

Configuration Manual

Academic Internship
MSc Cyber-security

Student ID: 22183981

School of Computing
National College of Ireland

Supervisor: Mr. Michael Pantridge

National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name: Nitin Vijla

Student ID: 22183981

Programme: MSc Cybersecurity

Year: 2023

Module: Academic Internship

Lecturer:

Submission Due Date: 12/08/2024

Project Title: “*Detecting Sql Injection and Xss Attack*”

Word Count: 740

Page Count:

11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Nitin

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies) ☐

Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies). ☐

You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. ☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:

Date:

Penalty Applied (if applicable):

Configuration Manual

1. Introduction

This research outlines the description of the dataset, hardware and software requirement, and deployment in stages using Jupyter Notebook regarding the identification of SQL Injection and XSS assaults.

2. Overview of the program

This project categorizes network access data in a bid to identify SQL Injection and XSS threats. There are various phases out there such as data pre-processing, feature extraction, modelling using CNN-RNN and Evaluation of output generated by the model.

3. Hardware/software requirements

3.1. *Hardware production*

The following hardware settings are recommended for smooth operation.

- Processor: Intel Core i5 or higher
- RAM: 4 GB or more

3.2. *Software*

The following software is required for the project:

- Jupyter Notebook: Used to run and write code.
- Anaconda: To manage the Python environment and dependencies.
- Google Colab: Used to run code and dependencies (Open Source Platform)

Version Requirements:

- Jupyter Notebook: Version 6.0 or later
- Anaconda: Version 2020.11 or later

Ensure that the necessary Python libraries are installed. These include pandas, numpy, matplotlib, seaborn, scikit-learn, tensorflow, torch and tqdm.

4. The data set

The dataset is selected from Kaggle. The content of this dataset mainly constitutes some of the labelled queries for training and testing sections using machine learning.

- Training data: Train.csv
- Testing data: Test.csv
- Validation data: Validation.csv

If using Jupyter Notebook then the datasets should be loaded in the project directory of Jupyter..

5. Implementation of Project in Jupyter

Explanation of the Code

5.1. Import Libraries

```
[1]: import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import pandas as pd
from sklearn.metrics import accuracy_score, classification_report
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tqdm import tqdm
import time
import random

WARNING:tensorflow:From C:\Users\User\AppData\Roaming\Python\Python311\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

- pandas: It is used for data analytics and data manipulation.
- numpy: This library is for numerical computations.
- matplotlib.pyplot: A visual library that supports creation of static, interactive or animated plots.
- torch: It supports the building and training of the neural networks with the help of the Py Torch library.
- seaborn: It helps in statistical data visualization.
- Tqdm: It presents various bar for training and evaluation of models.

5.2. Load and Inspect Data

```
[11]: import pandas as pd

# Loading a small portion of the data to inspect it
try:
    train_df = pd.read_csv('Train.csv', encoding='utf-8', nrows=100)
    test_df = pd.read_csv('Test.csv', encoding='utf-8', nrows=100)
    validation_df = pd.read_csv('Validation.csv', encoding='utf-8', nrows=100)
except pd.errors.ParserError as e:
    print(f"Error loading CSV file: {e}")

# Inspecting the data
print(train_df.head())
print(test_df.head())
print(validation_df.head())
```

	Query	Label
0	The film 'Nightbreed' is one of the best horro...	0
1	The story for Hare Rama Hare Krishna actually ...	0
2	1" where 6347 = 6347 union all select null,nul...	1
3	jk dv1z0r39amlwjiiumia9xrxdowuo87f 110dcc d0ej...	1
4	1' (select 'fdkl' where 4572 = 4572 uni...	1

	Query	Label
0	A pretentious but - to varying degrees - watch...	0
1	Tamar-Mattis said organization supports Austr...	0
2		5337
3	SELECT * FROM nothing WHERE quick NOT LIKE '[s...	0
4	SELECT TOP 50 PERCENT * FROM according SELECT ...	0

	Query	Label
0	Genius or utter madness? That depends on your ...	0
1	Most who go to this movie will have an idea wh...	0
2	SELECT safe (s) FROM stranger	0

- pd.read_csv: It reads the whole csv document and into data frames of pandas.
- head(): It displays the first few rows of the dataset. It can be modified by adding numeric parameters, which when added will display only that number of first few rows of the dataset.

5.3. Data Overview

