VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Project Report

Project Title: Phishing website detection

Course Name : Information and System Security

Course Code : SWE3002

Faculty : Prof. Mangayarkarasi R

SLOT : E2/TE2

Submitted By:

M.Jaswanth Kumar – 19MIS0364

B.Rohith – 19MIS0370

N.Navadeep – 19MIS0387

Review-1

Problem Definition:

Phishing is a type of social engineering attack often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.

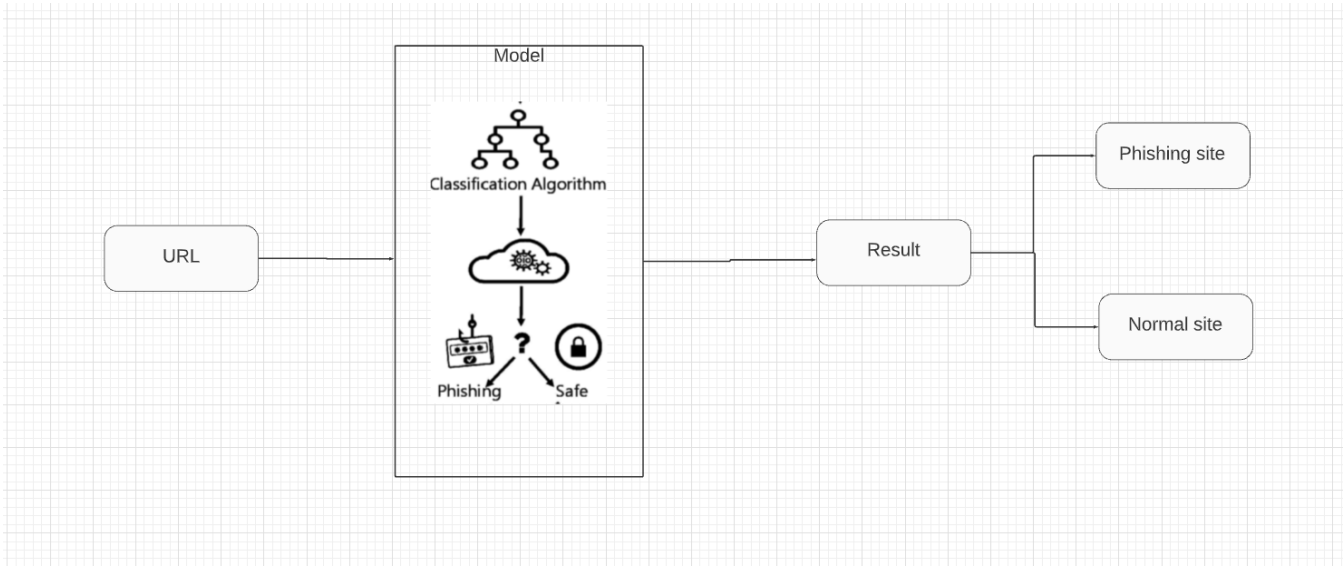
Phishing attack is a simplest way to obtain sensitive information from innocent users. Aim of the phishers is to acquire critical information like username, password and bank account details. Cyber security persons are now looking for trustworthy and steady detection techniques for phishing websites detection. This paper deals with machine learning technology for detection of phishing URLs by extracting and analyzing various features of legitimate and phishing URLs. Decision Tree, random forest and Support vector machine algorithms are used to detect phishing websites. Aim of the paper is to detect phishing URLs as well as narrow down to best machine learning algorithm by comparing accuracy rate, false positive and false negative rate of each algorithm.

Introduction:

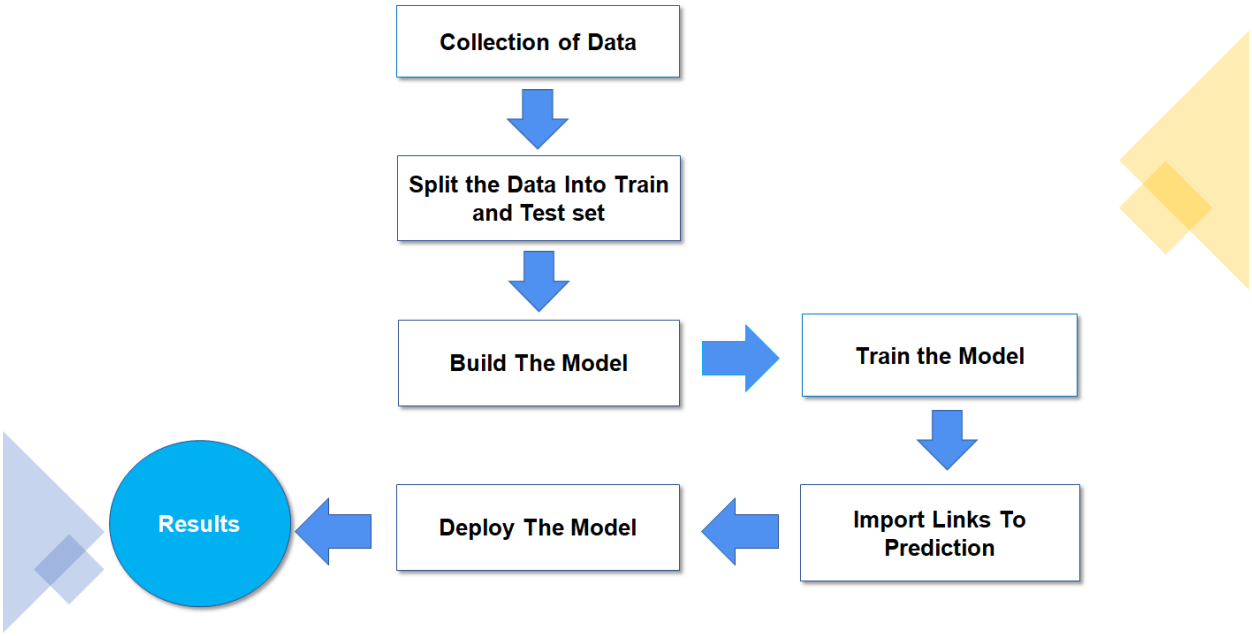
Nowadays Phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. Experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account credentials. In United States businesses, there is a loss of US $10billion per year because their clients become victim to phishing . In 3rd Microsoft Computing Safer Index Report released in February 2020, it was estimated that the annual worldwide impact of phishing could be as high as $5 billion. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them but it is very important to enhance phishing detection techniques.

Complete Design:

Proposed Approach:-



Detailed Work Flow Design:



Literature Survey:

Protecting user against phishing using Anti-phishing: -

Anti-Phish is used to avoid users from using fraudulent web sites which in turn may lead to phishing attack. Here, Anti-Phish traces the sensitive information to be filled by the user and alerts the user whenever he/she is attempting to share his/her information to a untrusted web site. The much effective elucidation for this is cultivating the users to approach only for trusted websites. However, this approach is unrealistic. Anyhow, the user may get tricked. Hence, it becomes mandatory for the associates to present such explanations to overcome the problem of phishing. Widely accepted alternatives are based on the creepy websites for the identification of “clones” and maintenance of records of phishing websites which are in hit list.

