# RESEARCH ON MOST EFFECTIVE URL VECTORS FOR FINDING MALICIOUS URLs AND PROPOSITION FOR A VOTING METHOD

SIDDHANT TIWARI
Computer Science
Undergraduate
SRM Institute of Science and Technology, Kattankulathur
Chennai, India

HAIDER RIZVI
Computer Science
Undergraduate
SRM Institute of Science and Technology, Kattankulathur
Chennai, India

DR. K. KALAISELVI
Assistant Professor
SRM Institute of Science and Technology, Kattankulathur
Chennai, India

**Abstract**: *This paper consists of our research on machine learning models that would help us detect malicious urls.*

*There are a variety of models available but we have taken CNNs and Basic ML models along with URL vectors and features, because using RNNs or CNN LSTMs is not feasible for 1D data.*

*The main highlights of our thesis have been the accuracy measures of the two main algorithms and comparison in terms of URL features used. Although these differences in accuracy arise and are evident, we propose a lightweight voting system for the most accurate system which does the job.*

*Our research has also led us to find the most important URL vector which we came across while testing different databases.*

## INTRODUCTION

Malicious URLs have been deemed the major factor in online security threats. Attackers can gain backdoor access to sensitive data by just sending one well disguised URL to an innocent person. Malicious URLs are the weapon of choice in Cyber Attacks [1].A survey shows that 75% of ransomware infected companies were using up to date protection systems.[2]

Blacklisting methods are fast but they are again outdated in the era of ever evolving internet.In our previous research project we analyzed various models to detect Malicious URLs and compared their accuracies. Extending the project and its implementation we faced time boundings in the traditional CNN model but we did gain a stellar accuracy with this character embedded CNN model.

Moving on we thought of finding of the URL features that are important or rather more relevant to the malicious or benign prediction, following through on this we proposed creating a voting algorithm that utilized all the other algorithms that we used before and also performed the same procedures by not including some URL features to find if they are more relevant that the others or not.

Also by finding which features are more important than the others users can manually see if the URL is malicious or not and have a pretty good idea of the analysis.

## I. RELATED WORKS

The technological advancements in the 21st century have led to a great need for online safety. And in the post covid era, where almost everyone has most of their life shifted to the virtual ways, whether its online transactions or even their work, a single URL could greatly jeopardize a big company’s resources.

The classification of URLs has utilized a variety of algorithms since the URL phishing was discovered.

Here we describe other methods that have been utilized in the past which we have not used in our project.

Blacklisting is a traditional but outdated method which employs pattern matching techniques and is used by web browsers/plugins.(eg. McAfee Site Advisor freemium add-on).[3] Other methods viz honeypots, web crawling etc. utilize analytics to scan sites but again both of these fail when a URL outside their database is faced.

URL features like lexical, headers and other data were used by Liang Bin [4].

Garera et al.[5] also employed the same lexical features to counter phishing like hostname length, hidden host domains, page ranks and IP addresses, domain tables.

Complex machine learning techniques like SVM, Naive Bayes etc. have been used in tandem with word embedding by Crisan et al. [6] where data processing is simpler and the traditional process of feature selection is innovatively used.

Singhal et al.[7] used ML along with the drift detection concept to compare data between feature vectors of the training dataset and another recently gathered one which would help curb the bypassing of the system.

Even deep learning models like CNN, CNN LSTM, RNN, and simple LSTMs have been drafted for the same purpose. Das et al.[8] gives a comparative analysis of these. On a careful analysis of his work its evident that all these models understand features differently and so fusing models is a good approach.

**1.**

## II. URL FEATURES AND DATA PROCESSING

URL features are a major part of our data processing system. We needed to extract the main features from the URL database, so that our ML models can learn from these.

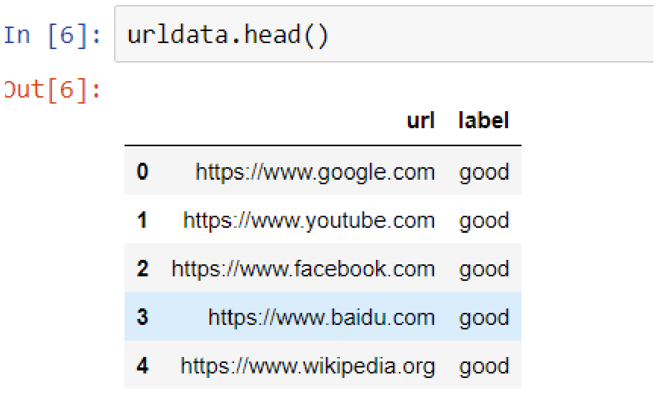
URLs mainly constitute these features :-
● Lexical
● Host Based
● Content Based Features

But after carefully researching these features we knew that taking every feature into account would be redundant. Lexical features were the most important features to be considered for our project.

