Spam Filter & Malicious URL Prediction

A Project
Report
Presented to
CMPE-272
Fall, 2021

By
Team Dragonite
Khushil Modi (015923115)
Sarjak Patel (015945046)
Nevil Shah (015964975)

12/18/2021

ABSTRACT

Spam Filter & Malicious URL Prediction

By

Khushil Modi

Sarjak Patel

Nevil Shah

The Spam Filter and Malicious URL Prediction is a web-based application that is designed specifically for managing data that comes into emails. There are a lot of unsolicited emails that people receive in bulk throughout the day. So, to counter spam emails we have made a spam filter using Machine learning models like RNN to train our model and it would test whether a message is spam or not when a person inputs text.

Similarly, we created another feature within the same application for predicting and detecting whether a URL that you have received in any email, messages etc. is safe or malicious. For preventing users from getting hacked and be the victims of phishing attacks it is very important that they can test any URL to see whether it is safe or not.

A spam filter is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to a user's inbox. Malicious websites are one of the most common cybersecurity threats. We have created a filter model which will detect if a link is malicious or safe so that users don’t get victim of phishing attacks.

Acknowledgements

We would like to express our deep sense of gratitude and sincere appreciation to our respected and esteemed Prof. Chandrasekar Vuppalapati, Assistant Professor, Computer Engineering Department, San Jose State University (SJSU) for his constant support and encouragement regarding the project work throughout this semester.

Yours sincerely,
Khushil Modi
Sarjak Patel
Nevil Shah

Table of Contents

- **Introduction - Project Description**
- **Requirements**
- **KDD**
- **Feature Engineering**
- **Data Science Algorithms & Features Used**
- **High Level Architecture Design**
- **DFD & Component Level Design**
- **Client Side Design**
- **Model Deployment**
- **Model Testing**
- **References**

Introduction – Project Description

1. Spam Filter: A spam filter is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to a user's inbox. If a user enters a text message, Spam Filter will detect if the message is spam or not.

2. Malicious Link detector: Malicious URL or link is a link created with the purpose of promoting scams, attacks, and frauds. If a user enters a link in the text-box, Spam filter will notify as output if the link is malicious or not.

Requirements

Req. 1) Users should be able to differentiate between secure and spam emails and links using our system.

Req. 2) The system should be able to differentiate between normal and spam emails and links.

Req. 3) The system should be at least 85% accurate.

Req. 4) The system should be able to predict and display a message whether a link is good or bad based on a spam filter.

Req. 5) Similarly, the system should also be able to predict and display a message whether any email is normal or spam.

Links eg: www.fakebillionsonline.blogspot.com - Spam

www.facebook.com/login - Not Spam

Emails eg: Congratulations! You have won a lottery of 1 million dollars in a lucky draw conducted in the UK. – Spam

Congratulations! You have been hired for the job that you applied for! – Not Spam

Knowledge Discovery in Database (KDD)

1) Selection: We have selected Spam and URL Dataset.
2) Preprocessing: We will preprocess the dataset by removing irrelevant data and dropping the null values and the data which is not in proper format.
3) Transformation: Transforming the data using CountVectorizer, Vectorizer, TF-IDF etc methods
4) Data Mining: We will reduce the dimensions of the dataset so that focus is only on the data that is really important and will create a model for the similar.
5) Interpretation/Evaluation: Will interpret the mined patterns and evaluate the efficiency of the model
6) Knowledge Representation: Will represent the results obtained maybe using visualization techniques.

Feature Engineering

<u>Feature 1: Link Detection</u>

<u>Phishing :</u> The practice of tricking Internet users (as through the use of deceptive email messages or scam links) into revealing personal or confidential information like bank details or personal sensitive information which can then be used illicitly is called phishing.

<u>Link Detection:</u>

Link Detection is the mechanism which will filter out phishing links and the good ones to save a person's sensitive information like credit card details from getting stolen and misused for any malicious purposes.

Example:

a)Good link:- https://www.facebook.com/login/

This is a good link asking for credentials to login into one's Facebook Account and the entered credentials in this link will be safe as this page is authorized and secured by facebook.

b) Phished link: faceboook.com/login/

The above link looks kind of similar to the actual link used by Facebook and the webpage will look similar to the real page , but this is a phishing link and the hackers can get the credentials entered in this vitriolic link and use them for their evil intentions.

<u>Feature 2:- Spam Detection</u>

Spam Detection is a binary classification problem whose aim is to unsolicited , unwanted and virus-infected emails to prevent spam messages from creeping into the user's inbox.

Examples:

Spam messages:

