Detection and categorization of Malicious URL’s

Url Prober Team
05 April 2022

Agenda

- Introduction
- Challenges
- Solutions
- Models and Results
- Conclusion and Future Study

Illustrations by Pixeltrue on icons8

Introduction

URLs are used as the main vehicle for online criminal activities. Security community developed techniques to counter the security issue by mostly blacklisting malicious URLs.

That approach works well but do not scale and trusted websites may also be compromised via defacement of URL.

This study will try to explore a lightweight approach to identify and classify malicious URL using machine learning.

Dataset

University of New Brunswick

Canadian Institute for Cybersecurity

URL dataset (ISCX-URL2016)

https://www.unb.ca/cic/datasets/url-2016.html

The five different types of malicious URLs

Benign URLs

Over 35,300 benign URLs were collected from Alexa top websites. The domains have been passed through a Heritrix web crawler to extract the URLs. Around half a million unique URLs are crawled initially and then passed to remove duplicate and domain only URLs. Later the extracted URLs have been checked through Virustotal to filter the benign URLs.

Spam URLs

Around 12,000 spam URLs were collected from the publicly available WEBSPAM-UK2007 dataset.

Phishing URLs

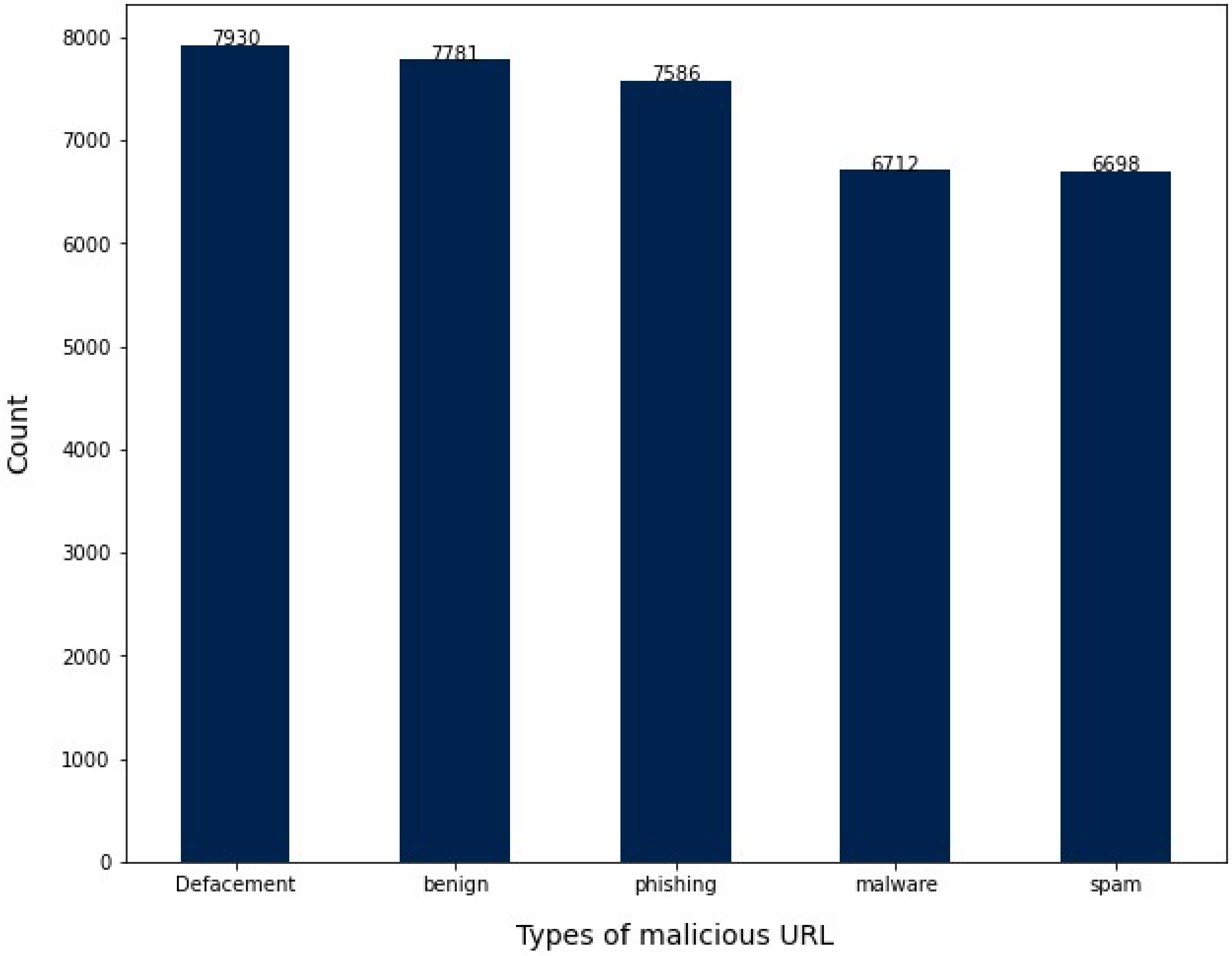
Around 10,000 phishing URLs were taken from OpenPhish which is a repository of active phishing sites.

