Phishing Website Detection using Deep Learning Algorithms

Project Report submitted in the partial fulfilment

Of

Bachelor of Technology

In

Electronics and Telecommunications Engineering

By

Apoorva Asgekar (C006)
Udita Katrolia (C022)
Devika Pillai (C031)

Under the supervision of

Mr. Vipul Gohil
(Assistant Professor, EXTC Department, MPSTME)

SVKM's NMIMS University
(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING
Vile Parle (W), Mumbai-56
2021-22

CERTIFICATE

This is to certify that the project entitled **"Phishing Website Detection using Deep Learning Algorithms"**, has been done by **Apoorva Asgekar, Udita Katrolia and Devika Pillai** under my guidance and supervision & has been submitted in partial fulfillment of degree of Bachelor of Technology in Electronics and Telecommunication of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

Vipul Gohil

(Internal Guide)

Examiner

Date

Place: Mumbai

Dr. Avinash More

(HOD EXTC)

ACKNOWLEDGEMENT

This project would not have been possible without the kind support and help of many individuals. We would like to express our deepest appreciation to all those who made it possible to complete this report. Firstly, we would like to thank the head of our department, Dr Avinash More who gave us the opportunity to develop this project and effectively apply our skills to it. A special gratitude we would like to give to our professor and mentor for our Major Project, Prof. Vipul Gohil, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project and helped us in writing this report.

We would also like to express our gratitude towards our parents & friends for their kind co-operation and encouragement which help us in completion of this project.

Apoorva Asgekar	Udita Katrolia,	Devika Pillai
B.Tech (EXTC)	B.Tech (EXTC)	B.Tech (EXTC)
Roll No. C006	Roll No. C022	Roll No. C031

ABSTRACT

Web service is one of the key communications software services for the Internet. With the rapid development of the Internet and the increasing popularity of electronic payment in web service, Internet fraud and web security have gradually been the main concern of the public. Phishing is a type of social engineering attack often used to steal user data. It occurs when an attacker disguised as a trusted entity, dupes a victim into opening an email, instant message, or text message. The recipient is then tricked into clicking a malicious link which can lead to the installation of malware. This leads to freezing of the system as part of a ransomware attack or the revealing of sensitive information. With the rapid development of the Internet and the increasing popularity of electronic payment in web service, Internet fraud and web security have gradually been the main concern of the public. Many users unwittingly click phishing domains every day and every hour. The attackers are targeting both the users and the companies. Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer's defence systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization's logos and other legitimate contents.

For our project we took a Deep Learning approach to detect phishing websites. The essence of these methods is to map all the features of the phishing website into the same space and then to use the deep learning algorithms to detect phishing. We propose to develop a model to detect the phishing attacks using Deep learning (DL) algorithms like ANN, CNN, CNN+LSTM, RNN+LSTM and Bidirectional LSTM using a custom dataset. Our aim is to obtain maximum accuracy using these algorithms and the custom dataset with minimum false positive rate during testing. The algorithm with best efficiency will be used to create a web interface to allow easy navigation for users.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Table Title	Page No.
KNN	K-Nearest Neighbours	2
SVM	Support Vector Machine	3
DL	Deep Learning	5
CNN	Convolution Neural Network	5
LSTM	Long Short-Term Memory	5
RNN	Recurrent Neural Network	5
PCA	Principal Component Analysis	5
ML	Machine Learning	6
DT	Decision Tree	6
RF	Random Forest	6
NB	Naïve Bayes	7
MFPD	Multidimensional Feature Phishing Detection	13
MRS	Microsoft Reputation Services	
GPU	Graphics processing unit	
ReLU	Rectified Linear activation Unit	
ANN	Artificial Neural Network	
IG	Information Gain	
HTTP	Hyper test transfer protocol	

HTTPS	Hypertext transfer protocol secured	
HTML	Hypertext Markup Language	
SLD	Second-Level Domain	
RMSProp	Root Mean Squared Propagation	

Chapter 1

Introduction

1.1 Background of the project topic

Web service is one of the key communications software services for the Internet. With the rapid development of the Internet and the increasing popularity of electronic payment in web service, Internet fraud and web security have gradually been the main concern of the public. Web Phishing is a way of such fraud, which uses social engineering techniques through short messages, emails, and WeChat to induce users to visit fake websites to get sensitive information like their private account, token for payment, credit card information, and so on. Phishing is a type of social engineering attack often used to steal user data. It occurs when an attacker disguised as a trusted entity, dupes a victim into opening an email, instant message, or text message. The recipient is then tricked into clicking a malicious link which can lead to the installation of malware. This leads to freezing of the system as part of a ransomware attack or the revealing of sensitive information.

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value. Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those categories, Logistic regression can be divided into following types:

Binary or Binomial: In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial: In such a kind of classification, dependent variables can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

Ordinal: In such a kind of classification, dependent variables can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

In statistics, the k-nearest neighbors algorithm (k-NN) is a non-parametric classification method which is used for classification and regression. In both cases, the input consists of the k closest training examples in a data set. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. k-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning. A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. Decision trees are commonly used in operations research and operations management. If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance. XG Boost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now.

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. SVM maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

AdaBoost, short for Adaptive Boosting, is a statistical classification meta-algorithm which can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. It is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner. AdaBoost is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.

1.2 Motivation and scope of the report

With the rapid development of the Internet and the increasing popularity of electronic payment in web service, Internet fraud and web security have gradually been the main concern of the public. Many users unwittingly click phishing domains every day and every hour. The attackers are targeting both the users and the companies. Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer’s defense systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization’s logos and other legitimate contents. Based on related research successful rates have been achieved on detection accuracy using different learning algorithms but still website phishing detection is very much open for research because the rate at which phishing websites are deployed and the method used is faster than the solutions proposed by researchers.

As a new type of cyber security threat, phishing websites appear frequently in recent years, which have led to great harm in online financial services and data security. The main reason is the lack of awareness of users. But security defenders must take precautions to prevent users from confronting these harmful sites. Preventing these huge costs can start with making people conscious in addition to building strong security mechanisms which are able to detect and prevent phishing domains from reaching the user. It has been projected that the vulnerability of most web servers have led to the evolution of most phishing websites such that the weakness in the web server is exploited by phishers to host counterfeiting websites without the knowledge of the owner. It is also possible that the phisher hosts a new web server independent of any legitimate web server for phishing activities.

1.3 Problem statement

With the popularity of machine learning, phishing detection has focused on the use of machine learning algorithms. For our project we took a Deep Learning approach to detect phishing websites. The essence of these methods is to map all the features of the phishing website into the same space and then to use the deep learning algorithms to detect phishing. We propose:

1. To develop a model to detect the phishing attacks using Deep learning (DL) algorithms like ANN, CNN, CNN+LSTM, RNN+LSTM and Bidirectional LSTM using a custom dataset.
2. The dataset consists of 21,581 websites collected from Phishtank for phishing sites and Alexa for legitimate sites.
3. To analyse the features of the dataset, the proposed model will use feature selection algorithms like Information Gain and principal component analysis (PCA).

4. Our aim is to obtain maximum accuracy using the above DL algorithms and custom dataset with minimum false positive rate during testing.
5. The algorithm with best efficiency will be used to create a web interface using Flask to allow easy navigation for users.

1.4 Salient Contribution

- Helps to detect attacks faster and remediate threats as quickly as possible.
- Increase user alertness to phishing risks
- Instill a cyber security culture and create cyber security heroes
- Change behavior to eliminate the automatic trust response
- Protect valuable corporate and personal data
- Meet industry compliance obligations
- Assess the impacts of cyber security awareness training

1.3 Organization of Report

Chapter 1: It includes the background, motivation, problem statement and salient contribution of the project.
Chapter 2: It includes the literature survey of 7 research papers and also the comparison of all the papers.
Chapter 3: It includes the description of the software used in this project. It also discusses about the methodology like how the features are extracted and implementation of the project.
Chapter 4: It is about the results and analysis of the project. We have put the screenshots of the confusion matrices, ROC curves and the web application.
Chapter 5: It includes the advantages, limitations and applications of the project.
Chapter 6: It includes the conclusion and future scope of the project.

Chapter 2

Literature survey

2.1 Introduction

[1] Phishing Attacks Detection using Machine Learning Approach

Evolving digital transformation has exacerbated cybersecurity threats globally. Digitization expands the doors wider to cybercriminals. Initially cyber threats approach in the form of phishing to steal the confidential user credentials. Usually, Hackers will influence the users through phishing in order to gain access to the organization's digital assets and networks. With security breaches, cybercriminals execute ransomware attacks, get unauthorized access, and shut down systems and even demand a ransom for releasing the access. Anti-phishing software and techniques are circumvented by the phishers for dodging tactics. Though threat intelligence and behavioural analytics systems support organizations to spot the unusual traffic patterns, still the best practice to prevent phishing attacks is defended in depth.

The proposed research work has developed a model to detect the phishing attacks using machine learning (ML) algorithms like random forest (RF) and decision tree (DT). DT was used to classify the website and RF was used for classification. To analyze the attributes of the dataset, the proposed model has used feature selection algorithms like principal component analysis (PCA). Finally, the confusion matrix was drawn to evaluate the performance of these two algorithms. RF had less variance, and it could handle the over-fitting problem. The random forest tree achieved an accuracy of 97% and the decision tree achieved an accuracy of 91%.

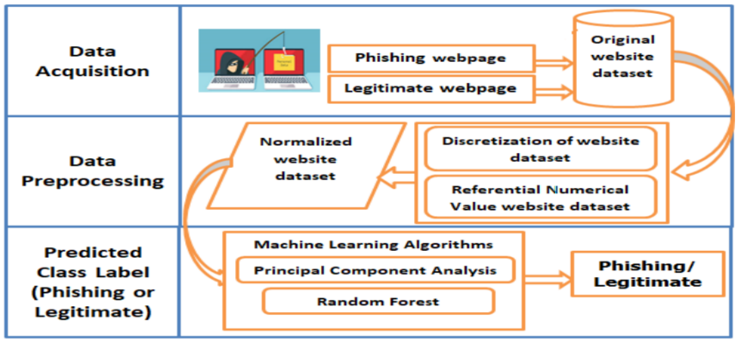

Data Acquisition
Phishing webpage
Legitimate webpage
Original website dataset
Data Preprocessing
Normalized website dataset
Discretization of website dataset
Referential Numerical Value website dataset
Predicted Class Label (Phishing or Legitimate)
Machine Learning Algorithms
Principal Component Analysis
Random Forest
Phishing/ Legitimate

Fig.1 - Framework for phishing detection

[2] Machine Learning Techniques for Detection of Website Phishing: A Review for Promises and Challenges

Website phishing is a cyber-attack that targets online users to steal their sensitive information including login credentials and banking details. Attackers fool the users by presenting the masked webpage as legitimate or trustworthy to retrieve their essential data. Several solutions to phishing websites attacks have been proposed such as heuristics, blacklist or whitelist, and Machine Learning (ML) based techniques.

This survey paper has revealed information about the advantages of the ML techniques for the detection of phishing websites. Results demonstrated that ML methods are effective in eradicating phishing. However, not all phishing related problems have been yet resolved by ML techniques. The research and the development of new approaches are an ongoing process as attackers think about new phishing ideas and develop new phishing methods every day.

The three types of phishing attacks discussed in this paper are as follows:

- The first type is spear phishing which is processed to collect the users information and details about their specific targets.
- Clone phishing is the second type of phishing used to pivot from the earlier infected machines. The received attachment detects this phishing type with a legitimate email, and a clone or an identical email can be developed.
- The third type of phishing is known as whaling which is coined of the earlier mentioned spear phishing. The target of these phishing attacks is the high profile and senior executives.

In this paper, a review of ML-based approaches for detection of phishing websites is conducted. This work presents a comprehensive review of conventional ML techniques which are significant for detection of malicious attacks on websites. The majority of the examined approaches are focused on traditional ML techniques. Random Forests (RFs), Support Vector Machines (SVMs), Naive Bayes (NB), and Ada Boosting are the powerful ML models that are examined in the paper.

Inefficiency of ML techniques on a large amount of data, images, and websites with captcha information have been identified. Overfitting, low accuracy, and hyper parameters tuning of ML techniques are widely studied in the paper. Small size of datasets to train the ML techniques is another challenge as identified in this research.

This survey paper also identifies deep learning-based techniques to demonstrate better performance for detecting phishing websites compared to the conventional ML techniques. Other protection strategies presented in this paper are heuristic and automated techniques.

[3] Deep Learning with Convolutional Neural Network and Long Short-Term Memory for Phishing Detection

This study focuses on design and development of a deep learning-based phishing detection solution that leverages the Universal Resource Locator and website content such as images and frame elements. A Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM) algorithm were used to build a classification model. The experimental results showed that the proposed model achieved an accuracy rate of 93.28%.

In this study, we use both the LSTM layer and the CNN to extract features from different websites, where the output is a binary number that reflects whether or not the input sequences are real or fake (phishing).

The concept of CNN is superior to that of NN in terms of time-delay. In the CNN concept, the weights are shared in a temporal dimension, which leads to a decrease in computation time. The general matrix multiplication in the standard NN is therefore replaced in the CNN. Hence the CNN approach reduces the weights, thereby decreasing the complexity of the network. Consequently, the feature extraction procedure in a standard learning algorithm can be enhanced by directly importing images into the network as raw inputs. This type of model for the training of the architecture layers led to the success of the first DL algorithms.

Based on this result, we can deduce that distinguishing websites by their URLs, similarity context, and the images on the site by their pattern is an effective way of detecting phishing websites. This method was able to respond with great agility and could verify a URL.

In addition, analysis revealed the advantages and disadvantages of both the CNN and LSTM methods. On the one hand, LSTM has an overall higher prediction performance, but there is a need for expert knowledge when creating the features. On the other hand, the CNN performs better, but achieves a performance that is, on average, a bit lower than that of LSTM. Hence, combining the two methods leads to a better result with less training time for LSTM architecture than the CNN model.

This study carried out the extensive experimental analysis of the proposed hybrid approach in order to evaluate its effectiveness in the detection of phishing web pages and phishing attacks on large datasets. Then the sensitiveness of the proposed approach to various factors such as the type of features, number misclassification and split issues was studied. The result of our experiments showed that the proposed approach was highly effective in the detection of website phishing attacks as well as in the identification of fake websites.

[4] Phishing Detection from URLs Using Deep Learning Approach

Phishing attacks present negative impacts on web owners and end-users. The reputation of website owners becomes questionable when attackers launch an attack, and as a result of the attack, website users lose their sensitive information. Cybercrime has become the center of attraction for cyber attackers in cyberspace. Phishing is one such technique where the unidentified structure of the Internet has been used by attackers/criminals that intend to deceive users with the use of the illusory website and emails for obtaining their credentials (like account numbers, passwords, and PINs). Consequently, the identification of a phishing or legitimate web page is a challenging issue due to its semantic structure.

In this paper, a phishing detection system is implemented using deep learning techniques to prevent such attacks. The system works on URLs by applying a convolutional neural network (CNN) to detect the phishing webpage. The program flow of the proposed framework is shown in the below figure. The data pre-processing phase includes URL concatenation followed by tokenization, padding, and sequencing which is fed to the embedded layer.

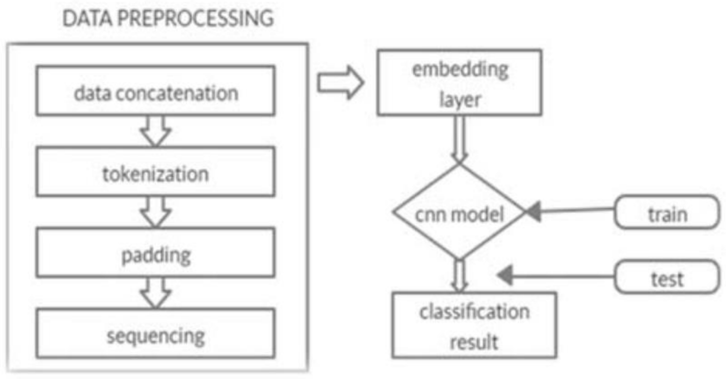


Fig. 2 – Program flow of phishing detection system

Firstly, we tokenize the URLs that pass it through the glove embedding layer and then it is passed through CNN Deep neural network, followed by the dense neural network to gain the F-1 score precision, recall, accuracy, support of the legitimate and phishing URLs.

The proposed system can classify URLs as phishing and legitimate and achieve an accuracy of 98.00%. This system doesn't require any feature engineering as the CNN extracts features from the URLs automatically through its hidden layers. This is another advantage of the proposed system as the feature engineering is a very time- consuming task.

[5] Research on Website Phishing Detection Based on LSTM RNN

Phishers sometimes exploit users trust of a known website's appearance by using a similar page that looks like the legitimate site. In recent times, researchers have tried to identify and classify the issues that can contribute to the detection of phishing websites.

In the phishing site testing process, many factors affect the test results, with a certain degree of non-linearity. This paper implements a LSTM deep learning method and optimizes the training method of the model in combination with the characteristics of Recurrent Neural Networks (RNN), which solves the problem that it is difficult for other machine learning methods to extract valid features from the data.

Bi-directional LSTM using finite sequence according to the context of elements in the past and the future to predict or tag sequence of each element. This is the output of two LSTM in series, one processing sequence from left to right, and the other from right to left. Composite output is the prediction of a given target signal. This technique has proved particularly useful. The essential feature of RNN neural network is that there is an internal feedback connection and a feedforward connection between the processing units.

Long-term short-term memory (LSTM) differs from the RNN mainly in that it incorporates an LSTM unit that determines the usefulness of information in the algorithm. Below figure shows a single LSTM cell.

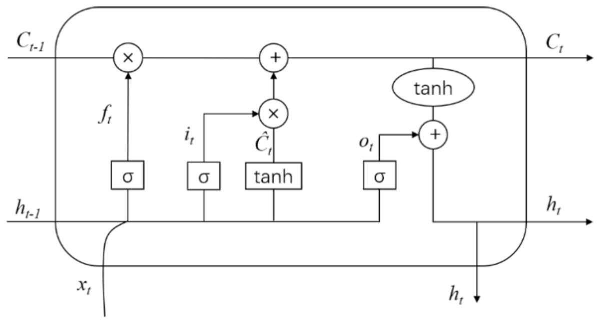


Fig. 3 – LSTM cell

LSTM has the advantage of capturing data timing and long-term dependencies. LSTM has strong learning ability, can automatically learn data characterization without manual extraction of complex features, and has strong potential in the face of complex high-dimensional massive data. Experimental results show that this model approaches the accuracy of 99.1%, which is higher than that of other neural network algorithms.

The training time of a deep learning model is generally possible from hours to days, and the optimization convergence time is strict on the timelines of power dispatching and other issues.

[6] Phishing Website Detection based on Multidimensional Features driven by Deep Learning

It is well known that a good phishing website detection approach should have good real-time performance while ensuring good accuracy and a low false positive rate. Feature engineering is important in phishing website-detection solutions, but the accuracy of detection critically depends on prior knowledge of features. Moreover, although features extracted from different dimensions are more comprehensive, a drawback is that extracting these features requires a large amount of time.

To address these limitations, this paper proposes a multidimensional feature phishing detection approach based on a fast detection method by using deep learning (MFPD). In the first step, character sequence features of the given URL are extracted and used for quick classification by deep learning, and this step does not require third-party assistance or any prior knowledge about phishing. Specially, the CNN (convolutional neural network) is used to extract local correlation features through a convolutional layer. In a URL, each character may be related to nearby characters. Generally speaking, a phishing website is likely to mimic the URL of a legitimate website by changing or adding some characters. This can cause the sequential dependency of the phishing URL to be different from the legitimate URL. The LSTM network can effectively learn the sequential dependency from character sequences. Therefore, the LSTM (long short-term memory) network is employed to capture context semantic and dependency features of URL character sequences, and at finally softmax is used to classify the extracted features. We call the first step CNN-LSTM. From a comprehensive perspective, in the second step, we combine URL statistical features, webpage code features, webpage text features and the classification result of deep learning into multidimensional features, which are then classified by XGBoost. Although the multidimensional feature detection method has higher accuracy, it requires extracting features from different aspects, resulting in longer detection time. In contrast, the method for the URL character sequences only needs to process the URL, and the detection time is short. To balance the contradiction between detection time and accuracy, we improve the output judgment condition of the softmax classifier in the deep learning process by setting a threshold to reduce the detection time. If the result of deep learning is not less than the specified threshold, the detection result is directly output; otherwise, go to the second step of detection.

The process of phishing website detection using MFPD is explained, and an extensive experiment on the dataset we built is conducted. The results show that our proposed approach exhibits good performance in terms of accuracy, false positive rate, and speed.

Testing on a dataset containing millions of phishing URLs and legitimate URLs, the accuracy reaches 98.99%, and the false positive rate is only 0.59%. By reasonably adjusting the threshold, the experimental results show that the detection efficiency can be improved.

