ANTI-PHISHING CLASSIFIER

Contents

Agrawal, Harshit – 90411685
Kalawatia, Noopur R K - 1980 9834
Kunal, Kumar - 5100 5964
Singh, Richa - 5044 5135

Problem Statement

Phishing is one of the most common online security threats which is targeted towards both users and companies. A phishing website poses itself as a legitimate business website to obtain users credentials and personal information. Using these phishers can access financial information such as bank details, credit card information, health insurance details, social security information etc. making the user vulnerable to identity theft and information breaches.

For this project, we created an anti-phishing classifier which uses various features of the website to detect if the website is a legitimate website or a phishing website. We utilized 10 different features broadly classified into the following features -

1. Website URL
2. Domain
3. Page Rank
4. Website content

Data Set and Experiment

Data Set

To train the model we needed sample which had both legitimate and phishing data. We pulled data from various sources –

1. PhishTank – public dataset containing ~10,000 phishing websites.
2. PhishStorm – dataset containing ~106,000 datapoints: ~39096 legitimate URLs and ~64,916 phishing URLs.
3. Alexa PageRank - ~1000 legitimate website URLs.

The dataset is segregated into training and testing datasets. The training dataset is used to perform k-fold validations over each of the models and the performance with each of the models is documented. The models are further evaluated using the testing data. This performance gives us the real-world application efficiency.

Experiment

1. Tools Used

1.1. Data Cleaning

- Pandas
- WhoIs

1.2. AWS

- EMR: In order to run the whois tool.
- EC2
- S3
- Boto3 (For AWS operations)

1.3. Data Visualization
- Matplotlib
- Seaborn

1.4. Classification
- Sklearn
- Numpy

1.5. UI application
- Tkinter
- Pandas

1. Data Cleaning, Feature extraction and Data Visulization

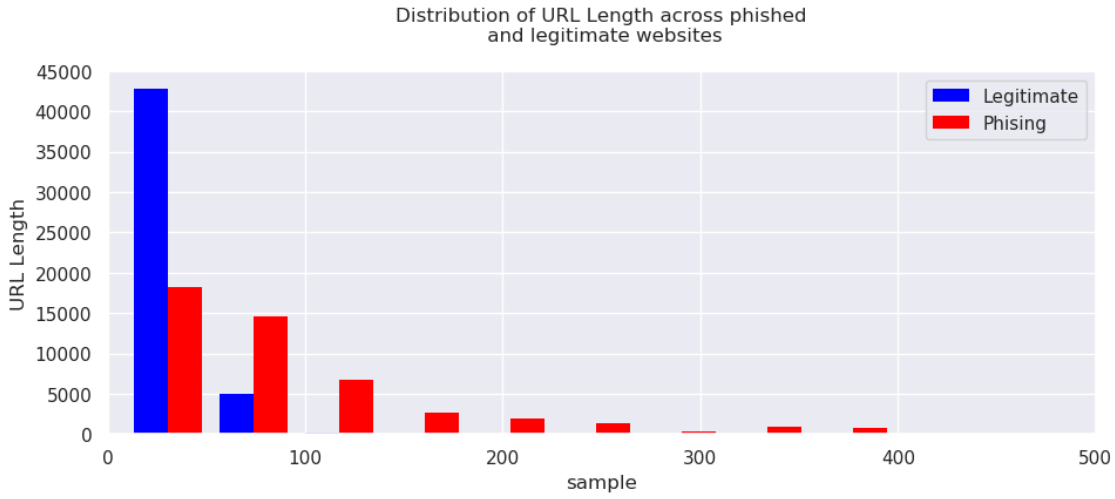
We merged the data from two sources mentioned above. We have utilized few features which were present in the existing dataset, but mostly extracted the other features using whoIs tool and also basic string manipulation functions.

Further we have also listed the mean and the standard deviation for each feature in the table shown at the end of this section.