Learning to Detect Phishing Emails:-

An alternative for detecting these attacks is a relevant process of reliability of machine on a trait intended for the reflection of the besieged deception of user by means of electronic communication. This approach can be used in the detection of phishing websites, or the text messages sent through emails that are used for trapping the victims. Approximately, 800 phishing mails and 7,000 non-phishing mails are traced till date and are detected accurately over 95% of them along with the categorization on the basis of 0.09% of the genuine emails. We can just wrap up with the methods for identifying the deception, along with the progressing nature of attacks.

Phishing detection system for e-banking using fuzzy data mining: -

Phishing websites, mainly used for e-banking services, are very complex and dynamic to be identified and classified. Due to the involvement of various ambiguities in the detection, certain crucial data mining techniques may prove an effective means in keeping the e- commerce websites safe since it deals with considering various quality factors rather than exact values. In this paper, an effective approach to overcome the “fuzziness” in the e-banking phishing website assessment is used an intelligent resilient and effective model for detecting e-banking phishing websites is put forth. The applied model is based on fuzzy logics along with data mining algorithms to consider various effective factors of the e-banking phishing website.

Collaborative Detection of Fast Flux Phishing Domains:-

Here, two approaches are defined to find correlation of evidences from multiple servers of DNS and multiple suspects of FF domain. Real life examples can be used to prove that our correlation approaches expedite the detection of the FF domain, which are based on an analytical model which can quantify various DNS queries that are required to verify a FF domain. It also shows implementation of correlation schemes on a huge level by using a distributed model, that is more scalable as compared to a centralized one, is publish N subscribe correlation model known as LARSID. In deduction, it is quite difficult to detect the FF domains in a accurate and timely manner, as the screen of proxies is used to shield the FF Mother ship. A theoretical approach is used to analyze the problem of FF detection by calculating the number of DNS queries required to get back a certain amount of unique IP addresses.

A Prior-based Transfer Learning Method for the Phishing Detection: -

A logistic regression is the root of a priority based transferrable learning method, which is presented here for our classifier of statistical machine learning. It is used for the detection of the phishing websites depending on our selected characteristics of the URLs. Due to the divergence in the allocation of the features in the distinct phishing areas, multiple models are proposed for different regions. It is almost impractical to gather enough data from a new area to restore the detection model and use the transfer learning algorithm for adjusting the existing model. An appropriate way for phishing detection is to use our URLbased method.

Phishing Website Detection using ML:

The purpose of this project is to design an intelligent system for detecting phishing websites. Phishing is one of the social attack which aims in stealing sensitive information of the users such as login credentials, credit card numbers etc. Here we have collected phishing dataset from phish Tanks as well as from phishing sites and are compared with the algorithms which classifies the phishing dataset into phishing or legitimate. We propose a web application for detection. The algorithm used is random forest in order to get better performance and accuracy. This system uses a database in order to store phishing websites which are already tested and can be used as blacklist, which makes the classification even faster, as it reduces repetition.

The paper resolves all the drawbacks of the existing system. This project employs machine learning technique for predic- tion task and supervised learning algorithm namely random forest technique for exploring results. Supervised learning is a type of machine learning technique in

which labelled training data are used to train the machine and on the basis of that data it predicts the output. Random forest is used for classification and is a machine learning algorithm that belongs to supervised learning technique. For the successful detection of phishing site, it should detect in less time and with high accuracy.

Phishing Detection: by Mahmoud Khonji Khalifa University, UAE:

Phishing is a social engineering attack that aims at exploiting the weakness found in system processes as caused by system users. For example, a system can be technically secure enough against password theft, however unaware end users may leak their passwords if an attacker asked them to update their passwords via a given Hypertext Transfer Protocol (HTTP) link, which ultimately threatens the overall security of the system.

This article surveys the literature on the detection of phishing attacks. Phishing attacks target vulnerabilities that exist in systems due to the human factor. Many cyber attacks are spread via mechanisms that exploit weaknesses found in end-users, which makes users the weakest element in the security chain. The phishing problem is broad and no single silver-bullet solution exists to mitigate all the vulnerabilities effectively, thus multiple techniques are often implemented to mitigate specific attacks. This paper aims at surveying many of the recently proposed phishing mitigation techniques. A high-level overview of various categories of phishing mitigation techniques is also presented, such as: detection, offensive defense, correction, and prevention, which we belief is critical to present where the phishing detection techniques fit in the overall mitigation process.

Using Case-Based Reasoning for Phishing Detection:

Predicting Phishing attacks remains a remarkable challenge. Our CBR-PDS system is proposed to predict phishing attacks with a high accuracy. It is scalable to different data set sizes and can adapt proactively. We used a small data set of 572 cases for both legitimate and malicious URLs. The features are extracted and weighted then formulated as cases to be used in the prediction process. CBR-PDS shows a high accuracy with and without Online Phishing Threats (OPT). This accuracy ranges from 95.62% up to 98.07% as explained in table 3. CBR-PDS combines between offline and online phishing detection that can easily predict zero-hour phishing attacks. Future work will be concentrated on implementing an integrated web-based CBR-PDS system.

Phishing detection system by HTML tags:

Cui, Qian, et al [19] They have purposed a method of counting HTML tags used in DOM of attacks. In this method they used the concept of clustering and made clusters of attacks

happening in a specific range of distance, they proposed that, these clusters can be combined and made publicly open. Their results showed that, a very large amount of newly phishing attacks can be caught by this method.

A Systematic Literature Review on Phishing and Anti-Phishing Techniques:

Phishing is the number one threat in the world of internet. Phishing attacks are from decades and with each passing year it is becoming a major problem for internet users as attackers are coming with unique and creative ideas to breach the security. In this paper, different types of phishing and anti-phishing techniques are presented. For this purpose, the Systematic Literature Review(SLR) approach is followed to critically define the proposed research questions. At first 80 articles were extracted from different repositories. These articles were then filtered out using Tollgate Approach to find out different types of phishing and anti- phishing techniques. Research study evaluated that spear phishing, Email Spoofing, Email Manipulation and phone phishing are the most commonly used phishing techniques. On the other hand, according to the SLR, machine learning approaches have the highest accuracy of preventing and detecting phishing attacks among all other anti-phishing approaches

When Kiren et al. presented a review on different types of phishing attacks and detection techniques. Also they presented some mitigation techniques of phishing. The paper proposed that 100% accuracy to detect phishing can be made possible by using machine learning approach among all other anti-phishing approaches. Rana et al. presented a review and comprehensive examination of the modern and state of the art phishing attack techniques to spread awareness of phishing techniques among the reader and to educate them about different types of attacks. The author proposed this paper to encourage the use of anti- phishing methods as well.