1) Free 1 week holiday trip to Vegas

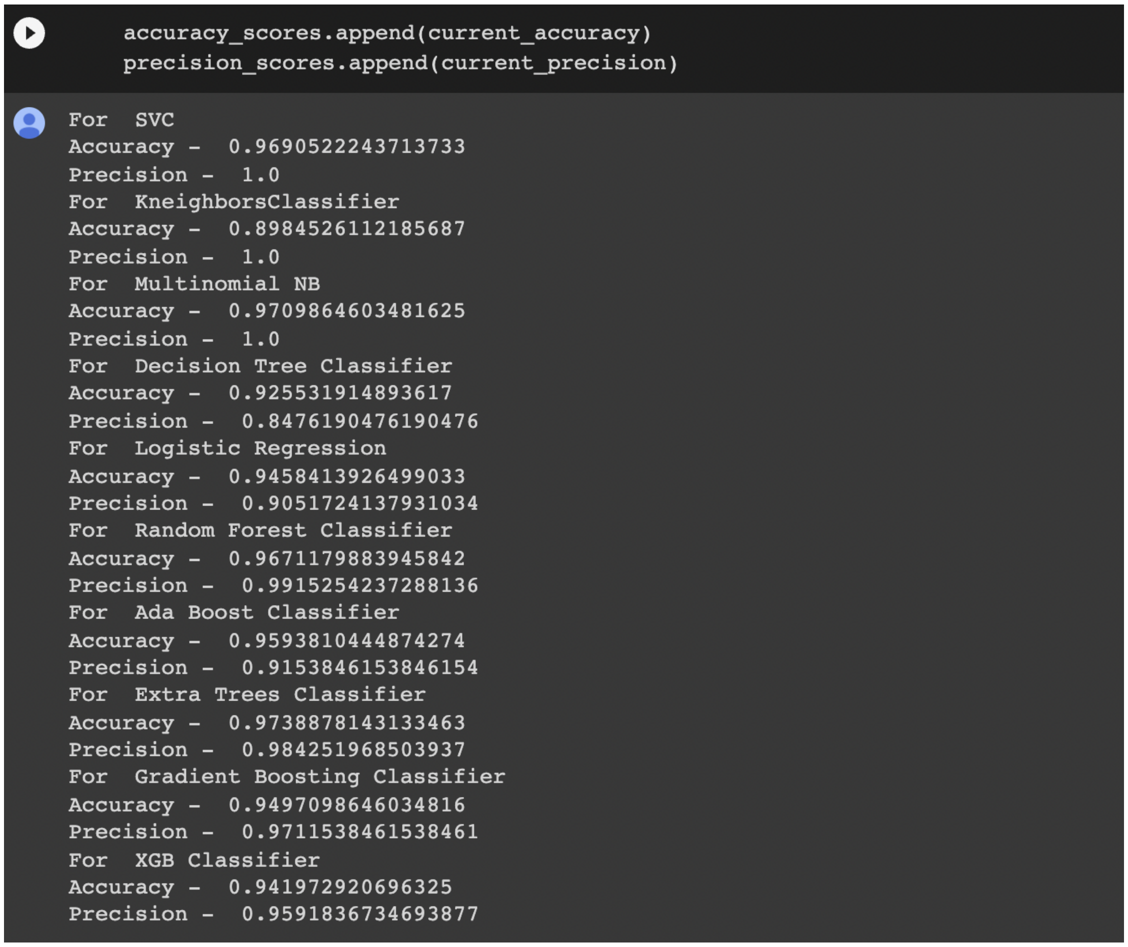
2) Congratulations! You are eligible to receive a free COVID vaccine. To Confirm, click on the below again.

Data Science Algorithms:

In this project, many models are trained using different types of machine learning(ML) algorithms and as a result, the accuracy calculated over every model is different from each other. Here is the list of the algorithms used :

Algorithms:

- Logistic Regression
- MultinomialNB
- Random Forest Classifier
- AdaBoost Classifier
- Extra Trees Classifier
- Gradient Boosting Classifier
- XGB Classifier
- Support Vector Classifier(SVC)
- Decision Tree Classifier
- K Neighbor Classifier
- Support Vector Classifier



```
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
For  SVC
Accuracy -  0.9690522243713733
Precision -  1.0
For  KneighborsClassifier
Accuracy -  0.8984526112185687
Precision -  1.0
For  Multinomial NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  Decision Tree Classifier
Accuracy -  0.925531914893617
Precision -  0.8476190476190476
For  Logistic Regression
Accuracy -  0.9458413926499033
Precision -  0.9051724137931034
For  Random Forest Classifier
Accuracy -  0.9671179883945842
Precision -  0.9915254237288136
For  Ada Boost Classifier
Accuracy -  0.9593810444874274
Precision -  0.9153846153846154
For  Extra Trees Classifier
Accuracy -  0.9738878143133463
Precision -  0.984251968503937
For  Gradient Boosting Classifier
Accuracy -  0.9497098646034816
Precision -  0.9711538461538461
For  XGB Classifier
Accuracy -  0.941972920696325
Precision -  0.9591836734693877
```

Features used

While developing the back-end of the project, many features have contributed in developing and reaching a particular accuracy like:

- **NLTK library**
- **WordCloud**
- **CountVectorizer**
- **Sklearn.model_selection (for splitting the data into train and test sets.)**
- **Bar Graph and Pie Chart (Exploratory Data Analysis(EDA))**

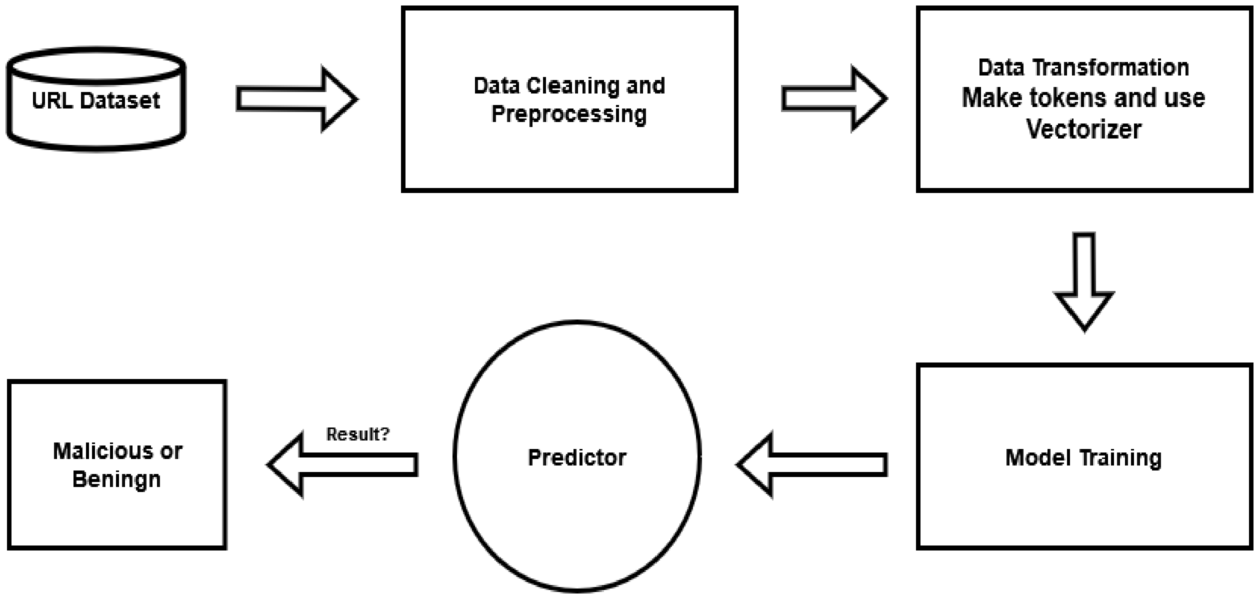
High Level Architecture Design

Talking about the architecture of the project, both the parts(Spam detector and malicious link detector) have almost the same architecture.First and foremost, step is the collection of data and storing it as a dataset. Afterwards, Appropriate Data cleaning and preprocessing is conducted in order to remove the unnecessary parts of the dataset and to obtain the highest accuracy.

