Final Report

by Ashish KUMAR JHA

Submission date: 16-May-2021 11:47PM (UTC+0400)
Submission ID: 1585876305
File name: Final_Report_2018A7PS0173U.pdf (1.05M)
Word count: 3803
Character count: 17987

DESIGN PROJECT

PHISHING WEBSITE DETECTION USING MACHINE LEARNING

Ashish Kumar Jha
BITS PILANI DUBAI CAMPUS | 2018A7PS0173U

Index

1. ACKNOWLEDGEMENT

I would like to thank our honorable Director Dr. RN Saha, for giving me the opportunity to work on this design project, under the field of Machine Learning. I would like to express my deepest appreciation to all those who provided me with the possibility to complete this report.

A special gratitude and note of appreciation to my Professor, mentor and guide, Dr. Raja Muthalagu, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project and understand all things better, especially in writing this report.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of our seniors, who gave me an inspiration to think better in terms of content quality.

2. ABSTRACT

In this growing age of science and technology, where almost all the information that humanity has is available on the internet, Cyber Security becomes a major concern. When all our personal, private, and organizational data is present on the World Wide Web, our privacy is at risk. Various organizations spend huge amount of money every year to safeguard themselves and their information and data from the cybersecurity attacks by hackers. Phishing is one of the most common attacks that targets the online user. The unique nature of this attack lies in the fact that instead of taking advantage of any sort of software vulnerabilities, like most cybersecurity attacks do, phishing is targeted towards human vulnerabilities. Humans are tricked into entering pages that appear to be legitimate, while they are not. These pages then collect sensitive information such as the Email, username, password or even bank account related information. In this project, the primary aim will be to identify phishing websites with the use of machine learning and then build a browser extension which can be added to any browser and then used to detect phishing websites in the real time.

3. INTRODUCTION

With the advancement in the science and technology, there is hardly any field that has remained untouched from the growth. One of the most rapidly growing fields has been, the internet. And the internet is now used as a hub of all the information and data in our world. Because of this increasing trust on the internet, it has led to a lot of security thieves to pry their eyes on the data and information of value. According to the statistics collected by various organizations including the Ant- Phishing Working Group and Kaspersky Lab, there has been a rapid growth in the number of phishing attacks in the recent years. With phishing sites being added every day, its becomes a necessity to find an effective technique to counter this and in order to deal with new sites being added every day, I am trying to develop a machine learning algorithm which can automatically identify a malicious website and warn the user in advance.

Machine Learning which can broadly be classified into two main types, viz, Supervised Learning and Unsupervised Learning. As the name suggests, supervised learning involves the algorithm to have the prior information and knowledge about a given specimen with proper label, and then it is provided with training and testing data. In unsupervised learning, labeling is not done before hand and the algorithm must find the best labels on its own.

In my project, I will be taking the approach of Supervised Learning, with the help of various known websites.

4. APPROACHES

There are different approaches form the problem that we defined in the previous section. On an upper scale, we can classify the different approaches based on the factor that decides how the website is labelled as malicious or not.

a. Content Based Approach

The first method that we will discuss in this phase is the "Content Bases Approach". As the name clearly suggest, in this method, we classify a website as a phishing, or a non-phishing website based on the contents on the given site. This approach works on the principle of "Term Frequency/ Inverse Document Frequency" or more commonly known as the TF-IDF algorithm for finding out phishing websites based on the page content. According to various researches done in the related field it can be said with fair confidence that this technique gives high accuracy in finding out phishing websites (around 97%). We can use the help of heuristics to reduce the number of false positives even more.

The advanced version of this technique was used and implemented by "Rao and Ali" and they managed to reduce the false positive to 0.035% with a high accuracy of 96.57%.

b. URL Approach

Another approach is to detect phishing websites using the information gathered from a site URL. URL or Uniform Resource Locator is the information present on the address bar of a website. We collect the various components and then compute the metric for all the components in question. Using "Apriori algorithm", we can pick up known information from a given data set. This can help us distinguish several features in a website that can point us towards the fact that whether the website is a phishing website or not.

The accuracy for this approach is high, with around 97% of websites being detected using this technique.

c. Machine Learning Approach

This approach can be explained more effectively as the combination of the above two approaches along with data training and automation. All the previous features, including the website content

and the features extracted from the site URL, along with the heuristics, are fed into a machine learning algorithm that is then trained to identify whether a given website is legitimate or not.

Initial work in this field showed around 92% true positive results and only 0.4% false positives. However, with advancement in the algorithms and better data collection and training, this number has been improved further.

5. EXPERIMENT

Our experiment is carried out in steps. This helps not only in breaking down the process, but also makes it easier to finish each step without any fault. The steps needed for the experiment to be carried out are as follows, Collection of Dataset, followed by the extraction of various features and then finishing it off by our methodology.

a. Collection of Dataset

To train our model for the machine to identify between a phishing website and a non-malicious website, we need to have a strong dataset. For better training, we will need the dataset for both types of sites.

I am going to use the Dataset from those present at “PhishTank”. PhishTank has a collection of various URLs, both legitimate and malicious, which can be used for out machine learning algorithm. The number of URLs will be based on the accuracy achieved after doing the first few runs of the algorithm; in case the system needs more data to be trained. While training the data, we need to make sure that the system is not being over fitted by excessive training.

b. Features Extraction

As discussed before, the features can be extracted in various ways, namely, the URL, page content and the Alexa page rank. Different characteristics can be collected form these three properties. Figure 1, the Features table, shows the various properties that can be extracted from the properties of URL, page content and Alexa page rank.