```
[5]: train_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  ---
0   Query   100 non-null    object
1   Label   100 non-null    int64
dtypes: int64(1), object(1)
memory usage: 1.7+ KB

[6]: test_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  ---
0   Query   100 non-null    object
1   Label   100 non-null    int64
dtypes: int64(1), object(1)
memory usage: 1.7+ KB

[7]: validation_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  ---
0   Query   100 non-null    object
1   Label   100 non-null    int64
dtypes: int64(1), object(1)
memory usage: 1.7+ KB
```

- info(): This method displays the information of all the three datasets including the null and non null columns.

5.4. Data Visualization

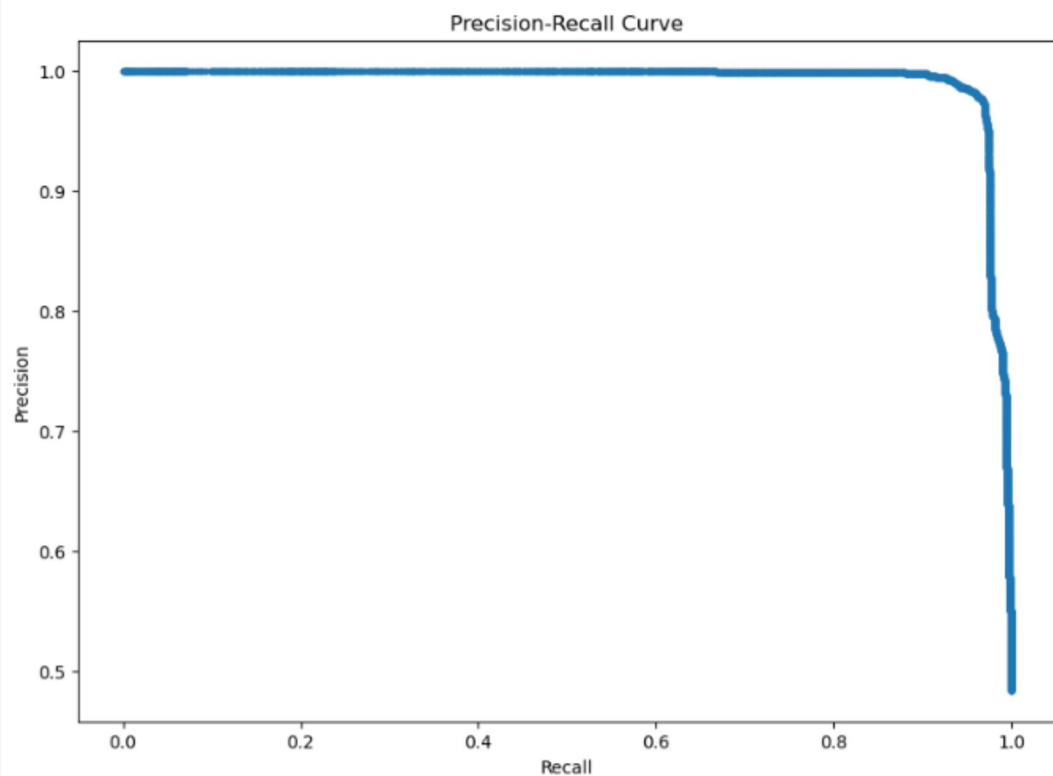
Precision-Recall and ROC Curves

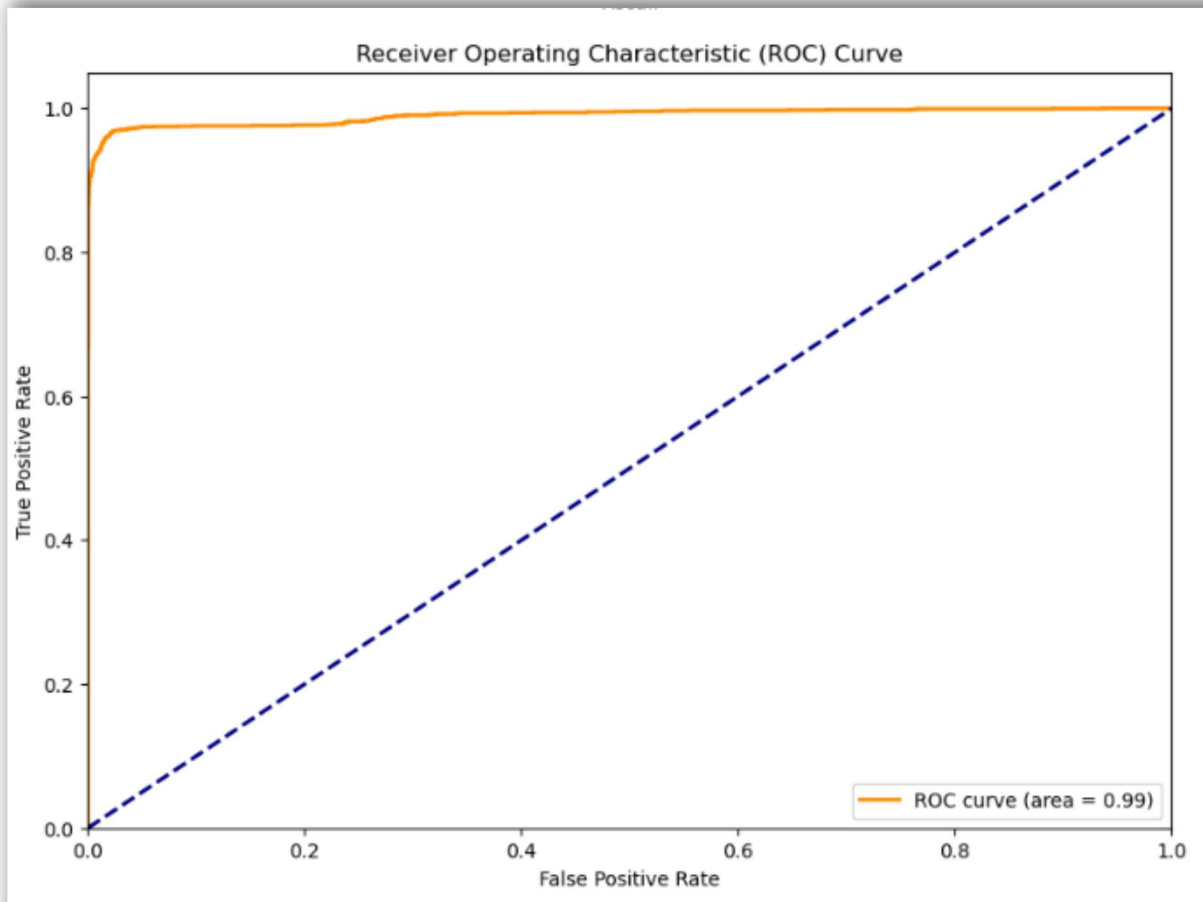
```
[22]: # Evaluating the model on the validation set
model.eval()
all_preds = []
all_labels = []
all_scores = []
with torch.no_grad():
    for X_batch, y_batch in tqdm(validation_loader, desc="Evaluating"):
        outputs = model(X_batch)
        _, predicted = torch.max(outputs, 1)
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(y_batch.cpu().numpy())
        all_scores.extend(torch.nn.functional.softmax(outputs, dim=1)[:, 1].cpu().numpy())