Malware URLs

More than 11,500 URLs related to malware websites were obtained from DNS-BH which is a project that maintain list of malware sites.

Defacement URLs

More than 45,450 URLs belong to Defacement URL category. They are Alexa ranked trusted websites hosting fraudulent or hidden URL that contains both malicious web pages.



7930
7781
7586
6712
6698
8000
7000
6000
5000
4000
3000
2000
1000
0
Count
Defacement
benign
phishing
malware
spam
Types of malicious URL

Challenges

The dataset contains Null and NaN values.

Lack of description on individual features.

Difficult to determine the correlations because of large number of features.

Solutions

The dataset contains Null and NaN values.

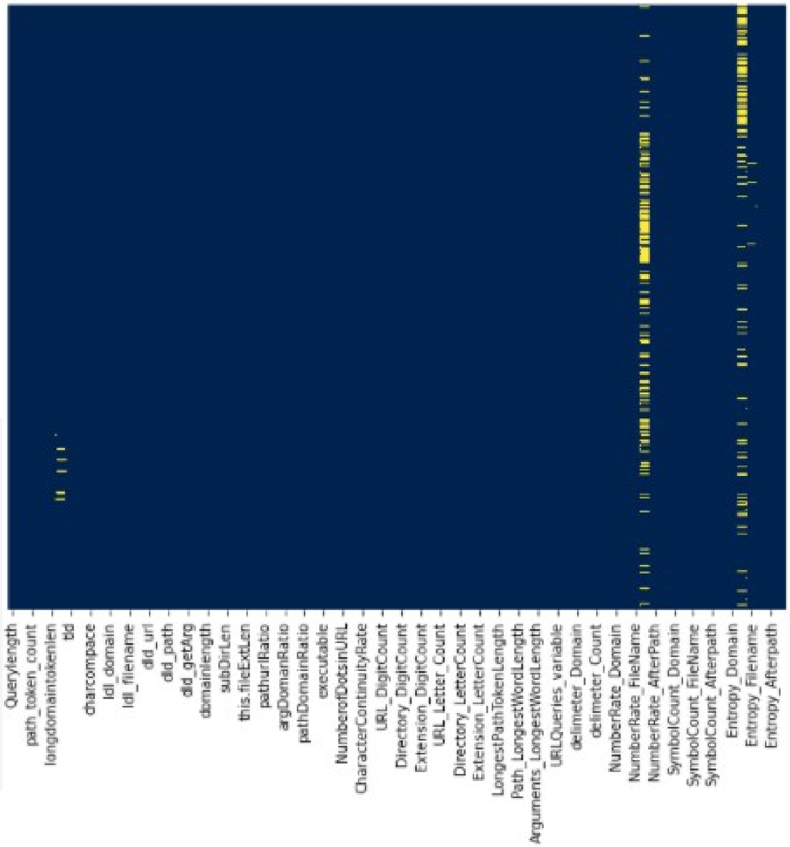
Replace with mean or drop the feature.

