

PHISHING WEBSITE DETECTION

Major project report submitted in fulfilment of the
requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering

By

Akriti Soni (181315)

Pranchal Abrol (181260)

UNDER THE SUPERVISION OF

Mr. Prateek Thakral



Department of Computer Science & Engineering and Information
Technology

Jaypee University of Information Technology, Wagnaghat,
173234, Himachal Pradesh, INDIA

TABLE OF CONTENTS

Content	Page No.
Declaration by Candidate	I
Certificate by Supervisor	II
Acknowledgment	III
Abstract	IV
1. Chapter No. 1: INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objective	4
1.4 Methodology	4
1.5 Organization	6
2. Chapter No. 2: Literature Survey	9
3. Chapter No. 3: System Development	13
3.1 Analysis of the algorithm	13
3.2 Design	23
3.3 Flow Chart	27
3.4 Algorithm	29
3.5 Model Development	37
4. Chapter No. 4: Performance Analysis	38
4.1 Predicting the result	38
4.2 Model Testing	39
4.3 Discussion on the result Achieved	46
5. Chapter No. 5: Conclusion	53
5.1 Future Work	55
REFERENCE	63

DECLARATION

I hereby declare that, this project has been done by me under the supervision of **Mr.Prateek Thakral, Assistant Professor**, Jaypee University of Information Technology. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Mr.Prateek Thakral

Assistant Professor

Department of Computer Science & Engineering and Information
Technology

Jaypee University of Information Technology

CERTIFICATE

I hereby declare that the work presented in this report entitled “**PHISHING WEBSITE DETECTION**” in fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee

University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2022 to May 2022 under the supervision of **Mr. Prateek Thakral**, Assistant Professor, department of Computer Science & Engineering and Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

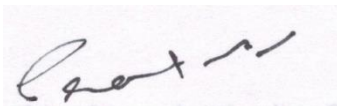


AkritiSoni, 181315



PranchalAbrol, 181260

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



Mr. Prateek Thakral

Assistant Professor

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat

AKCNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Mr. Prateek Thakral, Assistant Professor**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Mr. Prateek Thakral**, Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making the project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

AkritiSoni

PranchalAbrol

LIST OF FIGURES

<i>Figure name</i>	<i>Page no.</i>
1.1 Flow of Methodology	04
3.1 Classification and Regression	25
3.1 LogisticRegression	26
3.2 MultinomialNB	27
3.3 FastAPI	30
3.4 Autoencoder neurak network	30
3.2.1 New window	21
3.2.2 Nx.draw(graph)	21
3.4.1 Combining dataset into one frame	22
3.4.2 Dataset	22
4.1 Importing Libraries	23
4.2 Combining dataset into one frame	23
4.3 Dataset	24
4.4 Missing value	24
4.5 Visualizing	25
4.6 Regular Expression	25
4.7 SnowballStemmer	26
4.8 WordCloud	27-28
4.9 Networkx	52
4.10 Create model	53
4.11 LogisticRegression	53
4.12 Good and bad links	54
4.2.2 Graphs of network	55
4.2.3 Testing	55
4.2.4 Tokenizer	56
4.2.5 Classification of imbalanced	57
4.2.6 Similar words which we use	57
4.2.7 SnowballStemmer good or bad	58
4.2.8 Visualizing of wordcloud	59
4.2.9 Common good words	59
4.2.10 Common bad words	59

4.2.11 prediction	60
4.2.12 Prediction between logistic regression and multinomialnb	60
4.2.13 Confusion matrix	60
5.3.1 LogisticRegression	61
5.3.2 MultinomialNB	62
5.3.3 FastAPI	63
5.3.5 Autoencoder neural network	64

ABSTRACT

Online phishing is one of the most common attacks on the modern internet. The goal of phishing website uniform resource locators is to steal personal data including login credentials and credit card numbers. As technology keeps growing, phishing strategies began to develop rapidly.

Machine learning built an effective device used to attempt phishing attacks. In this project, we have built a phishing website by using fastAPI. We have used two so many different libraries and two algorithms which are logistic regression and multimodal NP. The purpose of this project is to check whether phishing websites are good URLs or bad URLs. We gathered data to create a dataset of malicious links and curate it for the machine learning model.

Chapter 01: INTRODUCTION

In modern era Phishing becomes a main area of concern for security researchers due to the fact it is not tough to create the fake internet site which looks so close to legitimate internet site. Experts can discover fake web sites however not all the customers can discover the fake website and such customers become the victim of phishing attack. Main purpose of the attacker is to steal banks account credentials. How hackers do their work, they send you just spam mail. In this mail though they will say that this email is mean to inform you that you're my university network password will expire in 24 hours and they have provide you to update the password and login when we click on that link we will redirect to that page which is a hacker server and they will be steal your data everything which is online.

In our project we have to predict phishing websites whether they are good uniform resource locators (URLs) or bad URLs. The set of phishing URLs are gather from open source service called Phish Tank. Benign URLs (uniform resource locator) with zero malicious detection were classified as benign and URLs with no less than eight detection were classified as malicious. It is being labeled as '0' and Phishing URL is being labeled as '1'. We study several machine learning algorithm for analysis of the characteristic in order to get a good understanding of the construction of the URLs that expand phishing. Phishing attacks are getting a success because lack of consumer awareness. Since phishing attack exploits the weaknesses found in customers, it's far very tough to mitigate them however it may be very vital to enhance phishing detection strategies. The general technique to discover phishing web sites through updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also recognized as "blacklist" technique. To evade blacklists attackers makes use of innovative techniques to fool customers through modifying the URL to appear valid via obfuscation and lots of

other easy techniques such as: fast-flux, in which proxies are automatically generated to host the web-page; algorithmic era of recent URLs etc. To entice human beings, Phisher sends “spooled” mails to several human beings as possible. When such emails are opened, the customers generally tend to redirect to the spoofed internet site. The internet site intuitively asks you to run a software program or download a document while you're not waiting to do so. The internet site tells you that your device is infested with malware or that your browser extensions' or software program the machine of date. Malicious URLs on the website could be easily recognized by examining them through machine learning techniques.

To avoid getting phished, people should have an understanding of phishing websites and how they look if the person is using an online browser. So, the high-end companies can even blacklist phishing websites or detect phishing in their early arrival, using machine learning and deep neural network algorithm to build a model that can classify URLs as phishing. Machine learning technique is proven to be efficient than the other technique.



1.1 Problem Statement

The major trouble is that phishing technique is bad accuracy and low adaptability to new phishing links. We plan to apply machine learning to overcome these limitation through imposing some classification algorithms and evaluating the overall performance of these algorithms on our dataset.

We have decided on the Random Forest method because of its excellent performance in classification but random forest and decision tree are not good with nlp data.

1.2 Objective

A phishing internet site is the most common social engineering approach that mimics trustful URLs and web pages. This project aims to predict phishing websites whether are good URLs or bad URLs. Both phishing and benign URLs of websites are collected to form a dataset and from them required URLs and these projects aimed functions are extracted. We have used so many different libraries and algorithms like LogisticRegression, numpy,

pandas, MultinomialNB, RegexpTokenizer many more. We gather data to create a dataset of malicious links and curate it for the ML model. The performance level of every model is measured and compared.

1.3Methodology

In this project, we have predicted phishing websites whether they are good URLs or bad URLs. Before performing a code we have done some surfing and found some datasets. We have collected the phishing and estimated websites from open source platforms by using FastAPI and developed a bit of code to extract the feature. As there are different types of URLs like spam, benign, phishing, malware etc from legitimate URLs. Benign has around 35,000 URLs, so out of them; we have used 5000 URL randomly. We have examined and pre-process the dataset and divide the datasets into training and test sets. Then we combine all the dataset into one frame. The data was containing more than 5lakhs unique approach and there were two columns which was further categories into Good and Bad. Further we have done some classification problems and vectorize our URLs by using CountVectorizer and tokenizer. We have used different libraries for different functions such as for visualizing most common words in good and bad URLs and turn URLs into data frame. After that we have create the model and splitting the data. Then we have import links to prediction and deploy the model. The metric that we used is accuracy

which is a simple one. We have compared the accuracies of the training datasets and the best one is declared.

