



## DETECTION of PHISHING WEBSITES USING MACHINE LEARNING and DEEP LEARNING TECHNIQUES

**Batchu suvidha** DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS  
ENGINEERING, ANDHRA UNIVERSITY, COLLEGE OF ENGINEERING  
VISAKHAPATNAM-530 003.

**Dr.G.Lavanaya Devi** Assistant Professor in DEPARTMENT OF COMPUTER SCIENCE  
AND SYSTEMS ENGINEERING, ANDHRA UNIVERSITY, COLLEGE OF ENGINEERING  
VISAKHAPATNAM-530 003.

### ABSTRACT

The Internet is changing how people study and work while also exposing us to increasingly serious security threats as a result of the increasing integration of social life and the Internet. One of the most important issues that needs to be investigated right away is how to identify different network threats, particularly those that have never been seen before. Phishing site URLs are designed to gather sensitive data such as user identities, passwords, and transactions involving online money. There are a number of ways to recognize phishing websites.

The use of machine learning techniques to identify phishing websites was previously suggested and put into practice. The implementation of the system with high accuracy, efficiency, and cost-effectiveness is the main goal of this project. This has been accomplished. Four machine learning supervised classification models are used to carry out the job. K-Nearest Neighbor, Kernel Support Vector Machine, Decision Tree, and Random Forest Classifier are the four classification models

### 1 INTRODUCTION

The goal of cyber security is to protect computers, networks, projects, and data from attacks and unauthorized access, modification, or destruction. A system assurance framework and a PC protection framework are both parts of a system security framework. These frameworks all include intrusion detection systems (IDS), antivirus software, and firewalls. IDSs aid in the discovery, determination, and differentiation of improper system behavior, such as usage, replication, change, and annihilation

Artificial intelligence is a brand-new, cutting-edge science that examines and develops theories, plans, methods, and applications that replicate, develop, and enhance human knowledge. ML is a branch of artificial intelligence that focuses on making predictions using computers. It is comparable to (and frequently overlaps with) computational



measurements. Scientific advancement, which communicates methodologies, hypotheses, and usage regions to the field, has a strong association with machine learning. Data mining and machine learning are occasionally mixed, although unsupervised learning is a branch of data mining that focuses primarily on exploratory information analysis. The goal of cyber security is to protect computers, networks, projects, and information from attacks and unauthorized access, modification, or destruction.

Framework in addition to a system assurance framework. These frameworks all include intrusion detection systems (IDS),

Antivirus software, and firewalls. IDSs aid in the discovery, determination, and differentiation of improper system behavior, such as usage, replication, change, and annihilation.

The following three types of network analysis are crucial for intrusion detection systems: Misuse-based, also called anomaly signature, and hybrid-based.

By utilizing the hallmarks of these attacks, misuse-based detection techniques aim to discriminate between realized attacks.

- Anomaly-based methods examine the usual system and its behavior in order to identify anomalies as deviations from the norm.
- Anomaly and misuse detection are combined in hybrid detection. It is used to increase the detection rate of authorized intrusions and lower the number of false positives for unidentified attacks.

Machine Learning (ML) techniques are being used more frequently than ever in cybersecurity. Machine learning is one of the greatest solutions that can influence against zero-day attacks, starting with the categorization of IP traffic and separating harmful traffic for intrusion detection. Utilizing measurable traffic features and ML approaches, new exploration is being done. In 1987, the term "phishing" was first used. Phishing is an online theft that steals a person's identity and personal information. It is a form of extortion where the attacker has full access to the victim's personal information

## **2. LITERATURE SURVEY AND RELATED WORK**

The phase of the software development process that is most crucial is the literature review. The time factor, economics, and company strength must all be assessed before the tool is



developed. Once these requirements have been met, the following step is to choose the operating system and language that can be utilized to construct the tool. Once the programmers begin creating the tool, they require a lot of outside assistance. Senior programmers, books, and websites are good sources of this support. The aforementioned factors are taken into account in developing the suggested system before beginning construction. A well-written piece that covers the current knowledge, including important findings as well as theoretical and methodological commitments to a particular issue, is known as a literature overview.

