CRYPTOGRAPHY

BITS F463

Term Project

A Report On

Real Time Detection of Malicious URLs using Machine Learning

Prepared by-

Raj Shree Singh 2017B4A70808P

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

SUBMITTED TO:
Dr. Ashutosh Bhatia

TABLE OF CONTENTS

OBJECTIVE....................3
MOTIVATION....................3
INTRODUCTION....................3
METHODOLOGY....................5
RESULTS 9
CONCLUSION....................12
REFERENCES....................13

OBJECTIVE

Building a **Real Time Detection System for Malicious URL using Machine Learning.**

MOTIVATION

While browsing the internet (particularly Mail and Web Interaction websites) we come across various URLs while sharing data and communicating readily. Attackers trick the users into clicking into clicking on Malicious URLs due to which a user looses sensitive information. Companies and Organizations used Intrusion Detection Systems to defend against these types of attacks. This projects attempts to identify such Malicious URLs in Realtime using Machine Learning and thus help the users in prevention against such attacks.

INTRODUCTION

The internet plays a major role in modern life. The primary reason for its widespread adoption is the flexibility it offers which comes with a price in the form of cyberattacks. These cyberattacks affect a large number of unsuspecting users every year. McAfee and the Center for Strategic and International studies (CSIS) estimated the global annual cost of cybercrimes to be almost 600 billion US dollars. [1]

Symantec's 2016 Internet Security Report [2] mentions various global threats that includes attacks on browsers and websites, corporate data breaches, spear phishing attempts, ransomware and several other types of fraudulent cyber-attacks. One such strategy used by attackers nowadays Drive-By-Download. The idea behind this attack it to users into visiting a website that contains malicious code. In this manner the attackers can also compromise sensitive customer information. The most popular techniques to detect malicious URLs is by using blacklists which are collected from various techniques such as honeypots, manual reporting and web crawlers. These blacklists are shared widely with Intrusion Detection System (IDS) vendors to block access to these websites. For example, KISA (Korea Internet Security Agency), a government funded security which maintains a list of blacklist URLs.

While URL blacklist might be effective to some extent it is easy for attacker to bypass it as the primary method to detect malicious URLs is static. Attackers modify one or more components of the URL string to deceive the system. So, an automated framework to detect new (previously unseen) Malicious URLs with high accuracy without any human interaction is required. Several studies in literature tackle this issue through a Machine Learning model [3]-[7]. Machine learning approach begins with collection of a dataset. This dataset compiles a list of URLs that have been classified as either benign or malicious and characterize them via a set of attributes. A classifier is then trained with the collected dataset with the underlying assumption that a malicious URL has significantly different feature distribution than a benign URL. The dataset used in this project is ISCX-URL-2016 by Canadian Institute for Cybersecurity [8]. This dataset contains over 110,000 URLs labelled into five different categories – Benign, Spam, Phishing, Malware and Defacement. For the purpose of this project, all the URLs were grouped into either Benign or Malicious (having Spam, Phishing, Malware and Defacement).

This project briefly aims to-

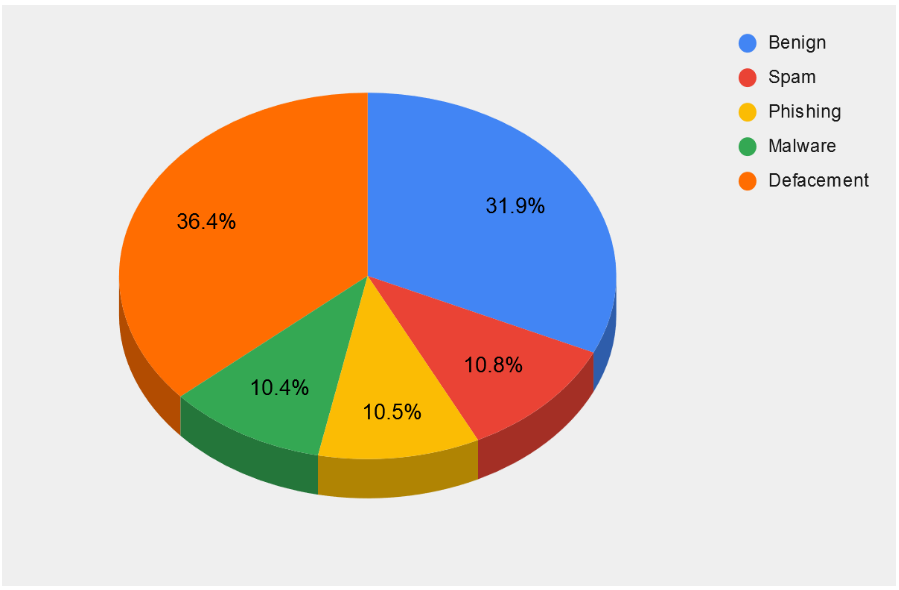
1. Formulate a Web Application to automatically detect Malicious URL.
2. Tackle the problem of unbounded growth of URL space using an incremental learning methodology.
3. As feature extraction from URLs is a time-consuming process, this project proposes a delayed feature collection algorithm to increase performance.
4. Test the accuracy and performance of this application via 10-fold cross-validation