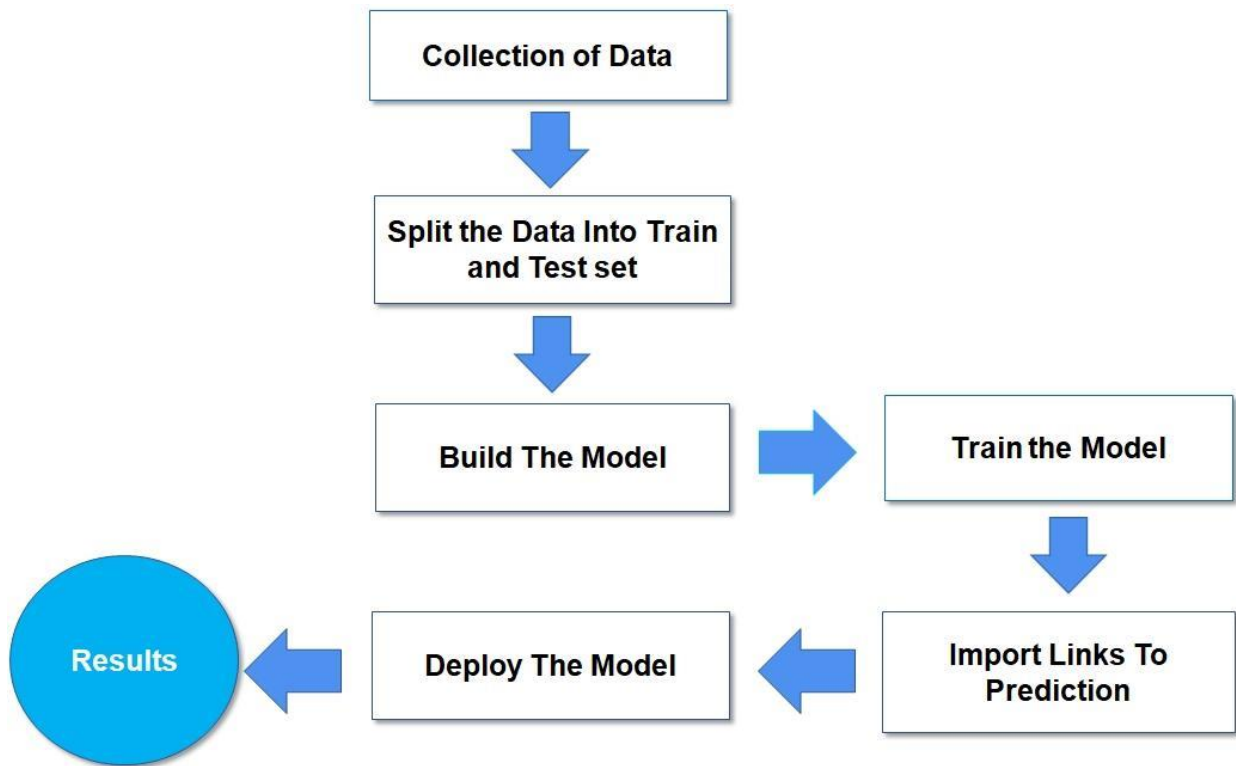


Figure 1.1 Flow of Methodology

1.4 Organization

In our project we have used FastAPI which is a python framework and import many libraries for different purposes. We have taken two algorithms which is LogisticRegression and MultinomialNB. LogisticRegression will predict the links are good or not and MultinomialNB work well with nlp data (natural language process). Then we have used some classification problems by using CountVectorizer and tokenizer. We have used some another visualization. We can

show that what is the hidden link in the phishing site which will redirect to another server. Then we have networkx it is creating a data structure, dynamic function and more. We are combining three datasets which we collected from several sites then we combine this dataset into one frame. The usability of this dataset is 10.0 which means very good. The data size is approx 30 mb. The data contains more than 5 lakhs unique approach. The label column means that its prediction column in which there were two categories first is good and second is bad. After that we have checked the imbalanced of target column. Now we have a data, we convert URLs into vector form. We have used regular expression tokenizers which divide the string using regular expression. So, in our code we are just splitting only alphabets and some URLs have numbers, dots , slash etc which are not important our data. So we only gather the string and simultaneously we have transformed this in all the rows. After converting into words we used snowball it's an nltk API (natural language toolkit) which is used to string words. It will remove all the English works and create some root words. Root words means that it will combine the common words like pictures, photos for this two words it will create the one word. Phishing data text streamer is equal to all the holder that list is words lists are converted into streamers. Then we join all the lists words into single sentence. We also use word cloud. In our code we have used this to convert most repeated word into the word cloud form. Then we use chrome webdriver. This will create a new

window of that chrome. So to this new chrome we will pass that link. Then by using BeautifulSoup, we gather the all html code from its page source and it is getting all the anchor tags. So we will get all the hidden link which will hacker use to redirect any users to this server and we create a data frame of this links. So it will give a two links: first is what we passed to this and second what we are getting from this link. Logistic regression object and we fit it by trainX, trainY. After that we checked the score and we are getting very good score which is 90.96. After that we just created the confusion matrix to see the actual prediction and normal prediction. Using Logistic Regression we are creating a pipeline. Then we are saving this pipeline model using pickle and we check the accuracy of it and it is giving very good accuracy.

Chapter 02: LITERATURE SURVEY

- **Chunlin Liu, Bo Lang : Finding effective type for malicious URL detection : In ACM,2018**

Chunlin et al. proposed method that primarily consciousness on individual frequency features. In this they've mixed statistical evaluation of URL with machine learning method to get end result this is more accurate for category of malicious URLs. Also they've compared six machine learning algorithms to confirm the effectiveness of proposed algorithm which offers 99.7% precision with fake positive rate much less than 0.4%.

- FadiThabtah et al. experimentally as compared massive numbers of ML techniques on actual phishing datasets and with admire to different metrics. The cause of the evaluation is to show the benefits and drawbacks of machine learning predictive models and to reveal their real performance in terms of phishing attacks. The experimental outcomes display that Covering method models are extra suitable as anti- phishing solutions. MuhemmetBaykara et al. proposed an application which is referred to as Anti Phishing Simulator; it offers records about the detection trouble of phishing and how to detect phishing emails. Spam emails are brought to the database through Bayesian algorithm. Phishing attackers use JavaScript to area a legitimate URL of the URL onto the browsers deal with bar. The recommended method in the have a look at is to apply the textual content of the email as a key-word simplest to carry out complicated word processing.

- **Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017**

Gupta et al. [11] put forward a novel anti phishing method that extracts features from client-aspect only. Suggested method is fast and reliable as it is now no longer depending on third party however it extracts capabilities only from URL and source code. In this paper, they have reached 99.09% of overall detection accuracy for phishing website. This paper has concluded that this method has limitation as it can detect website written in HTML. Non-HTML website can't detect through this method.

- **A Prior-based transfer learning techniques for the Phishing Detection:-**

A logistic regression is the basis of a concern based transferrable learning method, which is supplied right here for our classification of statistical machine learning. It is used for the detection of the phishing web sites relying on our decided on traits of the URLs.

Due to the divergence in the allotment of the capabilities in the distinct phishing areas, several models are proposed for distinctive regions.

It is sort of impractical to accumulate enough information from a new region to repair the detection model and use the transfer learning algorithm for adjusting the present version. A suitable manner for phishing detection is to apply our URL based technique. To deal with all of the conditions of failure of detecting traits, we need to undertake the shifting method to generate a extra effective version.

- **Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013**
Ahmad et al. [17] proposed three new capabilities to enhance accuracy rate for phishing internet site detection. In this paper, Author used both type of capabilities as usually recognized and new features for category of phishing and non-phishing site. At the cease author has concluded this work can be improve by the usage of this novel capabilities with decision tree machine learning classifiers

- **Sahingoz, O.K., Buber, E., Demir, O. and Diri, B., 2019. Machine learning primarily based totally phishing detection from URLs. Expert Systems with Applications, 117, pp.345-357.**

Attempts to detect phishing site the use of URL for preventing customer sensitive information. Computer customers fall for phishing because of the 5 major reasons:-

- Users don't have precise information about URLs,
- Users don't know, which internet pages may be trusted,
- Users don't see the entire address of the internet page, because of the redirection or hidden URLs,
- Users don't have plenty time for consulting the URL, or by chance enter a few internet pages,
- Users can't distinguish phishing internet pages from the valid ones.

In proposed system, writer used NLP based features and Word features for type of phishing and non-phishing sites. For category used Decision Tree, Adaboost, K-star, kNN(n=3), Random Forest, SMO(Sequential Minimal Optimization) and Naïve Bayes [12].

Chapter 03: SYSTEM DEVELOPMENT

3.1 Analysis of the Algorithms:

As our whole project is based on supervised machine learning.

Supervised learning is a subcategory of machine learning and artificial intelligence. It works as we pass a data along with label to that data to a model and once the model is trained it recognizes some patterns and associates the labels to that pattern and thus makes the new predictions. There can be so many different applications possible using supervised learning. Some of them can be to detect the spam or spam detection. This is the way in which we can detect if mail is a spam or not if mail is a spam it will automatically put it in a spam folder and if it is not a spam mail then put it in your inbox. Another can be object classification and many more. Supervised machine learning has two categories:-

- **Classification :**

It helps find things that we can search by keywords, but it actually helps you find our own invention that's very close to our own. An area that is grouped in subject areas called classes and subclasses. Used to classify characteristics of invention. The type predictive modelling is the challenges of approximating the mapping characteristic from enter variables to discrete output variables. Example: email spam detector. The major purpose of the Classification algorithm is to pick out the category of a given dataset, and those algorithms are especially used to expect the output for the specific data. The algorithm which implements the kind on a dataset is referred to as a classifier.

- **Regression:**

It is a supervised learning method which enables in finding the correlation among variables and allows us to be looking ahead to the non-stop output variable primarily based totally at the simplest or greater predictor variables. It is specifically used for prediction, forecasting, time collection modeling, and figuring out the causal-impact relationship among variables. In Regression, we plot a graph among the variables which satisfactory suits the given data points, the use of this plot, the machine learning version could make predictions about the data.

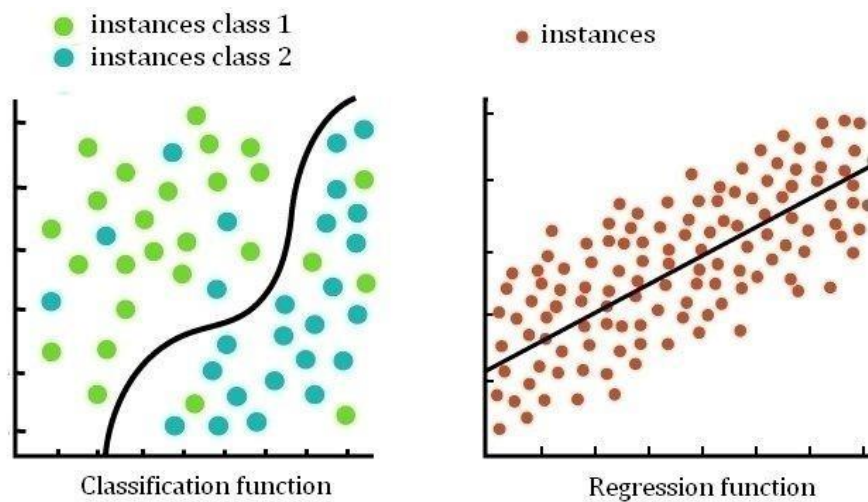


Fig 3.1 Classification and Regression

In our project we have used FastAPI which is a python framework and import many libraries for different purposes. We have taken two algorithms which is LogisticRegression and MultinomialNB. LogisticRegression will predict the links are good or not and MultinomialNB work well with nlp data (natural language process). Then we have used

some classification problems by using CountVectorizer and tokenizer. We have used some another visualization. We can show that what is the hidden link in the phishing site which will redirect to another server. We are combining three datasets which we collected from several sites then we combine this dataset into one frame. The usability of this dataset is 10.0 which means very good. The data size is approx 30 mb.