So for our project we mainly divided our URL features into 3 main parts:-

1. Length features
2. Count features
3. Binary features

We began with a normal dataset which looked as follows.

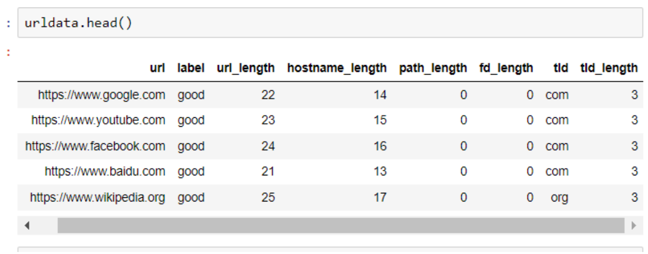


```
In [6]: urldata.head()
```

Out[6]:

| | url | label |
|---|---|---|
| 0 | https://www.google.com | good |
| 1 | https://www.youtube.com | good |
| 2 | https://www.facebook.com | good |
| 3 | https://www.baidu.com | good |
| 4 | https://www.wikipedia.org | good |

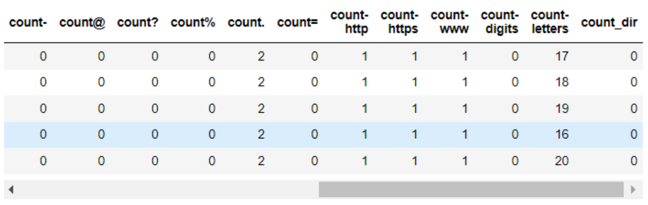
Then after our length feature extraction we got to the following results.



```
: urldata.head()
```

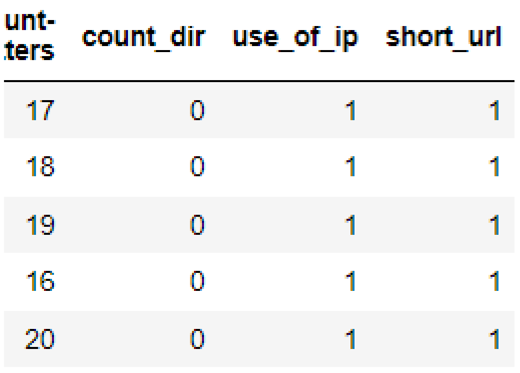
| url | label | url_length | hostname_length | path_length | fd_length | tld | tld_length |
|---|---|---|---|---|---|---|---|
| https://www.google.com | good | 22 | 14 | 0 | 0 | com | 3 |
| https://www.youtube.com | good | 23 | 15 | 0 | 0 | com | 3 |
| https://www.facebook.com | good | 24 | 16 | 0 | 0 | com | 3 |
| https://www.baidu.com | good | 21 | 13 | 0 | 0 | com | 3 |
| https://www.wikipedia.org | good | 25 | 17 | 0 | 0 | org | 3 |

Further we went on to take into account the count features like number of digits, letters,, counts of wwws etc.



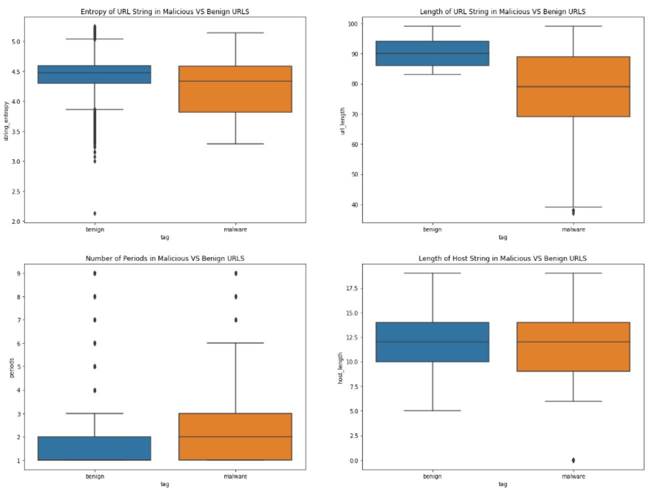
| count- | count@ | count? | count% | count. | count= | count-http | count-https | count-www | count-digits | count-letters | count_dir |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 0 | 17 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 0 | 18 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 0 | 19 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 0 | 16 | 0 |
| 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 0 | 20 | 0 |