[7] Phishing URL detection using URL Ranking

The openness of the Web exposes opportunities for criminals to upload malicious content. In fact, despite extensive research, email-based spam filtering techniques are unable to protect other web services. Therefore, a counter measure must be taken that generalizes across web services to protect the user from phishing host URLs.

The vector that is used to trap users is a Uniform Resource Locator (URL). The user must perform sanity checks before clicking a URL, such as paying close attention to the spelling of the website's address and evaluating the associated risk that might be encountered by visiting the URL. Security researchers have done extensive research to detect accounts on social networks used for spreading messages that lure the recipient to a potential phishing website. Their technique of using honey-profiles to identify single spam bots as well as large-scale campaigns does not provide complete protection for users. Social networks consist of real-time interaction, but the technique for detecting fraudulent accounts can incur delays due to the need for building a profile of inappropriate activity. The current work proposes a system capable of clustering, classifying, categorizing, and ranking URLs in real-time while adapting to new and evolving trends in URL characteristics.

An attempt to improve existing techniques for detecting phishing URLs is made by adding novel features (i.e., predictor variables), and making the learning process more efficient.

Online algorithms adapt to changes quickly and are a better choice since the values for predictor variables of URLs can change over time. One major contribution of the current paper is the implementation of a hybrid approach combining both clustering and classification. Clustering is performed prior to classification, and cluster IDs from the clustering step are added as a predictor variable to expand each feature vector present in the dataset. The process of performing clustering and using cluster IDs (or cluster labels) helps in obtaining a hyperplane with a larger margin which in turn results in higher classifier efficiency due to better generalization. Another major contribution of the paper is URL categorization by Microsoft Reputation Services (MRS). The categories obtained from MRS are used to perform ranking of URLs by using a novel URL Ranking approach, which consists of using the classification result and an internal scale (Red/Yellow/Green) of severity derived for each URL.

This paper describes an approach that classifies URLs automatically based on their lexical and host- based features. Clustering is performed on the entire dataset and a cluster ID (or label) is derived for each URL, which in turn is used as a predictive feature by the classification system. Online URL reputation services are used in order to categorize URLs and the categories returned are used as a supplemental source of information that would enable the system to rank URLs. The classifier achieves 93-98% accuracy by detecting a large number of phishing hosts, while maintaining a modest false positive rate. URLs are ranked by using URL clustering, classification, and categorization demonstrating that cluster labels increase the classifier accuracy from 97.08% to 98.46%.

2.2 Exhaustive Literature Survey

Table 1: Exhaustive Literature Survey

Paper	Algorithms used	Efficiency
Phishing Attacks Detection using Machine Learning Approach	Random Forests (RFs) and Decision Tree	95%
Machine Learning Techniques for Detection of Website Phishing: A Review for Promises and Challenges	Random Forests (RFs), Support Vector Machines (SVMs), Naïve Bayes (NB), and Ada Boosting	-
Deep Learning with Convolutional Neural Network and Long Short-Term Memory for Phishing Detection	CNN and LSTM	93.28%
Phishing Detection from URLs Using Deep Learning Approach	CNN	98%
Research on Website Phishing Detection Based on LSTM RNN	LSTM RNN	99.10%
Phishing Website Detection based on Multidimensional Features driven by Deep Learning	Multidimensional Feature Phishing Detection	98.99%
Phishing URL detection using URL Ranking	Lexical and Host- based features	98.46%

Chapter 3
Methodology and Implementation

3.1 Software description

The project has been implemented using the Python programming language. The software used in the implementation is Google Collaboratory. It allows us to write and execute Python in your browser, with zero configuration required, free access to GPUs as well as easy sharing. With Colab we can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. So with Colab we can harness the full power of popular Python libraries to analyze and visualize data to be used in our models implementation.

The various libraries for Machine Learning and Deep Learning that we have used are:

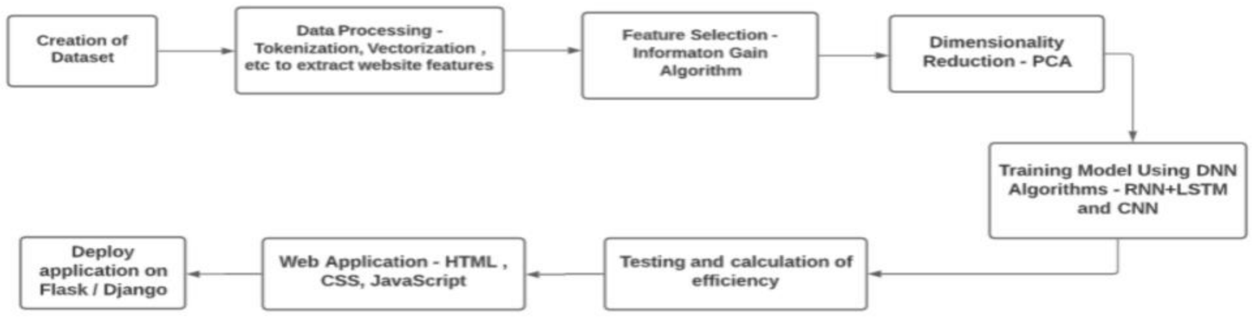
1. **pandas**: It is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. pandas is well suited for many different kinds of data like tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet.
2. **sklearn**: Scikit-learn is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.
3. **Tensorflow**: It is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.
4. **Keras**: It is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development.
5. **matplotlib. pyplot**: It is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a

figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Flask is used for developing web applications using python, implemented on Werkzeug and Jinja2. Advantages of using Flask framework are:

- There is a built-in development server and a fast debugger provided.
- Lightweight
- Secure cookies are supported.
- Templating using Jinja2.
- Request dispatching using REST.
- Support for unit testing is built-in.

3.2 Block diagram


Creation of Dataset
Data Processing - Tokenization, Vectorization , etc to extract website features
Feature Selection - Informaton Gain Algorithm
Dimensionality Reduction - PCA
Training Model Using DNN Algorithms - RNN+LSTM and CNN
Testing and calculation of efficiency
Web Application - HTML , CSS, JavaScript
Deploy application on Flask / Django

3.3 Algorithms Used

1. **Convolutional Neural Network**

   CNN is a class of deep, feed-forward artificial neural networks where connections between nodes do not form a cycle & use a variation of multilayer perceptron's designed to require minimal pre-processing. CNN is just a kind of neural network; its convolutional layer differs from other neural networks. To perform image classification, CNN goes through every corner, vector and dimension of the pixel matrix. Performing with this all features of a matrix makes CNN more sustainable to data of matrix form. We can consider text data as sequential data like data in time series, a one-dimensional matrix. We need to work with a one-dimensional convolution layer. The idea of the model is almost the same, but the data type and dimension of convolution layers changed. To work with Text CNN, we require a word embedding layer and a one-dimensional convolutional network.

   We implemented a CNN-1D model using Keras with a TensorFlow- GPU backend technique to tackle the problem of detecting false URLs. The main concept of CNN is quite similar to linear neural network, which takes raw data as an input (1D vector). The first convolutional layer has 64 filters with kernel size of 3 and The second convolutional layer has 64 filters with kernel size of 128 ReLU activation followed by a concatenate layer. The subdomain, domain and domain suffix are passed into the embedding layer followed by a dropout layer. The final layer is a dense layer with one neuron, allowing the complete model to output binary classifications i.e., a Malicious or legitimate URL.

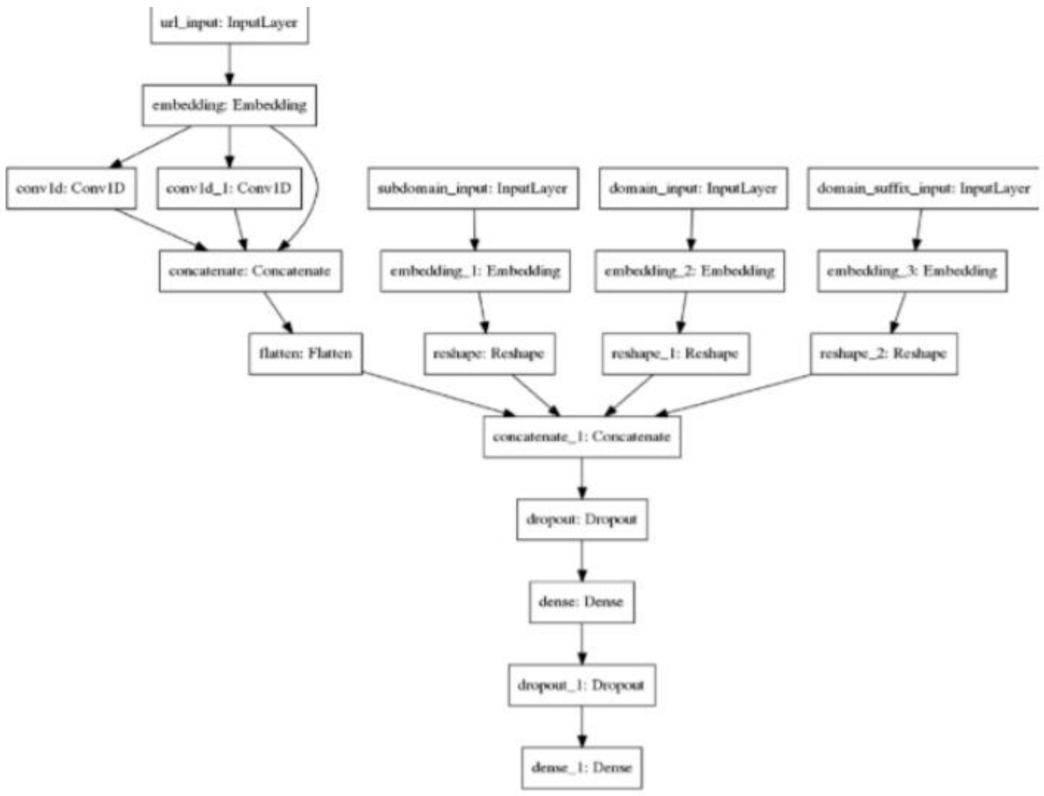


Fig.4 - Working of CNN algorithm for text classification

2. **Artificial Neural Network**

Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times [8].

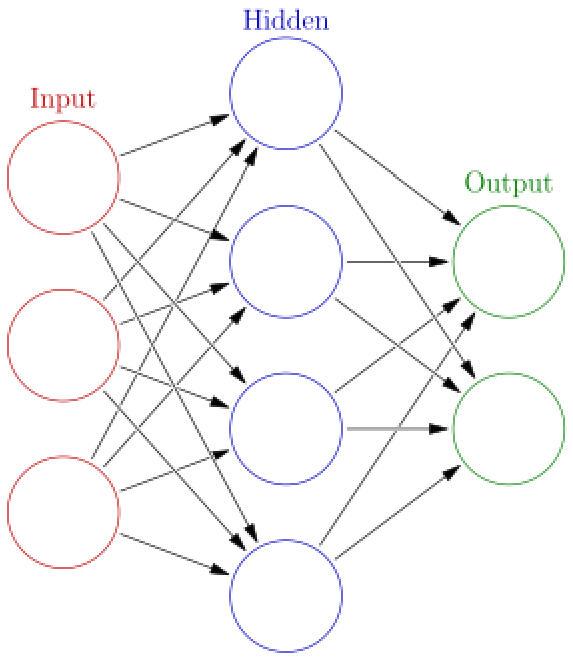


Fig. 5 - Artificial Neural Network Architecture

3. **CNN + Long Short-Term Memory (LSTM)**

CNNs are used in modeling problems related to spatial inputs like images. CNNs have been proved to successful in image related tasks like computer vision, image classification, object detection etc. LSTMs are used in modelling tasks related to sequences and do predictions based on it. LSTMs are widely used in NLP related tasks like machine translation, sentence classification, generation. Standard LSTM (Vanilla LSTM) can’t be used directly on sequences where input is spatial. So to perform tasks which need sequences of images to predict something we need more sophisticated model. That’s where CNN-LSTM model comes in. The CNN Long Short-Term Memory Network (CNN-LSTM) is an LSTM architecture specifically designed for sequence prediction problems with spatial inputs, like images or videos.

A CNN LSTM can be defined by adding CNN layers on the front end followed by LSTM layers with a Dense layer on the output. It is helpful to think of this architecture as defining two sub-models: the CNN Model for feature extraction and the LSTM Model for interpreting the features across time steps. The below picture describes how a general CNN-LSTM model work. CNN-LSTMs are generally used when their inputs have spatial structure in their input such as the 2D structure or pixels in an image or the 1D structure of words in a sentence, paragraph, or document and also have a temporal structure in their input such as the order of images in a video or words in text, or require the generation of output with temporal structure such as words in a textual description [9].

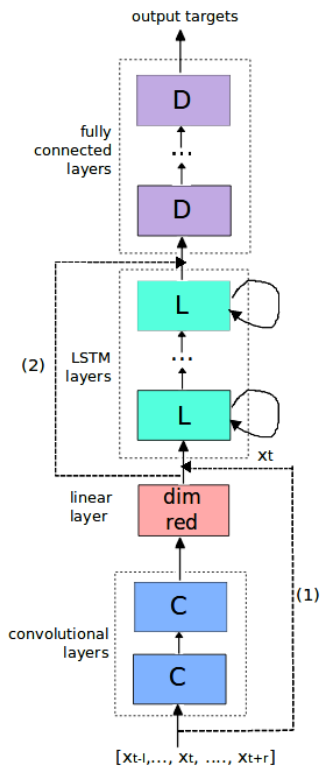


Fig. 6 - CNN LSTM Architecture

4. Recurrent Neural Networks (RNN) + LSTM

RNNs are a powerful and robust type of neural network, and belong to the most promising algorithms in use because it is the only one with an internal memory. It is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for machine learning problems that involve sequential data. Sequential data is basically just ordered data in which related things follow each other. RNN's and feed-forward neural networks get their names from the way they channel information. In a feed-forward neural network, the information only moves in one direction from the input layer, through the hidden layers, to the output layer. The information moves straight through the network and never touches a node twice. Feed-forward neural networks have no memory of the input they receive and are bad at predicting what's coming next. Because a feed-forward network only considers the current input, it has no notion of order in time. It simply can't remember anything about what happened in the past except its training. In a RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.

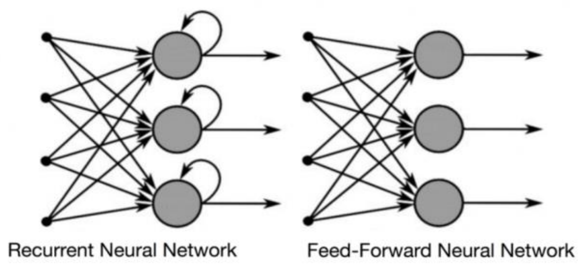


Fig. 7 - Difference in RNN and Feed-Forward Neural Network

A usual RNN has a short-term memory. In combination with a LSTM, they also have a long-term memory. Long short-term memory networks (LSTMs) are an extension for recurrent neural networks, which basically extends the memory. Therefore, it is well suited to learn from important experiences that have very long-time lags in between [10].

Long Short-Term Memory (LSTM) networks are an extension of RNN that extend the memory. LSTM are used as the building blocks for the layers of a RNN. LSTMs assign data "weights" which helps RNNs to either let new information in, forget information or give it importance enough to impact the output [10].
In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information

because it isn't important (forget gate), or let it impact the output at the current timestep (output gate). Below is an illustration of a RNN with its three gates:

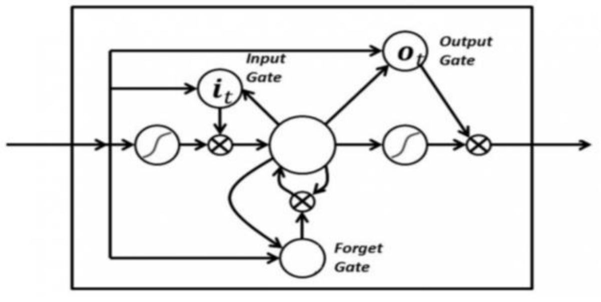


Fig. 8 - LSTM Gates Network

5. **Bidirectional LSTM**

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem [11].

It involves duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second. In bidirectional LSTM, instead of training a single model, we introduce two. The first model learns the sequence of the input provided, and the second model learns the reverse of that sequence [10] . A Bidirectional LSTM, or Bi-LSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. Bi-LSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm (e.g., knowing what words immediately follow *and* precede a word in a sentence).

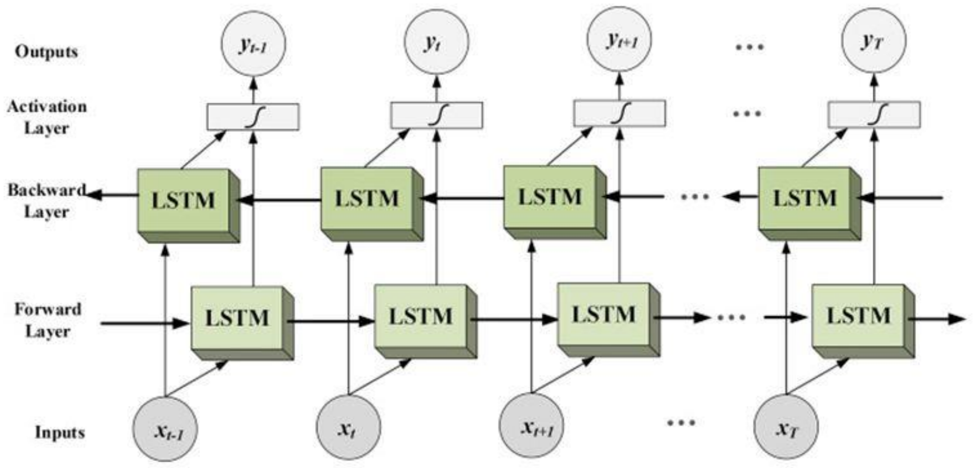


Fig. 9- Architecture of Bidirectional LSTM

3.4 Features

One of the challenges faced by our research was the unavailability of reliable training datasets. In fact, this challenge faces any researcher in the field. However, although plenty of articles about predicting phishing websites using data mining techniques have been disseminated these days, no reliable training dataset has been published publicly, maybe because there is no agreement in literature on the definitive features that characterize phishing websites, hence it is difficult to shape a dataset that covers all possible features.
In this article, we shed light on the important features that have proved to be sound and effective in predicting phishing websites. In addition, we proposed some new features, experimentally assign new rules to some well-known features and update some other features [12].

Address Bar based Features

a) Using the IP Address

If an IP address is used as an alternative of the domain name in the URL, such as "http://125.98.3.123/fake.html", users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link "http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html".

Rule: IF$\begin{cases}\text{If The Domain Part has an IP Address } \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate}\end{cases}$

b) Long URL to Hide the Suspicious Part

Phishers can use long URL to hide the doubtful part in the address bar. For example: http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd =_home&amp;dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8 dc1e7c2e8dd4105e8@phishing.website.html

To ensure accuracy of our study, we calculated the length of URLs in the dataset and produced an average URL length. The results showed that if the length of the URL is greater than or equal 54 characters then the URL classified as phishing. By reviewing our dataset, we were able to find 1220 URLs lengths equals to 54 or more which constitute 48.8% of the total dataset size.

Rule: IF$\begin{cases} URL\ length < 54 \ \rightarrow \ feature = \text{Legitimate} \\ else\ if\ URL\ length \geq 54\ and\ \leq 75 \ \rightarrow \ feature = Suspicious \\ otherwise \rightarrow \ feature = \text{Phishing}\end{cases}$

We have been able to update this feature rule by using a method based on frequency and thus improving upon its accuracy.

c) Using URL Shortening Services “TinyURL”

URL shortening is a method on the “World Wide Web” in which a URL may be made considerably smaller in length and still lead to the required webpage. This is accomplished by means of an “HTTP Redirect” on a domain name that is short, which links to the webpage that has a long URL. For example, the URL “http://portal.hud.ac.uk/” can be shortened to “bit.ly/19DXSk4”.

Rule: IF$\begin{cases}\text{TinyURL } \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate}\end{cases}$

d) URL's having "@" Symbol

Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol.

Rule: IF $\begin{cases} \text{Url Having @ Symbol} \rightarrow \text{ Phishing} \\ \quad\quad \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

e) Redirecting using "//"

The existence of "//" within the URL path means that the user will be redirected to another website. An example of such URL's is: "http://www.legitimate.com//http://www.phishing.com". We examine the location where the "//" appears. We find that if the URL starts with "HTTP", that means the "//" should appear in the sixth position. However, if the URL employs "HTTPS" then the "//" should appear in seventh position.

Rule: IF

$\begin{cases} \text{ThePosition of the Last Occurrence of "//" in the URL } > 7 \rightarrow \text{ Phishing} \\ \quad\quad\quad\quad\quad\quad \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

f) Adding Prefix or Suffix Separated by (-) to the Domain

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example, http://www.Confirme-paypal.com/.