Ref.	Author	Contribution Summary	Algorithms
[4]	Chandra sekaran	Structural features	Support vector machine(SVM) classifiers

[5]	Ganger	Training smart screen	Bayesian statistics
[6]	BazarhaniGilani	Semantic antology concept	Semantic antology concept by TFM method information gain
[7]	Chandra and Chinchani	Phoney	Phoney mimicking user response
[8]	Fette , Sadeh	Pilfers prototypes	Random forest and SVM
[9]	Bergholz	Statistical filtering	Dynamic markovchain
[10]	Ma,Ofoghi	Robust classifier model	Information gain algorithm and decision tree
[2]	shreyagopal	XGBoost Classifier	Multilayer Perceptrons (MLPs): Deep Learning And Autoencoder Neural Network
[1]	chamanthmvs	Structural features	DecisionTree Classifier
[3]	Abhishek Dobhal	PHISHCOOP	Logistic Regression And Random Forest Classification

S/W tools used:

Tools used for the development are as follows:

- Jupyter Notebook - (Anaconda Navigator)

- ❖ Python

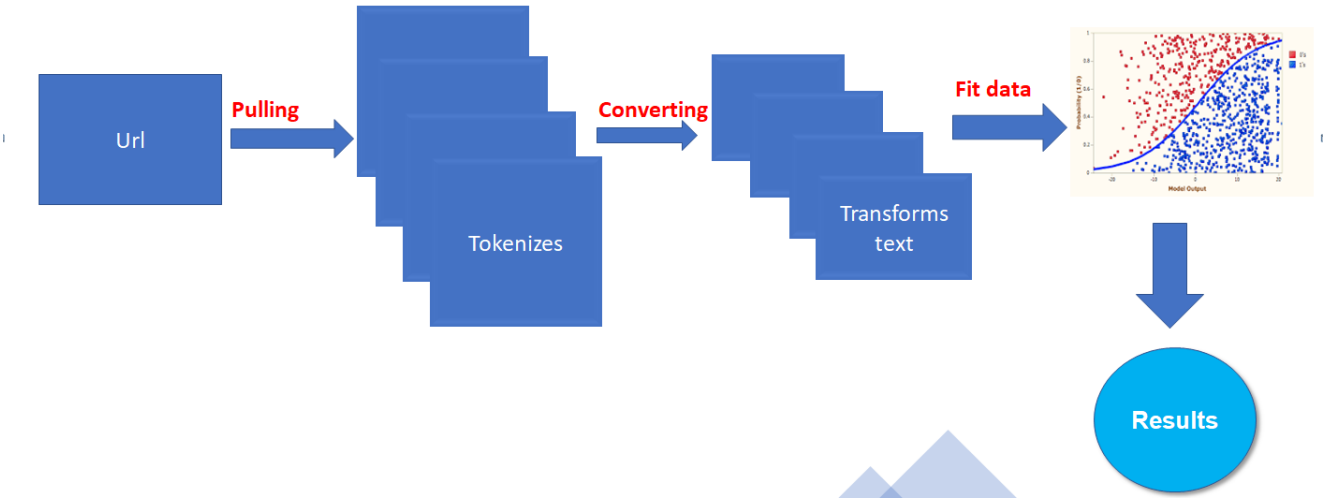
DATASET (name):

- Name of our dataset is phishing_site_urls.csv

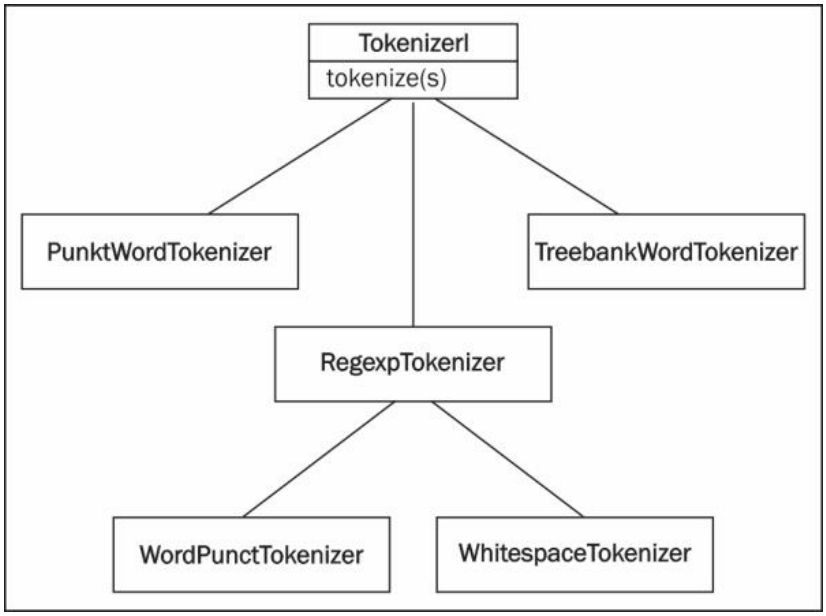
Review-2

Segmented Architecture diagram with mathematical/algorithmic/ Technical Description :

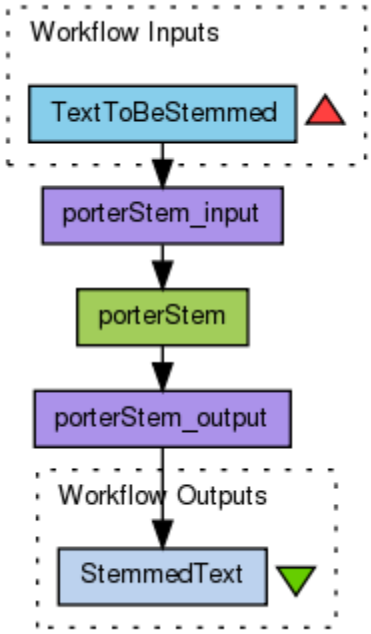
Project Overview
(How it actually Works)



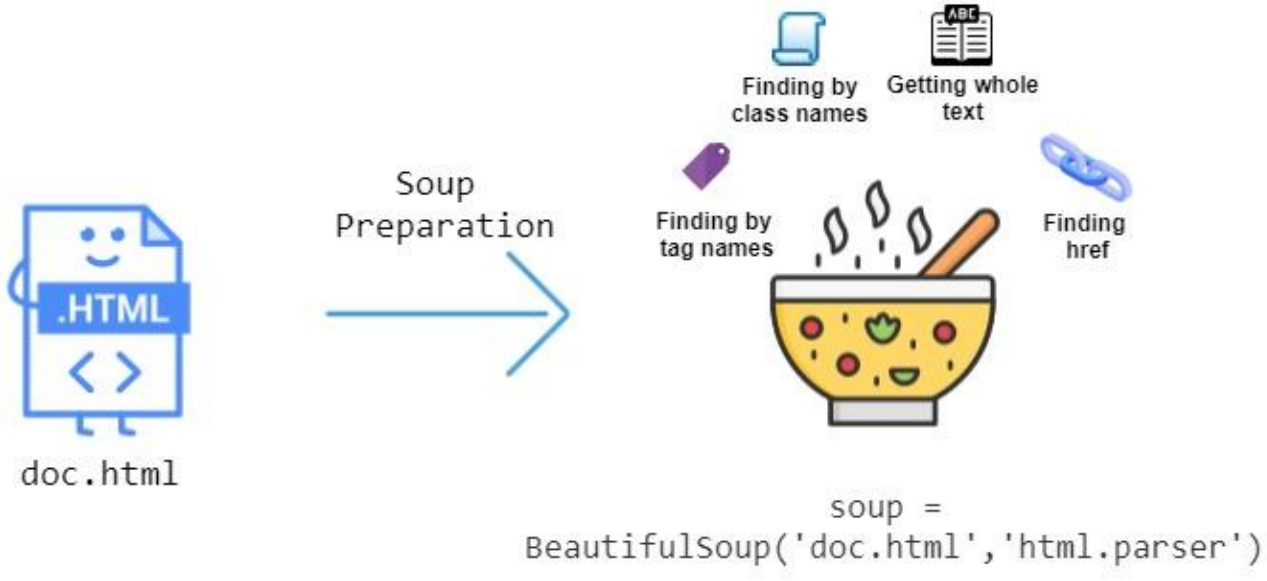
RegexpTokenizer:



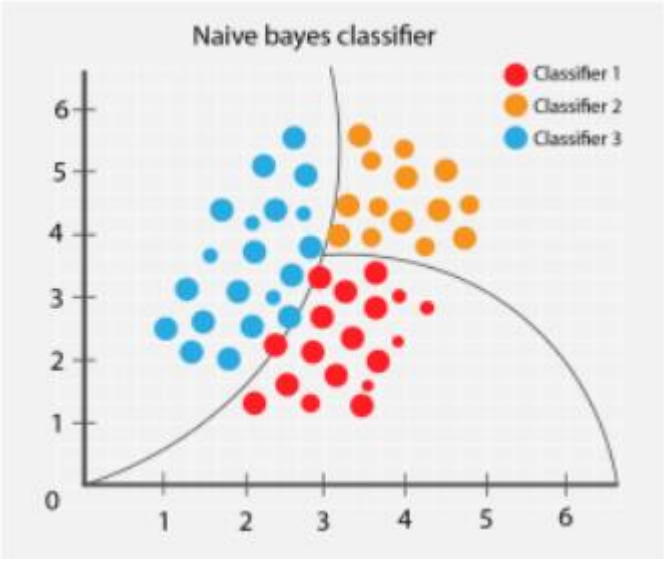
SnowballStemmer:



BeautifulSoup:



MultinomialNB:



$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

Complete technical description about your project:

RegexpTokenizer:

- A tokenizer that splits a string using a regular expression, which matches either the tokens or the separators between tokens.

```
In [13]: tokenizer = RegexpTokenizer(r'[A-Za-z]+')

In [14]: phish_data.URL[0]

Out[14]: 'nobell.it/70ffb52d079109dca5664cce6f317373782/login.SkyPe.com/en/cgi-bin/verification/login/70ffb52d079109dca5664cce6f317373/index.php?cmd=_profile-ach
&outdated_page_tmpl=p/gen/failed-to-load&nav=0.5.1&login_access=1322408526'

In [15]: # this will be pull letter which matches to expression
         tokenizer.tokenize(phish_data.URL[0]) # using first row

Out[15]: ['nobell',
          'it',
          'ffb',
          'd',
          'dca',
          'cce',
          'f',
          'login',
          'SkyPe',
          'com',
          'en',
          'cgi',
          'bin',
          'verification',
          'login',
          'ffb',
```

SnowballStemmer:

- Snowball is a small string processing language, gives root words
- It is a stemming algorithm which is also known as the Porter2 stemming algorithm as it is a better version of the Porter Stemmer since some issues of it were fixed in this stemmer.
- The Snowball compiler translates a Snowball program into source code in another language - currently Ada, ISO C, C#, Go, Java, Javascript, Object Pascal, Python and Rust are supported.

```
In [18]: stemmer = SnowballStemmer("english") # choose a language

In [19]: print('Getting words stemmed ...')
         t0= time.perf_counter()
         phish_data['text_stemmed'] = phish_data['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
         t1= time.perf_counter() - t0
         print('Time taken',t1 ,'sec')

         Getting words stemmed ...
         Time taken 178.71395260000008 sec
```

Visualization:

- Visualize some important keys using word cloud
- create a function to visualize the important keys from url

```
In [23]:
#sliceing classes
bad_sites = phish_data[phish_data.Label == 'bad']
good_sites = phish_data[phish_data.Label == 'good']
```

```
In [24]:
bad_sites.head()
```

Out[24]:

	URL	Label	text_tokenized	text_stemmed	text_sent
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad	[nobell, it, ffb, d, dca, cce, f, login, SkyPe...	[nobel, it, ffb, d, dca, cce, f, login, skype,...	nobel it ffb d dca cce f login skype com en cg...
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	www dghjdgf com paypal co uk cycgi bin webscrc...
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad	[serviciosbys, com, paypal, cgi, bin, get, int...	[serviciosbi, com, paypal, cgi, bin, get, into...	serviciosbi com paypal cgi bin get into herf s...
3	mail.printakid.com/www.online.americanexpress....	bad	[mail, printakid, com, www, online, americanex...	[mail, printakid, com, www, onlin, americanexp...	mail printakid com www onlin americanexpress c...
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad	[thewhiskeydregs, com, wp, content, themes, wi...	[thewhiskeydreg, com, wp, content, theme, wide...	thewhiskeydreg com wp content theme widescreen...

```
In [25]:
good_sites.head()
```

Out[25]:

	URL	Label	text_tokenized	text_stemmed	text_sent
18231	esxcc.com/js/index.htm?us.battle.net/noghn/en/...	good	[esxcc, com, js, index, htm, us, battle, net, ...	[esxcc, com, js, index, htm, us, battl, net, n...	esxcc com js index htm us battl net noghn en r...
18232	wwweira¯&nvinip¿ncH¯wVó%ÆåyDaHðú/ÏyEúuÊ\nÓ6...	good	[www, eira, nvinip, ncH, wV, yDaH, yE, u, rT, ...	[www, eira, nvinip, nch, wv, ydah, ye, u, rt, ...	www eira nvinip nch wv ydah ye u rt u g m i xz...
18233	'www.institutocgr.coo/web/media/syqvem/dk-óij...	good	[www, institutocgr, coo, web, media, syqvem, d...	[www, institutocgr, coo, web, media, syqvem, d...	www institutocgr coo web media syqvem dk ij r ...

