Fishing for Phishing Scams

Dil Dhaliwal, Evan Imtiaz

April 6th, 2023

CP322 – Machine Learning

Dr. Sukhjit Singh Sehra

Appendix

Abstract

With an increase in phishing attacks across not only Canada but worldwide it is important for tools to exist to help differentiate between malicious phishing attempts and legitimate webpages. Machine learning offers a way to classify information in order to distinguish between phishing and real webpages. With the availability of datasets containing webpage information, we decided it would be a worthwhile endeavour to create models to try and address the problem if even in a limited capacity. A dataset of 10,000 webpages both phishing and legitimate was cleaned of empty values and looked over to ensure quality. The dataset was then used for the training and testing of our models. Two models were created with the dataset split 70% for training data and 30% for testing data. The first model was a logistic regression model that was able to yield useful results with an accuracy around 85%. The second model ran was a decision tree which had an accuracy above 90% making it unusable due to how overfit it was. In the evaluation it was discovered that very few features held high significance due to the close resemblance of phishing and legitimate webpages. Despite that a handful of features yielded high significance with those being 'SelfRedirectHyperlinksRT' and 'PctExtHyperlinks'. Going forward to build on the model we plan to add new models such as random forests as well as incorporate new datasets to try and further refine our model.

Introduction

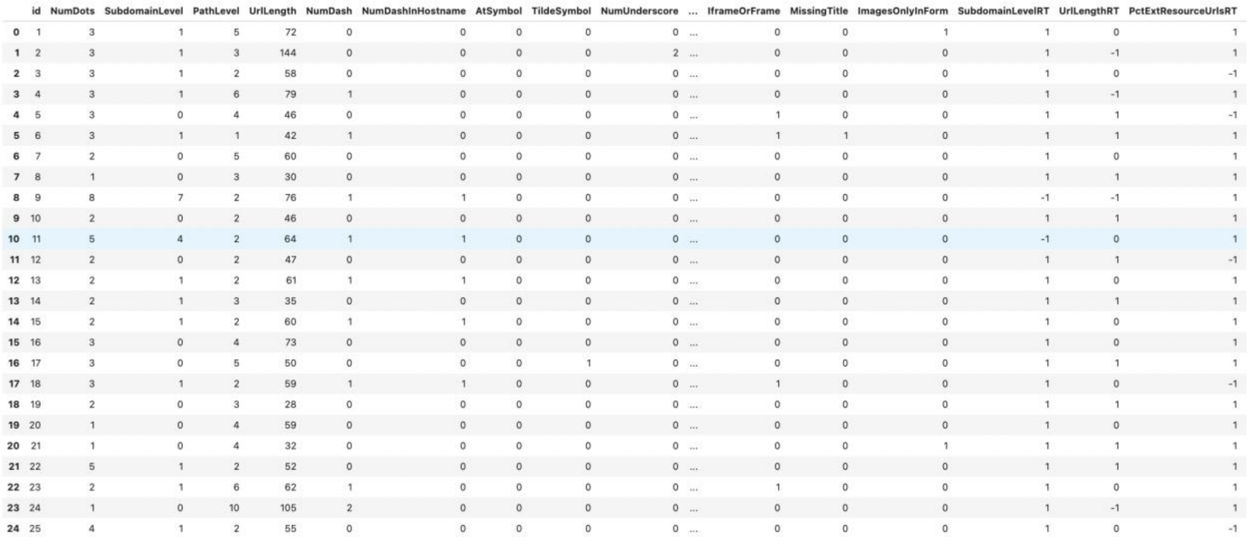
In the age of the internet that we live in today, phishing scams have become increasingly common and problematic. Thousands of Canadians are victim to phishing attacks yearly, resulting in millions of dollars in losses. The problem will only continue to worsen unless better measures are put in place to detect and prevent these attacks.

Project Description

Machine learning offers efficient and accurate ways to classify data that can be used to help find phishing webpages. The goal of our project was to create a model capable of taking numerous parameters from a dataset and be able to learn which parameters are the best indicators of phishing webpages and use the information to better classify webpages. Multiple models are used to see which model best suits the dataset and utilizes the features most effectively. The resulting model prioritizes general accuracy over multiple datasets rather than perfect accuracy on the training dataset. The reasoning being that perfect accuracy on the training datasets implies the model is overfitted and will suffer performing on new datasets.