Lack of description on individual features.

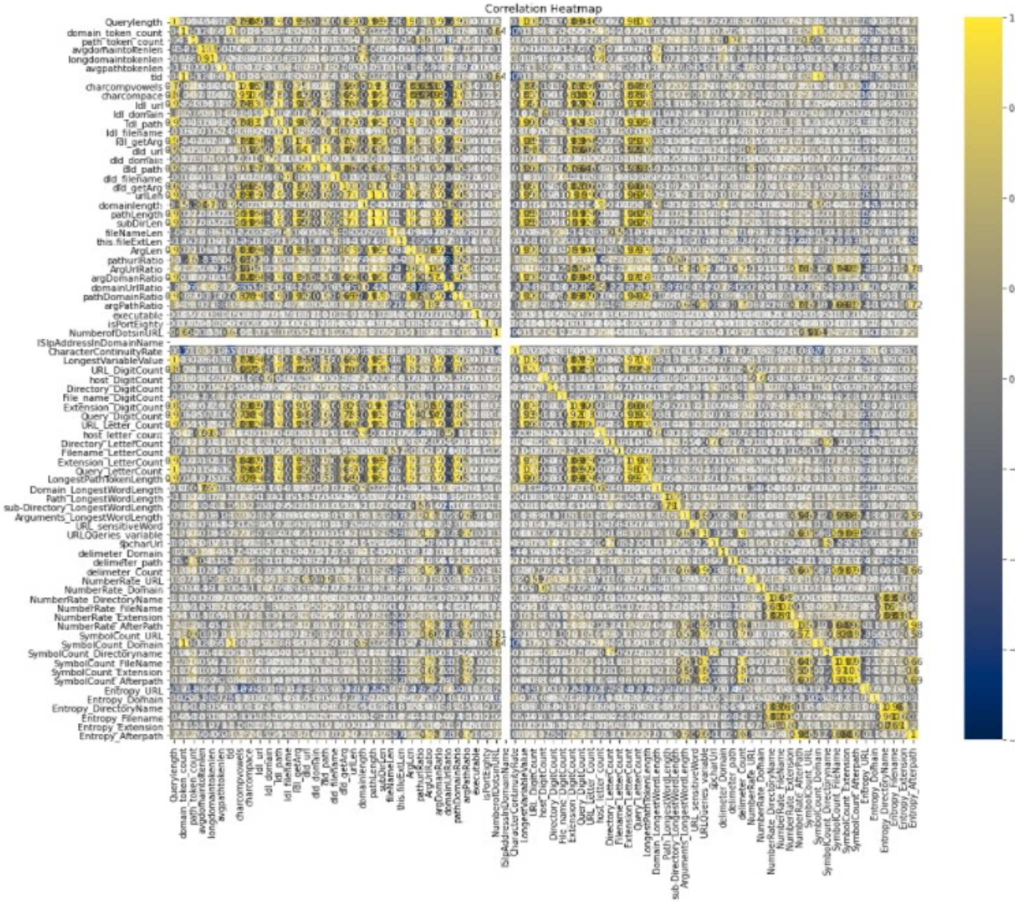
Difficult to determine the correlations because of large number of features.

