COVID-19 phish domain detection by Machine learning project summarization report

Yansong Li, 300083962

I. Abstract

The object of this project is aimed at detecting malicious phishing websites related to COVID-19 misleading information which might cause unexpected finical loss or personal information leakage, This project utilizes supervised and unsupervised learning algorithms in machine learning to classify and identify Covid-19-related malicious domain. Since the majority of our dataset is unlabeled, we implement the LSTM+atuoencoder mechanism, 1 neighbor KNN on our unlabeled data with time series feature and cross valid by the best estimator trained by sample data with high confidence label.

II. Introduction

Since our main data set are grouped into two clusters with labeled and unlabeled, and the majority of our data are unlabeled. We first implement DT(decision tree) with tuned parameters and visualize the tree structure in order to observe the main features which have higher weight in classification. And cross valid the result with the information gain score calculated by extra tree classifier. Detailed information would be specifically introduced in section IV 'Data observation by ML methodology'. Then we trained RF(random forest), ET(extra tree classifier), KNN(K nearest neighbor), NB(naive Bayes), and MLP(Multilayer perception model) with tuned parameter and choose the best model as our supervisor model to validate the result produced by the clustering algorithm in section V 'Outlier detection for unlabeled data'. Since we don't have a solid label for our unlabeled data set. extracting rules from sample data with labels and valid on the unlabeled data appears to be the only option to examine outlier detection model performance.

In section V, our original task is binary classification on identifying malicious Covid-19 related websites, in this case, we convert positive label in section three as outlier in section V, we implement three different methodologies which are clustering, 1-NN classification and LSTM-autoencoder mechanism and valid the result by section three's supervisor model.

III. Feature introduction

The final unlabeled data and sample data with label contain following features which are obtained by web crawler and reliable resource.

1) **Unified URL:** The URL of our dataset is obtained from PhishTank(related to COVID-19), RISKIQ, Spycloud, the whitelist of government official websites, and the first 500 pages of Covid-19-related main domain names crawled by the BING search engine.
2) **Way back archived:** Binary feature on checking whether the domain has record on Wayback machine(which record previous log of website). We believe the URL with full way back machine record has higher confidence to be classified as legitimate website.
3) **Freenom toplevel domain:** Binary feature on checking whether the top level domain of each corresponding URL has record on Freenom, since it provides free domain name with no verification, the domain recorded in Freenom is more likely to be malicious website.
4) **Previous malicious top level domain TLD:** Binary feature on checking whether the top level domain of each corresponding URL has record on the collection of suspicious TLD records. The collection of suspicious TLD dataset is based on Akamai high abuse TLD rank[2].
5) **Name length:** Ordinal feature on counting valid domain length(number of characters) by dropping 'http/https://' prefix.
6) **Wrong spell List:** Binary feature on checking whether the Covid-19 related keywords in domain are right spelled. Some malicious websites would intentionally misspell the keyword in domain name to fake legitimate websites.
7) **Longest word ratio:** Ordinal feature matches the longest English word that a domain name contains, and normalizes it by dividing by the length of the domain name. Attackers may generate pseudo-random names to avoid conflict with existing domains, or deliberately include readable words in the domain names to attract clicks from victims.
8) **sub domain:** Ordinal feature counts the number of subdomains belonging to a TLD. For example, in 'forum.yoursite.com', 'forum' is the subdomain of 'yoursite'. Therefore the total number of subdomain is 1 in this case.
9) **levenshtein distance:** Ordinal feature examines the similarity of a domain with a set of known malicious domains. We derive a set of previously known malicious domains based on prior knowledge, compute the Levenshtein edit distances to the currently considered domain and choose the minimum value as our feature

for each domain.