Features Based On		
URL	Length of URL Length of the path of URL Number of dot (.) in hostname Number of hyphen (-) in hostname Number of at (@) in the URL Number of underscore ( ) in hostname Number of certain key word in URL Transport layer security Presence of www Unicode in URL	Length of hostname of URL Number of dot (.) in the path Number of slashes (/) in URL Number of special char acters (: ; % & ? +) Number of digit in host name Number of underscore ( ) in path Number of hexadecimal with % IP address Port redirect Hexadecimal characters
Page Content	Number of forms Number of forms with action 'POST' Number of outer src script Number of < Applet > Number of < F rame > Number of non-link Number of input email Number of button	Number of forms with action 'GET' Number of script Number of < Iframe > Number of < Embed > Number of link Number of submit Number of input pass word
Rank	Alexa rank	Age of domain

FIGURE 1: FEATURES TABLE

6. Methodology

As we can say above, in Figure 1, the number of features is huge, and if we were to take each feature into account while training our model, we might not be able to obtain the best results. Also, all these futures may or may not affect the authenticity of a page. Some of the features may be of more importance than the others and it will be best for us to consider the features that are more important than considering all features. So, we take all these 36 features into account and then try to figure out the best feature group that can help us identify the authenticity of a given website in the most effective way.

If we were to view all possibilities in which we can group the features, 2 or more at a time, we would get a value given by the expression:

$$\sum_{f=1}^{36} = \frac{a!}{f!\,(a-f)!}$$

Here, 'f' represents the number of features that we took into consideration at a given point of time. It starts from 1 and goes till 36, taking every feature in a unique way and then together as a group. 'a' represents the number of features, which is 36 in our case.

We can therefore summaries the feature selection process, for best results and optimizations.

7. Machine Learning Implementation

Our problem here can be basically mentioned as a classification style problem, or something like a clustering problem. Just as we classify unknown elements into given groups in a classification problem, similarly, in our problem we must classify a given site as malicious or clean. For the process of classification, we have a lot of various algorithms, namely, Random Forest, Artificial Neural Networks, Support Vector Machines etc.

I conducted a sample test with a different classification problem, with similar style as ours, to figure out which one of these algorithms can give us the best results.

Our sample experiment consisted of a data with the following information:

5 sensors that send numerical data

Sensor data rounded off to integer for ease of classification

Activities 1,2 or 3

The aim of the experiment was to classify the Activity performed by the robot based on the different sensor readings. The results obtained using different algorithms has been created as models.

a. Model 1: Artificial Neural Networks

More that 225,000 parameters were trained for 100 epochs and the results obtained are, 67% on the training data and 70% on validation.

b. Model 2: CatBoost

The accuracy increased significantly to 84.6% during the use of CatBoost algorithm.

c. Model 3: Lightgbm (Light Gradient Boosted Machine)

The training accuracy was high at 89.42%, however the algorithm didn't perform equally while testing the same on the data. The testing data accuracy was only 77.56%, then the algorithm was trained using the entire data set. This also gave a training accuracy of 88.32% only.

d. Model 4: XGBoost (Depth Processing)

This is an overfitting model with infinite parameters. The results were not very encouraging, with training accuracy at 73.83% and testing accuracy at 71.20%, then I tried applying optimizations to improve the result, and the results were better than any other test so far. Training accuracy was at 93.91% and testing accuracy at 78.87%. The max depth of the tree has been updated and the model optimized to its best. The final training accuracy was at 93.96% while the testing data accuracy was at 82.14%.

e. Model 5: Random Forest

When the model was trained using the random Forest technique, the results were like those obtained using optimized XGBoost. Accuracy of the model over the training dataset was 92.27% and the testing accuracy for the same model was at 80.23%.

f. Model 6: Decision Tree

Again, the results were like the previous 2 cases, due to obvious reason being the similarity in the way the algorithm works on tree and depth-based analysis. The accuracy for training data was 91% and the testing data was at 79%.

g. Other Methods

Various other models were used too, but the results were not up to the mark set by the algorithms already mentioned. To have a comparison, I have put them in a tabular view.

Model	Training Accuracy	Testing Accuracy
Logistic Regression	56%	56%
Support Vector Machine	58%	58%
K – Nearest Neighbors	82%	77%
Linear SVC	56%	56%

After applying all big classification algorithms on our data, I did a full dataset training using the XGBoost algorithm because of its superior performance during the previous testing. The results are encouraging.

The final training data accuracy is at 94%, which is the highest so far.

8. Results and Discussion

	precision	recall	f1-score	support
Bad	0.9	0.97	0.93	36597
Good	0.99	0.96	0.97	100740
accuracy			0.96	137337
macro avg	0.95	0.96	0.95	137337
weighted avg	0.97	0.96	0.96	137337

The above table shows the classification report of our problem, using the technique of "Logistic Regression".

Precision here means the fraction or the ratio that we obtain from dividing the correct predictions to the total predictions made. So, a precision of 0.9 signifies that 90% of the data found was correctly identified by our model.

$$P = \frac{TP}{TP + FP}$$

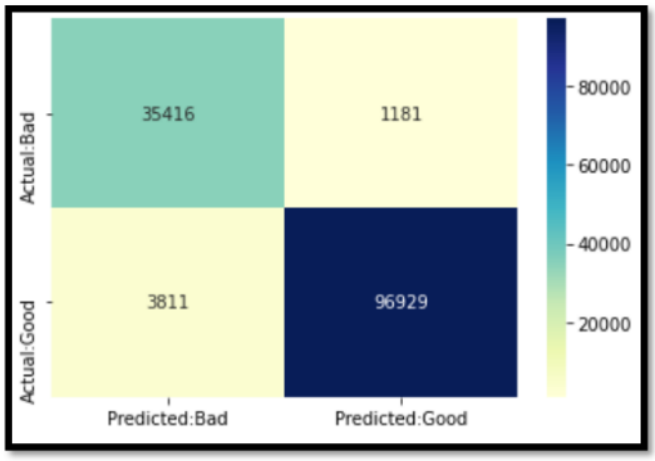
Recall here signifies the fraction or ratio of the correct predictions made divided by the total positive obtained from our model.

$$R = \frac{TP}{TP + FN}$$

F1 scores are calculated using the formula:

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

Where, TP signifies True Positives, FP signifies False Positives and FN signifies False Negatives. The physical relevance of F1 score is the measure of the accuracy of the model trained on a given dataset.