Data Collection

For this project a well-structured dataset consisting of 10,000 webpages both phishing and legitimate were used to train and test our models. Each webpage in the dataset has 50 features attached that are used as indicators to better predict phishing webpages. The list of features includes notable information about the webpage such as if there is http in hostname or if an IP address is provided. Other information about the webpage link includes things like number of dashes or dots and path length. The dataset can be found in the following link: https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning

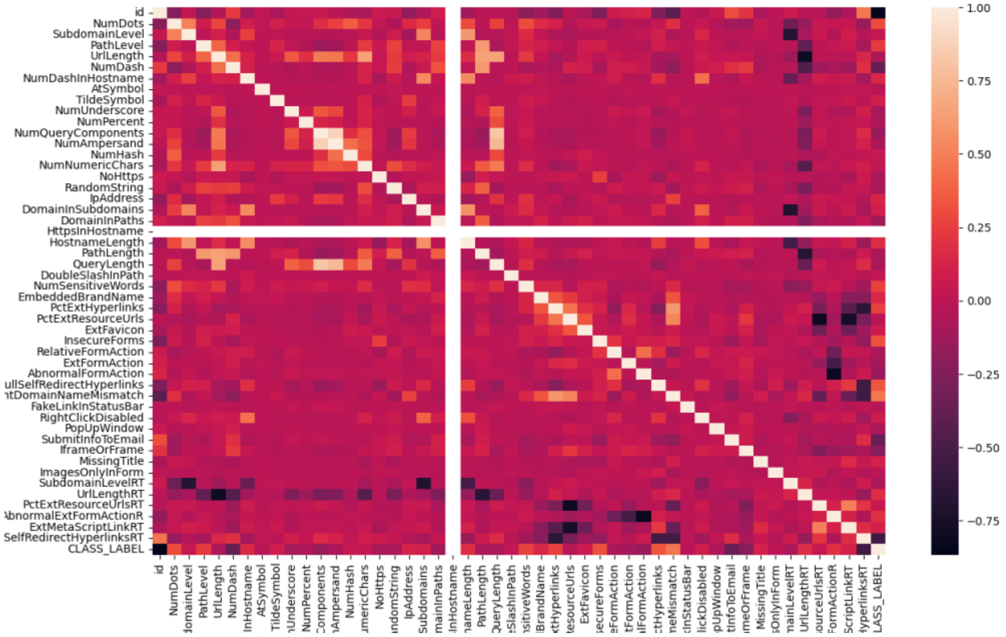


	id	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostname	AtSymbol	TildeSymbol	NumUnderscore	...	IframeOrFrame	MissingTitle	ImagesOnlyInForm	SubdomainLevelRT	UrlLengthRT	PctExtResourceUrlsRT
0	1	3	1	5	72	0	0	0	0	0	...	0	0	1	1	0	1
1	2	3	1	3	144	0	0	0	0	2	...	0	0	0	1	-1	1
2	3	3	1	2	58	0	0	0	0	0	...	0	0	0	1	0	-1
3	4	3	1	6	79	1	0	0	0	0	...	0	0	0	1	-1	1
4	5	3	0	4	46	0	0	0	0	0	...	1	0	0	1	1	-1
5	6	3	1	1	42	1	0	0	0	0	...	1	1	0	1	1	1
6	7	2	0	5	60	0	0	0	0	0	...	0	0	0	1	0	1
7	8	1	0	3	30	0	0	0	0	0	...	0	0	0	1	1	1
8	9	8	7	2	76	1	1	0	0	0	...	0	0	0	-1	-1	1
9	10	2	0	2	46	0	0	0	0	0	...	0	0	0	1	1	1
10	11	5	4	2	64	1	1	0	0	0	...	0	0	0	-1	0	1
11	12	2	0	2	47	0	0	0	0	0	...	0	0	0	1	1	-1
12	13	2	1	2	61	1	1	0	0	0	...	0	0	0	1	0	1
13	14	2	1	3	35	0	0	0	0	0	...	0	0	0	1	1	1
14	15	2	1	2	60	1	1	0	0	0	...	0	0	0	1	0	1
15	16	3	0	4	73	0	0	0	0	0	...	0	0	0	1	0	1
16	17	3	0	5	50	0	0	0	1	0	...	0	0	0	1	1	1
17	18	3	1	2	59	1	1	0	0	0	...	1	0	0	1	0	-1
18	19	2	0	3	28	0	0	0	0	0	...	0	0	0	1	1	1
19	20	1	0	4	59	0	0	0	0	0	...	0	0	0	1	0	1
20	21	1	0	4	32	0	0	0	0	0	...	0	0	1	1	1	1
21	22	5	1	2	52	0	0	0	0	0	...	0	0	0	1	1	1
22	23	2	1	6	62	1	0	0	0	0	...	1	0	0	1	0	1
23	24	1	0	10	105	2	0	0	0	0	...	0	0	0	1	-1	1
24	25	4	1	2	55	0	0	0	0	0	...	0	0	0	1	0	-1