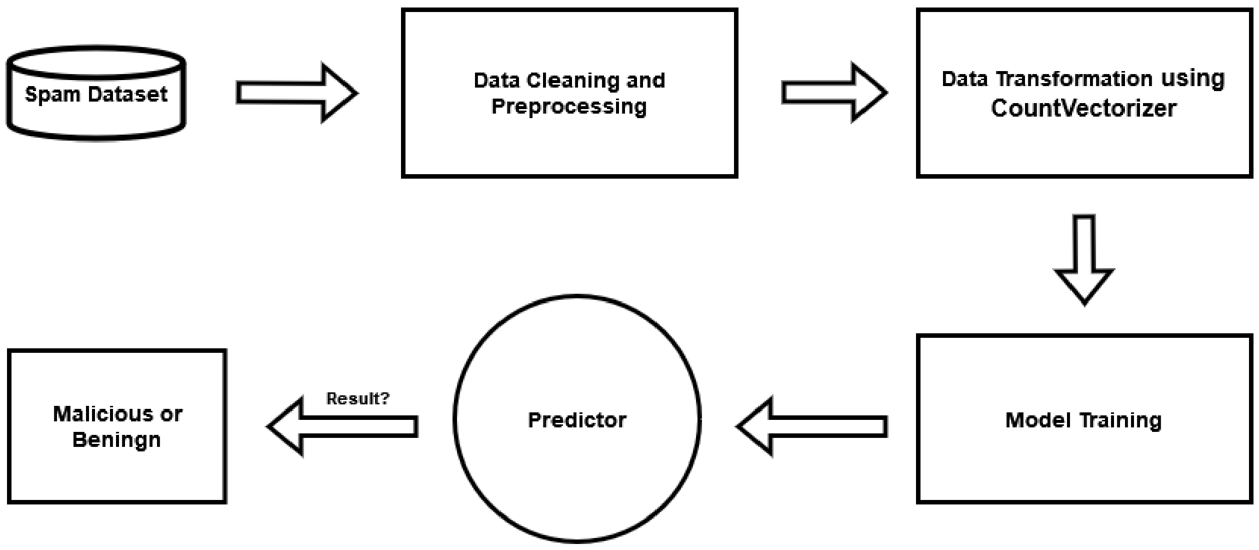
Thereafter, the messages inside the dataset are converted into the tokens and for that, either CountVectorizer or TF-IDF vectorizer is used and the process is called tokenization. This step is followed by Model training, where a model is trained using a supervised classification Machine Learning algorithm and then , the model is dumped into a pickle file.

Thereafter, The message or link passed as input by the user is given to the pickle file which will classify whether the entered input is malicious or not .

1) Malicious Link Detection:


URL Dataset
Data Cleaning and Preprocessing
Data Transformation Make tokens and use Vectorizer
Model Training
Predictor
Result?
Malicious or Beningn

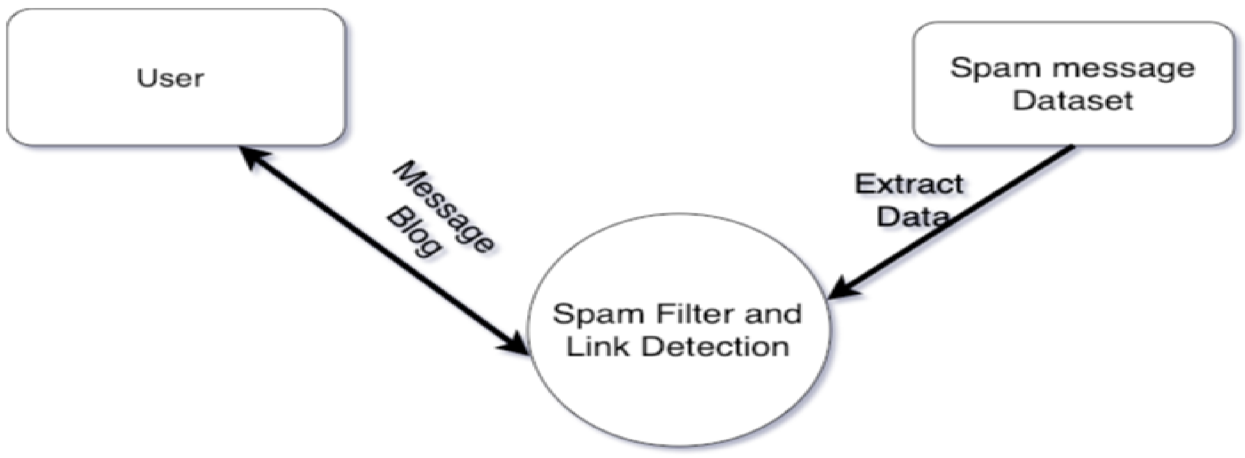
2) Spam Email Detection:


Spam Dataset
Data Cleaning and Preprocessing
Data Transformation using CountVectorizer
Model Training
Predictor
Result?
Malicious or Beningn

Data Flow Diagram

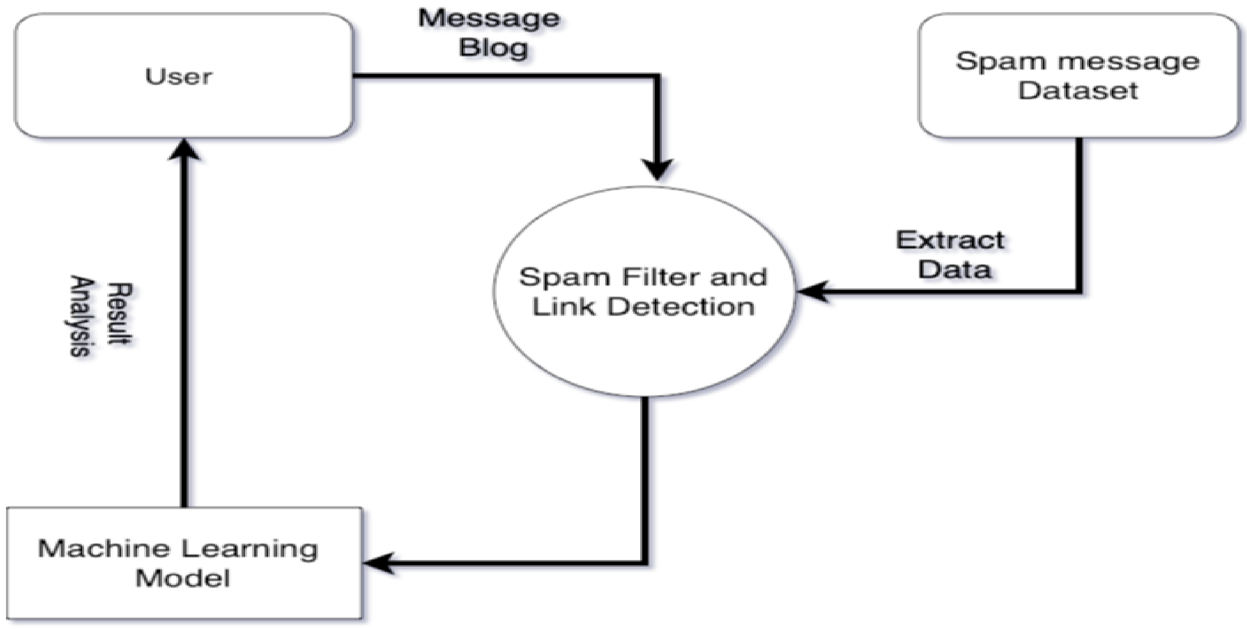
Level 0 Data-Flow Diagram:

This diagram is also known as a context diagram. With the aim to visualize the entire system at-a-glance, This diagram is designed keeping in mind the system and its characteristics with external entities.



<u>Level 1 Data-Flow Diagram</u>

Level 1 DFD is also known as the "Exploded view" of the context diagram. This diagram is designed in order to describe each and every of the system's sub-processes that combine and work together to form a complete system.Here, the context diagram is further divided into subprocesses.



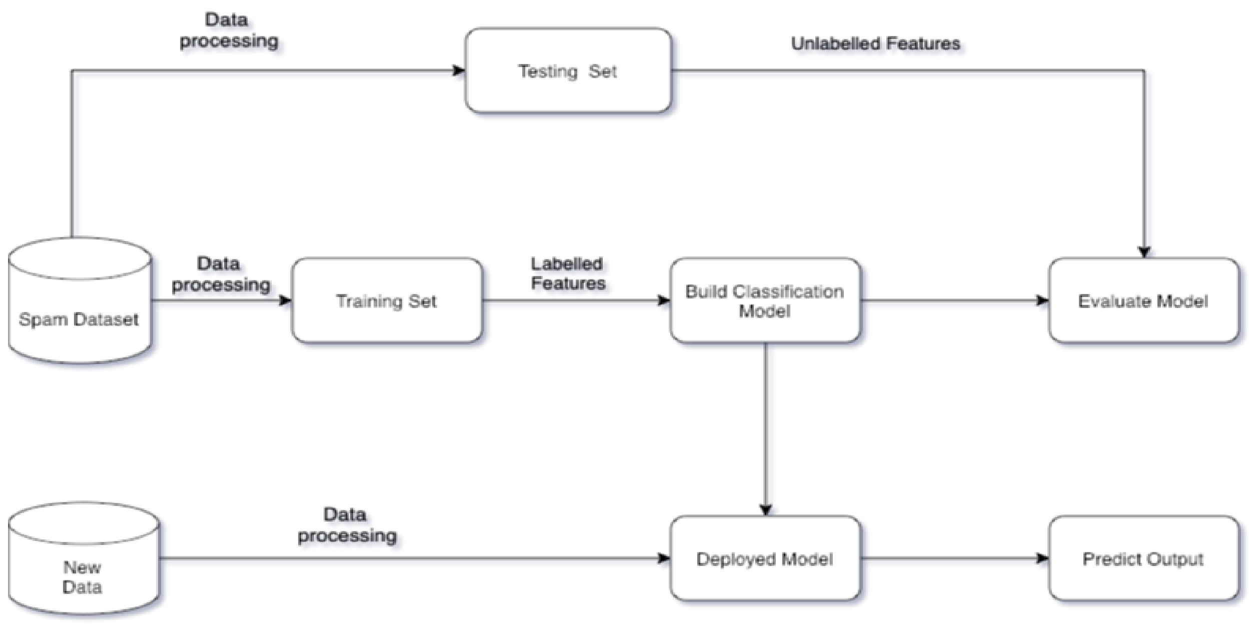
Component Level Design

After the design of the architecture, the diagram consisting of the modules and different components used inside the software is called component level design.

The diagram is almost similar to both the parts of the project.

- First and foremost, the dataset is prepared and afterwards, required techniques of data preprocessing is applied to assure the increase in overall accuracy of the model.
- Thereafter, the dataset is divided into training and testing parts , where with the help of training dataset, the model is created and built.

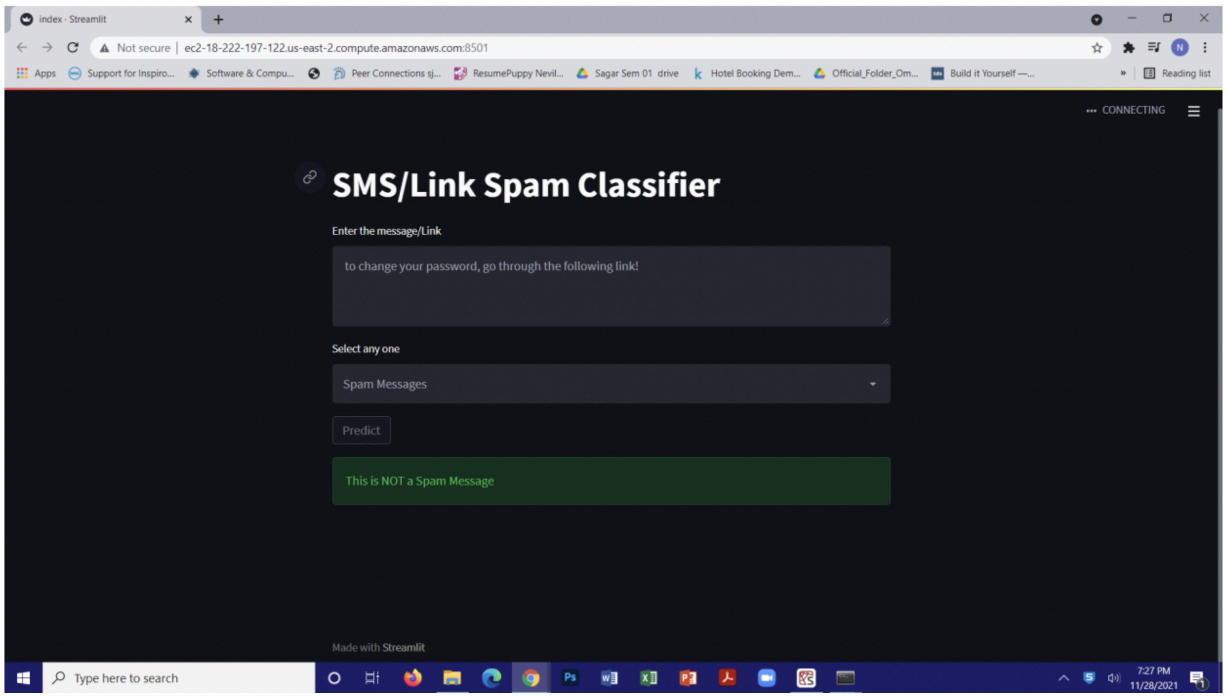
- After preprocessing the testing portion of the data, the model is evaluated using this section and thus, the accuracy of the model is calculated, which shows the peculiarity and correctness of the model.
- When the user enters a message or link , it undergoes some preprocessing tasks and thereafter, it is sent to the model as input to classify whether the given input is malign or not.



