Phishing URL Classification With Machine Learning

By: Haiman Wong and James White
April 21st, 2021

Table of Contents

01. Problem + Motivation

Classify whether a URL is malicious or benign

02. Literature Review

"Distributed Representations of Sentences and Documents", by Q. Le and T. Mikolov, Google, 2014.

- *Problem and Motivation* - overcoming the weaknesses of bag-of-words models in the status quo
- *Approach and Solution* - researchers propose the Paragraph Vector model
  - An unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of text, such as sentences, paragraphs, and documents, where the vector representations are learned to predict the surrounding words in contexts sampled from the paragraph
- *Conclusion* - Paragraph Vector model's good performance in several text classification experiments demonstrates that the model is successful at capturing the semantics of paragraphs, overcoming the weaknesses of the bag-of-words model, and remaining competitive with conventional state-of-the-art methods

02. Literature review

"A Multi-Tier Phishing Detection and Filtering Approach", by R. Islam and J. Abawajy, Journal of Network and Computer Applications, 2013.

- *Problem and Motivation* - keep up with the scale and sophistication of phishing attacks that continue to steadily increase and improve phishing email-filtering
- *Approach and Solution* - researchers propose the Multi-Tier Classification Model
  - An innovative method for extracting the features of phishing email based on weighting of message content and header and select the features according to priority ranking
  - Researchers also examine the impact of rescheduling the classifier algorithms in a multi-tier classification process to find out the optimum scheduling
- *Conclusion* - The Multi-Tier Classification Model reduces the computational burden of the overall mail server system and can offer a comparable performance (97%). Rescheduling the classifier may vary the overall classification performance. Based on the performance, we can select the best scheduling of multi-tier classification and their application to phishing filtering process. Overall, the model demonstrates high accuracy retention and lower false positive instances.

02. LITERATURE REVIEW

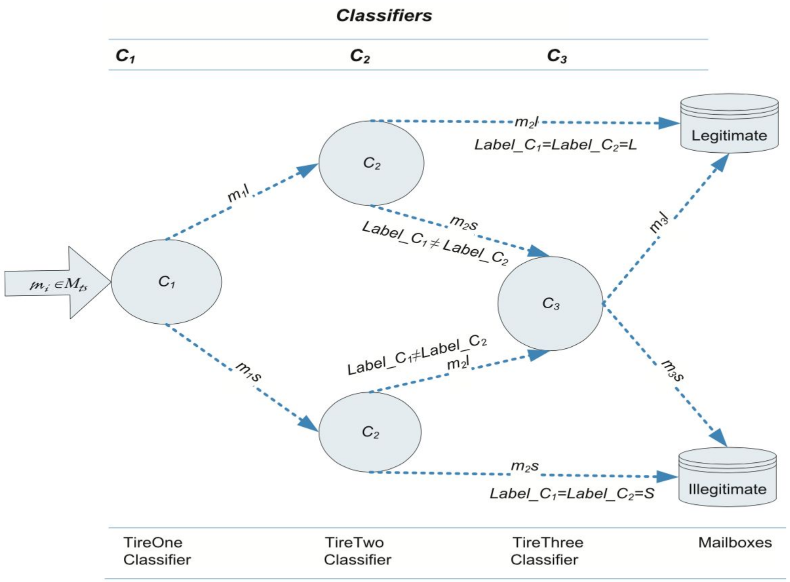


Fig. 1. Block diagram of the multi-tier classification model.

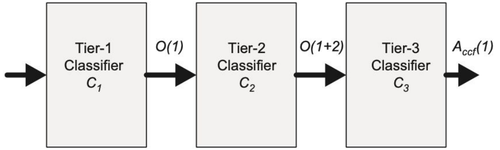


Fig. 3. The ML algorithm combinations for C_1–C_2–C_3.

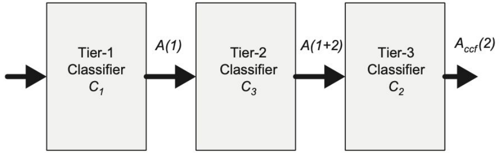


Fig. 4. The ML algorithm combinations for C_1–C_3–C_2.