10) **Alexa rank:** Ordinal feature checks whether the domain appears in Alexa3 top 1 million, and if so records the rank. Domains in the Alexa ranking are more likely to be legitimately owned domains.
11) **Certificate period:** Ordinal feature check the verification period of each domain on Google Transparency Report[1](With registration record and issuer information), if there's no record, the value would be set as default 0.
12) **Start date:** Ordinal feature on checking the registration date on Whois server with python backend support.
13) **Created on 2020:** Binary feature on checking whether the URL was created on 2020.
14) **wildcard_subdomain:** Binary feature checks whether the domain is registered to accept all subdomains, known as a "wildcard" subdomain. A wildcard DNS record will match any request for an undefined domain name. It is then easy for attackers to advertise a working URL with a subdomain of their choosing despite not controlling the DNS entries of the domain itself. This may influence determination of whether a domain is compromised or malicious.
15) **Redirect_URL:** This feature checks whether access to the domain redirects to a different domain. This constitutes a suspicious flag adding to other features.
16) **Contain_Weried_number_combination:** Binary feature on checking whether the URL contain IP address or number series divided by dot symbol.

A. Data prepossessing

a) Since 'start date' has 21% null value in unlabeled data and 40% null in sample data, we implement different strategies according to the base amount of each data set, for the sample data, we manually check the websites without start date through log history and script info with browser inspection mode. And for the unlabeled data, we simply cut the data set without the start date, since the start date is fundamental information of a website which is sensitive to our subject 'Covid-19' due to the real-time tendency. We considered implementing Null-value imputing algorithms could possibly add bias and unrealistic information to our data set.

b) For the rest features, we implement, we implement StandardScale standardization methodology which standardizes features by removing the mean and scaling to unit variance. The standard score of sample x is calculated as:

$$z = (x - u)/s$$

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False. In this case, we transfer the whole data frame as a tensor and feed into Sklearn TransformerMixin which would directly transform the whole dataset according to each corresponding feature.

c) In order to fit the unlabeled data into our LSTM+Autoencoder model. We convert the tensor from $(None, 12)$ to $(None, 1, 12)$ where None represents the total number of our training data set, 1 represents the timestep in LSTM structure, 12 represents the total number of features in training data. Detailed information would be explained in the following model introduction.

B. Data inspection

The following table describes the specific data information

TABLE I
NONLINEAR MODEL RESULTS

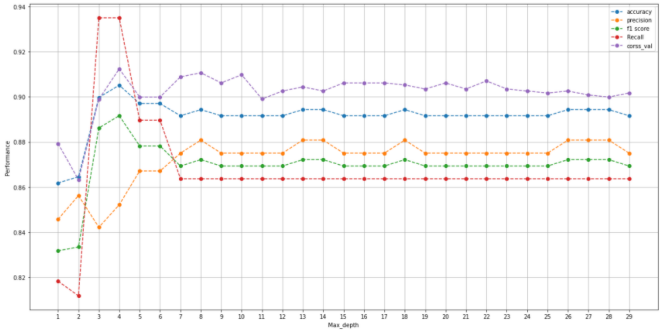
	Sample data(With label)	unlabeled data
# of features	12	12
# of rows(Without Null)	1117	89779
Pos/Neg rate	39.6%/60.4%	Unknown

IV. DATA OBSERVATION BY ML METHODOLOGY

A. DT visualization and ET observation

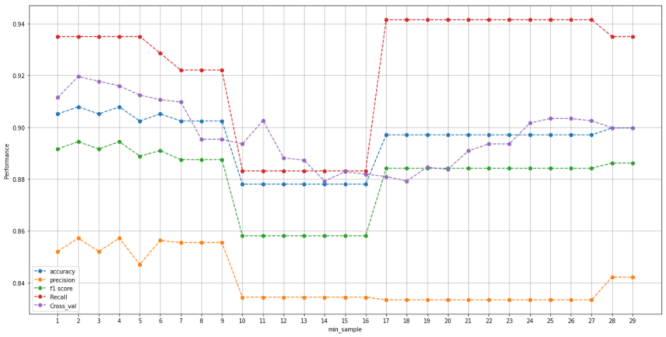
We first fine-tune decision tree's parameters, max_depth and min_sample at leaf node. As what Fig1 illustrates, when depth = 4, mean_cross validation score reach the top. Hence we chose 4 as our max depth of decision tree. Then as Fig2 shows, when min sample size = 2, the model reach the optimal mean cross validation score.