(Data frame resulting from dataset)

Exploratory Data Analysis

After collecting the data and before utilizing it, we wanted to lightly explore and analyze it first. It is important to note that because of the integrity of the data source and its description we did not dive too deep into the analysis. Our first step of exploratory data analysis was recognizing the target variable. In our case, the target variable was obvious. It was called 'CLASS_LABEL' with each instance either being 1 to indicate it is a phishing webpage or a 0 to indicate legitimacy. We then visualized a correlation matrix to view the correlation between all variables in the dataset. We looked more closely at the correlation between the target variable and our predictors. Many predictors were lightly correlated while a few were strong.



(Correlation matrix of the dataset)

Pre-Processing

With the data collected, explored, and analyzed it was time to take the last steps to prepare the raw data and turn it into processed usable data. Prior to editing the data at all, we checked for null values. Null values are problematic, and it is vital in nearly all machine learning projects to search the dataset. Thankfully, our data contained no null values, and we were good to proceed with feature selection and scaling.

Feature Selection

We decided to leave all variables in the dataset aside from 'id'. We removed 'id' because it is a redundant variable that is essentially the index. The reason we left all variables in the dataset is because we wanted to see how our models would perform on the complete set. We plan to further build our project and pursue methods. Therefore, feature selection is a next step for us in the coming versions.

Feature Scaling

While our dataset consists of all variables in numeric form int64 and float64, an underlying problem was that they were held on different scales. The solution to this problem was normalization and the method used was feature scaling. This allowed all features to be held on the same scale. This was the last step for processing our data for use.

Splitting the Data

We split the data into 70/30 split meaning 70% of the data or 7000 instances were to be used for training and 30% or 3000 instances of the data were to be used for testing. There are two reasons we decided on this specific split. One, since we were using all features, we felt that if we provided too much training data it would overfit. Two, since the dataset is rather large, a split with reduced balance would be intensive.

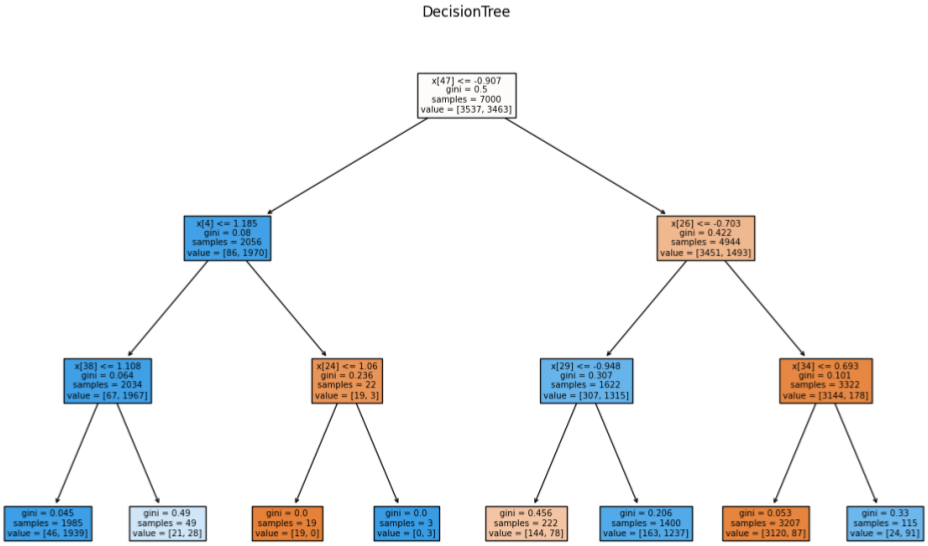
Models

Logistic Regression

We created a class for logistic regression with the following methods: initialize, fit, predict, and metrics. After creating the logistic regression model, we applied the fit and predict methods to fit the model and retrieve its predictions. While fitting the model we decided to apply L2 regularization to avoid overfitting the model. For evaluation, we produced a confusion matrix. We also determined the accuracy, precision, recall, and f1 score. To further evaluate the model, we used K-Fold evaluation with 8 splits and 4 repeats.