Feature Selection

Challenges



Querylength
path_token_count
longdomaintokenlen
tld
charcompace
ldl_domain
ldl_filename
dld_url
dld_path
dld_getArg
domainlength
subDirLen
this.fileExtLen
pathurlRatio
argDomanRatio
pathDomainRatio
executable
NumberofDotsinURL
CharacterContinuityRate
URL_DigitCount
Directory_DigitCount
Extension_DigitCount
URL_Letter_Count
Directory_LetterCount
Extension_LetterCount
LongestPathTokenLength
Path_LongestWordLength
Arguments_LongestWordLength
URLQueries_variable
delimeter_Domain
delimeter_Count
NumberRate_Domain
NumberRate_FileName
NumberRate_AfterPath
SymbolCount_Domain
SymbolCount_FileName
SymbolCount_Afterpath
Entropy_Domain
Entropy_Filename
Entropy_Afterpath



Correlation Heatmap
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Data Cleaning and Preparation

```
help(loader.prepare_data)
```

```
Help on method prepare_data in module loader_nb:

prepare_data(data, fill_na=True, feature_selection=True, show_graph=False) method of loader_nb.UrlDatasetLoader instance
    (DataFrame, boolean, boolean) --> X and y of the dataframe.

    This function returns the X and y of the malicious url dataframe.

    Parameters
    ----------
    fill_na : True to fill the na records with mean values otherwise drop the features.

    feature_selection : True to remove one or more features that have a correlation higher than 0.9 othewise do not perform that type of feature selection.
                        https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf

    show_graph : True to display the graph after applying fill_na or feature_selection.
```

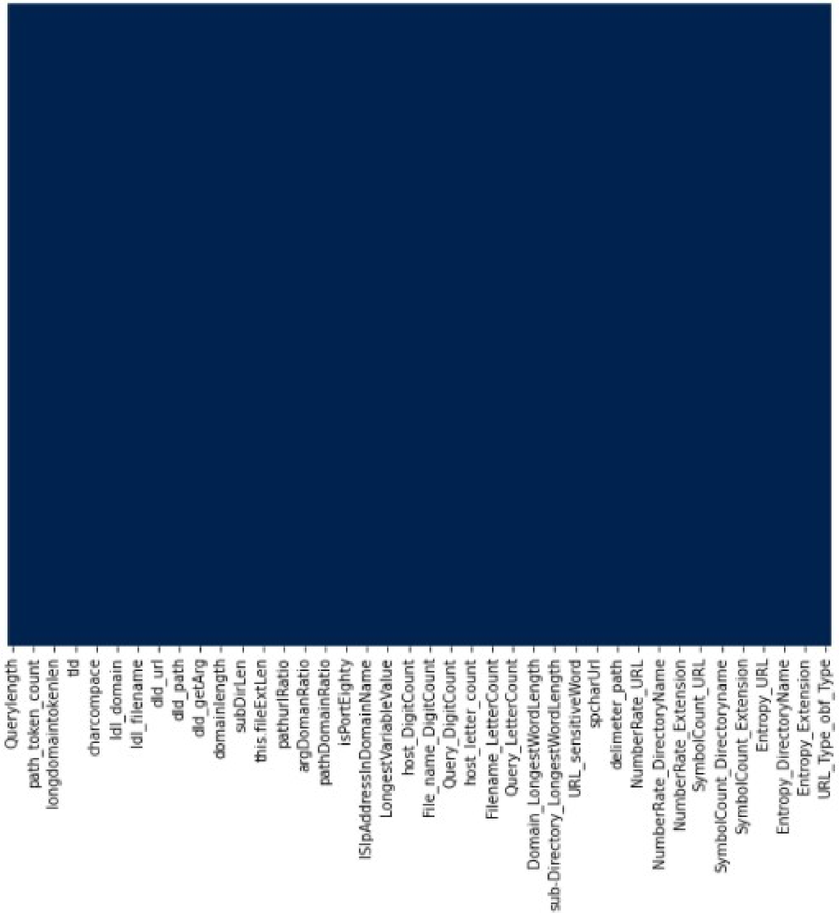
Data Cleaning and Preparation

```
help(loader.prepare_data)
```

```
Help on method prepare_data in module loader_nb:

prepare_data(data, fill_na=True, feature_selection=True, show_graph=False) method of loader_nb.UrlDatasetLoader instance
    (DataFrame, boolean, boolean) --> X and y of the dataframe.

    This function returns the X and y of the malicious url dataframe.

    Parameters
    ----------
    fill_na : True to fill the na records with mean values otherwise drop the features.

    feature_selection : True to remove one or more features that have a correlation higher than 0.9 othewise do not perform that type of feature selection.
                        https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf

    show_graph : True to display the graph after applying fill_na or feature_selection.
```