A model for phishing website identification that is based on neural networks and focuses on the best feature selection method is proposed. An index termed feature validity value (FVV) has been created in this suggested model to evaluate the influence of all of those factors on the identification of such websites. An algorithm is now designed to locate the best phishing websites based on this newly created index features. The issue of the neural network's over-fitting will be greatly reduced by the chosen algorithm. Although the accuracy of the model will mostly depend on the knowledge of the features, feature engineering plays a crucial role in identifying solutions for the identification of phishing web sites. The constraint is in the amount of time it takes to gather these features, even though the features obtained from all of these different dimensions are understandable. A three-phase detection system called Web Crawler based Phishing Attack Detector (WC-PAD) has been proposed to accurately detect instances of phishing. This uses the URL, traffic, and content of the web as input features. Now classification is complete taking these features into account.

There are several methods for identifying phishing websites, but the visual resemblance method is attracting the most attention. A database is used to keep the webpage snapshot that was taken. It determines whether the webpage screenshot entered matches the one that is saved in the database. If so, it is expected that the website is a phishing scam. However, if there are multiple websites that are similar, whatever website is submitted as input first is accepted as legitimate. As a result, it is unable to accurately forecast the real website, making it difficult to identify the desired website. This detection technique is suggested using target website finder and uses CSS and pictures.

### **3 PROPOSED WORK AND ALGORITHM**

The authors further claim that the system can detect emerging threats and can increase protection against zero-hour threats, in contrast to traditional blacklisting techniques which

operate reactively, to create a dynamic and extensible system for detection of present and emerging types of phishing domains. Real phishing assaults do exist, and there are ways to recognise them. However, the features may not always be present in such attacks, and a large percentage of false positives occur during detection. Using this element, one web page can be added to the primary webpage already there. The "iframe" tag allows phishers to make their content invisible, or without frame boundaries. Since the border of the inserted webpage is invisible, the user may enter sensitive information since they believe it to be a part of the main webpage

• **Algorithm: -**

auto-encoder proposed for phishing website classification. The features of phishing website become input to the first auto-encoder which extracted high level features and reduced dimension of features from 30 to 28. The output of first auto-encoder become input to the second auto-encoder which reduce the dimension of extracted features from first auto-encoder to the 27.

The purpose from two auto-encoder is to reduce the feature dimension gradually and learn most important and sensitive features. The output of the last auto-encoder become input to the SoftMax that classify the features in to two labels there is phishing or not.

• In the last stage, the two auto-encoders and SoftMax are stacked and trained by using supervised learning. Then, the system tested by using another data to check the performance of the proposed method. The accuracy is calculated after the testing is complete

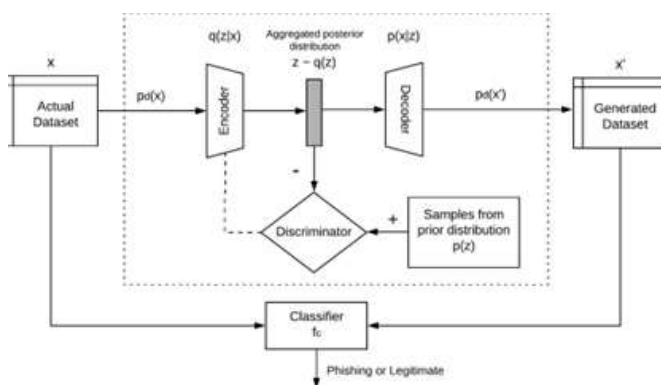
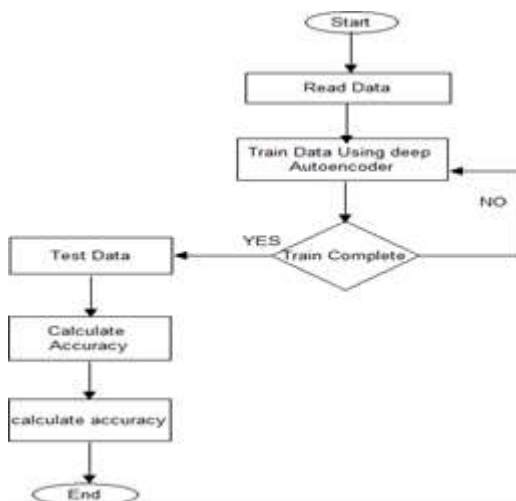


Fig. 1 Proposed Method Flowchart

**Fig 2:** - The design of the method we suggest. It is made up of a malicious autoencoder that creates synthesized data from phishing data. The typical autoencoder that creates the data from the latent code  $z$  is shown in the top row. The encoding and decoding distributions are denoted by  $q(z|x)$  and  $p(x|z)$ , respectively. Following the synthesis of the data, a machine learning classifier ( $fc$ ) as stated in Subsection III-C is utilized to identify whether the synthesized samples originate from fraudulent websites.