Decision Tree

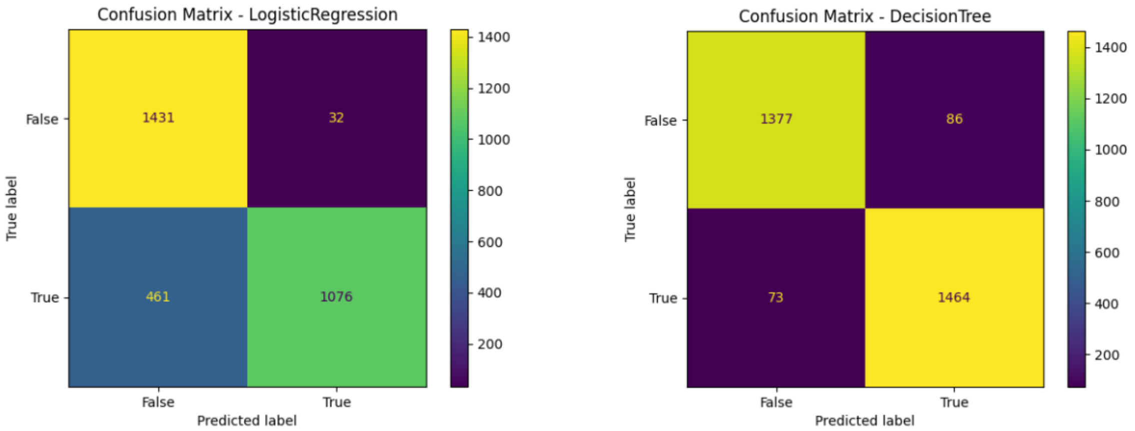
For our decision tree we began by creating a class and methods. The methods created were initialize, fit, predict and metrics. The fit and predict models were used to fit the model and receive the predictions. We decided to apply pre-pruning when fitting our model to ensure the model was not overfitted. To evaluate the model, not only were the accuracy, precision, recall and f1 score determined, a confusion matrix was also created and visualized to display the information. Furthermore, K-Fold evaluation was utilized with 8 splits and 4 repeats.


DecisionTree
x[47] <= -0.907 gini = 0.5 samples = 7000 value = [3537, 3463]
x[4] <= 1.185 gini = 0.08 samples = 2056 value = [86, 1970]
x[26] <= -0.703 gini = 0.422 samples = 4944 value = [3451, 1493]
x[38] <= 1.108 gini = 0.064 samples = 2034 value = [67, 1967]
x[24] <= 1.06 gini = 0.236 samples = 22 value = [19, 3]
x[29] <= -0.948 gini = 0.307 samples = 1622 value = [307, 1315]
x[34] <= 0.693 gini = 0.101 samples = 3322 value = [3144, 178]
gini = 0.045 samples = 1985 value = [46, 1939]
gini = 0.49 samples = 49 value = [21, 28]
gini = 0.0 samples = 19 value = [19, 0]
gini = 0.0 samples = 3 value = [0, 3]
gini = 0.456 samples = 222 value = [144, 78]
gini = 0.206 samples = 1400 value = [163, 1237]
gini = 0.053 samples = 3207 value = [3120, 87]
gini = 0.33 samples = 115 value = [24, 91]

(Visualization of the decision tree model)

Model Comparison Visualizations

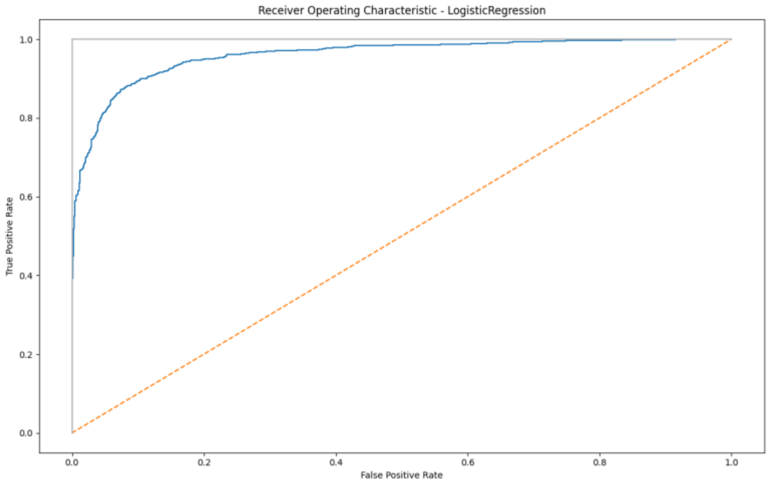
We decided that the best way to compare both models was to display visualizations that highlight the models' confusion matrices, metrics, and ROC curves.