Fig. 1. Correlation between max_depth and performance



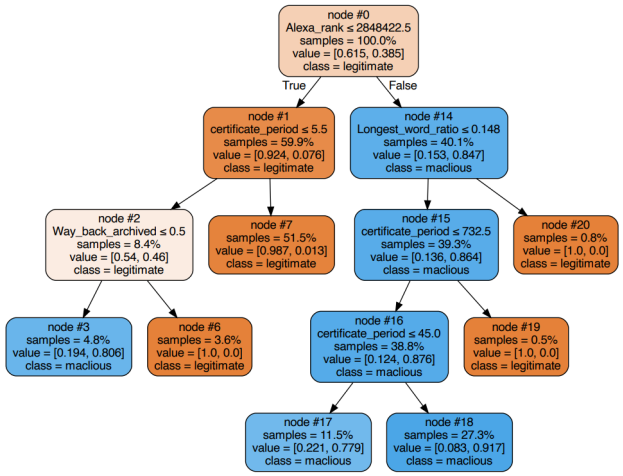
After tuning the DT parameters, we visualize the DT structure accordingly. In Fig3, each node is represented by five different values(node, feature,sample,value,class) where node represents the serial number of the node in DT, feature represents the splitting condition, sample represents the percentage of remaining data, value is a list with ratio of positive and negative label, class represents the label.

Fig. 2. Correlation between min_sample and performance



According to Fig3, we observe that certificate_period, Alexa_rank, Longest_word_ratio are the main factors in DT classification procedure. Especially when the domain has longer certificate period and solid way back machine archived. It's more likely to be legitimate domain.

Fig. 3. DT visualization



In order to quantify the weights of these features selected by the decision tree, we implement ET to calculate information gain for each features. As Fig4 shows

Fig. 4. ET information gain

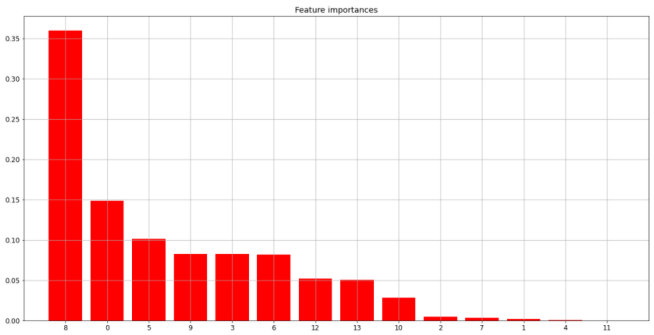


TABLE II
TOP 9 FEATURES

Feature name	Feature #	Gain value
Contain_Weried_number_combination	8	0.359945
Certificate_period	0	0.148983
Wrong_spell_List	5	0.101318
Alexa_rank	9	0.082754
Previous_malicious_top_level_domain_TLD	3	0.082633
Longest_word_ratio	6	0.082149
Redirect_URL	12	0.052189
Created on 2020	13	0.050640
Way_back_archived	10	0.082149

The overall features can be divided into two type: Log history and domain written structure. According to ET's information gain table and DT's visualization, we can observe that during the decision procedure, whether the domain related to COVID-19 is legally registered by Alexa rank, way back achieved machine, Google transparency report are real-time key factors as type1 for classification. Whether the keywords in URL are correctly spelled and the whole structure is well organized are key factors as type2 for classification.

B. Supervised learning algorithms construction and comparison

a) **KNN:** Number of neighbors is tuned during the training phase, the final optimal parameters are $(n\_neighbors = 3, weights =' distance', p = 2)$

b) **RF:** Number of estimators and max depth are tuned during the training phase, the final optimal parameters are $(n\_estimators = 54, max\_depth = 9, max\_features =' auto', criterion =' entropy')$

c) **ET:** Number of estimators and max depth are tuned during the training phase, the final optimal parameters are $(n\_estimators = 54, max\_depth = 13, random\_state = 0, criterion =' entropy')$

d) **NB:** Gaussian Naive Bayes (GaussianNB) $priors = None, var\_smoothing = 1e-09$.

e) **MLP:** $Optimizer = Adam, learningrate = 0.001, loss = binary\_crossentropy, epochs = 300, batch\_size = 16$

The following table shows the performance of each supervised learning algorithm under labeled data. We use 10 fold cross-validation as the main evaluation basis. And then we use fried man test as main significance comparison mechanism.