Rule: IF $\begin{cases} \text{Domain Name Part Includes } (-) \text{ Symbol } \rightarrow \text{ Phishing} \\ \quad\quad\quad\quad \text{Otherwise } \rightarrow \text{ Legitimate} \end{cases}$

g) Sub Domain and Multi Sub Domains

Let us assume we have the following link: http://www.hud.ac.uk/students/. A domain name might include the country-code top-level domains (ccTLD), which in our example is "uk". The "ac" part is shorthand for "academic", the combined "ac.uk" is called a second-level domain (SLD) and "hud" is the actual name of the domain. To produce a rule for extracting this feature, we firstly have to omit the (www.) from the URL which is in fact a sub domain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as “Suspicious” since it has one sub domain. However, if the dots are greater than two, it is classified as “Phishing” since it will have multiple sub domains. Otherwise, if the URL has no sub domains, we will assign “Legitimate” to the feature.

Rule: IF $\begin{cases} \text{Dots In Domain Part} = 1 \rightarrow \text{Legitimate} \\ \text{Dots In Domain Part} = 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

h) HTTPS (Hyper Text Transfer Protocol with Secure Sockets Layer)

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough. The authors in (Mohammad, Thabtah and McCluskey 2012) (Mohammad, Thabtah and McCluskey 2013) suggest checking the certificate assigned with HTTPS including the extent of the trust certificate issuer, and the certificate age. Certificate Authorities that are consistently listed among the top trustworthy names include: “GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo, Doster and VeriSign”. Furthermore, by testing out our datasets, we find that the minimum age of a reputable certificate is two years.

Rule:

IF

$$\begin{cases} \text{Use https and Issuer Is Trusted and Age of Certificate} \geq 1 \text{ Years} \rightarrow \text{Legitimate} \\ \text{Using https and Issuer Is Not Trusted} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

i) Domain Registration Length

Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

Rule: IF$\begin{cases} \text{Domains Expires on} \leq 1 \text{ years} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

j) Favicon

A favicon is a graphic image (icon) associated with a specific webpage. Many existing user agents such as graphical browsers and newsreaders show favicon as a visual reminder of the website identity in the address bar. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.

Rule: IF $\begin{cases} \text{Favicon Loaded From External Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

k) Using Non-Standard Port

This feature is useful in validating if a particular service (e.g., HTTP) is up or down on a specific server. In the aim of controlling intrusions, it is much better to merely open ports that you need. Several firewalls, Proxy and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected. If all ports are open, phishers can run almost any service they want and as a result, user information is threatened. The most important ports and their preferred status are shown in Table 2.

Rule: IF $\begin{cases} \text{Port \# is of the Preffered Status} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

Table 2 Common ports to be checked

PORT	Service	Meaning	Preferred Status
21	FTP	Transfer files from one host to another	Close
22	SSH	Secure File Transfer Protocol	Close
23	Telnet	provide a bidirectional interactive text-oriented communication	Close
80	HTTP	Hyper test transfer protocol	Open
443	HTTPS	Hypertext transfer protocol secured	Open
445	SMB	Providing shared access to files, printers, serial ports	Close

1433	MSSQL	Store and retrieve data as requested by other software applications	Close
1521	ORACLE	Access oracle database from web.	Close
3306	MySQL	Access MySQL database from web.	Close
3389	Remote Desktop	allow remote access and remote collaboration	Close

l) The Existence of "HTTPS" Token in the Domain Part of the URL

The phishers may add the "HTTPS" token to the domain part of a URL in order to trick users. For example, http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/.

Rule: IF$\begin{cases} \text{Using HTTP Token in Domain Part of The URL} \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

Abnormal Based Features

a) Request URL

Request URL examines whether the external objects contained within a webpage such as images, videos and sounds are loaded from another domain. In legitimate webpages, the webpage address and most of objects embedded within the webpage are sharing the same domain.

Rule: IF $\begin{cases} \text{\% of Request URL } < 22\% \rightarrow \text{ Legitimate} \\ \text{\%of Request URL} \geq 22\% \text{ and } 61\% \rightarrow \text{ Suspicious} \\ \text{Otherwise} \rightarrow \text{ feature} = \text{Phishing} \end{cases}$

b) URL of Anchor

An anchor is an element defined by the <a> tag. This feature is treated exactly as "Request URL". However, for this feature we examine:

If the <a> tags and the website have different domain names. This is similar to request URL feature.

If the anchor does not link to any webpage, e.g.:

<a href="#">

<a href="#content">

<a href="#skip">

<a href="JavaScript ::void(0)">

Rule: IF$\begin{cases} \text{\% of URL Of Anchor } < 31\% \rightarrow \mathit{Legitimate} \\ \text{\% of URL Of Anchor } \geq 31\% \text{ And } \leq 67\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

c) Links in <Meta>, <Script> and <Link> tags

Given that our investigation covers all angles likely to be used in the webpage source code, we find that it is common for legitimate websites to use <Meta> tags to offer metadata about the HTML document; <Script> tags to create a client-side script; and <Link> tags to retrieve other web resources. It is expected that these tags are linked to the same domain of the webpage.

Rule:

IF

$\begin{cases} \text{\% of Links in " } < \text{Meta} > \text{", " } < \text{Script} > \text{" and " } < \text{Link>" } < 17\% \rightarrow \text{Legitimate} \\ \text{\% of Links in } < \text{Meta} > \text{", " } < \text{Script} > \text{" and " } < \text{Link>" } \geq 17\% \text{ And } \leq 81\% \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

d) Server Form Handler (SFH)

SFHs that contain an empty string or "about:blank" are considered doubtful because an action should be taken upon the submitted information. In addition, if the domain name in SFHs is different from the domain name of the webpage, this reveals that the webpage is suspicious because the submitted information is rarely handled by external domains.

Rule: IF$\begin{cases} \text{SFH is "about: blank" Or Is Empty } \rightarrow \text{Phishing} \\ \text{SFH Refers To A Different Domain} \rightarrow \text{Suspicious} \\ \text{Otherwise } \rightarrow \text{Legitimate} \end{cases}$

e) Submitting Information to Email

Web form allows a user to submit his personal information that is directed to a server for processing. A phisher might redirect the user's information to his personal email. To that end, a server-side script language might be used such as "mail ()" function in PHP. One more client-side function that might be used for this purpose is the "mailto:" function.

Rule:

$$\text{IF}\begin{cases}\text{Using "mail()" or "mailto:" Function to Submit User Information} \rightarrow \text{Phishing} \\ \qquad\qquad\qquad\qquad \text{Otherwise} \rightarrow \text{Legitimate}\end{cases}$$

f) Abnormal URL

This feature can be extracted from WHOIS database. For a legitimate website, identity is typically part of its URL.

$$\text{Rule: IF}\begin{cases}\text{The Host Name Is Not Included In URL} \rightarrow \text{Phishing} \\ \qquad\qquad \text{Otherwise} \rightarrow \text{Legitimate}\end{cases}$$

HTML and JavaScript based Features

a) Website Forwarding

The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected. In our dataset, we find that legitimate websites have been redirected one time max. On the other hand, phishing websites containing this feature have been redirected at least 4 times.

$$\text{Rule: IF}\begin{cases} & \text{ofRedirect Page} \leq 1 \rightarrow \text{Legitimate} \\ & \text{of Redirect Page} \geq 2 \text{ And} < 4 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} & \end{cases}$$

b) Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the "onMouseOver" event, and check if it makes any changes on the status bar.

$$\text{Rule: IF}\begin{cases}\text{onMouseOver Changes Status Bar} \rightarrow \text{Phishing} \\ \quad \text{It Does't Change Status Bar} \rightarrow \text{Legitimate}\end{cases}$$

c) Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as "Using onMouseOver to hide the Link". Nonetheless, for this feature, we will search for event "event.button==2" in the webpage source code and check if the right click is disabled.

Rule: IF$\begin{cases} \text{Right Click Disabled} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

d) Using Pop-up Window

It is unusual to find a legitimate website asking users to submit their personal information through a pop-up window. On the other hand, this feature has been used in some legitimate websites and its main goal is to warn users about fraudulent activities or broadcast a welcome announcement, though no personal information was asked to be filled in through these pop-up windows.

Rule: IF $\begin{cases} \text{Popoup Window Contains Text Fields} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

e) IFrame Redirection

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e., without frame borders. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

Rule: IF $\begin{cases} \text{Using iframe} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

Domain based Features

1. Age of Domain

This feature can be extracted from WHOIS database (Whois 2005). Most phishing websites live for a short period of time. By reviewing our dataset, we find that the minimum age of the legitimate domain is 6 months.

Rule: IF $\begin{cases} \text{Age Of Domain} \geq 6 \text{ months} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

2. DNS Record

For phishing websites, either the claimed identity is not recognized by the WHOIS database (Whois 2005) or no records founded for the hostname (Pan and Ding 2006). If the DNS record is empty or not found then the website is classified as "Phishing", otherwise it is classified as "Legitimate".

$$\text{Rule: IF}\begin{cases}\text{no DNS Record For The Domain} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate}\end{cases}$$

3. Website Traffic

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time, they may not be recognized by the Alexa database (Alexa the Web Information Company., 1996). By reviewing our dataset, we find that in worst scenarios, legitimate websites ranked among the top 100,000. Furthermore, if the domain has no traffic or is not recognized by the Alexa database, it is classified as "Phishing". Otherwise, it is classified as "Suspicious".

$$\text{Rule: IF}\begin{cases}\text{Website Rank} < 100{,}000 \rightarrow \text{Legitimate} \\ \text{Website Rank} > 100{,}000 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phish}\end{cases}$$

4. PageRank

PageRank is a value ranging from "0" to "1". PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage. In our datasets, we find that about 95% of phishing webpages have no PageRank. Moreover, we find that the remaining 5% of phishing webpages may reach a PageRank value up to "0.2".

$$\text{Rule: IF}\begin{cases}\text{PageRank} < 0.2 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate}\end{cases}$$

5. Google Index

This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results (Webmaster resources, 2014).

Usually, phishing webpages are merely accessible for a short period and as a result, many phishing webpages may not be found on the Google index.

Rule: IF $\begin{cases} \text{Webpage Indexed by Google} \rightarrow \text{Legitimate} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$

6. **Number of Links Pointing to Page**

The number of links pointing to the webpage indicates its legitimacy level, even if some links are of the same domain (Dean, 2014). In our datasets and due to its short life span, we find that 98% of phishing dataset items have no links pointing to them. On the other hand, legitimate websites have at least 2 external links pointing to them.

Rule:

IF

$$\begin{cases} \text{Of Link Pointing to The Webpage} = 0 \rightarrow \text{Phishing} \\ \text{Of Link Pointing to The Webpage} > 0 \text{ and} \leq 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

7. **Statistical-Reports Based Feature**

Several parties such as PhishTank (PhishTank Stats, 2010-2012), and StopBadware (StopBadware, 2010-2012) formulate numerous statistical reports on phishing websites at every given period of time; some are monthly and others are quarterly. In our research, we used 2 forms of the top ten statistics from PhishTank: “Top 10 Domains” and “Top 10 IPs” according to statistical-reports published in the last three years, starting in January2010 to November 2012. Whereas for “StopBadware”, we used “Top 50” IP addresses.

Rule:

IF $\begin{cases} \text{Host Belongs to Top Phishing IPs or Top Phishing Domains} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$

3.5 Implementation

I) Phase 1

Firstly before starting with the implementation of Deep learning models on our dataset we analyzed the implementations done in [1] where Machine learning algorithms like Logistic Regression, K- Nearest Neighbor, Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, Ada-boost and XG-boost were utilized to train models using the dataset for phishing detection provided UCI Machine Learning Repository. The dataset has 11055 data points with 6157 legitimate URLs and 4898 phishing URLs. Benign URLs are labelled as “0” and phishing URLs are labelled as “1” in the target column.

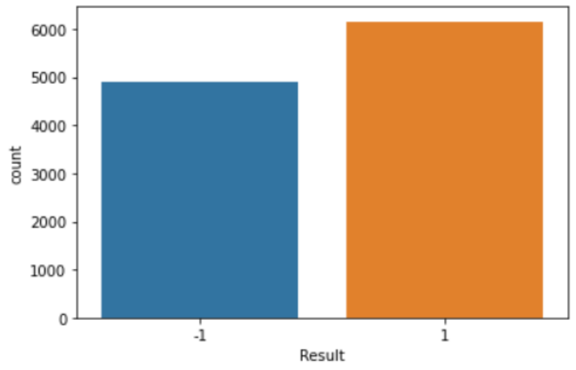
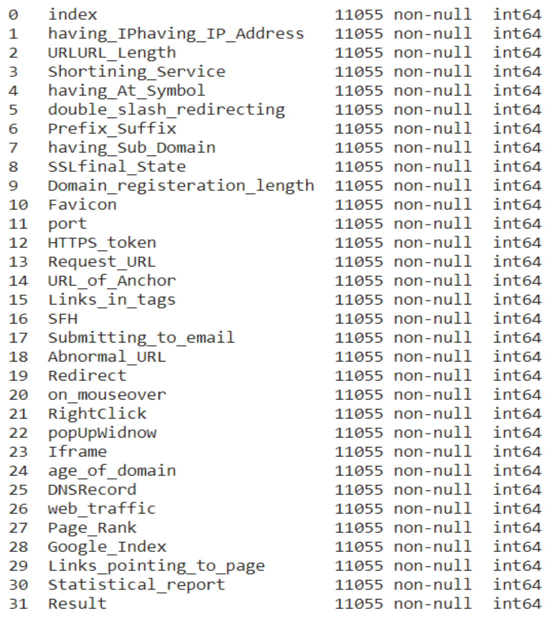


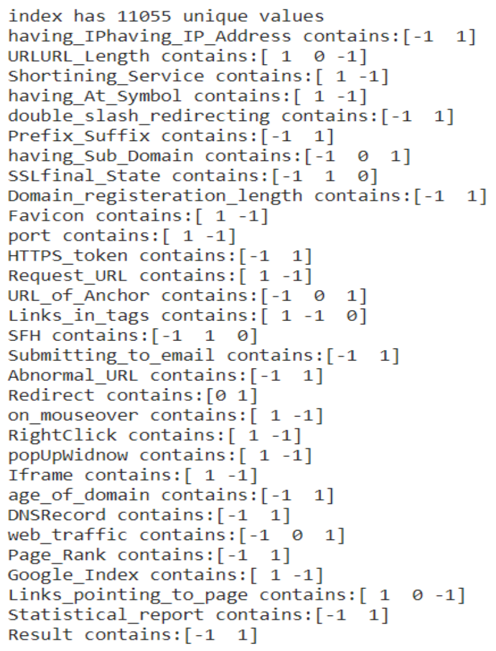
Fig. 10 - Bar plot for type count in UCI dataset

Each datapoint had 30 features subdivided into following categories having the [1,0,-1] in each category:



```
0   index                        11055 non-null  int64
1   having_IPhaving_IP_Address   11055 non-null  int64
2   URLURL_Length                11055 non-null  int64
3   Shortining_Service           11055 non-null  int64
4   having_At_Symbol             11055 non-null  int64
5   double_slash_redirecting     11055 non-null  int64
6   Prefix_Suffix                11055 non-null  int64
7   having_Sub_Domain            11055 non-null  int64
8   SSLfinal_State               11055 non-null  int64
9   Domain_registeration_length  11055 non-null  int64
10  Favicon                      11055 non-null  int64
11  port                         11055 non-null  int64
12  HTTPS_token                  11055 non-null  int64
13  Request_URL                  11055 non-null  int64
14  URL_of_Anchor                11055 non-null  int64
15  Links_in_tags                11055 non-null  int64
16  SFH                          11055 non-null  int64
17  Submitting_to_email          11055 non-null  int64
18  Abnormal_URL                 11055 non-null  int64
19  Redirect                     11055 non-null  int64
20  on_mouseover                 11055 non-null  int64
21  RightClick                   11055 non-null  int64
22  popUpWidnow                  11055 non-null  int64
23  Iframe                       11055 non-null  int64
24  age_of_domain                11055 non-null  int64
25  DNSRecord                    11055 non-null  int64
26  web_traffic                  11055 non-null  int64
27  Page_Rank                    11055 non-null  int64
28  Google_Index                 11055 non-null  int64
29  Links_pointing_to_page       11055 non-null  int64
30  Statistical_report           11055 non-null  int64
31  Result                       11055 non-null  int64
```

Fig. 11 - Features in UCI dataset



```
index has 11055 unique values
having_IPhaving_IP_Address contains:[-1  1]
URLURL_Length contains:[ 1  0 -1]
Shortining_Service contains:[ 1 -1]
having_At_Symbol contains:[ 1 -1]
double_slash_redirecting contains:[-1  1]
Prefix_Suffix contains:[-1  1]
having_Sub_Domain contains:[-1  0  1]
SSLfinal_State contains:[-1  1  0]
Domain_registeration_length contains:[-1  1]
Favicon contains:[ 1 -1]
port contains:[ 1 -1]
HTTPS_token contains:[-1  1]
Request_URL contains:[ 1 -1]
URL_of_Anchor contains:[-1  0  1]
Links_in_tags contains:[ 1 -1  0]
SFH contains:[-1  1  0]
Submitting_to_email contains:[-1  1]
Abnormal_URL contains:[-1  1]
Redirect contains:[0 1]
on_mouseover contains:[ 1 -1]
RightClick contains:[ 1 -1]
popUpWidnow contains:[ 1 -1]
Iframe contains:[ 1 -1]
age_of_domain contains:[-1  1]
DNSRecord contains:[-1  1]
web_traffic contains:[-1  0  1]
Page_Rank contains:[-1  1]
Google_Index contains:[ 1 -1]
Links_pointing_to_page contains:[ 1  0 -1]
Statistical_report contains:[-1  1]
Result contains:[-1  1]
```

Fig.12 - Unique values of features in UCI dataset

Here 1 means legitimate website feature, 0 is suspicious website feature and -1 is phishing website feature. The Result column that is our target variable has values [1,-1] that represent the labels for each of the records classifying them as legitimate or phishing websites.

Next, we analyze the correlation in the features using a heat map:

In the heatmap, the lighter the color corresponding to 2 variables the more correlated they are. The black boxes show negative correlation. We can see here that popup window and favicon features are highly correlated and one of them could be dropped. Eliminating features based on their correlation actually gives good accuracy which is why we did the above steps.

The features pop up Window and favicon are highly correlated. Dropping one of the features or not is an experimental decision. A favicon is a graphic image (icon) associated with a specific webpage. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.

Some features are negatively correlated. Negative correlations are where one feature marks it as phishing and other don't. The variable “SSL final state” is the most correlated with the result whereas “domain registration length” is the least. SSL final state checks the certificate assigned with HTTPS including the extent of the trust certificate issuer, and the certificate age and Domain registration length checks whether the domain registration expires within a year as most fraudulent websites have been used for 1 year only.

