National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

Akash Muralidharan
Student ID: 23233192

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Akash Muralidharan		
Student ID:	23233192		
Programme:	MSc Cyber Security	**Year:**	2024-2025
Module:	MSc Practicum/Internship part 2		
Lecturer:	Vikas Sahni		
Submission Due Date:	29th January 2025		
Project Title:	Detection of Phishing Websites Using Machine Learning		
Word Count:	922 **Page Count:** 8		

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: Akash Muralidharan

Date: 29th January 2025

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

Akash Muralidharan
Student ID: 23233192

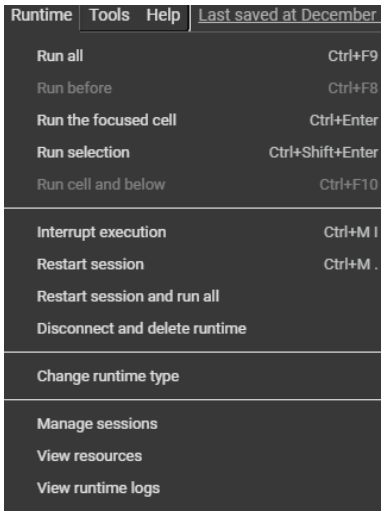
1 Introduction

The threats of phishing have grown to become a very dangerous issue in the world of cybersecurity, with users unaware of them and system security being vulnerable, that are being exploited. The configuration and implementation of the thesis “Detection of Phishing Websites Using Machine Learning” is detailed in this manual. Using the principles of AI, the system defines phishing websites from their features and patterns in such a way that it will help increase online security. The format of this manual is to ensure that all of the project is thoroughly understood. The hardware, software and dataset requirements to setup environment for the project are described in the Experiment Setup section. The next section lists the programming tools, libraries and platforms needed for implementation. The Implementation section gives stepwise instructions on data preparation and training models. Lastly, the references document contains resources and material utilized during the project. Users will be able to easily understand how to set up and perform this phishing detection system by following the sections of this manual.

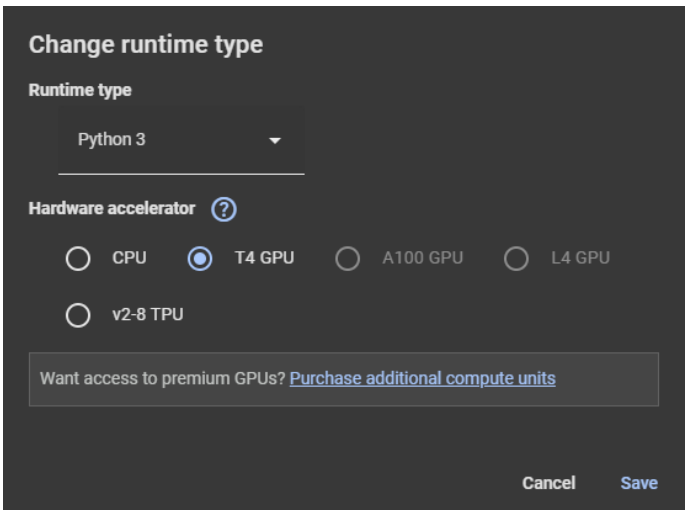
2 Experimental Setup

This project was implemented on Kaggle using a T4 GPU because the RAPIDS cuML library leverages machine learning capabilities on a GPU. Such a Kaggle runtime on GPU helps in optimal performance during model training.

In order to enable T4 GPU in Kaggle, go to Runtime -> Change Runtime Type-> select ‘T4 GPU’ and hit save. (Fig.1 and Fig.2)

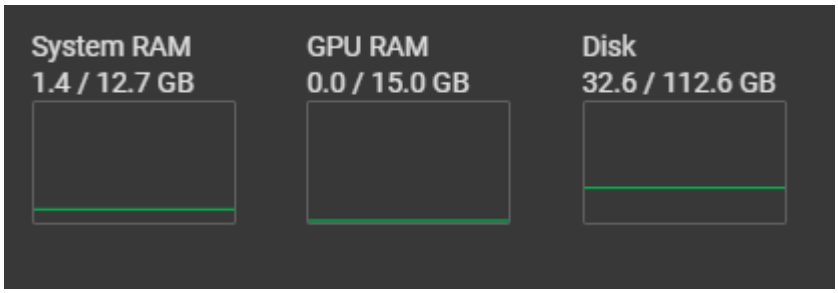


(Fig.1)



(Fig.2)

The Kaggle runtime specification is as follows (Fig.3):



(Fig.3)

Also, there is a need for installing RAPIDS into the environment so as to use the cuml library which uses the GPU for accelerated model training (Data, 2024). It is done by just cloning the repository provided by RAPIDS directly into the working folder (Fig.4).

```
!git clone https://github.com/rapidsai/rapidsai-csp-utils.git
!python rapidsai-csp-utils/colab/pip-install.py
```

(Fig.4)

Running this code clones the required files and libraries into the working directory so that these libraries can be used in during the training process. The following code snippet was taken from Collab Notebook provided by RAPIDS.