Now we were left with binary features like checking shortening services or checking if ip addresses are used in the URL.



| unt- ters | count_dir | use_of_ip | short_url |
|---|---|---|---|
| 17 | 0 | 1 | 1 |
| 18 | 0 | 1 | 1 |
| 19 | 0 | 1 | 1 |
| 16 | 0 | 1 | 1 |
| 20 | 0 | 1 | 1 |

All these features were utilized to feed to both our models specified in the later stages.



Lexical features comparison plots[8]

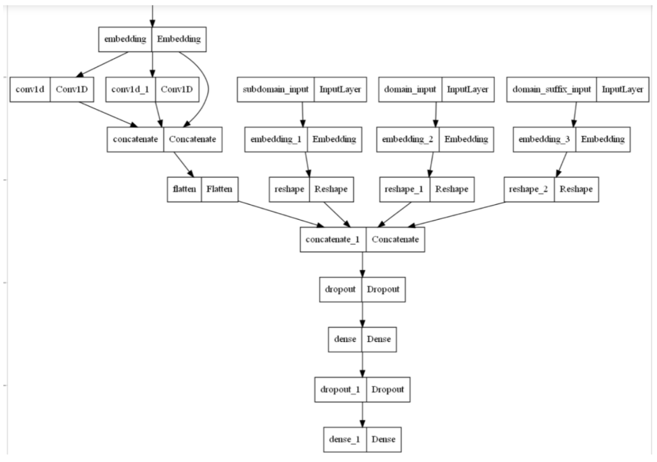
## III. CNNS AND HOW WE UTILIZED THEM

CNNs are deep learning algorithms where the images/data fed to the input layer are assigned importance in terms of biases/weights and then these features are learnt to recognize their importance in the given data.

We have used character level embedding here to use CNNs for text level filtering.

In this paper we have utilized a 1D CNN character embedded model with two layers of convolution with 64 filters each having a kernel size of 5 and 3 each with padding specified as same ("zeros added evenly to left/right and up/down") using the elu activation.

As CNNs tend to overfit textual data as their primary purpose is for images, we have used the early stopping feature from keras, where we monitored the values of val_precision parameter over 5 epochs and bring the model to an early stop if no significant improvements are made over that course of processing.



## IV. LOGISTIC REGRESSION

This type of statistical analysis is often used for predictive analytics and modeling, and extends to applications in machine learning. In Logistic regression the output is finite, usually binary (0, 1) or categorical (Type A, Type B, Type C, Type D).

The biggest use of logistic regression can be said to be in statistical software applications where it needs to be determined what relation exists between the variables.

## V. RANDOM FOREST CLASSIFIER

Random forest is a supervised ML model which basically uses decision trees and then takes as a vote/average for classification/regression.

Here the accuracy is increased due to the combination of multiple classifiers. Also the training time is lesser in random forests model.

## VI.DECISION TREES CLASSIFIER

Decision Trees are the only supervised learning machine algorithm that can be used both for classification and regression.Their structure matches a lot with the classic Data Structure Trees, which consist of internal and leaf nodes and a root node.

Decision tree learning employs a divide and conquer strategy by conducting a greedy search to identify the optimal split points within a tree. This process of splitting is then repeated in a top-down, recursive manner until all, or the majority of records have been classified under specific class labels.