METHODOLOGY

1. Data Pre-Processing-

After removing the invalid URLs from the ISCX-URL 2016 dataset, the following composition of database was used for the model-

1. 35378 Benign URLs
2. 75622 Malicious URLs
   a. 12000 Spam URLs
   b. 11682 Phishing URLs
   c. 11500 Malware URLs
   d. 40450 Defacement URLs

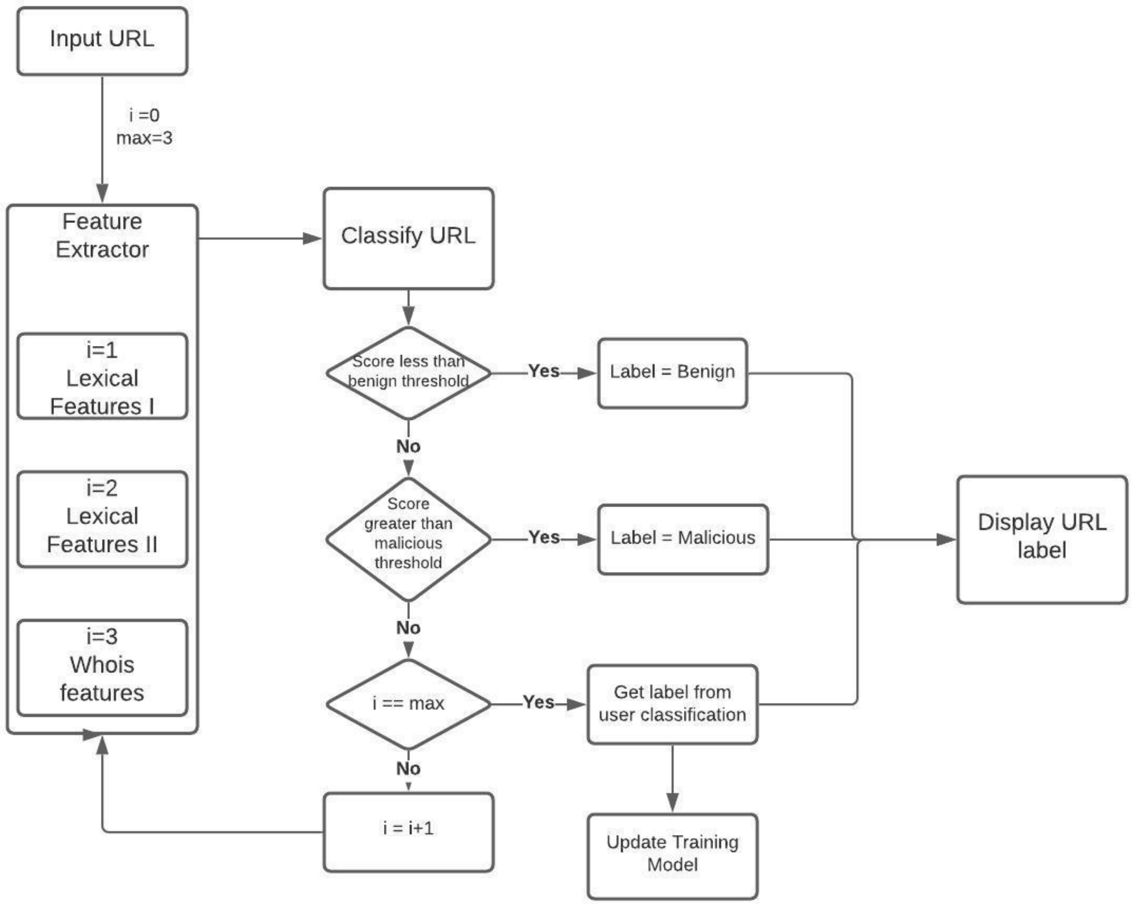


The dataset was then normalized using the normalize function from keras.util module.