Client Side Design

- Front End is developed using :
    1. Streamlit : Streamlit is used by developers to build highly interactive web applications and custom web pages for machine learning.
    2. Python

1. Spam Detector :



SMS/Link Spam Classifier

Enter the message/Link

to change your password, go through the following link!

Select any one

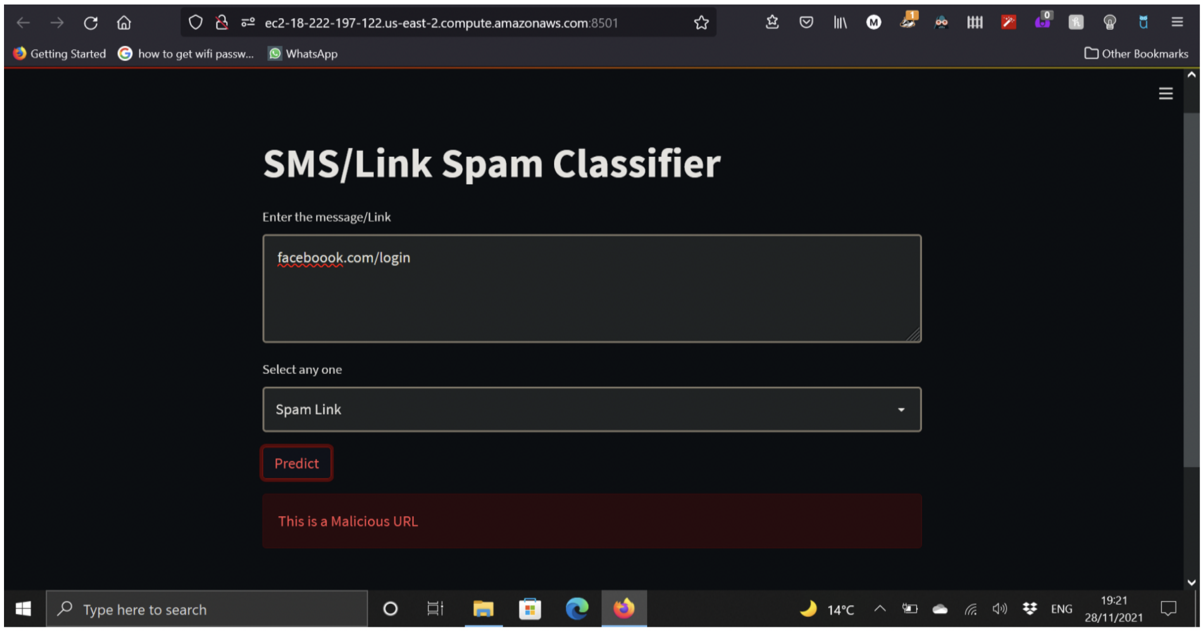
Spam Messages

Predict

This is NOT a Spam Message

2. Malicious Link Detector:

- **Malicious Links :** Malicious links are created with the aim to steal important piece of data and personal information like bank details, birth-date and much more.