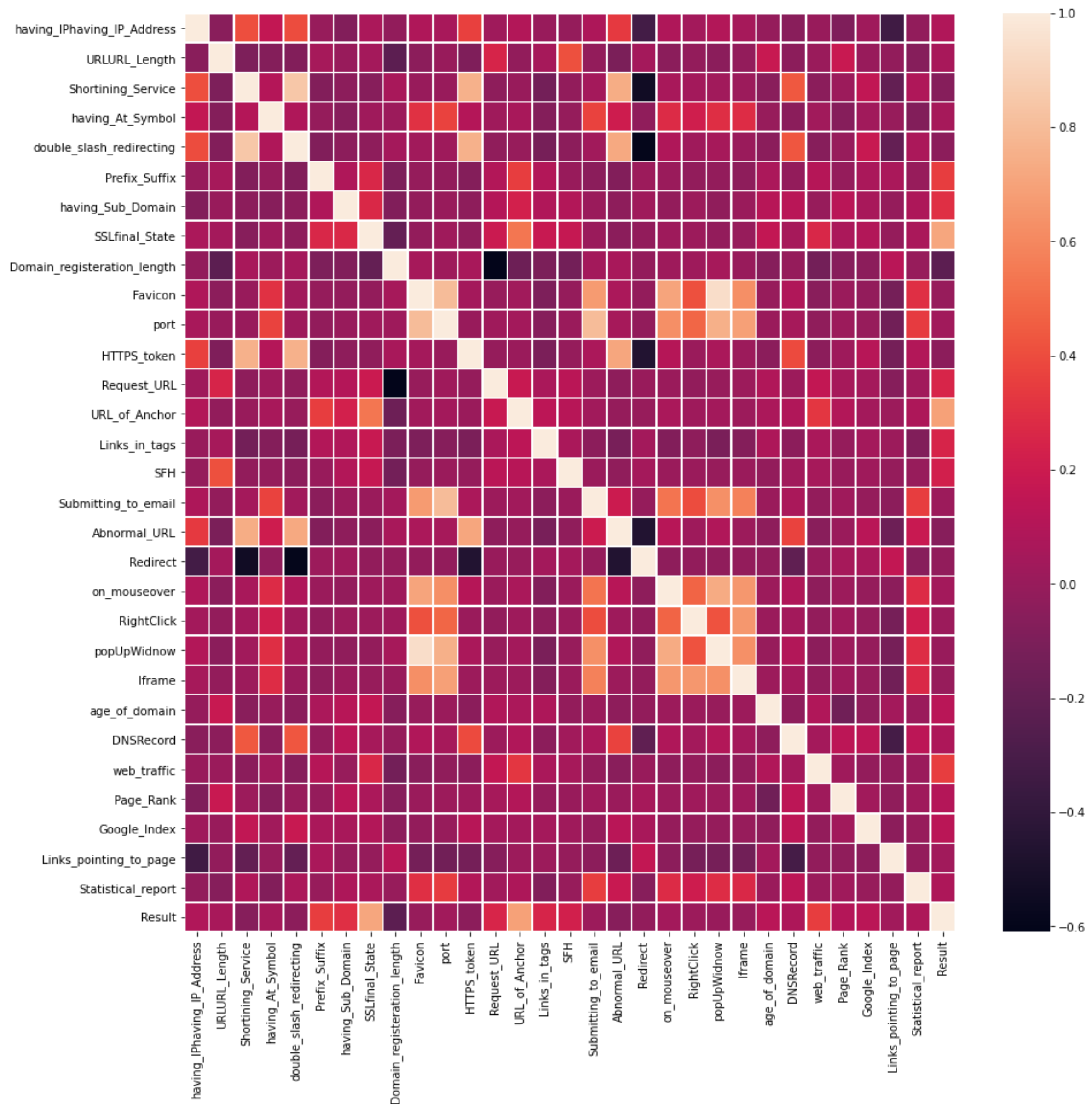


Fig. 13 - Heatmap showing correlation between features in UCI dataset

The next step is the feature selection that is choosing our independent variables on which we train our model for which we have used concept of Information gain:

Information Gain (IG) can be used to filter out the predominant features from the given datasets and use it as a ranking tool. The ranking criterion can be used to analyze the relevance of these features which may help in classification and detection of malicious and phishing websites. The features which are mutually exclusive and dependent on class labels can be used to differentiate from those features which are independent of each other thus helping in detection of phishing websites. The following two steps are used to calculate the gain of a given feature:

1. Entropy (class-label) calculated for the whole dataset, can be calculated from the below formula:

$$\text{info}(S) = \sum_{i=1}^{k} p_i \times \log_2 p_i$$

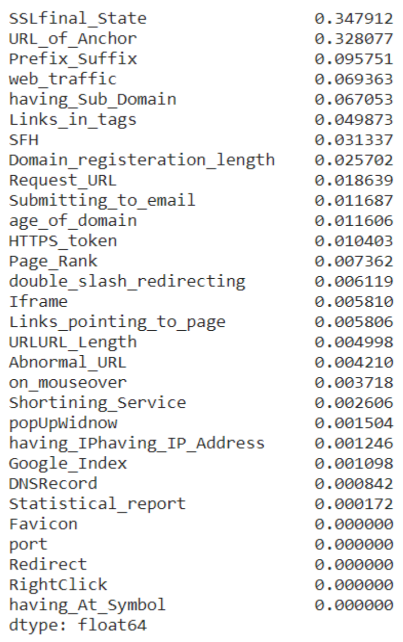
2. Calculation of entropy for each inimitable value of that definite feature:

$$\text{info}_A(S) = \sum_{i=1}^{y} |S_i| \big/ |S| * \text{info}(S_i)$$

3. The calculation of the IG is given by:

$$IG(A) = \text{info}(S) - \text{info}_A(S)$$

The information gain arranged in decreasing order for this dataset and we then select only those features that give us IG value greater than 0 to form a data frame to train our model:



```
SSLfinal_State                 0.347912
URL_of_Anchor                  0.328077
Prefix_Suffix                  0.095751
web_traffic                    0.069363
having_Sub_Domain              0.067053
Links_in_tags                  0.049873
SFH                            0.031337
Domain_registeration_length    0.025702
Request_URL                    0.018639
Submitting_to_email            0.011687
age_of_domain                  0.011606
HTTPS_token                    0.010403
Page_Rank                      0.007362
double_slash_redirecting       0.006119
Iframe                         0.005810
Links_pointing_to_page         0.005806
URLURL_Length                  0.004998
Abnormal_URL                   0.004210
on_mouseover                   0.003718
Shortining_Service             0.002606
popUpWidnow                    0.001504
having_IPhaving_IP_Address     0.001246
Google_Index                   0.001098
DNSRecord                      0.000842
Statistical_report             0.000172
Favicon                        0.000000
port                           0.000000
Redirect                       0.000000
RightClick                     0.000000
having_At_Symbol               0.000000
dtype: float64
```

Fig. 14 - Information Gain values for features in UCI dataset

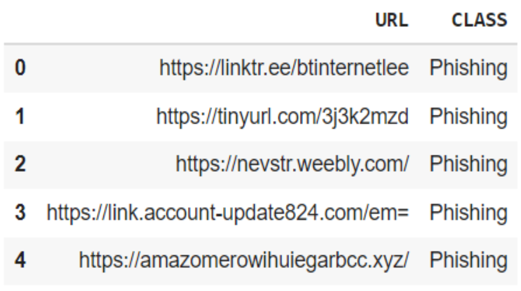
We have then applied PCA or Principal Component Analysis to our reduced dataset in order to perform dimensionality reduction. Following are reasons for Dimensionality Reduction:

- Dimensionality Reduction helps in data compression, and hence reduced storage space.
- It reduces computation time and helps remove redundant features, if any.
- Removes Correlated Features.
- It is helpful in noise removal also and as a result of that, we can improve the performance of models.

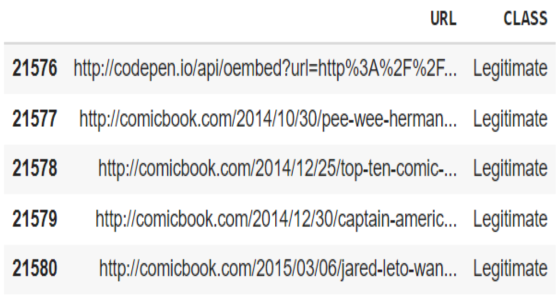
Principal component analysis is a statistical technique to reduce the dimension of the data by selecting the most important features that capture maximum information about the dataset. The features are selected on the basis of variance that they cause in the output. The feature that causes the highest variance is the first principal component, the one responsible for the second highest value is the second component and so on. We then applied the Machine learning algorithms mentioned in the beginning and calculated their efficiency using confusion matrix as well as their accuracies as have been mentioned in the results section.

II) Phase 2

We have firstly created a custom dataset that consists of 21,581 websites collected from Phish tank for phishing sites and Alexa for legitimate sites. The dataset contains websites in raw form along with the class labels as 'Phishing' or 'Legitimate'.



	URL	CLASS
0	https://linktr.ee/btinternetlee	Phishing
1	https://tinyurl.com/3j3k2mzd	Phishing
2	https://nevstr.weebly.com/	Phishing
3	https://link.account-update824.com/em=	Phishing
4	https://amazomerowihuiegarbcc.xyz/	Phishing



	URL	CLASS
21576	http://codepen.io/api/oembed?url=http%3A%2F%2F...	Legitimate
21577	http://comicbook.com/2014/10/30/pee-wee-herman...	Legitimate
21578	http://comicbook.com/2014/12/25/top-ten-comic-...	Legitimate
21579	http://comicbook.com/2014/12/30/captain-americ...	Legitimate
21580	http://comicbook.com/2015/03/06/jared-leto-wan...	Legitimate

Fig.15 - Custom Dataset

The entire dataset consists of 11000 legitimate websites and 10581 phishing websites.

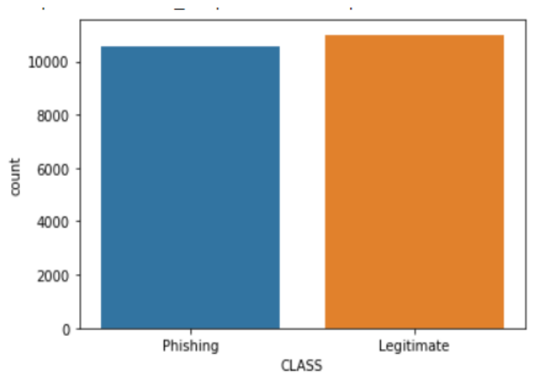
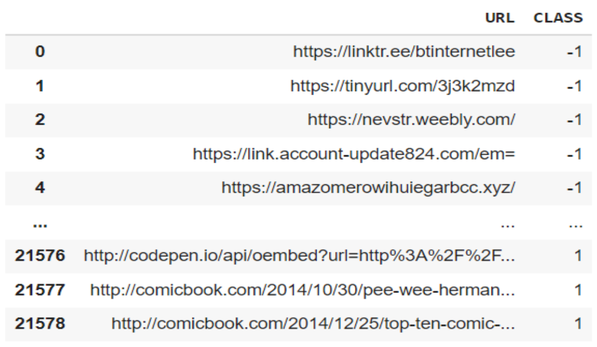


Fig.16 - Bar plot for type count in Custom Dataset

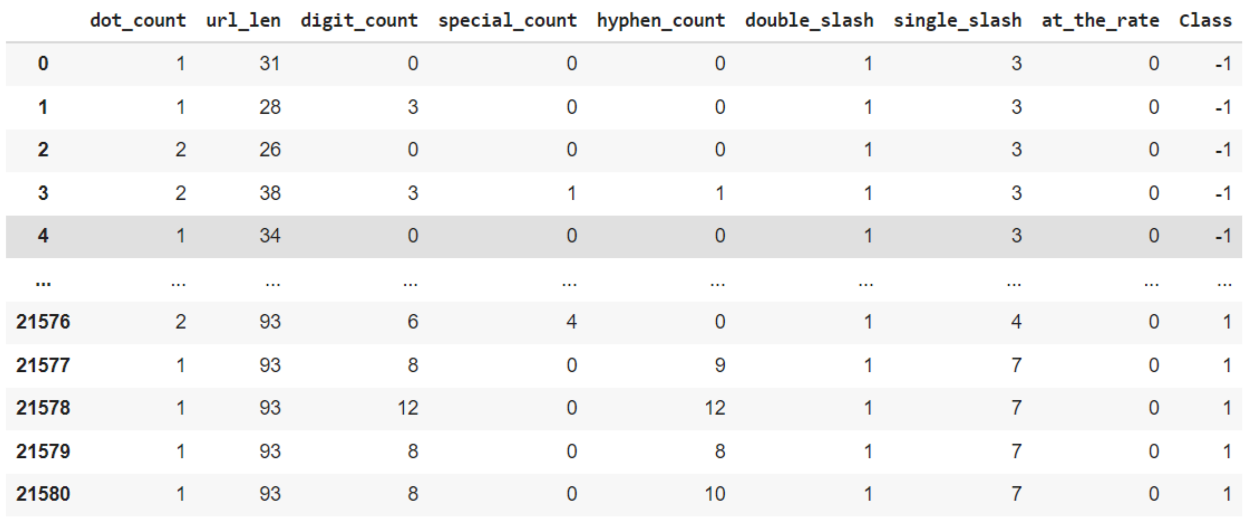
We have then replaced the categorical terms Phishing and Legitimate in the dataset by -1 and 1 respectively



	URL	CLASS
0	https://linktr.ee/btinternetlee	-1
1	https://tinyurl.com/3j3k2mzd	-1
2	https://nevstr.weebly.com/	-1
3	https://link.account-update824.com/em=	-1
4	https://amazomerowihuiegarbcc.xyz/	-1
...	...	...
21576	http://codepen.io/api/oembed?url=http%3A%2F%2F...	1
21577	http://comicbook.com/2014/10/30/pee-wee-herman...	1
21578	http://comicbook.com/2014/12/25/top-ten-comic-...	1

Fig.17 - New Labelled Custom Dataset

As we cannot directly train our model using raw URLs we first perform lexical feature extraction for a number of features like counting the number of dots in the URL, the URL length, counting the number of digits in the URL, counting the number of special characters like ';' , '+’, ‘=', '_', '?', '=' , '&' , '[',']' in the URL, counting the number of hyphens ,counting the number of double and single slashes, counting the number of ‘@’ symbols, etc.

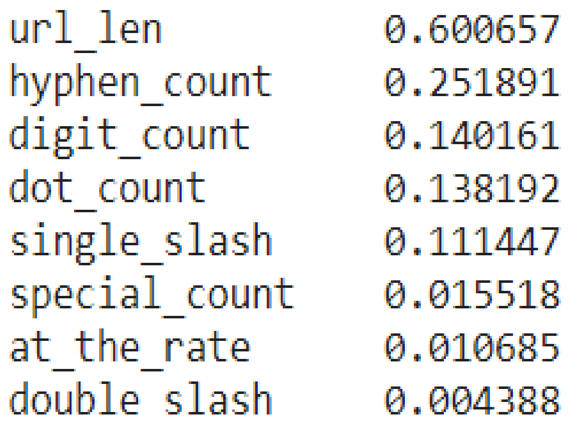


	dot_count	url_len	digit_count	special_count	hyphen_count	double_slash	single_slash	at_the_rate	Class
0	1	31	0	0	0	1	3	0	-1
1	1	28	3	0	0	1	3	0	-1
2	2	26	0	0	0	1	3	0	-1
3	2	38	3	1	1	1	3	0	-1
4	1	34	0	0	0	1	3	0	-1
...	...	...	...	...	...	...	...	...	...
21576	2	93	6	4	0	1	4	0	1
21577	1	93	8	0	9	1	7	0	1
21578	1	93	12	0	12	1	7	0	1
21579	1	93	8	0	8	1	7	0	1
21580	1	93	8	0	10	1	7	0	1

Fig.18 - New dataset formed by extracting features from custom dataset

We then perform the same steps of calculating the information gain and applying all the Machine Learning algorithms to know how the accuracy of their models differs with the dataset that we have created.

Here since we have 8 features and all give us IG values greater than 0 we don’t drop any features:



```
url_len          0.600657
hyphen_count     0.251891
digit_count      0.140161
dot_count        0.138192
single_slash     0.111447
special_count    0.015518
at_the_rate      0.010685
double_slash     0.004388
```

Fig.19 - Information gain values for New Dataset

III) Phase 3

After this we begin the implementation of the Deep learning algorithms on our dataset. Following are the parameters and their descriptions that we have considered while making the model:

1. **Optimizer**

   Optimizers are algorithms or methods used to minimize an error function (*loss function*) or to maximize the efficiency of production. Optimizers are mathematical functions which are dependent on model's learnable parameters, i.e., weights and biases. Optimizers help to know how to change weights and learning rate of neural network to reduce the losses.

   - Adam (Adaptive Moment Estimation): Adam optimizer is one of the most popular and famous gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients, similar to momentum and also the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods.

   - RMSprop: The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster. The gist of RMSprop is to maintain a moving (discounted) average of the square of gradients and divide the gradient by the root of this average

2. **Layers**

   The 1D Convolution layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. Finally, if activation is not None, it is applied to the outputs as well but in this case, we have given Relu as activation. When using this layer as the first layer in a model, provide an input_shape argument which in this model is tuple of (21581,7). The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer.

Embedding layer is mainly used in Natural Language Processing related applications such as language modeling, but it can also be used with other tasks that involve neural networks. Embedding layer enables us to convert each word into a fixed length vector of defined size. The resultant vector is a dense one with having real values instead of just 0's and 1's. The fixed length of word vectors helps us to represent words in a better way along with reduced dimensions. 1D Max Pooling layer is used to perform max pooling operations on temporal data. Here, pool_size refers the max pooling windows and strides refer the factors for downscale. Similarly, MaxPooling2D and MaxPooling3D are used for Max pooling operations for spatial data.

Stacked LSTMs are now a stable technique for challenging sequence prediction problems. A Stacked LSTM architecture can be defined as an LSTM model comprised of multiple LSTM layers. An LSTM layer above provides a sequence output rather than a single value output to the LSTM layer below. Specifically, one output per input time step, rather than one output time step for all input time steps. The SimpleRNN processes data in batches, just like every other neural network. So, this layer takes as input the tensor (batch_size, timesteps, input_features). All recurrent layers in keras can be run in two different modes - All of the successive outputs aka the states and just the last state.

3. Activation Function

The activation function is a non-linear transformation that we do over the input before sending it to the next layer of neurons or finalizing it as output. We have used ReLu.

- Rectified Linear Unit (ReLU): The ReLU is the most used activation function in almost all the convolutional neural networks or deep learning.

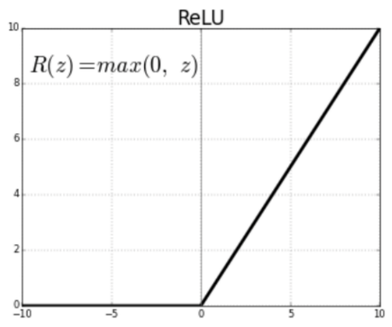


Fig. 20 - Rectified Linear Unit

The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

- Sigmoid: The Sigmoid Function curve looks like a S-shape.

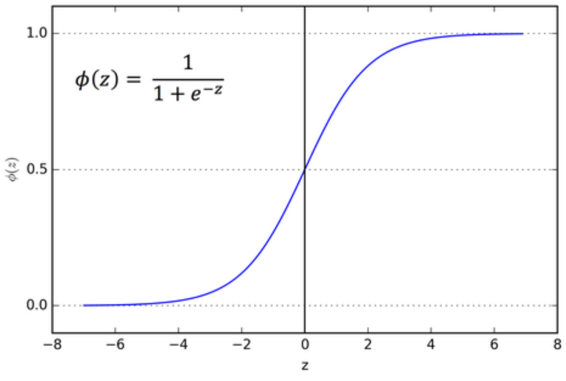


Fig. 21 - Sigmoid Function

The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points. The function is monotonic but function’s derivative is not. The logistic sigmoid function can cause a neural network to get stuck at the training time. The softmax function is a more generalized logistic activation function which is used for multiclass classification.

4. **Loss Function**

   The loss function tells how good the model is in predictions. If the model predictions are closer to the actual values the loss will be minimum and if the predictions are totally away from the original values the loss value will be the maximum.

   - Binary Cross entropy: Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.

5. **Callbacks**

   A callback is a set of functions to be applied at various stages of the training process. Callbacks can be used to get a view on internal states and statistics of the model during training.

   We use a callback when we want to automate some tasks after every training/epoch that help us to have control over the training process. This includes stopping training when reach a certain accuracy/loss score, saving the model as a checkpoint after each successful epoch, adjusting the learning rates over time, and more.

6. **Epoch**

   One Epoch is when an entire dataset is passed forward and backward through the neural network only once. The batch size is a hyperparameter of gradient descent that controls the number of training samples to work through before the model’s internal parameters are updated.

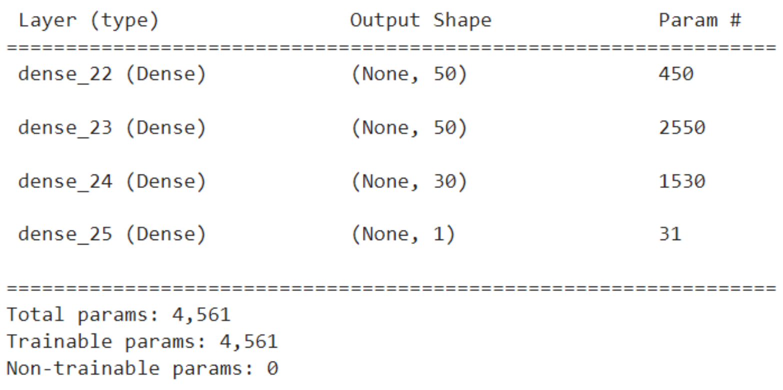
7. **Verbose**

By setting verbose 0, 1 or 2 we just say how do we want to 'see' the training progress for each epoch. We have assumed verbose to be 1.

- verbose=0 will show nothing (silent)
- verbose=1 will show an animated progress.
- verbose=2 will just mention the number of epochs.

A. ANN – Artificial Neural Network

1. **Optimizer -** Adam Optimizer.
2. **Layers -** We have assumed dense layers to be penetrating in ratio of 50, 50, 30 and 1.