## VII. OUR PROPOSED VOTING SYSTEM

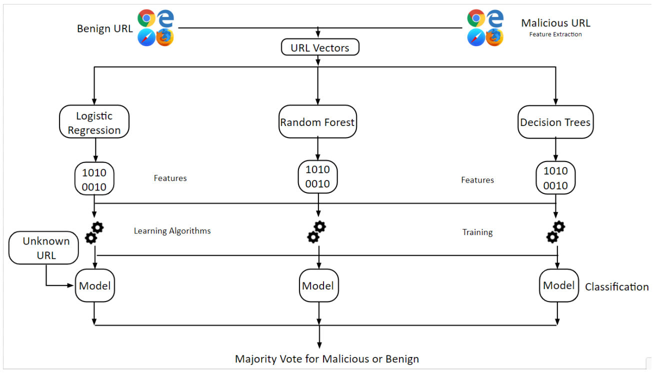
We came across the works of Das et al.[8] and concluded that different learning methods evidently understand features in their own separate ways.

To test this we were curious in involving a voting system method to test accuracies against individual methods.

Here our proposed voting system model combines the 3 given models - Random forest, Decision Trees and Logistic Regression. These 3 were the best fit for a lightweight and efficient model.

We have trained and used the system on large and variable datasets containing around 4.6 lakhs of URLs each and discovered major URL vectors and accuracy findings. Again we tested our system against another dataset of around 4 lakhs values.

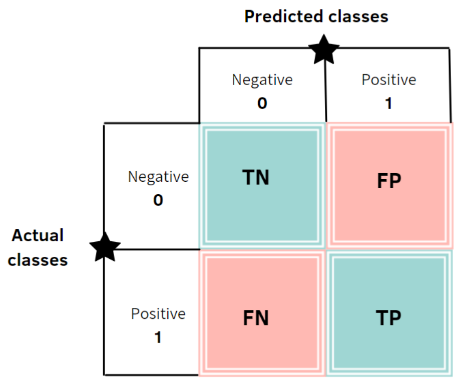
Interestingly it is to be noted that there was a drastic difference in accuracies in the two datasets discussed in section IX. Yet our voting system could indeed improve on individual accuracies and was again faster than individual deep learning methods.



## VIII. EVALUATION METRICS

We have resorted to classics for the evaluation metrics by going with accuracies, f-scores and also last but not the least confusion metrics.

Confusion matrices are shown as below :-



True Positives (TP): All the predictions that were benign and were predicted correctly as being benign.

True Negatives (TN): All the predictions that were malicious and were predicted correctly as being malicious.

False Positives (FP): All the predictions that were malicious and were incorrectly predicted as being benign.

False Negatives (FN): All the predictions that were benign and were incorrectly predicted as being malicious.

Also, precision and recall was taken into account in our project being one of the most important evaluation metrics.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

Precision is the proportion of positive identifications which were truly predicted.

Recall is the proportion of actual positives that were identified correctly.

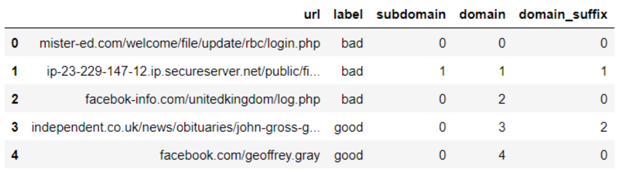
$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{All Samples}}$$

Accuracy is the ratio of correct predictions to total samples.

## IX.ANALYSIS OF ACCURACY AND DRAWBACKS (CNN VS VOTING SYSTEM)

As with the original paper [9] we had started with establishing a method to differentiate and predict different URLs being good or bad, or speaking in a little bit more technical sense to being malicious or benign, but at the end along with achieving our original plan we also created a new algorithm that utilized all the good features of the old three and gave a better result at the end.

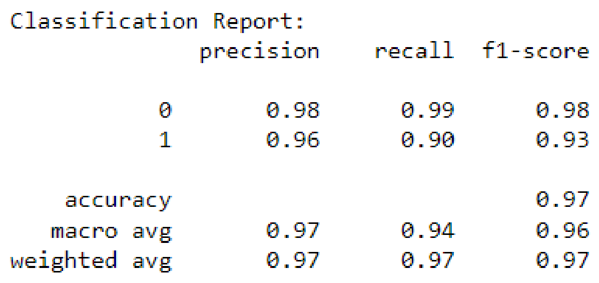
Apart from the aforementioned futuristic approach to the problem we also thought of finding out which URL features are more relevant than the others and we concluded that the https/http feature of the URL are significantly important in a sense that they can impact about 15% of the accuracy of almost all algorithms when not taken into account as opposed to the fact when taken into account for the prediction of the project.