So, this has different types of URLs like spam, benign, phishing etc. We took the benign URL from them and it has around 35,000 URLs. So, out of them we have used 5000 URL randomly. Benign URLs (uniform resource locator) with zero malicious detection were classified as benign and URLs with no less than eight detection were classified as malicious.

We study several machine learning algorithm for analysis of the characteristic in order to get a good understanding of the construction of the URLs that expand phishing. So, supervised machine learning that we have chosen for this project are:-

3.1.1 LogisticRegression:

Logistic Regression is set becoming a curve to the facts. It is utilized in statistical software program to apprehend the connection among the based variable and one or greater unbiased variables with the aid of using estimating possibilities the usage of a LogisticRegression equation. These

sorts of estimation allow that we are expecting the probability of an occasion taking place or a desire being made.

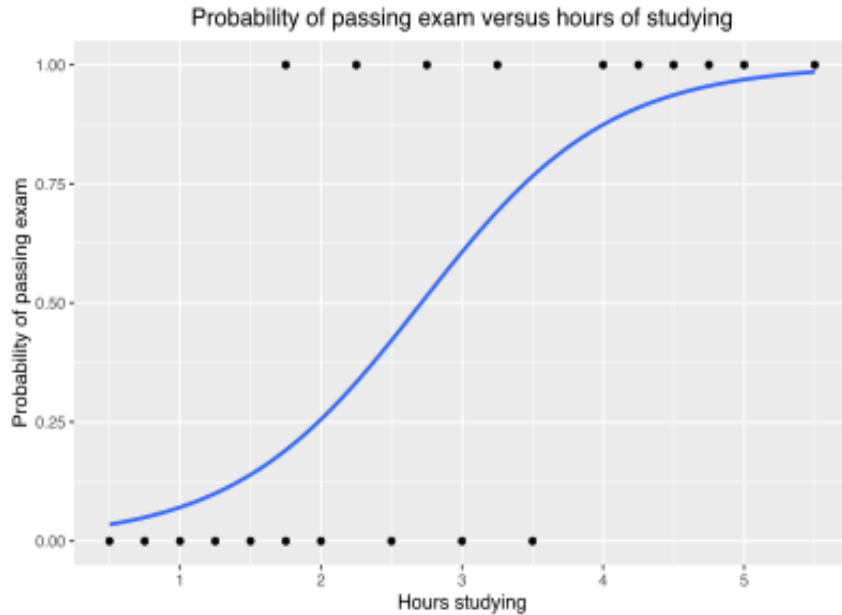


Fig 3.2 LogisticRegression

3.1.2 MultinomialNB :

Multinomial Naïve Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). For example : As in this definition how many times the word is coming or what is the frequency of word in this definition and it is a machine learning method. Naïve Bayes is also called conditional probability in the world of statistics. Multinomial is described discrete frequency counts in other words word counts or something like that.

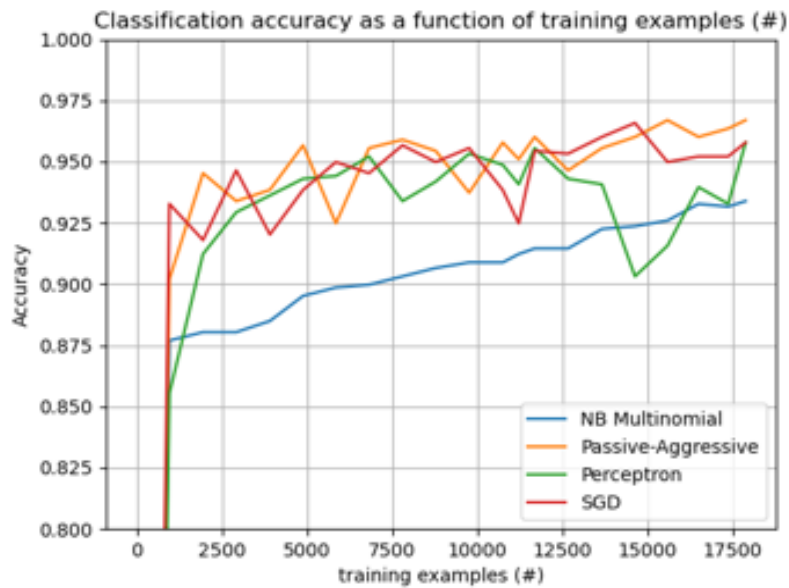


Fig 3.3 MultinomialNB

3.1.3 FastAPI :

There are already popular frameworks based on flask and django. It is relatively new framework. So it was able to learn great features from other tools and also having new features such as better python type hints.

Key benefits of FastAPI :-

- **Fast (high-performance):** It is one of the fastest python frameworks available. This is mainly due to the framework being built on top of tools like Starlette and Pydantic. FastAPI features parallelism FastAPI is especially a good match for building web API for machine learning systems.

- **Easy data Validation** in FastAPI based on standard python type hints handled by pydantic. The python type hints are a special syntax to indicate the type of variable. It is relatively new feature that's only available in recent version of python with such declaration of types in FastAPI pydantic enforces it at run time and provides user friendly error messages when data is invalid.
- **Automatic interactive documentation:** As we program in FastAPI it automatically documents everything with open standards like OpenAPI and JSON Schema. We can view this documentation through systems like Swagger UI and ReDoc in our browsers.
Also FastAPI was tested on multiple editors. So we can get great editor support including auto completion. This means we can code faster and have fewer bugs using FastAPI .

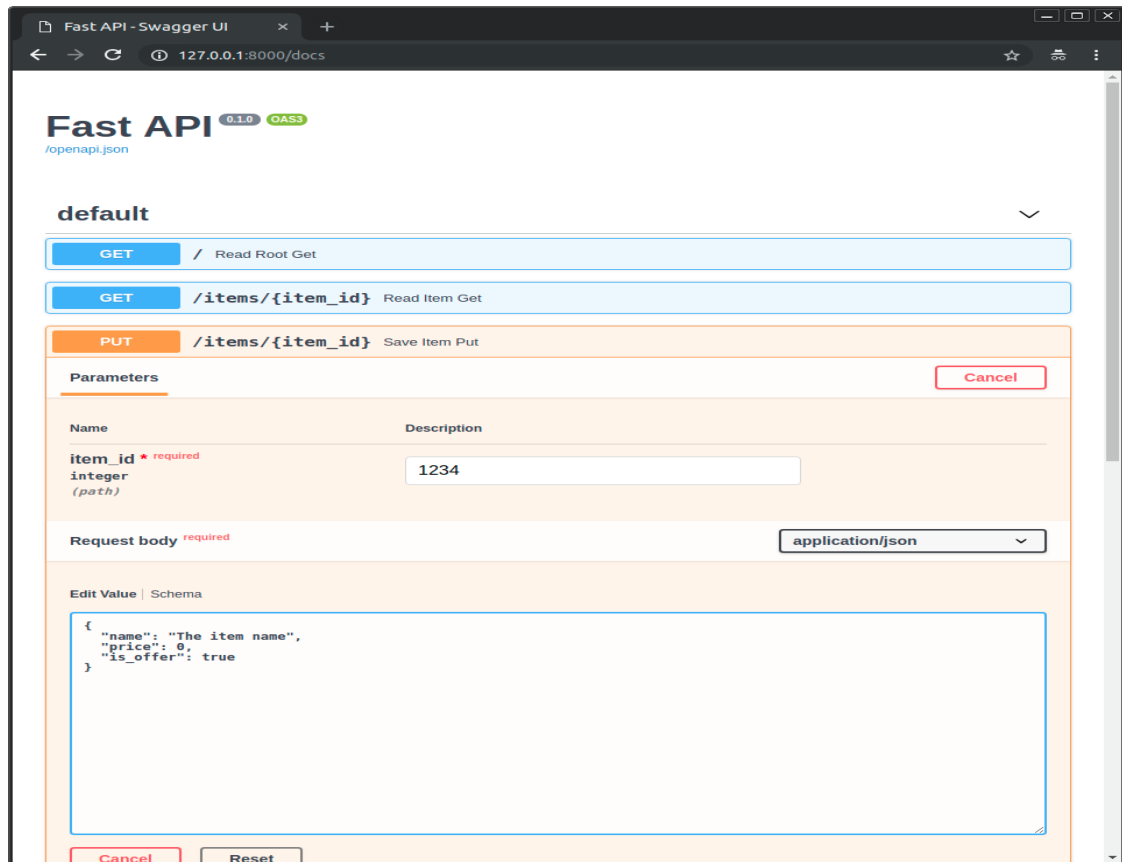


Fig 3.4 FastAPI

3.1.4 RegexpTokenizer:

Regular Expression is quite useful or generalize the kind of expression that we can use. The way we can simply avoid is by just telling tokenizer or python that whatever words it is scanning in that word if it is finding an apostrophe do not break into two parts. So for that what we can do is we can use something called a regular expression or tokenizer. It allows us to check a series of characters for 'matches'

3.1.5: NLTK (Natural Language Toolkit) :

Suite of open source tools created to make NLP processes in Python easier to build. NLP has revolutionised many areas, like it may be parts of speech tagging. It may be sentence translation. It may be even Text generation and many applications. So there are many inbuilt of functions and libraries that are included inside this NLTK library. For example, we may have a stem function inside this so what it does it if many words like coder, coding, coders and many more then it all comes down to its these all words are a stemmed to their root word 'code'. Similarly, words separation so given a sentence or a text we want to tokenize it into a list of words. So we don't need to write a custom function of ourselves. We can just use the tokenize function there and similarly, there are many more things like stop words and many more applications which are inbuilt in NLTK and many more are being added continuously due to its open source nature. So it's very useful library and if we don't know how to use it then anything we want to design ar develop will be very slow because we will need to do everything ourselves.