3 Technologies and Software used for Implementation

3.1 Software Used

- **Kaggle:** This project was conducted within Kaggle's computational environment which offered access to T4 GPU for GPU accelerated model training.
- **Jupyter Notebook:** Also used in Kaggle for interactive development and visualization of model results.
- **Python (3.11.9):** Python was used as the programming language used for this project (Python Software Foundation, 2019).

3.2 Libraries Used

- **pandas**: For data manipulation and analysis.
- **numpy**: For numerical computations.
- **scikit-learn**: For machine learning model implementation.
- **matplotlib and seaborn**: For plotting and visualizing data.
- **cuml**: For GPU-accelerated machine learning algorithms provided by RAPIDS (NVIDIA).

4 Implementation

4.1 Installing Libraries

- Import the required python libraries using the following command. (Fig.5) (Fig.6)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

(Fig.5)

```
from sklearn.linear_model import LogisticRegression
from cuml.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

(Fig.6)

- For the workloads that uses GPU, ensure cuml library is installed as described in the Setup section.

4.2 Loading the Dataset

The data is loaded using the 'read_csv' method from pandas into a dataset (Fig.7).

```
[ ]  # Reading the dataset
     data = pd.read_csv("Dataset/data.csv")
```

(Fig.7)

4.3 Data Preprocessing

The float64 and int64 values are converted into their 32-bit counterparts to improve training efficiency and speed (Fig.8).

```
# Converting data types for better performance
float_columns = data.select_dtypes('float64').columns
for c in float_columns:
    data[c] = data[c].astype('float32')

int_cols = data.select_dtypes('int64').columns
for c in int_cols:
    data[c] = data[c].astype('int32')

data.info()
```

(Fig.8)

Renaming the 'CLASS_LABEL' column to 'labels' for easy understanding (Fig.9).

```
# Renaming the 'CLASS_LABEL' column to 'labels' for better understanding
data.rename(columns={'CLASS_LABEL': 'labels'}, inplace=True)
```

(Fig.9)

4.4 Feature Selection

Mutual info classifier is used to find linear correlation between features and labels. The data is being split into features and labels (Fig.10).

```
# Splitting the data into features and labels
X = data.drop(['id', 'labels'], axis=1)
y = data['labels']
```

(Fig.10)

4.5 Model Training

- Logistic Regression (Fig.11):

```
def train_logistic_regression(data, top_n):
    top_n_features = mutualinfo_scores.sort_values(ascending=False).head(top_n).index.tolist()
    X = data[top_n_features]
    y = data['labels']

    # 20% for testing and 80% for training
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)

    logisticregression = LogisticRegression(max_iter=10000)
    logisticregression.fit(X_train, y_train)

    y_pred = logisticregression.predict(X_test)

    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    return precision, recall, f1, accuracy
```

(Fig.11)

- Random Forest Classifier on GPU (Fig.12):

```python
def train_random_forest_classifier(data, top_n):
    top_n_features = mutualinfo_scores.sort_values(ascending=False).head(top_n).index.tolist()
    X = data[top_n_features]
    y = data['labels']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)

    randomforestclassifier = RandomForestClassifier(n_estimators=500,
                split_criterion=1,
                max_depth=32,
                max_leaves=-1,
                max_features=1.0,
                n_bins=128)

    randomforestclassifier.fit(X_train, y_train)

    y_prediction = randomforestclassifier.predict(X_test, predict_model='GPU')

    precision = precision_score(y_test, y_prediction)
    recall = recall_score(y_test, y_prediction)
    f1 = f1_score(y_test, y_prediction)
    accuracy = accuracy_score(y_test, y_prediction)

    return precision, recall, f1, accuracy
```

(Fig.12)

4.6 Final Random Forest Classifier

Training the final Random Forest Model based on the optimal number of features (Fig.13).

```python
top_n_features = mutualinfo_scores.sort_values(ascending=False).head(50).index.tolist()
X = data[top_n_features]
y = data['labels']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)

randomforestclassifier = RandomForestClassifier(n_estimators=500,
            split_criterion=1,
            max_depth=32,
            max_leaves=-1,
            max_features=1.0,
            n_bins=128)

randomforestclassifier.fit(X_train, y_train)

y_prediction = randomforestclassifier.predict(X_test, predict_model='GPU')

precision = precision_score(y_test, y_prediction)
recall = recall_score(y_test, y_prediction)
f1 = f1_score(y_test, y_prediction)
accuracy = accuracy_score(y_test, y_prediction)

print("Performance for RFC Model with Top {} features is precision : {}, recall : {}, f1 score : {}, accuracy : {}".format(50, precision, recall
```

(Fig.13)

5 Reference

Data, A. (2024). *RAPIDS | GPU Accelerated Data Science*. [online] RAPIDS | GPU Accelerated Data Science. Available at: https://rapids.ai [Accessed 11 Dec. 2024].

Matplotlib (2012). *Matplotlib: Python plotting — Matplotlib 3.1.1 documentation*. [online] Matplotlib.org. Available at: https://matplotlib.org.

Python Software Foundation (2019). *Welcome to Python.org*. [online] Python.org. Available at: https://www.python.org.

scikit-learn.org. (n.d.). *scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation*. [online] Available at: https://scikit-learn.org.