| | url | label | subdomain | domain | domain_suffix |
|---|---|---|---|---|---|
| 0 | mister-ed.com/welcome/file/update/rbc/login.php | bad | 0 | 0 | 0 |
| 1 | ip-23-229-147-12.ip.secureserver.net/public/fi... | bad | 1 | 1 | 1 |
| 2 | facebok-info.com/unitedkingdom/log.php | bad | 0 | 2 | 0 |
| 3 | independent.co.uk/news/obituaries/john-gross-g... | good | 0 | 3 | 2 |
| 4 | facebook.com/geoffrey.gray | good | 0 | 4 | 0 |

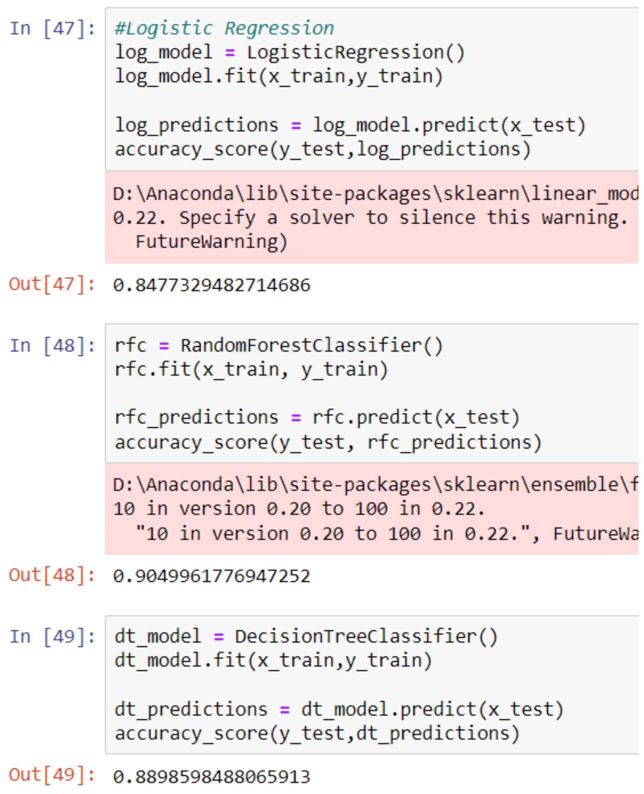
The comparisons made are between the Logistic Regression, Decision Tree and the Rainforest algorithm, which have been later combined to create the voting system at the end of the project as the last step.Two different databases were utilized to get to the results.

The convoluted neural networks had the following output -



```
Classification Report:
              precision    recall  f1-score

           0       0.98      0.99      0.98
           1       0.96      0.90      0.93

    accuracy                           0.97
   macro avg       0.97      0.94      0.96
weighted avg       0.97      0.97      0.97
```