We can obviously observe that the tree structure algorithm has similar results and the best classification performance. In the next section, we would use one of the tree structure based algorithms as supervisor classifier to test the result of our outlier detection algorithms.

TABLE III

ALGORITHMS PERFORMANCE COMPARISION

Name	Accuracy	Precision	Recall	F1 score	Mean cross val score
DT	91.1%	89.6%	93.5%	89.6%	91.5%
KNN	89.7%	89.6%	86.3%	87.5%	87.0%
RF	92.1%	89.7%	93.5%	90.1%	91.6%
NB	85.1%	84.6%	78.6%	81.5%	87.7%
ET	91.9%	90.4%	92.2%	90.3%	91.6%
MLP	85.1%	84.6%	78.6%	81.5%	–

TABLE IV

FRIED-MAN TEST ON SIGNIFICANCE COMPARISON

Name	P-Value	Statistics	threshold	result
DT-RF-ET	0.128	4.111%	0.05	Same distributions
DT-KNN-NB	0.021	7.684	0.05	Different distributions
KNN-NB-MLP	0.039	6.500	0.05	Different distributions

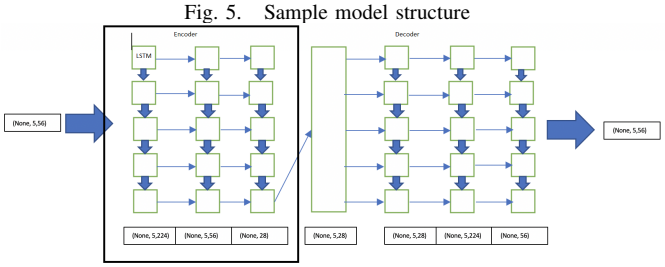
V. OUTLIER DETECTION FOR UNLABELED DATA

A. AE(Autoencoder)+LSTM

In this section, we use Autoencoder mechanism to detect outlier in our unlabeled dataset. The essential principle of Autoencoder is to encrypt, compress and decrypt the input data. We use Autoencoder to iteratively train on our data, and optimize the loss by calculating the MAE between the generated data and the input data. The lower the loss, the closer the AE generated data is to the input data, which means the AE model can better capture the latent space features of the data. The reason why we choose LSTM as basic cell in AE model is because our data has time series feature ”start date”, since our topic ”COVID-19” is real-time event which is rapidly breaking out in 2020. Year ’2020’ could possibly be one of the key factors on filtering Phishing websites created in real time and legitimate websites of health organizations with a long history.

1) Model configuration and introduction: The model parameters are shown in Table 5, and the model structure is shown in Figure 6. The model input data dimension is (amount, 1, 15). The amount is the total number of data, and 1 is the timestep. Here, because each domain only corresponds to one registration date, set this to 1, and the last dimension is the total number of features.Figure 5 is a sample structure of the sub-model

Fig. 5. Sample model structure



The overall structure of the model consists of three parts. The encoder consists of two layers of LSTM and dense layer. The data is compressed by passing the hidden state of LSTM, and the return sequnce of the last layer of LSTM is set to False to reduce dimensionality. Then use repeat vector to achieve dimensional expansion, and then access LSTM to restore the information through the hidden state.

After the model training is over, we calculate the MAE by calculating each individual data and the tensor generated by AE+LSTM, and draw the histogram according to the MAE, so that we can select the threshold to divide the outlier by the Loss distribution of the histogram.