**Fig 2:** - The design of the method

• **Algorithm Implementation**

Input: Vector(Num),URL

Output: URL\_Type

1. Procedure Training Phase(Num)
2. While num=Num do
3. If num=Feature(URL)then
4. Op=Phishing URL
5. else
6. Op=Legitimate URI
7. If op=LSTM(ib(feature))then
8. Op=Pishing URL
9. else



10. op=Legitimate URL
11. end if
12. end if
13. end while
14. return op
15. end procedure

**Fig 3: step by step Training phase implementation Algorithm for phishing URL detection**

Input: URL

Output: Type of URL

1. Procedure Testing Phase(URL)
2. While url=URL do
3. element=LSTM Memory=Feature(URL)then
4. op= Phishing URL
5. else
6. op= Legitimate URL
7. feedback= phishtank(op)
8. if element =LSTM Memory=f=feedback then
9. op = Legitimate URL
10. else
11. op = Legitimate URL
12. end if
13. end if
14. end while
15. return op
16. end procedure

**Fig 4: step by step Testing phase implementation Algorithm for phishing URL detection**

### 3 METHODOLOGIES

This is the most prevalent kind of phishing assault, in which a cybercriminal pretends to be a well-known company, domain, or organization in an effort to get sensitive personal data from the victim, including login credentials, passwords, bank account information, credit card information, etc. For instance, when emails with malicious URLs are sent out in bulk to a broad user base, the cybercriminal would anticipate that many people will receive the emails, visit the malicious URLs, or download the infected attachments. Most of the time,



this type of communication makes the victims feel anxious and pressed to , once a user open such messages or visit the URLs the damage is done.

The fraud victim suffers financial loss, loss of personal information, and reputational harm. Therefore, it is crucial to quickly discover a solution that could reduce such security vulnerabilities. Traditionally, phishing websites are identified using employing a blacklist. Many well-known websites, including PhisTank, maintain lists of websites that are blacklisted. The eradication Blacklists might not

### **Train Test Split:**

Dividing the dataset into a training portion and a testing portion. Using the "train test split" method, the dataset was divided into training and testing datasets, with 75% for training and 25% for testing. After determining the dependent and independent variables, the splitting was carried out.

- **Preprocessing:**

Preprocessing entails adding missing data or eliminating it altogether to get a clean dataset. The chosen dataset, however, already had preprocessing done on it, so I didn't need to do any more. The only preprocessing procedure required was feature scaling.

- **Feature Scaling:**

This process, which normalizes the independent variable present in the data in a specified range, is known as feature scaling.