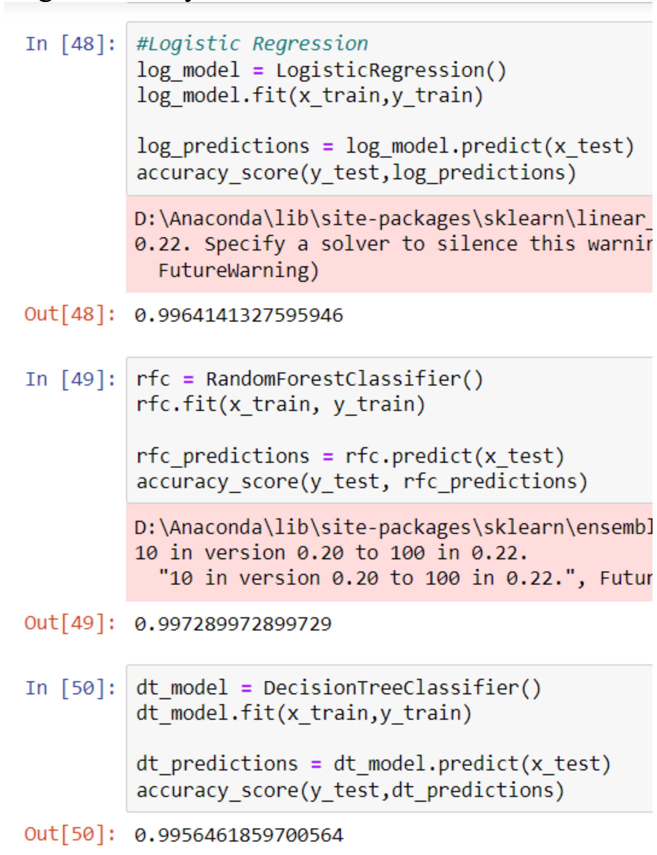
Now, coming to the URL vectors, the output received are as follows -



```
In [47]: #Logistic Regression
         log_model = LogisticRegression()
         log_model.fit(x_train,y_train)

         log_predictions = log_model.predict(x_test)
         accuracy_score(y_test,log_predictions)

         D:\Anaconda\lib\site-packages\sklearn\linear_mod
         0.22. Specify a solver to silence this warning.
           FutureWarning)

Out[47]: 0.8477329482714686

In [48]: rfc = RandomForestClassifier()
         rfc.fit(x_train, y_train)

         rfc_predictions = rfc.predict(x_test)
         accuracy_score(y_test, rfc_predictions)

         D:\Anaconda\lib\site-packages\sklearn\ensemble\f
         10 in version 0.20 to 100 in 0.22.
           "10 in version 0.20 to 100 in 0.22.", FutureWa

Out[48]: 0.9049961776947252

In [49]: dt_model = DecisionTreeClassifier()
         dt_model.fit(x_train,y_train)

         dt_predictions = dt_model.predict(x_test)
         accuracy_score(y_test,dt_predictions)

Out[49]: 0.8898598488065913
```

The accuracy achieved here using the URL vectors is lesser as compared to the CNN model.

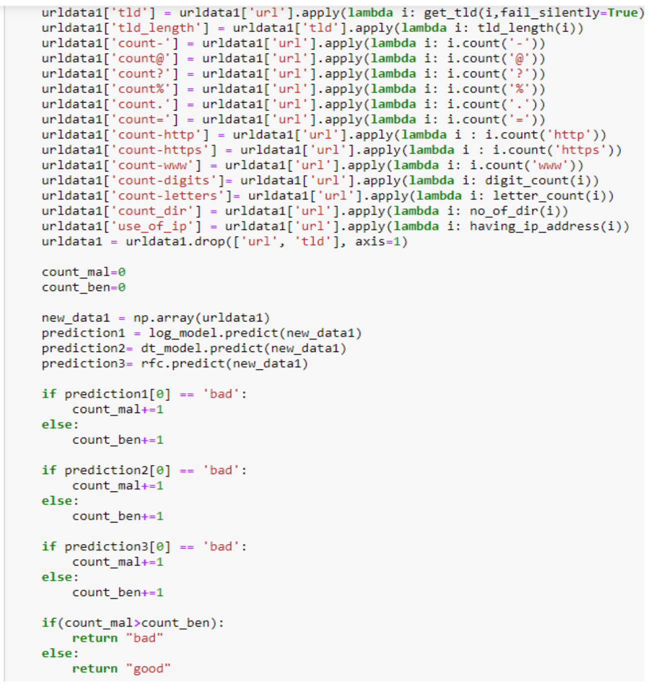
But during our testing we found that, if we take a dataset containing HTTPS and HTTP in the majority of URLs we end up with a significantly larger accuracy for each of these models.



```
In [48]:  #Logistic Regression
           log_model = LogisticRegression()
           log_model.fit(x_train,y_train)

           log_predictions = log_model.predict(x_test)
           accuracy_score(y_test,log_predictions)

           D:\Anaconda\lib\site-packages\sklearn\linear_
           0.22. Specify a solver to silence this warnir
             FutureWarning)

Out[48]:   0.9964141327595946

In [49]:   rfc = RandomForestClassifier()
           rfc.fit(x_train, y_train)

           rfc_predictions = rfc.predict(x_test)
           accuracy_score(y_test, rfc_predictions)

           D:\Anaconda\lib\site-packages\sklearn\ensembl
           10 in version 0.20 to 100 in 0.22.
             "10 in version 0.20 to 100 in 0.22.", Futur

Out[49]:   0.997289972899729

In [50]:   dt_model = DecisionTreeClassifier()
           dt_model.fit(x_train,y_train)

           dt_predictions = dt_model.predict(x_test)
           accuracy_score(y_test,dt_predictions)

Out[50]:   0.9956461859700564
```