2. Dataflow Overview-

- The user inputs the URL on the webpage and the application collects first set of features. Then it classifies the URL accordingly to give a score for each class.
- For the purpose of this project, instead of an online batch classifier an incremental learning classifier is chosen. The dataset grows in an unbounded manner and it is time consuming to retrain a batch learning classifier every time to URL however an incremental classifier can be efficiently updated with one data sample making it more fit for the purpose at hand.
- The score obtained above is used to decide whether to collect the next set of features or not. If the probability is considerably high, then the class label is displayed on the webpage otherwise the application continues to collect feature sets unless a high confidence on its class is achieved.
- In case it fails to classify the URL even after collecting all the feature sets, the application asks the user to decide its label and the classification model is updated with the features and label of this URL.

The following flowchart concludes dataflow discussed above-



3. Feature Selection and Delayed Feature Collection-

As seen above, the application extracts 3 sets of features-

3.1. Lexical Features I-
These features are extracted from the URL string itself. The lexical features used here are- 'entropy’, 'numDigits','urlLength', 'numParams', 'numFrags','numSubDomains', 'urlMaxSymSeq'.

3.2. Lexical Features II-
These lexical features are extracted after parsing the URL using urllib. These features consume more time for extraction than Lexical Features I.
For eg- 'entropyDomain', 'domainTokenCount', 'symCountDomain', ‘entropyPath', 'domainExtension','maxDigitSequencePath', 'pathToDomainRatio', 'domainDigitCount', 'pathTokenCount', 'pathLength', 'queryPathRatio'.

3.3. WHOIS Features-
These features contain information about the domain registrant.
For eg- urlCreationDate, urlExpirationDate, daysSinceRegistration, daysSinceExpiration.

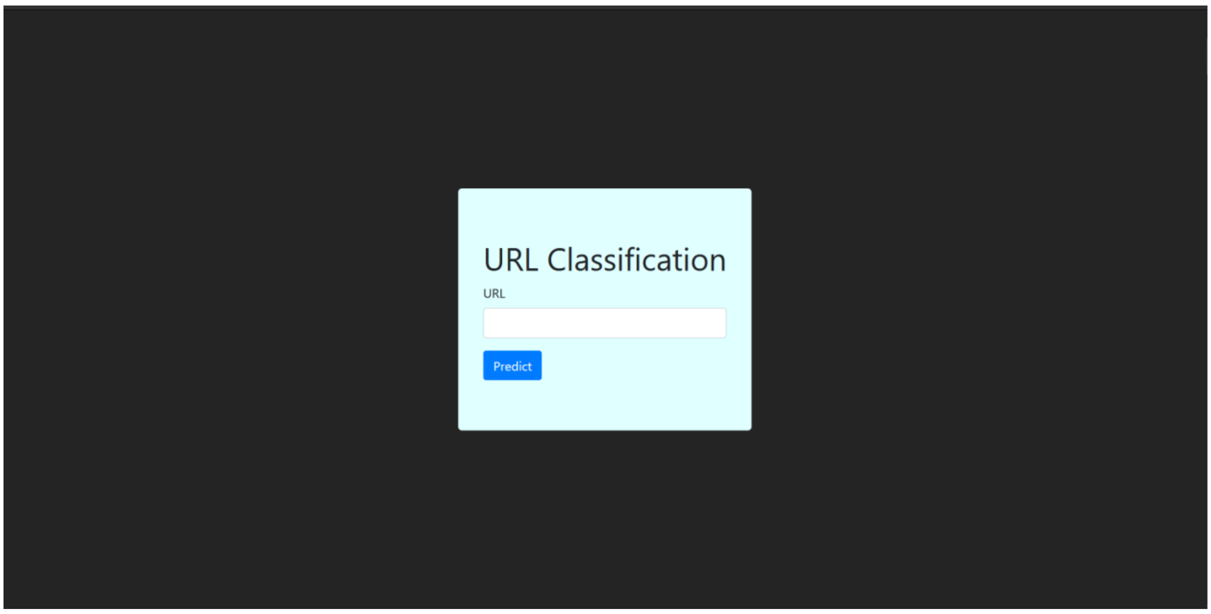
In total the application collects 23 features. These features are used by the application to classify URLs with reasonable accuracy. The application uses a delayed feature collection system to achieve improved performance in real time. On average the time taken to calculate the above feature sets are-