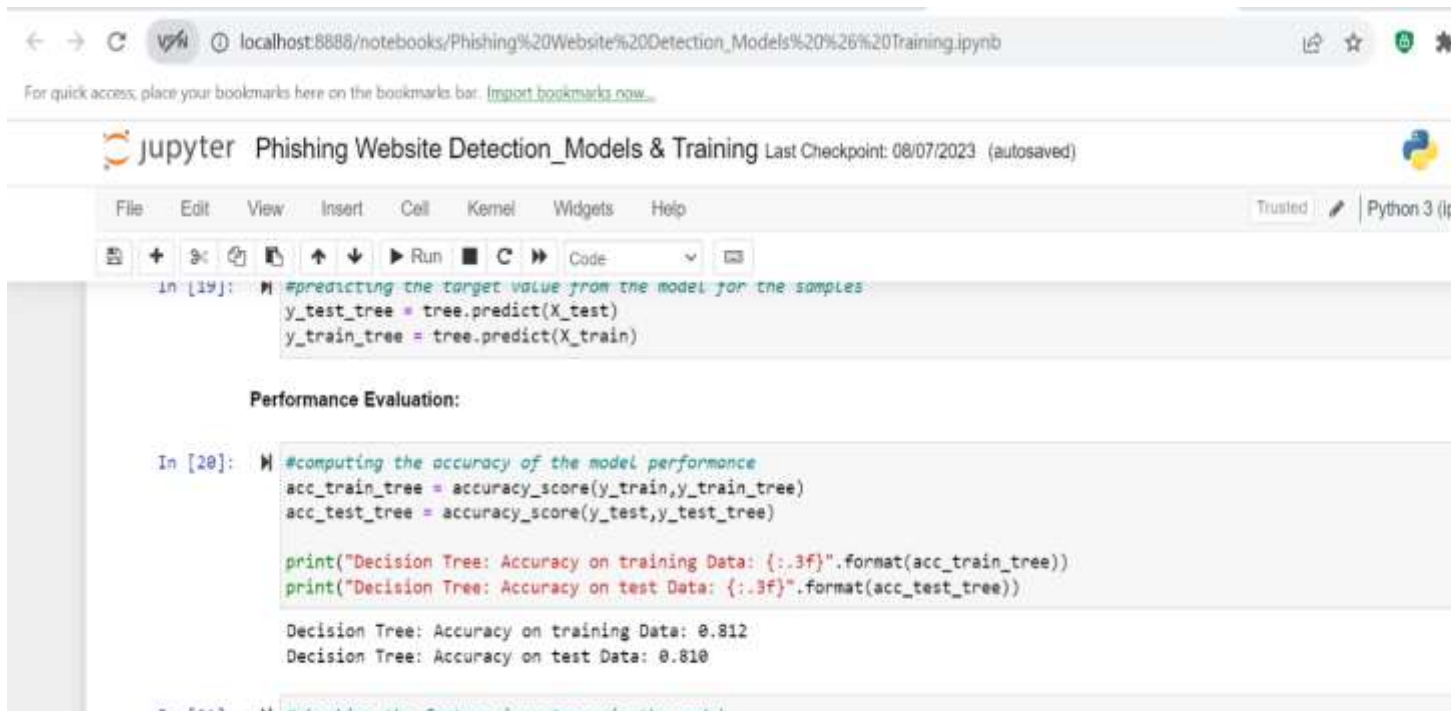
The variables should be scaled equally; otherwise, one variable may take precedence. so, the outcome could be impacted by others. Standardization is a different scaling technique in which the values are based on the mean and have a unit standard deviation. This means that the attribute's mean is equal to zero and that the distribution that results has a unit standard deviation. Normalization is a scaling technique where values are moved and rescaled until they fall somewhere between 0 and 1. It is also known as Min-Max scaling. It makes use of Standard Scalar.

- **Feature Extraction:**

In order to obtain information about an IP address, the length of a URL, a domain name, subdomains, the presence of a favicon, etc., feature values are extracted using python modules such as who is, requests, socket, re, ip address, Beautiful Soup, etc. The result is saved as a value in a list. The dataset is in this format; hence the classifier will be trained using input in this format,