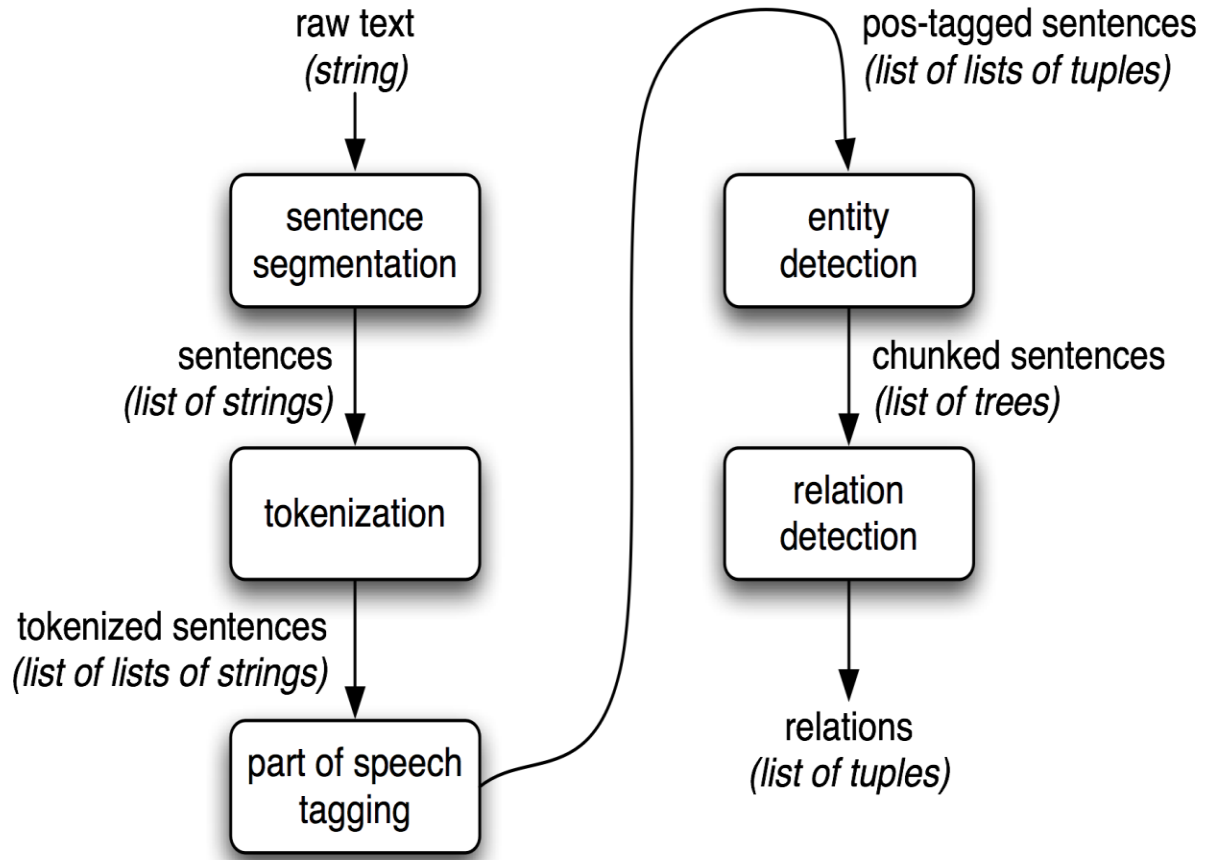


Fig 3.5 Autoencoder neural network

We have examined and preprocess the dataset and divide the datasets into training and checking out sets. The metric that we used is accuracy which is a simple one. We have compared the accuracies of the training datasets and the best one is declared.

3.2 Design:

In this project, we have mainly used machine learning techniques. In which we have used their algorithms for training and prediction whether the links are good or bad and done some accuracy of it. We have used FastAPI which is a python framework and import many libraries for different purposes. We have taken two algorithms which is LogisticRegression and MultinomialNB. LogisticRegression will predict the links are good or not and MultinomialNB work well with nlp data (natural language process). Then we have classification_report it will give whole information of metrics like recall precision and advanced curve.

- The series of phishing URLs is instead easy due to the open source provider referred to as Phish Tank. This provider offer a fixed of phishing URLs in a couple of formats like CSV, JSON etc. that receives up to date hourly. The CSV record of phishing URLs is received through the usage of wget command. After downloading the dataset, it is loaded right into a Data Frame.
- For the legitimate URLs, we discovered a supply that has a set of benign, spam, phishing, malware & defacement URLs. The variety of legitimate URLs on this collection is 35,300. This record is then uploaded to the Colab for the feature extraction.

We have used some classification problems by using CountVectorizer and tokenizer. The CountVectorizer to use to transform the data into a space matrix and at last we do the pipelining work using SQL and pipeline. We have used someanother visualization. We can show that what is the hidden link in the phishing site which will redirect us to the another server. Then we

have networkx it is creating a data structure, dynamic function and more. We are combining three datasets which we collected from several sites then we combine this dataset into one frame. After combining into one frame we create creator and then save into my drive. The usability of this dataset is 10.0 which means very good. The data size is approx 30 mb. There is two columns first are URLs and second is label. S when we saw our information about the dataset data contains more than 5 lakhs unique approach. The label column means that its prediction column in which there were two categories first is good and second is bad. After that we have checked the imbalanced of target column. From that we can saw that there is lots of difference between on both classes. After that we started the pre-processing work. So now we have a data, we convert URLs into vector form. We have used regular expression tokenizer which divides the string using regular expression. So, in our code we are just splitting only alphabets and some URLs have numbers, dots , slash etc which are not important our data. So we only gather the string and simultaneously we have transformed this in all the rows into this tokenizer text. After converting into words we used snowball it's an NLTK API (natural language toolkit) which is used to string words. It will remove all the english works and create some root words. Root words means that it will combine the common words like pictures, photos for this two words it will create the one word. Phishing data text streamer is equal to all the holder that list is a words list is converted into streamers. Then we join all the lists words into single sentence. We also use word cloud. In our code we have used this to convert most repeated word into the word cloud form. Then we use chrome webdriver. This will create a new window of that chrome. So to this new chrome we will pass that link.

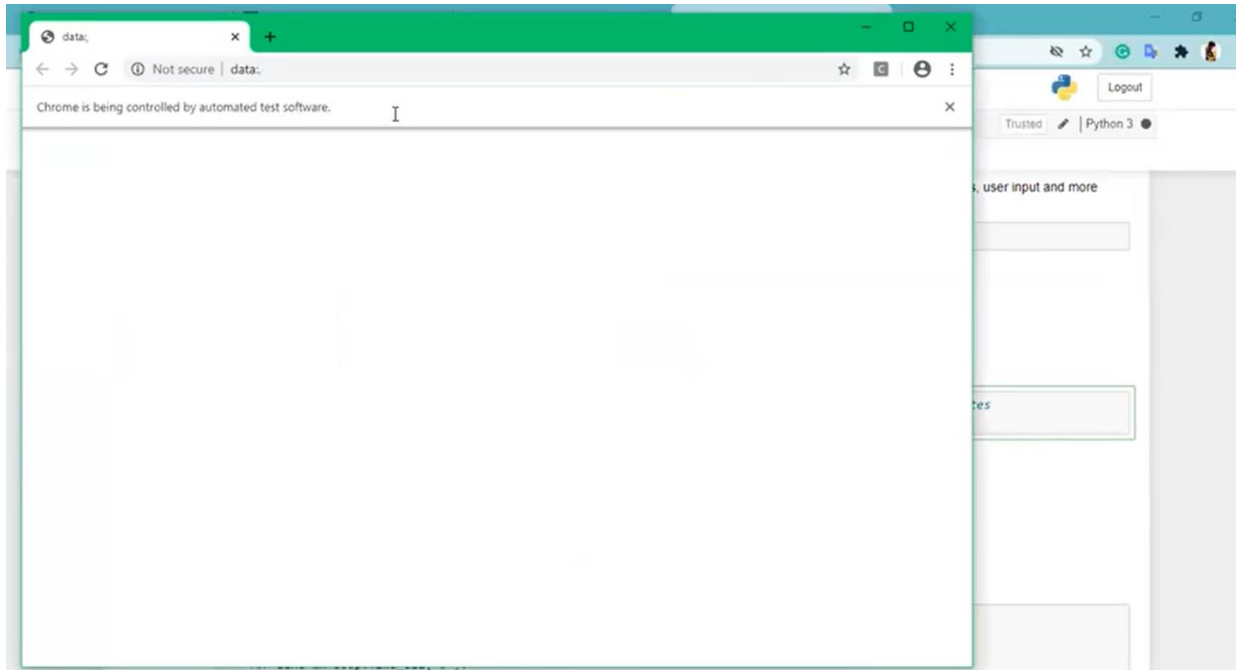


Fig 3.2.1 New Window

Then by using BeautifulSoup, we gather the all html code from its page source and it is getting all the anchor tags. So we will get the entire hidden link which will hacker use to redirect any users to this server and we create a data frame of this links. So it will give a two links: first is what we passed to this and second what we are getting from this link. After that we have use networkx and networkx is doing here by visualize all the internal links in the structure by feeding network from panda edge list first and draw it by calling dot nx dot draw like and here this the redirected links which is shown by the networkx like nodes.