SMS/Link Spam Classifier

Enter the message/Link

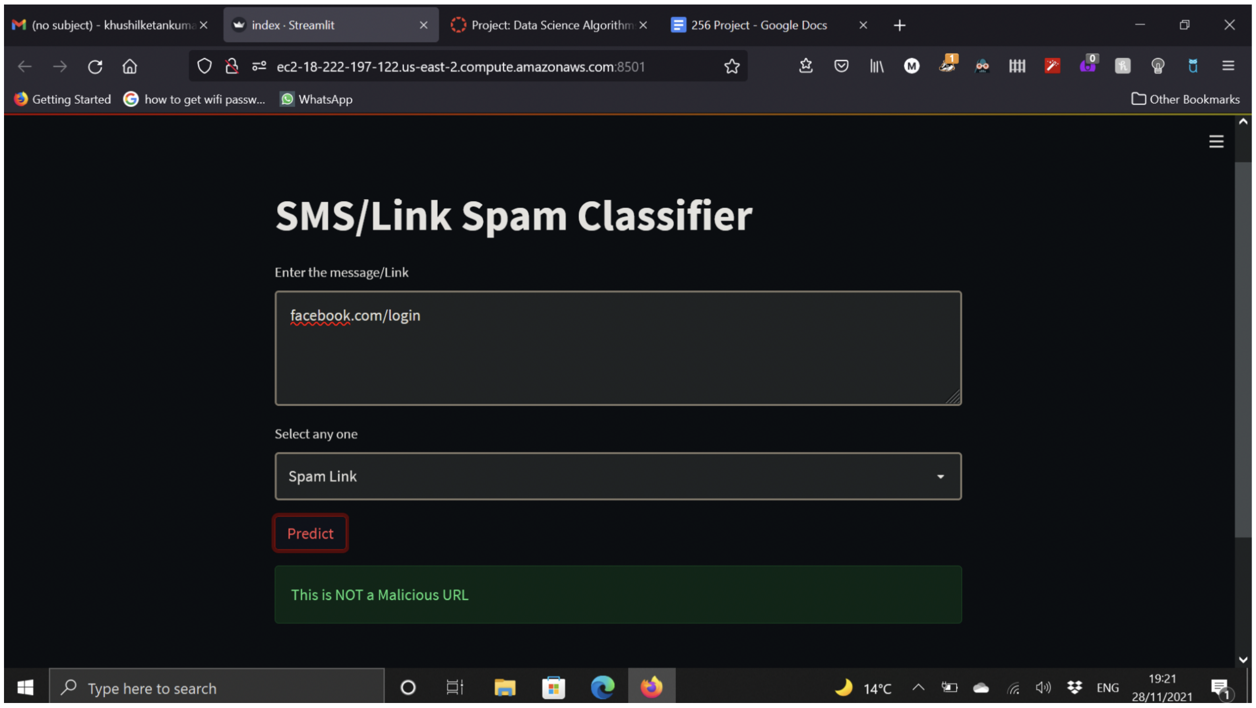
```
faceboook.com/login
```

Select any one: Spam Link

Predict

This is a Malicious URL

- **Non-malicious link:** A link which is not phised is called a good link or a Non-malicious link.


SMS/Link Spam Classifier

Enter the message/Link

```
facebook.com/login
```

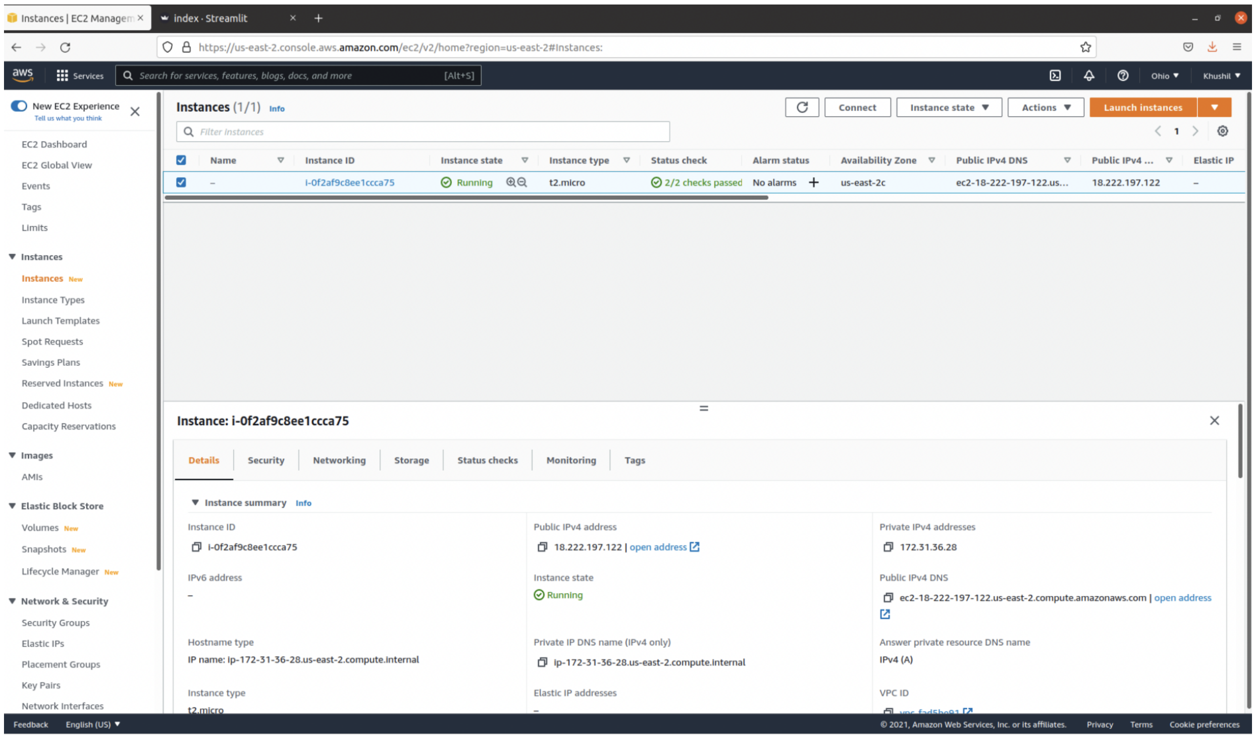
Select any one: Spam Link

Predict

This is NOT a Malicious URL

Model Deployment:

- For deployment purposes, we have created an Amazon EC2 instance and deployed the web-application in the same instance.



https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:

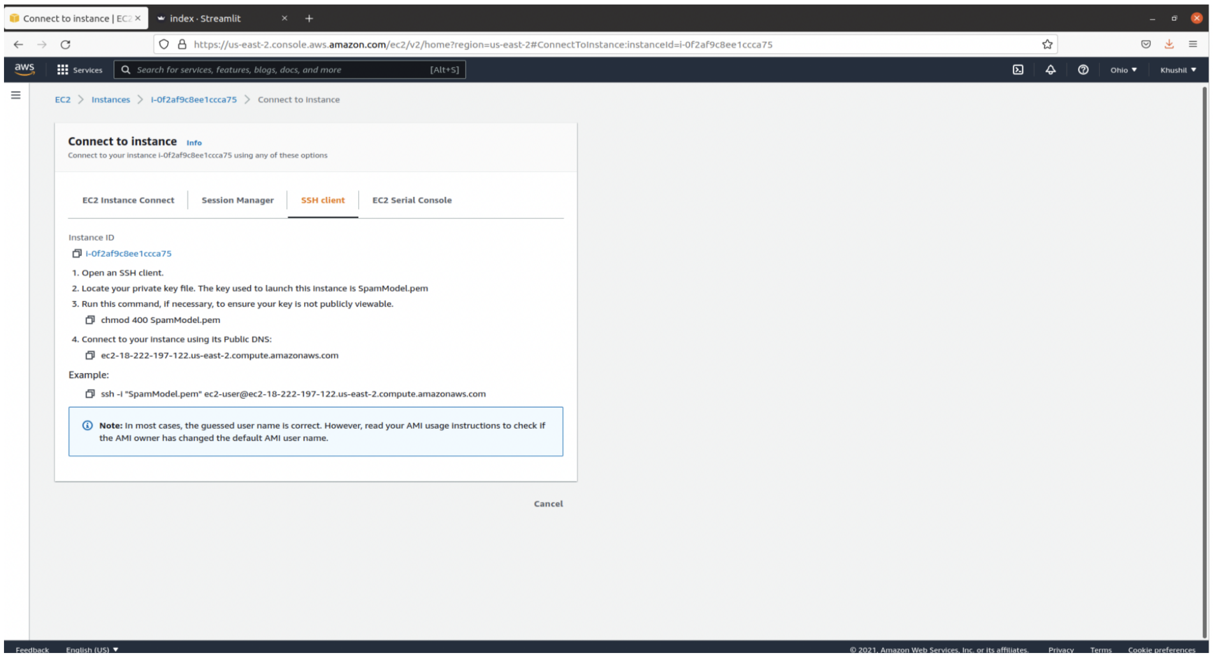
Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
–	i-0f2af9c8ee1ccca75	Running	t2.micro	2/2 checks passed	No alarms	us-east-2c	ec2-18-222-197-122.us...	18.222.197.122	–