## **5 Results and Evolution Metrics**





```
In [19]: #predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)

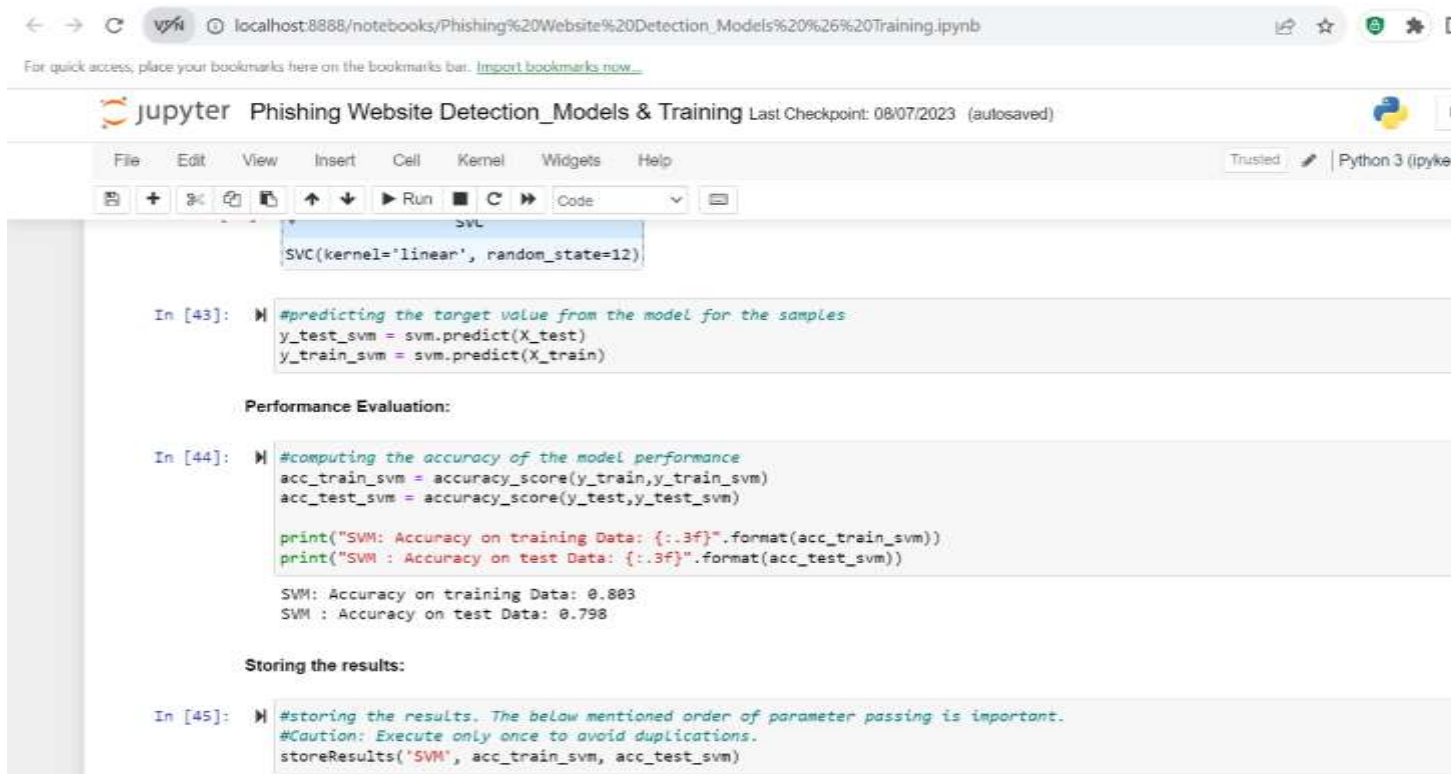
Performance Evaluation:

In [20]: #computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc_test_tree))

Decision Tree: Accuracy on training Data: 0.812
Decision Tree: Accuracy on test Data: 0.810
```

