National College of Ireland

Configuration Manual

MSc Research Project
MSc.in. Cyber Security

Charan Deep Chinthalapalli
Student ID: x23231866

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Charan Deep Chinthalapalli
Student ID:	X23231866
Programme:	MSc in Cyber Security **Year:** 2024-2025
Module:	MSc Research Project
Lecturer:	Michael Pantridge
Submission Due Date:	12-12-2024
Project Title:	Enhancing Phishing URL Detection by Leveraging Machine Learning and Deep Learning Models
Word Count:	350 **Page Count:** 5

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: Charan deep Chinthalapalli

Date: 11-12-2024

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

Charan Deep Chinthalapalli
Student ID: x23231866

1 Overview of code

The code is designed to detect phishing URLs using various machine learning and deep learning techniques. Key steps in the script include:

1. **Dataset Handling**:
    - Downloading the dataset from Kaggle.
    - Cleaning and preprocessing the data.
    - Extracting features from URLs (e.g., presence of IP, length, redirection, subdomain usage).
2. **Feature Engineering**:
    - Creation of custom features such as url_length, prefix_suffix, digits_present, etc.
    - Scaling features using MinMaxScaler.
3. **Machine Learning Models**:

    Implemented models include:
    - Decision Tree
    - Random Forest
    - Gradient Boosting
    - AdaBoost
    - Logistic Regression
    - XGBoost
4. **Deep Learning Models**:
    - Fully Connected Neural Networks (FNN).
    - Long Short-Term Memory Networks (LSTM).
    - TabNet for tabular data classification.
5. **Evaluation**:
    - Metrics: Accuracy, confusion matrix, classification report.

- Validation and test set evaluation.

2 Libraries

Python Version

- Python 3.8 or higher

Install the required libraries using the commands below:

1. pip install pandas numpy matplotlib seaborn scikit-learn
2. pip install tensorflow keras
3. pip install xgboost pytorch-tabnet

3 System Requirements

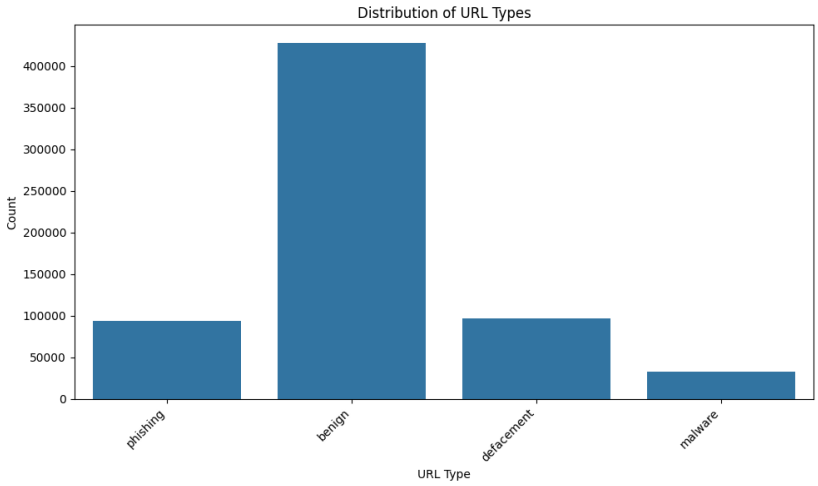
1) **Hardware:**
   - Minimum 8 GB RAM (16 GB recommended),
   - GPU-enabled for deep learning(optional).
2) **Software:**
   - Python 3.8+,
   - Jupyter Notebook, Google Colab.

4 Section 3

1) First few rows of dataset

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement

2) Distribution of URLs in dataset



3) Feature engineering by creating new columns based on URL data

```
df['use_of_ip'] = df['url'].apply(lambda i: has_ip(i))
df['use_of_at'] = df['url'].apply(lambda i: has_at(i))
df['url_length'] = df['url'].apply(lambda i: url_length(i))
df['url_redirection'] = df['url'].apply(lambda i: has_redirect(i))
df['prefix_suffix'] = df['url'].apply(lambda i: has_prefix_suffix(i))
df['https_token'] = df['url'].apply(lambda i: has_https(i))
df['digits_present'] = df['url'].apply(lambda i: has_digits(i))
df['subdomain_present'] = df['url'].apply(lambda i: has_subdomain(i))
df['shortening_service'] = df['url'].apply(lambda i: has_shortening_service(i))

print(df.head())
```

4) Test train and validation split

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (416761, 15)
y_train shape: (416761,)
X_test shape: (130239, 15)
y_test shape: (130239,)
```

5) Custom Deep learning model

```
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),  # First layer with 128 neurons
    Dense(64, activation='relu'),  # Second hidden layer with 64 neurons
    Dense(32, activation='relu'),  # Third hidden layer with 32 neurons
    Dense(16, activation='relu'),  # Fourth hidden layer with 16 neurons
    Dense(8, activation='relu'),   # Fifth hidden layer with 8 neurons
    Dense(2, activation='softmax')  # Output layer for binary classification; modify neurons for multiclass
])
```

6) Training and validation loss of custom deep learning model

7) Tabnet model initialisation

```
#  TabNet Model
tabnet_model = TabNetClassifier(
    n_d=8,  # Number of decision steps
    n_a=8,  # Attention dimensions
    seed=42,  # Seed for reproducibility
    optimizer_fn=torch.optim.Adam,  # Optimizer
    optimizer_params=dict(lr=2e-2),  # Learning rate
)
```

8) Hyperparamters used for various models

1. Decision Tree Classifier

- max_depth: [2, 4, 6, 8, 10, None]

2. Random Forest Classifier

- n_estimators: [50, 100, 150, 200, 250]

3. Gradient Boosting Classifier

- n_estimators: [50, 100, 150, 200, 250]

4. AdaBoost Classifier

- n_estimators: [50, 100, 150, 200, 250]

5. Logistic Regression

- C: [0.01, 0.1, 1, 10, 100]

6. XGBoost Classifier

- n_estimators: [50, 100, 150, 200, 250]

7. Fully Connected Neural Network (FNN)

- layers: [128, 64, 32, 16, 8]
- epochs: 20
- batch_size: 32

8. LSTM (Long Short-Term Memory)

- units: 64
- epochs: 50
- batch_size: 32

9. TabNet Classifier

- n_d: 8
- n_a: 8
- virtual_batch_size: 128
- learning_rate: 0.02

References

Arik, S.Ö. and Pfister, T., 2021, May. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, No. 8, pp. 6679-6687).

Breiman, Leo. "Random forests." *Machine learning* 45 (2001): 5-32.

Hochreiter, S., 1997. Long Short-term Memory. *Neural Computation MIT-Press*.