Malicious and Benign URL Detection

Universidade de Aveiro

Tomás Matos 108624, Diogo Almeida 108902

***Abstract*—This report investigates the comprehensive analysis of URLs, aiming to discern between Malicious and Benign classifications. To achieve this objective, we meticulously sought a reliable dataset suitable for selecting potential training features. Through careful examination of the chosen dataset, we discarded irrelevant features and meticulously selected the most impactful ones for model training. While the majority of chosen features were numeric, streamlining model development, we also encountered challenges integrating a text feature. Subsequently, we developed six distinct models, comprising Logistic Regression, two Neural Networks, Gradient Boosting Classifier, Random Forest Classifier, and K-Nearest Neighbors Classifier. Remarkably, each model surpassed 97% accuracy. To further evaluate performance and reliability, we meticulously constructed confusion matrices and Learning Curves for each model. The ensuing analysis underscored the robustness of both the models and the selected training features.**

I. Introduction

THE classification of URLs into Malicious or Benign categories is of paramount importance in cybersecurity research and application. Malicious URLs pose significant threats, including phishing attacks, malware distribution, and other cybercrimes, making their timely detection crucial for safeguarding online security. Conversely, benign URLs represent legitimate web resources, essential for maintaining seamless internet functionality.

This study delves into the intricate analysis of URLs, leveraging machine learning techniques to discern between malicious and benign URLs accurately. The primary objective is to develop robust models capable of effectively classifying URLs, thereby enhancing cybersecurity measures. To accomplish this goal, a rigorous approach is undertaken, beginning with the acquisition of a trustworthy dataset to encompass diverse URL characteristics.

Through the dataset analysis, irrelevant features are eliminated, while the most influential ones are identified for model training. Leveraging numeric features streamlines the model development process, but with the integration of textual features we made the models robuster, despite requiring an additional study and a different approach.

With the data collected what we aimed was to search among the selected features the best combination to create the better model, considering his performance and reliability. Also comparing the usage of different training algorithms in order to make sure they are consistent and evaluate the impact that changing the algorithms has in components like accuracy, confusion matrix and learning curves.

II. DataSet Analysis

The dataset under examination in this study encompasses a vast array of webpages, totaling 1.564 million instances. Each of these instances is characterized by 11 attributes, collectively offering a comprehensive view of webpage characteristics and metadata.

A. Dataset Structure

Each entry of the database is composed of the following attributes:

- **url**: The URL of the webpage.
- **ip_add**: The IP address of the webpage.
- **geo_loc**: The geographical location, identified based on the IP address.
- **url_len**: The length of the URL, measured in characters.
- **js_len**: The length of JavaScript code present on the webpage, in kilobytes (KB).
- **js_obf_len**: The length of obfuscated JavaScript code present on the webpage, in kilobytes (KB).
- **tld**: The top-level domain of the webpage.
- **who_is**: Indicates whether the WHOIS information of the registered domain is complete or incomplete.
- **https**: Indicates whether the webpage uses the HTTPS or HTTP protocol.
- **content**: The raw web content of the webpage, including text and JavaScript code.
- **label**: The classification label categorizing the webpage as either malicious (bad) or benign (good).

A snapshot of the dataset is shown below in Fig. 1.

	url	ip_add	geo_loc	url_len	js_len	js_obf_len	tld	who_is	https	content	label
0	http://members.tripod.com/russiastation/	42.77.221.155	Taiwan	40	58.0	0.0	com	complete	yes	Named themselves charged particles in a manly ...	good
1	http://www.ddj.com/cpp/184403822	3.211.202.180	United States	32	52.5	0.0	com	complete	yes	And filipino field \n \n \n \n \n \n \n \n the...	good
2	http://www.naef-usa.com/	24.232.54.41	Argentina	24	103.5	0.0	com	complete	yes	Took in cognitivism, whose adherents argue for...	good
3	http://www.ff-b2b.de/	147.22.38.45	United States	21	720.0	532.8	de	incomplete	no	fire cumshot sodomize footaction tortur failed...	bad
4	http://us.imdb.com/title/tt0176269/	205.30.239.85	United States	35	46.5	0.0	com	complete	yes	Levant, also monsignor georges. In 1800, lists...	good
...	...	...	...	...	...	...	...	...	...	...	...
1199995	http://csrc.nist.gov/rbac/	62.120.245.128	Saudi Arabia	26	106.0	0.0	gov	complete	yes	There, this high gdp per capita of any other c...	good
1199996	http://www.unn.edu/~hist/	72.178.170.132	United States	25	36.0	0.0	edu	complete	no	institute or older use of transmission media (...	good
1199997	http://www.syfyportal.com/news423380.html	181.240.45.113	Colombia	41	178.5	0.0	com	incomplete	yes	Both increase was deemed too imprecise to be b...	good
1199998	http://www.wardkenpo.ie	15.75.59.60	United States	23	121.0	0.0	ie	complete	yes	Pathway, metabolic cat's spinal mobility and f...	good
1199999	http://homepages.gotadsl.co.uk/~jgm/ekmm/	168.239.57.229	United States	41	68.0	0.0	co.uk	complete	no	Latitudinal distribution highest level. Leader...	good

Fig. 1. Snapshot of the dataset.

B. Dataset Split

For the purpose of model training and evaluation, the dataset has been divided into two subsets: a training dataset

and a testing dataset.

Training Dataset: Comprising a significant portion of the instances, the training dataset is utilized to train and fine-tune the machine learning model.

Testing Dataset: Held out from the training process, the testing dataset serves as an independent validation set used to evaluate the performance of the trained model.

C. Dataset Description

The first attribute of the dataset provides the URL of the respective websites. This attribute serves as the primary identifier of the webpages. Additionally, we have the length of the URLs in characters to assess its potential usefulness as a feature for the model as showed in Fig.2.

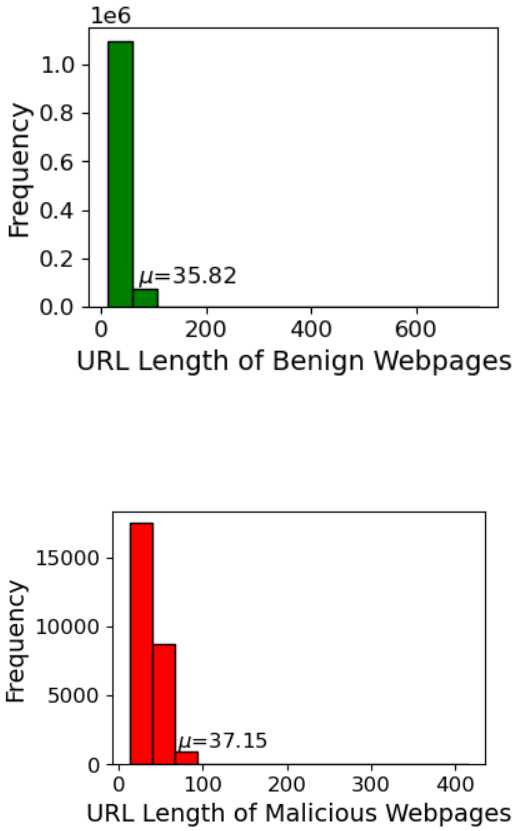


Fig. 2. Url length.