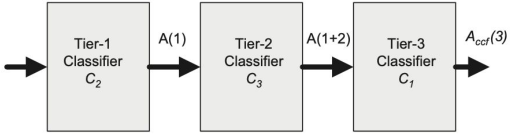


Fig. 5. The ML algorithm combinations for C_2–C_3–C_1.

02. LITERATURE REVIEW

"Email Phishing: An Enhanced Classification Model to Detect Malicious URLs", by S. Sankhwar, D. Pandey, and R. Khan, EAI Endorsed Transactions on Scalable Information Systems, 2019.

- ❖ *Problem and Motivation* - reduce a user's susceptibility to a phishing attack by improving the detection and classification of URLs
- ❖ *Approach and Solution* - researchers propose the Enhanced Malicious URL Detection (EMUD) Model
  - ➢ An algorithm that selects 14 heuristics to detect malicious or phishing URL using machine learning techniques, such as Naive Bayes (NB) and Support Vector Machine (SVM), as classifiers to differentiate between phishing and legitimate URLs and sites
  - ➢ The EMUD model then analyzes the URL set to detect whether the website is benign or malicious
- ❖ *Conclusion* - EMUD model demonstrates more effective detection capabilities because it uniquely includes more detection parameter (relevant URL heuristic) that catch and detect malicious URLs. The SVM classifier yielded the best results due to its shorter processing time, better accuracy, and results.

02. LITERATURE REVIEW

"Phishing Detection Through Email Embeddings", by L. Gutierrez, F. Abri, M. Armstrong, A. Namin, and K. Jones, Texas Tech University, 2020.

- *Problem and Motivation* - improve understanding of the phishing features that contribute to variations of the classifiers and investigate whether these features are sufficiently captured or disregarded by email embeddings or vectorizations
- *Approach and Solution* - researchers craft a set of phishing and legitimate emails with similar phishing indicators to investigate which indicators are detected and feed machine learning classifiers with the crafted emails to determine the email embedding performance
  - Employed Random Forest, Support Vector Machine, Logistic Regression, and Naive Bayes machine learning techniques as classifiers
- *Conclusion* - SVM reports the highest classification performance with an accuracy and F1 score of 81.6% and 76.6% in the original feature space. Using the linear PCA projection suggests that the underlying structure of the Doc2Vec document embeddings achieved the highest results and is likely to be linear.