# Calculating accuracy
accuracy = accuracy_score(all_labels, all_preds)
print(f'Validation Accuracy: {accuracy}')
print(classification_report(all_labels, all_preds))

# Plotting precision-recall curve
plot_precision_recall_curve(all_labels, all_scores)

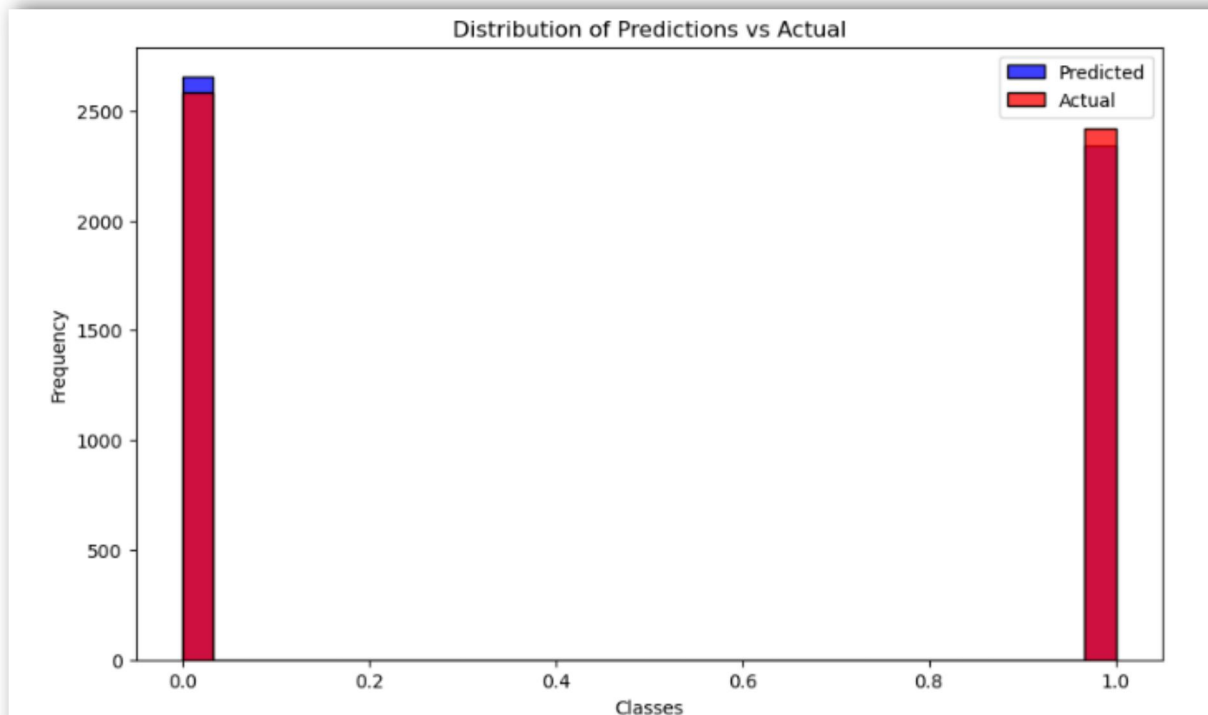
# Plotting ROC curve
plot_roc_curve(all_labels, all_scores)
```





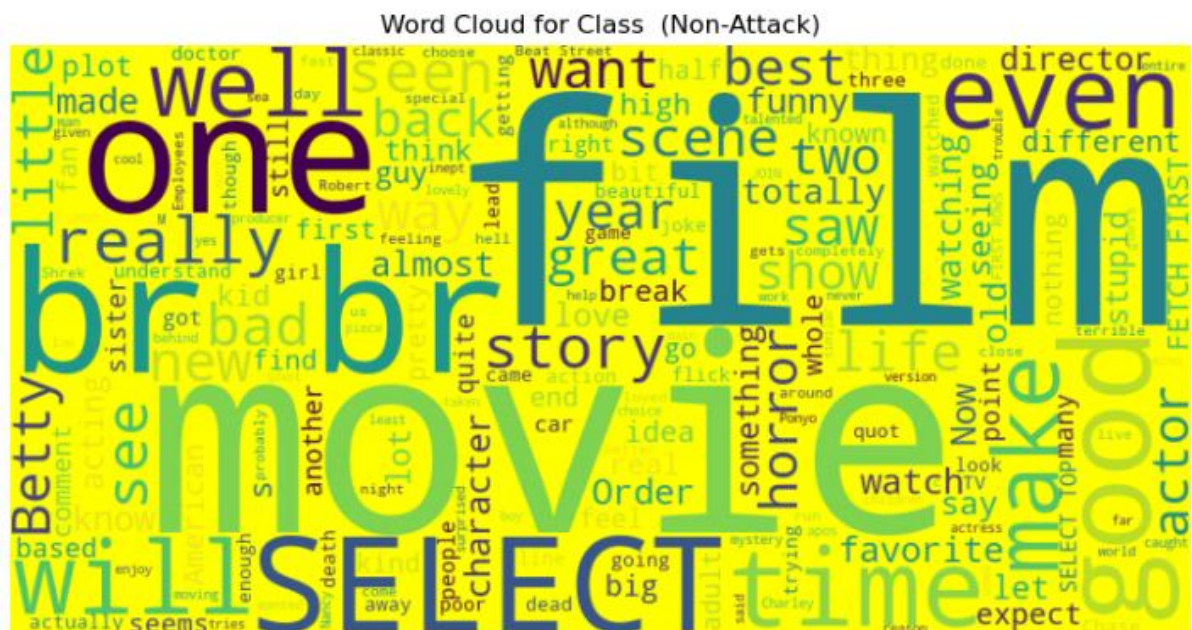
- Precision-Recall Curves: It helps to evaluate the performance of the CNN-RNN model at particular thresholds of precision and recall.
- Roc Curve: It helps to visualize the true positive rates versus false positive rates.

Distributions of Predictions vs Actual



- `sns.histplot()`: It helps to plot and visualize the model's performance comparatively by visualizing the actual and predicted classes of the CNN-RNN model.

Word Cloud



XSS Attack Queries



- Word Cloud: In Non attack class the word cloud visualizes the most common terms from the queries whereas the XSS attack classes displays the most common terms.

5.5. Pre-processing

Sub -Setting the data