Chrome webdriver:

- WebDriver tool use for automated testing of webapps across many browsers. It provides capabilities for navigating to web pages, user input and more.

```
In [7]:
browser = webdriver.Chrome(r"chromedriver.exe")
```

```
In [8]:
list_urls = ['https://www.ezeephones.com/','https://www.ezeephones.com/about-us'] #here i take phishing sites
links_with_text = []
```

BeautifulSoup:

- It is use for getting data out of HTML, XML, and other markup languages.
- Use the BeautifulSoup library to extract only relevant hyperlinks for Google, i.e. links only with '<'a'>' tags with href attributes.
- Turn the URL's into a Dataframe

- After you get the list of your websites with hyperlinks turn them into a Pandas DataFrame with columns “from” (URL where the link resides) and “to” (link destination URL)

In [9]:
```
for url in list_urls:
    browser.get(url)
    soup = BeautifulSoup(browser.page_source,"html.parser")
    for line in soup.find_all('a'):
        href = line.get('href')
        links_with_text.append([url, href])
```

Turn the URL's into a Dataframe

- After you get the list of your websites with hyperlinks turn them into a

In [10]:
```
df = pd.DataFrame(links_with_text, columns=["from", "to"])
```

In [11]:
```
df.head()
```

Out[11]:

	from	to
0	https://www.ezeephones.com/	None
1	https://www.ezeephones.com/	https://www.ezeephones.com/
2	https://www.ezeephones.com/	/cart
3	https://www.ezeephones.com/	/category/notch-phones
4	https://www.ezeephones.com/	/category/Deals - Of The Day

LogisticRegression:

- Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

as a function of X.

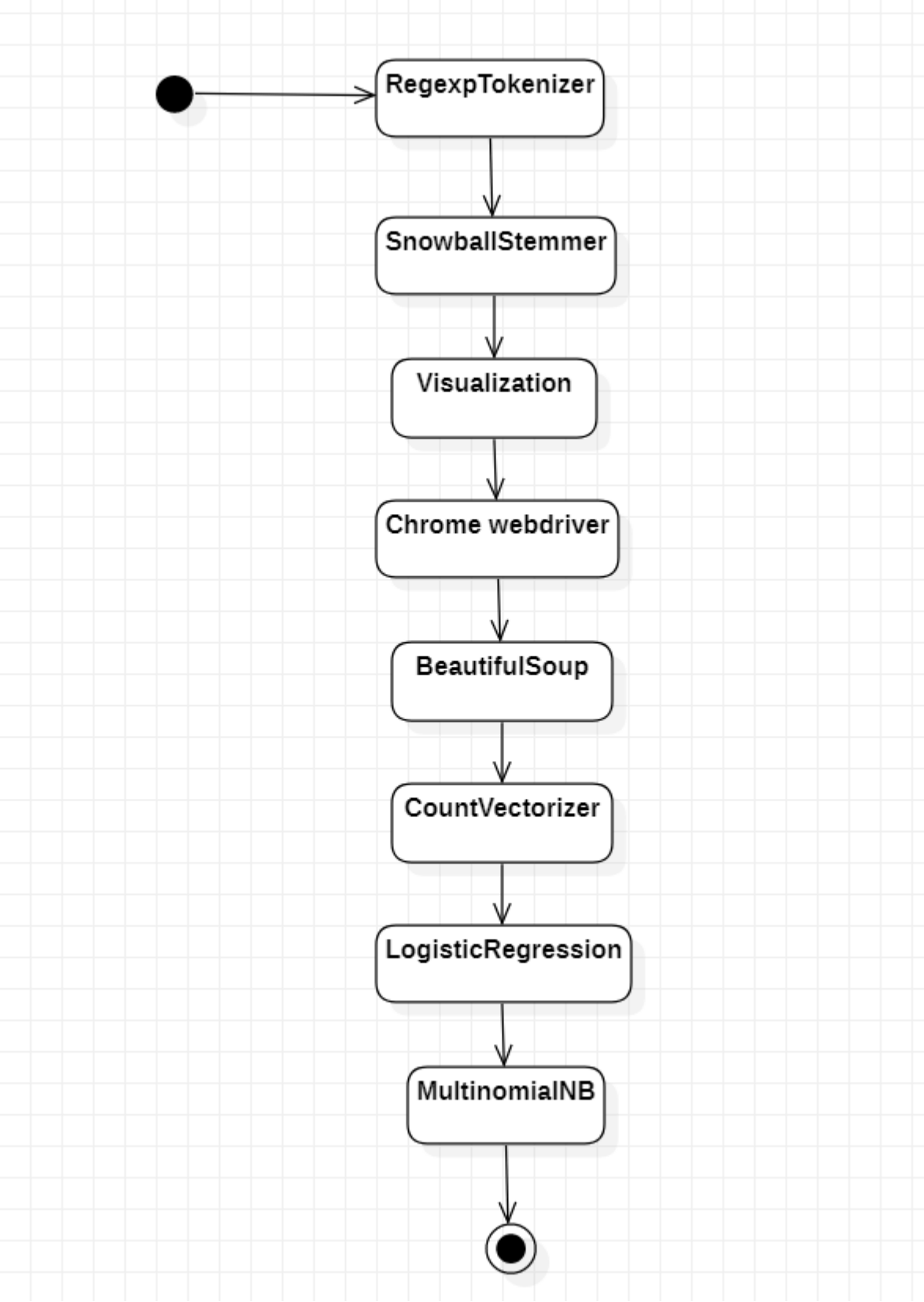
```
In [42]: # create lr object
         lr = LogisticRegression()

In [43]: lr.fit(trainX,trainY)

Out[43]: LogisticRegression()

In [44]: lr.score(testX,testY)

Out[44]: 0.9636514559077306
```

MultinomialNB:

- Applying Multinomial Naive Bayes to NLP Problems. Naive Bayes Classifier Algorithm is a family of probabilistic algorithms based on applying Bayes' theorem with the “naive” assumption of conditional independence between every pair of a feature.

```
In [47]: # create mnb object
         mnb = MultinomialNB()

In [48]: mnb.fit(trainX,trainY)

Out[48]: MultinomialNB()

In [49]: mnb.score(testX,testY)

Out[49]: 0.9574550194048217
```

Complete Design flowchart with technical details (i,e diagram with algorithm/technique used) :



Complete technical description about your project:

We have developed our project using a website as a platform for all the users. This is an interactive and responsive website that will be used to detect whether a website is legitimate or phishing. This website is made using different web designing languages which include HTML, CSS, Javascript and Django.

The basic structure of the website is made with the help of HTML. CSS is used to add effects to the website and make it more attractive and user-friendly. It must be noted that the website is created for all users, hence it must be easy to operate with and no user should face any difficulty while making its use. Every naÃ¨ve person must be able to use this website and avail maximum benefits from it.

Beautiful Soup is a Python package for parsing HTML and XML documents (including having malformed markup, i.e. non-closed tags, so named after tag soup). It creates a parse tree for parsed pages that can be used to extract data from HTML,which is useful for web scraping.

Beautiful Soup was started by Leonard Richardson, who continues to contribute to the project,and is additionally supported by Tidelift, a paid subscription to open-source maintenance.

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

Logistic Regression Assumptions

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other. That is, the model should have little or no multicollinearity.
- The independent variables are linearly related to the log odds.

- Logistic regression requires quite large sample sizes.

