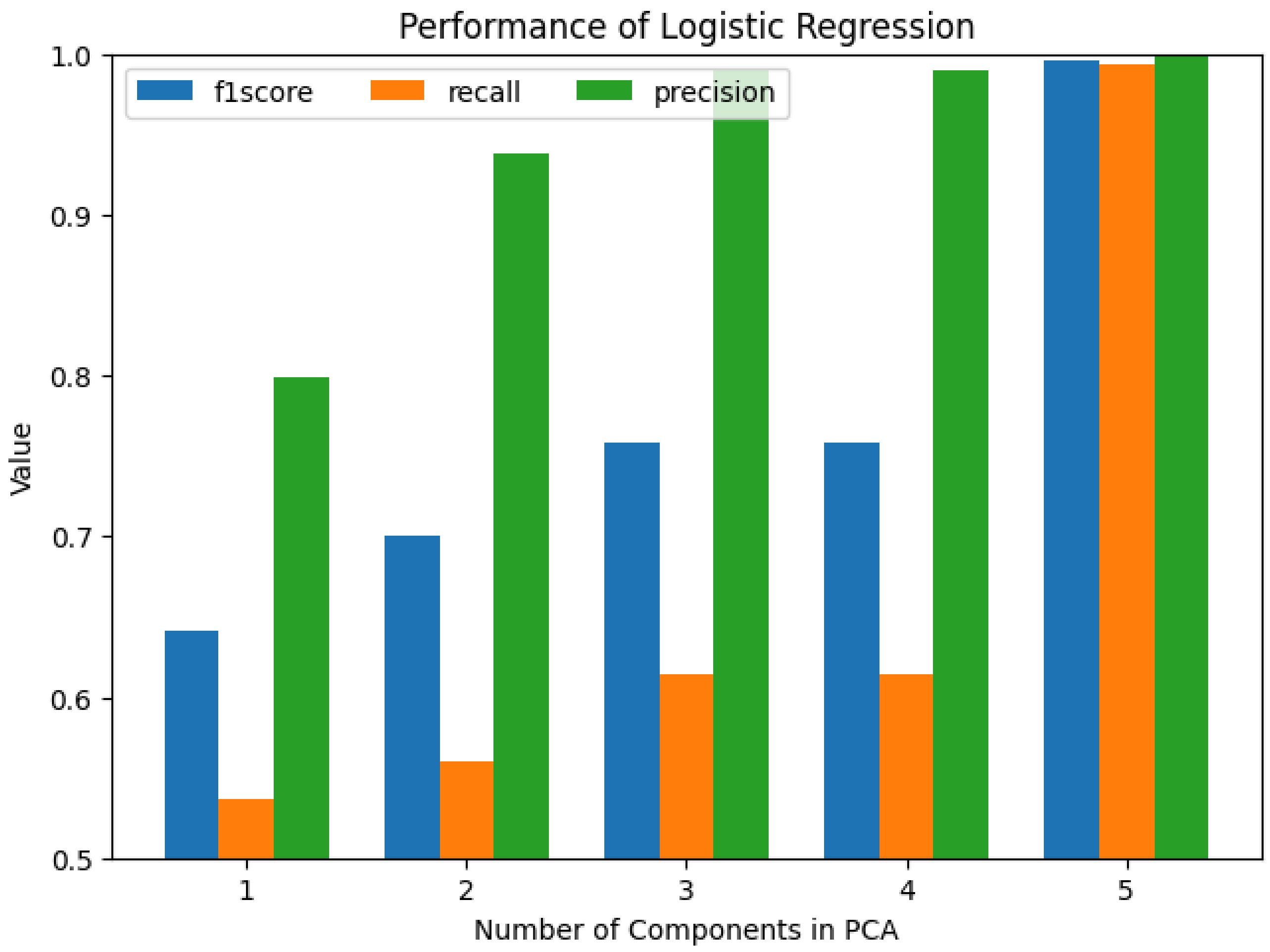
- **Coefficient Magnitude:** Logistic Regression does not have a set way of implementing feature importance like the Random Forest model. However, after normalizing our features, we can get a kind of sense of feature importance using the magnitude of coefficients in our regression.