This is a confusion matrix which is obtained from the classification report of our "Logistic Regression" model. The four blocks on the chart signify True Positive, True Negative, False Positive and False Negative.

True positive here means that, the site was malicious, and our model identified it as malicious.

True negative means that the site was malicious, however, our model identified it as a safe site.

False positive means that the site was a safe site, however, our model identified it as malicious.

False Negative means that the site was safe, and our model predicted it correctly to be a good site.

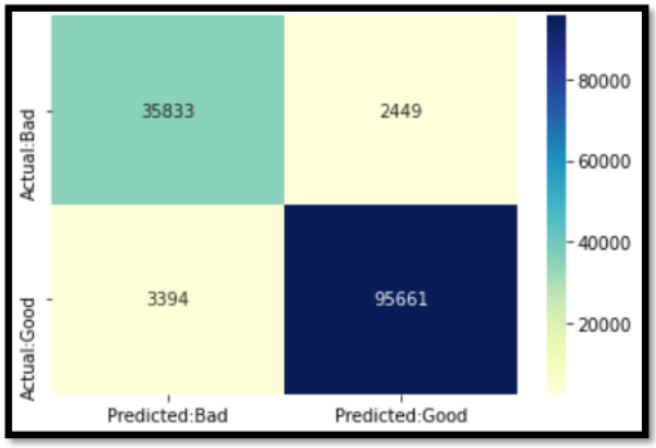
So, for our models, high number of true positives and False Negatives signify that the model worked correctly for most of the data values.

These values are for the model trained under "Logistic Regression".

I then proceeded to use MultinomialNB for the same data set in order to obtain competitive results.

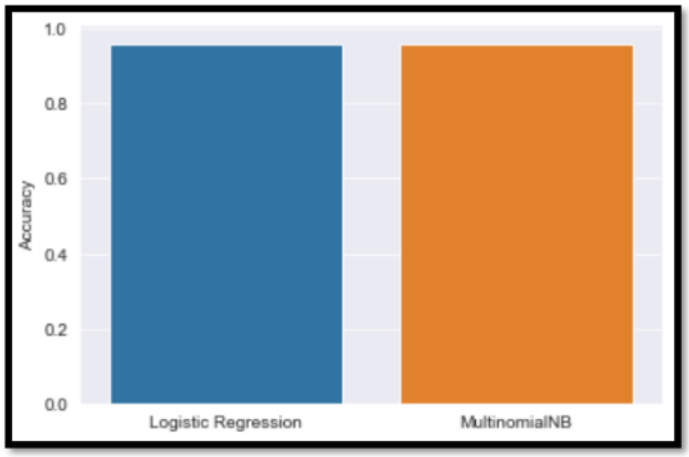
	precision	recall	f1-score	support
Bad	0.91	0.94	0.92	38282
Good	0.95	0.97	0.97	99055
accuracy			0.96	137337
macro avg	0.94	0.95	0.95	137337
weighted avg	0.96	0.96	0.96	137337

This was the classification report for the model trained under MultinomialNB model.



The confusion matrix for the same is given here. This model gave an average accuracy of around 95% during the analysis of the result.

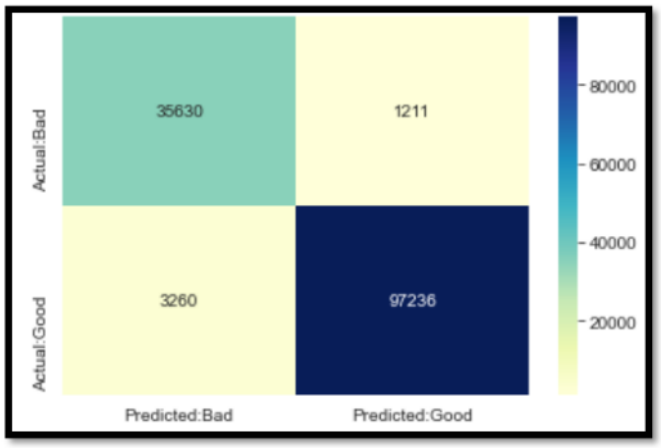
So, when plotted against each other as shown:



We can see that the Logistic Regression outperforms the MultinomialNB by a small margin. So, we can conclude that the Logistic Regression is the best fit model for our case. So, next step includes creation of a “sklearn” pipeline for our model using “Logistic Regression”.

	precision	recall	f1-score	support
Bad	0.91	0.94	0.92	36841
Good	0.98	0.97	0.97	100496
accuracy			0.96	137337
macro avg	0.94	0.95	0.95	137337
weighted avg	0.96	0.96	0.96	137337

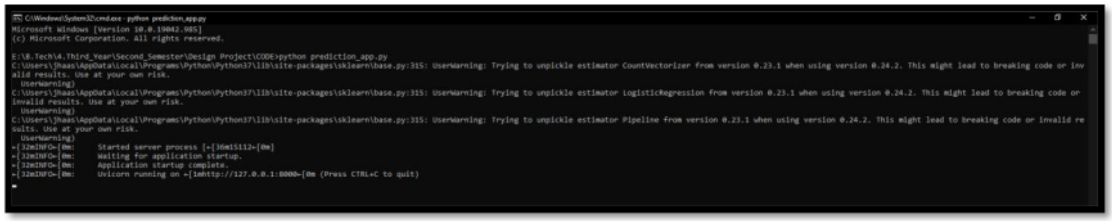
This was the classification model for the model trained according to the pipeline as defined above.