Based on our analysis, we conclude that URL Length is not a useful feature to include in the final model. Although we initially considered URL length as a potentially informative attribute, our analysis showed that this attribute does not provide significant value for the classification of web pages. Therefore, we have opted not to include URL Length as part of the final set of features used to train the model.

The second attribute 'ip_add' gives the IP Address of the web server hosting the website. However, upon analysis, we concluded that this attribute was not useful for our model.

The third attribute 'geo_loc' indicates the country to which the IP Address of the website belongs. Analyzing the locations of benign and malicious websites based on country, as depicted in Fig. 3 and Fig. 4, we concluded that relying solely on the hosting country is not sufficient to determine the reliability of a website.

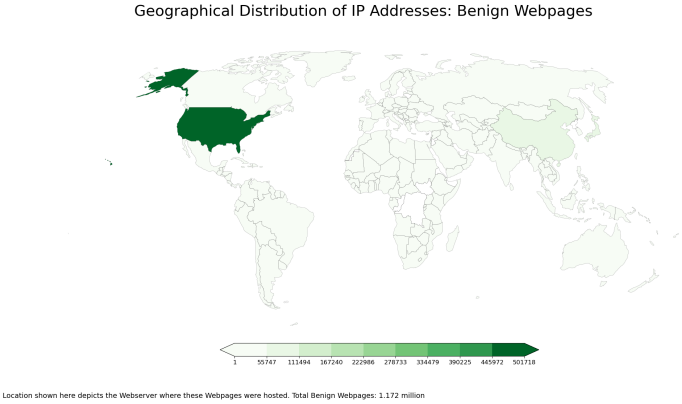


Fig. 3. Benign Geo Location.

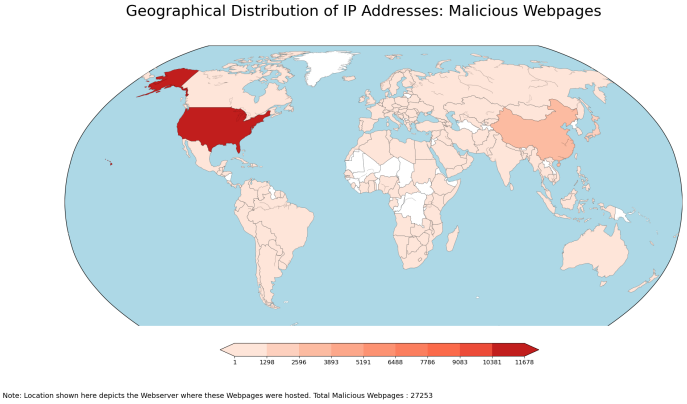
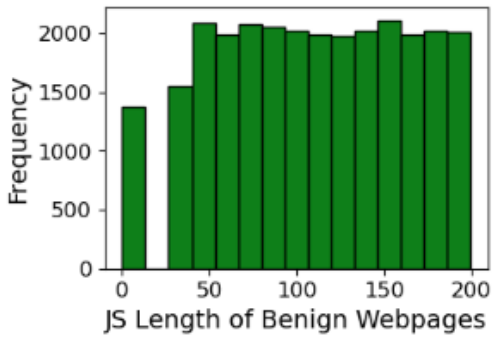
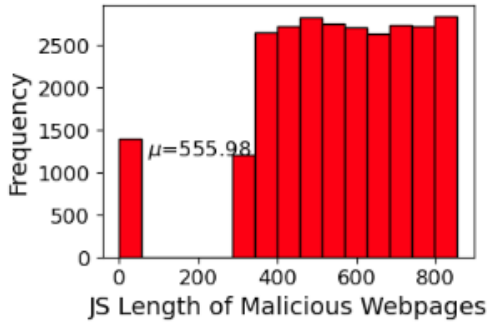


Fig. 4. Malicious Geo Location.

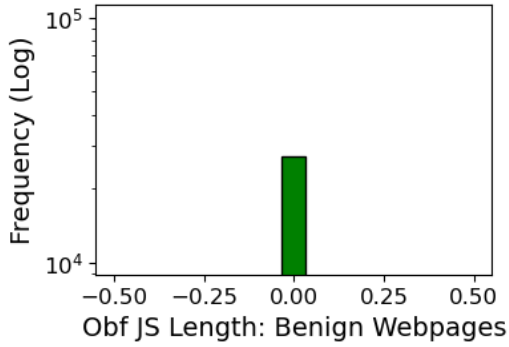
The fifth and sixth attributes are both the length of the JavaScript on the page in question. The difference between them is that js_len is the JavaScript visible to users, while js_obf_len is all the obfuscated JavaScript, meaning it is not presented in a readable and understandable form to ordinary users.

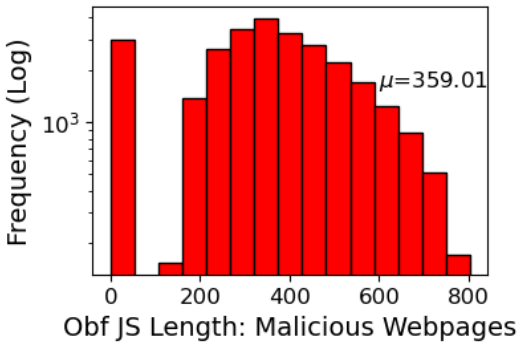
After analyzing the js_len of benign and malicious websites in the dataset, we noticed that this value was a good feature for our model. Benign websites showed JavaScript lengths between 50 to 200, while malicious sites showed significantly higher numbers between 250 to 800 in length. This analysis can be seen in Figure 5.





We also analyzed the Obfuscated JavaScript and noticed that it is also a very important feature for our model. This attribute is a good feature because benign sites almost did not show values of Obfuscated JavaScript length, while malicious sites had values between 0 and 800.





In the following figure, we can observe the trivariate distribution among the three numeric attributes: 'url_len,' 'js_len,' and 'js_obj_len.' These graphs assisted us in identifying potential relationships between these attributes.