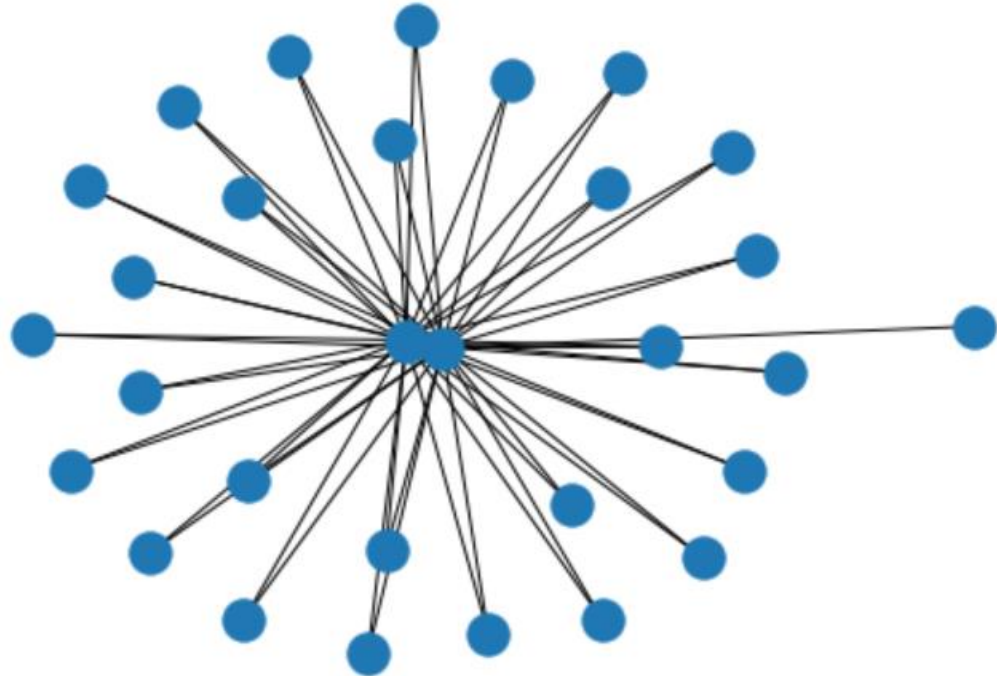
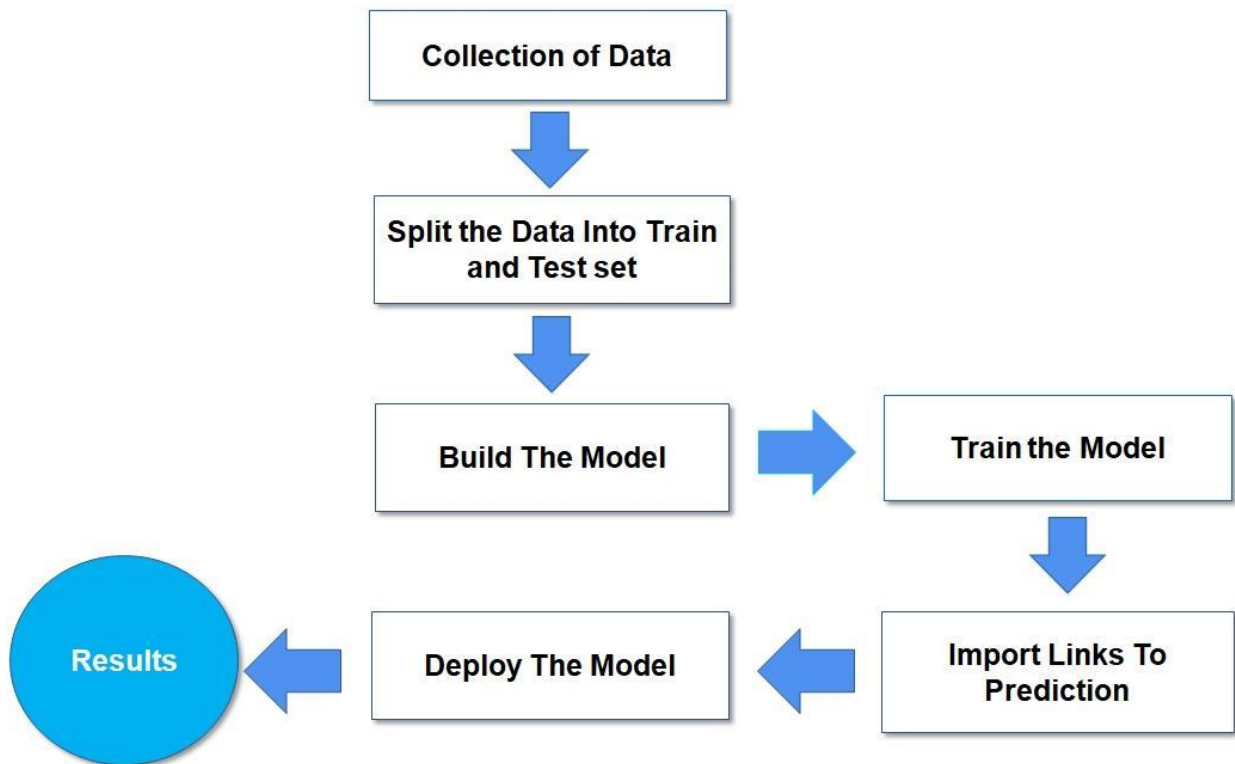


Fig 3.2.2 Nx.draw(graph)

Logistic regression object and we fit it by trainX, trainY. After that we checked the score and we are getting very good score which is 90.96. After that we just print the score so model is not over fitted or undefeated. After that we just created the confusion matrix to see the actual prediction and normal prediction. Multinomial is giving us 95 which are not good as a logistic regression but fine. Using Logistic Regression we are creating a pipeline and we are using the regular tokenizer and stopwatch equals English. There using stopwatch it will be stops all the English words and we also used the tokenizers but these are not giving me the best accuracy. Then we are saving this pipeline model using pickle and we check the accuracy of it and

it is giving very good accuracy. After that we are dumping this model and after dumping we just load model to check our model is working well and we see that load model dot score it will be giving me the 96 percent again.

3.3 Flow Chart



In this project, we have predicted phishing websites whether they are good URLs or bad URLs. Before performing a code we have done some surfing and found some datasets. We have collected the phishing and estimated websites from open source platforms by using FastAPI and developed a bit of code to extract the feature. As there are different types of URLs like spam, benign, phishing, malware etc from legitimate URLs. Benign has around 35,000 URLs, so out of them; we have used

5000 URL randomly. We have examined and pre-process the dataset and divide the datasets into training and test sets. Then we combine all the dataset into one frame. The data was containing more than 5laks unique approach and there were two columns which was further categories into Good and Bad. Further we have done some classification problems and vectorize our URLs by using CountVectorizer and tokenizer. We have used different libraries for different functions such as for visualizing most common words in good and bad URLs and turn URLs into data frame. After that we have create the model and splitting the data. Then we have import links to prediction and deploy the model. The metric that we used is accuracy which is a simple one. We have compared the accuracies of the training datasets and the best one is declared.

3.4 Algorithm

In this project, we have done various implementations for training and we have predicted phishing websites whether they are good URLs or bad URLs. As we have used some of the supervised algorithm.

- Firstly, we imported some libraries such as pandas, numpy, multinomialNB, LogisticRegression and many more.

- We have examined and pre-process the dataset and divide the datasets into training and test sets. Then we combine all the dataset into one frame.

```
In [2]: phishing_data1 = pd.read_csv('phishing_urls.csv',usecols=['domain','label'],encoding='latin1', error_bad_lines=False)
phishing_data1.columns = ['URL','Label']
phishing_data2 = pd.read_csv('phishing_data.csv')
phishing_data2.columns = ['URL','Label']
phishing_data3 = pd.read_csv('phishing_data2.csv')
phishing_data3.columns = ['URL','Label']

In [3]: for l in range(len(phishing_data1.Label)):
if phishing_data1.Label.loc[l] == '1.0':
    phishing_data1.Label.loc[l] = 'bad'
else:
    phishing_data1.Label.loc[l] = 'good'
```

Fig 3.4.1 Combining dataset into one frame.

- After combining into one frame we created creator and then save into my drive.

```
In [7]: phish_data.head()
```

Out[7]:

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/websrcr...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad

Fig 3.4.2 Dataset

- So we can see that there is two columns first is URLs and second is Label. So when we see our information about dataset data contains more than 5lakhs unique entries which means that more than 5 lakhs unix URL in their data.

- After that we check the imbalanced of that target column.
 1. Label underscore counts is equal to `pd.dataframe (phish underscore data.label.value underscore counts())`
 2. `Sns.set underscore style ('darkgrid')`
`Sns.barplot (label underscore counts. Index, label underscore counts.label)`

- Now that we have our data we have to vectorize our URLs. We have use regular expression tokenizer .
 1. Tokenizer is equal to `RegexpTokenizer(r'[A-Za-z]+')`
In this expression `{r' [A-Za-z]+}` we just splitting only alphabets.
 2. `Phish underscore data.URL[0]`
After we only gather the strings.
 3. `Tokenizer.tokenize (phish underscore data.URL[0])`

- After doing this with one rows we transform all the rows although all the URLs into tokenizer text. So it takes 4 seconds something so to convert all the rows more than 50 URLs into that word.
 1. Print ('getting words tokenizer')
 2. `T0 is equal to time.perf underscore counter()`
 3. `Phish underscore data ['text underscore tokenizer'] is equal to phish underscore data.URL.map (Lambda t: tokenizer.tokenize(t))`
 4. `T1 is equal to time.perf underscore counter()`
 5. Print ('time taken', t1,'sec')

6. Phish underscore data.sample(5)

- So after converting into words we use a snowball. It is a NLTK API which is used to stream words.
- Stemmer is equal to SnowballStemmer (“English”)
- Print (‘getting words tokenizer’)
- T0 is equal to time.perf underscore counter()
- Phish underscore data [‘text underscore tokenizer’] is equal to phish underscore data [‘text_tokenized’].map (lambda 1: [stemmer. stem (word) for word in 1])
- T1 is equal to time.perf underscore counter() subtract T0
- Print (‘time taken’, T1,’sec’)

All the holder that a words list are converted by stemmers. Then we just join the all the list words into just single sentence.