Instance: i-0f2af9c8ee1ccca75

Instance summary

Instance ID: i-0f2af9c8ee1ccca75	Public IPv4 address: 18.222.197.122 | open address	Private IPv4 addresses: 172.31.36.28
IPv6 address: –	Instance state: Running	Public IPv4 DNS: ec2-18-222-197-122.us-east-2.compute.amazonaws.com | open address
Hostname type: IP name: ip-172-31-36-28.us-east-2.compute.internal	Private IP DNS name (IPv4 only): ip-172-31-36-28.us-east-2.compute.internal	Answer private resource DNS name: IPv4 (A)
Instance type: t2.micro	Elastic IP addresses: –	VPC ID



https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#ConnectToInstance:instanceId=i-0f2af9c8ee1ccca75

EC2 > Instances > i-0f2af9c8ee1ccca75 > Connect to instance

Connect to instance

Connect to your instance i-0f2af9c8ee1ccca75 using any of these options

SSH client

Instance ID

i-0f2af9c8ee1ccca75

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is SpamModel.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 SpamModel.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-18-222-197-122.us-east-2.compute.amazonaws.com
```

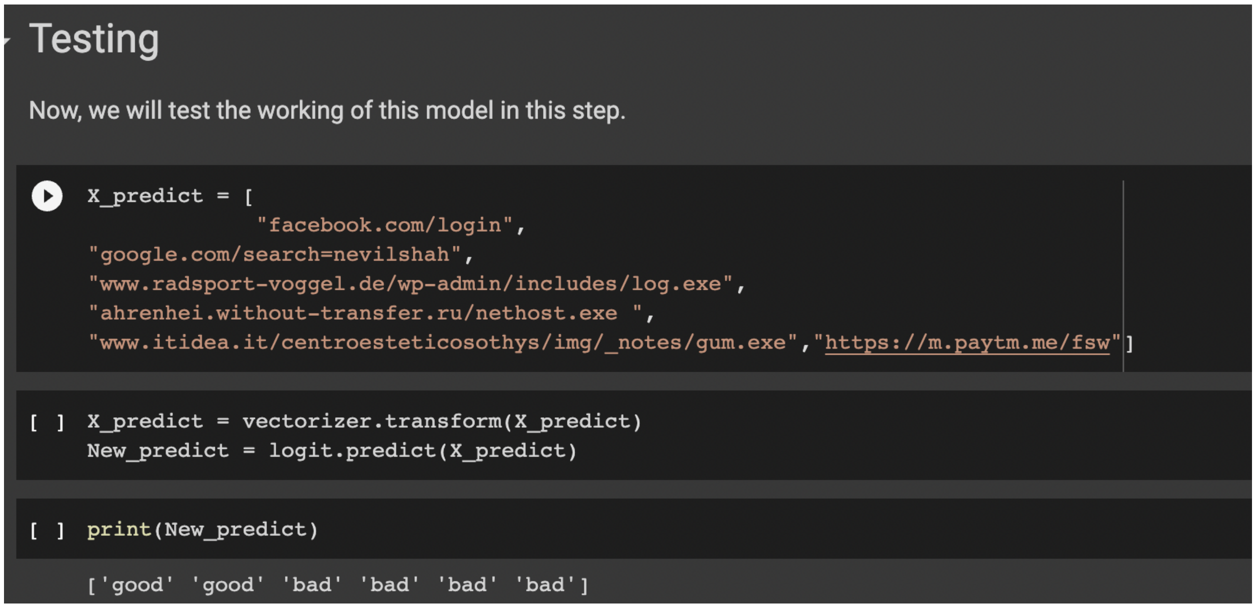
Example:

```
ssh -i "SpamModel.pem" ec2-user@ec2-18-222-197-122.us-east-2.compute.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Model Testing

After compiling the model, we would test the model by giving a message or a link as input to the trained model and check whether the model correctly and classifies the difference between malicious links or not.


Testing

Now, we will test the working of this model in this step.

```
X_predict = [
             "facebook.com/login",
"google.com/search=nevilshah",
"www.radsport-voggel.de/wp-admin/includes/log.exe",
"ahrenhei.without-transfer.ru/nethost.exe ",
"www.itidea.it/centroesteticosothys/img/_notes/gum.exe","https://m.paytm.me/fsw"]
```

```
X_predict = vectorizer.transform(X_predict)
New_predict = logit.predict(X_predict)
```