- Lexical Features I- 3 ms
- Lexical Features II - 155 ms
- Whois Features – 1453 ms

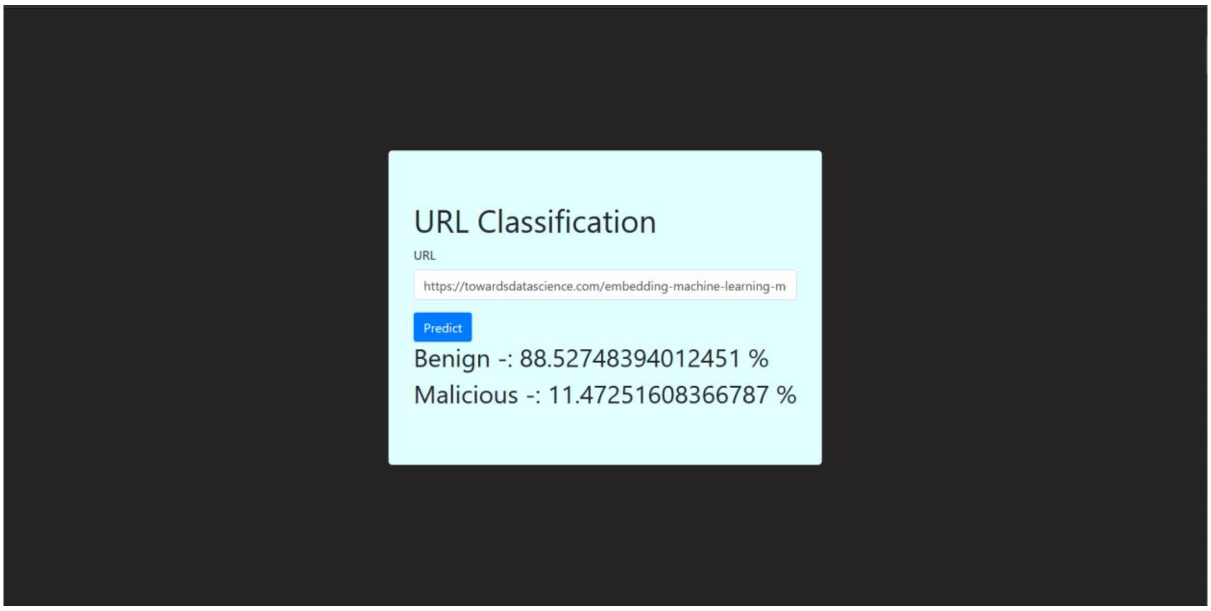
Thus, the application saves a great amount of time by delaying the collection of expensive features and trying to classify on the features which take less time for collection. If it is unable to do so then it extracts the next set of features.

4. Web Application-

In the backend, Flask is used to host the application on local host. In the front end, HTML and CSS is used. Bootstrap 4 is used along with CSS for styling the webpage. Jinja2 is used for modularity of webpages.