- After that we thought we do some visualization like showing what is the keywords using in bad sites and what is the keywords using by good sites.
- Bad underscore is equal to phish underscore data [phish underscore data. Label == “bad”]
- Good underscore is equal to phish underscore data [phish underscore data. Label == “good”]
- Bad underscore sites.head()
- defplot underscore wordcloud(text,mask is equal to None,max underscore words is equal to 400,max underscore font underscore size is equal to 120,figure underscore size is equal to (24.0,16.0),
- title is equal to None,title underscore size is equal to 40,image underscore color is equal to False):
- stopwords is equal to set(STOPWORDS)
- More underscore stopwords is equal to {'com','http'}
- stopwords is equal to stopwords.union(more is equal to stopwords)

- wordcloud is equal to WordCloud(background_color is equal to 'white',
- stopwords is equal to stopwords,
- Max words is equal to max_words,
- Max font size is equal to max_font_size,
- Random state is equal to 42,
- Mask is equal to mask)
- wordcloud.generate(text)
-
- plt.figure(figsize is equal to figure_size)
- image_color:
- Image colors is equal to ImageColorGenerator(mask);
- plt.imshow(wordcloud.recolor(color_func is equal to image_colors),interpolation is equal to "bilinear");
- plt.title(title,fontdict is equal to {'size':title_size,
- 'verticalalignment':'bottom'})
- else:
- plt.imshow(wordcloud);
- plt.title(title,fontdict is equal to {'size':title_size,'color':'green',
- 'verticalalignment':'bottom'})
- plt.axis('off');
- plt.tight_layout()
- data is equal to good_sites.text sent
- data.reset_index(drop is equal to True, inplace=True)
- In [28]:
- Common text is equal to str(data)

- Common underscore mask is equal to `np.array(Image.open('star.png'))`
- Plot underscore wordcloud(common underscore text, common underscore mask, max underscore words is equal to 400, max underscore font underscore size is equal to 120,
- `title` is equal to 'Most common words use in good urls', title underscore size is equal to 15)
- We will show that all the redirect links. We are using that chrome driver.
- Browser is equal to `webdriver.chrome(r"chromedriver.exe")`
- After this we are using BeautifulSoup and it will gather the all HTML code from its page source.
- 1. `forurlinlist` is equal to `urls`:
- 2. `browser.get(url)`
- 3. `souphis` equal to `BeautifulSoup(browser.page underscore source, "html.parser")`
- 4. `forlineinsoup.find underscore all('a')`:
- 5. `hrefis` equal to `line.get('href')`
- 6. `Links` is equal to `with` is equal to `text.append([url,href])`
- We gather the all html code from its page source and it is getting all the anchor tags. So we will get the entire hidden link which will hacker use to redirect any users to this server and we create a data frame of this links.
- `GAis` equal to `nx.from underscore pandas underscore edgelist(df,source is equal to "from",target is equal to "to")`
- `nx.draw(GA,with underscore labelsis equal to False)`
- Scores underscore `ml` is equal to `{ }`
- Scores underscore `ml['Logistic Regression']` is equal to `np.round(lr.score(testX,testY),2)`
- In [46]:
- `print("Training Accuracy :",lr.score(trainX,trainY))`
- `print("Testing Accuracy :",lr.score(testX,testY))`

- con underscore mat is equal to `pd.DataFrame(confusion_matrix(lr.predict(testX), testY),`
- `columns` is equal to `['Predicted:Bad', 'Predicted:Good'],`
- `index` is equal to `['Actual:Bad', 'Actual:Good']`)
- `print("\nCLASSIFICATION REPORT\n")`
- `print(classification_report(lr.predict(testX), testY,`
- `target_names=['Bad','Good']))`
- `print("\nCONFUSION MATRIX')`
- `plt.figure(figsize=(6,4))`
- `sns.heatmap(con_mat, annot=True,fmtis='d',cmapi="YlGnBu")`
- `Scores_ml['MultinomialNB']` is equal to `np.round(mnb.score(testX,testY),2)`
- In [51]:
- `print('Training Accuracy :',mnb.score(trainX,trainY))`
- `print('Testing Accuracy :',mnb.score(testX,testY))`
- con underscore mat is equal to `pd.DataFrame(confusion_matrix(mnb.predict(testX), testY),`
- `columns` is equal to `['Predicted:Bad', 'Predicted:Good'],`
- `index` is equal to `['Actual:Bad', 'Actual:Good']`)
- `print("\nCLASSIFICATION REPORT\n")`
- `print(classification_report(mnb.predict(testX), testY,`
- `target_names=['Bad','Good']))`
- `print("\nCONFUSION MATRIX')`
- `plt.figure(figsize=(6,4))`
- `sns.heatmap(con_mat, annot=True,fmtis='d',cmapi="YlGnBu")`

- `print('Training Accuracy :',pipeline_ls.score(trainX,trainY))`
- `print('Testing Accuracy :',pipeline_ls.score(testX,testY))`
- `con underscore matis equal topd.DataFrame(confusion underscore matrix(pipeline underscore ls.predict(testX),testY),`
- `columnsis equal to['Predicted:Bad','Predicted:Good'],`
- `['Actual:Bad','Actual:Good'])`
- `print('\nCLASSIFICATION REPORT\n')`
- `print(classification underscore report(pipeline underscore ls.predict(testX),testY,`
- `target underscore namesis equal to['Bad','Good']))`
- `print('\nCONFUSION MATRIX')`
- `plt.figure(figsizeis equal to(6,4))`
- `sns.heatmap(con underscore mat,annotis equal toTrue,fmtis equal to'd',cmapis equal to"YlGnBu")`

3.5 Model Development

In our project we have used FastAPI which is a python framework and import many libraries for different purposes. We have taken two algorithms which is LogisticRegression and MultinomialNB. LogisticRegression will predict the links are good or not and MultinomialNB work well with nlp data (natural language process). Then we have used some classification problems by using CountVectorizer and tokenizer. We have used someanother visualization. We can show that what is the hidden link in the phishing site which will redirect to another server. Then we have networkx it is creating a data structure, dynamic function and more. We are combining three datasets which we collected from several sites then we combine this dataset into one frame. The usability of this dataset is 10.0 which means very good. The data size is approx 30 mb. The data contains more than 5 lakhs

unique approach. The label column means that its prediction column in which there were two categories first is good and second is bad. After that we have checked the imbalanced of target column. Now we have a data, we convert URLs into vector form. We have used regular expression tokenizers which divide the string using regular expression. So, in our code we are just splitting only alphabets and some URLs have numbers, dots , slash etc which are not important our data. So we only gather the string and simultaneously we have transformed this in all the rows. After converting into words we used snowball it's an nltk API (natural language toolkit) which is used to string words. It will remove all the English works and create some root words. Root words means that it will combine the common words like pictures, photos for this two words it will create the one word. Phishing data text streamer is equal to all the holder that list is words lists are converted into streamers. Then we join all the lists words into single sentence. We also use word cloud. In our code we have used this to convert most repeated word into the word cloud form. Then we use chrome webdriver. This will create a new window of that chrome. So to this new chrome we will pass that link. Then by using BeautifulSoup, we gather the all html code from its page source and it is getting all the anchor tags. So we will get the entire hidden link which will hacker use to redirect any users to this server and we create a data frame of this links. So it will give a two links: first is what we passed to this and second what we are getting from this link. Logistic

regression object and we fit it by trainX, trainY. After that we checked the score and we are getting very good score which is 90.96. After that we just created the confusion matrix to see the actual prediction and normal prediction. Using Logistic Regression we are creating a pipeline. Then we are saving this pipeline model using pickle and we check the accuracy of it and it is giving very good accuracy.

We have done the featured extraction where we have saw that each category is described properly and we have done implementation for that. So, all these features are based on URL structured

Chapter 04: Performance Analysis

In this project, we have done various implementations for training and we have predicted phishing websites whether they are good URLs or bad URLs. As we have used some of the supervised algorithm.

- Firstly, we imported some libraries such as pandas, numpy, multinomialNB, LogisticRegression and many more.
- We have examined and pre-process the dataset and divide the datasets into training and test sets. Then we combine all the dataset into one frame.

```
In [2]: import pandas as pd # use for data manipulation and analysis
import numpy as np # use for multi-dimensional array and matrix

import seaborn as sns # use for high-level interface for drawing attractive and informative statistical graphics
import matplotlib.pyplot as plt # It provides an object-oriented API for embedding plots into applications
%matplotlib inline
# It sets the backend of matplotlib to the 'inline' backend:
import time # calculate time

from sklearn.linear_model import LogisticRegression # algo use to predict good or bad
from sklearn.naive_bayes import MultinomialNB # nlp algo use to predict good or bad

from sklearn.model_selection import train_test_split # splitting the data between feature and target
from sklearn.metrics import classification_report # gives whole report about metrics (e.g, recall, precision, f1_score, c_m)
from sklearn.metrics import confusion_matrix # gives info about actual and predict
from nltk.tokenize import RegexpTokenizer # regexp tokenizers use to split words from text
from nltk.stem.snowball import SnowballStemmer # stemmes words
from sklearn.feature_extraction.text import CountVectorizer # create sparse matrix of words using regextokenizes
from sklearn.pipeline import make_pipeline # use for combining all preroceessors techniuges and algos

from PIL import Image # getting images in notebook
# from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator# creates words colud

from bs4 import BeautifulSoup # use for scraping the data from website
from selenium import webdriver # use for automation chrome
import networkx as nx # for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks

import pickle# use to dump model

import warnings # ignores pink warnings
warnings.filterwarnings('ignore')
```

Fig 4.1 importing the libraries

```
In [2]: phishing_data1 = pd.read_csv('phishing_urls.csv',usecols=['domain','label'],encoding='latin1', error_bad_lines=False)
phishing_data1.columns = ['URL','Label']
phishing_data2 = pd.read_csv('phishing_data.csv')
phishing_data2.columns = ['URL','Label']
phishing_data3 = pd.read_csv('phishing_data2.csv')
phishing_data3.columns = ['URL','Label']

In [3]: for l in range(len(phishing_data1.Label)):
if phishing_data1.Label.loc[l] == '1.0':
    phishing_data1.Label.loc[l] = 'bad'
else:
    phishing_data1.Label.loc[l] = 'good'
```

Fig 4.2 Combining dataset into one frame

- After combining into one frame we created creator and then save into my drive.

```
In [7]: phish_data.head()
```

```
Out[7]:
```

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/websrcr...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad

Fig 4.3 Dataset

```
In [10]: phish_data.isnull().sum() # there is no missing values

Out[10]: URL      0
          Label    0
          dtype: int64
```