```
def create_subset(file_path, subset_size, chunk_size=10000):
    subset_list = []
    for chunk in pd.read_csv(file_path, encoding='utf-8', chunksize=chunk_size):
        subset_list.append(chunk.sample(frac=subset_size / chunk_size, random_state=42))
        if len(subset_list) * chunk_size >= subset_size:
            break
    return pd.concat(subset_list).reset_index(drop=True)

# Defining subset sizes
train_subset_size = 10000
test_subset_size = 5000
validation_subset_size = 5000

# Creating subsets
try:
    train_subset = create_subset('Train.csv', train_subset_size)
    test_subset = create_subset('Test.csv', test_subset_size)
    validation_subset = create_subset('Validation.csv', validation_subset_size)
except pd.errors.ParserError as e:
    print(f"Error creating subsets: {e}")

# Verifying the subset sizes
print(f"Train subset size: {len(train_subset)}")
print(f"Test subset size: {len(test_subset)}")
print(f"Validation subset size: {len(validation_subset)}")
```

```
Train subset size: 10000
Test subset size: 5000
Validation subset size: 5000
```

- `create_subset`: This function creates a small part (subset of the data) so that it reduces the load of the Kernel thereby focussing on the model performance.

Tokenization and Padding

```
[13]: from tensorflow.keras.preprocessing.text import Tokenizer
      from tensorflow.keras.preprocessing.sequence import pad_sequences
      import torch

      # Combining subsets for tokenization
      all_queries = pd.concat([train_subset['Query'], test_subset['Query'], validation_subset['Query']], axis=0)

      # Tokenizer and padding setup
      tokenizer = Tokenizer()
      tokenizer.fit_on_texts(all_queries)
      max_length = max([len(seq) for seq in tokenizer.texts_to_sequences(all_queries)])

      def preprocess_data(df, tokenizer, max_length):
          sequences = tokenizer.texts_to_sequences(df['Query'])
          padded_sequences = pad_sequences(sequences, maxlen=max_length, padding='post')
          labels = df['Label'].fillna(0).astype(int).values
          return padded_sequences, labels

      X_train, y_train = preprocess_data(train_subset, tokenizer, max_length)
      X_test, y_test = preprocess_data(test_subset, tokenizer, max_length)
      X_validation, y_validation = preprocess_data(validation_subset, tokenizer, max_length)

[14]: from torch.utils.data import DataLoader, TensorDataset
```

- `tokenizer`: It helps to convert the data of the columns into its respective tokens or sequences.
- `Padding`: Padding the data ensures its length in a uniform manner.

6. Model Architecture

CNN-RNN Model