Index Page


URL Classification

URL

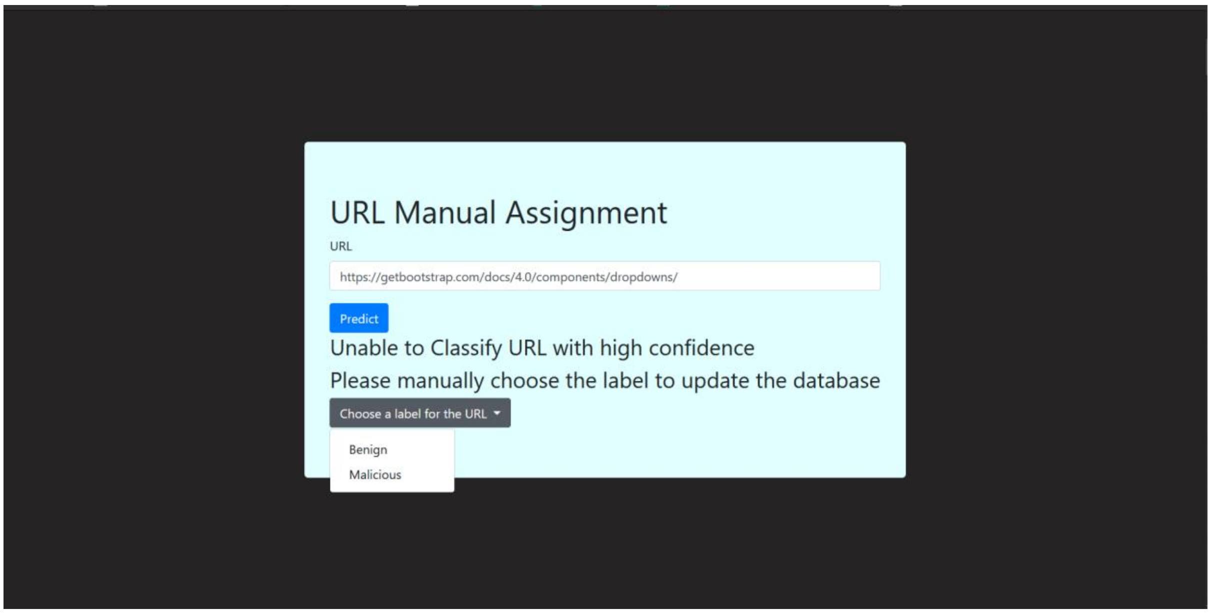
https://towardsdatascience.com/embedding-machine-learning-m

Benign -: 88.52748394012451 %

Malicious -: 11.47251608366787 %

Classification Probability Results

If the application fails to assign a label with high confidence then the below webpage appears where we can assign a label to the URL and store its features and label in the database.


URL Manual Assignment

URL

https://getbootstrap.com/docs/4.0/components/dropdowns/

Unable to Classify URL with high confidence

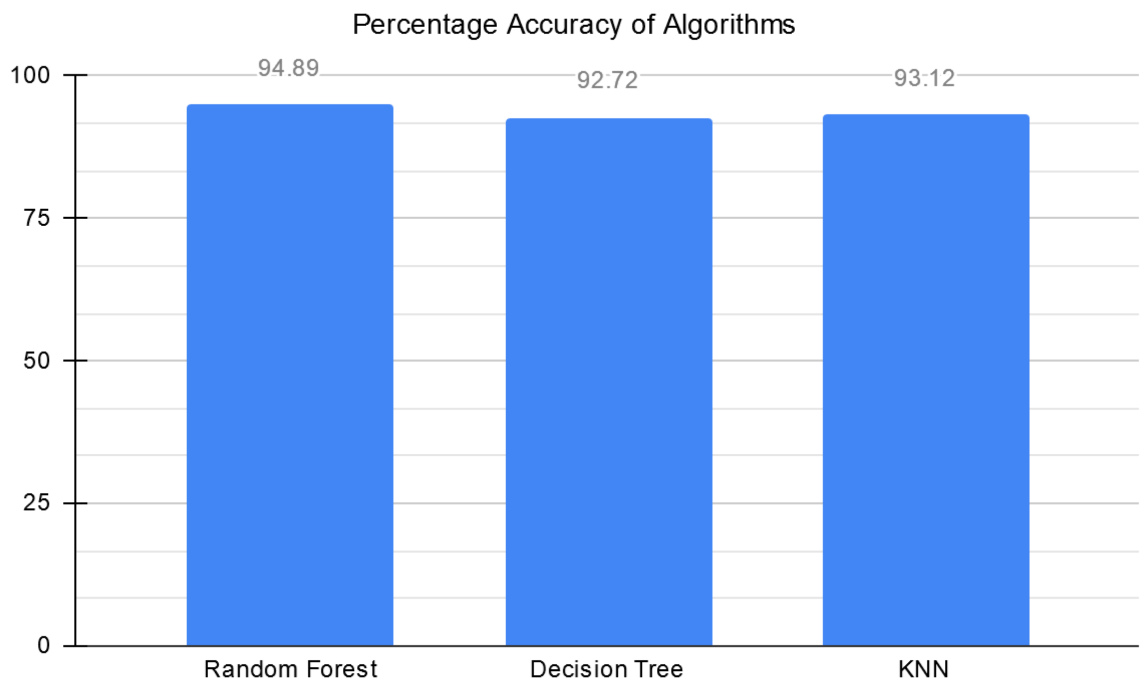
Please manually choose the label to update the database