Here we present a horizontal bar graph representing the magnitude of the coefficient used in the model:



One interesting thing to note is the magnitude of the ratio features versus the count features, showing how ratios provide more info. Conceptually, this makes sense since the counts are dependent on the length of the url which does not correlate directly with whether or not it is a phishing website, and the ratios provide more info on the kind of string itself.

- **Results:** Using our PCA preprocessing step, as we did with Random Forest, we calculate recall, precision, and accuracy:



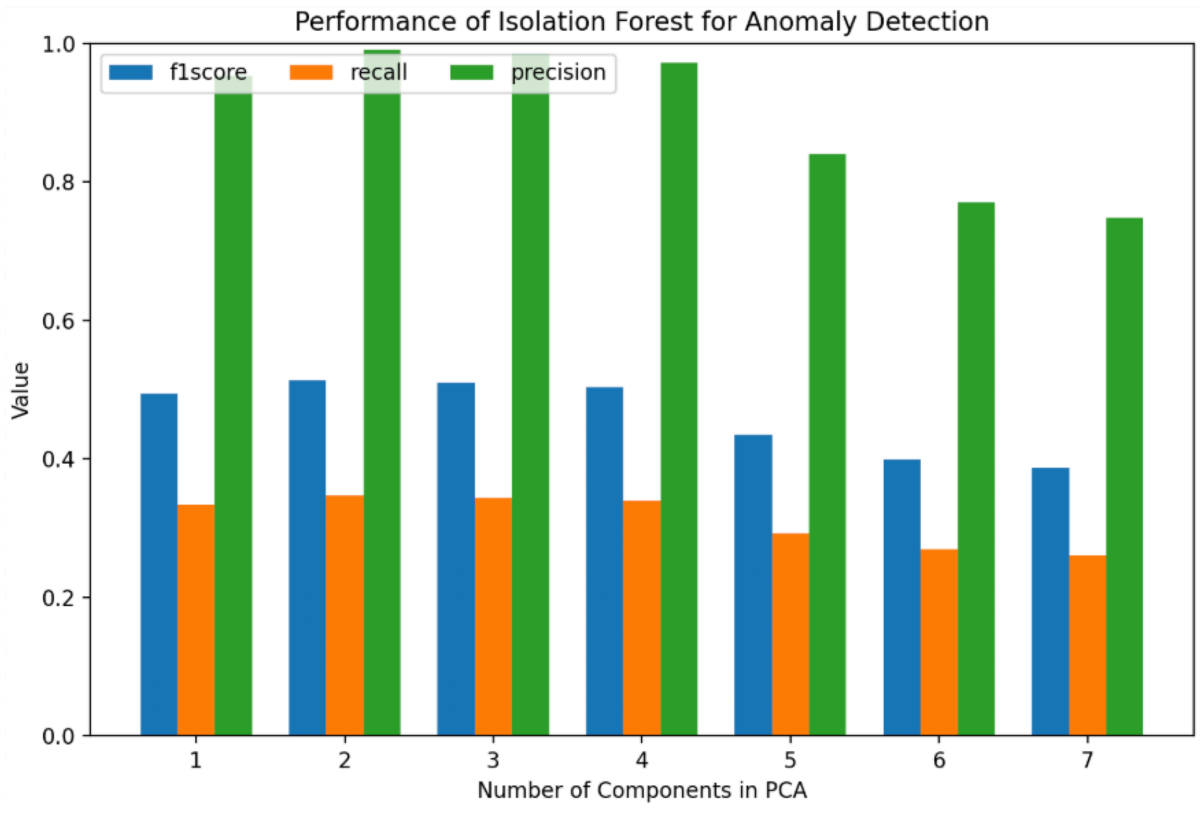
Given 5 components from PCA, the algorithm was able to achieve optimal performance. Interestingly, in the other cases, it generally has high precision meaning it rarely misclassifies as spam. However, this is the opposite of what we want since we want less false negatives than false positives, a higher recall.