TABLE V

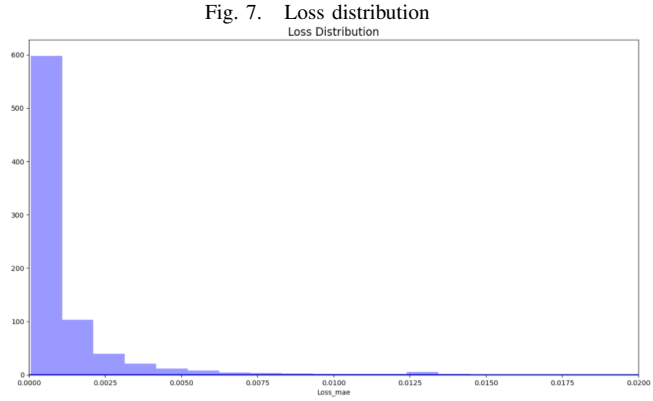
FRIED-MAN TEST ON SIGNIFICANCE COMPARISON

Parameter	Value
Optimizer	Adam
initial_learning_rate	1e-3
latent_space size	15
Epoch	100
batch_size	5

Fig. 6. AE+LSTM structure

```
Model: "lstm_encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 15)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 1, 60)             18240
_________________________________________________________________
lstm_1 (LSTM)                (None, 15)                4560
_________________________________________________________________
dense_1 (Dense)              (None, 15)                240
=================================================================
Total params: 23,040
Trainable params: 23,040
Non-trainable params: 0
_________________________________________________________________
Model: "lstm_decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 15)]              0
_________________________________________________________________
repeat_vector (RepeatVector) (None, 1, 15)             0
_________________________________________________________________
lstm_2 (LSTM)                (None, 1, 15)             1860
_________________________________________________________________
lstm_3 (LSTM)                (None, 1, 60)             18240
_________________________________________________________________
time_distributed (TimeDistri (None, 1, 15)             915
=================================================================
Total params: 21,015
Trainable params: 21,015
Non-trainable params: 0
_________________________________________________________________
Model: "lstm_autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 15)]           0
_________________________________________________________________
lstm_encoder (Functional)    (None, 15)                23040
_________________________________________________________________
lstm_decoder (Functional)    (None, 1, 15)             21015
=================================================================
Total params: 44,055
Trainable params: 44,055
Non-trainable params: 0
_________________________________________________________________
```

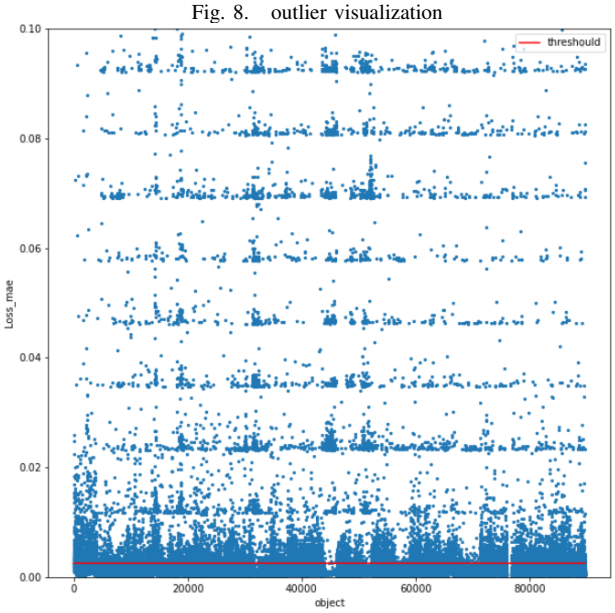
Fig. 7. Loss distribution



According to the loss distribution, we set threshold to 0.0035, for the data whose MAE higher than threshold would be considered as outlier.

In order to test AE+LSTM model's performance, we first test the model with sample data, Accuracy is chosen as main measurement, AE+LSTM get 62.3% accuracy on sample data, then implement the supervisor algorithm(RF) selected from section IV on the unlabeled data and compare the prediction similarity between RF and AE+LSTM, the similarity score is 62.4% in this case.

Fig. 8. outlier visualization

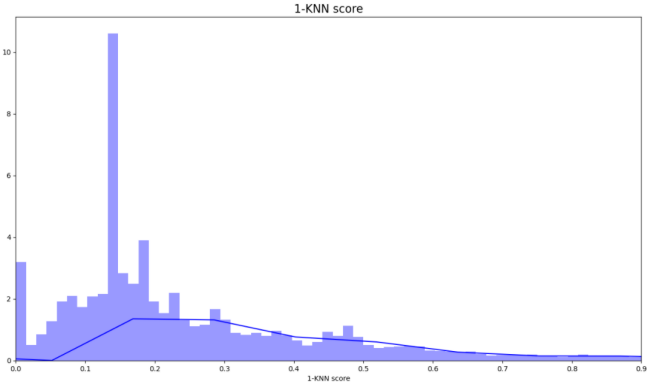


As shown in Fig8, the red line is our tuned threshold, the data point above the read line would be classified as outliers.