03. THE DATA

Labeled Data

domain	ranking	isIp	valid	activeDuration	urlLen	is@	isredirect	haveDash	domainLen	nosOfSubdomain	label
www.voting-yahoo.com	10000000	0	0	0	20	0	0	1	20	2	1
www.zvon.org/xxl/WSDL1.1/Output/index.html	194914	0	1	7305	42	0	0	0	12	2	0
tecportais.com/file-security-update-infonfmation-pp-ll/nirvana.php?cmd=_login-run&dispatch=8cf66a395c5c3dda310e8fb8bd0bd8888cf66a395c5c3dda310e8fb8bd0bd888	10000000	0	0	0	155	0	0	0	14	1	1
bima.astro.umd.edu/nemo/linuxastro/	7001	0	0	0	35	0	0	0	18	3	0
huarui-tec.com/js/?us.battle.net/login/en/?ref=gofcuyeus.battle.net/d3/en/index	10000000	0	1	730	79	0	0	1	14	1	1
diannaopeizhi.com/js/	10000000	0	1	1096	21	0	0	0	17	1	1
www.synchrotech.com/support/install.html	10000000	0	1	12053	40	0	0	0	19	2	0
www.ansi.okstate.edu/breeds/swine/largeblackwhite/	23191	0	0	0	50	0	0	0	20	3	0
www.strum.co.uk/webbery/	10000000	0	0	0	24	0	0	0	15	3	0
www.grok2.com/vi-emacs.html	10000000	0	1	6210	27	0	0	0	13	2	0
www.pbs.org/newshour/topic/business/	1451	0	0	0	36	0	0	0	11	2	0
expertwear.pk/img/glyph/1/beveilings/online/index1.php	10000000	0	0	0	54	0	0	0	13	1	1
tools.ietf.org/html/rfc1162	14969	0	1	9133	27	0	0	0	14	2	0
www.iwiwueyrtrueruie.x10.mx/paypal./PayPal/paypal/webscrcmd=_login-run/cgi-bin/update/login/index.php	10000000	0	1	3651	101	0	0	0	27	3	1
www.perl.com/pub/a/2001/07/25/onion.html	193405	0	1	12783	40	0	0	0	12	2	0
www.autotrader.co.uk/BIKES/	1603	0	0	0	27	0	0	0	20	3	0
friendswoodexpress.homestead.com/tubes.html	3533	0	1	10591	43	0	0	0	32	2	0
tools.ietf.org/html/rfc1945	14969	0	1	9133	27	0	0	0	14	2	0
badluck42.tripod.com/CINDY.html	1267	0	1	9496	31	0	0	0	20	2	0
www.cliki.net/PLisp	10000000	0	1	6940	19	0	0	0	13	2	0
swlucylawless.tripod.com/BrendaSchad.htm	1267	0	1	9496	40	0	0	0	24	2	0
remax.com.jpginnovations.com/remax/index.htm	10000000	0	1	1827	44	0	0	0	28	3	1
www.sedit.com/rexxgrph.html	10000000	0	1	9133	27	0	0	0	13	2	0
thethinklab.com/ttl/wp-content//themes/westpa/west.html	10000000	0	1	5479	55	0	1	0	15	1	1
tools.ietf.org/html/rfc239	14969	0	1	9133	26	0	0	0	14	2	0
tools.ietf.org/html/rfc2339	14969	0	1	9133	27	0	0	0	14	2	0
www415.paypal.ca.20053.securessl-150.mx/js/web.apps/ca/m.pp/?cmnd=_lg&nav=_4792E694B2CF7F2A9778	10000000	0	0	0	95	0	0	1	39	5	1
paypal.com-us-cgi-bin.web.iscr.cmd-homeiielocale.xx13-en-us.vaidicastro.com/paypal/1c79f5bb3e07a1e02e1a42f955adfa41/webscr.php?cmd=_login-run&dispatch=5885d80a13c0db1f998ca054efbdf2c298	10000000	0	1	1095	283	0	0	1	75	7	1
www.marketingprofs.com/5/buresh8.asp	7644	0	1	7671	36	0	0	0	22	2	0
schoolopdracht.herobo.com/socialedienst/login_scr.html	10000000	0	1	3653	54	0	0	0	25	2	1
www.princeducation.com/paypl/webscr.html?cmd=5885d80a13c0db1f22d2300ef60a67593b79a4d03747447e6b625328d36121a11a7ab6441e2fc3f80ebe740491bfa9901a7ab6441e2fc3f80ebe740491bfa990	10000000	0	1	1827	173	0	0	0	22	2	1

12 unique features

Sample Size: 71, 677

Download at: https://www.kaggle.com/kunal4892/phishingandlegitimateurls

04. METHODOLOGY

Random forest classification

Decision trees

Logistic regression

Support vector machines (SVM)

05. Results: Decision Trees

Model 1

Accuracy Score (Training): 0.860
Accuracy Score (Test): 0.861
Criterion: Entropy
Max_Depth: 5
Max_Leaf_Nodes: 5

Model 2

Accuracy Score (Training): 0.900
Accuracy Score (Test): 0.898
Criterion: Entropy
Max_Depth: 15
Max_Leaf_Nodes: 15

Model 3

Accuracy Score (Training): 0.94813
Accuracy Score (Test): 0.94216
Criterion: Gini
Max_Depth: 300
Max_Leaf_Nodes: 300

Model 4

Accuracy Score (Training): 0.98369
Accuracy Score (Test): 0.94566
Criterion: Gini
Max_Depth: 10,000
Max_Leaf_Nodes: 10,000

Gridsearch

Accuracy Score: 0.931
Best Criterion: Gini
Best Max_Depth: 9
Best Min_Samples_Split: 2
Best Min_Samples_Leaf: 1
AUC Score: 0.959284

5-fold

Mean Cross-Validated Accuracy Scores: 0.948
Mean Cross-Validated AUC Scores: 0.962