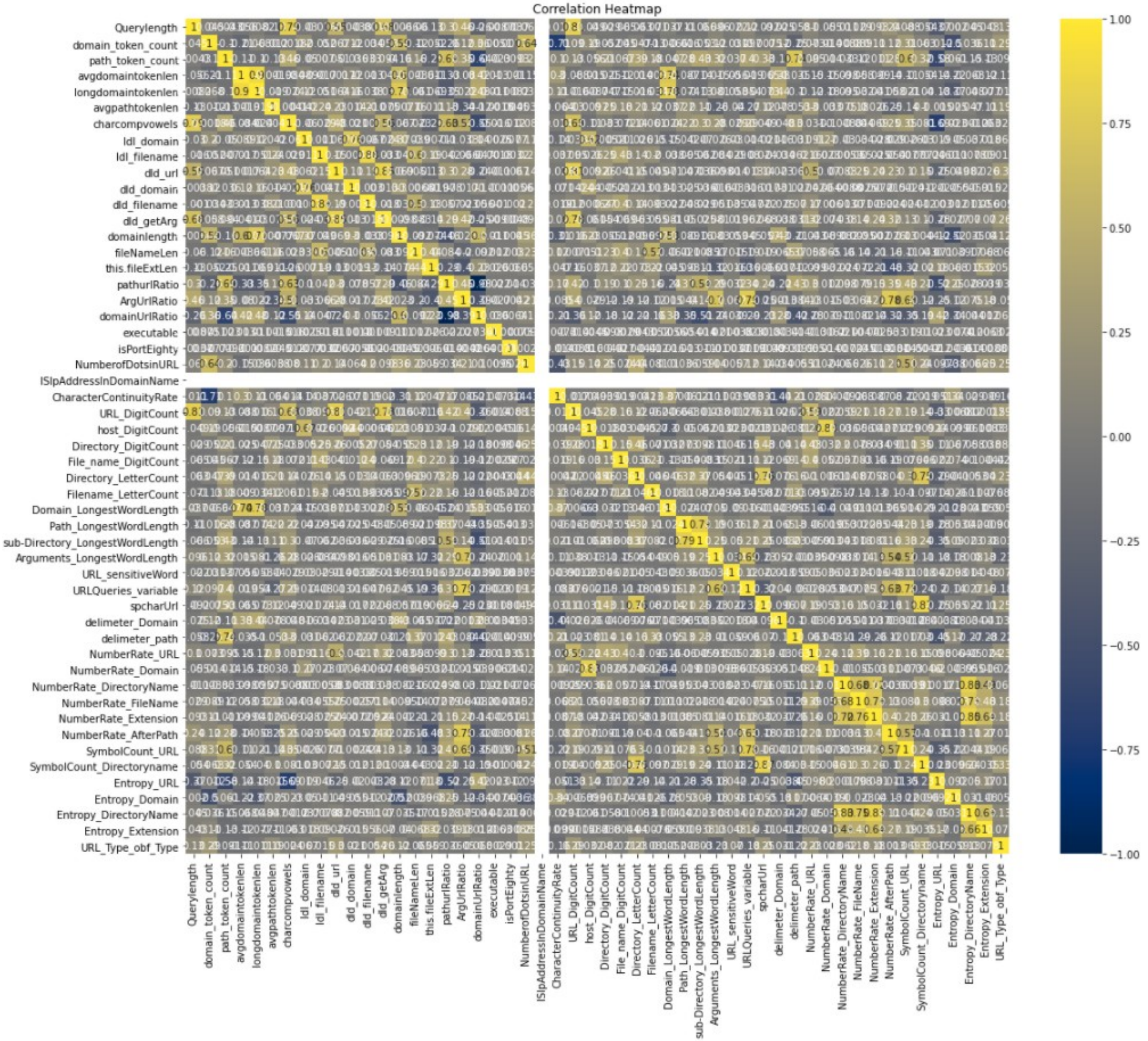
This study conducts various experients based on different combinations of fill_na and feature_selection when preparing the data.