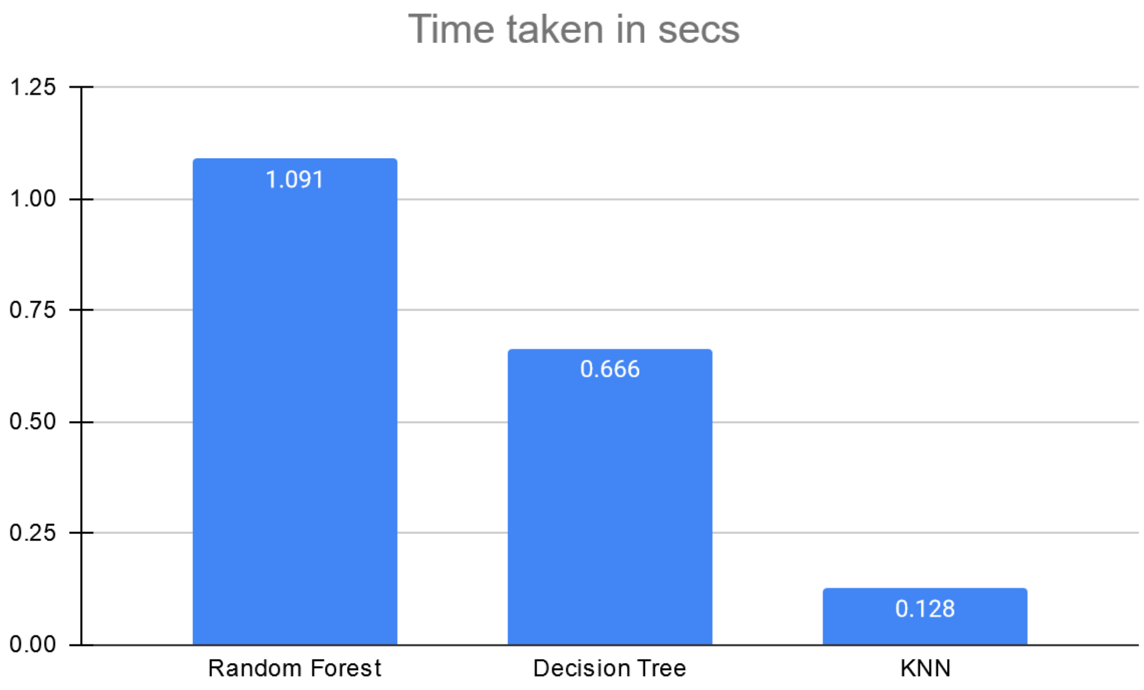
Assign a Label to URL manually

RESULTS

1. Comparison of Batch Learning Algorithms-

For this project, 3 batch learning models were compared- a) Random Forest Model b) K-Nearest Neighbours c) Decision Tree Model.