This was the confusion matrix thus obtained. As it is very clear form the matrix, our model was highly accurate in predicting the sites correctly into the groups of good and bad. With this simple, yet effective method, we increased our accuracy to around 98%.

9. Application of the Code

As we discussed in the previous section, we use the method of "Logistic Regression" to build our prediction app. The app runs on the local host server and then identifies website as safe or malicious based on the features that we trained it on.



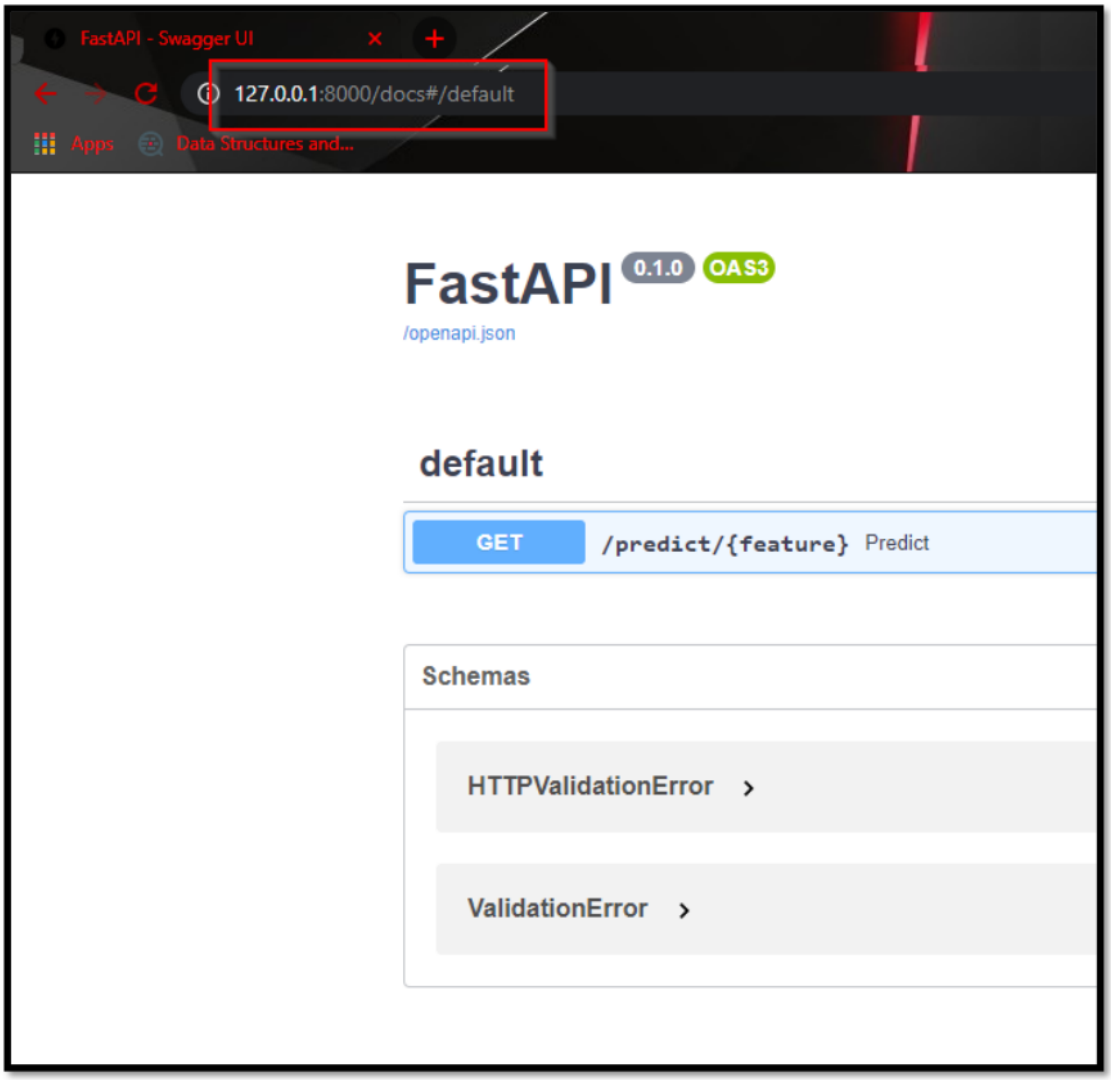
This is a snippet of the code running at command prompt, once the code is up and running, we can now use the server to get our web app working as well. As I have chose the local host as our host system, therefore providing the IP as "127.0.0.1" and then giving the port "8000" for the app to run on.

prediction_app.py

E: > B.Tech > 4.Third_Year > Second_Semester > Design Project > CODE > prediction_app.py