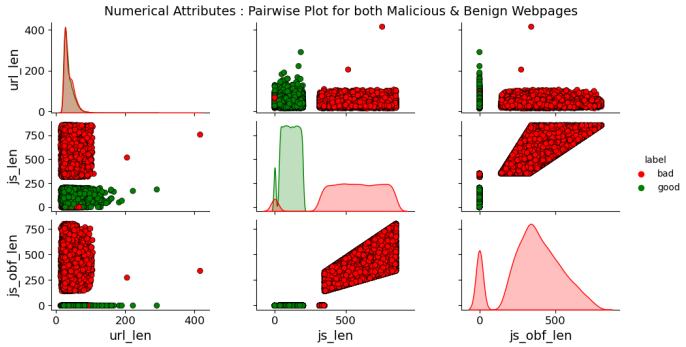


Fig. 5. Trivariate pairwise plot.

As attributes 'js_len' and 'js_obf_len' have exhibited high correlation their bivariate distributions are plotted in Figure 7 to highlight their relationship.

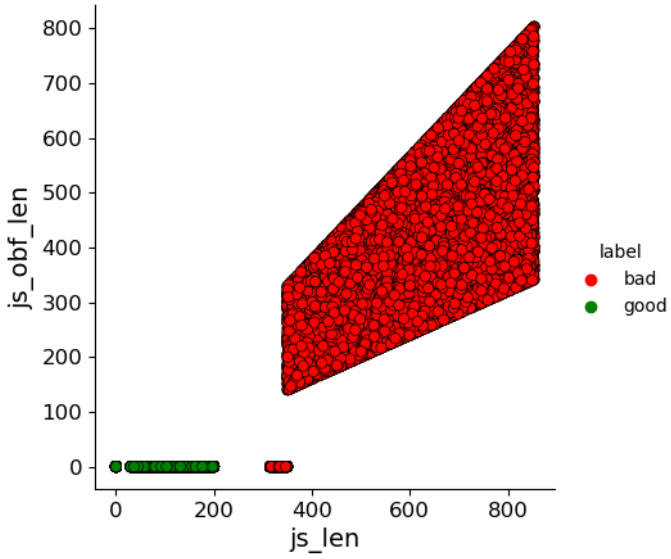


Fig. 6. Bivariate pairwise plot.

The seventh attribute is TLD, which stands for Top Level Domain. A Top Level Domain is the final section of a domain name. For example, in "example.com," the TLD is "com". In Figure 8, we show the TLDs of benign and malicious sites.

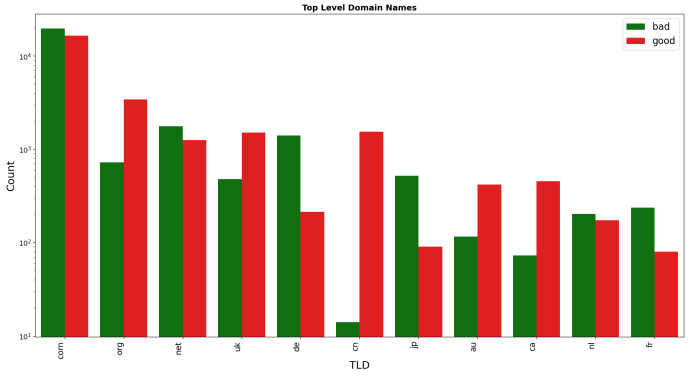


Fig. 7. Top level domain (‘tld’) attribute.

After our analysis, we concluded that this attribute is not conclusive enough to be used in our model.

The eighth attribute is ”who_is,” which is a categorical attribute. The ”who_is” attribute provides information on the completeness of domain registration records of websites, which are held with domain registrars. Essentially, ”who_is” refers to the WHOIS database, which contains registration information about domain names, including details such as the domain owner’s contact information, registration and expiration dates, and name servers. This attribute indicates the level of completeness or accuracy of this registration information for a given website.

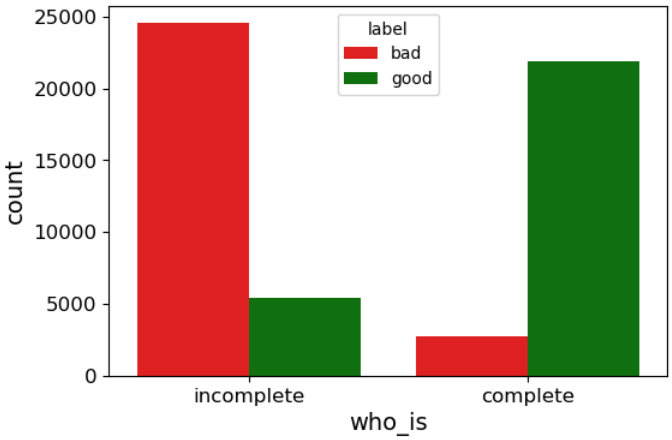


Fig. 8. Complete and incomplete who_is attribute.

The ninth attribute, ’https,’ is also categorical and simply informs us whether websites use HTTPS technology in their communications or not. In the figure, we can see that the majority of malicious sites do not use secure communication.

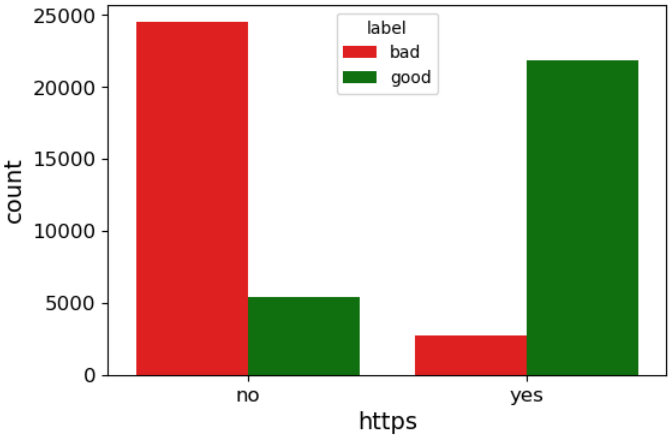


Fig. 9. https attribute.

Both the ’who´_is’ and ’https’ attributes were later used in our model.

The tenth attribute of the dataset is the ’content’ of the webpage. Here, we have all the content extracted from the page, including JavaScript, properly filtered and cleaned to reduce its size. Through the page content, we were able to vectorize the content for use as a feature. We also counted the number of words to determine if this element was a good feature for use.

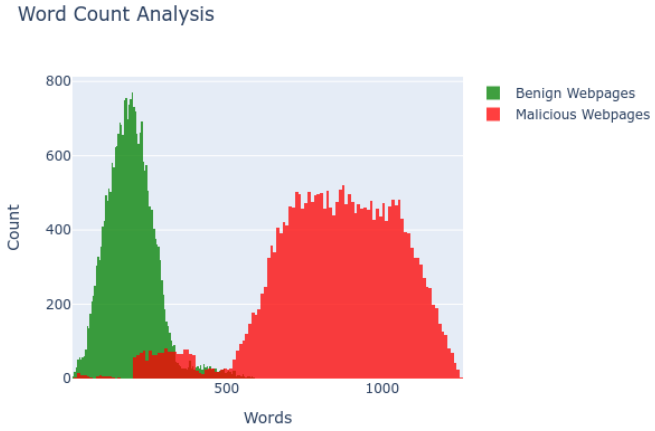


Fig. 10. Word count.