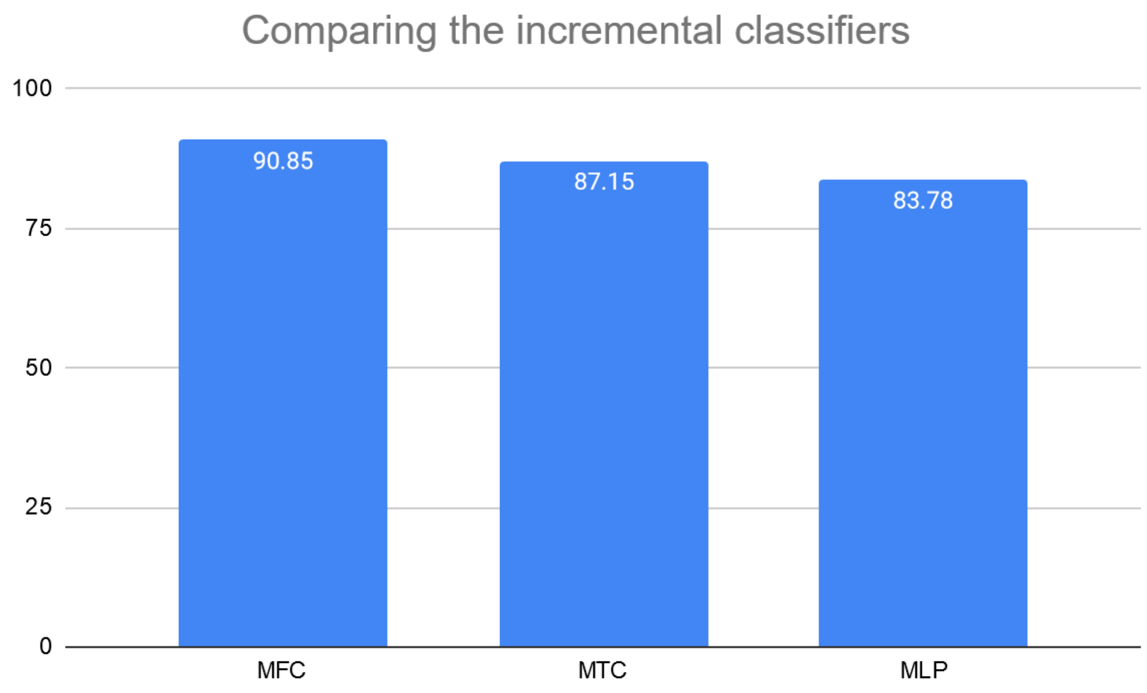


From the results we can see that Random Forest outperforms all other classifiers for this dataset. But the time taken for training and classification in case of Random Forest Model is the largest. K-Nearest Neighbours takes the least amount of time for training and classification. The time duration specified in the graph above may vary from machine to machine thus it should only be considered for relative comparison. Random Forest works best for the dataset as the dataset is sparse with many missing feature values. However, we cannot use the Random Forest Model for this project because of the unbounded growth of the dataset.

2. Comparison of Incremental Learning Algorithms-

As the Batch Learning Algorithms were not able to suffice the purpose of the project, Incremental Classifiers were used. The performances of 3 incremental classifiers are compared below-

1. Mondrian Forest Classifier (MFC) - Available in Scikit-Garden Library
2. Mondrian Tree Classifier (MTC) - Available in Scikit-Garden Library
3. Multi-Layer Perceptron Classifier (MLP) – Available in Sklearn.neural_network module.



From the above graph we observe that Mondrian Forest Classifier performs the best amongst all the 3 classifiers. It has comparable accuracy to batch learning classifiers but is remarkably faster than them. The results for time taken by each Incremental Learning Classifier is not shown as they will be trained per sample and not in batches.
Thus, for this project Mondrian Forest Classifier is chosen.

3. Feature Importance Analysis-

To test the feature importance, chi squared test was used. To perform the chi squared test following steps were followed-

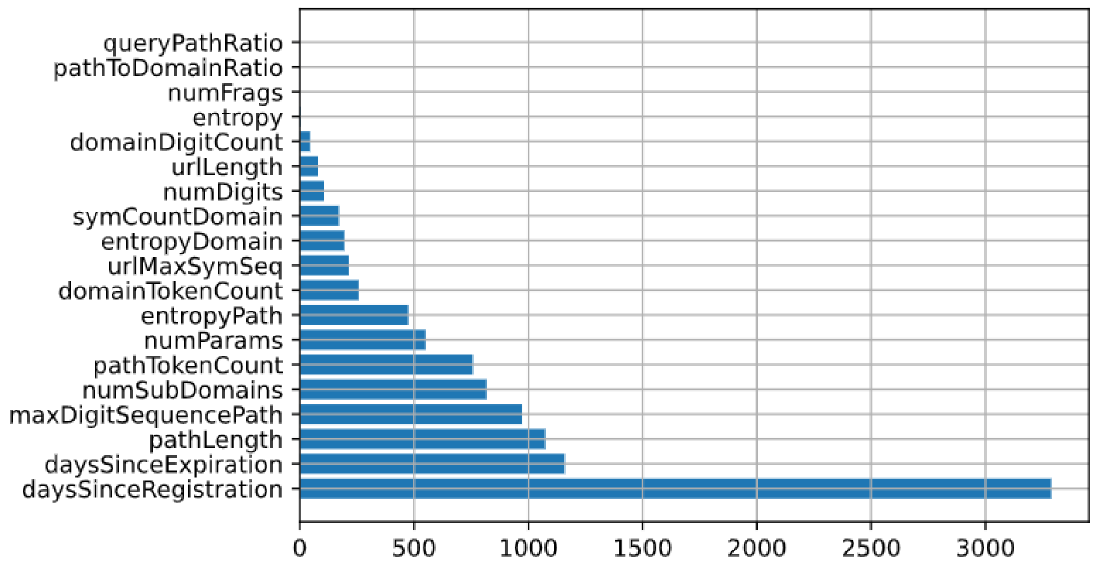
- Hypothesis definition:
  - Null Hypothesis (H0): Two variables are independent.
  - Alternate Hypothesis (H1): Two variables are not independent.