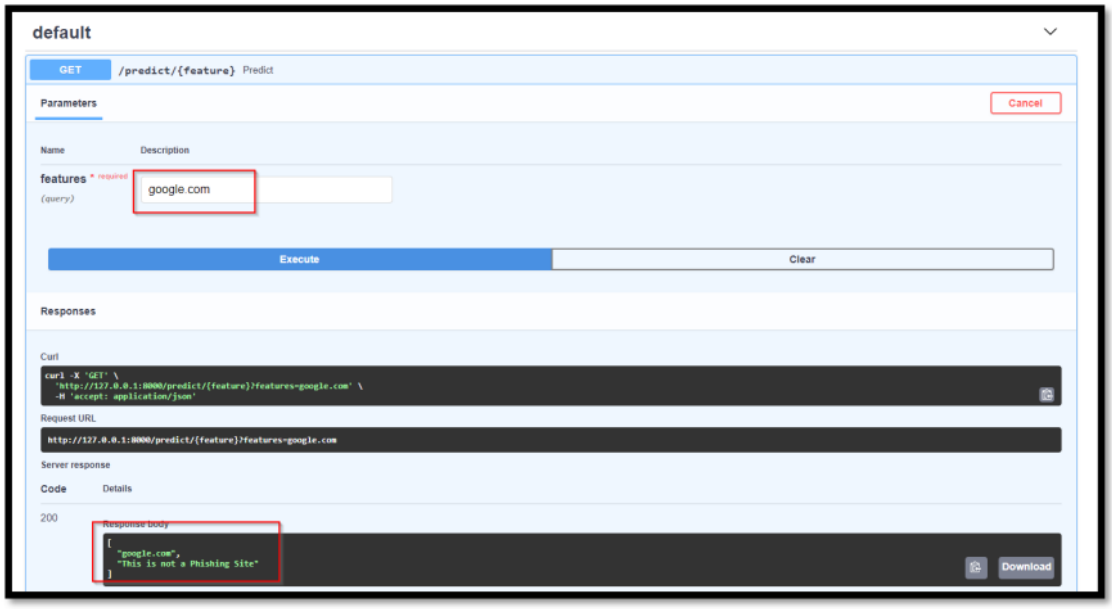
```python
import uvicorn
from fastapi import FastAPI
import joblib,os

app = FastAPI()

#pkl
phish_model = open('phishing.pkl','rb')
phish_model_ls = joblib.load(phish_model)

# ML Aspect
@app.get('/predict/{feature}')
async def predict(features):
    X_predict = []
    X_predict.append(str(features))
    y_Predict = phish_model_ls.predict(X_predict)
    if y_Predict == 'bad':
        result = "This is a Phishing Site"
    else:
        result = "This is not a Phishing Site"

    return (features, result)
if __name__ == '__main__':
    uvicorn.run(app,host="127.0.0.1",port=8000)
```

We can now go to a web browser and run the web app on it. To go to the web app, as designed, the link should be as shown in the figure.

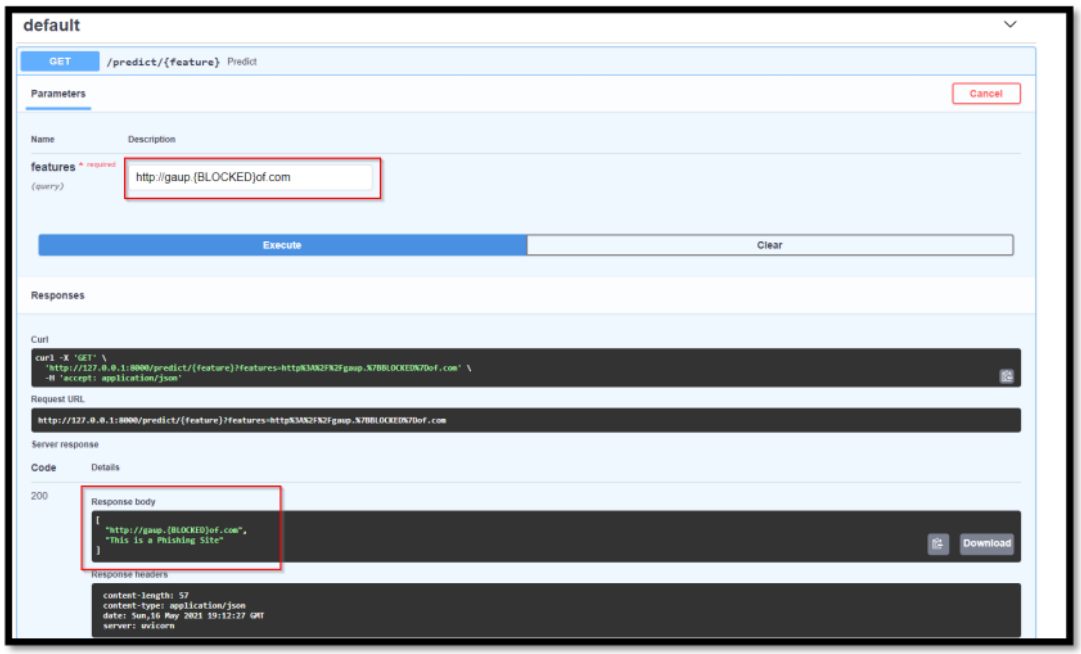


This is how our web app will look like. This is still the beta version of the actual product and needs more training and improved feature extraction technique. Also, the scope of improvement comes in the outlook of the final product and it can be made more user friendly in the next phase of development.

Now, to show how the app actually runs, we will try it with a normal URL, for example taken here as "Google.com". And to show the positive work of the app, we will try using a malicious link as well.



As the input was a valid URL, the app did not flag it as a malicious website. Now, in order to try and see a malicious website, we used “http://gaup.{BLOCKED}of.com” which is a blocked site as it is a malicious software which has been identified by various sources so far.



As we can see, the app flags the site URL as a phishing website.