Implementation (75%) :

In [22]:
```
def plot_wordcloud(text, mask=None, max_words=400, max_font_size=120, figure_size=(24.0,16.0),
                   title = None, title_size=40, image_color=False):
    stopwords = set(STOPWORDS)
    more_stopwords = {'com','http'}
    stopwords = stopwords.union(more_stopwords)

    wordcloud = WordCloud(background_color='white',
                    stopwords = stopwords,
                    max_words = max_words,
                    max_font_size = max_font_size,
                    random_state = 42,
                    mask = mask)
    wordcloud.generate(text)

    plt.figure(figsize=figure_size)
    if image_color:
        image_colors = ImageColorGenerator(mask);
        plt.imshow(wordcloud.recolor(color_func=image_colors), interpolation="bilinear");
        plt.title(title, fontdict={'size': title_size,
                                   'verticalalignment': 'bottom'})
    else:
        plt.imshow(wordcloud);
        plt.title(title, fontdict={'size': title_size, 'color': 'green',
                                   'verticalalignment': 'bottom'})
    plt.axis('off');
    plt.tight_layout()
d = 'C:/Users/ADMIN/OneDrive/Desktop/Phishing Site URLs Prediction/'
```

In [23]:
```
data = good_sites.text_sent
data.reset_index(drop=True, inplace=True)
```

In [24]:
```
common_text = str(data)
common_mask = np.array(Image.open(d+'star.png'))
plot_wordcloud(common_text, common_mask, max_words=400, max_font_size=120,
               title = 'Most common words use in good urls', title_size=15)
```

In [25]:
```
data = bad_sites.text_sent
data.reset_index(drop=True, inplace=True)
```

In [26]:
```
common_text = str(data)
common_mask = np.array(Image.open(d+'comment.png'))
plot_wordcloud(common_text, common_mask, max_words=400, max_font_size=120,
               title = 'Most common words use in bad urls', title_size=15)
```

Most common words use in bad urls


```
print('\nCONFUSION MATRIX')
plt.figure(figsize= (6,4))
sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")
```

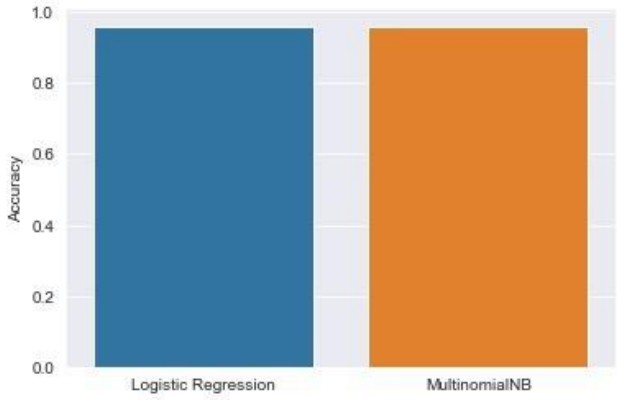
```
Training Accuracy : 0.9773597178702407
Testing Accuracy : 0.9636878627026948

CLASSIFICATION REPORT

              precision    recall  f1-score   support

         Bad       0.90      0.97      0.93     36576
        Good       0.99      0.96      0.97    100761

    accuracy                           0.96    137337
   macro avg       0.95      0.96      0.95    137337
weighted avg       0.97      0.96      0.96    137337


CONFUSION MATRIX
```

9]: <AxesSubplot:>

Results and discussion:

```
In [46]: acc = pd.DataFrame.from_dict(Scores_ml,orient = 'index',columns=['Accuracy'])
         sns.set_style('darkgrid')
         sns.barplot(acc.index,acc.Accuracy)
```

```
Out[46]: <AxesSubplot:ylabel='Accuracy'>
```



* **So, Logistic Regression is the best fit model, Now we make sklearn pipeline using Logistic Regression**

DATASET (description):

- Name of our dataset is phishing_site_urls.csv
- The given data set is in comma separated values(.csv file).
- File is containing 5,49,346 unique entries.
- There are two columns.
- Label column is prediction col which has 2 categories
- A. Good - which means the URLs is not containing malicious stuff and this site is not a Phishing Site.
- B. Bad - which means the URLs contains malicious stuffs and this site isa Phishing Site.
- There is no missing value in the dataset.

Review-3

Innovation Introduced in Design:

- In this project we used a app called fastapi.
- We can use this API by importing a library called uvicorn.
- The API is connected to pishing detection model.
- It is done by creating a PKL file it is a python module that enables objects to be strialized to files on disk and deserialized back into the program at run time.

Process to execute the model is:

- First we have to run .ipynb file and then the PKL file is strialized.
- The next step is to run the python file.It will open the fastapi, from that API we can detect the website.

Ultimate Utility value of the project to the society and industry:

Improve on Inefficiencies of SEG and Phishing Awareness Training

- As increasingly-sophisticated phishing attacks, such as BEC, become more difficult to detect, even by trained security personnel. Thus there is an urgent need for the channel to provide customers with technology that not only strives to prevent intrusion, but can also help users after an attack has passed through the secure email gateway.
- A mailbox-level anti-phishing solution offers an additional layer of protection by analyzing account information and understanding users' communication habits. This delivers an enhanced level of phishing protection to detect attacks faster, alert users and remediate threats as quickly as possible. Machine learning scores sender reputation enabling a baseline for what "normal communications" with a user should look like. It can then compare correspondence and incoming messages with multiple data points to identify and learn from anomalies.

It Takes a Load off the Security Team

- Customers now have many tools on the market to enhance their email security. The best of these use artificial intelligence and machine learning to better identify some of the suspected threats. This not only improves security, but can significantly reduce the workloads of IT and security teams. According to a survey by Fidelis Cybersecurity, less than one in five organizations have a dedicated threat hunting team, and only half of those could handle more than eight investigations per day.

- Security teams need all the help they can get, and must look beyond human intelligence to additional aids that protect the integrity of their corporate information. Automation can improve efficiency by flagging and analyzing the growing number of investigations, and by filtering out false positives. An automation detection and mitigation system reduces response time, identifies threats and can classify and remediate them with one click. It also delivers better metrics and intelligence with real-time insight into the strategies employed by the latest threats. Pushing theses duties to an innovative solution not only increases security, but also enables security teams to focus more on policy and prevention.

It Offers a Solution, Not a Tool

- The goal of security isn't solely to bring tools to the table, it is to offer solutions. resellers ultimately need to ensure that solutions work for the organization, and that calls for listening to customers, understanding the channels, the issues and how they are impacting the organization.
  Whereas basic tools simply offer information and basic applications, automated and advanced phishing threat protection solutions can help solve the challenges that customers face. This ultimately helps the channel discuss solutions with their customers and offer an overall picture and architecture of solving the challenges that are now inherent to email security. Automated advanced phishing threat protection defends against today's and tomorrow's threats with a system that continually learns and makes the entire organization more security conscious and aware.

Separate You from Your Competitors

- As the channel is sometimes slow to move to some of the more advanced technologies, those that are fast will gain the upper hand and a competitive advantage. Merely having these conversations will automatically distinguish you from many of your competitors. It demonstrates that you have a pulse on the latest threats and an insight into how artificial intelligence and machine learning can improve the security posture, without adding more burdens to them.