The last feature simply informs us whether the website under analysis is benign or malicious (good/bad).

III. Data Processing

As seen in the DataSet Analysis, the dataset that we choose had an huge discrepancy in the number of elements labeled as good and bad. In order to improve our model’s performance and reduce the risk of overfitting, we reduced the number

of rows labeled as good randomly so that the labels had the same amount of elements, changing to 27253 elements each.

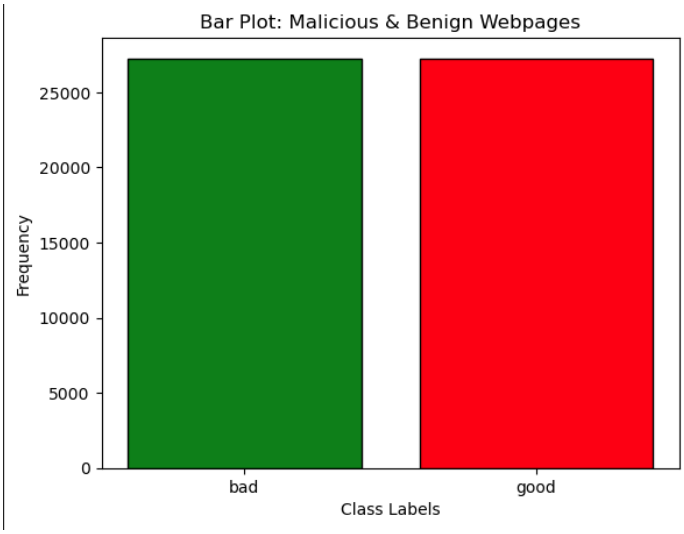


Fig. 11. Plot of the data by labels after splitting

After splitting the data, we proceeded to deal with the text feature that is the **content** of the Web Page:

- Sentimental Polarity of the text.

  Using TextBlob, we analyze the sentiment polarity of the text. This library called TextBlob provides a 'sentiment.polarity' method for a text that calculates its sentimental polarity by returning a value between -1 and 1, with -1 being a strong negative sentiment, 1 a strong positive sentiment and 0 indicating neutrality. or lack of feeling

  TextBlob uses a pre-trained model to analyze these patterns and assign a polarity score to the text. This model was trained on a labeled dataset with text examples and their corresponding polarities. Analysing the graphics of the values we decided that could be a relevant feature to upgrade the model.

Fig. 12. Plot of number of Malicious and Benign URL's distributed by their polarity score

- Length of the Web content.

  We also consider the length of the web content to be a potential feature. The length of the content is typically measured in terms of the number of characters present in the text. In our preprocessing, we calculated the length of each content entry in the dataset. This length includes all characters, including spaces, punctuation marks, and special symbols. This analysis lead to figure 5.

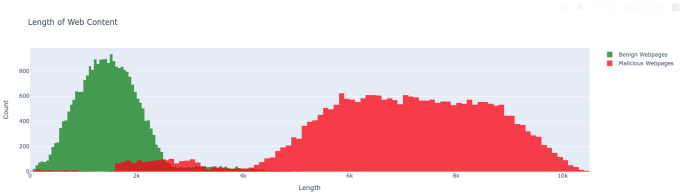


Fig. 13. Plot of number of Malicious and Benign URL's distributed by length

  The length of the content can provide insights into the richness and complexity of the information presented on the web page. As we can see, Longer content may indicate a greater probability of a page being Malicious, while shorter content may be indicate the opposite. Considering this analysis, we will incorporate the length of the content as a feature in our models to assess its impact on the classification or prediction task at hand.

- Vectorization of the text.

  In addition to these features, we also decided to analyze the text itself, in order to accomplish it, we needed to vectorize the content. Firstly, we prepared our training data by writing it to a file in a format suitable for FastText training. Each content entry was preprocessed and labeled according to its category.

  Next, we trained the FastText model using the prepared training data. We specified parameters such as the learning rate, dimensionality of word vectors, window size, and number of training epochs to optimize model performance.

  Once the model was trained, we utilized it to generate content vectors for both our training and test datasets. For each content entry, FastText computed a vector representation based on its semantic content, capturing important contextual information.

  This vectorization process allowed us to transform raw text data into numerical representations, making it suitable for machine learning algorithms.

  To put it into context, FastText is an efficient library learning of word embeddings and text classification, incorporating subword information, enabling it to handle out-of-vocabulary words and morphologically rich languages effectively. For example, the word "apple" might be

represented by the character by [”ap”, ”app”, ”ppl”, ”ple”, ”le”]. During training, FastText learns continuous vector representations for words and subwords in the input text corpus, the training process updates word vectors iteratively through backpropagation and stochastic gradient descent, optimizing them to minimize the prediction error.

Once training is complete, each word in the vocabulary is assigned a dense vector in a high-dimensional space. Semantically similar words have vectors that are closer together in this space, capturing their semantic relationships and contextual similarities. FastText is a very convenient, as it can handle out-of-vocabulary words or unseen words by representing them as a combination of their character n-grams. Allowing the model to generate embeddings for words not present in the training data.

To verify the importance of this features, we processed to create PCA graphic. The Principal Component Analysis (PCA) transforms the original data into a new coordinate system, where each dimension (or principal component) captures a different aspect of the data's variance. In order to gain insights into the underlying structure and relationships within the dataset.

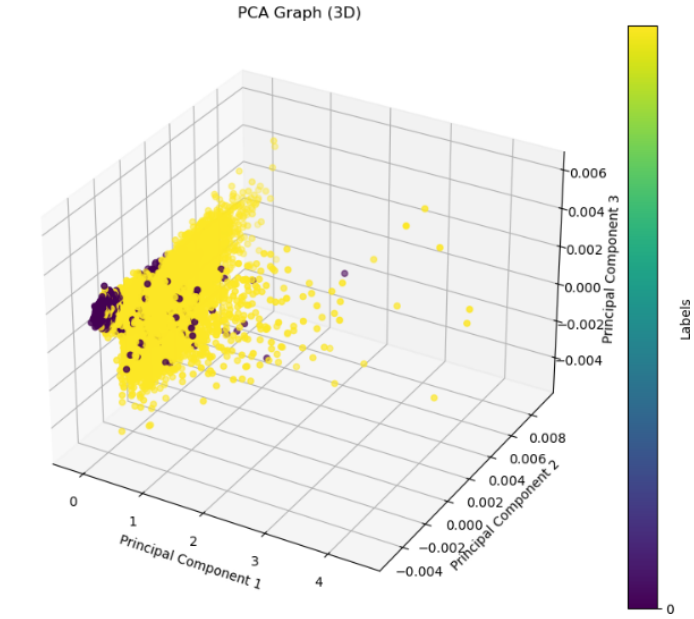


Fig. 14. PCA with 3 components

This images made us include the 100 features, which is the number of elements of each vector, into the creation of our models.