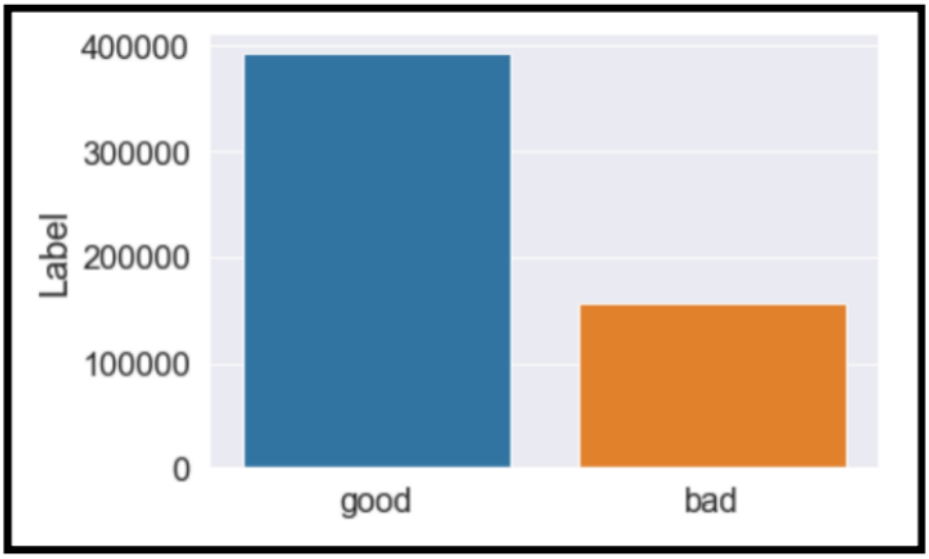
10. Scope Of improvement and Development

As discussed in the previous section of the report, when we tried running the application mode of the code, there are a lot of areas of improvement. The app can be further made into a website extension for the ease of use which can be directly downloaded from the chrome store and used on the device and it will flag the URLs automatically to save time and give the best results.

Also, the data was trained on my personal computer, which has high specifications as it is a gaming laptop. However, as we will see in the next section, the size of data that we worked on was pretty huge. To train a model effectively we need better specifications and faster computers. With the help if these we can train the data not only in lesser time than on my setup, but also, we can obtain better results by training more and more data using parallel computing.

11. Experimental Setup

The results for the experiments have been acquired from "PhishTank". The data consists of 5,49,346 unique entries in a "csv" file format. The data is further divided into two columns, which are namely, the URL and the Label associated with it. The URL contains website URL's, both malicious and non-malicious ones. The label has two values, Good and Bad, signifying the website is a safe site or a malicious site respectively. The data is further divided by the labels as shown below.



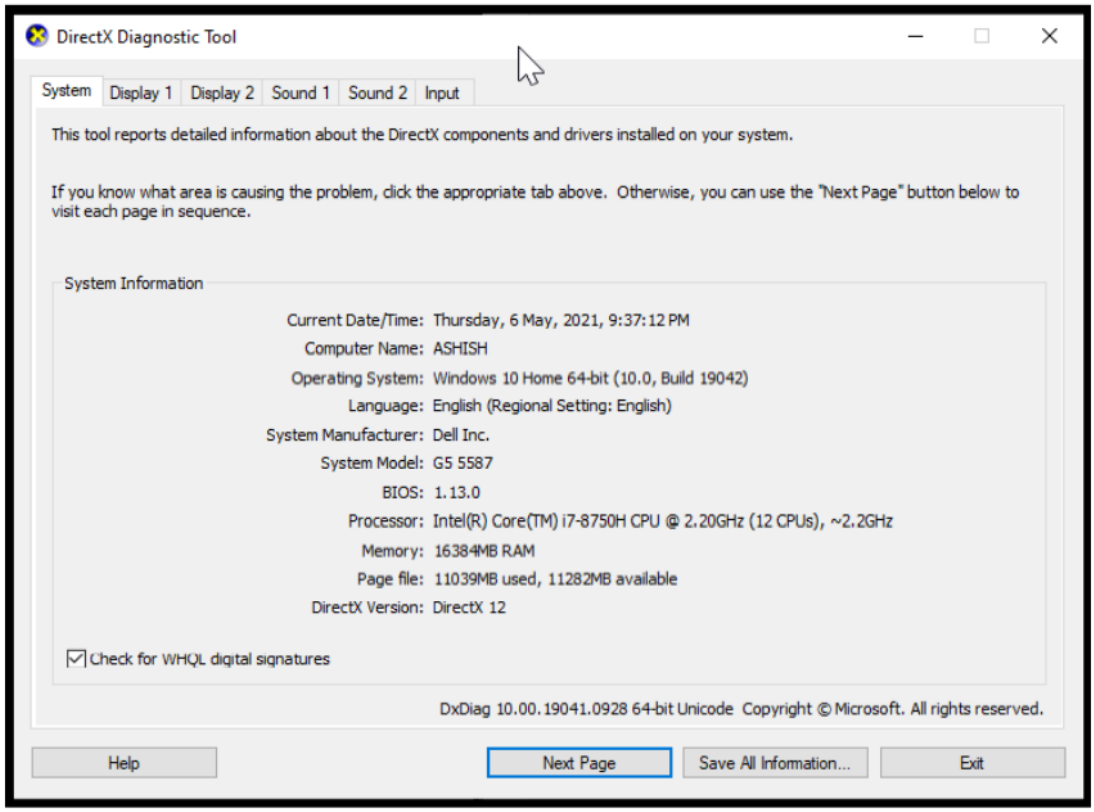
The image above shows the percentage of data set labelled as good and bad. Now that we have the data, we must vectorize it and then tokenize it to find out the more important words, example, 'virus','.exe','.dat' etc. After that we convert the URLs into vector form.