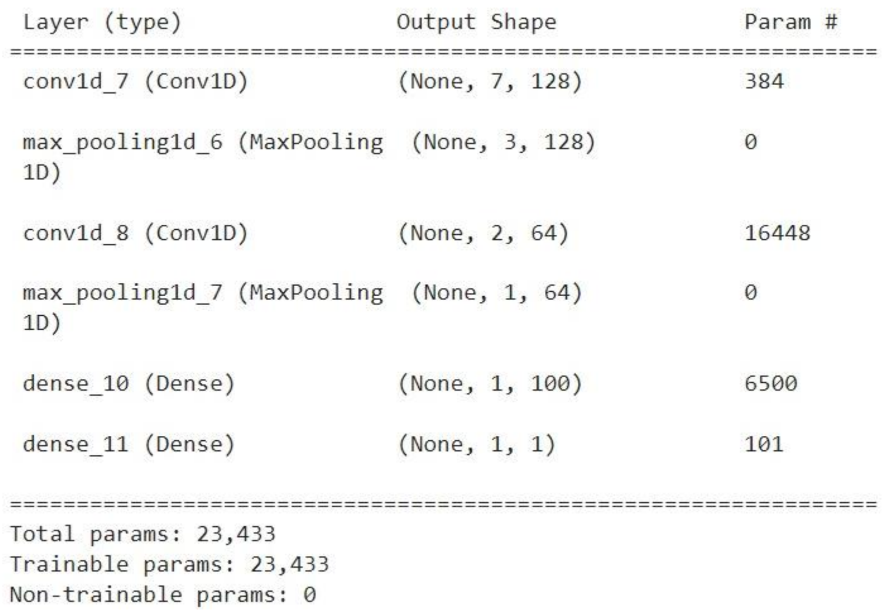
```
 Layer (type)                Output Shape              Param #
=================================================================
 dense_22 (Dense)            (None, 50)                450

 dense_23 (Dense)            (None, 50)                2550

 dense_24 (Dense)            (None, 30)                1530

 dense_25 (Dense)            (None, 1)                 31

=================================================================
Total params: 4,561
Trainable params: 4,561
Non-trainable params: 0
```

Fig. 22 – ANN layers summary

3. **Activation Function-** We have used ReLu and sigmoid.
4. **Loss Function -** For ANN we have used Binary Cross entropy.
5. **Callbacks-** We used early stopping for ANN
6. **Epoch-** We have taken epochs to be 30 with a batch size of 32.
7. **Verbose -** We have assumed verbose to be 1. Which means it will show an animated progress.

B. CNN – Convolutional Neural Network

1. **Optimizer -** Adam Optimizer.
2. **Layers -** We have assumed dense layers to be penetrating in ratio of 50, 50, 30 and 1.



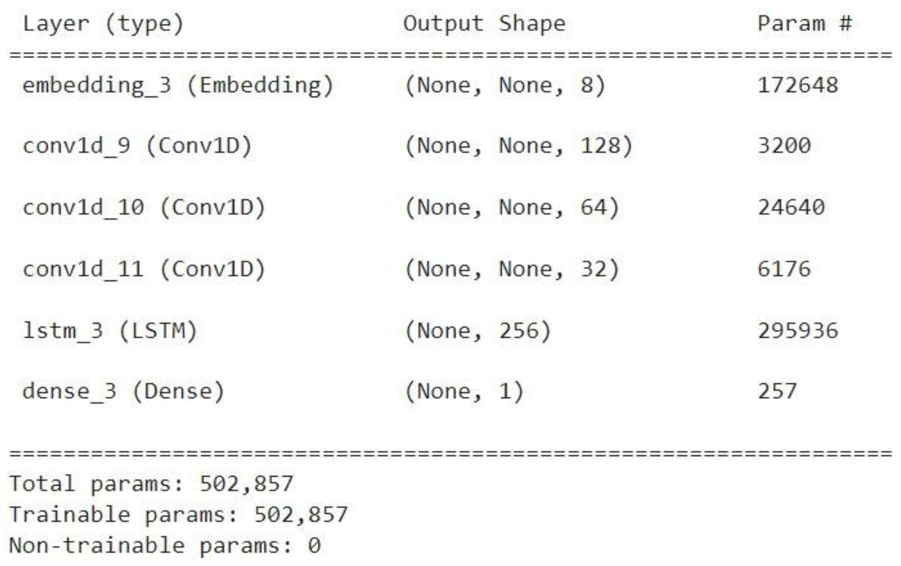
```
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_7 (Conv1D)           (None, 7, 128)            384

 max_pooling1d_6 (MaxPooling  (None, 3, 128)           0
 1D)

 conv1d_8 (Conv1D)           (None, 2, 64)             16448

 max_pooling1d_7 (MaxPooling  (None, 1, 64)            0
 1D)

 dense_10 (Dense)            (None, 1, 100)            6500

 dense_11 (Dense)            (None, 1, 1)              101

=================================================================
Total params: 23,433
Trainable params: 23,433
Non-trainable params: 0
```

Fig. 23 – CNN layers summary

3. **Activation Function-** We have used ReLu and sigmoid.
4. **Loss Function -** For ANN we have used Binary Cross entropy.
5. **Callbacks-** We used early stopping for ANN
6. **Epoch-** We have taken epochs to be 30 with a batch size of 32.
7. **Verbose -** We have assumed verbose to be 1. Which means it will show an animated progress.

C. CNN + LSTM

1. **Optimizer -** We have used the Adam Optimizer.
2. **Layers -** We first used an embedding layer, 3 1D Convolution layers and 1 LSTM layer. Lastly, we have added 1 dense layer at the end of the model.



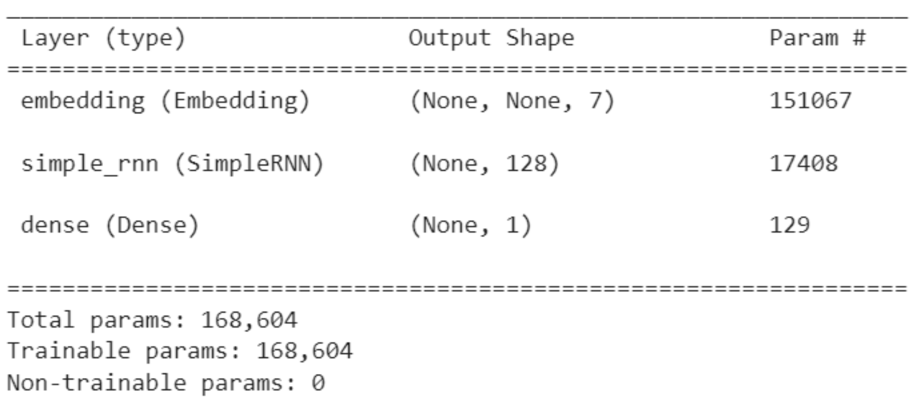
```
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, None, 8)           172648

 conv1d_9 (Conv1D)           (None, None, 128)         3200

 conv1d_10 (Conv1D)          (None, None, 64)          24640

 conv1d_11 (Conv1D)          (None, None, 32)          6176

 lstm_3 (LSTM)               (None, 256)               295936

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 502,857
Trainable params: 502,857
Non-trainable params: 0
```

Fig. 24 – CNN + LSTM layers summary

3. **Activation Function-** We have used ReLu and sigmoid.
4. **Loss Function -** For CNN+LSTM we have used Binary Cross entropy.
5. **Epoch-** We have taken epochs to be 30 with a batch size of 32.
6. **Verbose-** We have assumed verbose to be 1. Which means it will show an animated progress.

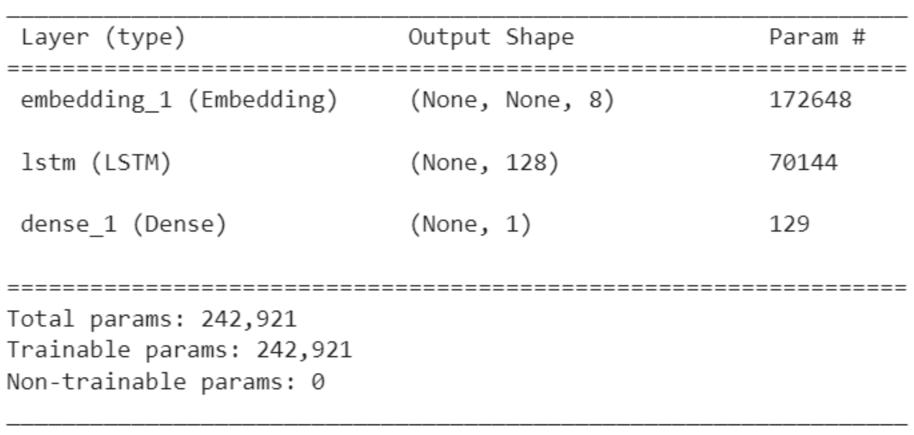
D. RNN + LSTM

1. **Optimizer -** We have used the RMSprop Optimizer.
2. **Layers –** For the RNN model, we first used an embedding layer and then used a SimpleRNN layer. Lastly, we have added 1 dense layer at the end of the model. For the LSTM model, again we used 1 embedding layer and then used an LSTM layer followed by a dense layer. We combined both models to create the RNN LSTM model.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 7)           151067

 simple_rnn (SimpleRNN)      (None, 128)               17408

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 168,604
Trainable params: 168,604
Non-trainable params: 0
```

Fig. 25 – RNN layers summary



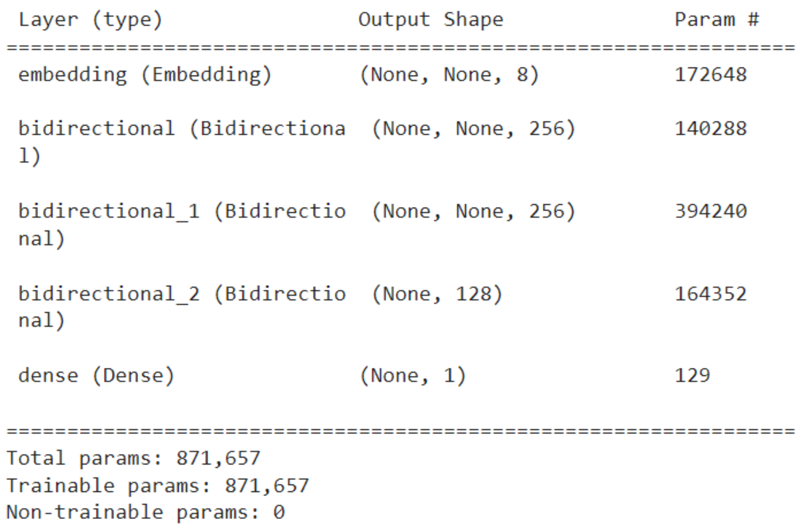
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 8)           172648

 lstm (LSTM)                 (None, 128)               70144

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 242,921
Trainable params: 242,921
Non-trainable params: 0
_________________________________________________________________
```

Fig. 26- LSTM layers summary

3. **Activation Function-** We have used sigmoid.
4. **Loss Function -** For RNN+LSTM we have used Binary Cross entropy.
5. **Epoch-** We have taken epochs to be 10 with a batch size of 60.
6. **Verbose-** We have assumed verbose to be 1. Which means it will show an animated progress.

E. Bidirectional LSTM

1. **Optimizer -** We have used the Adam Optimizer.
2. **Layers -** We first used an embedding layer. Then we used 4 Bidirectional layers. Lastly, we have added 1 dense layer at the end of the model.



```
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 8)           172648

 bidirectional (Bidirectiona  (None, None, 256)        140288
 l)

 bidirectional_1 (Bidirectio  (None, None, 256)        394240
 nal)

 bidirectional_2 (Bidirectio  (None, 128)              164352
 nal)

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 871,657
Trainable params: 871,657
Non-trainable params: 0
```

Fig 27 – Bidirectional LSTM layers summary

3. **Activation Function-** We have used sigmoid as activation.
4. **Loss Function -** For Bidirectional LSTM we have used Binary Cross entropy.
5. **Callbacks-** We used early stopping for Bidirectional LSTM.
6. **Epoch-** We have taken epochs to be 30 with a batch size of 32.
7. **Verbose-** We have assumed verbose to be 1. Which means it will show an animated progress.

IV) Phase 4

In phase 4 we built a website using Flask and ngrok. It would take an input of the website URL and output whether the URL is legitimate or phishing. Once the software is checked, and no bugs or errors are occured, then it is deployed. For deploying our model into web application, we will use python framework Flask/Django to connect CNN model with HTML, CSS & JavaScript file. Creation of web application will be done using HTML, CSS & JavaScript and then connect web application with Flask/Django. After the software is deployed, then its maintenance begins. ngrok link is a specialized, enhanced version of ngrok specifically designed for running in production environments. Specifically, it is intended for two major use cases:

- It is a lightweight VPN alternative that provides the automation and security necessary to establish targeted, secure links into customer environments.

- It enables IoT devices to expose control functionality to customers or administrators at stable, secured endpoints. Remote shell access for debugging and administration can be safely exposed in this manner as well.

ngrok link is tuned for running optimally as part of your infrastructure and exposes a number of additional security features to give fine grained access and authentication control. Most importantly, these features are exposed via APIs so that you can automate your entire workflow with ngrok.

For production systems, every client must authenticate with a unique authtoken credential. This allows you to deactivate devices that are old or compromised. Further, it allows you to enforce a separate ACL policy on every connected device that limits what tunnels it is allowed to bind. You can create authtokens from the Tunnel Authtokens tab in your ngrok dashboard. Click "Add Tunnel Authtoken" and then enter a human-readable description of the device or location where you intend to install the authtoken for tracking purposes. In both the UI and API, the full authtoken you generate will only be shown once, immediately after creation.

Chapter 4
Results and Analysis

Results of Phase 1

In [1] only the “Random Forest” and “Decision Tree" algorithms are used. But along with these we decided to do a comparison between various algorithms, such as the Logistic Regression algorithm, Naive Bayes algorithm, KNN Classifier, Support Vector, AdaBoost and XG Boost Classifier. The comparison conducted on them is based on the efficiency and inferences made on them based on the confusion matrix for each algorithm. A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. Calculating a confusion matrix can give us a better idea of what the classification model is getting right and what types of errors it is making.

With KNN Classifier algorithm, the number of false positives are very less compared to the correct prediction. The chart depicts that out of all the -1 predicted 92% of the time it is predicted correctly and out of the total times -1 is actually there in the dataset, 94% of the time it is predicted correctly.

```
              precision    recall  f1-score   support

          -1       0.92      0.94      0.93      1474
           1       0.95      0.94      0.94      1843

    accuracy                           0.94      3317
   macro avg       0.93      0.94      0.94      3317
weighted avg       0.94      0.94      0.94      3317

<matplotlib.axes._subplots.AxesSubplot at 0x7f1cbb114ed0>
```

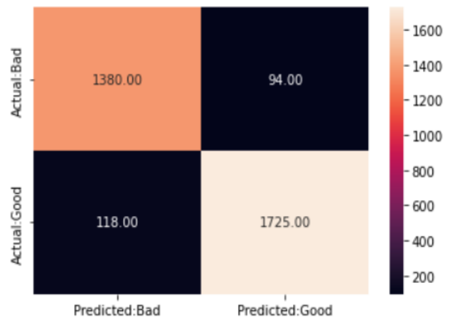


Fig. 28 - Confusion Matrix for KNN algorithm using UCI dataset

For Decision Tree, the precision for -1 is 80% and recall is 88 and the map shows the number of false positives have increased:

```
              precision    recall  f1-score   support

          -1       0.80      0.88      0.84      1361
           1       0.91      0.85      0.88      1956

    accuracy                           0.86      3317
   macro avg       0.86      0.87      0.86      3317
weighted avg       0.87      0.86      0.86      3317

<matplotlib.axes._subplots.AxesSubplot at 0x7f1cbb06e8d0>
```

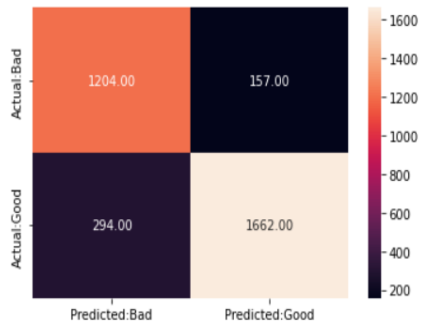


Fig. 29 - Confusion Matrix for Decision Tree algorithm using UCI dataset

Support Vector Machine gave the best results overall. The precision for -1 being 92% and recall being 96. The map also shows the number of false positives has decreased and true positives have increased.

```
              precision    recall  f1-score   support

          -1       0.92      0.96      0.94      1445
           1       0.97      0.94      0.95      1872

    accuracy                           0.95      3317
   macro avg       0.94      0.95      0.95      3317
weighted avg       0.95      0.95      0.95      3317

<matplotlib.axes._subplots.AxesSubplot at 0x7f1cba0cc050>
```

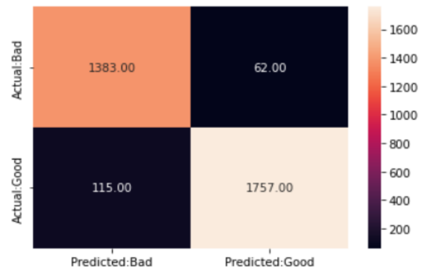
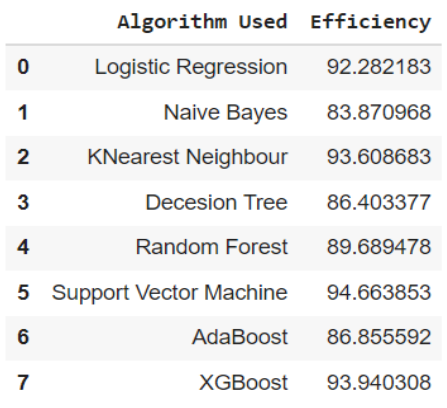


Fig.30 - Confusion Matrix for SVM algorithm using UCI dataset

The accuracy calculated from each of the algorithms are tabulated in the image given below. From this the conclusion is drawn that the maximum accuracy of 94.7% is for the Support Vector Machine algorithm. Random forest is 89.4% accurate and Decision Tree is 86.4% accurate whereas the accuracy achieved in the paper for the same algorithms were 95% and 91% respectively.



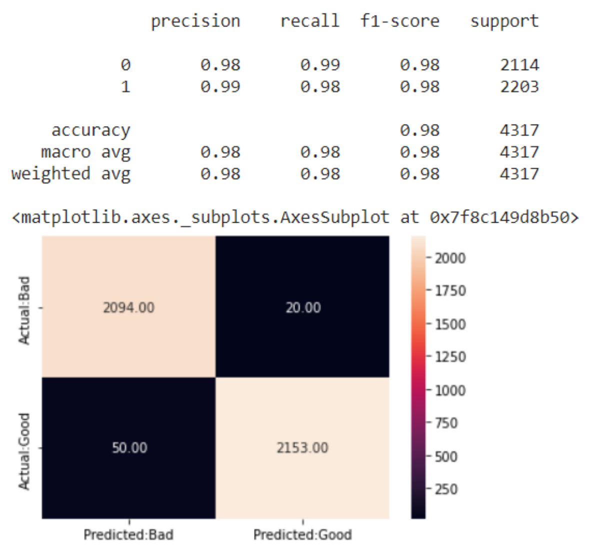
	Algorithm Used	Efficiency
0	Logistic Regression	92.282183
1	Naive Bayes	83.870968
2	KNearest Neighbour	93.608683
3	Decesion Tree	86.403377
4	Random Forest	89.689478
5	Support Vector Machine	94.663853
6	AdaBoost	86.855592
7	XGBoost	93.940308

Fig. 31 - Accuracy comparison for ML algorithms using UCI dataset

Results of Phase 2

For the dataset that we created we performed the analysis of the same Machine Learning algorithms and following were the results:

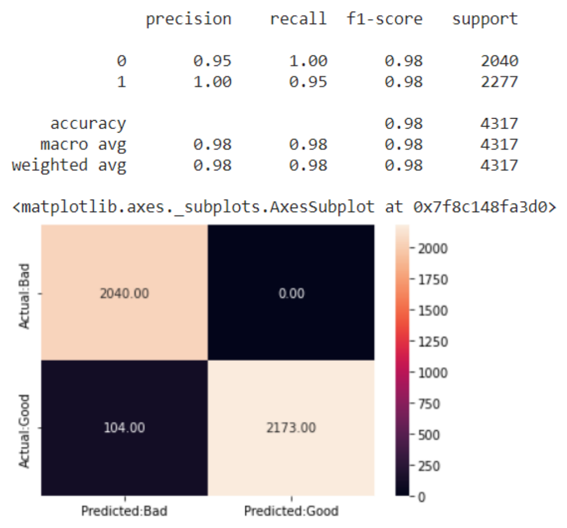
With KNN Classifier algorithm, the number of false positives are very less compared to the correct prediction. The chart depicts that out of all the 0 predicted 98% of the time it is predicted correctly and out of the total times +1 is actually there in the dataset, 99% of the time it is predicted correctly. The KNN Classifier gives best results overall.



```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      2114
           1       0.99      0.98      0.98      2203

    accuracy                           0.98      4317
   macro avg       0.98      0.98      0.98      4317
weighted avg       0.98      0.98      0.98      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f8c149d8b50>
```

Fig. 32 - Confusion Matrix for KNN algorithm using New Dataset

For Decision Tree, the precision for 0 is 95% and recall is 100 and the map shows the number of false positives have reduced significantly.



```
              precision    recall  f1-score   support

           0       0.95      1.00      0.98      2040
           1       1.00      0.95      0.98      2277

    accuracy                           0.98      4317
   macro avg       0.98      0.98      0.98      4317
weighted avg       0.98      0.98      0.98      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f8c148fa3d0>
```

Fig. 33 - Confusion Matrix for Decision Tree algorithm using new dataset

The precision for 0 being 92% and recall being 100%. The map also shows the number of false positives is very low and true positives are very high.