Fig 5: accuracy of Decision Tree achieved 81%



```
SVC(kernel='linear', random_state=12)

In [43]: #predicting the target value from the model for the samples
y_test_svm = svm.predict(X_test)
y_train_svm = svm.predict(X_train)

Performance Evaluation:

In [44]: #computing the accuracy of the model performance
acc_train_svm = accuracy_score(y_train,y_train_svm)
acc_test_svm = accuracy_score(y_test,y_test_svm)

print("SVM: Accuracy on training Data: {:.3f}".format(acc_train_svm))
print("SVM : Accuracy on test Data: {:.3f}".format(acc_test_svm))

SVM: Accuracy on training Data: 0.803
SVM : Accuracy on test Data: 0.798

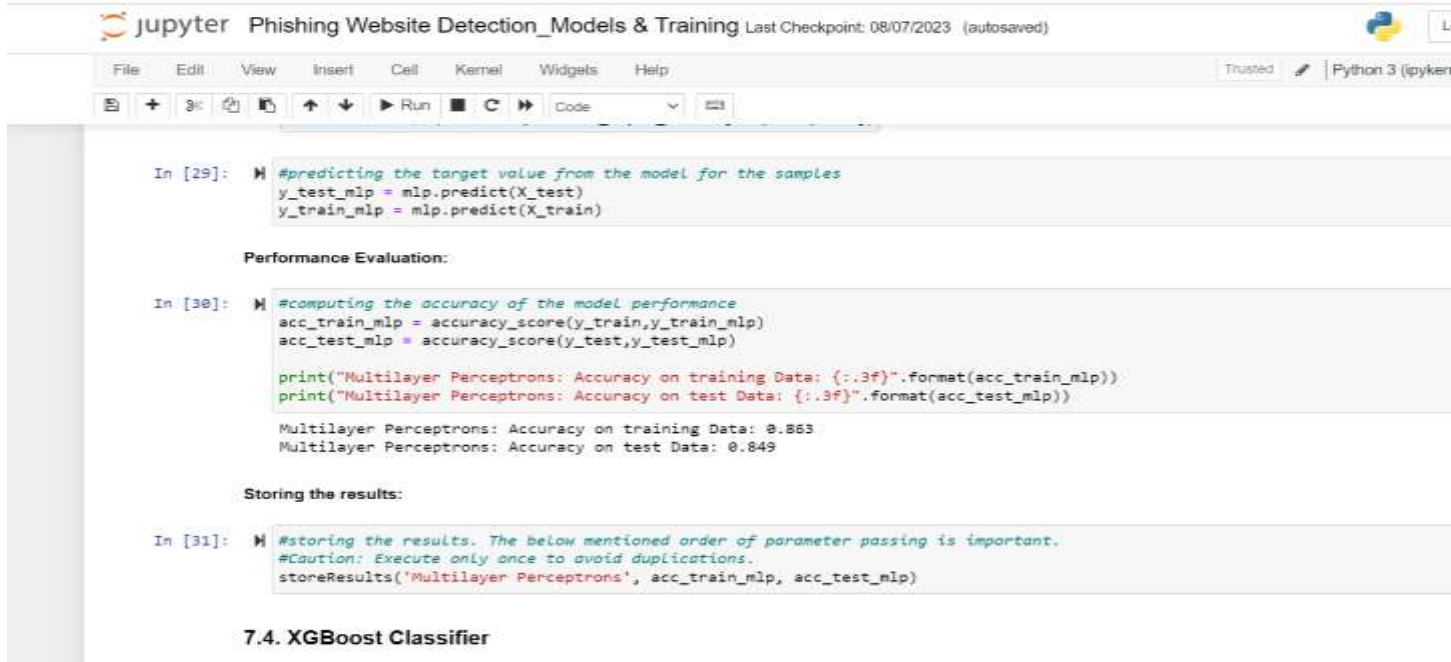
Storing the results:

In [45]: #storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('SVM', acc_train_svm, acc_test_svm)
```

Fig 6: accuracy SVM achieved 79.8%



For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [29]: #predicting the target value from the model for the samples
y_test_mlp = mlp.predict(X_test)
y_train_mlp = mlp.predict(X_train)

Performance Evaluation:

In [30]: #computing the accuracy of the model performance
acc_train_mlp = accuracy_score(y_train,y_train_mlp)
acc_test_mlp = accuracy_score(y_test,y_test_mlp)

print("Multilayer Perceptrons: Accuracy on training Data: {:.3f}".format(acc_train_mlp))
print("Multilayer Perceptrons: Accuracy on test Data: {:.3f}".format(acc_test_mlp))