New Learning experience gained:

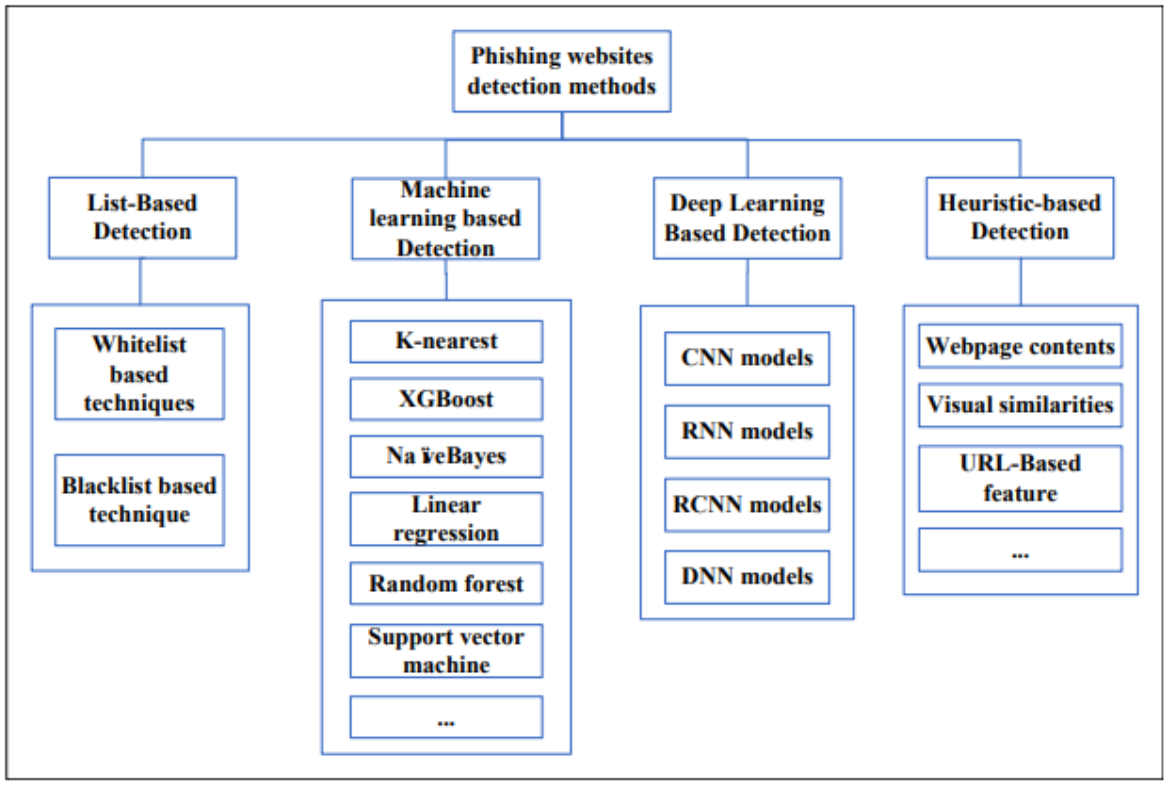
Everyone are familiar with traditional methods like ,

Random forest algorithm creates the forest with number of decision trees. High number of tree gives high detection accuracy. Creation of trees is based on bootstrap method. In bootstrap method features and samples of dataset are randomly selected with replacement to construct single tree. Among randomly selected features, random forest algorithm will choose best splitter for classification.

Decision tree begins its work by choosing best splitter from the available attributes for classification which is considered as a root of the tree. Algorithm continues to build tree until it finds the leaf node. Decision tree creates training model which is used to predict target value or class in tree representation each internal node of the tree belongs to attribute and each leaf node of the tree belongs to class label.

And there are so many methods to detect phishing websites. which are mentioned in the diagram.

Figure 1. Example of phishing website.

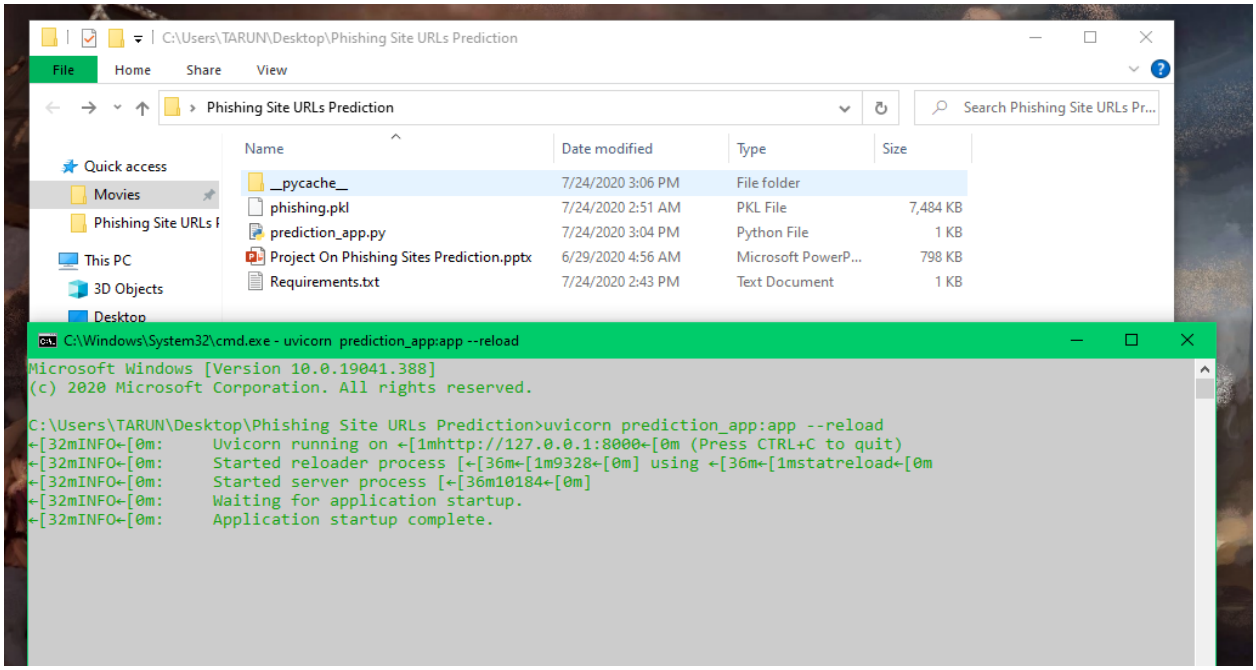


- In our model first we split the string using regular expressions and Tonkenizer.