05. Decision tree source code

Best decision tree variation

```
lab_clf5 = DecisionTreeClassifier(criterion='gini', max_depth=10000, max_leaf_nodes=10000)
lab_clf5.fit(X_train, y_train)
print("Accuracy on training set: {:.5f}".format(lab_clf5.score(X_train, y_train)))
print("Accuracy on test set: {:.5f}".format(lab_clf5.score(X_test, y_test)))
```

```
Accuracy on training set: 0.98369
Accuracy on test set: 0.94566
```

Grid search CV

```
from sklearn.model_selection import GridSearchCV
classifier = GridSearchCV(dt, hyperparam_grid)
classifier.fit(X_train, y_train)
predicted_gs_classifier = classifier.predict(X_test)
print("Accuracy: %.3f" % metrics.accuracy_score(y_test, predicted_gs_classifier ))
print('Best Criterion: %s' % classifier.best_estimator_.criterion)
print('Best Max_Depth: %s' % classifier.best_estimator_.max_depth)
print('Best Min_Samples_Split: %s' % classifier.best_estimator_.min_samples_split)
print('Best Min_Samples_Leaf: %s' % classifier.best_estimator_.min_samples_leaf)
```

```
Accuracy: 0.931
Best Criterion: gini
Best Max_Depth: 9
Best Min_Samples_Split: 2
Best Min_Samples_Leaf: 1
```

5-fold cross validation

```
y_pred = cross_val_predict(dt, X, y, cv=5)
print('Cross-validated predictions: ', y_pred)
```

```
Cross-validated predictions:  [1 0 1 ... 1 0 1]
```

```
y_scores = cross_val_score(dt, X, y, cv=5, scoring='accuracy')
print('Cross-validated accuracy scores: ', y_scores)
print('Mean cross-validated accuracy scores: ', y_scores.mean())

y_scores_auc = cross_val_score(dt, X, y, cv=5, scoring='roc_auc')
print('Cross-validated auc scores: ', y_scores_auc)
print('Mean cross-validated auc scores: ', y_scores_auc.mean())
```

```
Cross-validated accuracy scores:  [0.94781566 0.94854551 0.94599103 0.94859764 0.94713794]
Mean cross-validated accuracy scores:  0.9476175581274111
Cross-validated auc scores:  [0.96155114 0.96186445 0.95965137 0.96307021 0.96022679]
Mean cross-validated auc scores:  0.9612727927799473
```

05. Results: random forest

Model 1

Accuracy Score (Training): 0.983
Accuracy Score (Test): 0.957
Criterion: gini
Max_depth: default

Model 2

Accuracy Score (Training): 0.983
Accuracy Score (Test): 0.954
Criterion: entropy
Max_depth: default

Model 3

Accuracy Score (Training): 0.910
Accuracy Score (Test): 0.906
Criterion: gini
Max_depth: 5

Model 4

Accuracy Score (Training): 0.952
Accuracy Score (Test): 0.943
Criterion: entropy
Max_depth: 12

Gridsearch

Accuracy Score: 0.958
Best Criterion: Gini
Best Max_Depth: 9
Best Min_Samples_Split: 2
Best Min_Samples_Leaf: 1
AUC Score: 0.9885

5-fold

Mean Cross-Validated Accuracy Scores: 0.956
Mean Cross-Validated AUC Scores: 0.989

05. Random Forest Source Code

Best Random Forest Variation

```
rfc = RandomForestClassifier(criterion = 'gini')
rfc.fit(X_train,y_train)
predicted_rfc = rfc.predict(X_test)
X = df5[feature_col]
y = df5['label'].values
fpr,tpr,thresh = roc_curve(y_test, predicted_rfc)
roc_auc  = accuracy_score(y_test, predicted_rfc)
y_pred_prob = rfc.predict_proba(X_test)[:, 1]
print("AUC Score: ", metrics.roc_auc_score(y_test, y_pred_prob))
print('Training Accuracy :',rfc.score(X_train, y_train))
print('Testing Accuracy :',rfc.score(X_test, y_test))
```

```
Accuracy with RF classifier: 0.9574610059220953
AUC Score:  0.9895044066636701
Training Accuracy : 0.9827197909136407
Testing Accuracy : 0.9574610059220953
```