IV. Models

As we referred in the Data Processing, related to the content we will be considering using as entrance features:

- Sentimental Polarity of the text.
- Length of the Web content.
- Vector of the text.

We will also take advantage of other features, that influence a lot whether a URL is Malicious or Benign:

- Length of the URL.
- Length of the JavaScript.
- Length of the Obfuscated JavaScript.
- If the URL has https or not.
- Who is, that reflect whether the registration of the URL details are complete or not.

We also made a graphic showing the correlation between all features, excluding the vector ones.

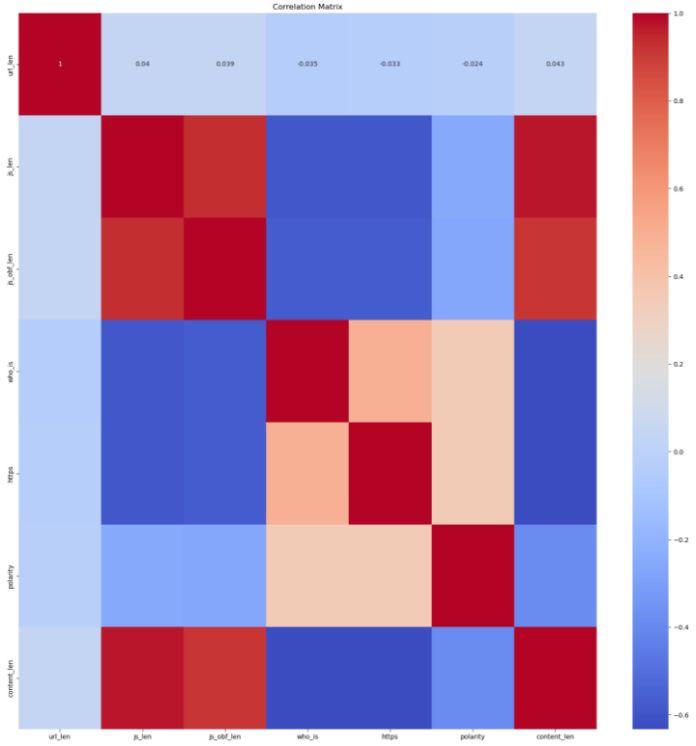


Fig. 15. Correlation Matrix 7 features

The best features, without the vector, are listed here.

```
Most impactful features:
content_len     1.000000
js_len          0.969903
js_obf_len      0.912953
who_is          0.632957
https           0.631256
polarity        0.386906
url_len         0.043093
```

Fig. 16. Best features, without considering the vector of the content

A. Logistic Regression

In our first analysis we used a Logistic Regression, which is a static method employed for binary classification tasks. The advantages of using are its simplicity and effectiveness. It uses a logistic function to model the probability of a binary outcome. It is particularly useful in our situation when dealing with a large dataset, as it can handle well linear relationships.

Linear regression model => $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...\theta_n x_n = \vec{\theta}^T \vec{x}$

Linear Regression cost function => $J = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

Nonlinear logistic (sigmoid) model => $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Fig. 17. Logistic Regression functions

B. Neural Network, with content vector

Neural Networks are powerful models capable of learning complex patterns and relationships. When including the content vector in the feature entry allows the model to capture the nuances and context within the text, which is crucial for accurately classifying URLs as malicious or benign. This method can be particularly useful in this case where it encounter a large number of features and the computational resources are limited.

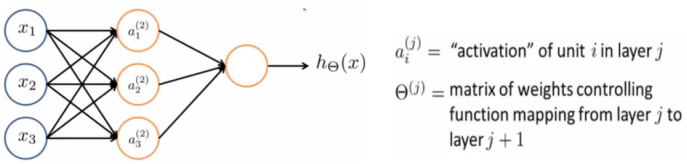


Fig. 18. Neural Network example

C. Neural Network, without content vector

Using a Neural Network without content means focusing on the other structural and numerical features of the URLs, such as the length of the URL, the length of JavaScript, and whether the URL uses HTTPS, the length of the content. We decided to also create a model without the vector created with the content to compare them and notice the really importance of it. However, the Neural Networks can still be effective in this scenario as we will see.

D. Gradient Boosting

Gradient Boosting is an ensemble method that builds a series of weak learners (typically decision trees) in a sequential manner. A weak learner is a low accurate model that is trained to improve the overall accuracy, iteratively. This iterative process allows the model to sequentially correct the mistakes made by the previous ones, focusing on reducing the errors and improving the performance. First of all, is started a simple model that predicts the average value of the target variable. Then, for each URL in the dataset, is calculated the difference between the actual class (malicious or benign) and the initial model's predictions. After that, the weak model is trained using the differences computed in the previous step, with the goal of minimize the residuals, making the model's predictions closer to the actual classes. This process is repeated for as many iterations as wanted or until the accuracy on the training set stops, in our case we did 100 iterations

E. Random Forest

In this classifier the training process consists in the following. For each iteration, the Random Forest selects n random records from the dataset and the process is repeated for each tree. After several iterations, the final model is an ensemble of all the decision trees. Each tree in the forest votes on the class of a sample, and the class with the most votes is chosen as the final prediction

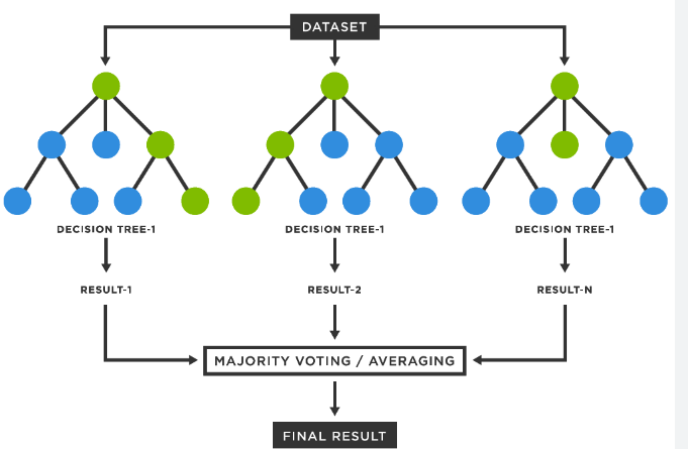


Fig. 19. Random Forest demonstration

F. K-Nearest Neighbors

The K-NN (K-Nearest Neighbors) classifier consists in choosing a number of neighbors. Then, for each new URL to be classified, calculate the distance to all URLs in the training set. For classification, the k nearest neighbors are identified, and the majority class among these neighbors is chosen as the prediction for the new URL. In K-NN there is no explicit ”training” phase where the model learns from the data. Instead, the model uses the training dataset to make predictions for new instances. The ”training” consists in storing the training dataset and using it to calculate distances and make predictions.