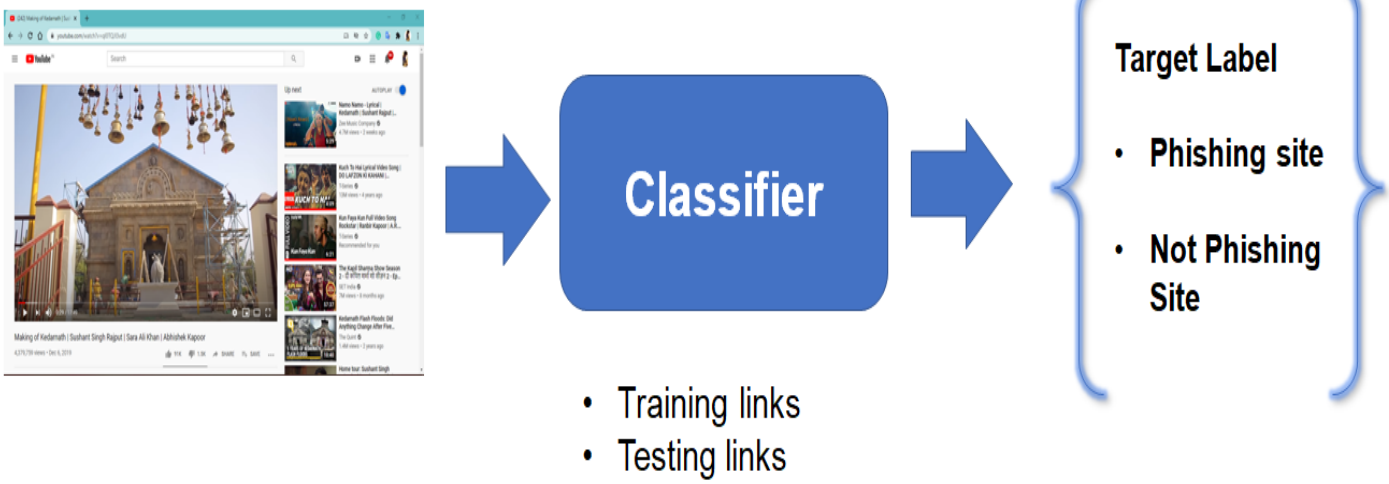
- And we used snowball streamer which is a string processing language which gives root words.
- We use web drivers tool for automated testing of webapps across many browser. it provides capabilities for navigating to web pages, user input and more.
- And then we used a library called beautiful soap to extract only relevant hyperlinks which are good to use.
- when coming to decision making we used logistic regression and multinomial NB which provides a good accuracy.

Implementation:

```
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)
```



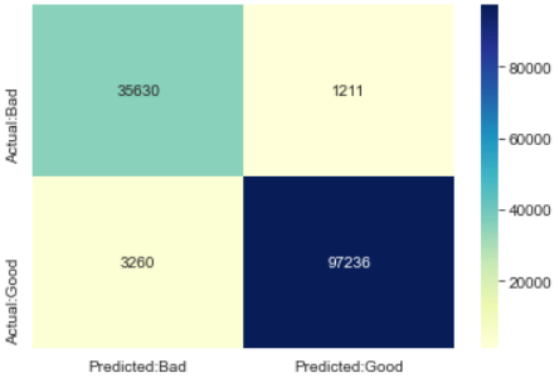
Project Overview:



Results and discussion:

Out[57]:

```
CONFUSION MATRIX
<matplotlib.axes._subplots.AxesSubplot at 0x1ec83fca588>
```



In [58]:

```
pickle.dump(pipeline_ls,open('phishing.pkl','wb'))
```

In [59]:

```
loaded_model = pickle.load(open('phishing.pkl', 'rb'))
result = loaded_model.score(testX,testY)
print(result)
```

```
0.9674450439430016
```

it's that simple yet so effective. We get an accuracy of 98%. That’s a very high value for a machine to be able to detect a malicious URL with.

Want to test some links to see if the model gives good predictions

In [ ]:

```
* Bad links => this are phishing sites
yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php
fazan-pacir.rs/temp/libraries/ipad
www.tubemoviez.exe
svision-online.de/mgfi/administrator/components/com_babackup/classes/fx29id1.txt

* Good links => this are not phishing sites
www.youtube.com/
youtube.com/watch?v=qI0TQJI3vdU
www.retailhellunderground.com/
restorevisioncenters.com/html/technology.html
```

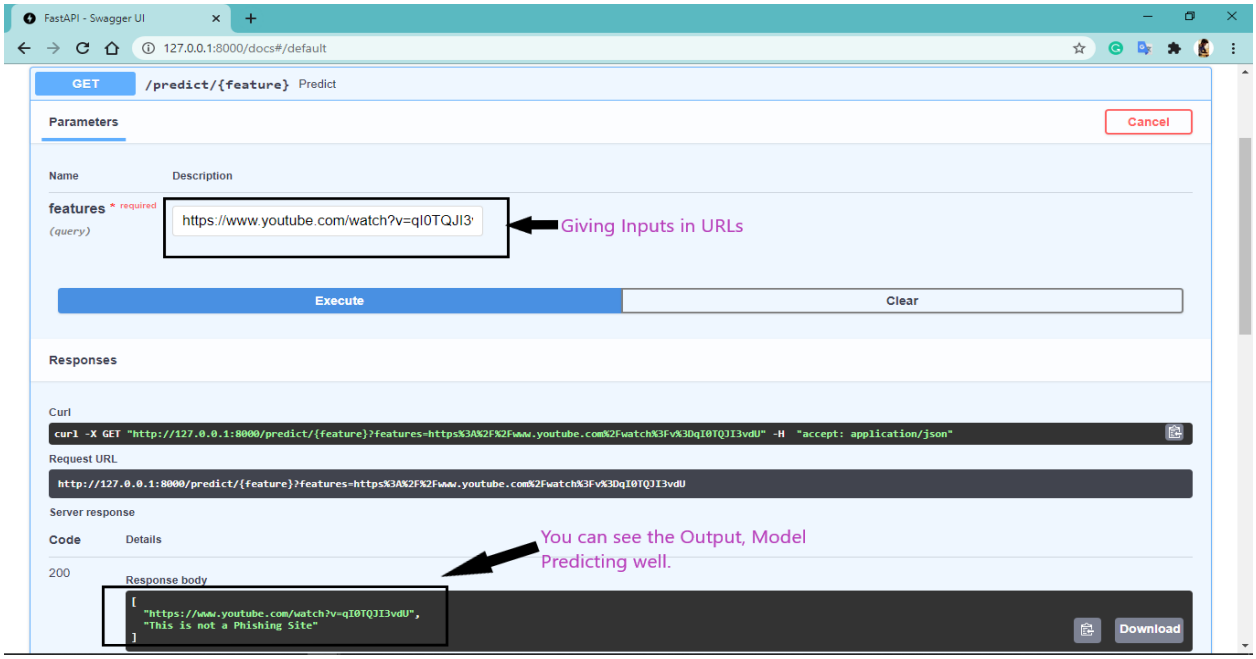
In [60]:

```
predict_bad = ['yeniik.com.tr/wp-admin/js/login.alibaba.com/login.jsp.php','fazan-pacir.rs/
predict_good = ['youtube.com/','youtube.com/watch?v=qI0TQJI3vdU','retailhellunderground.com
loaded_model = pickle.load(open('phishing.pkl', 'rb'))
#predict_bad = vectorizers.transform(predict_bad)
# predict_good = vectorizer.transform(predict_good)
result = loaded_model.predict(predict_bad)
result2 = loaded_model.predict(predict_good)
print(result)
print("*"*30)
print(result2)
```

```
['bad' 'bad' 'bad' 'bad']
******************************
['good' 'good' 'good' 'good']
```

API output:

Prediction Good Sites



Prediction Phishing Site