## Isolation Forest Model:

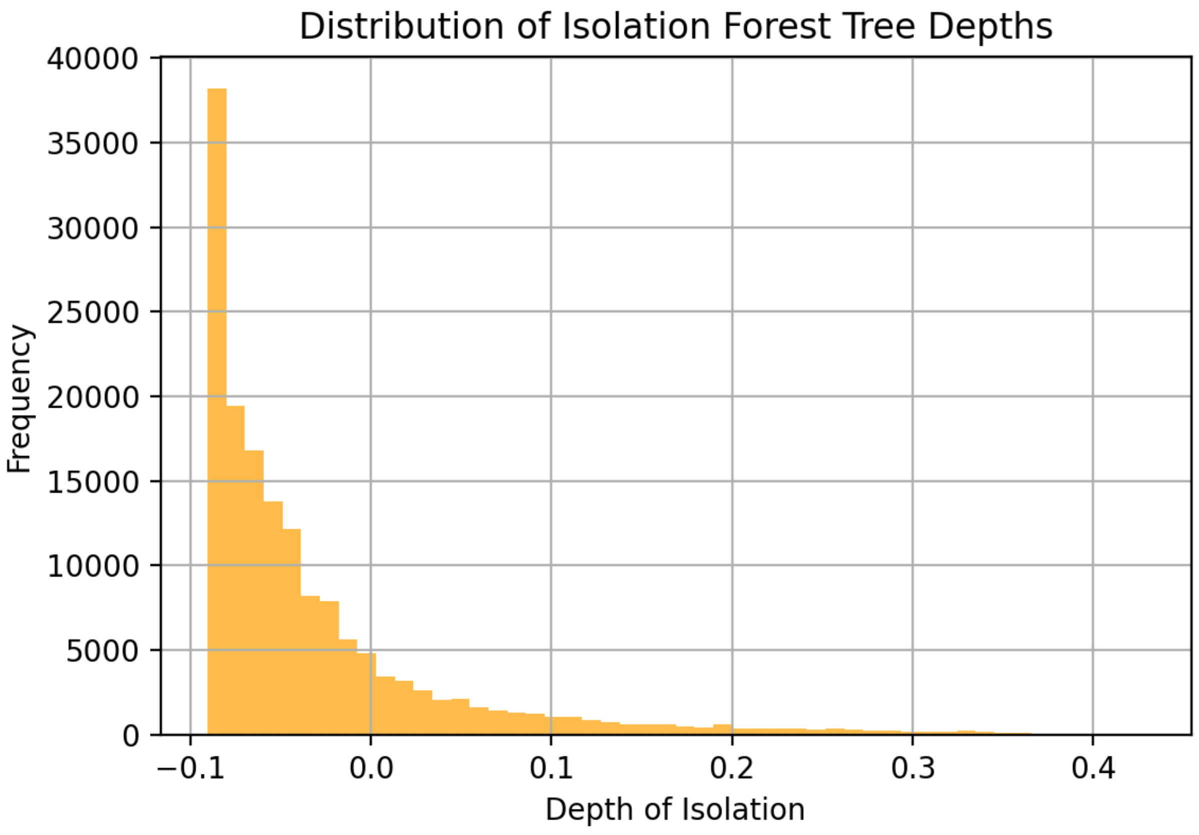
For our third model, we implemented Isolation Forest, an unsupervised anomaly detection technique to identify outliers in our dataset. Isolation Forest works by isolating observations instead of profiling normal data, which makes it effective in detecting events such as phishing URLs in highly imbalanced datasets. This method is particularly useful when we don’t have access to explicit labels for training, which is often the case in anomaly detection tasks.

- **Feature Transformation:** Similar to the Random Forest approach, we first performed feature preprocessing and used Principal Component Analysis (PCA) to reduce the dimensionality of our dataset. The goal was to retain the most important features while reducing noise. We evaluated the model’s performance with different numbers of principal components, starting with 1 and gradually increasing up to 5 components, to see how the model’s accuracy changes with more compressed representations of the data.

- **Results:** The Isolation Forest model performed well for detecting phishing URLs, using unsupervised anomaly detection. We tested the model with 2 PCA components, which yielded the best performance. Precision: 0.986. The model achieved the best balance between precision (~98%) and recall (~34%) with 2 PCA components. Although recall was lower, the high precision indicates reliable detection of phishing sites.



- **Isolation Forest Depth:** We visualized the decision function depths to show how easily observations are isolated. Anomalies, with lower depths, are easier to separate from normal data. The distribution of tree depths helps explain how the model identifies phishing URLs as anomalies.