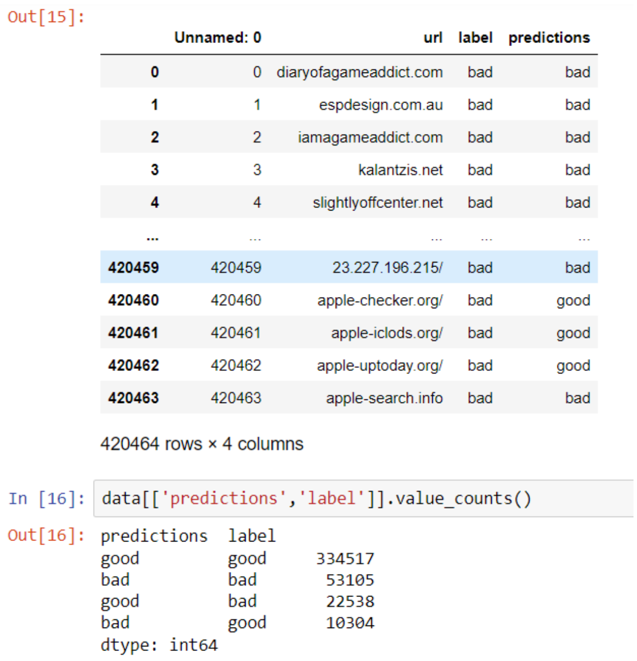
So, we can infer that the most important URL vector is the presence of HTTPS/HTTP.

As we could find the accuracy lacking for our 3 models, we tried to create a voting pipeline for these 3.



```
    urldata1['tld'] = urldata1['url'].apply(lambda i: get_tld(i,fail_silently=True)
    urldata1['tld_length'] = urldata1['tld'].apply(lambda i: tld_length(i))
    urldata1['count-'] = urldata1['url'].apply(lambda i: i.count('-'))
    urldata1['count@'] = urldata1['url'].apply(lambda i: i.count('@'))
    urldata1['count?'] = urldata1['url'].apply(lambda i: i.count('?'))
    urldata1['count%'] = urldata1['url'].apply(lambda i: i.count('%'))
    urldata1['count.'] = urldata1['url'].apply(lambda i: i.count('.'))
    urldata1['count='] = urldata1['url'].apply(lambda i: i.count('='))
    urldata1['count-http'] = urldata1['url'].apply(lambda i : i.count('http'))
    urldata1['count-https'] = urldata1['url'].apply(lambda i : i.count('https'))
    urldata1['count-www'] = urldata1['url'].apply(lambda i: i.count('www'))
    urldata1['count-digits']= urldata1['url'].apply(lambda i: digit_count(i))
    urldata1['count-letters']= urldata1['url'].apply(lambda i: letter_count(i))
    urldata1['count_dir'] = urldata1['url'].apply(lambda i: no_of_dir(i))
    urldata1['use_of_ip'] = urldata1['url'].apply(lambda i: having_ip_address(i))
    urldata1 = urldata1.drop(['url', 'tld'], axis=1)

    count_mal=0
    count_ben=0

    new_data1 = np.array(urldata1)
    prediction1 = log_model.predict(new_data1)
    prediction2= dt_model.predict(new_data1)
    prediction3= rfc.predict(new_data1)

    if prediction1[0] == 'bad':
        count_mal+=1
    else:
        count_ben+=1

    if prediction2[0] == 'bad':
        count_mal+=1
    else:
        count_ben+=1

    if prediction3[0] == 'bad':
        count_mal+=1
    else:
        count_ben+=1

    if(count_mal>count_ben):
        return "bad"
    else:
        return "good"
```

We utilize this voting system and we have improved on the accuracy to 92.18% from individual models. Also this result helps in concluding that the most important URL vector turns out to be HTTPS when it comes to categorizing malicious and benign websites through lightweight and fast ML models.