Solutions



Querylength
path_token_count
longdomaintokenlen
tld
charcompace
ldl_domain
ldl_filename
dld_url
dld_path
dld_getArg
domainlength
subDirLen
this.fileExtLen
pathurlRatio
argDomanRatio
pathDomainRatio
isPortEighty
ISIpAddressInDomainName
LongestVariableValue
host_DigitCount
File_name_DigitCount
Query_DigitCount
host_letter_count
Filename_LetterCount
Query_LetterCount
Domain_LongestWordLength
sub-Directory_LongestWordLength
URL_sensitiveWord
spcharUrl
delimeter_path
NumberRate_URL
NumberRate_DirectoryName
NumberRate_Extension
SymbolCount_URL
SymbolCount_Directoryname
SymbolCount_Extension
Entropy_URL
Entropy_DirectoryName
Entropy_Extension
URL_Type_obf_Type

Solutions



Correlation Heatmap
Querylength
domain_token_count
path_token_count
avgdomaintokenlen
longdomaintokenlen
avgpathtokenlen
charcompvowels
ldl_domain
ldl_filename
dld_url
dld_domain
dld_filename
dld_getArg
domainlength
fileNameLen
this.fileExtLen
pathurlRatio
ArgUrlRatio
domainUrlRatio
executable
isPortEighty
NumberofDotsinURL
ISIpAddressInDomainName
CharacterContinuityRate
URL_DigitCount
host_DigitCount
Directory_DigitCount
File_name_DigitCount
Directory_LetterCount
Filename_LetterCount
Domain_LongestWordLength
Path_LongestWordLength
sub-Directory_LongestWordLength
Arguments_LongestWordLength
URL_sensitiveWord
URLQueries_variable
spcharUrl
delimeter_Domain
delimeter_path
NumberRate_URL
NumberRate_Domain
NumberRate_DirectoryName
NumberRate_FileName
NumberRate_Extension
NumberRate_AfterPath
SymbolCount_URL
SymbolCount_Directoryname
Entropy_URL
Entropy_Domain
Entropy_DirectoryName
Entropy_Extension
URL_Type_obf_Type
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Challenge

Finding Outliers?

Solution

Isolation Forest

Unsupervised Anomaly Detection

Isolation Forest

Isolation:

The term isolation means 'separating an instance from the rest of the instances'. Since anomalies are 'few and different' and therefore they are more susceptible to isolation.

Isolation Forest

Isolation:

The term isolation means 'separating an instance from the rest of the instances'. Since anomalies are 'few and different' and therefore they are more susceptible to isolation.

Benefits:

Unsupervised algorithm and therefore it does not need labels to identify outlier/anomaly.

Computationally efficient and low memory requirement.

Isolation Forest

Isolation:

The term isolation means 'separating an instance from the rest of the instances'. Since anomalies are 'few and different' and therefore they are more susceptible to isolation.

Benefits:

Unsupervised algorithm and therefore it does not need labels to identify outlier/anomaly.

Computationally efficient and low memory requirement.