5-Fold Cross Validation

```
y_pred = cross_val_predict(rfc, X, y, cv=5)
print('Cross-validated predictions: ', y_pred)

y_scores = cross_val_score(rfc, X, y, cv=5, scoring='accuracy')
print('Cross-validated accuracy scores: ', y_scores)
print('Mean cross-validated accuracy scores: ', y_scores.mean())

y_scores_auc = cross_val_score(rfc, X, y, cv=5, scoring='roc_auc')
print('Cross-validated auc scores: ', y_scores_auc)
print('Mean cross-validated auc scores: ', y_scores_auc.mean())
```

```
Cross-validated predictions:  [1 0 1 ... 1 0 1]
Cross-validated accuracy scores:  [0.95667814 0.95730372 0.95610468 0.95834637 0.95480138]
Mean cross-validated accuracy scores:  0.9566468564279011
Cross-validated auc scores:  [0.98927242 0.98931184 0.9895165  0.99013417 0.98852626]
Mean cross-validated auc scores:  0.9893522392277898
```

Grid Search CV

```
rfc = RandomForestClassifier()
param_grid = {
    'n_estimators': [50, 150],
    'max_features': ['auto', 'sqrt', 'log2'],
    'criterion' :['gini', 'entropy']
}
classifier_rf = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
classifier_rf.fit(X_train, y_train)
predicted_rf_classifier = classifier_rf.predict(X_test)
print("Accuracy: %.3f" % metrics.accuracy_score(y_test, predicted_rf_classifier))
classifier.best_params_
```

```
Accuracy: 0.957
```

05. Results: Logistic Regression

Model 1

Accuracy Score (Training): 0.726
Accuracy Score (Test): 0.725
Solver: liblinear

Model 2

Accuracy Score (Training): 0.732
Accuracy Score (Test): 0.732
Solver: lbfgs

Model 3

Accuracy Score (Training): 0.839
Accuracy Score (Test): 0.837
Solver: newton-cg

Model 4

Accuracy Score (Training): 0.711
Accuracy Score (Test): 0.710
Solver: sag

Model 5

Accuracy Score (Training): 0.709
Accuracy Score (Test): 0.710
Solver: saga

5-Fold

Mean Cross-Validated Accuracy Scores: 0.744
Mean Cross-Validated AUC Scores: 0.876

05. Logistic regression source code

Best Logistic Regression variation

```
logreg3 = LogisticRegression(solver='newton-cg')
logreg3.fit(X_train,y_train)
predicted_lr3 = logreg3.predict(X_test)
X = df5[feature_col]
y = df5['label'].values
fpr,tpr,thresh = roc_curve(y_test,predicted_lr3)
roc_auc = accuracy_score(y_test,predicted_lr3)
print('Logreg Training Accuracy :',logreg3.score(X_train, y_train))
print('Logreg Testing Accuracy :',logreg3.score(X_test, y_test))
print('Logistic Regression AUC: ', metrics.roc_auc_score(y_test,
      logreg3.predict_proba(X_test)[:,1]))
```

```
Logreg Training Accuracy : 0.8424067174553745
Logreg Testing Accuracy : 0.8422303778463591
Logistic Regression AUC:  0.93218140049362
```