1. Features–

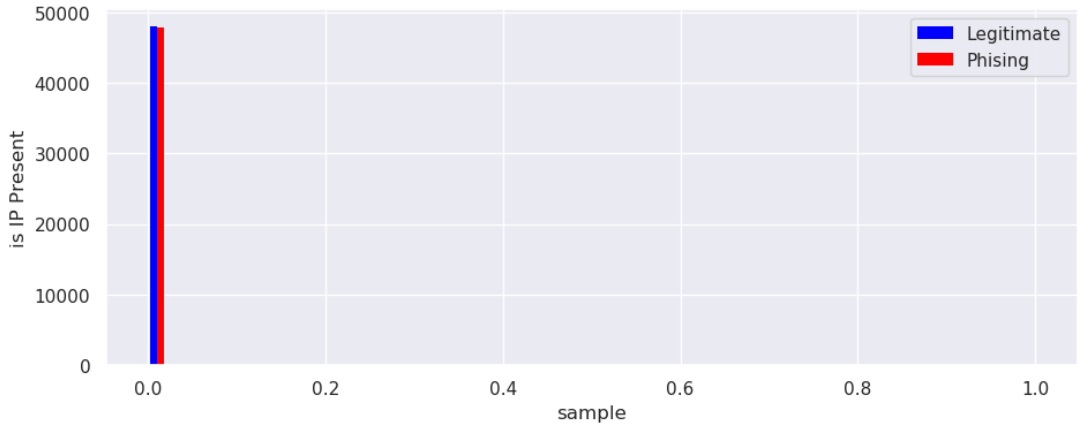
1.1. URL Length

Phishers generally uses long URLs to hide the doubtful part in the address bar. Our assumption was further cemented when the visualization of the data showed that the length of the URLs for legitimate websites almost is a constant while the distribution of the phishing URLs is scattered.



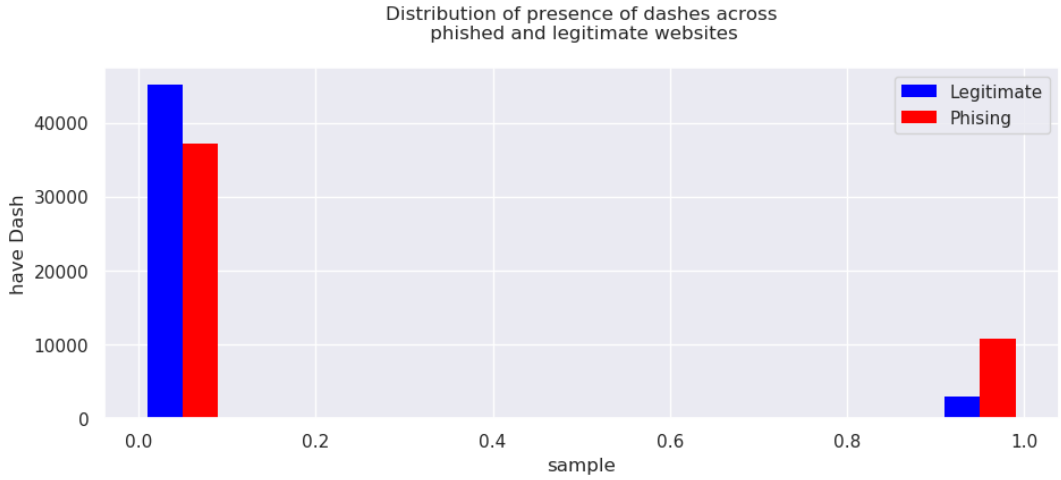
1.2. Presence of IP address in the URL

If an IP address is used as an alternative of the domain name in the URL users can be sure that someone is trying to steal their personal information. Although this feature clearly identifies a phishing website, but phishing websites are also evolving, hence, this doesn't play a strong role anymore.



1.3. Present of dash (-) in URL

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage.