B. 1-NN outlier detection

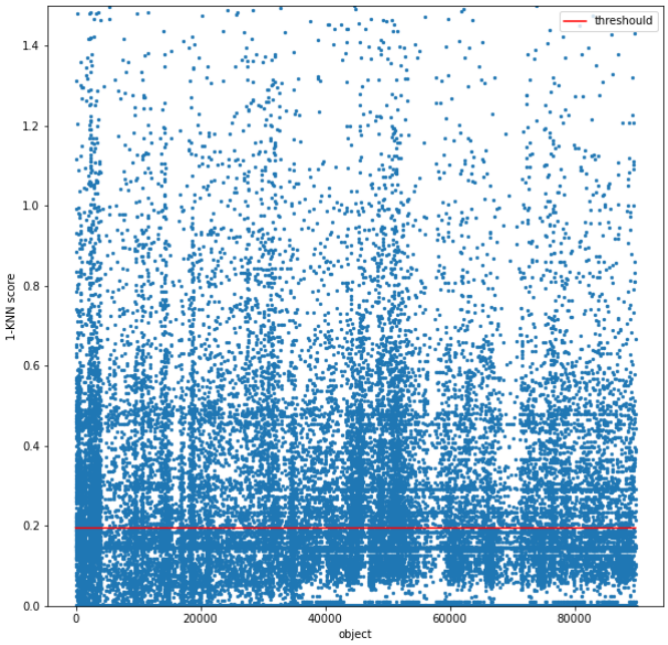
In this section, we build 1-neighbor KNN outlier detection algorithm. 1-NN is sensitive to outliers, since a single mislabeled example dramatically changes the class boundaries. Anomalies affect the method significantly, because k-NN gets all the information from the input, rather than from an algorithm that tries to generalize data. Parameter configuration is $(contamination = 0.172, n\_neighbors = 1, n\_jobs = -1)$.

Fig. 9. 1-NN score distribution



In this case, 1-NN score = 0.1945 is calculated by the model as threshold in this case.

Fig. 10. outlier visualization by 1NN



As shown in Fig10, the red line is the threshold calculated by 1-NN model, all the points whose score higher than the threshold would be classified as outlier.

TABLE VI
PERFORMANCE COMPARISON

Name	Acc on Sample data	similarity score with RF
AE+LSTM	62.3%	62.3%
1-NN	66.2%	39.6%

In Table VI, 'Acc on Sample data' represents the model's accuracy on testing sample data, 'similarity score with RF' represents the similarity between test model and RF on unlabeled dataset.

VI. Conclusion

Through DT and ET, we get the classification conditions for determining whether the Covid-related domain is malicious. The rationality of the domain name structure and the validity of the online archive are the main classification conditions. After that, we get the tree structure model with the best cross validation score by tuning parameters. Through the tree structure model and the cross validate with AE-LSTM and 1-NN, we can conclude that AE+LSTM is more similar to the tree structure algorithm. It provides a more reliable performance. Although 1-NN has a higher similarity on unlabeled data in the end, the accuracy on the sample data doesn‘t reach the half point. The original plan of this experiment was to use the DBSCAN algorithm for further testing, but due to its huge computational complexity and high requirements for RAM, we did not get an effective result of a cluster algorithm.

References

[1] Google. *Sharing data that sheds light on how the policies and actions of governments and corporations affect privacy, security, and access to information.* URL: https://transparencyreport.google.com/?hl=en. (accessed: 10.14.20120).

[2] Donghoon Shin. *A VIEW INTO TOP LEVEL DOMAIN (TLD) ABUSE.* URL: https://blogs.akamai.com/2019/10/a-view-into-top-level-domain-tld-abuse.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+TheAkamaiBlog+%28The+Akamai+Blog%29. (accessed: 10.25.2019).