5-fold cross validation

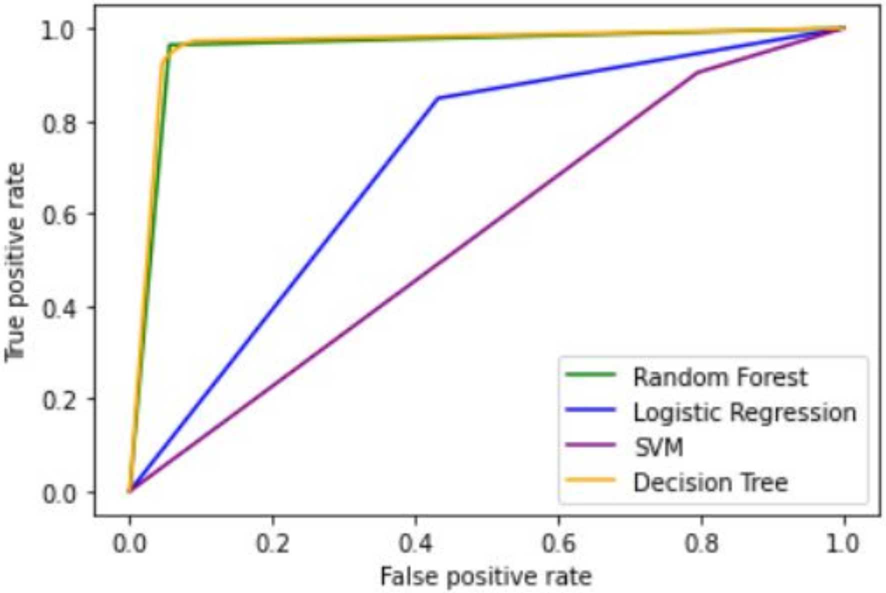
```
y_pred = cross_val_predict(logreg, X, y, cv=5)
print('Cross-validated predictions: ', y_pred)
y_scores = cross_val_score(logreg, X, y, cv=5, scoring='accuracy')
print('Cross-validated accuracy scores: ', y_scores)
print('Mean cross-validated accuracy scores: ', y_scores.mean())
y_scores_auc = cross_val_score(logreg, X, y, cv=5, scoring='roc_auc')
print('Cross-validated auc scores: ', y_scores_auc)
print('Mean cross-validated auc scores: ', y_scores_auc.mean())
```

```
Cross-validated predictions:  [1 0 1 ... 1 0 1]
Cross-validated accuracy scores:  [0.72964237 0.72510687 0.72901679 0.81034303 0.72599312]
Mean cross-validated accuracy scores:  0.7440204358252529
Cross-validated auc scores:  [0.86828515 0.8680556  0.86643645 0.907838   0.86750212]
Mean cross-validated auc scores:  0.8756234658115674
```

05. Results: SVM

Model 1	Model 2	Model 3
Accuracy Score (Training): 0.791 Accuracy Score (Test): 0.787 Kernel: linear Max_iter: 4000	Accuracy Score (Training): 0.557 Accuracy Score (Test): 0.555 Kernel: poly Max_iter: 4500	Accuracy Score (Training): 0.583 Accuracy Score (Test): 0.582 Kernel: sigmoid Max_iter: 5500
Model 4	**Model 5**	**5-Fold**
Accuracy Score (Training): 0.593 Accuracy Score (Test): 0.592 Kernel: sigmoid Max_iter: 6000	Accuracy Score (Training): 0.767 Accuracy Score (Test): 0.767 Kernel: rbf Max_iter: 4000	Mean Cross-Validated Accuracy Scores: 0.708 Mean Cross-Validated AUC Scores: 0.794

Comparative ROC curves



05. Comparative results summary

Best Decision tree

Accuracy Score (Training): 0.98340
Accuracy Score (Test): 0.94583
Criterion: Gini
Max_Depth: 10,000
Max_Leaf_Nodes: 10,000

Best Random forest

Accuracy Score (Training): 0.983
Accuracy Score (Test): 0.956
Criterion: Gini
Max_Depth: Default

Best SVM

Accuracy Score (Training): 0.791
Accuracy Score (Test): 0.787
Kernel: Linear
Max_Iter: 4000

Best Logistic regression

Accuracy Score (Training): 0.839
Accuracy Score (Test): 0.837
Solver: Newton-CG

Best 5-fold, ROC, and AUC

5-Fold Mean Cross-Validated Accuracy Score: 0.957
5-Fold Mean Cross-Validated AUC Score: 0.989
ROC: Random Forest
AUC: 0.989

Best Confusion matrix

Precision: 0.96
Recall: 0.96
F1 Score: 0.96
Support: 23978
FP: 486
TP: 13586
TN:9360
FN: 546

06. DISCUSSION

Opportunities and recommendations for future work and research include:

- ❖ Connecting our models to web-browser extensions that can monitor potential phishing URLs and safeguard human error
- ❖ Proposing a new model that synthesizes across our best model results
- ❖ Developing our own dataset from scratch

THANKS

Questions or Feedback?