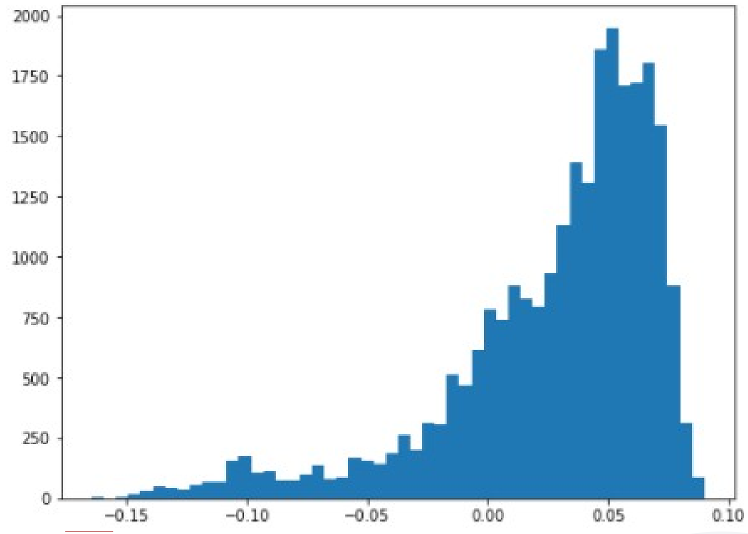
Drawbacks:

Requires to pick the percentage of anomalies in the dataset hence we need to have at least an idea of the proportion of anomalies in our data.

Isolation Forest

Isolation:

The term isolation means 'separating an instance from the rest of the instances'. Since anomalies are 'few and different' and therefore they are more susceptible to isolation.



Benefits:

Unsupervised algorithm and therefore it does not need labels to identify outlier/anomaly.

Computationally efficient and low memory requirement.

Drawbacks:

Requires to pick the percentage of anomalies in the dataset hence we need to have at least an idea of the proportion of anomalies in our data.

https://medium.com/@y.s.yoon/isolation-forest-anomaly-detection-identify-outliers-101123a9ff63
https://practicaldatascience.co.uk/machine-learning/how-to-use-the-isolation-forest-model-for-outlier-detection

Isolation Forest

```
help(loader.perform_anomaly_detection)
```

```
Help on method perform_anomaly_detection in module loader_nb:

perform_anomaly_detection(X, y) method of loader_nb.UrlDatasetLoader instance
    (X, y) --> X, y

    This function perform unsupervised anomaly detection using Isolation Forest.
    https://practicaldatascience.co.uk/machine-learning/how-to-use-the-isolation-forest-model-for-outlier-detection
```

```
help(loader.train_test_split)
```

```
Help on method train_test_split in module loader_nb:

train_test_split(X, y, test_size, random_state, anomaly_detection=True) method of loader_nb.UrlDatasetLoader instance
    This is a convenience method to train test split and have an option to perform anomaly detection or not after the split.

    Read more in sklearn.model_selection.train_test_split

    Parameters
    ----------
    anomaly_detection: True to perform unsupervised anomaly detection using Isolation Forest.
```

```
X_train, X_test, y_train, y_test = loader.train_test_split(X, y, test_size=TEST_SIZE, random_state=RANDOM_STATE)
```