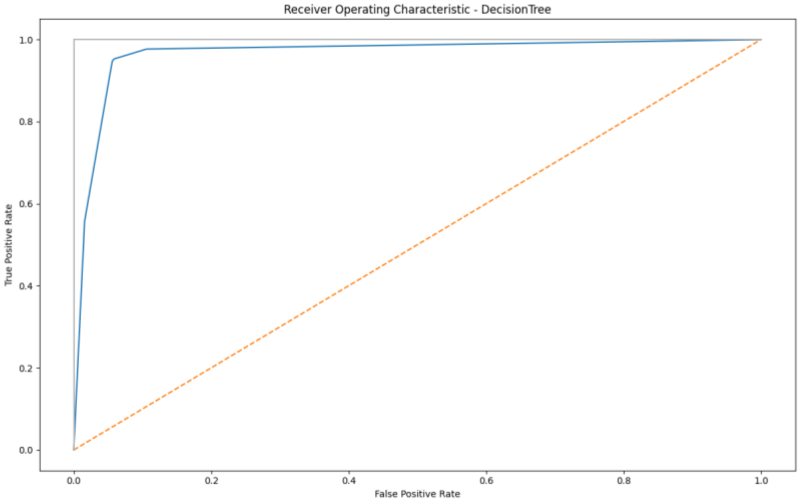
(Confusion matrix of logistic regression model, Confusion matrix of decision tree model)

	Logistic Regression	Decision Tree
Initial Accuracy	0.836	0.947
Precision	0.864	0.947
Recall	0.839	0.947
F1-Score	0.883	0.947
K-Fold Accuracy	0.884	0.942

(Metrics table for logistic regression and decision tree models)



(ROC curve for logistic regression model)



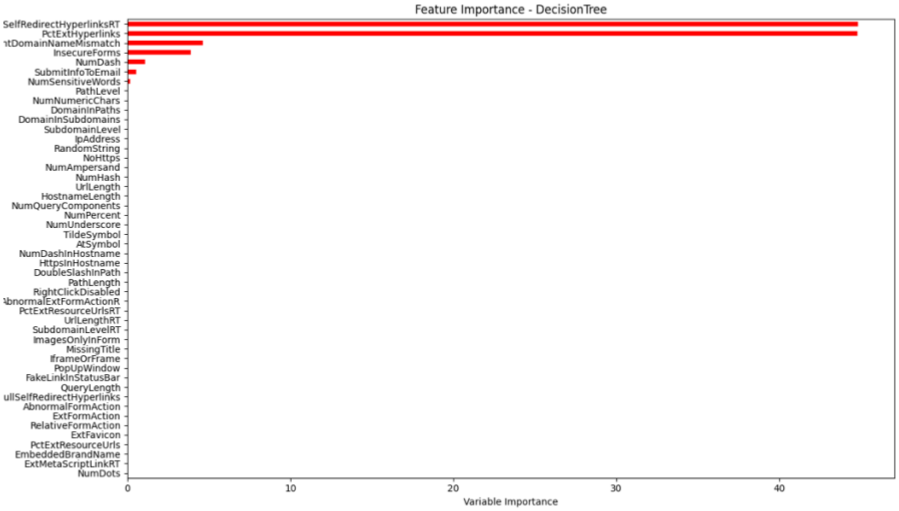
(ROC curve for decision tree model

Results

According to all our model evaluations it seemed that our decision tree model is overfit while our logistic regression is model is appropriately fit. All the former's metrics are incredibly high for example, the accuracy is above 90%. The latter's metrics are in a good range between 80% to 90%. Also, it is easy to perceive this from the ROC curves.

Conclusion

After running the models and analyzing the results we found that most variables are shared between both phishing and legitimate webpages with only a few features being strong indicators. This is due to phishing webpages sharing many similarities with legitimate webpages leading us to rely heavily on the few strong indicators we have. Those strong indicators act as the best way to tell the distinguish phishing webpages due to them being traits primarily held by them. According to the decision tree and importance table some of these strong indicators are 'SelfRedirectHyperlinksRT' and 'PctExtHyperlinks.' As mentioned earlier, we want to continue to work on this dataset. Our next steps include implementing new models such as Random Forest and comparing them to current models, using feature selection to see how dropping weak indicators will affect model accuracy, and tuning parameters for current models to see how we can continue to optimize them.



(Importance chart of decision tree model)

References

Government of Canada (2021), *Canadian Anti-Fraud Centre Annual Report*, retrieved from https://publications.gc.ca/collections/collection_2022/grc-rcmp/PS61-46-2021-eng.pdf