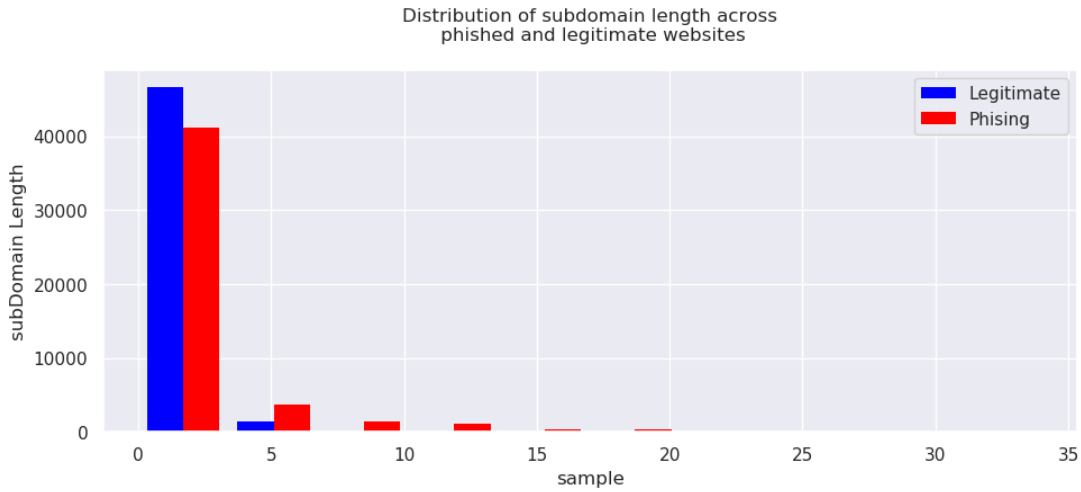


1.4. Number of Domain Names

Varying number of domains also is a sign of phishing website

1.5. Number of Sub Domains

If the dots are greater than two, it is classified as “Phishing” since it will have multiple sub domains.

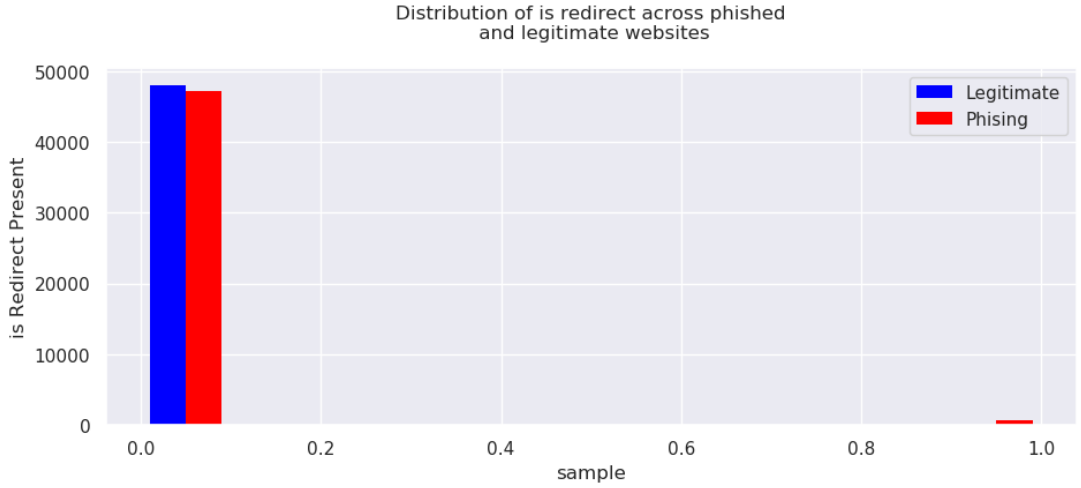


1.6. Present of @ symbol

Using “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.

1.7. Is Redirect link

If a website is a redirect website, identified by the presence of “//”, there is a high possibility that the website is a phishing website.



1.8. Domain name presence in WhoIs

From ICANN Lookup using WhoIs API we are identifying if the domain is registered or not.

1.9. Active Duration

If a domain is registered in WhoIs, we are also extracting active duration of the website. Generally, phishing websites have short active duration as they are quickly taken down once they are reported. For the missing data we added default value of ‘0’.