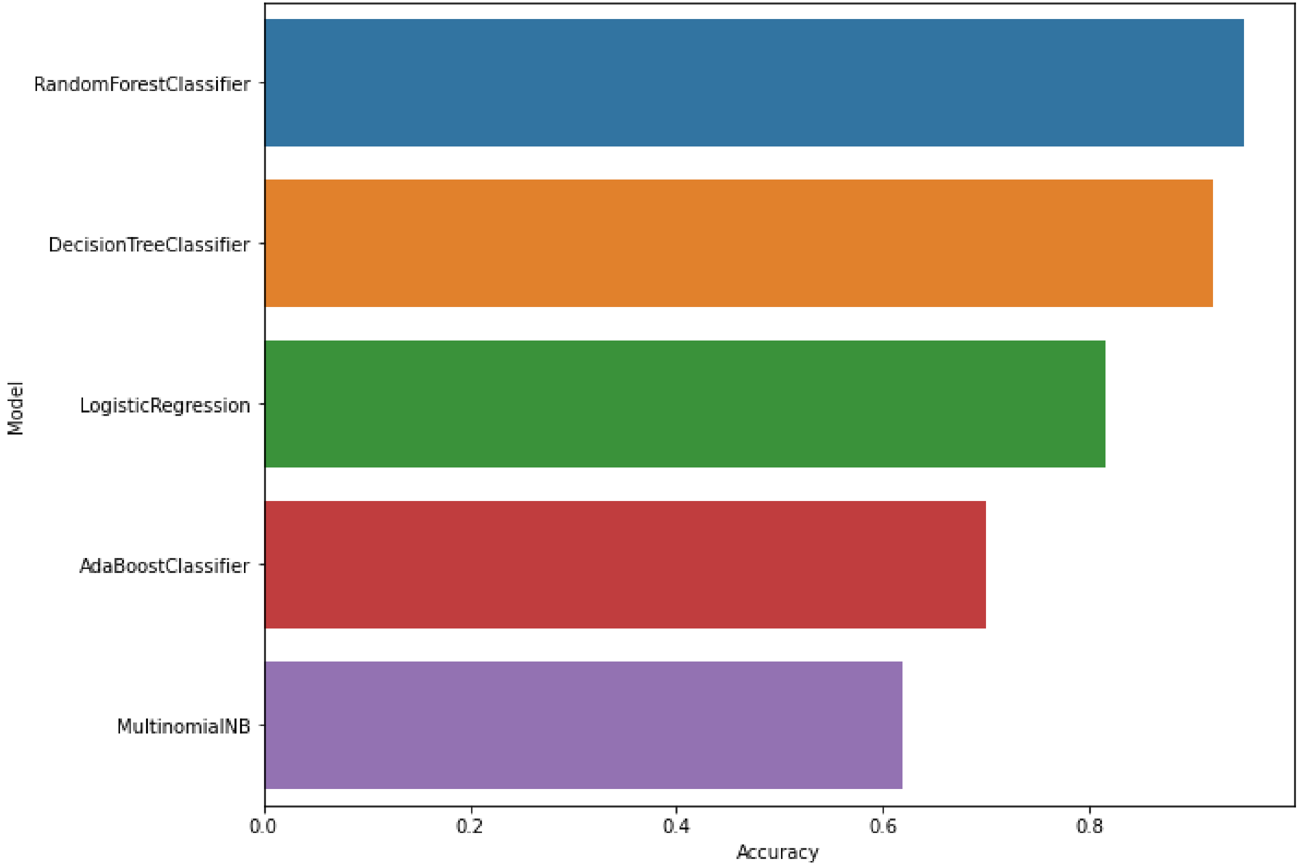
```
The X_train, y_train shape:
(25694, 51)
(25694,)
The shape after unsupervised anomaly detection:
(25437, 51)
(25437,)
```

Model Selection

Using GridSearchCV with five folds we built five models

- Random Forest
- Decision Tree
- Logistic Regression
- AdaBoost
- Naive Bayes

Results



	Model	Accuracy	F1-score
0	RandomForestClassifier	0.950376	0.950739
1	DecisionTreeClassifier	0.919464	0.920220
2	LogisticRegression	0.816639	0.815096
3	AdaBoostClassifier	0.698954	0.691719
4	MultinomialNB	0.618694	0.610399

Conclusion

This study concludes that **Random Forest** is the best model to use to build a URL filter application using machine learning.

Future Study

The focused of the study is the used of supervised learning algorithms. Future work could look on semi-supervised classification algorithms and new developing supervised algorithms.

Photo by Dave Hoefler on Unsplash

Thank you

Website:

https://quickheaven.github.io/scs-3253-machine-learning/

Github:

https://github.com/quickheaven/scs-3253-machine-learning

Team members

Arjie Cristobal

Omair Amjad