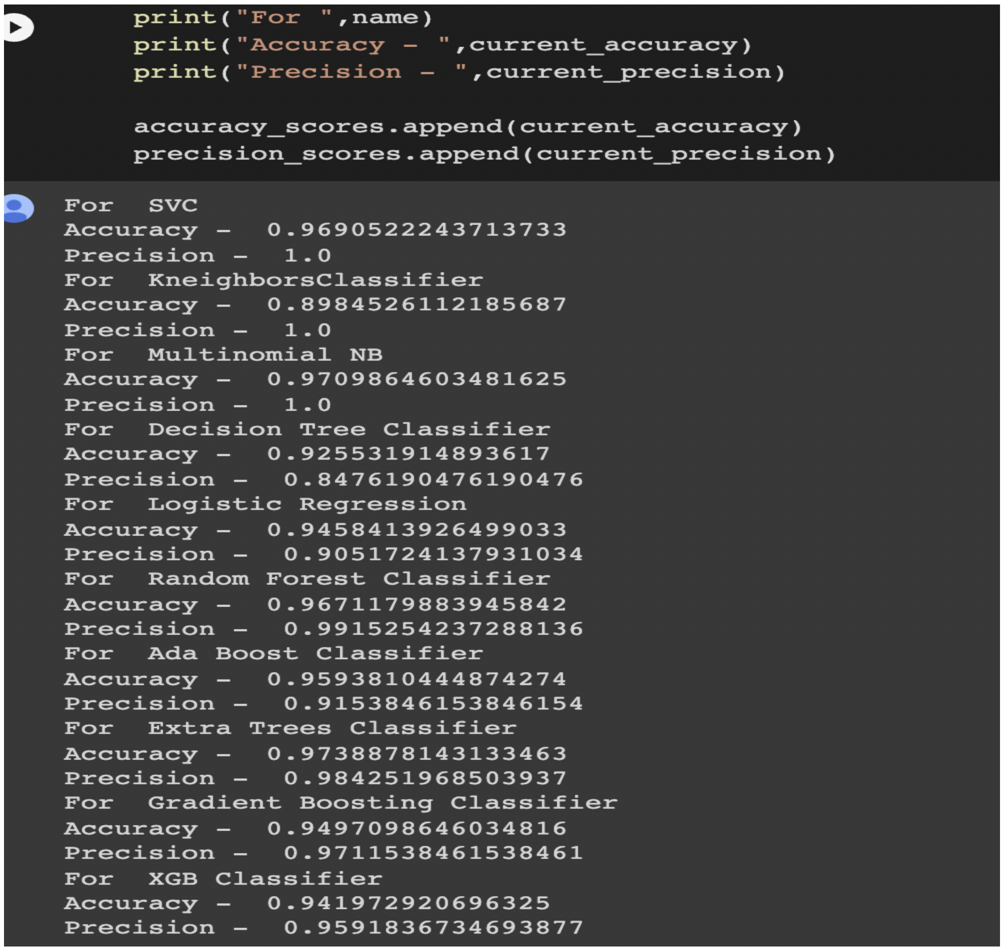
```
print(New_predict)
```

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

The accuracy of the model can be seen as follows:

1) Spam Detector

- The highest accuracy is achieved using the Multinomial Naive Bayes algorithm , which is 97.1% for spam detection and 97.21% for Malicious URL Prediction



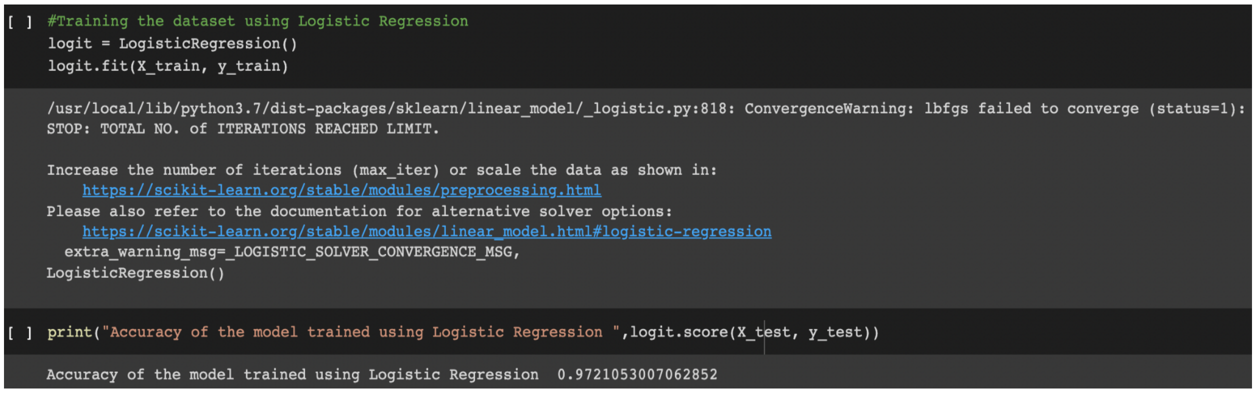
```
        print("For ",name)
        print("Accuracy - ",current_accuracy)
        print("Precision - ",current_precision)

        accuracy_scores.append(current_accuracy)
        precision_scores.append(current_precision)
```

```
For  SVC
Accuracy -  0.9690522243713733
Precision -  1.0
For  KneighborsClassifier
Accuracy -  0.8984526112185687
Precision -  1.0
For  Multinomial NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  Decision Tree Classifier
Accuracy -  0.925531914893617
Precision -  0.8476190476190476
For  Logistic Regression
Accuracy -  0.9458413926499033
Precision -  0.9051724137931034
For  Random Forest Classifier
Accuracy -  0.9671179883945842
Precision -  0.9915254237288136
For  Ada Boost Classifier
Accuracy -  0.9593810444874274
Precision -  0.9153846153846154
For  Extra Trees Classifier
Accuracy -  0.9738878143133463
Precision -  0.984251968503937
For  Gradient Boosting Classifier
Accuracy -  0.9497098646034816
Precision -  0.9711538461538461
For  XGB Classifier
Accuracy -  0.941972920696325
Precision -  0.9591836734693877
```

2) Link Detector

- The highest accuracy is achieved using the Logistic Regression algorithm , which is 97.21% for Malicious URL Prediction



```
[ ] #Training the dataset using Logistic Regression
    logit = LogisticRegression()
    logit.fit(X_train, y_train)
```

```
    /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
    LogisticRegression()
```

```
[ ] print("Accuracy of the model trained using Logistic Regression ",logit.score(X_test, y_test))
```

```
    Accuracy of the model trained using Logistic Regression  0.9721053007062852
```

References :

- https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- https://scikit-learn.org/stable/modules/svm.html
- https://docs.streamlit.io/
- https://docs.streamlit.io/library/get-started
- https://docs.aws.amazon.com/ec2/index.html
- https://towardsdatascience.com/how-to-deploy-a-streamlit-app-using-an-amazon-free-ec2-instance-416a41f69dc3
- https://pypi.org/project/wordcloud/
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://scikit-learn.org/stable/modules/naive_bayes.html
- https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
- https://scikit-learn.org/stable/modules/ensemble.html#adaboost
- https://www.kaggle.com/uciml/sms-spam-collection-dataset
- https://www.kaggle.com/sid321axn/malicious-urls-dataset