1.10. Page Ranking

We are using PageRank information provided in PhisStorm data set, and if not present we are using Google page ranking. From the PhishStorm data we observed that PageRank for phishing websites was set as 10,000,000. We noticed some patterns regarding PageRanking feature in the PhisStorm and emulated the same in the dataset acquired from other resources.

Mean and Standard Deviation of the Feature Set Considered

Feature	Type	Mean	Standard Deviation
Length of the URL	*Legitimate*	36.803995084255035	13.506704904452745
	Phishing	92.16715377228508	82.00782207638198
IsIP in URL	*Legitimate*	0.0	0.0
	Phishing	0.00037576719134900424	0.01938107299317842
HasRedirectZero	*Legitimate*	0.0	0.0
	Phishing	0.013569370798714042	0.11569461082885862
haveDashInURL	*Legitimate*	0.060051240392426423	0.23758175207653773
	Phishing	0.22412425368460606	0.41700428366495756
Domain Length	*Legitimate*	16.676998062863213	4.083401720773331
	Phishing	28.042545196442738	31.845283945595213
noOfSubDomain	*Legitimate*	2.120956487325293	0.5627639260266778
	Phishing	2.5628783766857333	2.70033449840379

2. Supervised learning models used with K-fold validations

The goal of using 10-fold validations is to obtain the best possible hyper parameter tuning over various runs over different divisions of the training partition of the dataset. Our goal is to compare how consistent each model is during the training and testing phase. This gives us an insight of how good the mined features are for all the different models and hints at whether the selected features need to be refined/rethought for more effective training. Performed Cross Validation with an 80-20 split for training and testing data for a data of ~106,000 datapoints

2.1. Logistic -Regression

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
score:  0.8805526315789474
```

```
Log-Regression
Accuracy:  0.8746797930584158
Classification report
              precision    recall  f1-score   support

        Spam       0.85      0.85      0.85      8233
  Legitimate       0.89      0.89      0.89     11676

    accuracy                           0.87     19909
   macro avg       0.87      0.87      0.87     19909
weighted avg       0.87      0.87      0.87     19909
```

2.2. K Nearest Neighbors

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=15, p=2,
                     weights='distance')
score:  0.9518157894736842
```

```
K-Nearest Neighbors
Accuracy:  0.9529358581546035
Classification report
              precision    recall  f1-score   support

        Spam       0.94      0.95      0.94      8233
  Legitimate       0.96      0.96      0.96     11676

    accuracy                           0.95     19909
   macro avg       0.95      0.95      0.95     19909
weighted avg       0.95      0.95      0.95     19909
```

2.3. Decision Tree

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
score:  0.9461184210526316
```

```
Decision Tree
Accuracy:  0.9468079762921292
Classification report
              precision    recall  f1-score   support

       Spam       0.93      0.94      0.94      8233
  Legitimate      0.96      0.95      0.95     11676

    accuracy                           0.95     19909
   macro avg      0.94      0.95      0.95     19909
weighted avg      0.95      0.95      0.95     19909
```

2.4. Random Forest

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=19, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=83,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
score:  0.9560263157894737
```

```
Random Forest
Accuracy:  0.9536390577125923
Classification report
              precision    recall  f1-score   support

        Spam       0.94      0.94      0.94      8233
  Legitimate       0.96      0.96      0.96     11676

    accuracy                           0.95     19909
   macro avg       0.95      0.95      0.95     19909
weighted avg       0.95      0.95      0.95     19909
```