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      1947
           1       1.00      0.91      0.95      2370

    accuracy                           0.95      4317
   macro avg       0.95      0.95      0.95      4317
weighted avg       0.95      0.95      0.95      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f8c14669f50>
```

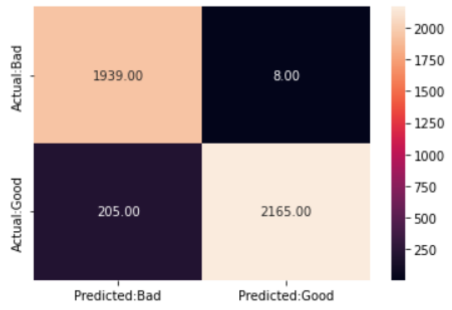


Fig. 34 - Confusion Matrix for SVM algorithm using new dataset

Random Forest algorithm is very accurate with a precision for 0 as 96% and recall of 99. The map also shows the number of false positives are very low but slightly higher than that of Decision tree and true positives are very high.

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      2059
           1       0.99      0.96      0.97      2258

    accuracy                           0.97      4317
   macro avg       0.97      0.98      0.97      4317
weighted avg       0.98      0.97      0.97      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f8c147cee90>
```

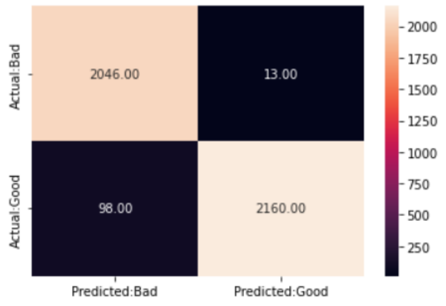
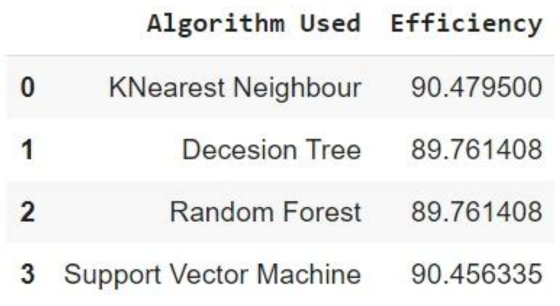


Fig. 35 - Confusion Matrix for Random Forest algorithm using new dataset

The accuracy calculated from each of the algorithms are tabulated in the image given below. From this the conclusion is drawn that the maximum accuracy of 98% is for the KNN Classifier algorithm. Random forest and Decision tree have almost the same accuracy where random forest is 97.42% accurate and Decision Tree is 97.5% accurate. Support Vector Machine is the lowest with an accuracy of 95%.



	Algorithm Used	Efficiency
0	KNearest Neighbour	90.479500
1	Decesion Tree	89.761408
2	Random Forest	89.761408
3	Support Vector Machine	90.456335

Fig. 36 - Accuracy comparison for ML algorithms using new dataset

So, from this we see that for the dataset we created the accuracies of KNN as well as SVM have increased and also the accuracies of both Decision Tree and Random Forest have increased in such a way that they are now very close to the accuracies obtained in [1].

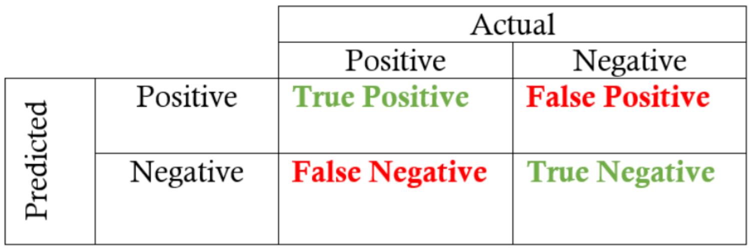
Results of Phase 3

True Positive (TP) model correctly predicts the positive class (prediction and actual both are positive).

True Negative (TN) model correctly predicts the negative class (prediction and actual both are negative).

False Positive (FP) model gives the wrong prediction of the negative class (predicted-positive, actual-negative).

FP is also called a. TYPE I error. False Negative (FN) model wrongly predicts the positive class (predicted-negative, actual-positive). FN is also called a TYPE II error.



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Fig. 37 - Confusion matrix

Classification metrics used are as follows:

Accuracy:

Accuracy is the proportion of true results among the total number of cases examined.

Accuracy = (TP+TN)/(TP+FP+FN+TN)

Accuracy is a valid choice of evaluation for classification problems which are well balanced and not skewed or No class imbalance.

Precision:

Precision, answers the question: what proportion of predicted Positives is truly Positive?

Precision = (TP)/(TP+FP)

Precision is a valid choice of evaluation metric when we want to be very sure of our prediction.

Recall

Recall which answers a different question: what proportion of **actual Positives** is correctly classified?

Recall = (TP)/(TP+FN)

Recall is a valid choice of evaluation metric when we want to capture as many positives as possible.

F1 Score:

The F1 score is a number between 0 and 1 and is the harmonic mean of precision and recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

We use it when we want to have a model with both good precision and recall. *F1 score sort of maintains a balance between the precision and recall for your classifier*. If your precision is low, the F1 is low and if the recall is low again your F1 score is low.

AUC-ROC Curve:

AUC is the area under the ROC curve. ROC AUC is a performance metric for binary classification problems. AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes. The AUC represents a model's ability to discriminate between positive and negative classes. An area of 1.0 represents a model that made all predictions perfectly. An area of 0.5 represents a model as good as random.

A ROC Curve is a plot of the true positive rate and the false positive rate for a given set of probability predictions at different thresholds used to map the probabilities to class labels. The area under the curve is then the approximate integral under the ROC Curve.

We have got the probabilities from our classifier. We can use various threshold values to plot our sensitivity (TPR) and (1-specificity) (FPR) on the cure and we will have a ROC curve.

Where True positive rate or TPR is just the proportion of trues, we are capturing using our algorithm.

Sensitivity = TPR (True Positive Rate) = Recall = TP/(TP+FN)

and False positive rate or FPR is just the proportion of false we are capturing using our algorithm.

1- Specificity = FPR (False Positive Rate) = FP/(TN+FP)

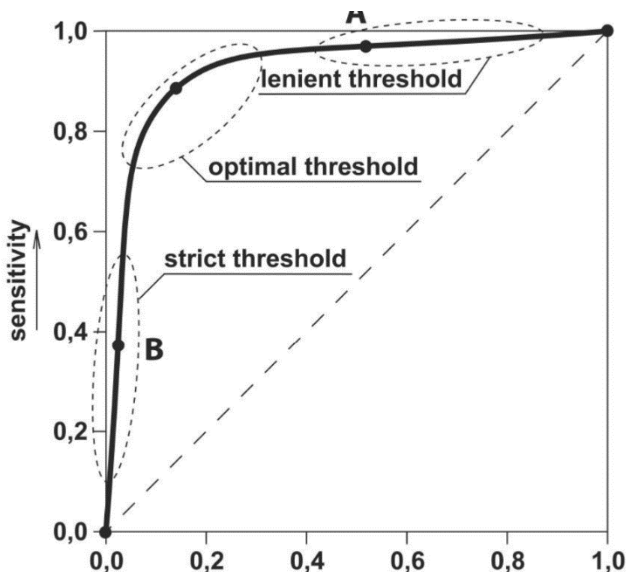


Fig. 38 – Threshold values in ROC

We can use the ROC curves to decide on a Threshold value. The choice of threshold value will also depend on how the classifier is intended to be used. AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values. Another benefit of using AUC is that it is classification-threshold-invariant like log loss. It measures the quality of the model's predictions irrespective of what classification threshold is chosen, unlike F1 score or accuracy which depend on the choice of threshold.

Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.

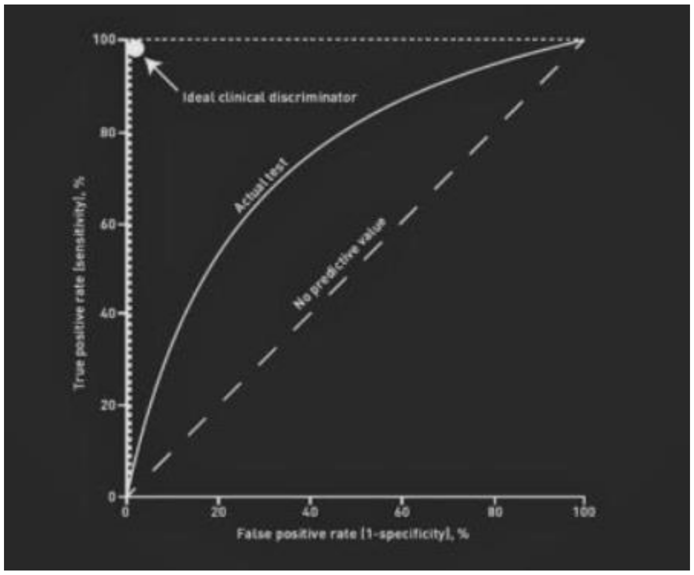


Fig. 39 – ROC curve

I) ANN

The following are the results for ANN:

```
              precision    recall  f1-score   support

           0       0.87      0.93      0.90      2004
           1       0.94      0.88      0.91      2313

    accuracy                           0.90      4317
   macro avg       0.90      0.91      0.90      4317
weighted avg       0.91      0.90      0.90      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f0ad3b4d310>
```

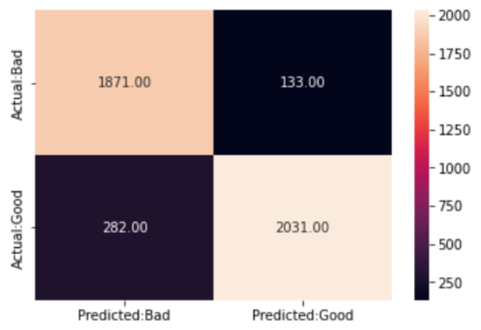


Fig. 40 - Confusion matrix for ANN

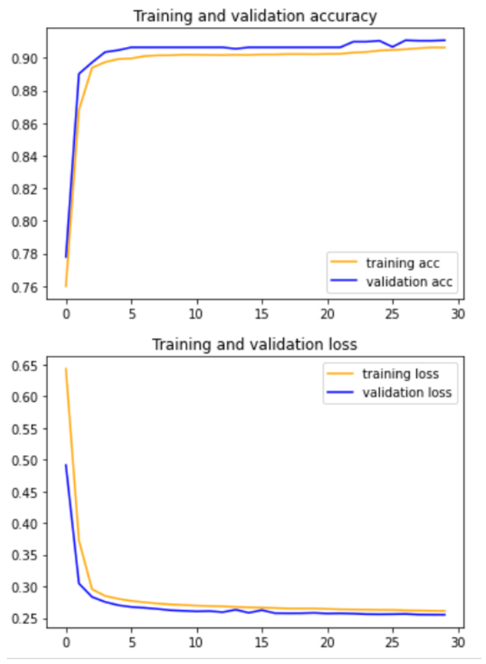


Fig. 41 - Training and Validation accuracy for ANN

```
AUC for our classifier is: 93.4116025182589
```

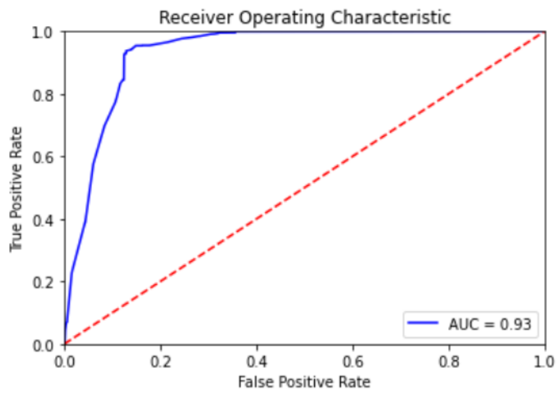


Fig. 42 - ROC curve for ANN

ANN algorithm is accurate with a precision for 0 as 87% and recall of 93. The map also shows the number of false positives are very low but slightly higher than some algorithms, and true positives are high. We can see from the graphs that the training and validation accuracy are almost equal and the training and validation loss are also almost equal. The ROC curve tells us that our model is nearly perfect classifier, with high accuracy.

II) CNN

The following are the results for CNN:

```
              precision    recall  f1-score   support

           0       0.86      0.94      0.90      1976
           1       0.95      0.87      0.91      2341

    accuracy                           0.90      4317
   macro avg       0.90      0.91      0.90      4317
weighted avg       0.91      0.90      0.90      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f0991b5bb10>
```

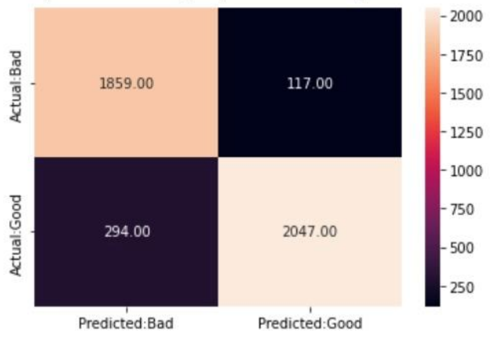


Fig. 43 - Confusion matrix for CNN

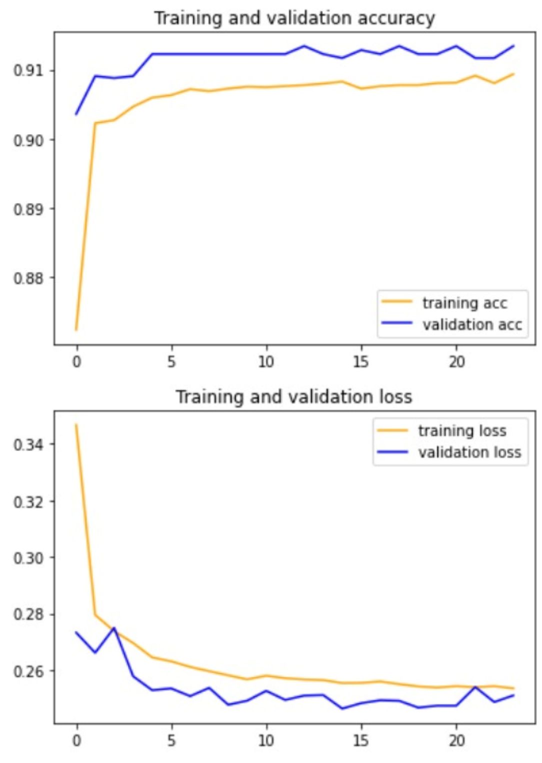


Fig. 44 - Training and Validation accuracy for CNN

```
AUC for our classifier is: 93.63605612424053
```

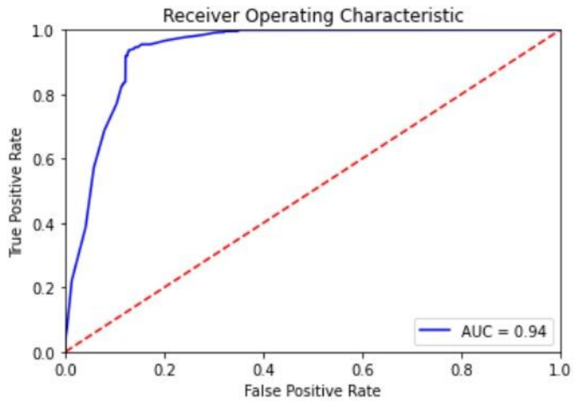
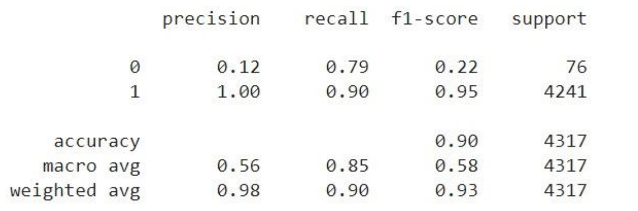


Fig. 45 - ROC Curve for CNN

CNN algorithm is accurate with a precision for 0 as 86% and recall of 94. The map also shows the number of false positives are lower than ANN but slightly higher than some algorithms, and true positives are high. We can see from the graphs that the training and validation accuracy are almost equal and the training and validation loss are also almost equal but not as much as ANN. The ROC curve tells us that our model is nearly perfect classifier, with high accuracy.

III) CNN + LSTM

The following are the results for CNN + LSTM:



```
              precision    recall  f1-score   support

           0       0.12      0.79      0.22        76
           1       1.00      0.90      0.95      4241

    accuracy                           0.90      4317
   macro avg       0.56      0.85      0.58      4317
weighted avg       0.98      0.90      0.93      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f03485bd290>
```

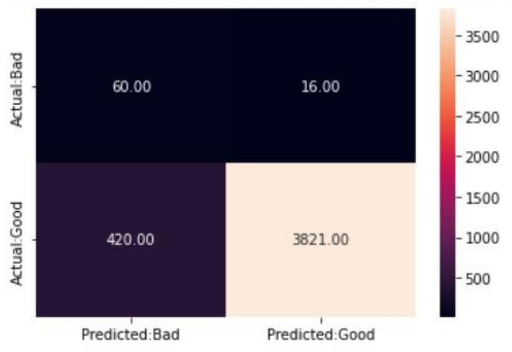


Fig. 46 - Confusion matrix for CNN + LSTM

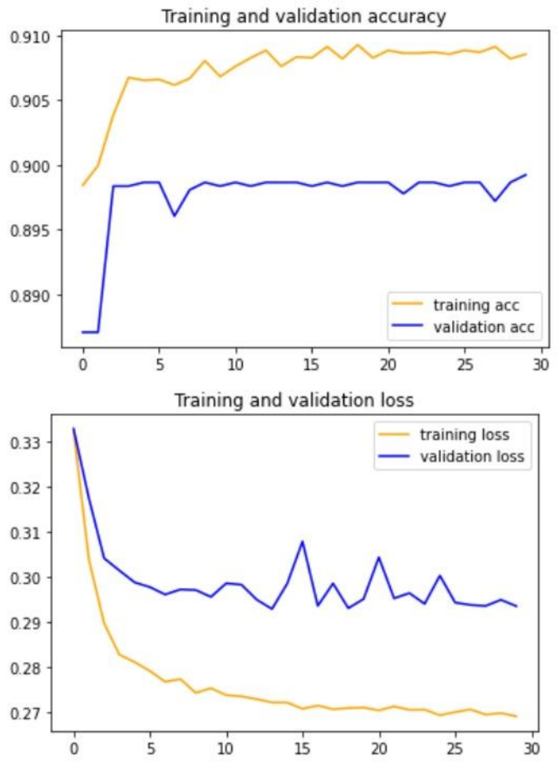


Fig. 47 - Training and Validation accuracy for CNN + LSTM

```
AUC for our classifier is: 76.15283750325776
```

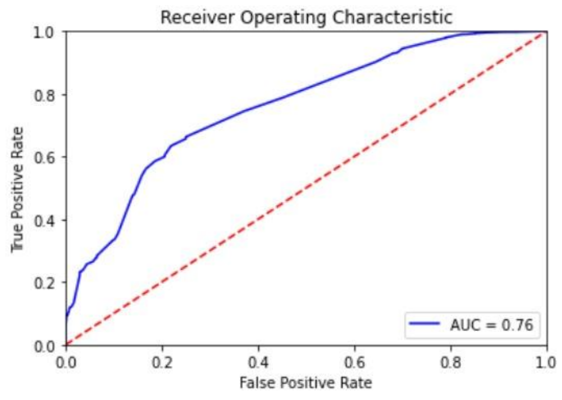


Fig. 48 - ROC Curve for CNN + LSTM

CNN + LSTM algorithm is accurate with a precision for 1 as 100% and recall of 90. The map also shows the number of false positives are very low but slightly higher than some algorithms, and true positives are high. We can see from the graphs that the training and validation accuracy are not equal and the training and validation loss are also not equal. The ROC curve tells us that our model is a less perfect classifier, with low accuracy.