Fig 4.4 Missing value

```
label_counts = pd.DataFrame(phish_data.Label.value_counts())

In [20]: #visualizing target_col
          sns.set_style('darkgrid')
          sns.barplot(label_counts.index, label_counts.Label)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x15e4f1821f0>
```

Fig 4.5 Visualizing

- We have used regular expression tokenizers which divide the string using regular expression. So, in our code we are just splitting only alphabets and some URLs have numbers, dots, slash etc which are not important our data. So we only gather the string and simultaneously we have transformed this in all the rows.

```

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_tpl=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',
          'd',
          'dca',
          'cce',
          'f',
          'index',
          'php',

```

Fig 4.6 Regular Expression

```

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

In [20]: phish_data.sample(5)

Out[20]:

```

	URL	Label	text_tokenized	text_stemmed
314008	detroitlionsticket.com/	good	[detroitlionsticket, com]	[detroitlionsticket, com]
502478	pertclinic.com/qsipz9g	bad	[pertclinic, com, qsipz, g]	[pertclin, com, qsipz, g]
423029	revolutionmyspace.com/pictures-1/ami_dolenz	good	[revolutionmyspace, com, pictures, ami, dolenz]	[revolutionmyspac, com, pictur, ami, dolenz]
236352	ryanscottfrost.com/	good	[ryanscottfrost, com]	[ryanscottfrost, com]
227770	photobucket.com/images/matsumoto%20jun/	good	[photobucket, com, images, matsumoto, jun]	[photobucket, com, imag, matsumoto, jun]

Fig 4.7 SnowballStemmer

- After converting into words we used snowball it's an nltk API (natural language toolkit) which is used to string words. It will remove all the English works and create some root words. Root words means that it will combine the common words

like pictures, photos for this two words it will create the one word. Phishing data text streamer is equal to all the holder that list is words lists are converted into streamers.

- We also use word cloud. In our code we have used this to convert most repeated word into the word cloud form. Then we use chrome webdriver. This will create a new window of that chrome. So to this new chrome we will pass that link.

```
In [23]: #slicing 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/webscr...	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 webscr...
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()
```

Fig 4.8 WordCloud


```
In [26]: 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()

In [27]: data = good_sites.text_sent
data.reset_index(drop=True, inplace=True)
```

Fig 4.9 Networkx

```
cv = CountVectorizer()

In [21]: #help(CountVectorizer())

In [39]: feature = cv.fit_transform(phish_data.text_sent) #transform all text which we tokenize

In [40]: feature[:5].toarray() # convert sparse matrix into array to print transformed features
Out[40]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)

* Splitting the data

In [41]: trainX, testX, trainY, testY = train_test_split(feature, phish_data.Label)
```

Fig 4.10 create model

- Using Logistic Regression we are creating a pipeline. Then we are saving this pipeline model using pickle and we check the accuracy of it and it is giving very good accuracy.

```
In [45]: Scores_m1 = {}
          Scores_m1['Logistic Regression'] = np.round(lr.score(testX,testY),2)

In [46]: print('Training Accuracy :',lr.score(trainX,trainY))
          print('Testing Accuracy :',lr.score(testX,testY))
          con_mat = pd.DataFrame(confusion_matrix(lr.predict(testX), testY),
                                columns = ['Predicted:Bad', 'Predicted:Good'],
                                index = ['Actual:Bad', 'Actual:Good']))

          print('\nCLASSIFICATION REPORT\n')
          print(classification_report(lr.predict(testX), testY,
                                     target_names =['Bad', 'Good']))

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

Training Accuracy : 0.9782480479795345
Testing Accuracy : 0.9636514559077306
```

Fig 4.11 Logistic Regression

- MultinomialNB work well with nlp data (natural language process). Multinomial is giving us 95 which are not good as a logistic regression but fine.

```
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/temp/libraries/ipad','tubemoviez.exe','svision-online.de/
predict_good = ['youtube.com/', 'youtube.com/watch?v=qI0TQJi3vdu', 'retailhellunderground.com/', 'restorevisioncenters.com/html/technology.html']
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']
```

Fig 4.14 Good and Bad links

4.2 Result

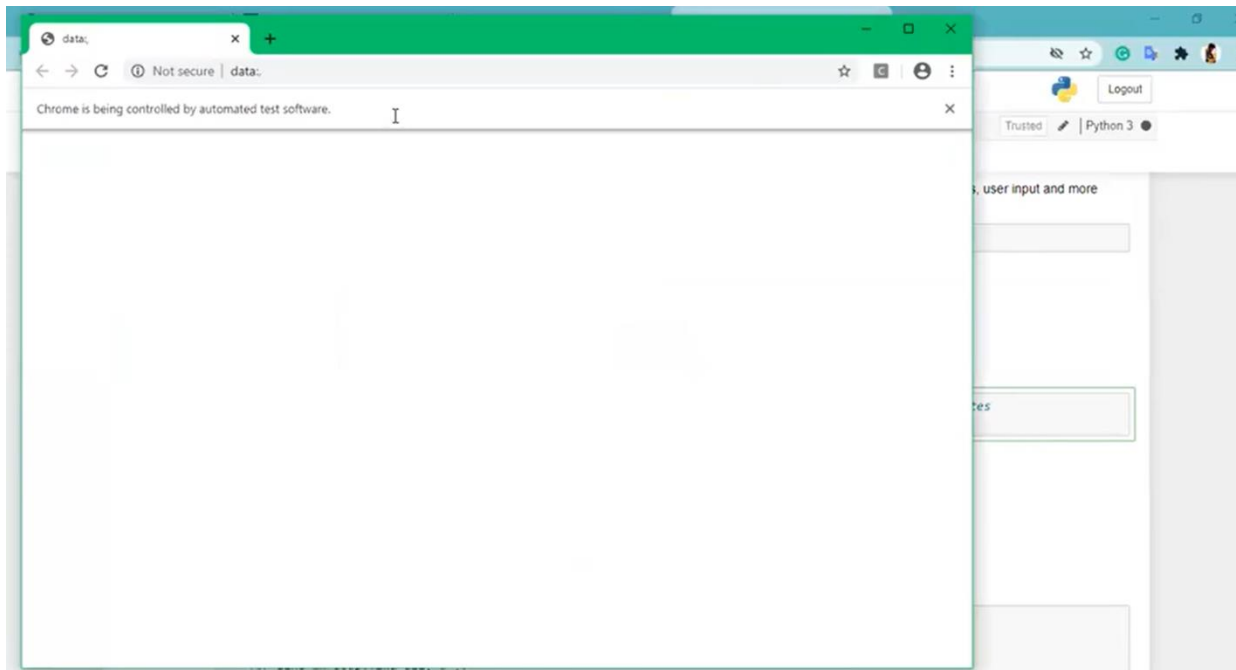


Fig 4.2.1 new chrome window

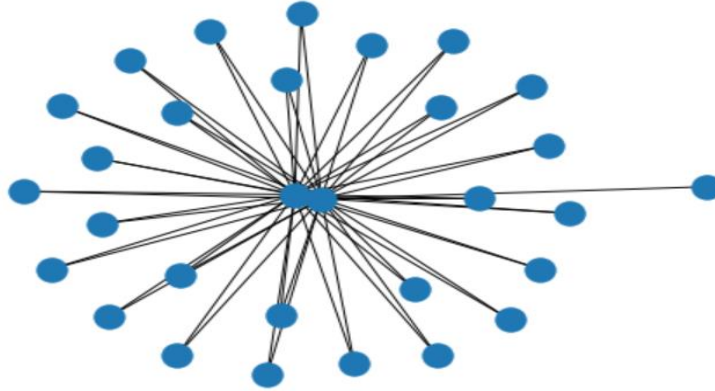


Fig 4.2.2 graph of networkx

```
Out[8]:
```

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	iFrame
0	graphicriver.net	0	0	1	1	0	0	0	0	0	1	1	1	0
1	ecnavijp	0	0	1	1	1	0	0	0	0	1	1	1	0
2	hubpages.com	0	0	1	1	0	0	0	0	0	1	0	1	0
3	extratorrent.cc	0	0	1	3	0	0	0	0	0	1	0	1	0
4	icicibank.com	0	0	1	3	0	0	0	0	0	1	0	1	0

Fig 4.2.3 Testing

```
In [23]: #slicing 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/70ffb52d079109dca5664cce6f3173782/...	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/webscr...	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 webscr...
2	serviciosbys.com/paypal.cgi.bin.get-into.herf...	bad	[serviciosbys, com, paypal, cgi, bin, get, into...	[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()
```

Fig 4.2.4 Tokenizers

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x15e4f1821f0>

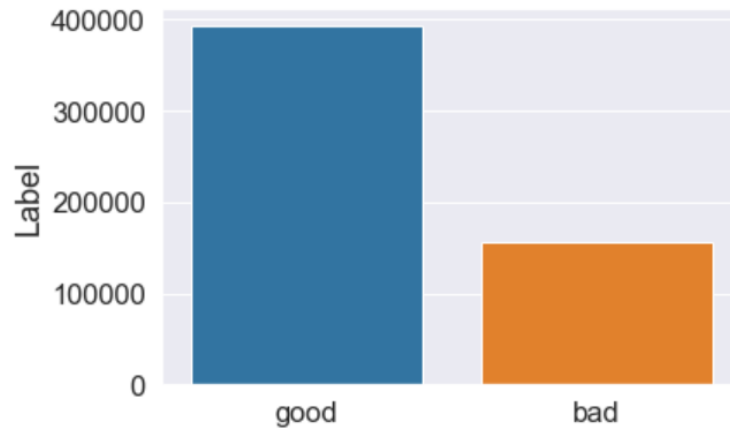


Fig 4.2.5 Classification of imbalanced

```
Out[15]: ['nobell',  
'it',  
'ffb',  
'd',  
'dca',  
'cce',  
'f',  
'login',  
'SkyPe',  
'com',  
'en',  
'cgi',  
'bin',  
'verification',  
'login',  
'ffb',  
'd',  
'dca',  
'cce',  
'f',  
'index',  
'php',  
'cmd',  
'profile',  
'ach',  
'outdated',  
'page',  
'tpl',  
'p',  
'gen',  
'failed',  
'to',  
'load',  
'nav',  
'login',
```