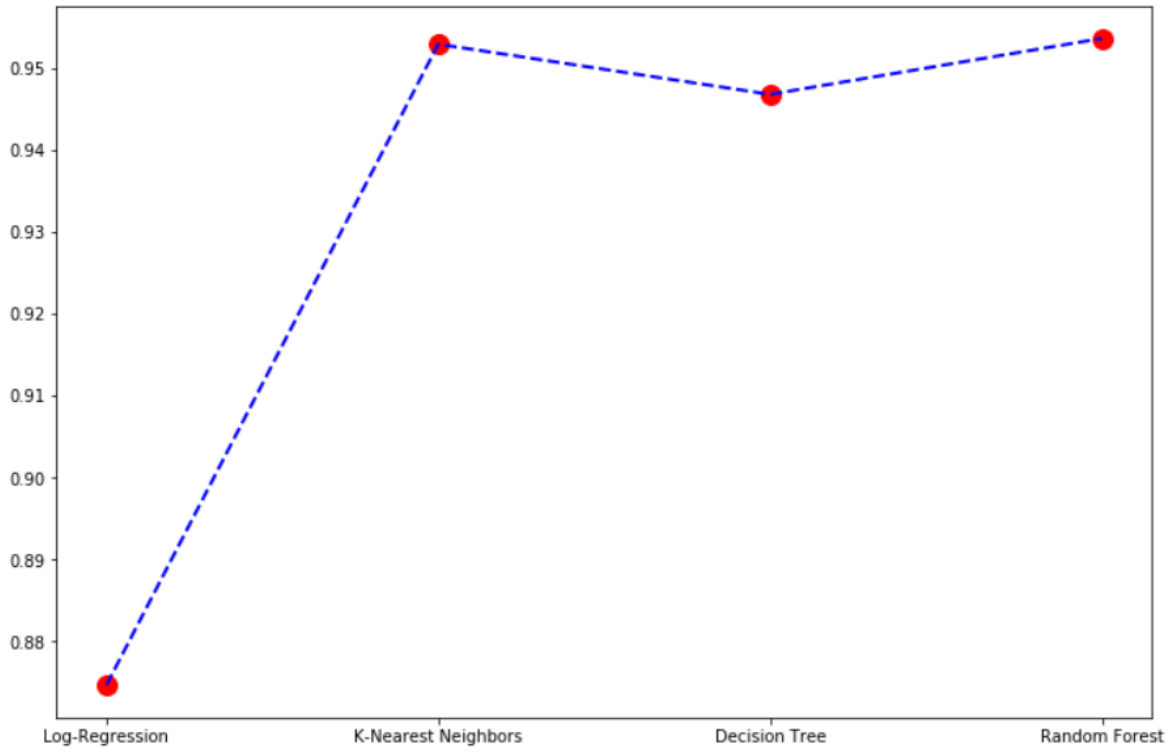
Evaluation

The metric used for analyzing the performance is accuracy. This mitigates the problems imposed by "accuracy" for imbalanced datasets.

Models used for Classification	**Accuracy**
LogisticRegression	87%
KNeighborsClassifier	95%
DecisionTreeClassifier	94.6%
RandomForestClassifier	95.4%