IV) RNN + LSTM

The following are the results for RNN + LSTM:

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.90      1982
           1       0.95      0.88      0.91      2335

    accuracy                           0.91      4317
   macro avg       0.91      0.91      0.91      4317
weighted avg       0.91      0.91      0.91      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f4ed7830e10>
```

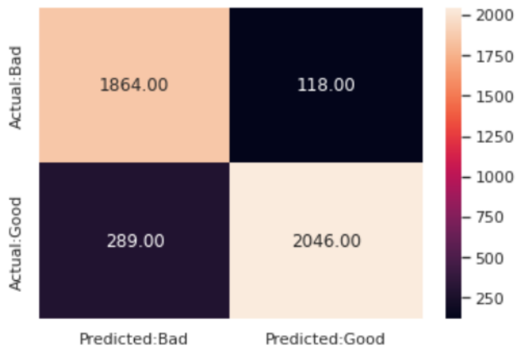


Fig. 49 - Confusion matrix for RNN + LSTM

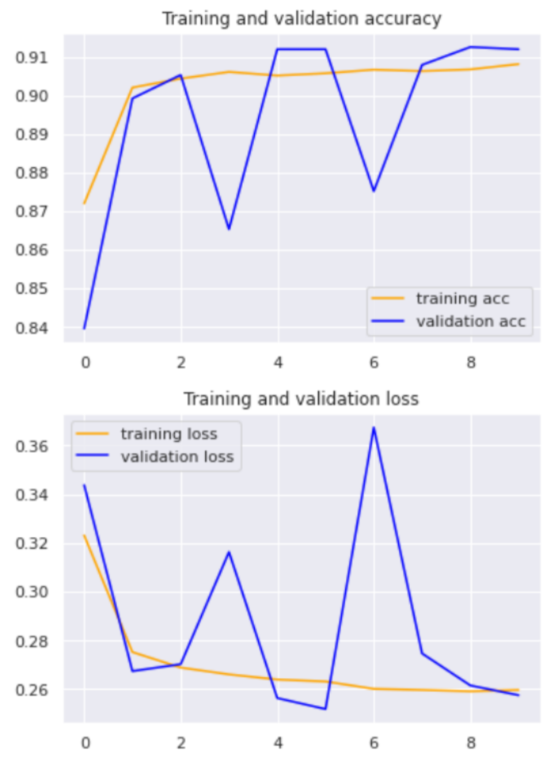


Fig. 50 - Training and Validation accuracy of RNN

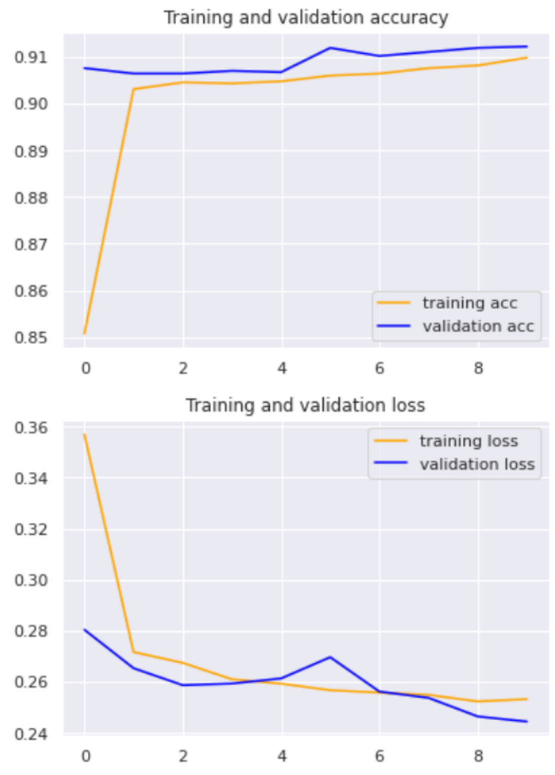


Fig. 51 - Training and Validation accuracy of LSTM

```
AUC for our classifier is: 94.55880029842724
```

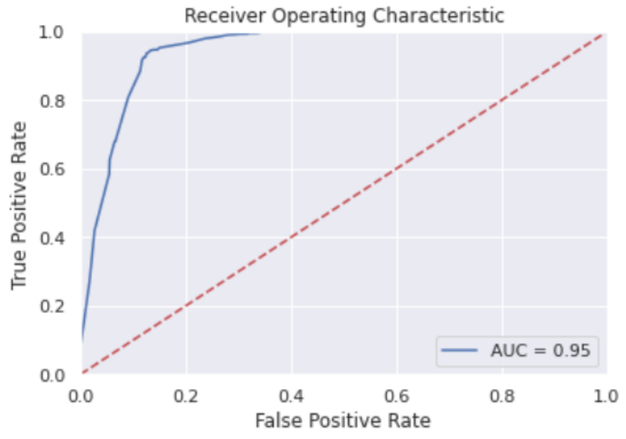


Fig. 52 - ROC curve for RNN + LSTM

RNN + LSTM algorithm is accurate with a precision for 1 as 95% and recall of 88. The map also shows the number of false positives are very low and true positives are high. We can see from the graphs that the training and validation accuracy for both algorithms are almost equal and the training and validation loss are also almost equal. The ROC curve tells us that our model is nearly perfect classifier, with high accuracy.

V) Bidirectional LSTM

The following are the results for Bidirectional LSTM:

```
              precision    recall  f1-score   support

           0       0.86      0.94      0.90      1976
           1       0.94      0.87      0.91      2341

    accuracy                           0.90      4317
   macro avg       0.90      0.90      0.90      4317
weighted avg       0.90      0.90      0.90      4317

<matplotlib.axes._subplots.AxesSubplot at 0x7f0edd855190>
```

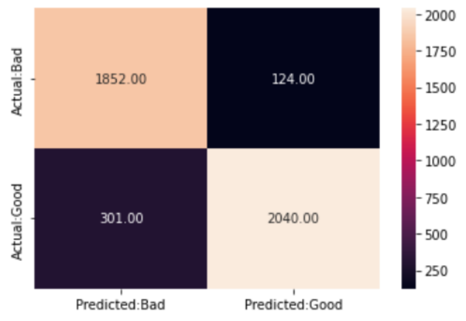


Fig. 53 - Confusion matrix for Bidirectional LSTM

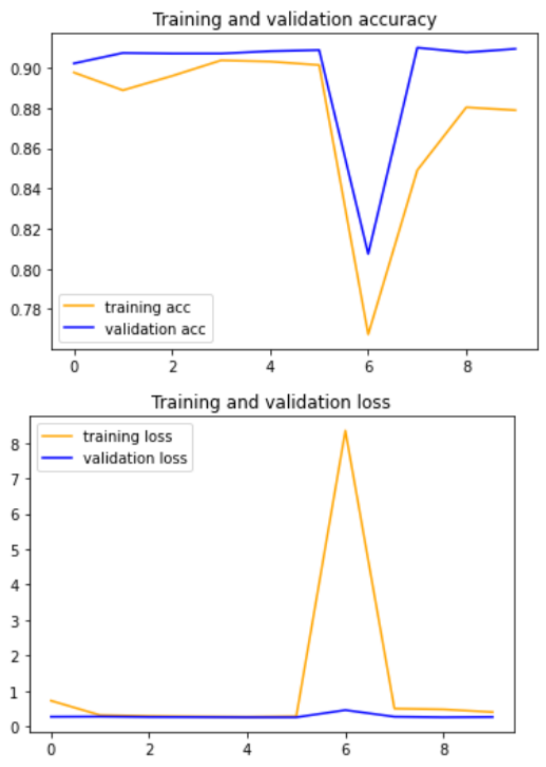
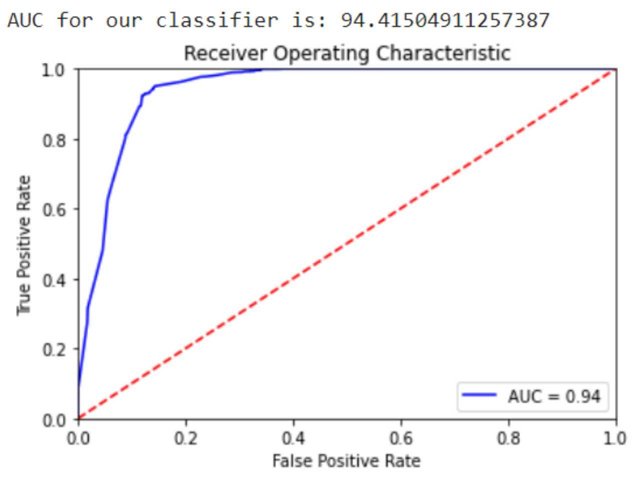


Fig. 54 - Training and Validation accuracy for Bidirectional LSTM

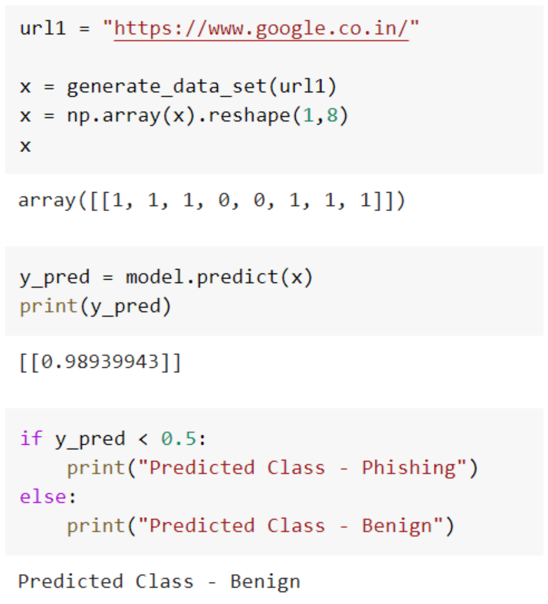


```
AUC for our classifier is: 94.41504911257387
```

Fig. 55 - ROC Curve for Bidirectional LSTM

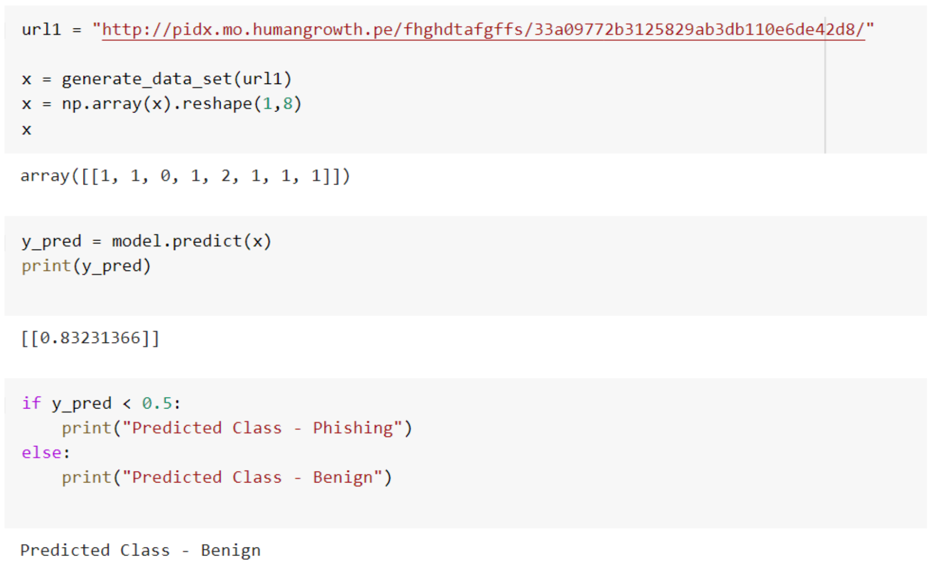
Bidirectional LSTM algorithm is highly accurate with a precision for 1 as 94% and recall of 87. The map also shows the number of false positives are very low but slightly higher than some algorithms, and true positives are high. We can see from the graphs that the training and validation accuracy are almost equal and the training and validation loss are also almost equal except for a peak around 6. The ROC curve tells us that our model is nearly perfect classifier, with high accuracy.

The below images show how the parameters for a website are calculated, the accuracy of the prediction and the final prediction of the website for both CNN + LSTM algorithm and Bidirectional LSTM algorithm. The predictions given by both algorithms are correct.



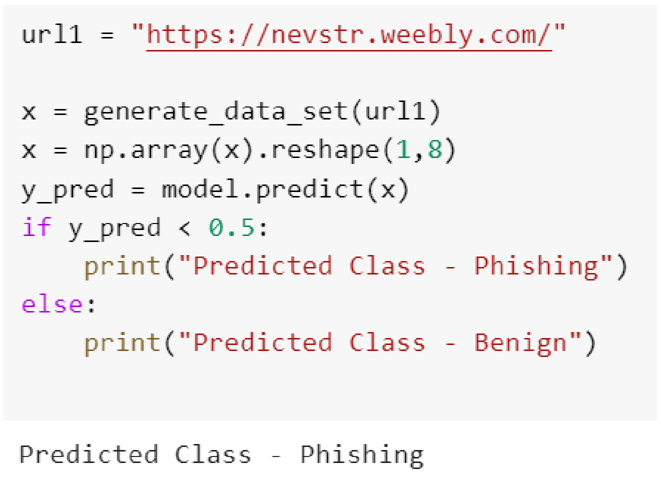
```
url1 = "https://www.google.co.in/"

x = generate_data_set(url1)
x = np.array(x).reshape(1,8)
x
```
```
array([[1, 1, 1, 0, 0, 1, 1, 1]])
```
```
y_pred = model.predict(x)
print(y_pred)
```
```
[[0.98939943]]
```
```
if y_pred < 0.5:
    print("Predicted Class - Phishing")
else:
    print("Predicted Class - Benign")
```
```
Predicted Class - Benign
```

Fig. 56 - Prediction using CNN + LSTM model



```
url1 = "http://pidx.mo.humangrowth.pe/fhghdtafgffs/33a09772b3125829ab3db110e6de42d8/"

x = generate_data_set(url1)
x = np.array(x).reshape(1,8)
x
```
```
array([[1, 1, 0, 1, 2, 1, 1, 1]])
```
```
y_pred = model.predict(x)
print(y_pred)
```
```
[[0.83231366]]
```
```
if y_pred < 0.5:
    print("Predicted Class - Phishing")
else:
    print("Predicted Class - Benign")
```
```
Predicted Class - Benign
```

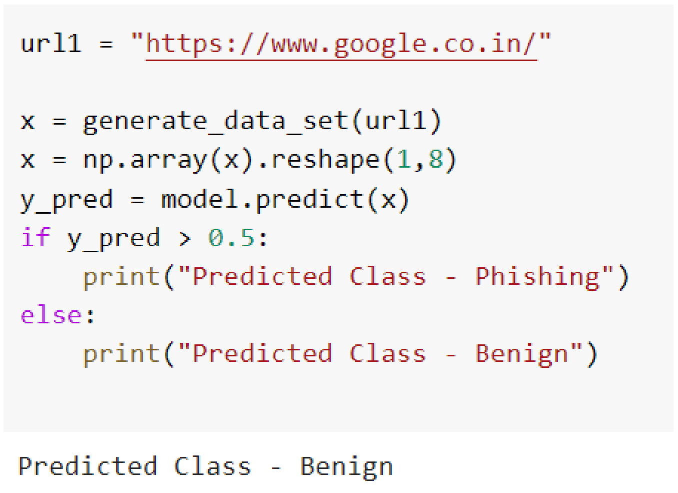
Fig. 57 - Prediction for website 1 using Bidirectional LSTM model



```
url1 = "https://nevstr.weebly.com/"

x = generate_data_set(url1)
x = np.array(x).reshape(1,8)
y_pred = model.predict(x)
if y_pred < 0.5:
    print("Predicted Class - Phishing")
else:
    print("Predicted Class - Benign")
```

```
Predicted Class - Phishing
```

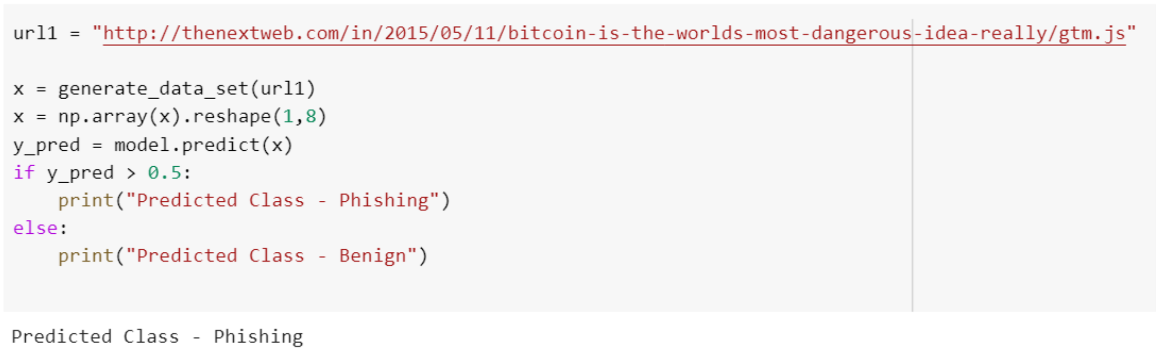
Fig. 58 - Prediction for website 2 using Bidirectional LSTM model



```
url1 = "https://www.google.co.in/"

x = generate_data_set(url1)
x = np.array(x).reshape(1,8)
y_pred = model.predict(x)
if y_pred > 0.5:
    print("Predicted Class - Phishing")
else:
    print("Predicted Class - Benign")
```

```
Predicted Class - Benign
```

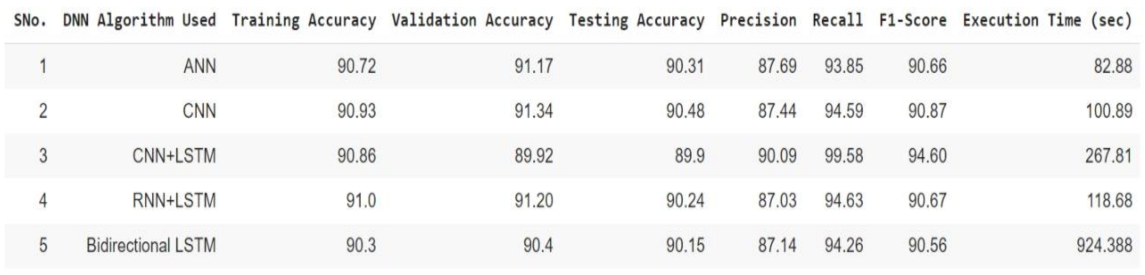
Fig. 59 - Prediction for website 3 using Bidirectional LSTM model



```
url1 = "http://thenextweb.com/in/2015/05/11/bitcoin-is-the-worlds-most-dangerous-idea-really/gtm.js"

x = generate_data_set(url1)
x = np.array(x).reshape(1,8)
y_pred = model.predict(x)
if y_pred > 0.5:
    print("Predicted Class - Phishing")
else:
    print("Predicted Class - Benign")
```