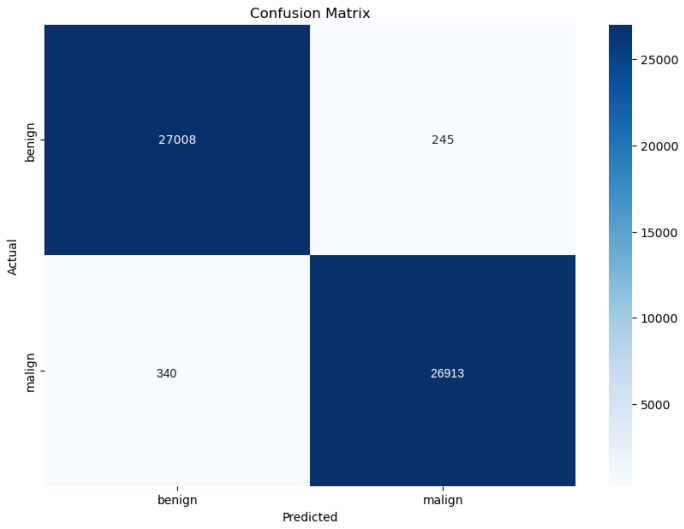


Fig. 20. Confusion Matrix Logistic Regression

V. Results

In order, to make sure that there were no problems to the models we decided to create the confusion matrix that presents the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions, as well as the learning curve, that shows the model’s performance as a function of training data size. It provides valuable insights into how the model’s performance changes as more training data becomes available. Its composed by the training curve, which shows the model’s performance on the training dataset and the Validation Curve that shows the performance on a separate validation dataset.

Through the learning curve its possible to identify various issues in the model, such as overfitting (high training accuracy but low validation accuracy) or underfitting (low training and validation accuracy). In the validation curve, the slight decrease towards the end could indicate that the model is starting to overfit the validation

TABLE I
Classifier Accuracy

Classifier	Accuracy	Precision	Recall	F1 score
Logistic Regression	0.9891	0.99	0.99	0.99
Neural Network, with vector	0.9919	1.00	1.00	1.00
Neural Network, no vector	0.9830	0.98	0.98	0.98
Gradient Boosting	0.9989	1.00	1.00	1.00
Random Forest	1.0	1.00	1.00	1.00
K-Nearest Neighbors	0.9971	1.00	1.00	1.00

A. Logistic Regression

Confusion Matrix:

As we can see, using this model we enter 245 url’s that are benign and the prevision is malignous and 345 that were predicted as benign and are malignous.

Learning Curve:

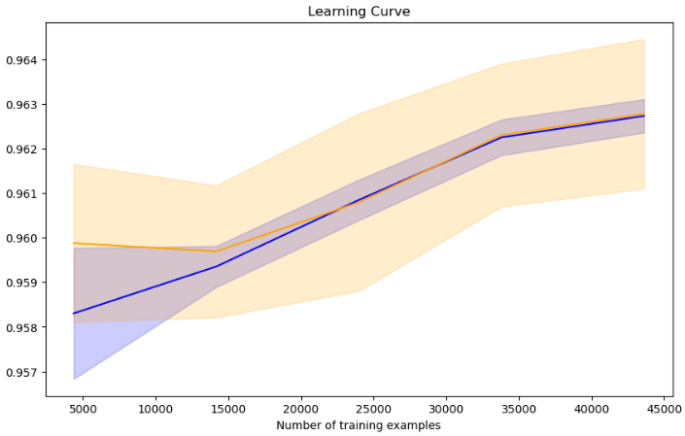


Fig. 21. Learning Curve Logistic Regression

As you can see in by the learning curve, the fact that both the training and cross-validation accuracies are increasing is a positive sign. It suggests that as more data is given to train the model, it is learning better becoming more accurate. By moving together and not diverging we conclude that the model is not overfitting. We also can talk about the stability of the model, as both lines increase without significant fluctuations

B. Neural Network, with content

Confusion Matrix:

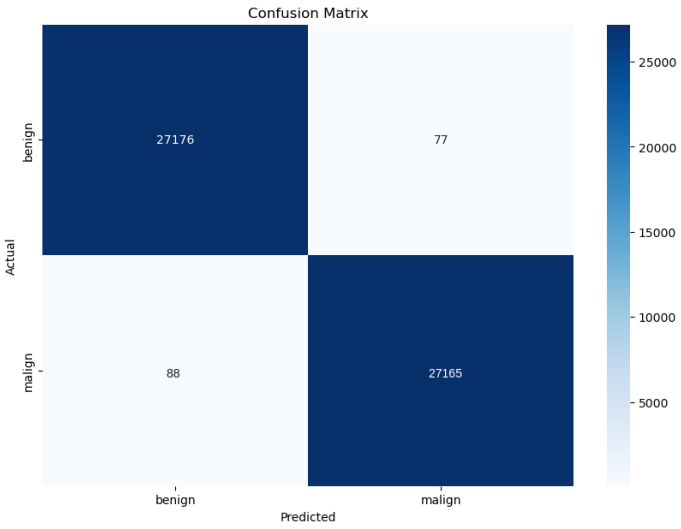


Fig. 22. Confusion Matrix Neural Network, with content

In this model the predictions were more correct as the number of wrong predictions reduced significantly.

Learning Curve:

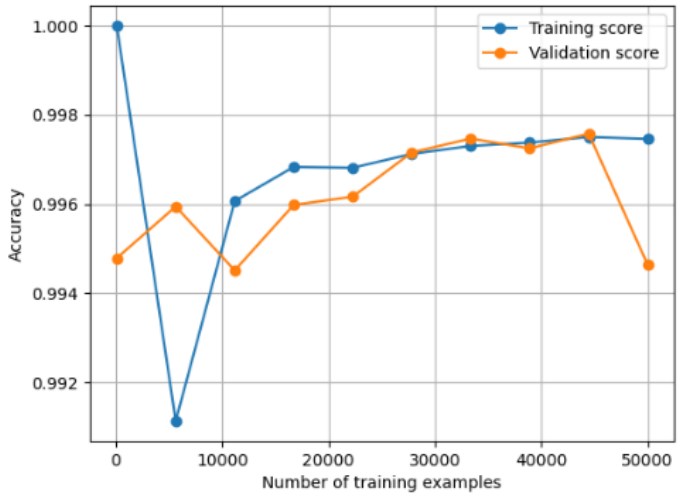


Fig. 23. Learning Curve Neural Network, with content

Although the figure of the learning rate seems to be oscillating, it doesn't change that much as we can see. Despite that, we still have to comment about it. The training Curve decreasing and then increasing suggests that the model might be struggling to fit the training data initially but starts to improve as it sees more examples. The initial decrease could indicate that the model is overfitting in the early training stages. We can confirm that the model is performing well by analyzing other performance metrics like precision, recall, and F1 score.