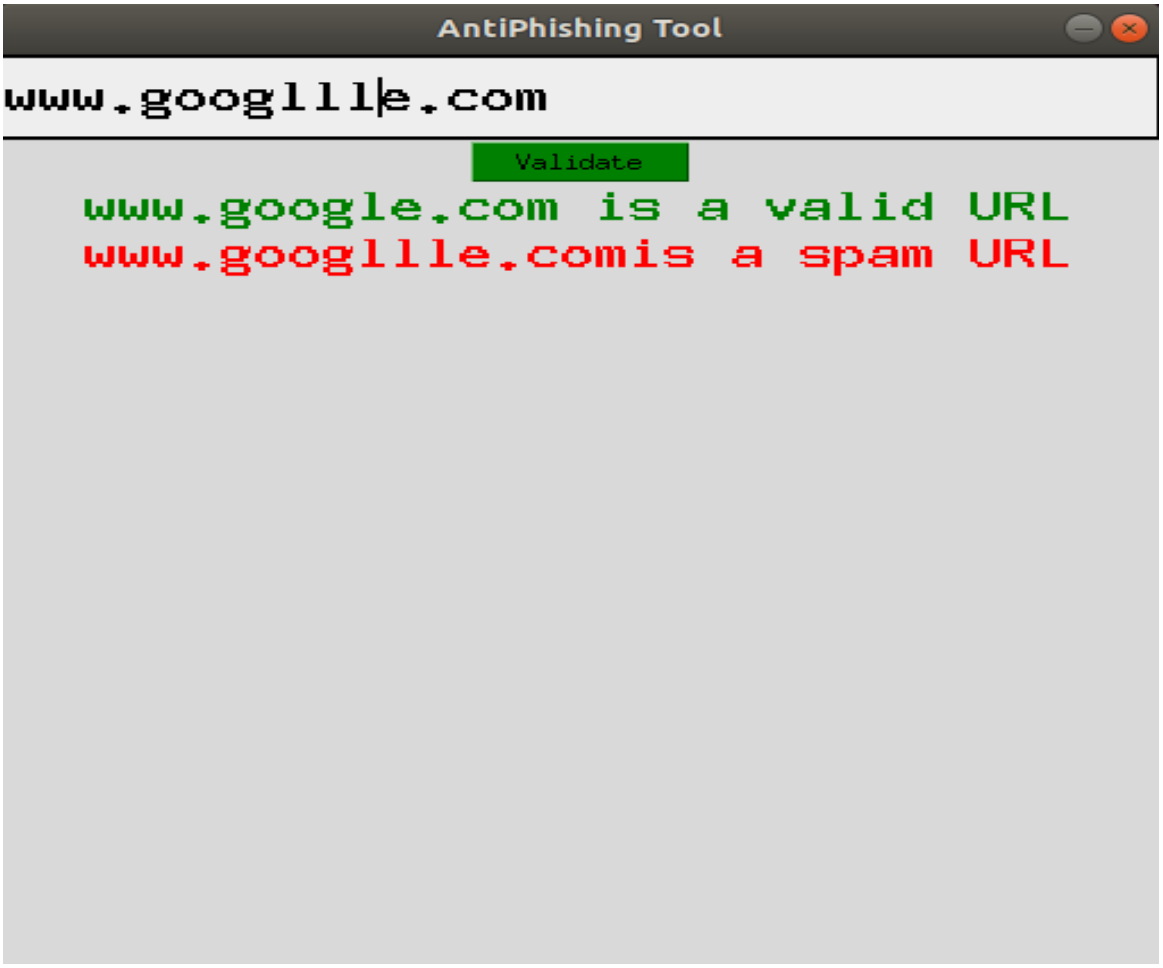
```
print(accuracies)


plt.figure(figsize=(12, 8))
plt.plot(classifiers, accuracies,'ro',markersize=12)
plt.plot(classifiers, accuracies,color = 'blue', linestyle = 'dashed',linewidth=2, markersize=12)
plt.show()
```

```
[0.8746797930584158, 0.9529358581546035, 0.9468079762921292, 0.9536390577125923]
```



Additions and Improvements

1. The dataset was increased and now we have a dataset of ~106,000 tuples. The dataset is acquired by prominent sources and is presently one of the largest dataset for phishing website detection.
2. Tweaked some features and fixed a bug while extracting isValid feature using WhoIs service which increased the accuracy of the model.
3. Used EMR cluster in order to execute whois over URLs for training dataset creation.
4. Created a UI based application using *Tkinter* in python, which takes user's input and identifies if the site is phishing or a legitimate site. The input is now the testing data tuple. The input is fed into the best performing model (we saved this model prior) in order to classify it as a phishing or a legitimate website. This can also be integrated with other applications and can be directly used to identify if the received link is phishing or not.



5. The dataset and the kernel book is uploaded on Kaggle. Our dataset has a size which is significantly bigger (10x) than present online resources to acquire legitimate and phishing websites. The data tuples are equally balanced and can be used freely by anyone to design their models.

Member Contributions

- Data Cleaning, feature creation and UI based application
  - Agrawal, Harshit
  - Singh, Richa
- Machine Learning models and Data visualization
  - Kalawatia, Noopur R K
  - Kunal, Kumar

GitHub and Kaggle Link

- GitHub – https://github.com/kunal4892/AntiPhishingClassifier
- Kaggle – https://www.kaggle.com/kunal4892/phishingandlegitimateurls