```
Predicted Class - Phishing
```

Fig. 60 - Prediction for website 4 using Bidirectional LSTM model



SNo.	DNN Algorithm Used	Training Accuracy	Validation Accuracy	Testing Accuracy	Precision	Recall	F1-Score	Execution Time (sec)
1	ANN	90.72	91.17	90.31	87.69	93.85	90.66	82.88
2	CNN	90.93	91.34	90.48	87.44	94.59	90.87	100.89
3	CNN+LSTM	90.86	89.92	89.9	90.09	99.58	94.60	267.81
4	RNN+LSTM	91.0	91.20	90.24	87.03	94.63	90.67	118.68
5	Bidirectional LSTM	90.3	90.4	90.15	87.14	94.26	90.56	924.388

Fig. 61 - Comparison between all algorithms

Results for Phase 4

In this phase we created the web interface for the phishing detection which would give the prediction for a website URL. The images given blow show some examples for legitimate and phishing site predictions. The algorithm used for the final predictions is Bidirectional LSTM.

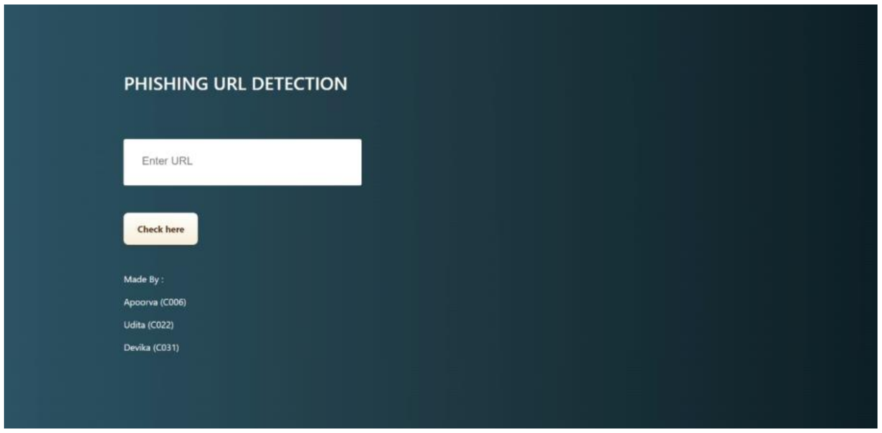
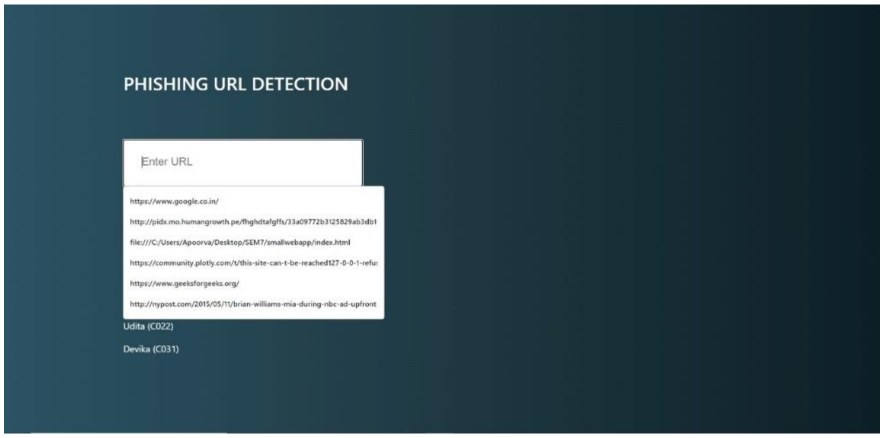
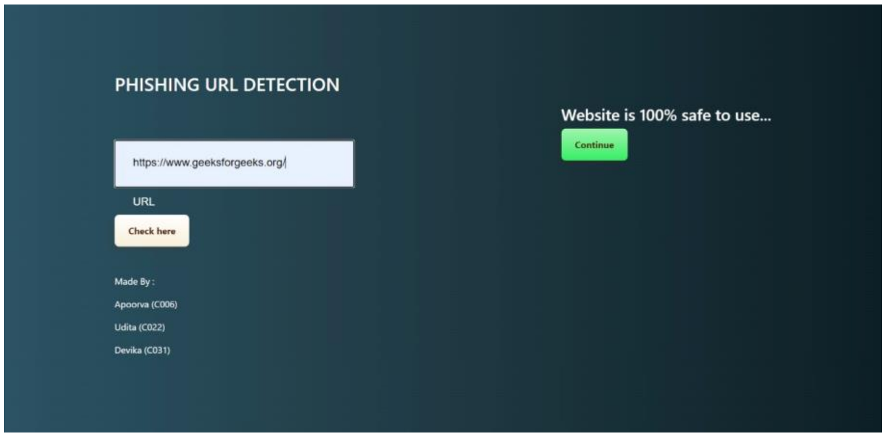


Fig. 62 - Phishing Detection Website



https://www.google.co.in/
http://pidx.mo.humangrowth.pe/fhghdtafgffs/33a09772b3125829ab3db1
file:///C:/Users/Apoorva/Desktop/SEM7/smallwebapp/index.html
https://community.plotly.com/t/this-site-can-t-be-reached127-0-0-1-refu
https://www.geeksforgeeks.org/
http://nypost.com/2015/05/11/brian-williams-mia-during-nbc-ad-upfront

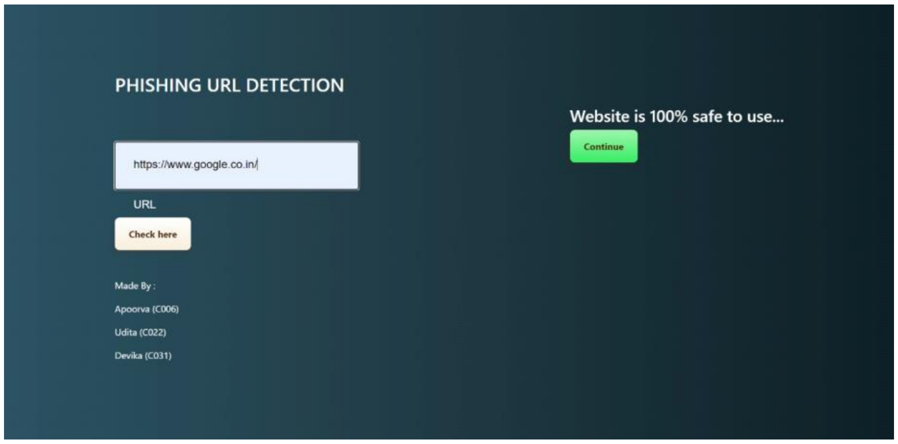
Fig. 63 - Feeding the site into the detector



https://www.geeksforgeeks.org/

Website is 100% safe to use...

Fig. 64 - Legitimate site prediction for GeeksforGeeks



https://www.google.co.in/

Website is 100% safe to use...

Fig. 65 - Legitimate site prediction for Google

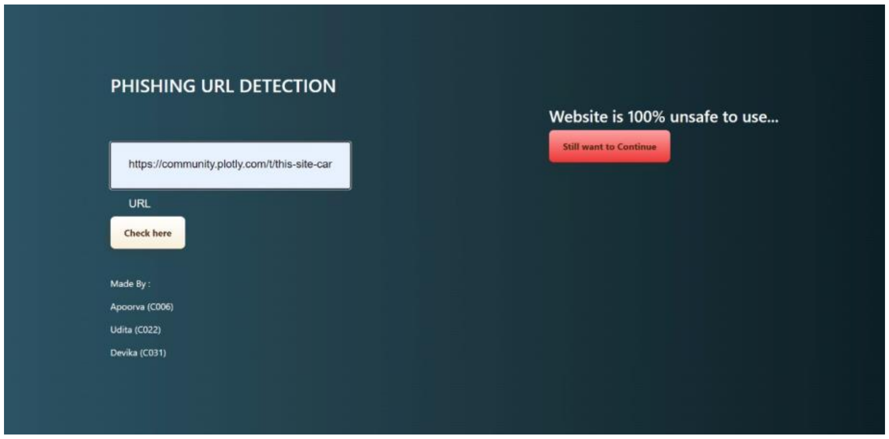


Fig. 66 - Phishing site prediction for Community plotly

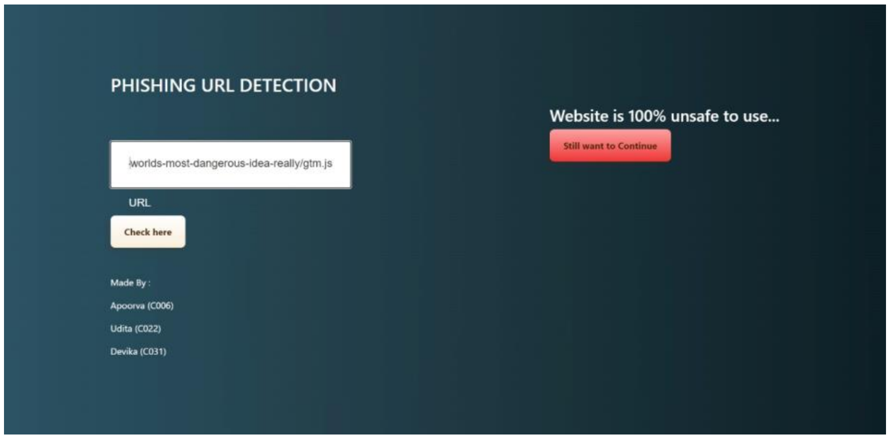


Fig. 67 - Phishing site prediction for TheNextWeb

Chapter 5

Advantages, Limitations and Applications

5.1 Advantages

- It helps to reduce the load on antivirus software's and also is a good substitute for them in case the anti-virus is absent in our systems
- 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.
- 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. Automated advanced phishing threat protection defends against todays and tomorrow's threats with a system that continually learns and makes the entire organization more security conscious and aware.
- An increasing number of services, including banking and social networking, are being integrated with world wide web in recent years. The crux of this increasing dependence on the internet is the rise of different kinds of cyberattacks on unsuspecting users such as phishing, which aims at stealing user information via deceptive websites. The primary defence against phishing consists of maintaining a black list of the phishing URLs like are used in Anti-virus software's. However, a black list approach is reactive and cannot defend against new phishing websites. For this reason, using machine learning or deep learning techniques to detect previously unseen phishing URLs could be of immense help to large scale businesses.

5.2 Limitations

- It is time consuming because the models are trained on the dataset for large amount of time.
- We have to extract huge number of features for efficient detection.
- High memory requirement
- We have not built a secure web browser that protects the user while surfing through the web by providing better security against phishing attacks in real time.
- The underlying assumption is that a malicious URL has a significantly different feature distribution than a benign URL. As a result, a good deep learning approach should be able to differentiate between benign and malicious URLs based on those features. However, in situations where this condition is not met the detector will fail to make proper predictions.
- From our evaluation dataset it is clear that our test and train domains are the English language based so this may limit the performance of our system when the large percentage of the testing dataset is non-English.
- If a phisher mimics a webpage using sophisticated tools, then the look and feel of such website may be a replica of the legitimate page. In this case, the use of visual attributes may not produce significant positive results.

5.3 Applications

We can build a secure web browser with all new browser architecture. This protects the user while surfing through the web by providing better security against phishing attacks in real time. The proposed prototype has got a new outlook which provides a wider view area for the webpage and UI. Apart from this, it is important to think about security while we are in the internet world, so the browser with an intelligent engine protects the user from being hacked by phishing websites. The most interesting thing is that this engine will protect us from an attacker in real-time. The prototype provides a fast, reliable, and secure browsing experience for the users. As of now, the prototype lets the users gain the advantages of the security as well as basic features of the browser. In the future, the project can be helpful in various aspects of a normal person.

The same approach as used in our project can be applied to detect phishing attacks in real-time if the models that we have created are deployed on to our browser such that the detector is always running in the background to notify us if we are visiting a phishing website during our internet surf. While this may sound like an anti-virus running in the background of our system, this Deep learning approach allows the models to learn from their experiences instead of just abiding to certain rules set by an organisation. So, whenever a new kind of phishing website is encountered the model includes as a part of its learning to be able to identify something like it in the future [13].

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

Phishing is a type of social engineering attack often used to steal user data. It occurs when an attacker disguised as a trusted entity, dupes a victim into opening an email, instant message, or text message. We propose to develop a model to detect the phishing attacks. We created a custom dataset which consists of 21,581 websites collected from Phishtank for phishing sites and Alexa for legitimate sites. By performing feature extraction, we narrowed down the important features for our project which would help give us the most accurate prediction. We did a comparison based on accuracy and confusion matrices on 4 machine learning algorithms– KNN, Decision Tree, Random Forest and SVM out of which KNN showed the best efficiency on our dataset.

We compared the performance of 5 Deep learning (DL) algorithms on our dataset - ANN, CNN, RNN+LSTM, CNN + LSTM and Bidirectional LSTM. Based on multiple parameters like number of epochs used, precision, training and validation accuracy achieved, the model built using Bidirectional algorithm was the most efficient on our dataset. Using this algorithm, we built our project and performed prediction on our dataset. We also built a website using Flask for easy user navigation. On inputting a website URL, the website will output whether the URL is of a phishing website or a legitimate website.

6.2 Future Scope

- Today, phishing attacks are targeted, difficult to detect, grant malicious individuals broad permissions over user data and devices
- More extensive research with wider range of libraries
- Could use specifically hybrid libraries
- Use more structured dataset
- Design neural networks based on specific needs
- More thorough feature selection could be conducted

- More features like DNS query results, black list presence, bag of words, web page content, web page network traffic etc could be extracted from the URLs to train the model and achieve more accuracy [13].
- Work could also be done to collect more URLs to decrease the variance of the probability scores.
- We can also use on the selective sampling algorithm and unbalanced datasets so as to take into account time varying features of the URLs [13].

References

[1] M. N. Alam, D. Sarma, F. F. Lima, I. Saha, R. -E. -. Ulfath and S. Hossain, "Phishing Attacks Detection using Machine Learning Approach," *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT),* 2020, pp. 1173-1179

[2] A. Odeh, I. Keshta and E. Abdelfattah, "Machine Learning Techniques for Detection of Website Phishing: A Review for Promises and Challenges,” *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC),* 2021

[3] M. A. Adebowale, K. T. Lwin and M. A. Hossain, "Deep Learning with Convolutional Neural Network and Long Short-Term Memory for Phishing Detection,” *2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA),* pp. 1-8, 2019

[4] S. Singh, M. P. Singh and R. Pandey, "Phishing Detection from URLs Using Deep Learning Approach," *2020 5th International Conference on Computing, Communication and Security (ICCCS),* pp. 1-4, 2020

[5] Y. Su, "Research on Website Phishing Detection Based on LSTM RNN," *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC),* vol. 1, pp. 284-288, 2020

[6] P. Yang, G. Zhao and P. Zeng, "Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," *IEEE Access, vol. 7,* pp. 15196-15209, 2019

[7] M. N. Feroz and S. Mengel, "Phishing URL Detection Using URL Ranking," *2015 IEEE International Congress on Big Data, 2015,* pp. 635-638

[8] K. Mridha, J. Hasan, S. D and A. Ghosh, "Phishing URL Classification Analysis Using ANN Algorithm," *2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON)*, 2021, pp. 1-7, doi: 10.1109/GUCON50781.2021.9573797.

[9] Y. Luan and S. Lin, "Research on Text Classification Based on CNN and LSTM," *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2019, pp. 352-355, doi: 10.1109/ICAICA.2019.8873454.

[10] H. Hu, M. Liao, C. Zhang and Y. Jing, "Text classification based recurrent neural network," *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 652-655, doi: 10.1109/ITOEC49072.2020.9141747.

[11] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.

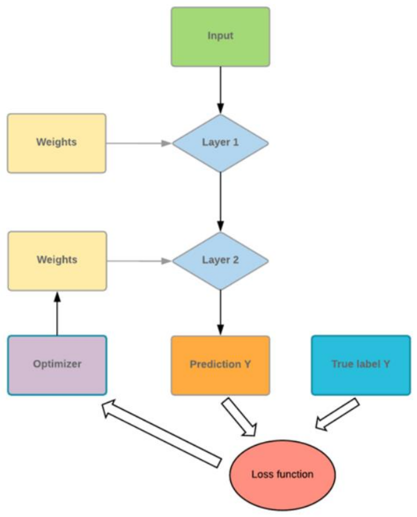
[12] M. Aydin and N. Baykal, "Feature extraction and classification phishing websites based on URL," *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015, pp. 769-770, doi: 10.1109/CNS.2015.7346927.

[13] M. Sameen, K. Han and S. O. Hwang, "PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System," in *IEEE Access*, vol. 8, pp. 83425-83443, 2020, doi: 10.1109/ACCESS.2020.2991403.

[14] A. Das, A. Das, A. Datta, S. Si and S. Barman, "Deep Approaches on Malicious URL Classification," *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1-6, doi: 10.1109/ICCCNT49239.2020.9225338.

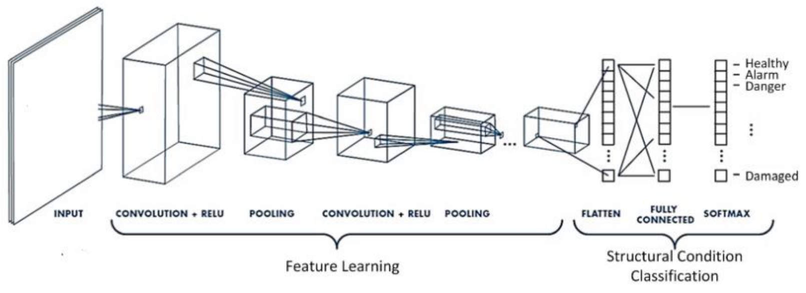
Appendix-I

I.1 ANN Block Diagram

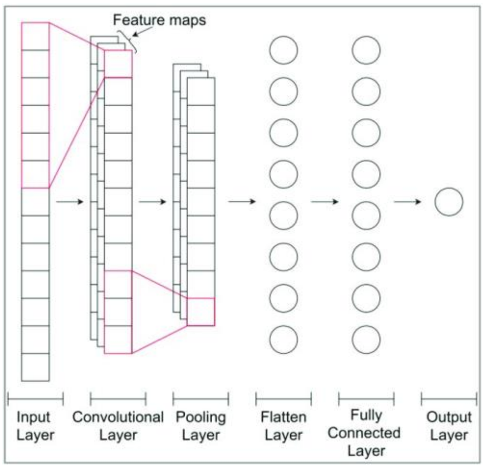


ANN

I.2 CNN Block Diagram

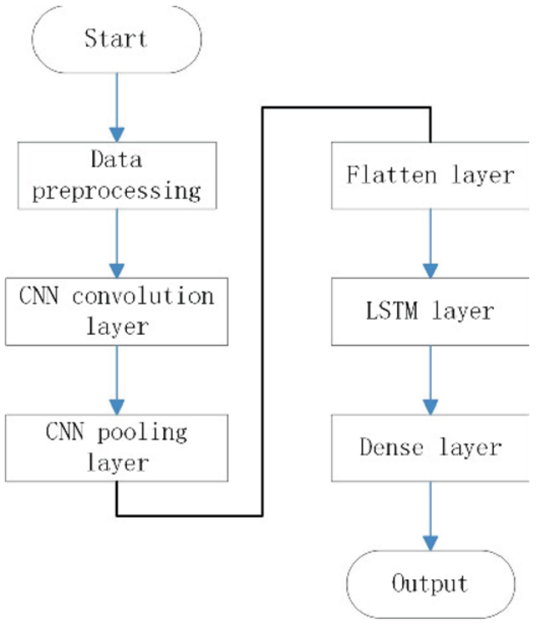


Conv-2D



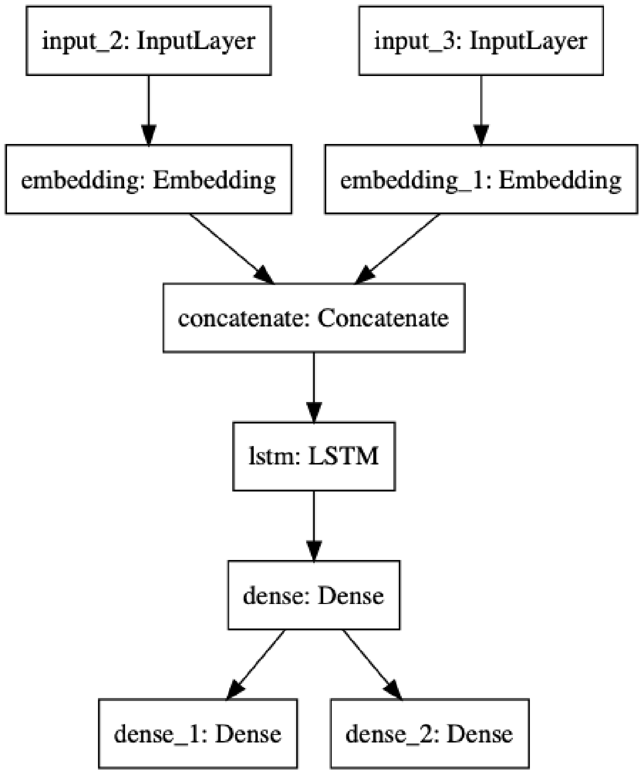
Conv-1D

I.3 CNN+LSTM Block Diagram



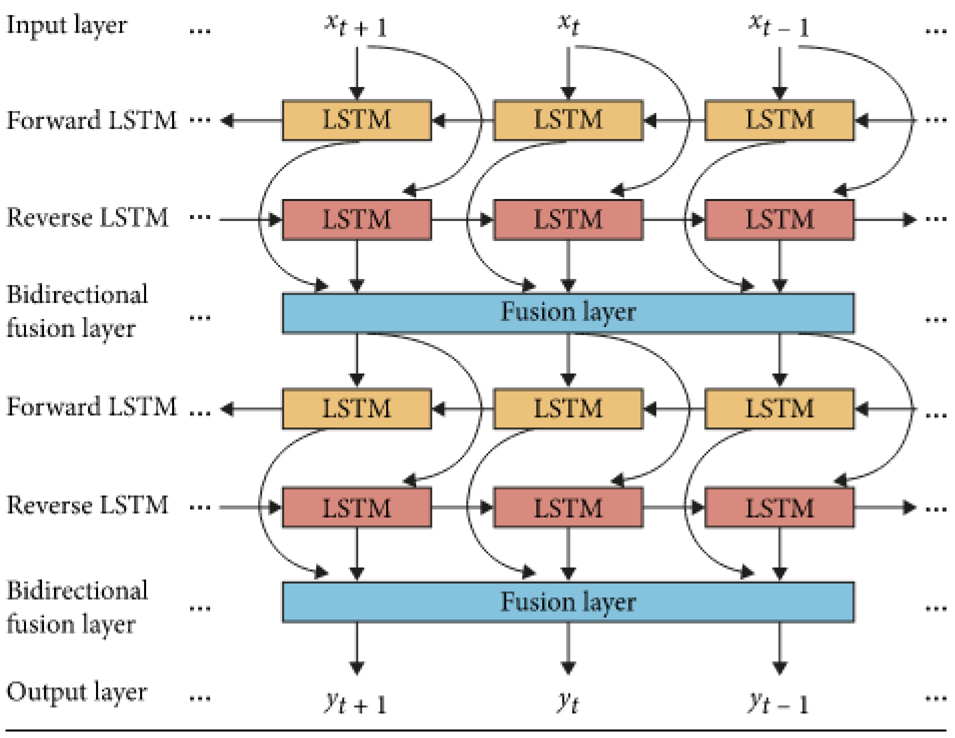
CNN+LSTM

I.4 RNN+LSTM Block Diagram



RNN+LSTM

I.5 Bi-Directional LSTM Block Diagram



Bi-Directional LSTM