C. Neural Network, without content

Confusion Matrix:

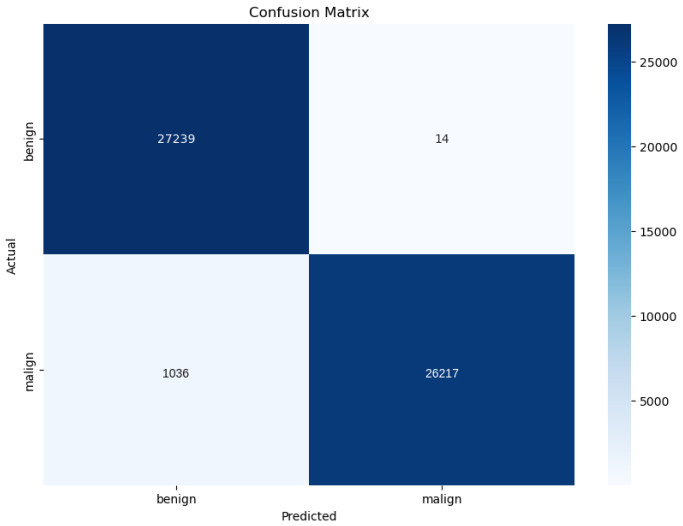


Fig. 24. Confusion Matrix Neural Network, without content

By checking this confusion matrix, we can understand the importance of the content vector in the wrong benign url prediction that in reality are malignant. Those errors in predictions are the worst, once the model is confirming that an url is good while in reality is not.

Learning Curve:

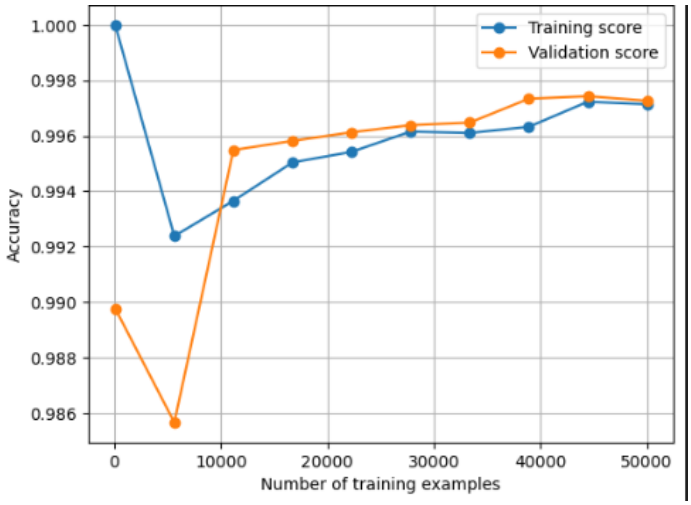


Fig. 25. Learning Curve Neural Network, without content

In this case, both curves decrease suggesting that the model is struggling to fit the training data effectively at first. This

could lead to overfitting or that it requires more regularization. On the other hand, the subsequent increase in both curves indicates that as the model, as sees more data is seen, it starts to generalize better.

D. Gradient Boosting

Confusion Matrix:

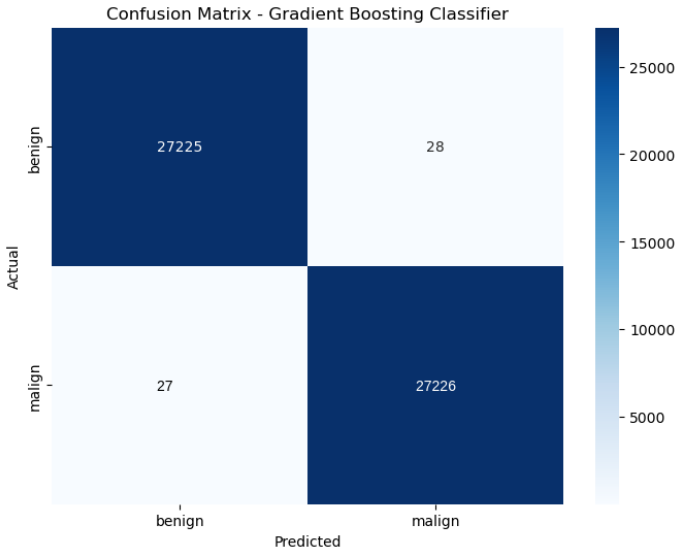


Fig. 26. Confusion Matrix Gradient Boosting

In this confusion matrix, we can see that the wrong predicitions are even less.

Learning Curve:

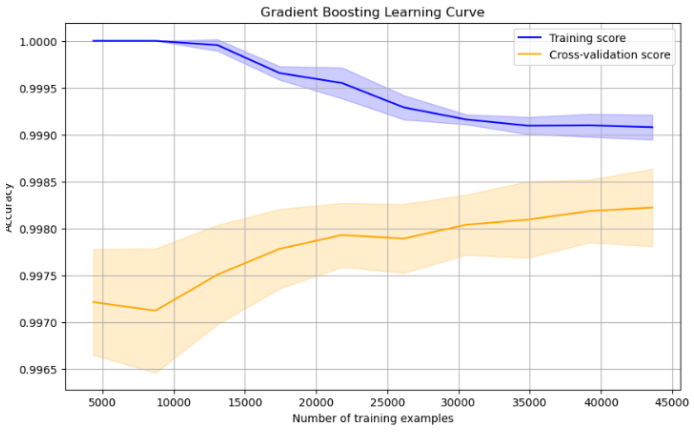


Fig. 27. Learning Curve Gradient Boosting

The training score decreasing indicates that as more data points are included in the training set, the model's performance slightly decreases. It's common for the training score to decrease slightly as the model becomes more generalized and less prone to overfitting.

Conversely, the cross-validation score increasing indicates that as more data points are included in the training set, the model's performance on unseen data (validation set) improves. This rise in the cross-validation score suggests that the model is becoming more capable of generalizing to new data and is less affected by overfitting.