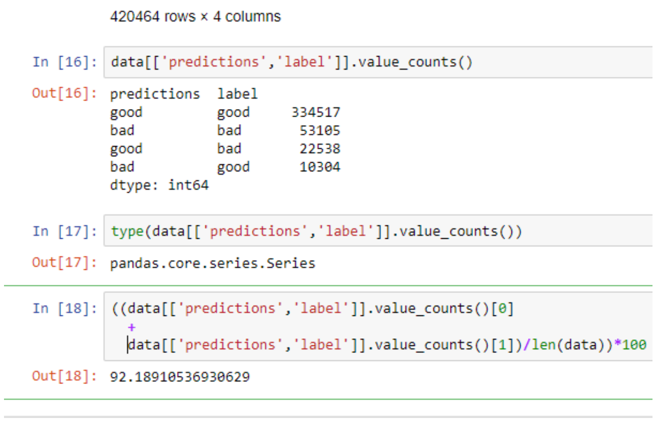


Out[15]:

| | Unnamed: 0 | url | label | predictions |
|---|---|---|---|---|
| **0** | 0 | diaryofagameaddict.com | bad | bad |
| **1** | 1 | espdesign.com.au | bad | bad |
| **2** | 2 | iamagameaddict.com | bad | bad |
| **3** | 3 | kalantzis.net | bad | bad |
| **4** | 4 | slightlyoffcenter.net | bad | bad |
| **...** | ... | ... | ... | ... |
| **420459** | 420459 | 23.227.196.215/ | bad | bad |
| **420460** | 420460 | apple-checker.org/ | bad | good |
| **420461** | 420461 | apple-iclods.org/ | bad | good |
| **420462** | 420462 | apple-uptoday.org/ | bad | good |
| **420463** | 420463 | apple-search.info | bad | bad |

420464 rows × 4 columns

```
In [16]:  data[['predictions','label']].value_counts()

Out[16]:  predictions  label
          good         good     334517
          bad          bad       53105
          good         bad       22538
          bad          good      10304
          dtype: int64
```



```
420464 rows × 4 columns

In [16]: data[['predictions','label']].value_counts()

Out[16]: predictions  label
         good         good     334517
         bad          bad       53105
         good         bad       22538
         bad          good      10304
         dtype: int64

In [17]: type(data[['predictions','label']].value_counts())

Out[17]: pandas.core.series.Series

In [18]: ((data[['predictions','label']].value_counts()[0]
           +
          data[['predictions','label']].value_counts()[1])/len(data))*100

Out[18]: 92.18910536930629
```

## X. REFERENCES

[1]https://www.vircom.com/blog/what-is-a-malicious-url/ - Joey Tanny

[2] https://purplesec.us/resources/cyber-security-statistics/

Jason - Purplesec author

[3] Vazhayil, Anu & Ravi, Vinayakumar & Kp, Soman. (2018). Comparative Study of the Detection of Malicious URLs Using Shallow and Deep Networks. 1-6. 10.1109/ICCCNT.2018.8494159.

[4] B. Liang, J. Huang, F. Liu, D. Wang, D. Dong, and Z. Liang, "Maliciousweb pages detection based on abnormal visibility recognition," in E-Business and Information System Security, 2009. EBISS'09. Interna-tional Conference on. IEEE, 2009, pp. 1–5.

[5]S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in Proceedings of the2007 ACM workshop on Recurring malcode. ACM, 2007, pp. 1–8

[6] A. Crisan, G. Florea, L. Halasz, C. Lemnaru, and C. Oprisa, "Detecting malicious URLs based on machine learning algorithms and word embeddings," in Proceedings of the 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 187–193, IEEE, Cluj-Napoca, Romania, September 2020.

[7] S. Singhal, U. Chawla, and R. Shorey, "Machine learning & concept drift based approach for malicious website detection," in Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 582–585, IEEE, Bengaluru, India, January 2020.

Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01. IEEE Computer Society

[8]A. Das, A. Das, A. Datta, S. Si, and S. Barman, "Deep approaches on malicious URL classification," in Proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–6, IEEE, Kharagpur, India, 2020, July.

[9] S. Tiwari, H. Rizvi and K. Kalaiselvi, "Malicious Website Navigation Prevention Using CNNs and URL Vectors: A Study," 2022 International Conference on Computer Communication and Informatics (ICCCI), 2022, pp. 1-6, doi: 10.1109/ICCCI54379.2022.9741056

[10] Dataset sources -1 https://www.unb.ca/cic/datasets/url-2016.html

-2 https://www.kaggle.com/sid321axn/malicious-urls-dataset