Fig 4.2.6 Similar words which we use

Fig 4.2.9 Common good words



Fig 4.2.10 Common bad words

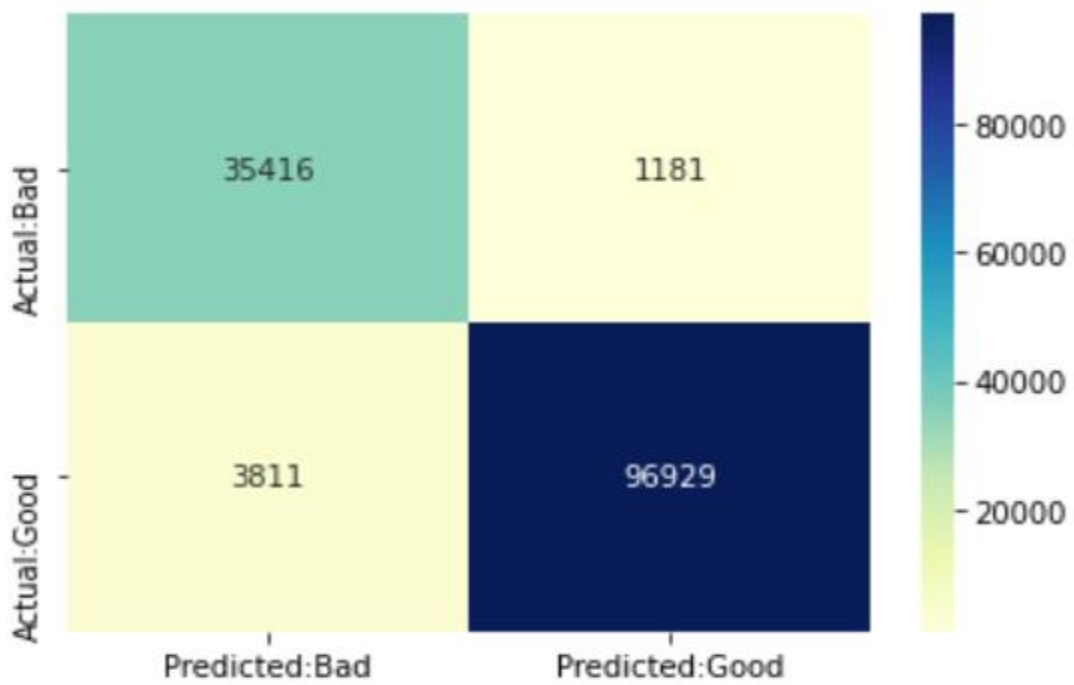


Fig 4.2.11 prediction

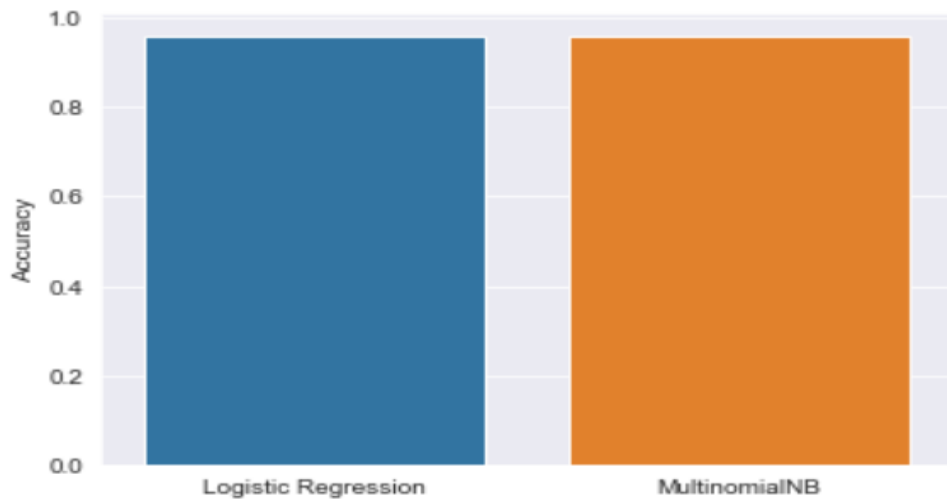


Fig 4.2.12 prediction between logistic regression and multinomialnb

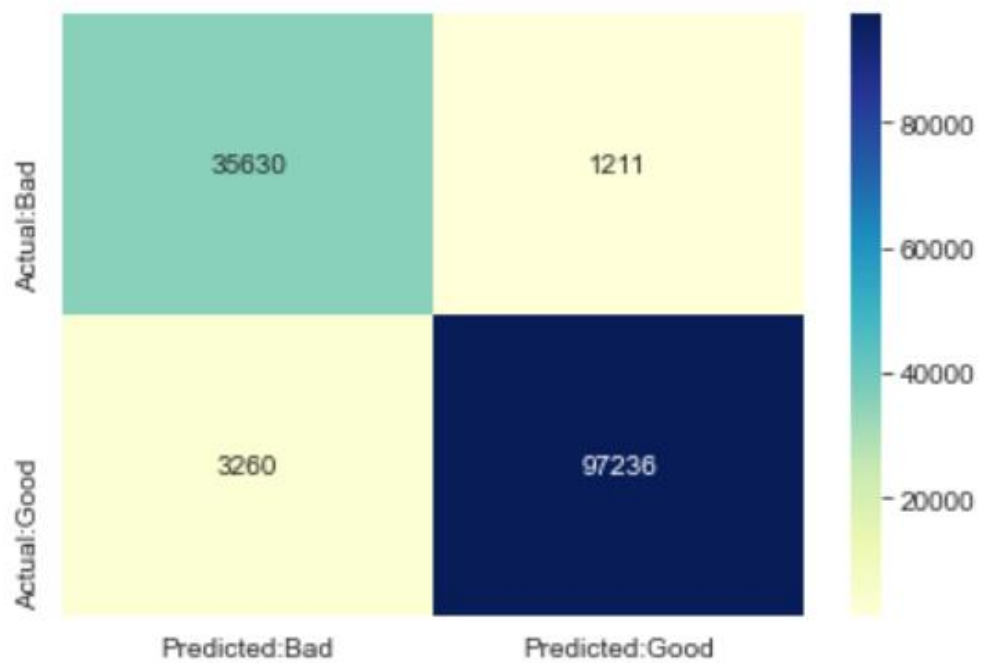


Fig 4.2.13 Confusion matrix

Chapter 05: CONCLUSIONS

In our project we have used FastAPI which is a python framework and import many libraries for different purposes. We have taken two algorithms which is LogisticRegression and MultinomialNB. LogisticRegression will predict the links are good or not and MultinomialNB work well with nlp data (natural language process). Then we have used some classification problems by using CountVectorizer and tokenizer. We have used someanother visualization. We can show that what is the hidden link in the phishing site which will redirect to another server. Then we have networkx it is creating a data structure, dynamic function and more. We are combining three datasets which we collected from several sites then we combine this dataset into one frame. The usability of this dataset is 10.0 which means very good. The data size is approx 30 mb. The data contains more than 5 lakhs unique approach. The label column means that its prediction column in which there were two categories first is good and second is bad. After that we have checked the imbalanced of target column. Now we have a data, we convert URLs into vector form. We have used regular expression tokenizer which divide the string using regular expression. So, in our code we are just splitting only alphabets and some URLs have numbers, dots , slash etc which are not important our data. So we only gather the string and simultaneously we have transformed this in all the rows. After converting into words we used snowball it's an nltk API (natural language toolkit) which is used to string words. It will remove all the English

works and create some root words. Root words means that it will combine the common words like pictures, photos for this two words it will create the one word. Phishing data text streamer is equal to all the holder that list is words lists are converted into streamers. Then we join all the lists words into single sentence. We also use word cloud. In our code we have used this to convert most repeated word into the word cloud form. Then we use chrome webdriver. This will create a new window of that chrome. So to this new chrome we will pass that link. Then by using BeautifulSoup, we gather the all html code from its page source and it is getting all the anchor tags. So we will get the entire hidden link which will hacker use to redirect any users to this server and we create a data frame of this links. So it will give a two links : first is what we passed to this and second what are we getting from this link. Logistic regression object and we fit it by trainX, trainY. After that we checked the score and we are getting very good score which is 90.96. After that we just created the confusion matrix to see the actual prediction and normal prediction. Using Logistic Regression we are creating a pipeline. Then we are saving this pipeline model using pickle and we check the accuracy of it and it is giving very good accuracy.

5.2Future Scope

Through this project, one could recognize plenty approximately the phishing web sites and how they're differentiated from legitimate ones. This project may be taken in addition through developing browser extensions of growing a GUI. These have to classify the inputted URL to legitimate or phishing with the use of the stored model.

REFERENCES

- 1) https://www.researchgate.net/profile/Rishikesh-Mahajan/publication/328541785_Phishing_Website_Detection_using_Machine_Learning_Algorithms/links/5d0397fd92851c9004394af4/Phishing-Website-Detection-using-Machine-Learning-Algorithms.pdf
- 2) Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017
- 3) Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection : In ACM,2018 https://www.researchgate.net/profile/Er-Purvi-Pujara/publication/331198983_Phishing_Website_Detection_using_Machine_Learning_A_Review/links/5c6bd4ae4585156b5706e727/PhishingWebsite-Detection-using-Machine-Learning-A-Review.pdf
- 4) Sahingoz, O.K., Buber, E., Demir, O. and Diri, B., 2019. Machine learning based phishing detection from URLs. Expert Systems with Applications, 117, pp.345-357.
- 5) Sci-kit learn, SVM library. <http://scikit-learn.org/stable/modules/svm.html>.
- 6) <https://www.unb.ca/cic/datasets/url-2016.html>
- 7) Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013

APPENDICES

Raw code:

```
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)
```