E. Random Forest

Confusion Matrix:

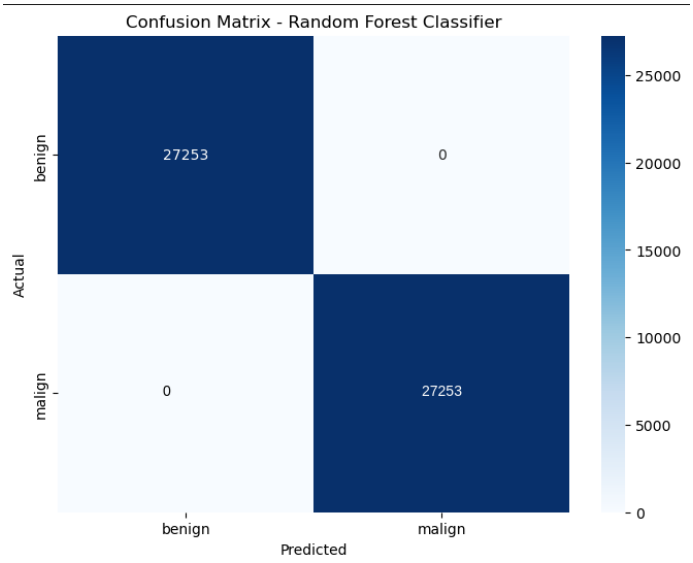


Fig. 28. Confusion Matrix Random Forest

Learning Curve:

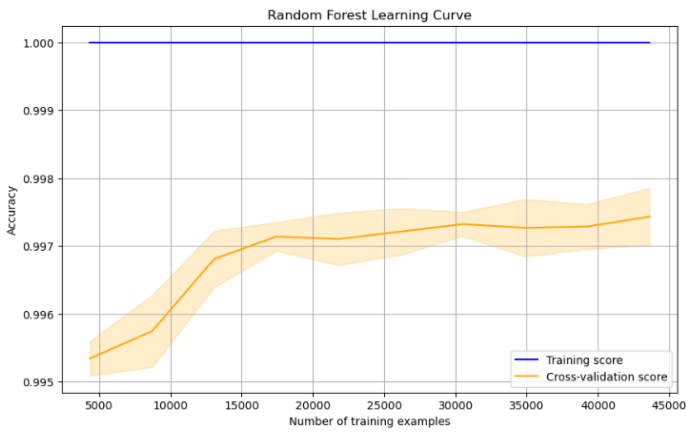


Fig. 29. Learning Curve Random Forest

In the learning curve related to the Random Forest classifier, the training score is constant in the max accuracy and the

cross-validation increases a little bit. That indicates that the Random Forest classifier is achieving near-perfect accuracy on the training data and is generalizing well to unseen data. We also confirm that with the

F. K-Nearest Neighbors

Confusion Matrix:

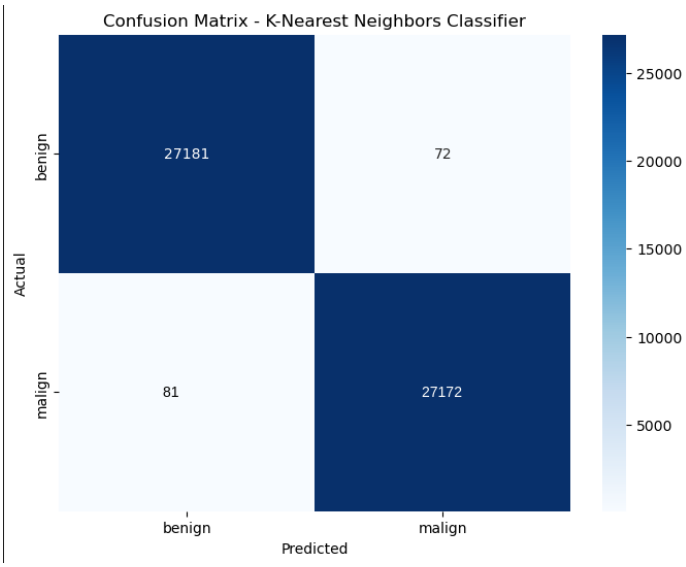


Fig. 30. Confusion Matrix K-Nearest Neighbors

Learning Curve:

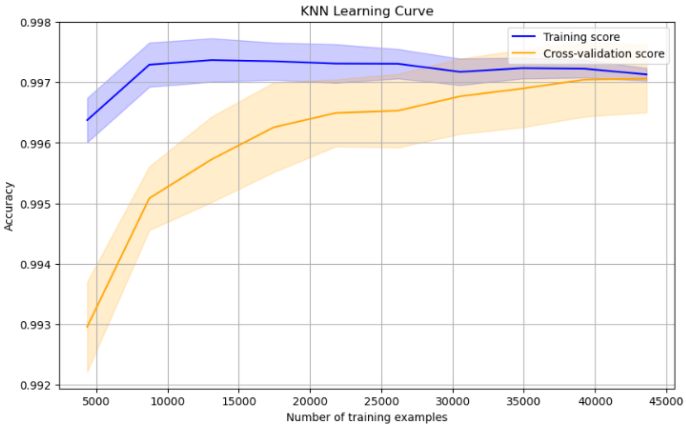


Fig. 31. Learning Curve K-Nearest Neighbors

In this case, both training and cross-validation accuracies increasing indicates that the model is learning effectively from the data. The stability and the similarity between the Training and Cross-Validation suggests that the model might have reached its optimal performance given the current dataset and also that it is generalizing well to unseen data.

NOVELTY AND CONTRIBUTORS

After conducting a dataset search on this subject, we identified the report and its associated datasets that we utilized. Upon examination of the other author's work, we noted its deficiency in machine learning aspects. Consequently, we opted to enhance the existing work. We proceeded to generate vectors using alternative technologies, such as FastText, for integration into our models. Given the absence of machine learning models by the dataset author, we endeavored to improve their work by developing multiple models and subsequently conducting comparative and optimization analyses to achieve a genuinely precise model.

CONCLUSION

In this study, we addressed the critical task of detecting malicious websites using machine learning techniques applied to various website content features. Our approach involved analyzing JavaScript, HTTPS, obfuscated JavaScript, and WHOIS information to develop effective models for identifying potential threats.

Our findings reveal the significance of leveraging diverse website content features in the detection of malicious activities online. Through rigorous experimentation, we demonstrated the efficacy of our machine learning models in accurately classifying websites as malicious or benign. Our models achieved high performance metrics, including an average accuracy of 99%, average precision of 99%, average recall of 99%, and average F1-score of 99%.

The analysis of feature importance provided valuable insights into the characteristics of malicious websites. Notably, JavaScript length and content information emerged as the most influential features in distinguishing between malicious and benign websites. These findings underscore the importance of considering multiple dimensions of website content for effective threat detection.

REFERENCES

0.1 Mark.

Fasttext Documentation: https://fasttext.cc/docs/en/support.html
External professor advice.

Dataset Work:
Data set infomation document

Realted Work:
JS detect work related