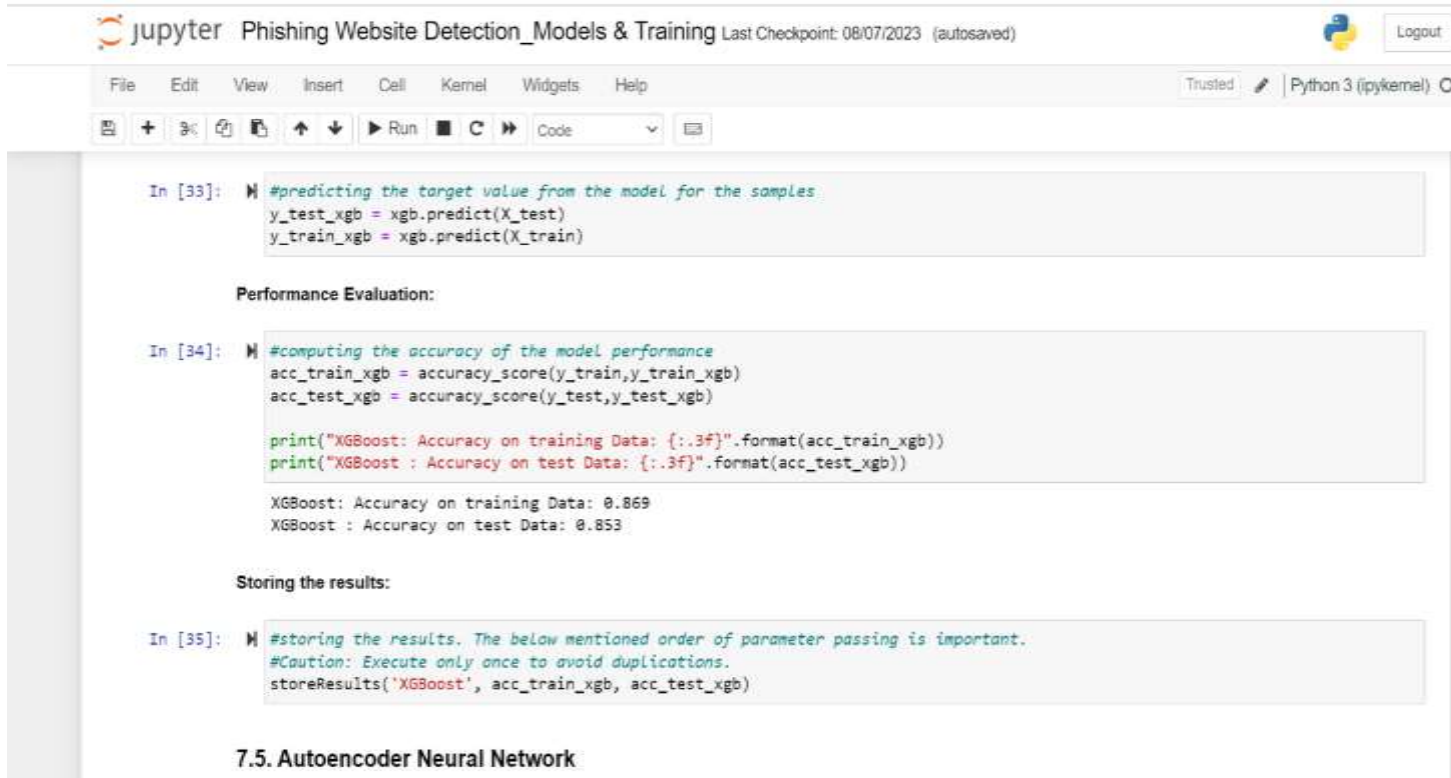
Multilayer Perceptrons: Accuracy on training Data: 0.863
Multilayer Perceptrons: Accuracy on test Data: 0.849

Storing the results:

In [31]: #storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('Multilayer Perceptrons', acc_train_mlp, acc_test_mlp)
```

**7.4. XGBoost Classifier**

**Fig 7: accuracy of MLP achieved 84.9 %**



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [33]: #predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)

Performance Evaluation:

In [34]: #computing the accuracy of the model performance
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

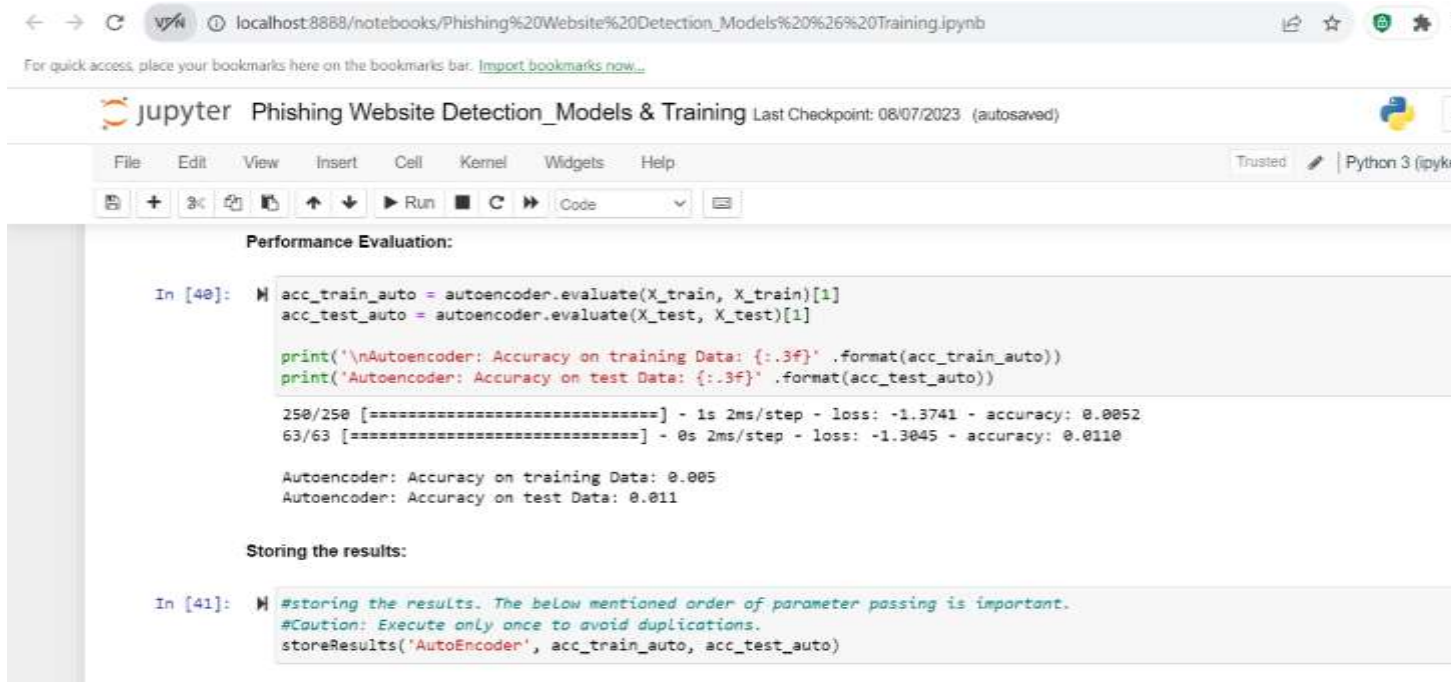
XGBoost: Accuracy on training Data: 0.869
XGBoost : Accuracy on test Data: 0.853

Storing the results:

In [35]: #storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('XGBoost', acc_train_xgb, acc_test_xgb)
```

**7.5. Autoencoder Neural Network**

**Fig 8: accuracy of XGboost achieved 85.3 %**



```
In [40]: M acc_train_auto = autoencoder.evaluate(X_train, X_train)[1]
acc_test_auto = autoencoder.evaluate(X_test, X_test)[1]

print('\nAutoencoder: Accuracy on training Data: {:.3f}' .format(acc_train_auto))
print('Autoencoder: Accuracy on test Data: {:.3f}' .format(acc_test_auto))

250/250 [=====] - 1s 2ms/step - loss: -1.3741 - accuracy: 0.0052
63/63 [=====] - 0s 2ms/step - loss: -1.3045 - accuracy: 0.0110

Autoencoder: Accuracy on training Data: 0.005
Autoencoder: Accuracy on test Data: 0.011

Storing the results:

In [41]: M #storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('AutoEncoder', acc_train_auto, acc_test_auto)
```

**Fig 9: accuracy of CNN achieved 91.9 %**

## 7 References

- Reid G. Smith and Joshua Eckroth. Building ai applications: Yesterday, today, and tomorrow. *AI Magazine*, 38(1):6–22, Mar. 2017.
- Panos Louridas and Christof Ebert. Machine learning. *IEEE Software*, 33:110–115, 09 2016.
- Michael Jordan and T.M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science (New York, N.Y.)*, 349:255–60, 07 2015.
- Steven Aftergood. Cybersecurity: The cold war online. *Nature*, 547:30+, Jul 2017. 7661.
- Aleksandar Milenkoski, Marco Vieira, Samuel Kounev, Alberto Avritzer, and Bryan Payne. Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys*, 48:12:1–, 09 2015.
- Chirag N. Modi and Kamatchi Acha. Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: a comprehensive review. *The Journal of Supercomputing*, 73(3):1192–1234, Mar 2017.
- Eduardo Viegas, Altair Santin, Andre Fanca, Ricardo Jasinski, Volnei Pedroni, and Luiz Soares de Oliveira. Towards an energy-efficient anomaly-based intrusion detection



engine for embedded systems.

8. Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang. Machine learning and deep learning methods for cybersecurity. *IEEE Access*, 6:35365–35381, 2018.
9. Neha R. Israni and Anil N. Jaiswal. A survey on various phishing and anti-phishing measures. *International journal of engineering research and technology*, 4, 2015.