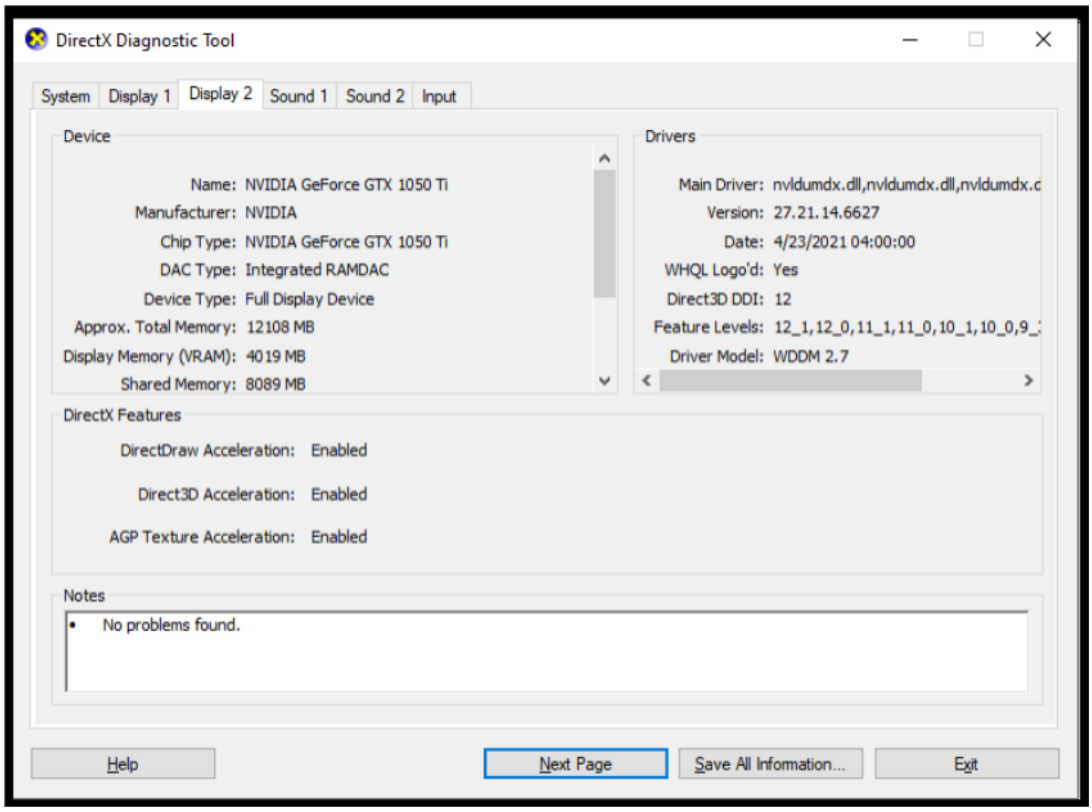
12. Setup Used

I ran all the above tests on my system itself. All the training and testing of the entire data set was performed on my personal computer. The specifications for which are:

- Processor: Intel Core i7 – 8750H
- RAM: 16GB
- Dedicated Graphics: 4 GB of Nvidia GeForce GTX 1050Ti

The remaining specs can be seen from the screen shots provided here with.