## Results and Discussion

The quantitative metrics we will use for this project are:

- **Precision** - How well we identify phishing urls
- **Recall** - How well we don’t misidentify phishing urls
- **F1-Score** - General performance of our algorithm

The goal of this project is have high recall (catching phishing urls, having low false negatives) while keeping precision as high as possible. Due to the success of past research, we believe this is something we can achieve. From these models, we see that we are generally able to classify phishing urls very easily, especially if we are using the random forest or logistic regression.

Our strongest model is the random forest:

- Strength: The model can reach 0.995 F1 score even with just one PCA component. It stands out because it offers the most balanced precision-recall tradeoff, this is especially important because we wand our model to have high recall while maintaining precision as high as possible.
- Limitation: The model can be overfitted to the dataset and it takes more training time than models like logistic regression.

- Tradeoff: Random Forest has excellent predictive performance, and the risk of overfitting and slower training times could be mitigated by refining preprocessing steps and feature selection. This model is the best out of the three we experimented with.

Logistic Regression is also a strong model:

- Strength: It is easy and computationally efficient to set up and train.
- Limitation: Some of the binary features like whether uses https might not be as useful to the model. This model does have a lower recall than random forest so it won't be as good in catching phishing URLs.
- Tradeoff: The model does require 5 PCA components to reach its full potential but it is easier to set up and slightly computationally efficient than random forest so it might be a better choice if we want a lightweight model.

Isolation forest has a slightly worse performance than the other models:

- Strength: It is a unsupervised model so if we have datasets that is missing labels, it would excel in phishing detection.
- Limitation: The model has a significantly lower recall (~34%). This is a major limitation so if the dataset have labels like the one we used, other models are preferred.
- Tradeoff: This model excels if we need to have unsupervised learning or if want to detect anomalies in urls. However, with its low recall, it might be risky to use this model to classify URLs.

## Next Steps:

We are looking to fine-tune our pre-processing by identifying and removing potential noisy features to reduce overfitting and ensure faster training times and produce quality results. When it comes to improving accuracy, we are looking to prioritize reducing false negatives (phishing identified as non-phishing) over false positives (non-phishing identified as phishing) since clicking on a phishing URL that is deemed safe is more dangerous than avoiding a safe URL labeled as phishing. Some approaches could be:

- **Data balancing** - Providing more phishing URLs than non-phishing URLs to increase recall
- **Feature Engineering** - Create new features that may capture unique patterns, such as embedding-based representations of URLs or domain-specific features like WHOIS lookup results.
- **Hyperparameter Searching** - Finding optimal hyperparameters such as tree depths, number of trees, number of estimators, etc for our models.
- **Attack Testing** - Test models with adversarial examples (e.g., obfuscated phishing URLs) to evaluate robustness and improve defenses against evasion.

We could also explore more models:

- **Voting** - We could use a voting system that combines outputs of many models together.
- **Deep Learning Models** - We could explore using deep neural networks to do this classification task to see if there is a better result.

## References

[1] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011. [Online]. Available: https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[2] Scikit-learn Documentation, “Preprocessing Data,” 2024. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html.

[3] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5-32, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324.

[4] J. MacQueen, “Some Methods for Classification and Analysis of Multivariate Observations,” in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 1967, vol. 1, pp. 281-297. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200512992.

[5] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation Forest,” in 2012 IEEE International Conference on Data Mining, pp. 413-422, 2012. [Online]. Available: https://ieeexplore.ieee.org/document/4781136.

[6] S. Furnell, “Still on the hook: the persistent problem of phishing,” Computer Fraud & Security, vol. 2013, no. 10, pp. 7–12, Oct. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361372313700927.

[7] A. Prasad and S. Chandra. “PhiUSIIL Phishing URL (Website),” UCI Machine Learning Repository, 2024. [Online]. Available: https://doi.org/10.1016/j.cose.2023.103545.

## Contribution Table

| Name | Proposal Contributions |
|---|---|
| Eric | Problem/Motivation, Next Steps, Random Forest, Isolation Forest |
| Jacob | Results and Discussion, GitHub Pages, PCA Preprocessing, Random Forest |
| Jordan | Introduction, Github Pages, Testing various Data Preprocessing, Result Analysis |

| Name | Proposal Contributions |
|---|---|
| Rohan | Model Visualization, Evaluation Metrics, Data Visualization |

## GANTT Chart Link

GANTT Chart