- Calculate the chi squared test statistic value: For this the chi2 module in sklearn.featureselection was used.
- Accept or reject the Null Hypothesis: For this we check whether the chi squared test statistic value falls in the accepting region or the rejecting region. We accept the null hypothesis if the chi squared test statistic value is less than the critical value else, we reject the null hypothesis. Thus, greater is the value of chi squared test statistic more is the probability of both the variables being independent.

Those features for which we are able to reject the Null hypothesis, we conclude that these features are dependent on the Target variable (URL Class Label).

The graph below is plot of Chi Sq. test statistic value vs top 19 most important features Feature. From the below graph we see that the most important features are-

1. daysSinceRegistration
2. daysSinceExpiration
3. pathlength
4. maxDigitSequencePath
5. numSubDomains



Plot of Feature vs Chi Sq. Value

4. Cross Validation-

For testing the classifier, 10-fold cross validation was performed using the Stratified Fold module of sklearn.model_selection library. The dataset consisted of over 1,10,000 URLs. The classifier achieved an average accuracy of 90.72 percent on this data.

CONCLUSION

In this work, an automated web application for detection of Malicious URLs was built. To deal with the unbounded growth of URL space an incremental learning algorithm was used. Delayed Feature Collection system was used to improve the performance of the system. The application was able to classify URLs as Benign or Malicious with 90.72 percent accuracy which is quite satisfactory.

Several methods maybe adopted to improve the accuracy like expanding the feature space to include n-gram, DNS query results, black list presence, bag of words, web page content, web page network traffic etc., improving the training dataset by collect more new labelled URLs to decrease the variance in classification scores and using methods like selective sampling algorithm for performance boost.

REFERENCES

1. J. Lewis, “Economic Impact of Cybercrime, No Slowing Down.” McAfee, 2018.
2. Symantec, “2016 Internet security threat report,” https://www.symantec.com/security-center/threat-report, 2016.
3. Le, Q. Pham, D. Sahoo, and S. C. Hoi, “Urlnet: learning a URL representation with deep learning for malicious URL detection,” arXiv preprint arXiv:1802.03162, 2018.
4. Y.-L. Zhang, L. Li, J. Zhou, X. Li, Y. Liu, Y. Zhang, and Z.-H.Zhou, “Poster: A pu learning based system for potential malicious URL detection,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 2599–2601.
5. R. Verma and A. Das, “What’s in a url: Fast feature extraction and malicious URL detection,” in Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics. ACM, 2017, pp. 55–63.
6. J. Jiang, J. Chen, K.-K. R. Choo, C. Liu, K. Liu, M. Yu, and Y. Wang, “A deep learning based online malicious URL and DNS detection scheme,” in International Conference on Security and Privacy in Communication Systems. Springer, 2017, pp. 438–448.
7. A. Astorino, A. Chiarello, M. Gaudioso, and A. Piccolo, “Malicious URL detection via spherical classification,” Neural Computing and Applications, vol. 28, no. 1, pp. 699–705, 2017
8. Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova and Ali A. Ghorbani, "Detecting Malicious URLs Using Lexical Analysis", Network and System Security, Springer International Publishing, P467--482, 2016.