```

import torch.nn as nn

class CNNRNNModel(nn.Module):
    def __init__(self, vocab_size, embed_size, num_classes):
        super(CNNRNNModel, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_size)
        self.conv = nn.Conv1d(in_channels=embed_size, out_channels=128, kernel_size=5)
        self.pool = nn.MaxPool1d(kernel_size=2)
        self.lstm = nn.LSTM(input_size=128, hidden_size=128, num_layers=2, batch_first=True)
        self.fc = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.embedding(x).permute(0, 2, 1)
        x = self.pool(torch.relu(self.conv(x))).permute(0, 2, 1)
        x, _ = self.lstm(x)
        x = self.fc(x[:, -1, :])
        return x

# Instantiating model
vocab_size = len(tokenizer.word_index) + 1
embed_size = 100
num_classes = 2
model = CNNRNNModel(vocab_size, embed_size, num_classes).to(device)

```

- CNNRNN Model: It is a hybrid model that combines both the CNN and LSTM layers of the model for the sequential layers.

7. Model Evaluation and Training with accuracy

```

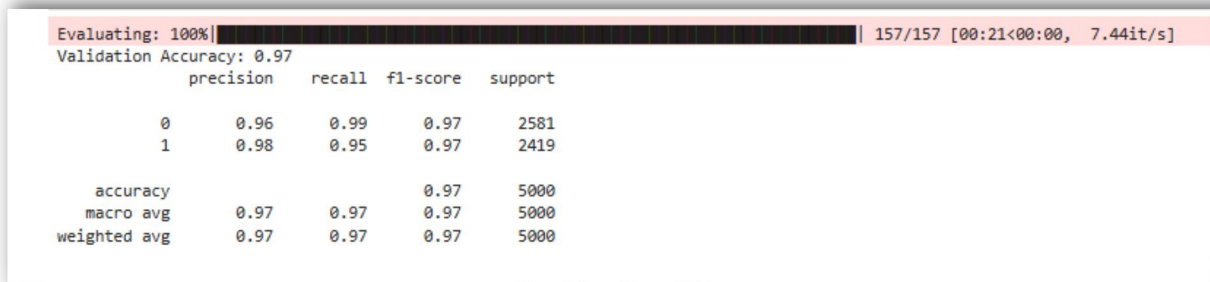
# Training Loop
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    for X_batch, y_batch in tqdm(train_loader, desc=f"Epoch {epoch + 1}/{num_epochs}"):
        optimizer.zero_grad()
        outputs = model(X_batch)
        loss = criterion(outputs, y_batch)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    avg_loss = total_loss / len(train_loader)
    print(f"Epoch {epoch + 1}, Loss: {avg_loss:.4f}")

```

Epoch 1/10: 100%	313/313	[04:43:00:00, 1.10it/s]
Epoch 1, Loss: 0.0561		
Epoch 2/10: 100%	313/313	[03:54:00:00, 1.34it/s]
Epoch 2, Loss: 0.0358		
Epoch 3/10: 100%	313/313	[03:50:00:00, 1.36it/s]
Epoch 3, Loss: 0.0261		
Epoch 4/10: 100%	313/313	[03:50:00:00, 1.36it/s]
Epoch 4, Loss: 0.0228		
Epoch 5/10: 100%	313/313	[04:14:00:00, 1.23it/s]
Epoch 5, Loss: 0.0195		
Epoch 6/10: 100%	313/313	[04:08:00:00, 1.26it/s]
Epoch 6, Loss: 0.0188		
Epoch 7/10: 100%	313/313	[03:51:00:00, 1.35it/s]
Epoch 7, Loss: 0.0207		
Epoch 8/10: 100%	313/313	[03:43:00:00, 1.40it/s]
Epoch 8, Loss: 0.0160		

- Adam: It is an optimizer that adjusts the weights of the model.

A terminal window showing the evaluation of a model. The top line indicates 'Evaluating: 100%' with a progress bar and '157/157 [00:21<00:00, 7.44it/s]'. Below this, the 'Validation Accuracy: 0.97' is displayed. A table follows, showing precision, recall, f1-score, and support for classes 0 and 1, as well as overall accuracy, macro avg, and weighted avg.

Evaluating: 100% 157/157 [00:21<00:00, 7.44it/s]				
Validation Accuracy: 0.97				
	precision	recall	f1-score	support
0	0.96	0.99	0.97	2581
1	0.98	0.95	0.97	2419
accuracy			0.97	5000
macro avg	0.97	0.97	0.97	5000
weighted avg	0.97	0.97	0.97	5000

- Classification report: It displays the 0.97 accuracy with the metrics like precision, recall and f1 score.

References:

[1] Zhang, W., Li, Y., Li, X., Shao, M., Mi, Y., Zhang, H., & Zhi, G. (2022). Deep Neural Network-Based SQL Injection Detection Method. *Security and Communication Networks*, 2022(1), 4836289.

<https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/4836289>

[2] Krishnan, S. A., Sabu, A. N., Sajan, P. P., & Sreedeeep, A. L. (2021). SQL injection detection using machine learning. *Revista Geintec-Gestao Inovacao E Tecnologias*, 11(3), 300-310.

<http://revistageintec.net/old/wp-content/uploads/2022/02/1939.pdf>

[3]Kaur, J., Garg, U., & Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. *Artificial Intelligence Review*, 56(11), 12725-12769.

