Project name: Using machine learning to detect malicious URLs

Student: Yunjie Wang

Supervisor: Tzipora Halevi

**Project Goal:**

This project aims to use machine learning method to detect malicious URLs.

The project is following these steps: preparing data, creating training/testing sets, instantiating the classifier, training the classifier, making predictions, evaluating performance, improving performance.

In order to choose the best classifier, I implement different classifiers and then compare the performance. For better performance, I also use Primary Component Analysis and feature selection.

After my exploration, the accuracy of my model can reach more than 93% correction.

**Part I Preparing data**

Dataset resource is ISCX-URL2016 from [www.unb.ca](http://www.unb.ca).

There are five different types of URLs: over 35000 benign URLs, around 12000 spam URLs, around 10000 phishing URLs, more than 11500 malware URLs, more than 45450 defacement URLs. Defacement URLs are trusted websites containing malicious webpages. This dataset has enough data and is worthwhile to explore.

**Part II Creating training/testing sets**

I choose all.csv. This CSV has 79 features and 5 types of URLs.

I split the training and testing data with the ratio 8:2.

**Part III Instantiating the classifier**

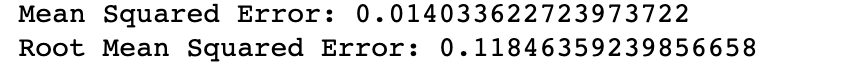
**In this part, I try to understand how classifiers work.**

I use K-nearest Neighbors and Random forest classifier as the original paper suggests.

I learn to use Random Forest as classifier on my one of CVSs. I import random forest regressor from scikit-learn package as the method to explore the raw data. I can easily find out the mean absolute error, mean squared error, root mean squared error.

Example:





I also find out the accuracy using random forest classifier.

Example:

A picture containing text, receipt, screenshot

Description automatically generated

**Part IV Training the classifier**

**In this part, I try to use Random forest as primary classifier.**

**Random forest with multiclass and one-class**

One-class means all malicious URLs will be together as 1, and benign URLs will be 0. There are only two categories. I use one of CSVs as example, and result is shown below:

Table

Description automatically generated

Multiclass means each kind of URLs as marks uniquely.

0: benign 1:defacement 2: phishing 3: malware 4:spam

The result is shown below:

Table

Description automatically generated

Clearly, one-class gets higher result.

**Part V Making prediction**

Using accuracy\_score as tool to determine the performance. Classification Accuracy is the simplest out of all the methods of evaluating the accuracy, and the most commonly used.

**Part VI Evaluating performance**

**In this part, I try to find the best classifier by comparing the performance.**

I also use KNN classifier to produce the result on the same dataset all.csv. multiclass.

Here is the result:

A picture containing table

Description automatically generated

And I compare KNN and random forest, and I find out that random forest classifier produces high result.

Chart, bar chart

Description automatically generated

5. comparison with the original paper:

Graphical user interface, table

Description automatically generated

On the paper, using Random forest, the average is as high as 0.97 compared to 0.935.

**Since the performance is not as high as desired, I think of other ways to improve it except choosing different classifier.**

**Part VII Improving performance**

**In this part, I try to solve this question: can PCA help reduce number of features while maintaining good classification results?**

**VII - I. Primary Component Analysis (PCA)**

In my second part of my project, I start to learn Primary Component Analysis (PCA) to find out more feature information. PCA can be used to visualization and speed up the algorithm.

In PCA\_1.ipynb, I use All.csv as my data. I Import PCA package and choose 40 components as an example. Using the PCA algorithm, I choose the first two features as principal component 1 and principal component 2. The reason I choose the first two features is that PCA algorithm sorts out all features due to their importance. Using two features, it is easy to plot the whole data in the two-dimensional graph via matplotlib.Chart, scatter chart

Description automatically generated

In PCA\_2. Ipynb, I use PCA to reduce feature from 79 to 40 as an example on the same dataset. I use Random Forest as my classifier, and import accuracy\_score on sklearn.metrics package to calculate the accuracy. As a result, I can easily compare the difference between the accuracy without PCA and the other with PCA. The conclusion is obvious that the accuracy without PCA is higher than the other. Maybe PCA results in this consequence, but I am not sure, so I continue to compare the result using different amounts of features.

Chart, bar chart

Description automatically generated

In comparePCA1.ipynb, I compare them with different components. I use variance from 0.1 to 0.9 and n\_components from 10 to 79. On the graph, it is quite clear to see that generally with more features, the accuracy is high. But also I find out even with PCA (79) all features, the accuracy is lower. I am not sure what happens to this, so I will keep looking for reasons.

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

In comparePCA2.ipynb, I import PCA with various components to plot the data graph. I find out these graphs are identical because of the PCA algorithm. The algorithm will calculate out the most important feature on first column and so on. Therefore, no matter how many components are, the first two features are the same that generate the same graphs.

**Since the PCA doesn’t help improving the performance, I choose to use feature selection as the next step.**

**VII -II feature correlation**

I use feature importance of random forest to find out the order of importance features. I deleted five least important features, and then I delete another five features. Here is the result.

Chart, bar chart

Description automatically generated

Conclusion: with deleting 5 least related features, the score gets higher. with deleting 10 least related features, the score gets lower. Therefore precise feature selection does help improve the performance.

**Whole project experience:**

**From this project, I build a complex machine learning model from scratch. I learn the basics of machine learning structure. Machine learning pipeline of seven steps as shown above is very helpful. More importantly, I know how to improve the performance by using PCA, feature correlation. Of course, there are many approaches out there worthwhile learning. Moreover, I learn some visualization techniques that are so easy to present my model. They also do help me to understand what I did better.**

Related resource:

https://github.com/yunjiewong/maliciousURLproject