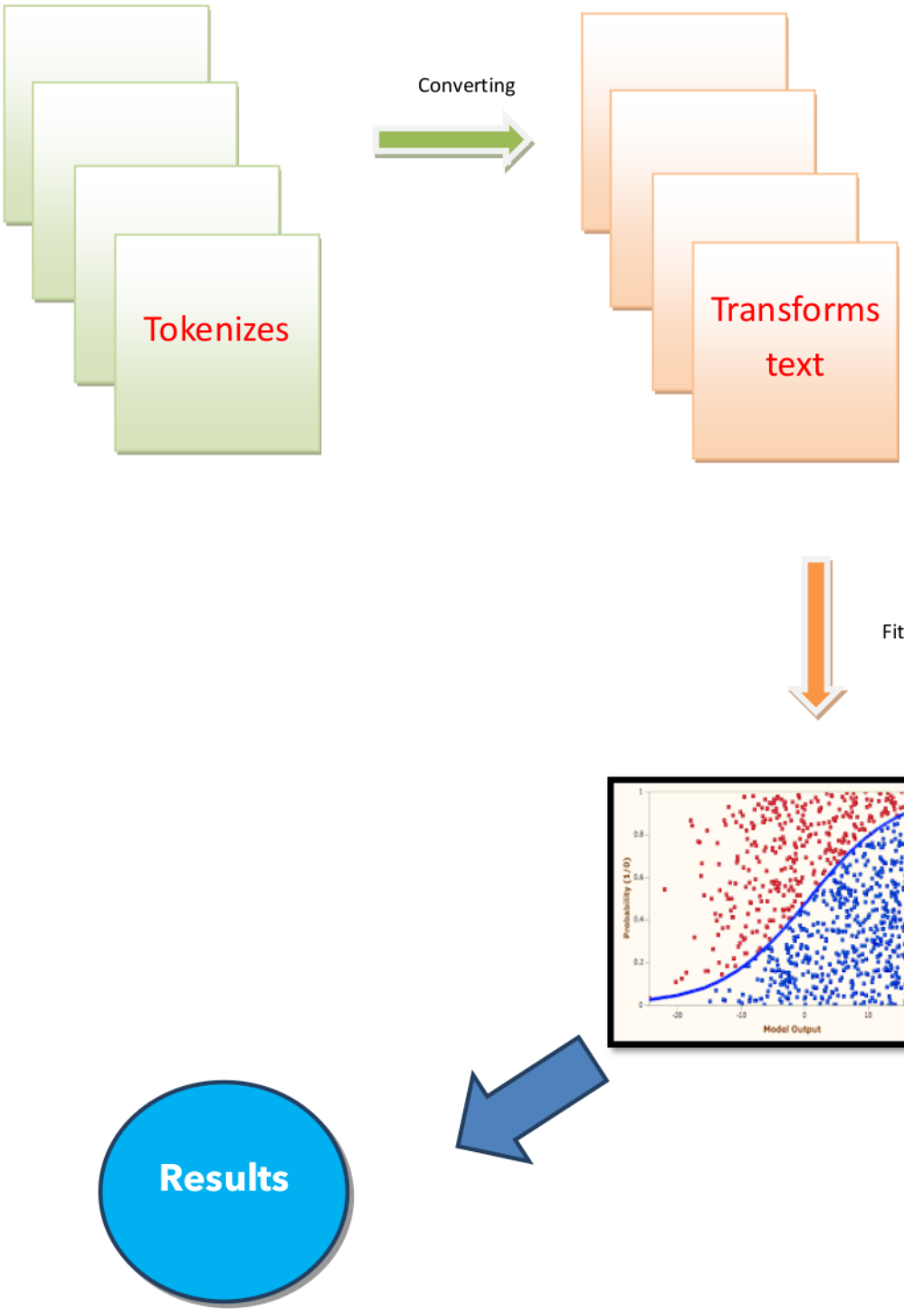




13. Conclusion

Our model works in the following way:

First, the website features are pulled and tokenized. Then these tokenized features are converted into Transformed text, which is fit as the data and then results are obtained. Which is shown below pictorially.


Converting
Tokenizes
Transforms text
Fit Data
Probability (1/0)
Model Output
0's
1's
Results

So, these steps are carried out every time we enter a URL into our system.

I have worked by taking several features that was extracted from the URL provided. This is then used to create a feature list that contains all the features that I used for training the set of data. I could not train the model on all the features, that is so 36 in total. Our model, however, achieved very high accuracy of more than 98%. This points us to think towards the direction that maybe not all the data that we start with is necessary for the purpose of computation.

Combination of features	Maximum accuracy	Minimum accuracy
1	0.94281	0.564542
2	0.958333	0.556373
3	0.966503	0.544935
4	0.972222	0.549837
5	0.979575	0.539216
6	0.984477	0.547386
7	0.982843	0.553104
29	0.988562	0.913399
30	0.987745	0.908497
31	0.984477	0.914216
32	0.986111	0.916667
33	0.982026	0.933824
34	0.981209	0.939542
35	0.974673	0.940359
36	0.959967	0.959967

Data shown in the above table has been collected by the works of various researchers in the mentioned field and points out towards the possibility of a list of features that does not affect our work as much as the other features do. So, even using only 7 features out of a possible 36, the accuracy can be achieved to be around 99% which remains almost same for even 31 features.

This table makes our belief stronger in the fact that not all the extractable features on a site are deciding factors and effective in deciding the true nature of the site and URL. We can use the more prominent features instead of all the features to boost up our speed as well.

~THANK YOU~

14. References

[1] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A Content-based Approach to Detecting Phishing Web Sites," New York, NY, USA, 2007, pp. 639-648.

[2] L. A. T. Nguyen, B. L. To, H. K. Nguyen, and M. H. Nguyen, "Detecting phishing web sites: A heuristic URL-based approach," in 2013 Inter-national Conference on Advanced Technologies for Communications (ATC 2013), 2013, pp. 597-602.

[3] N. Sanglerdsinlapachai and A. Rungsawang, "Web Phishing Detection Using Classifier Ensemble," New York, NY, USA, 2010, pp. 210-215.

[4] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites," ACM Trans. Inf. Syst. Secur., vol. 14, no. 2, pp. 21:1-21:28, Sep. 2011.

[5] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural networks," Neural Comput & Applic, vol. 25, no. 2, pp. 443-458, Aug. 2014.

[6] Pradeepthi K V and Kannan A, "Performance study of classification techniques for phishing URL detection," in 2014 Sixth International Conference on Advanced Computing (ICoAC), 2014, pp. 135-139.

[7] S. Marchal, J. Franois, R. State, and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," IEEE Transactions on Network and Service Management, vol. 11, no. 4, pp. 458-471, Dec. 2014.

[8] A. Sirageldin, B. B. Baharudin, and L. T. Jung, "Malicious Web Page Detection: A Machine Learning Approach," in Advances in Computer Science and its Applications, Springer, Berlin, Heidelberg, 2014, pp.

217-224.

[9] R. Verma and K. Dyer, "On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifiers," New York, NY, USA, 2015, pp. 111-122.

[10] H. H. Nguyen and D. T. Nguyen, "Machine Learning Based Phishing Web Sites Detection," in AETA 2015: Recent Advances in Electrical Engineering and Related Sciences, V. H. Duy, T. T. Dao, I. Zelinka, H.-

S. Choi, and M. Chadli, Eds. Cham: Springer International Publishing, 2016, pp. 123-131.

[11] M. A. U. H. Tahir, S. Asghar, A. Zafar, and S. Gillani, "A Hybrid Model to Detect 76 Phishing-Sites Using Supervised Learning Algorithms," in 2016 International Conference on Computational Science and Compu-tational Intelligence (CSCI), 2016, pp. 1126-1133.

[12] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting Malicious URLs Using Lexical Analysis," in Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings, J. Chen, V. Piuri, C. Su, and M. Yung, Eds. Cham: Springer International Publishing, 2016, pp. 467- 482.

Final Report

ORIGINALITY REPORT

0%	0%	0%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	On	Exclude matches	< 5%
Exclude bibliography	On		