https://www.researchgate.net/profile/Jasleen-Kaur-15/publication/369476572_Detection_of_cross-site_scripting_XSS_attacks_using_machine_learning_techniques_a_review/links/64a17c328de7ed28ba6c232d/Detection-of-cross-site-scripting-XSS-attacks-using-machine-learning-techniques-a-review.pdf?origin=journalDetail&_tp=eyJwYWdlIjoiam91cm5hbERldGFpbCJ9

[4]Habibi, G., & Surantha, N. (2020, August). XSS attack detection with machine learning and n-gram methods. In *2020 International conference on information management and technology (ICIMTech)* (pp. 516-520). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9210946/>

[5]Nair, S. S. (2024). Securing Against Advanced Cyber Threats: A Comprehensive Guide to Phishing, XSS, and SQL Injection Defense. *Journal of Computer Science and Technology Studies*, 6(1), 76-93.

<https://www.al-kindipublisher.com/index.php/jcsts/article/view/6537>

[6]Bhardwaj, A., Chandok, S. S., Bagnawar, A., Mishra, S., & Uplaonkar, D. (2022, September). Detection of cyber attacks: XSS, sqli, phishing attacks and detecting intrusion using machine learning algorithms. In *2022 IEEE Global Conference on Computing, Power and Communication Technologies (GlobConPT)* (pp. 1-6). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9938367/>

[7]Qbea'h, M., Alrabae, S., & Mouheb, D. (2020). An analytical scanning technique to detect and prevent the transformed SQL injection and XSS attacks. In 6th International Conference on Information Systems Security and Privacy, ICISSP 2020 (pp. 603-610). SciTePress.

<https://nchr.elsevierpure.com/en/publications/an-analytical-scanning-technique-to-detect-and-prevent-the-transf>

[8]Alsaffar, M., Aljaloud, S., Mohammed, B. A., Al-Mekhlafi, Z. G., Almurayziq, T. S., Alshammari, G., & Alshammari, A. (2022). Detection of Web Cross-Site Scripting (XSS) Attacks. *Electronics*, 11(14), 2212.

<https://www.mdpi.com/2079-9292/11/14/2212>

[9]Choudhary, R. R., Verma, S., & Meena, G. (2021, December). Detection of SQL injection attack using machine learning. In 2021 IEEE international conference on technology, research, and innovation for betterment of society (TRIBES) (pp. 1-6). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9751616/>

[10]Abikoye, O. C., Abubakar, A., Dokoro, A. H., Akande, O. N., & Kayode, A. A. (2020). A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. *EURASIP Journal on Information Security*, 2020, 1-14.

<https://link.springer.com/article/10.1186/s13635-020-00113-y>

[11]Stiawan, D., Bardadi, A., Afifah, N., Melinda, L., Heryanto, A., Septian, T. W., ... & Budiarto, R. (2023). An Improved LSTM-PCA Ensemble Classifier for SQL Injection and XSS Attack Detection. *Computer Systems Science & Engineering*, 46(2).

https://cdn.techscience.cn/files/csse/2023/TSP_CSSE-46-2/TSP_CSSE_34047/TSP_CSSE_34047.pdf

[12]Muslihi, M. T., & Alghazzawi, D. (2020, October). Detecting SQL injection on web application using deep learning techniques: a systematic literature review. In 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE) (pp. 1-6). IEEE.

https://cdn.techscience.cn/files/csse/2023/TSP_CSSE-46-2/TSP_CSSE_34047/TSP_CSSE_34047.pdf

[13]Muslihi, M. T., & Alghazzawi, D. (2020, October). Detecting SQL injection on web application using deep learning techniques: a systematic literature review. In 2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE) (pp. 1-6). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9243198/>

[14]Farea, A. A., Wang, C., Farea, E., & Alawi, A. B. (2021, December). Cross-site scripting (XSS) and SQL injection attacks multi-classification using bidirectional LSTM recurrent neural network. In 2021 IEEE International Conference on Progress in Informatics and Computing (PIC) (pp. 358-363). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9687064/>

[15]Schoenborn, J. M., & Althoff, K. D. (2021). Detecting SQL-Injection and Cross-Site Scripting Attacks Using Case-Based Reasoning and SEASALT. In LWDA (pp. 66-77).

<https://ceur-ws.org/Vol-2993/paper-07.pdf>

[16]Cui, Y., Cui, J., & Hu, J. (2020, February). A survey on xss attack detection and prevention in web applications. In Proceedings of the 2020 12th International Conference on Machine Learning and Computing (pp. 443-449).

<https://dl.acm.org/doi/abs/10.1145/3383972.3384027>

[17]Nasereddin, M., ALKhamaiseh, A., Qasaimeh, M., & Al-Qassas, R. (2023). A systematic review of detection and prevention techniques of SQL injection attacks. *Information Security Journal: A Global Perspective*, 32(4), 252-265.

<https://www.tandfonline.com/doi/abs/10.1080/19393555.2021.1995537>

[18]Tadhani, J. R., Vekariya, V., Sorathiya, V., Alshathri, S., & El-Shafai, W. (2024). Securing web applications against XSS and SQLi attacks using a novel deep learning approach. *Scientific Reports*, 14(1), 1803.

<https://www.nature.com/articles/s41598-023-48845-4>

[19]Rahul, S., Vajrала, C., & Thangaraju, B. (2021, November). A novel method of honeypot inclusive WAF to protect from SQL injection and XSS. In 2021 International Conference on

Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON) (Vol. 1, pp. 135-140). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9688059/>

[20]Chen, D., Yan, Q., Wu, C., & Zhao, J. (2021). Sql injection attack detection and prevention techniques using deep learning. In Journal of Physics: Conference Series (Vol. 1757, No. 1, p. 012055). IOP Publishing.

<https://iopscience.iop.org/article/10.1088/1742-6596/1757/1/012055/meta>

[21]Arock, M. (2021, February). Efficient detection of SQL injection attack (SQLIA) Using pattern-based neural network model. In 2021 International conference on computing, communication, and intelligent systems (ICCCIS) (pp. 343-347). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9397066/>

[22]Sivasangari, A., Jyotsna, J., & Pravalika, K. (2021, June). SQL injection attack detection using machine learning algorithm. In 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1166-1169). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9452914/>

[23]Tanakas, P., Ilias, A., & Polemi, N. (2021, December). A novel system for detecting and preventing SQL injection and cross-site-script. In 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET) (pp. 1-6). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9698688/>

[24]Yadav, A. K., & Kumar, A. (2022). String matching algorithm based filter for preventing SQL injection and XSS attacks. In Inventive Computation and Information Technologies: Proceedings of ICICIT 2021 (pp. 793-807). Singapore: Springer Nature Singapore.

https://link.springer.com/chapter/10.1007/978-981-16-6723-7_59

[25]Marashdeh, Z., Suwais, K., & Alia, M. (2021, July). A survey on sql injection attack: Detection and challenges. In 2021 International Conference on Information Technology (ICIT) (pp. 957-962). IEEE.

<https://ieeexplore.ieee.org/abstract/document/9491117/>